Electrical Engineering Department
New Mexico Tech

Presents the

Theses

From the Course
Senior Design Project
(EE-481 & EE 482)

for 2001-2002

Instructor:

Scott W. Teare
Assistant Professor of Electrical Engineering

Preface

The Senior Design Project course in the Electrical Engineering Department at New Mexico Tech is a two semester course that provides students with the opportunity to develop a technical project in Electrical Engineering or closely aligned discipline while building their project management skills. This course introduces the students to working in teams, developing a project's statement of work, reporting milestones, presentations, design reviews, status reports and of course bringing in a project on time and under budget.

In this year's class, the students were presented with 20 projects from industry, Department of Defense and faculty. The students rated each project by their interests and then were assigned to the projects, nominally in groups of 3, based on their choices combined with a lottery to break ties. As a result, 9 projects were undertaken.

The final tasks in this course are for the students to present to the faculty, their sponsors and their peers a talk and a thesis on the work the accomplished over the past 8 months. This document is a collection of the thesis prepared by the students describing their projects.

The page numbers in this document are set for each thesis. As such the easiest way to get to a particular thesis is through the bookmarks of this PDF file.

I would like to express my sincere gratitude to the faculty and staff of the Electrical Engineering Department for their support of the Senior Design program, in particular, Robert Bond, Stephen Bruder, William Rison, Hassan Shanechi, Ronald Thomas & Kevin Wedeward for their participation as faculty sponsors on the various projects and Carrol Teel for her patience in dealing with all of us. I would also like to thank Dr. Deidre Hirschfeld of the Materials Engineering Department for her support and help on a multidisciplinary project between our departments.

Scott W. Teare
New Mexico, May 2002

Table of Contents

Design of Self-Watering Planter with Emphasis on Power Scavenging

A Thesis

Presented to the Faculty of

the Electrical Engineering Department

of

New Mexico Tech

By

Michael Carpenter
Rocky Ginnani
Randy Sena

In partial fulfillment of the requirements

for the course

EE-482 Senior Design Project II

May, 2002

ABSTRACT

A lot of people in this day and age go on sabbaticals or vacations and they have a tendency to neglect the watering of their household plants. Without human assistance, plants will not gain the necessary watering for survival. Obviously plants cannot water themselves, so we have developed a revolutionary planter that will automatically water any plant species based on its type (water intake level) utilizing a microcontroller, moisture sensor, pump, battery charging through solar cells, and other vital components. This whole project utilizes the idea of power scavenging. This is a basic system, which includes the main sub-systems of recyclable watering, solar rechargeable/battery power, and user-selectable program.

The moisture sensor supplies the analog measurement of moisture within the soil, sending the value to the microcontroller. This will allow the watering system to supply water for the plant based on one of the chosen sub-programs. The chosen sub-program (depending on the plant's water intake level) will decide the amount of water to release, as it activates the pump for a predetermined length of time. The water will be released directly into the soil since the tube is submerged in it and wrapped around the plant's roots. For some plants, too much water can be devastating; therefore, if there has been too much water induced into the plant, the water will fall through the soil enough to be recycled back into the holding tank for future use. Once the water drops below the predetermined level, more water will be added.

The power will be delivered from a battery and recharged from solar cells that will be attached to the planter. The choice of battery and solar panel was based on conceived power consumption (as low as possible).

The user-selectable program is sort of the "brains" of the project. The HC12 microcontroller will have several sub-programs that are based on the user's plant choice. A switch will also determine the program used as well. Until another plant selection has been made, the microcontroller will continue executing the chosen program to water the plant.

All selections (battery, charger, solar panel, water pump, programmable logic device, etc.) have been made in the fall semester and the actual assembly began this semester – which includes moisture sensor development. The programming of the logic device began as well. We found that the majority of our tasks went according to the gant chart and had very little errors.

Acknowledgments

First, we would like to thank our customer, Dr. Greg Donohoe, for offering us the opportunity to endeavor on an exciting project. His assistance and advice on energy scavenging as well as moisture sensor development were greatly appreciated. In addition to Dr. Bill Rison and our advisor, Dr. Stephen Bruder, as they were always willing to offer us helpful advice, mainly pertaining to the HC12 microcontroller. Dr. Ron Thomas was instrumental in helping with the actual circuitry and analog design (mainly pertaining to the battery charger), and we thank him gratefully for that.

We would also like to thank the engineer at Honeywell who informed us that we were investigating the wrong type of sensors for detecting moisture level, as well as Environmental Engineering Professor Dr. Mark Cal, for presenting us helpful advice pertaining to a moisture sensor.

Without the donated parts and assistance of the Instrument Room Manager, Mr. Norton Euart, we would not have been able to actually construct our planter.

Finally, we would like thank our instructor Dr. Scott Teare, for his witty and beneficial comments regarding to the Senior Design course itself. We feel he has prepared us tremendously on what to expect once we graduate from college. Thank you.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Project Description

On September 10, 2001, Dr. Greg Donohoe presented a project for us to undertake. His idea was for us to design a self-contained system that would automatically water a plant when the plant needed it – all without human intervention. We decided to accept the challenge. He also wanted us to scavenge energy from an external source to maintain its power storage requirements.

1.2 Definitions

There are several definitions to this project. First, we needed to determine if it is feasible to design a self-contained system that will water a plant when necessary. The type of plant to be used will be pre-programmed into the HC12 microcontroller. All the user has to do is input his/her plant species. The HC12 will control the water schedule of that plant from that point on or until the input has changed. Second, we needed to realize what type of power source to use. We used solar cells and to store the power, we used a lead acid battery. Other ideas thrown around include harmonic vibrations, wind, chemical, mechanical, or nuclear. Third, we needed to realize the type of moisture sensor to employ. We used a basic saturated transistor with a current source. This sensor needs to utilize as little power as possible to accommodate the power distribution.

1.3 Literature Review

The Art of Electronics book by Horowitz and Hill covers a wide range of subjects in electronics. This includes transistors, FETs (field effect transistors), OP-AMPS

(operational amplifiers), active filters and oscillators, voltage regulators, low-noise circuits, digital electronics, microprocessors, high frequency techniques, low-power designs, and signal processing. It helped us develop the majority of our components for the self-watering planter. The plants.usda.gov website provided vital information in plant water intake level among other characteristics. Siliconsolar.com website provided answers for solar cell needs among other vital parts. Human assistance included Dr. Greg Donohoe, who was our customer and offered assistance in the moisture sensor development; Dr. Bill Rison, who greatly assisted us in the HC12 microcontroller programming; and Dr. Ron Thomas, who greatly assisted us in solving most of the circuitry problems.

1.4 History

The history behind the self-watering planter revolves mainly around the inconvenience of constantly taking good care of the plant(s). Our customer, Dr. Greg Donohoe, is constantly away from his office on trips; hence, the plants die easily under improper care. Donohoe thought of the self-watering planter as a solution to proper plant care while away from the plants for weeks at a time.

Chapter 2
Background Information

2.1 Technical Background

2.1.1 Moisture Sensor

In the Fall Semester, we researched common commercial sensors but realized the majority of them are not applicable to our situation. More specifically, most of them are either too big, too expensive, or cannot be submerged into the dirt. We went on to use a saturated transistor with a current source. That is the sensor itself as it is connected to a copper probe. The saturated transistor allows more current (larger) to flow into the current source, as it supplies little current (since we are allowing little power as possible). The current is what gives it its power. The copper probe is like the arm of the sensing device as it senses for ion flow in the dirt (i.e. water).

2.1.2 Light Sensor/Real Time Clock

The light sensor idea began at the beginning of the spring semester. After choosing the components during the fall semester, we decided that we needed a way for the planter to wake up. Since the light sensor is an IR phototransistor, this requires an output circuit that takes the analog voltage from the emitter of the phototransistor and then sends that to the positive input of the comparator. Then based on a 100mV change that is determined from a selection of resistors, the output of the comparator will swing from 0V to 5V for darkness and sunlight respectively.

If a Real Time Clock (RTC) is to be implemented, we should use the DS1306 SPI RTC. This requires the SPI communication to be set up on the microcontroller. After attaching a 32KHz clock to the X1 input of the RTC, as well as the various power and ground lines, the RTC can then be programmed. It is programmed Address first and Data

second. The device comes with the appropriate data sheet, though it is a little vague as to how to program the device.

2.1.3 Microcontroller

The controller we chose to go with was the 68HC12912EVB, which already contains all the necessary hardware for the entire controller to be programmed. If we chose to buy the controller chip separate, then a board will be required to house the necessary components as well as the controller chip. Then utilizing a compiler program (like the COSMIC Compiler IDEA CPU12), this will allow us to generate and then compile our programs. Then all is needed is a file transfer application. An HC12 HyperTerminal, or even the WIN 3.11 Terminal, will do in order to transfer the .s19 file that programs the controller.

2.1.4 5V Regulator

The 5V regulator is a chip which receives an input voltage greater than 5V. With an input greater than 5V, it then outputs a steady 5V. The chip we used was a MAX667, which is a low dropout, positive, linear voltage regulator that supplies 250mA of output current. The regulator was placed between the battery and the circuitry. The use of this device is required since all of our analog components require a 5V input.

2.1.5 Mosfet Switches

To limit our power usage we implemented some switches, which will provide power to components such as the adjustable program switch and the moisture sensor only when power is needed. The use of switches will then consume power by not supplying power to devices that don't need to be continuously on. To do this, we used analog Mosfet IRF9510 switches. From the microcontroller we can then send a signal to

activate the switches when power is desired at various components.

Chapter 3
Member Contributions

3.1 Michael Carpenter

My contribution to this project includes research and development of the moisture sensor, distinguishing plant species and helping Rocky incorporate the sub-programs into the HC12, and power distribution. Our customer advised that I spent time in the fall semester researching on what type of moisture sensor to use or develop, try to categorize plant species (if need be) in some fashion for watering, and scavenge ways to distribute power accordingly to utilize as little as possible. Although I got off to a slow start in the spring semester due to many mishaps, I still was able to pull through and make my decisions on all three aspects.

3.1.1 Moisture Sensor

My contribution of the project includes developing a moisture sensor as well as researching the various plant species' moisture intake. At first, I decided to approach the conventional method of researching various commercial moisture sensors. I realized there were two types: capacitive and resistive. Capacitive sensors are typically more efficient as they are more tolerant to high humidity levels. The most important feature to focus on is one with internal logic. These sensors typically require a ~5V source to provide input frequency. With the internal circuitry capabilities, it will convert the frequency to voltage, then convert that analog signal to a digital one. There is no need for time and power-consuming calibration. Other traits that were investigated upon were accuracy (only $\pm$ 2% error), long-term stability, ability to recover from condensation, chemical/physical contaminant resistance, size, cost, and low current consumption (<uA

or mA, little power used), a cover of some sort (to protect it from dust, dirt, oil, etc.), linear voltage output (stability with low relative humidity error), operation in wide humidity ranges, fast response time (once fed with power, needs to operate quickly).

However, everything came to a complete halt, when 2 events occurred. First, a Honeywell engineer bluntly informed me that I was investigating the wrong type of sensors and none of them would be able to be submerged into any forms of media. I realized that was 2.5 months of wasted research. Second, I spoke with an environmental engineering professor and his descriptions of his homemade moisture sensors match perfectly to what the specs desire; however, he did not allow me to use it for the following semester. At this time, I came to the simple realization that I had to construct one. Dr. Donohoe suggested we spent the first semester researching the project and the second semester actually developing it. Already 2.5 months went under way, so I had about half a month to figure out a way to develop an efficient moisture sensor. I spoke with Dr. Donohoe and we came up with a schematic.

When I returned for the next semester, I had problems right off the bat (other problems that plagued me throughout the semester was bad parts overall – mainly multimeters and resistors), and I decided to flirt with other schematic designs. The original schematic was in the form of a resistance-to-voltage converter. The resistive probes would be in the soil and it would feed into the converter. Vref (reference voltage) of preferably 5V is used from a voltage divider. That connects to a series resistor and connects to a negative input of the inverter. The positive input is grounded. Then a potentiometer of 100k (variable feedback resistance – moisture probe) connects itself to the output from the negative input. Vout (output voltage) would be the negative R

(resistance) of the pot divided by the series R. As long as Vref is fixed, Vout depends only on the variable feedback R. This is like a single-supply OP-AMP. Furthermore, the output connects to some level detector (some adjustable level threshold circuit or an A/D (analog-to-digital) converter which is conveniently in the HC12 microcontroller – 0V for resistance below certain level. 5 for above). The whole setup requires an OP-AMP, a resistor, a potentiometer (seems like a Schmitt trigger displaying hysteresis characteristics). We were also thinking about creating a circuit with LP (low-pass) frequency characteristics to eliminate potential noise. I found a JFET that can do this; however, this plan also fluttered for one simple reason: it draws way too much power (in the W range).

From this point on, I tested several circuit designs. One was somewhat of a discrete circuit. It is a comparator, or once again, a single-supply OP-AMP (LM319. LF411 is considered a dual-supply). Vref and Vin (input voltage) displays hysteresis. Once the probe and some resistors are connected to the positive input and Vref is connected to the negative input, the resistance connecting Vout displays the hysteresis width. I plan on hooking the output to the input of the HC12. To realize the output current, one must subtract the base voltage by .7. Then divide that by the R connected to GND (ground – perhaps 100k).

Another circuit was a generic transistor. The base would be connected to the HC12 (5V – on, 0V – off). Vcc would be sending 5V through to the emitter. It would go to the A/D converter where it's 0 ohms. Basically, the comparator will switch the sensor on and off with a stable current source, acting like a buffer. The problem I realized is that this particular OP-AMP requires a lot of power.

Another circuit was a comparator-type of device (determining dirt is dry or not at a specific resistance), but it can only go up to 2V with the power supply for efficiency (to be discussed later – the minimum necessary voltage for dry dirt in order for a sensor to operate is 3V).

Finally, the working circuit consisted of simple current sources. Current sources supply .18mA when Vb (base voltage) is properly tuned using the variable resistor at the base of the transistor. .18mA is provided, so Vo (output voltage) from each of the sensor should match the graphs I provided (Figs. 12-15). I could not set up a special voltage (0V = dry, 5V = wet) without adding extra active circuitry. I did not really do a quantitative analysis (little wet, really wet, etc.), but I figure the HC12 microcontroller should switch on the waterer when the sensor goes out of a certain range of voltages between the dry and wet bars on .18mA bar graph. There is also a saturated transistor switch incorporated into it (Fig. 4). This allows a small control current to enable a much larger current to flow in another circuit (for this case – the current source). The result is once it is fed .18mA, it should say 3.6V (near necessary minimum according to the graph) for the dry dirt. The actual value resulted in 3.61V (Fig. 13).

I decided to measure resistance of dirt as well as it may vary with different voltages applied to dirt. Voltage varies with a set resistance range. The dirt's resistance changes with applied voltage because there were significant changes in the resistance readings when switching ranges.

Our advisor, Dr. Bruder, threw in a few ideas. One was to use two sensors and taking the average of the two readings to receive a better resolution. I decided to go with one because as long as it's placed near the actual roots, that is precise enough to receive

accurate readings. He also thinks I should derive measurements at specific water levels (hysteresis analysis), even utilizing a measuring cup. I decided to just measure wet dirt and dry dirt and the readings produced constant results.

I decided to provide graphs to search for consistencies (Figs. 10-13). I dipped the electrodes (copper probes) into dry soil, wet soil, and water at a specific depth and distance apart. I got several voltage and current readings and noticed specific patterns. Of course I had to fiddle with several current sources since several were malfunctioned. Variables in this test include length of submerged portions of electrodes at certain depth, independent voltage, dependent current (measured value), resistance voltage/current (dependent).

Furthermore, I took three separate points (current on a current vs. resistance graph) on these graphs. There are nine parameters (dry dirt, moist dirt, water, at .1mA, .14mA, and .18mA). They all had a downward trend (i.e. as current increases, resistance decreases – which makes sense as the voltage is increasing). This is the comparator I was looking for as it compares one voltage to another. I set it at 15k since that is the highest resistance (about 14.25k at the lowest current). If it is above the threshold, then the soil is dry. If below – consider it wet. In addition, I can use the lower current (.1mA) due to the widest marginal behavior (very high resistance for dry dirt – near 30k to moderately-low resistance for wet dirt – 14.25k). So .1mA is good and I thought maybe even lower. The lower the better since we need to consume as little power as possible. The maximum power allowed running the sensor now would be about .3mW (power = (.1mA)^2 * (28k)). The minimum voltage necessary is about 3V (voltage = (.1mA) * (28k)). The next step was to connect all this to one of the A/D

converters available on the HC12 microcontroller and realize the HC12's input impedance.

3.1.2 Plant Species Investigation

In addition to developing the moisture sensor, I investigated various plant species, mainly their water intake level (Fig. 9). Each plant species display various characteristics and are inconsistent in comparing with others. The moisture intake level is the amount of water a plant can/should receive – which are typically categorized as: low, medium, or high). This, as well as plants' humidity tolerance, is the typical characteristic of determining proper water usage for plants. Plants that allow fewer levels of moisture than normal tend to tolerate humidity more – and vice versa. No exact recommendations were exactly given for watering plants (including a lady that insisted that it had to be watched by a human), but there are some general guidelines. The soil at each watering should be thoroughly wetted or moist – water should drain out the bottom of the pot (or wherever) after watering; hence, we should use containers with drainage holes. Plastic pots (what we intended to use, but went with Plexiglas, which is still alright) is typically better than clay ones. Clay pots dry out faster and are more difficult to clean than plastic ones due to the fact that clay absorbs chemicals and salts. Frequency of watering depends on many factors, but mainly plant species' moisture intake level. If all cases fail, we should not water more frequently than required; wet soils can lead to root rot problems. In addition, excessively wet soils lack the oxygen required for root growth. Watering plants at room temperature water is also a must – as long as the water is not cold. Since plants do not conform to just one moisture intake level (and as explained, each plant displays various characteristics), we will be implementing three

programs (deduced from the original five-ten) into the HC12 microcontroller. The sensor will act like a…well, "sensing" device to measure how much moisture it needs. It will inform the HC12 whether if it's dry or wet by the predetermined threshold. This device will send a signal to the HC12 and execute the desired program.

3.1.3 Power distribution

The self-watering planter will require power to various subsystems that include moisture sensor, the PLD, and pump. The theme of the project is power scavenging; hence, to consume power, components were chosen based on their performance and power draw. The water pump pulls 6V @ 500mA, the moisture sensor utilizes 5V @ 20mA, and the PLD/circuitry utilizes 5V @ .2mA. In order to determine the total power draw, calculations were computed and the planter draws .533A*Hr./day. With this in mind, I was then able to predict what type of solar panel and battery we would need. For the other electronic components we installed a MAX667 (LDO) 5V regulator. This low dropout voltage regulator only uses 20uA of current. This 5V regulator provides power to the moisture sensor, HC12, and adjustable switches. Mosfet IRF9510 switches were placed on the circuit board to provide power to the moisture sensor circuitry and the pump when needed. These switches are activated by a low signal from HC12 ports. A low dropout voltage regulator was used to regulate a steady 5V source, the moisture sensor, HC12, and switches used 5V (Fig. 1).

My contributions were an overall success with all milestones achieved (Appendices B-C). The moisture sensor is efficient as it acts like a hand, sensing for dryness or wetness. The HC12 microcontroller is the brain that sends the water. All components utilize as little power as possible. In order to gain conclusive measures, one must realize

the problems. There are three main problems/objectives that I had to solve. First, I came to the realization that dirt can easily become dry, which in turn will easily and quickly kill the plant. So a moisture sensor had to be developed in order to alleviate this problem. Realizing that the majority of the commercial sensors are inapplicable to our situation, I developed my own circuit (after looking at an immense number of circuits) – a saturated transistor switch with a current source. This utilizes a simple copper electrode which is submerged into the dirt. This electrode tries to realize if the dirt has no ion flow or current (i.e. no water). Upon a few experiments, I set it to a certain threshold and when it approaches that voltage, the electrode sends a signal and triggers the HC12 to execute one of the desired programs. Once executed, water flows through the tube, which is embedded in the dirt and very close to the root as possible. Second, since I mentioned "one of the desired programs," I came to the realization that there are many species of plants, just like animals. Since watering is our main objective here, I classified each plant according to their water intake level. The three main ones are: lots of water, medium water, little water – hence, I utilized three different subprograms. Fortunately for us, if we cannot decide what kind of plant species it is or some other circumstance arises, the golden rule of botany is that if the dirt is dry – water it. It may not grow to its maximum capabilities, but as long as it is watered, it is still alive. As long it is not watered frequently and excessively, it will be okay. Frequent watering will cause root rot among other extremities. Third, just like in real-life applications, I came to the realization that all components use power. In addition, just like a typical consumer, they want to utilize as little power as possible. Hence, Randy and I specifically distributed the power to make sure the overall planter is sufficiently satisfied with power. The water pump

utilizes 3W, the moisture sensor utilizes .1W, the HC12 and circuitry utilizes .01W. Once the calculations were realized, there is very little power being dissipated.

3.2 Rocky Ginanni

My contribution to this project lies in the area of energy storage, microcontroller, external interrupt device, and programming. At the beginning of the fall semester, we were approached by our client to construct a self-watering planter that employs many features. These features include storing energy that is derived from a means other than the power from a wall outlet, and then some way of controlling the watering of a plant based on some form of sensor input.

3.2.1 Energy Storage

So, my criteria in determining energy storage requirements was that the storage device had to be cheap, easily rechargeable, has a high-energy density, has a long life cycle, and has low-to-no maintenance. So, after a considerable amount of time looking for ways to fulfill the requirement, I narrowed our choices down to "chemical" (batteries and/or supercapacitors). Below is a list of commonly available batteries and capacitors as well as their benefits and drawbacks.

Batteries

Sealed Lead Acid:

It has many benefits. It is inexpensive, reliable, and relatively forgiving to overcharging and deep charging. In addition, it has a fairly long cycle life, has high energy densities, and its cells have a nominal discharge potential of 2V. However, it does have its drawbacks. It has a relatively low power density and relatively rapid self-discharge.

Nickel Cadmium (Ni-Cd):

It has many benefits. It has a longer lifetime than Lead Acid, it has a greater density than Lead Acid, and it has a high constant discharge rate. However, it does have its drawbacks. It has lower cell voltage (1.1V to 1.3V), it is more expensive than Lead Acid, it is sensitive to overcharging or undercharging, and it requires special charging techniques for full usage. In addition, at shallow discharge cycles, the batteries can develop voltage depression or memory effect. A memory effect occurs when the battery repeats a few full cycles of complete discharging and charging restoring the full capacity, therefore erasing the memory. The unused capacity of a cell cannot be utilized if the cell is not fully discharged. Also, at high temperatures, the battery has possible performance problems.

Nickel Metal Hydride (Ni-MH):

It has many benefits. It has low maintenance, a long cycle life, a high energy and power density. In addition, it is non-toxic and it does not suffer from the memory effect. However, it does have its drawbacks. It has a very high self-discharge rate and it is very expensive. In addition, it requires specialized charging techniques and it is more sensitive to overcharging and complete discharging than Ni-cads.

Lithium (Ion, Polymer, etc):

It has many benefits. It has a high energy density, a long cycle life (very low self-discharge rate), and it has a very good power-to-weight ratio. In addition, it is compact and it has more power than alkaline. However, it does have its drawbacks. It is very

flammable when exposed to water, very expensive, Li-ion is difficult to charge, and Li-polymer is cheaper and it is a higher energy density version of Li-ion.

Reusable Alkaline

It has many benefits. It has a long storage life, it is very powerful (for a limited time), it has high energy densities, and it is available everywhere. However, it does have its drawbacks. It is very expensive over time, it is not very dependable, its power drops off with repeated charging, it has low power densities, and it is environmentally unfriendly.

Note: The various types of batteries and their performance characteristics are shown in (Table 1).

Capacitors

Capacitors are devices that store energy by maintaining two charged interfaces of opposite polarity, separated by a dielectric/electrolyte. The energy stored by capacitors is related to the charge at each interface, q, and potential difference, del V, between the electrodes.

$E_{capacitor} = q\ V$

Since capacitors store charge at the interface only, rather than within the entire electrode, they tend to have lower energy densities. The charge/discharge reaction is not limited by ionic conduction into the electrode bulk, so capacitors can be run at high rates and provide high specific powers. Typical numbers for capacitors and batteries are given in (Table 2). Since there is no bulk change in the electrodes, the charge/discharge reactions can typically be cycled many more times than batteries (10^8 cycles/device have been achieved).

Supercapacitors

The operating principle of the Supercapacitor is based on an electric double layer appearing at the interface between activated carbon particles and sulfuric acid solution as electrolyte. An ionically-conducting but electrically-insulating porous membrane separates the two electrodes. Conductive rubber membranes contain the electrode and electrolyte material and make contact to the cell. Several cells are stacked in series to achieve the 5.5V and 11V rated voltages. Since Supercapacitors exhibit relatively high ESR, they are not recommended for ripple absorption in DC power supply applications.

Supercapacitors or electrochemical double-layer capacitors (EDLC) take advantage of the charge stored in the electrochemical double layer and provide extremely high capacities of more than 1000 farads. These devices have applications in computer power backup, power electronics, electric vehicles, and space flight technology. However, power and energy demands of these applications vary significantly. The Supercapacitors do have its drawbacks. It has a low energy density (energy-to-weight ratio), its varying voltage supply is proportional to the stored energy, and it has high material costs. However, it does have its benefits. Little maintenance is required, it has a long life, it has a high cycle efficiency, it has high power and power density, it does not use toxic or explosive fluids, and it has a high recharge and discharge rates.

After evaluating all batteries and capacitors, I decided to go with a Sealed Lead-Acid Battery type. The battery that I got has a 6V/2.8 A*Hr. capacity. Besides being readily available, they are more forgiving to overcharging and undercharging which means that our charge controller need not have to be complex in function. They also meet all of the criteria stated previously.

From investigation, the use of supercapacitors would not be a cost effective alternative because they do not have high enough energy densities to sustain a high continuos load draw over time.

3.2.2 Light Sensing/Real Time Clock

The key critical component of the project is actually the device that generates the external interrupt that is fed into the IRQ pin of the controller. The device that the planter is presently running is an IR Phototransistor circuit, (Fig. 5). The light sensor, when used in conjunction with a comparator circuit, will generate a 0 or 5V signal for darkness and sunlight respectively. This signal is CMOS compatible with the HC12's IRQ input. When I tested this circuit, it worked perfectly. It was able to output 5V when light hit the sensor, and 0V when the light was taken away. The only drawback to this sensor was that it responds to any kind of light, sunlight or artificial light. So for this planter, as soon as the sunlight hits it in the morning, it will constantly trigger until sunset; thus, causing the controller and the rest of the circuitry in the planter to use power needlessly. So I decided to use this type of sensor as a last resort if I could not find some other kind of device that was more efficient.

So, after searching extensively for an alternative replacement, I found a suitable test subject called a Real Time Clock (RTC). This device functions almost like your bedside alarm clock. You program time and date, as well as the alarm time and date. When the alarm time matches the current time, it generates an interrupt that can be fed into the HC12. It can be programmed to generate a signal once/day that would wake the HC12 up and then once the HC12 executes the program, it would go to sleep until the next day.

Two RTCs that I found are the Dallas DS1306 SPI RTC and the Unitrode BQ3287E Parallel RTC. I preferred the DS1306 RTC due to its SPI clocked communication. So with just a little time left before the Senior Design Presentation, I started trying to program the RTC. Unfortunately, I was unable to get the device to accept the program, so I had to resort to the light sensor. But, with power conservation being the main focus of this project, it would be beneficial to further explore an RTC as the source for an external interrupt.

3.2.3 Microcontroller

The choices for a device that can handle the complexity and the length of a program that can handle the various mechanical and electrical functions of this project have to be considered. The various controllers that could handle these parameters are the Altera Programmable Logic Device, the Motorola 16-bit 68HC12, the PIC, and the STAMP. I eventually chose the Motorola 68HC12. I utilized the Motorola 68HC12912B32-EVB, which is an evaluation board that houses the 68HC12912B32FU8(80) Microcontroller chip. This board was provided to us for our EE 308 Microcontrollers class. You can buy this particular evaluation board from the manufacturer for ~$90, or buy the chip for ~$30 and then create your own board.

This 16-bit controller can handle rather complex programs with minimal power usage. It also has the ability to go into a power-saving mode. This mode consumes minimal power when processing is not required during the night, as in our case. Though this type of controller is a bit overkill for our small application, I happen to be familiar with the various functions of the device that can allow me to prototype the program that

will be used. From this prototype, I could use it to program controllers that are less costly and just as functional.

To make this planter work, I had to utilize 1-Pulse Width Modulated signal port, 5-Analog-to-Digital ports, and 2-Output ports, (Fig. 6).

3.2.4 Program

In order for this planter to successfully work, it is necessary to measure if the battery needs to be or is able to be charged and to determine if the plant needs watering. So, to realize this, it is necessary to create a program that can execute these two criteria and then after it has done something or not, tell the controller to go into a sleep mode.

For the code, I used C, a high-level programming language. This language allowed me to create a behavioral-type program that is relativity short in comparison to other types of programming languages like Assembly, BASIC, FORTRAN, etc.

To write the code and download the necessary file into the controller, I used the COSMIC Compiler IDEA CPU12 to create the program and a HC12 HyperTerminal window to handle the data transfer. For the completed program refer to Appendix A.

I created a flowchart that shows the structure of the program that controls the function of the planter (Fig. 8).

3.3 Randy Sena

My contribution to the project includes research and development of the planter design, pump, solar panel, charge circuit, and adjustable program input switch. For the majority of the first semester, research was done on all of these areas in an effort to investigate the methods we used along with alternative methods. Through research, I was

able to narrow down my decision by comparison of cost and effectiveness in order to make my final choice on specific components.

3.3.1 Planter Design

The planter design was first thought out to be a modification of an existing planter. The intention was to modify a planter in effort to meet our needs. Shortly after trying to accommodate various planters, it was finally decided to build our own planter. The planter is constructed entirely of clear Plexiglas (Fig. 7). The use of Plexiglas offers a unique style which allows visibility of all circuitry. With the clear Plexiglas, you can also visibly monitor the water level inside the water storage area. The lower two thirds of the planter offers a 285 cubic inch water-holding area, which has a refill spout and a drain plug. Also, on the lower two thirds is a containment area for the pump. The upper third of the planter houses the soil, which holds the plant. Between the upper third and the lower two thirds of the planter is a divider which has perforations with a coffee filter. The coffee filter allows added protection against soil entering into the water-holding area. The idea behind the filtering is to implement water recycling. The total dimensions of the planter are 9″ x 13″. Under the planter is a separate water -tight compartment to house the controller, circuitry, and battery. It has dimensions of 2.5″ x 9″ x 9″. A 6-pin wiring harness was designed to allow communication between the pump, moisture sensor, and solar panel. Inside the lower circuitry containment area, Velcro was used to hold the components in place.

3.3.2 Pump

A pump was necessary in effort to get water from the holding area to the soil. We looked at two methods to perform this task: a pump and a bladder (pressure) method.

The bladder method involves steady watering which does not quite fit our needs since we must provide variable watering for different plant categories. The pump turned out to be the best fit for our needs. We found numerous pumps that would accommodate our needs. After comparing cost, voltage, and current consumption, we decided to go with a Techtronics 6V @ 500mA water pump. The pump will pump 12 oz./min. at a height of 42 cm. We ordered the pump and tested it as soon as we received it. The pump did live up to its minimal expectations and thereafter we decided upon it as our final choice for water pumping. The cost on this pump was also very minimal in comparison with its competitors.

3.3.3 Solar Panel

Power scavenging is one of the themes of the project, so we had to look at various sources for obtaining power. The choices we had to choose from include harmonic vibrations, wind-generated power, and solar power. For our application, we could not find a steady source of wind in a window pain area, so the method of wind-generated power was eliminated. Harmonic vibrations were recommended by our customer; however, in a planter and its' stand, we could not find sufficient vibrations to produce the desired power. As a result, we turned toward solar power since plants require sun and they are usually placed near a window. When we turned toward solar power, we investigated many solar panels. From resources, we found that we would have to allow for around 20% loss of power due to imperfect sun conditions. Through calculations, we estimated that we would require at most .533 A*H/day. With the desired voltage and needs calculated, we now knew what voltage and current we would be needing. We needed an 8V @ 400mA solar panel. We were allowing for 2V @ 200mA loss since our

desired need was 6V @ 200mA. When we received the panel we ordered, we immediately tested it. Amazingly, we were proved wrong by the panel, as we only had .2V @ 20mA loss. The solar panel was nearly perfect! With this in mind, we immediately chose the Silicon Solar Panel as our final choice. The panel was placed at the top portion of the planter side, so max sunlight would hit it during lighted hours (Fig. 7).

3.3.4 Charge Circuit

The solar panel could not be connected directly to the battery for charging due to over and undercharging. We needed to implement some type of charge circuitry. At first, we looked into various charge circuits and methods from our EE 322 Advanced Electronics book, The Art of Electronics. We thought we could use an LM-317; however, after testing it, we found it to be inapplicable due to the fact that there needs to be at least a 3V drop across the solar panel to the battery. We researched various designs and received many ideas for charge circuits. We tested several designs, but could never get any of them to properly work. After weeks of research and still no answer to charge circuitry, I went to consult with Dr. Ron Thomas. He gave several leads on charge circuitry, one of which included the use of mosfets. Finally, we came up with a design that theoretically worked (Fig, 3). Once we had the design, we tested it. We found charging success within the circuit. It consists of an N-channel and a P-channel FET. The gate of the N-channel receives a pulse width modulated signal (PWM). The source is grounded and the drain is tied to a 100k resistor and the solar panel input voltage. The gate of the P-channel FET is tied to the drain of the N-channel FET and the source is tied to the solar panel input voltage. The drain is then tied to the battery and a 5V low

dropout voltage regulator (LDO). The large value of the resistor was used to limit current. By varying the PWM signal you can limit the amount of voltage passing through the charge circuitry to the battery. You can pass anywhere from 0V to Vs (Vs is the solar panel voltage which is around 7V). The charge circuit is placed between the solar panel and battery. It is located under the planter in the circuitry housing area.

3.3.5 Adjustable Program Switch

The adjustable programming switch consists of a three-pole toggle switch that outputs 0V, 2V, and 4V (Fig. 2). High values of resistors were used in order to consume power. The switch outputs are tied with the HC12 A/D converters. From this point, the program monitors which value corresponds to which of the three watering programs that is to be implemented.

Chapter 4
Summary/Conclusions

The Self-Watering Planter was a project that required research and development. Our customer, Dr. Greg Donohoe, emphasized that we heavily researched various areas that would contribute to the project for the majority of the Fall semester. Through research, we were able to look at various methods of problem solving along with ideas on how to achieve the common goal of designing a self-watering planter. The Self-Watering Planter provides: power scavenging, water recycling, water-holding tank, adjustable programs for various plant species, microcontroller, moisture sensor, battery charger, battery, pump, and a unique planter design. Utilizing the above attributes, we have successfully designed a revolutionary planter that meet all the milestones and will automatically water a plant species without human intervention. During the duration of the project, we met weekly with our advisor to discuss progress and problems with the project. We emailed our customer weekly to update him on our status. Biweekly, we provided our customer and advisor with a progress report. After conferring with our customer and advisor, we have all determined that we have successfully met all criteria for the project.

Chapter 5
References

Books

The Art of Electronics

Paul Horowitz and Winfield Hill

Websites

Plants.usda.gov

"all plants are unique"

Siliconsolar.com

People

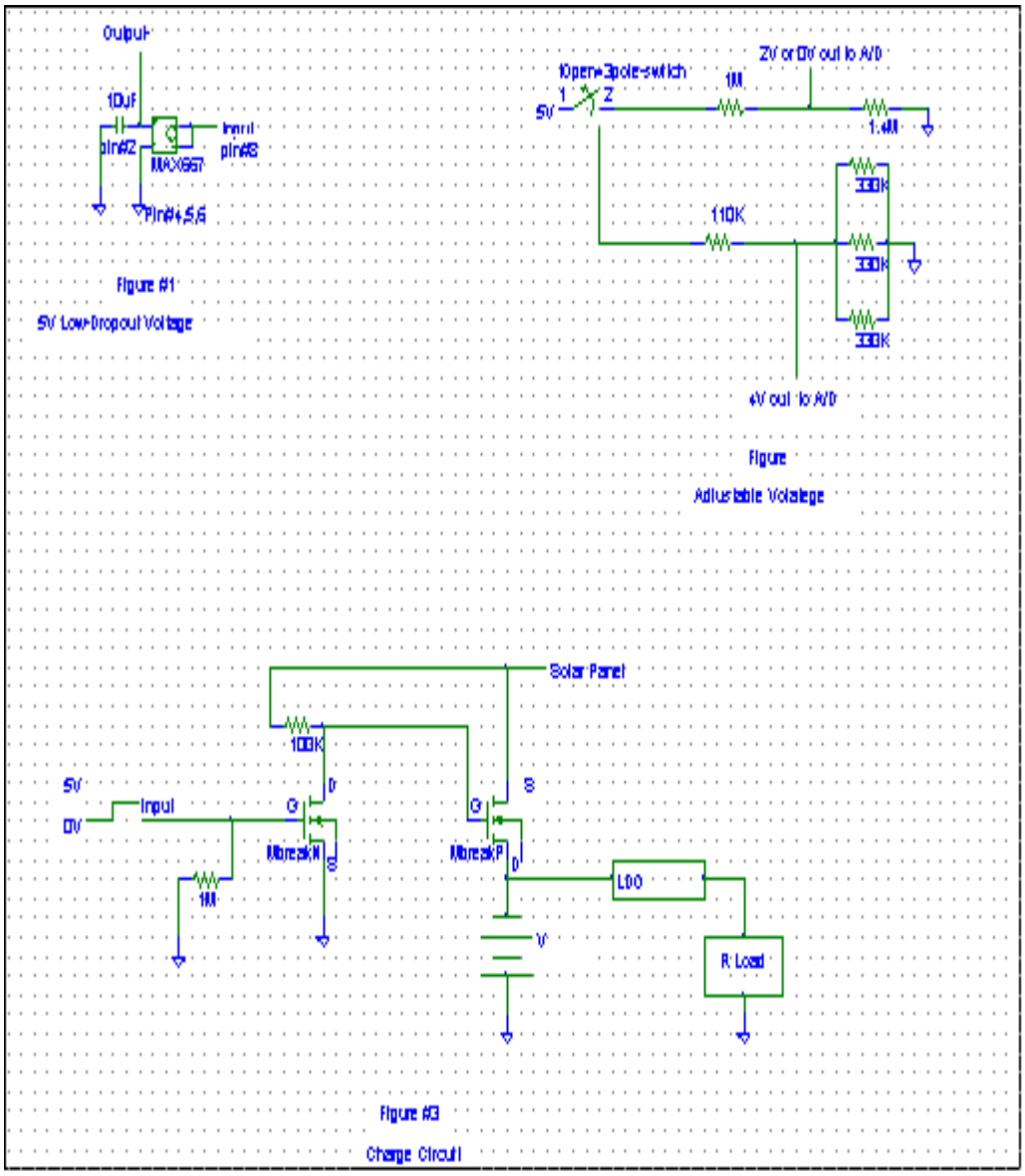
Dr. Greg Donohoe (UNM Electrical Engineering Department)

"I appreciate all this research. This is an important part of the Senior Design effort, and should be a prominent part of your reports…this is science!"

Dr. Bill Rison (NMT Electrical Engineering Department)

Dr. Ron Thomas (NMT Electrical Engineering Department)

Figures: 1, 2, 3


Output
10uF
pin#2
MAX667
Input
pin#3
Pin#4,5,6
Figure #1
5V Low-Dropout Voltage
10pcs 3pole-switch
1
2
5V
1M
2V or 0V out to A/D
1.4M
330K
330K
330K
110K
4V out to A/D
Figure
Adjustable Voltage
Solar Panel
100K
5V
0V
Input
G
D
S
Mbreak:N
Mbreak:P
1M
LDO
V
R Load
Figure #3
Charge Circuit

Figures: 4, 5

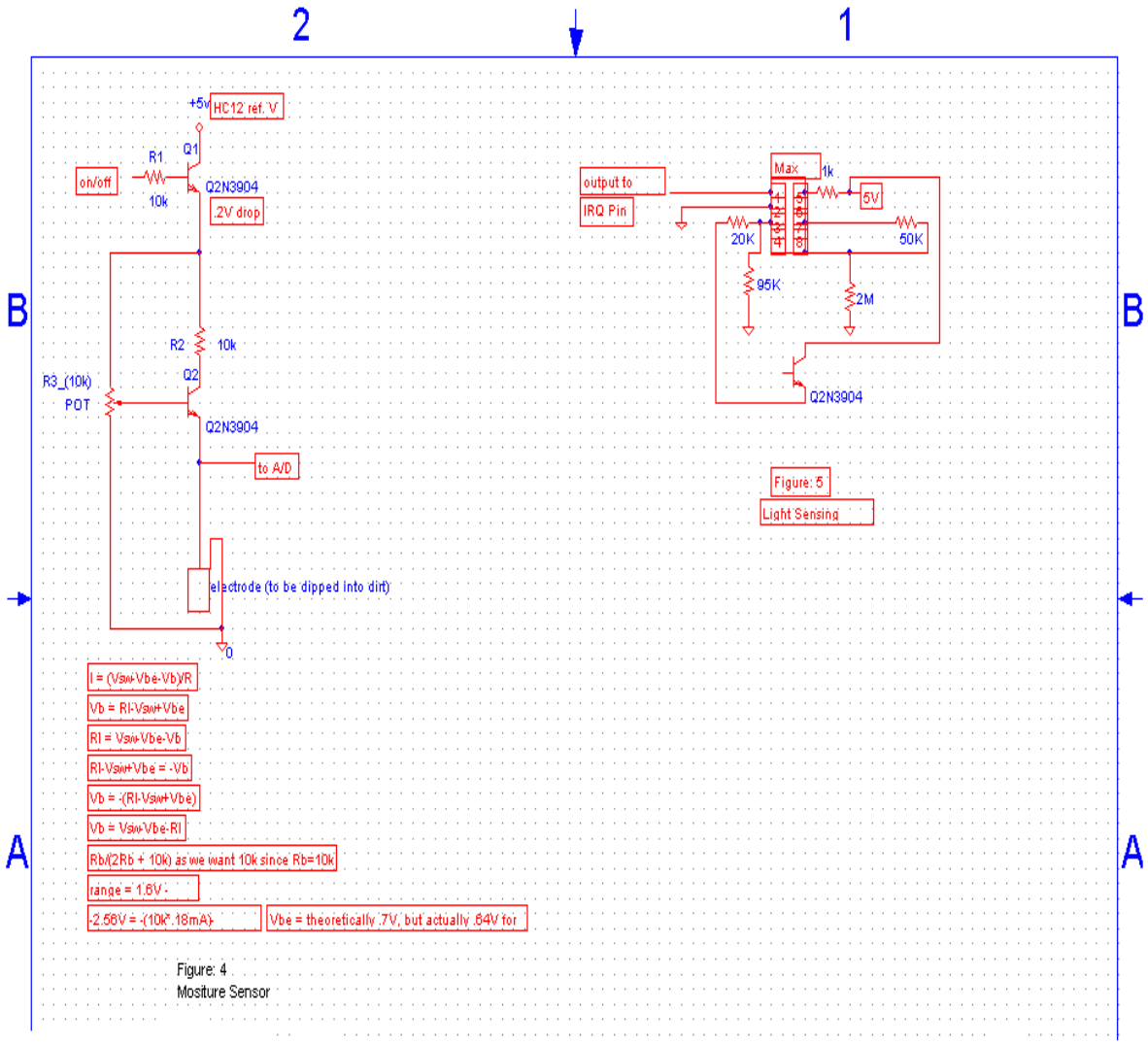


Figure: 4
Mositure Sensor

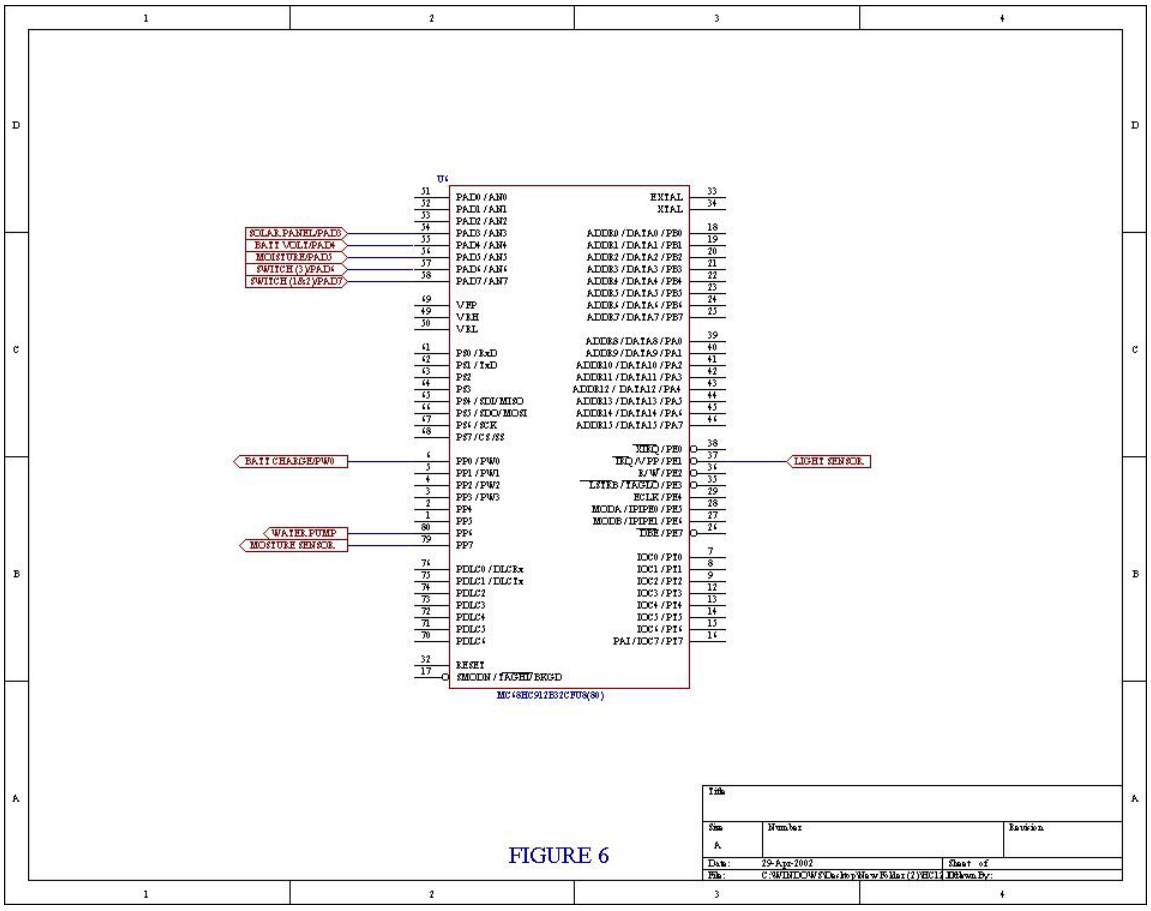
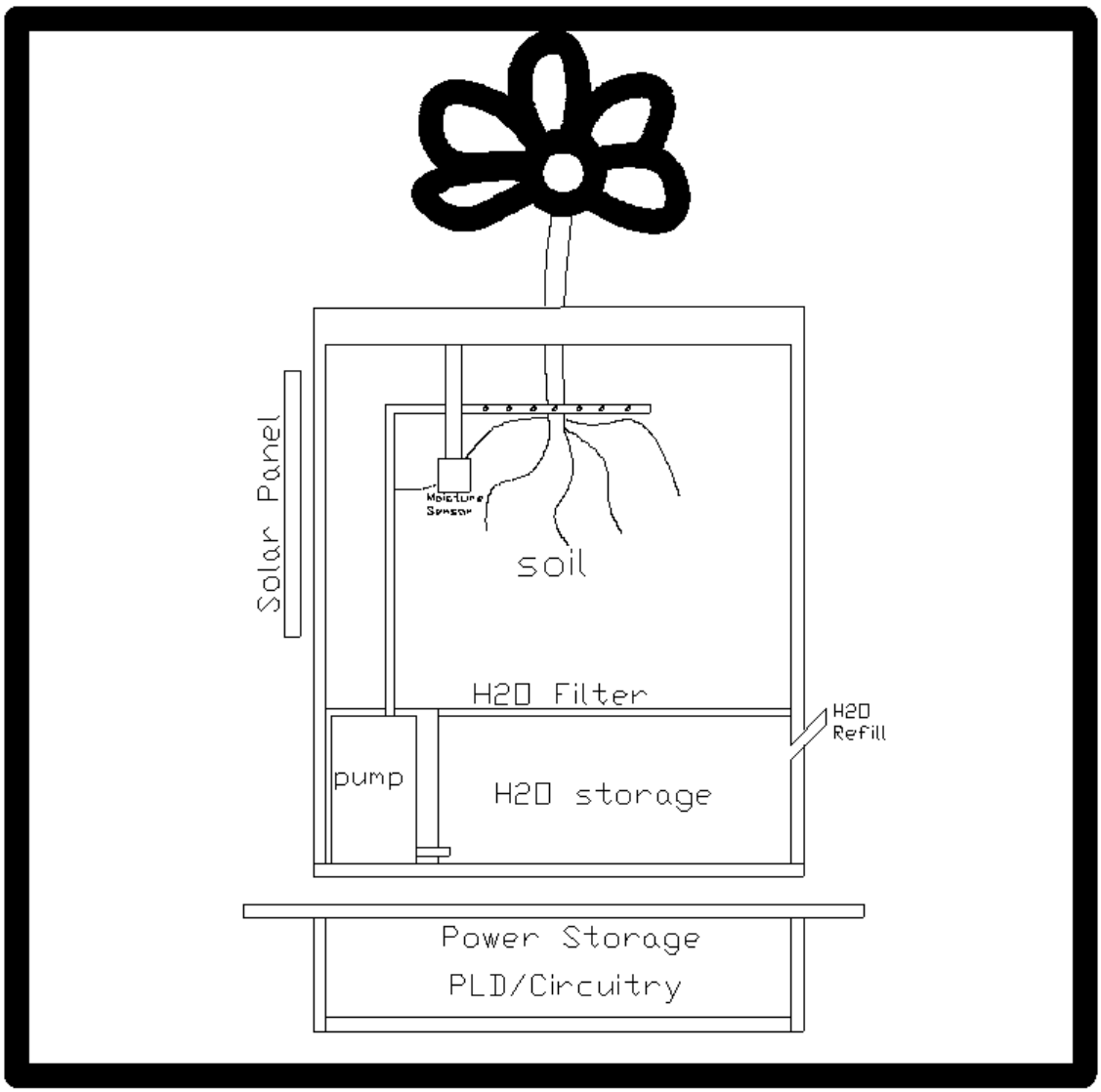


FIGURE 6

Figure 6: Motorola 68HC12912(80) Microcontroller

Figure: 7

Self Watering Planter



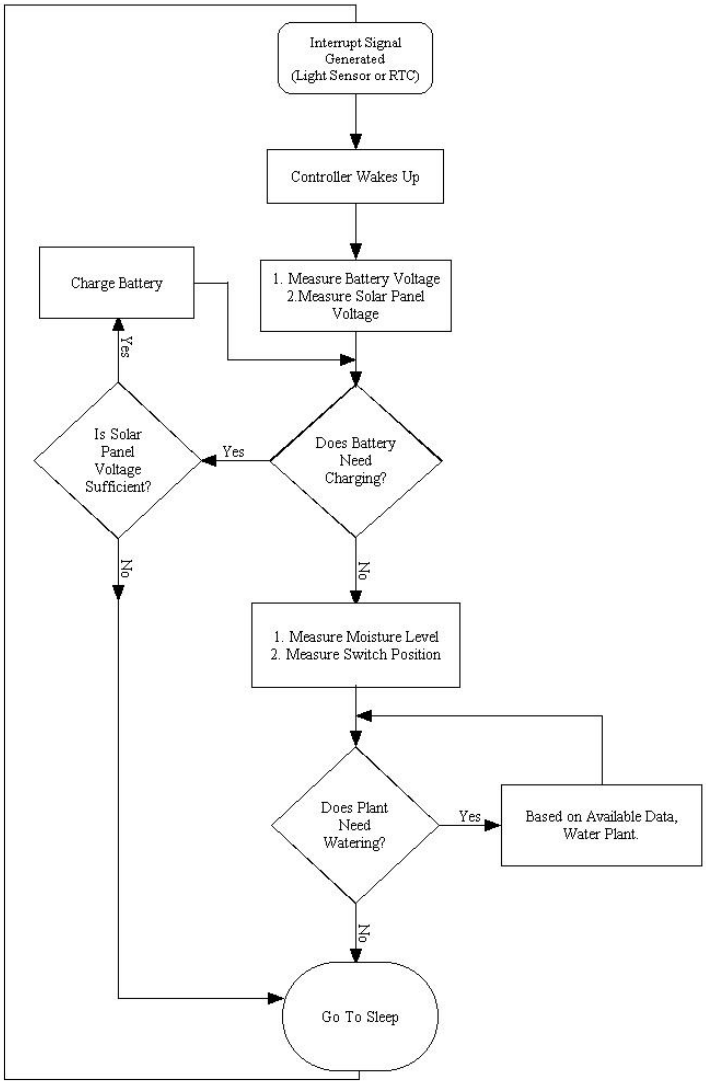


Figure 8: Flowchart of controller program

Figure 9: Plant Species

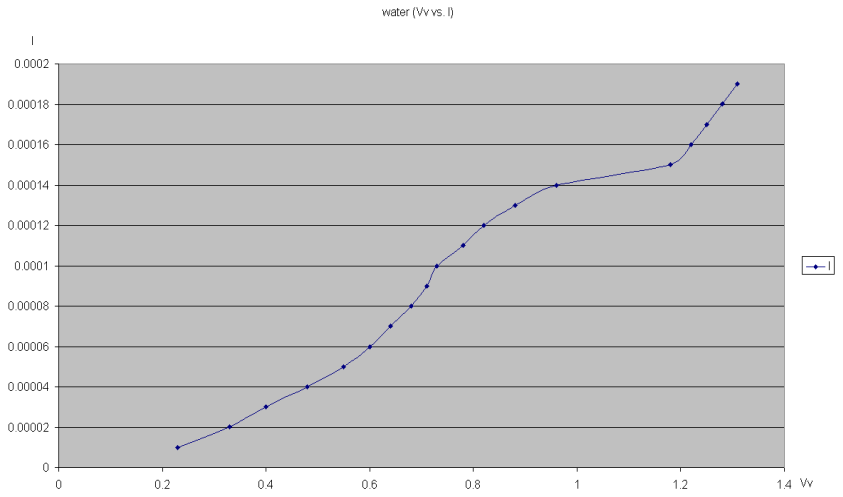
Moisture level	Common name
Thoroughly wet	Dracaena
Evenly Moist	African Violet, Begonia, Bromeliads, Chinese Evergreen, Chrysanthemum, Ivy, Fern, Fiddleleaf fig, Gardenia, Hibiscus, Pine, Palm, Lily, Philodendron, Spineless Yucca, Velvet Plant
Drench, then dry	Cactus, Citrus, Orchid, Poinsettia, Umbrella Tree, Wax Plant

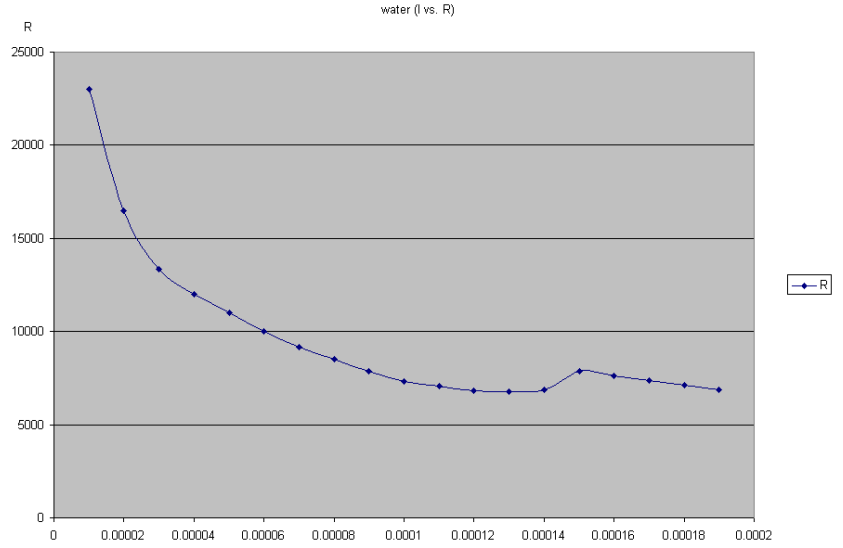
Description of terms (for a few examples of common plant species above)

1) Thoroughly wet (low, typically tolerant of humidity) — Daily watering generally required. May stand in water for brief periods.
2) Evenly moist (medium) — Frequent watering required, but must never stand in water. Soil surface should always feel moist.
3) Drench, then dry (high, typically intolerant of humidity) — Soak root ball thoroughly, then allow the soil to become fairly dry before watering again. Do not allow the plant to wilt, however.

Fig. 10 (Sensor Test for water)

Vv	Vi	I	R	R=	100000
0.23	1	0.00001	23000		
0.33	2	0.00002	16500		
0.4	3	0.00003	13333.33		
0.48	4	0.00004	12000		
0.55	5	0.00005	11000		
0.6	6	0.00006	10000		
0.64	7	0.00007	9142.857		
0.68	8	0.00008	8500		
0.71	9	0.00009	7888.889		
0.73	10	0.0001	7300		
0.78	11	0.00011	7090.909		
0.82	12	0.00012	6833.333		
0.88	13	0.00013	6769.231		
0.96	14	0.00014	6857.143		
1.18	15	0.00015	7866.667		
1.22	16	0.00016	7625		
1.25	17	0.00017	7352.941		
1.28	18	0.00018	7111.111		
1.31	19	0.00019	6894.737		





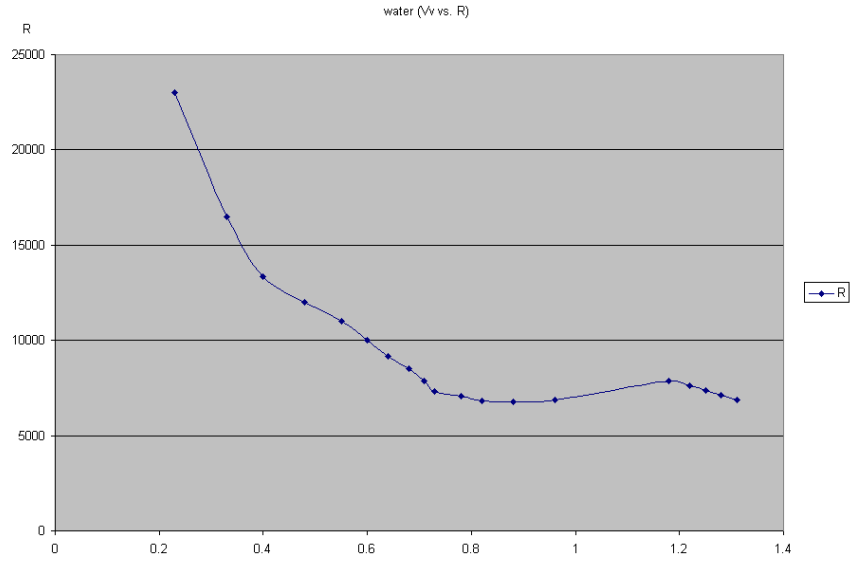
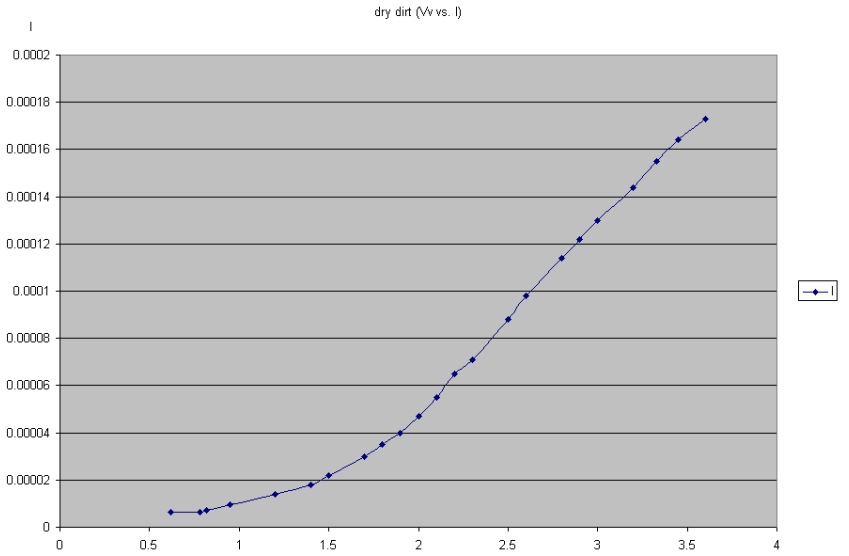
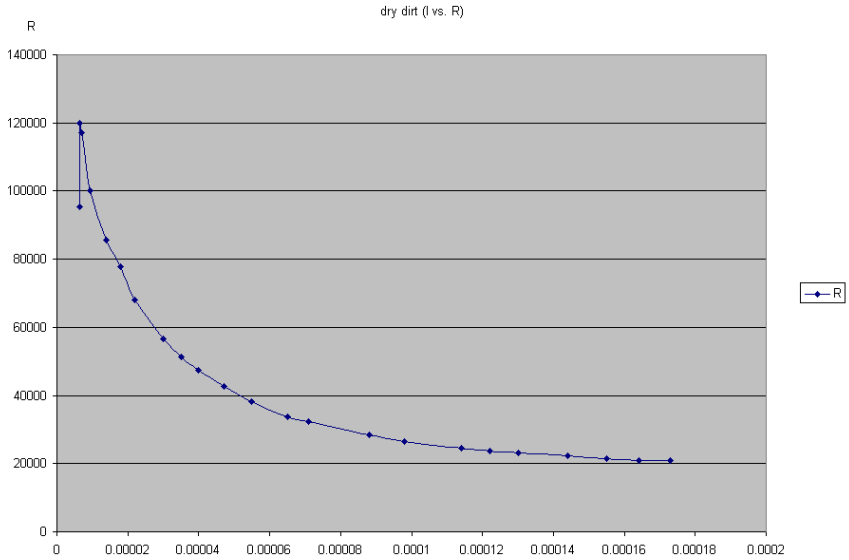


Fig. 11 (Sensor Test for dry dirt)

Vv	Vi	I	R	R=	100000
0.62	0.65	0.0000065	95384.62		
0.78	0.65	0.0000065	120000		
0.82	0.7	0.000007	117142.9		
0.95	0.95	0.0000095	100000		
1.2	1.4	0.000014	85714.29		
1.4	1.8	0.000018	77777.78		
1.5	2.2	0.000022	68181.82		
1.7	3	0.00003	56666.67		
1.8	3.5	0.000035	51428.57		
1.9	4	0.00004	47500		
2	4.7	0.000047	42553.19		
2.1	5.5	0.000055	38181.82		
2.2	6.5	0.000065	33846.15		
2.3	7.1	0.000071	32394.37		
2.5	8.8	0.000088	28409.09		
2.6	9.8	0.000098	26530.61		
2.8	11.4	0.000114	24561.4		
2.9	12.2	0.000122	23770.49		
3	13	0.00013	23076.92		
3.2	14.4	0.000144	22222.22		
3.33	15.5	0.000155	21483.87		
3.45	16.41	0.0001641	21023.77		
3.6	17.3	0.000173	20809.25		





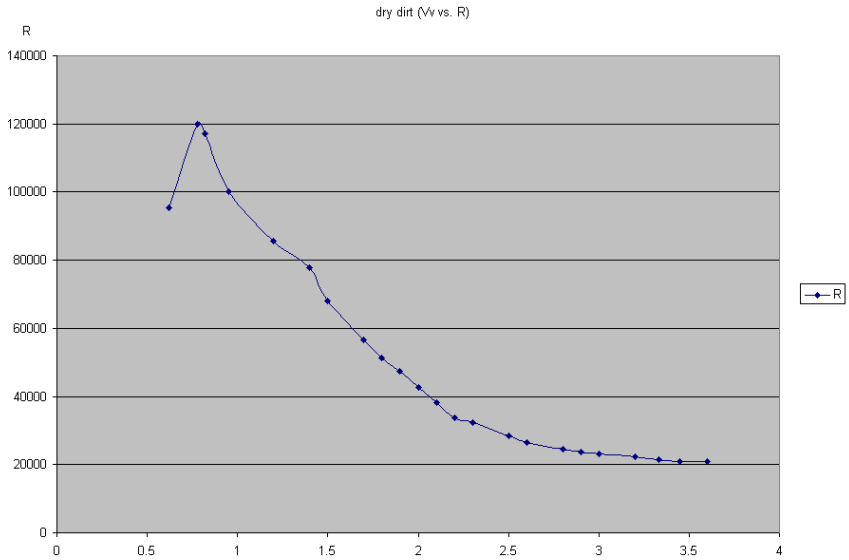
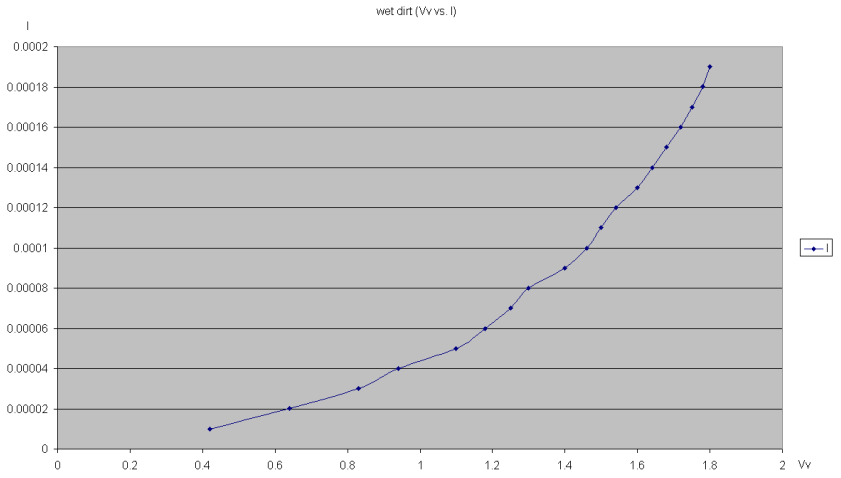
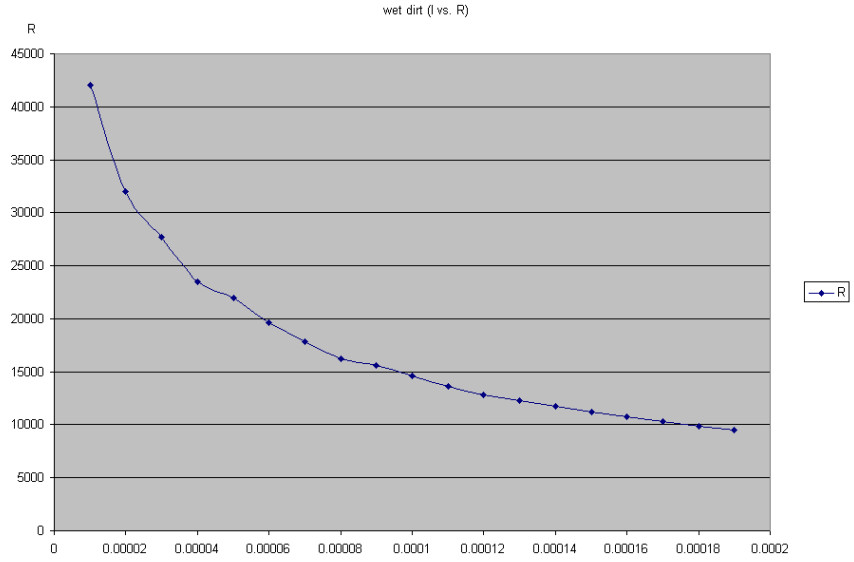


Fig. 12 (Sensor Test for wet dirt)

Vv	Vi	I	R	R=	100000
0.42	1	0.00001	42000		
0.64	2	0.00002	32000		
0.83	3	0.00003	27666.67		
0.94	4	0.00004	23500		
1.1	5	0.00005	22000		
1.18	6	0.00006	19666.67		
1.25	7	0.00007	17857.14		
1.3	8	0.00008	16250		
1.4	9	0.00009	15555.56		
1.46	10	0.0001	14600		
1.5	11	0.00011	13636.36		
1.54	12	0.00012	12833.33		
1.6	13	0.00013	12307.69		
1.64	14	0.00014	11714.29		
1.68	15	0.00015	11200		
1.72	16	0.00016	10750		
1.75	17	0.00017	10294.12		
1.78	18	0.00018	9888.889		
1.8	19	0.00019	9473.684		





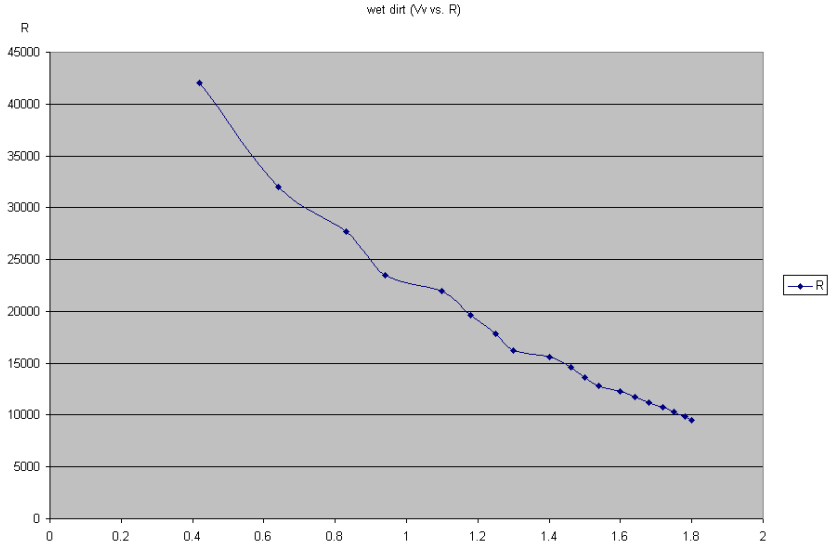


Fig. 13 (Final Test)

sample	resistance (ohms)		Volts
dry (.1mA)	28000	0.0001	2.8
moist (.1mA)	14250	0.0001	1.425
water (.1mA)	7500	0.0001	0.75
dry (.14mA)	22000	0.00014	3.08
moist (.14mA)	11500	0.00014	1.61
water (.14mA)	7000	0.00014	0.98
dry (.18mA)	20000	0.00018	3.6
moist (.18mA)	10000	0.00018	1.8
water (.18mA)	7250	0.00018	1.305

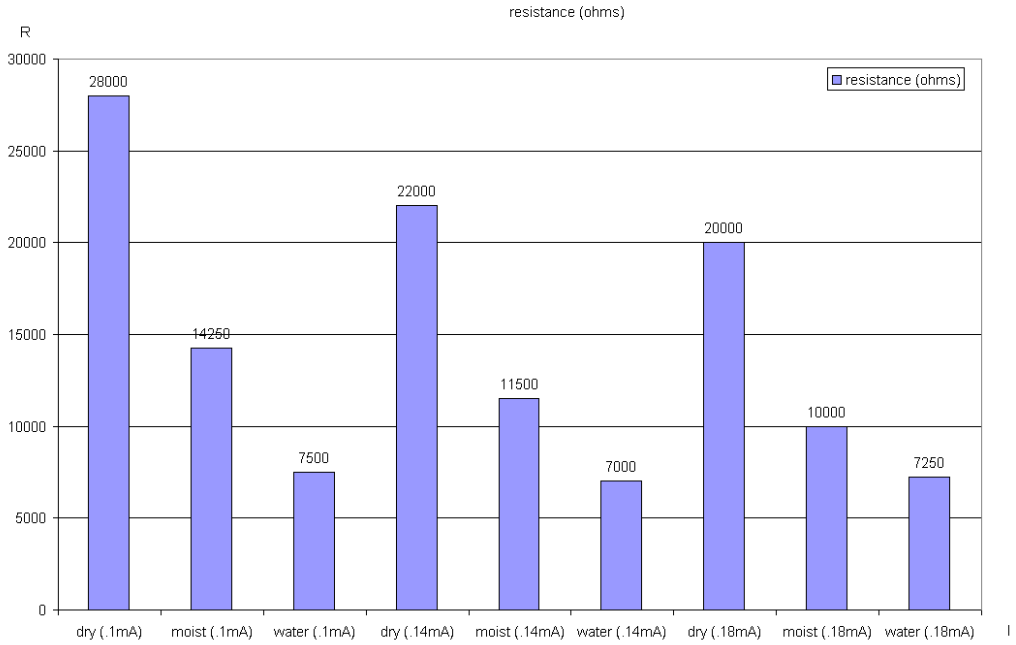


Figure 14

Self-Watering Planter
Budget

Item	**Cost**
Water Pump	$ 3.75
Battery	$ 15.00
Solar Panel	$ 19.00
Shipping	$ 20.00
Plexiglas	$ 12.00
Potting Soil	$ 5.00
Velcro	$ 1.20
Silicon	$ 5.00
Super Glue	$ 3.00
1/8” Hose	$ 1.00
Custom Built HC-12	$ 70.00
Analog Components	Donated/Sampled
Total	**$ 154.95**

	Ni-Cad	Ni-MH	SLA	Li-Ion	Li-Polymer	Reuse Alkaline
Energy Density (Wh/L)	40-60	60-80	30	100	150-200	80
Cell Voltage (V)	1.2V	1.2V	2V	3.7V	2.7V	1.5V
Typical Battery Cost ($)	$2.00 to $11.50	$2.00 to $12.50	$4.50 to $35.00	$1.00 to $9.00	$1.00 to $9.00	$3.00 to $10.00
Cycle Life (charge& discharge)	1500	500	200-500	500-1000	150-200	80

Table 1: The various types of batteries available and their performance/cost characteristics.

device	Volumetric energy density (Wh/L)	Power density (W/L)	number of charge/discharge (cycles)	Discharge time (s)
batteries	50-250	150	$1 - 10^3$	> 1000
capacitors	0.05 – 5	$10^5 - 10^8$	$10^5 - 10^6$	< 1

Table 2: The difference between batteries and capacitors.

Appendix A

Program Code

```
#include "hc12.h"
#include "dbug12.h"

/*** Battery & solar panel voltages are attenuated by .5 ***/
#define TRUE 1
#define V_BATT_HIGH 160         // Battery volt conversion const. of 3.15V out of 3.0V
#define V_BATT_MED 153          // Battery volt conversion const. of 3.0V out of 3.0V
#define V_BATT_LOW 140          // Battery volt conversion const. of 2.75V out of 3.0V
#define V_SOLAR 179     // Solar panel volt conversion const. of 3.5V out of 3.5V
#define SWITCH_POSITION_1 20// Switch select @ 2.0V, water dry
#define SWITCH_POSITION_2 0  // Switch select @ 0.0V, water medium
#define SWITCH_POSITION_3 204           // Switch select @ 4.0V, water wet
#define V_MOISTURE 128  // Voltage level of fully dry soil

void charge_battery();       // Charge battery function
void water_plant();          // Water plant function
void water_routine();        // Function that holds the amount of water to be pumped
void delay(unsigned int num);        // Function that implements a delay for num

/*** Setup the appropriate variables that will be needed for the program ***/
volatile unsigned char s7=0, s6=0, s5=0, s4=0, s3=0;
volatile signed int leave=0;


main()
{
/* Enable bit 6,7 output, PP7 = moisture sensor MOSFET, PP6 = pump MOSFET */
   DDRP = DDRP | 0xC0;
   PORTP = PORTP | 0xC0;

/*** Setup for PWM on Channel 0 ***/
   PWCLK = PWCLK & ~0xF8;      // Choose 8-bit mode(C0), Select N = 0 for Channel 0(38)
   PWCTL = PWCTL & ~0x08;      // Choose left-aligned
   PWPOL = PWPOL | 0x1f;       // Choose positive polarity,Select clock mode 1 for Channel 0
   PWSCAL0 = 0x02;             // Select M = 2 for Channel 0
   PWPER0 = 255;               // Select period of 256 for Channel 0
   PWEN = PWEN | 0x01;         // Enable PWM on Channel 0
   PWDTY0 = 254;               // 100% duty cycle on Channel 0

/*** Setup for A/D conversion on all channels ***/
   ATDCTL2 = ATDCTL2 | 0x80;   // Power up A/D
   ATDCTL4 = 0X01;             // 2MHZ E-clock, 9us conversion
   ATDCTL5 = 0X70;             // 8-bit convert, continuous convert, multichannel

/*** Setup for real time interrupt ***/
   RTICTL = 0x82;              // Set interrupt rate at 2ms
   RTIFLG = 0x80;              // Clear source of interrupt

/*** To use STOP MODE of HC12 ***/
```

```
  INTCR = 0x40;
  _asm(" andcc #$6f");

/*** Enable interrupts ***/
  enable();


/*** Main program to control battery charging & planter watering cycle ***/
while(TRUE)
 {

  DBug12FNP->printf("Start\n\r");

  leave = 0;
  if(s4 < V_BATT_MED && s3 >= V_SOLAR)
   {
    charge_battery();                    // Charge battery with determined level
   }
  else if(s4 >= V_BATT_MED)
   {
    PWDTY0 = 127;                        // 50% duty cycle
    water_plant();              // Jump to water plant routine
   }

  DBug12FNP->printf("      Stop\n\r");
 } _asm(" stop");  // Puts the controller into stop mode to conserve power
}



/*
  Charges the battery using 100% duty cycle
*/
void charge_battery()
{
 while(s4 < V_BATT_MED && s3 >= V_SOLAR)
  {
   PWDTY0 = 254;
  }
}


/*
  Determines switch position & moisture level value,
  then determines if plant needs watering
*/
void water_plant()
{
 PORTP = PORTP & ~0x80;                  // Turns moisture circuit on
 PWDTY0 = 127;              // Charges battery with 50% duty cycle

 while(leave == 0)
 {
  switch(s7)         // Looks at the value of A/D ADR7H
  {
   case(SWITCH_POSITION_1):
```

```
   {
    water_routine();                    // Jump to routine
    leave = 1;                          // Increment leave
   }
   case(SWITCH_POSITION_2):
   {
    water_routine();                    // Jump to routine
    leave = 1;                          // Increment leave
   }
   PORTP = PORTP | 0x40;                // Turns water pump off
  }
   while(s5 > V_MOISTURE)               // Looks at the value of A/D ADR5H
   {
    PWDTY0 = 127;                       // 50% duty cycle
    PORTP = PORTP & ~0x40;              // Pump on
    delay(15000);                 // Wait 15 seconds
   PORTP = PORTP | 0x40;                // Pump off
   }

  PORTP = PORTP | 0xC0;                         // Turns moisture circuit off
  leave = 1;                        // Exits water_plant routine
 }
}



/*
  The routine to turn pump on,
  while measuring soil
  moisture.
*/
void water_routine()
{
 while(s5 > V_MOISTURE)
 {
  PWDTY0 = 127;                         // Charge battery with 50% duty cycle
  PORTP = PORTP & ~0x40;                // Pump on
  delay(5000);                  // 5 second delay
 }
}


/*
  Creates a delay of (num*1ms)
*/
void delay(unsigned int num)
{
 unsigned int i;
 while(num > 0)
 {
  i = 1000;
  while(i > 0)
  {
   i = i - 1;
  }
  num = num - 1;
```

```
 }
}


/*
  Interrupt to take voltage measurements of systems every 2ms
*/
@interrupt void rti_isr(void)
{
 s7 = ADR7H;     //Looks at position 1 & 2 of switch select
 s6 = ADR6H;     //Looks at position 3 of switch select
 s5 = ADR5H;     //Looks at moisture sensor output level
 s4 = ADR4H;     //Looks at battery voltage level
 s3 = ADR3H;     //Looks at solar panel voltage level
 RTIFLG = 0x80;
}


/*
  Interrupt that allows the stop mode to work
*/
@interrupt void irq_isr(void)
{
}
```

REMOTE SHIPBOARD HEAT DETECTION PROTOTYPE

A Thesis

Presented to the Faculty of

The Electrical Engineering Department

Of

New Mexico Tech

By
Scott Dearie
Camden Mullen
Summer Rhodes
Stephen Stange

In partial fulfillment of the requirements

For the course

EE-482 Senior Design Project II

May 1st, 2002

Acknowledgements:

We would like to recognize the following for their contributions to this project:

- **Dr. R. H. Bond, NMT EE Dept.**
- **Dr. D. Hirschfeld, NMT MATE Dept.**
- **Dr. Scott Teare, NMT EE Dept.**
- **Dr. Stephen Bruder, NMT EE Dept.**
- **Dr. Kevin Wedeward, NMT EE Dept.**
- **Dr. Richard Scott, NMT EE Dept.**
- **Dr. Ron Thomas, NMT EE Dept.**
- **Dr. Aly El-Osery, NMT EE Dept.**
- **Scott Savage, Savage Industries**
- **Mark Whitney, Acroname Inc.**
- **Steven Hunyady, Langmuir Labs**
- **Tucson Amateur Radio Association**
- **Jim Sanchez, Solectek Industries**
- **Paul Vinsand, Mini-Circuits**
- **Rita Moore, Mini-Circuits**
- **John Hardy, NMT MATE Dept**
- **Kevin Loveall, TA Mfg.Co.**
- **Rand Poplar, Ferrotel-USA**
- **Maxim-IC Applications Department and Online Services**
- **Steve Wasson, NMT**
- **Thomee Wright, NMT**

Executive Summary

A prototype of a heat detector was designed for the Naval Air Warfare Center (NAVAIR), China Lake, CA. The purpose of the prototype was to provide early detection and warning about the presence of fires and their relative danger to ordnance. This device was to have an indefinite shelf life before activation. Once activated by heat, the device was to transmit, via a radio frequency (RF) signal, an encrypted data set. The data set would provide stored inventory data related to the ordnance, as well as real time temperature readings. This signal would be transmitted through the electronic material casing and the noise produced by the hazardous environment of a jet fuel (JP8) fire to an intelligent fire-fighting receiver station.

The materials chosen for the casing of the prototype had to insulate against the heat of a JP8 fuel fire, withstand flames, and, as stated, allow the signal generated by the internal electronics to pass through this casing. Secondly, this casing had to be lightweight, cost effective and durable. The electronics must be reliably powered during operation. Use of the heat produced by a fire to generate this power was considered, in addition to alternatives.

Several power supply sources were considered. These included a thermopile, or array of thermopiles in a Peltier junction configuration. This could scavenge the needed power from heat. A battery in conjunction with a thermal switch was also considered.

The insulation materials considered were commercial ceramic-based materials. These included dense refractory ceramics, ceramic foams, fibrous materials, adhesives and coating.

A processor was needed that could accommodate stored data and interpret incoming thermal readings provided by a temperature sensor. A transmitter was needed to send this signal to a receiving station.

The final prototype was created using the following parts. The insulation used a ceramic resin material, Fastblock 301, Kirkhill-TA Co.,Valencia, CA, which had a low thermal conductivity of 0.07btu/(ft*hr*°F) at 400°F. The inner layer of insulation was composed of Cerablanket, Thermal Ceramics Co., which had a thermal conductivity of 2.05btu/(ft*hr*°F) at 1800°F. This insulating casing was capable of maintaining an internal temperature near 150°F for thirty minutes when exposed to a continuous flame. Electronic components where found that are rated to operate at temperatures above the 150°F originally required. The electronics employed a K-type thermocouple to read the external temperature, a Brainstem processor, Acroname Co.,Boulder, CO, an op-amp circuit to amplify the signal from the thermocouple and a quadrature transmitter, Maxim, Dallas, TX. A lithium battery, in conjunction with a thermal switch, served as the power supply.

Abstract

A prototype of a heat detector was designed for the Naval Air Warfare Center (NAVAIR), China Lake, CA. The purpose of the prototype was to provide early detection and warning about the presence of fires and their relative danger to ordnance. This device was to have an indefinite shelf life before activation. Once activated by heat, the device transmitted, via a radio frequency (RF) signal, an encrypted data set. The data set provided stored inventory data related to the ordnance, and real time temperature readings. This signal would be transmitted through the material casing, and the noise produced by the hazardous environment of a jet fuel (JP8) fire, to an intelligent fire-fighting receiver station.

The final prototype was created using the following components. The insulation used a ceramic resin material, Fastblock 301, Kirkhill-TA Co., which had a low thermal conductivity of 0.07btu/(ft*hr*°F) at 400°F. The inner layer of insulation was composed of Cerablanket, Thermal Ceramics Co., which had a thermal conductivity of 2.05btu/(ft*hr*°F) at 1800°F. This insulating casing was capable of maintaining an internal temperature near 150°F for nearly thirty minutes when exposed to a continuous flame. Electronic components where found that are rated to operate at temperatures above the 150°F originally required. The electronics employed a K-type thermocouple to read the external temperature, a Brainstem processor, Acroname Co., an op-amp circuit to amplify the signal from the thermocouple and a quadrature transmitter, Maxim/Dallas. A lithium battery, in conjunction with a thermal switch, served as the power supply.

Table of Contents

List of Figures and Tables

Figures

Tables

1.0 Background Information

1.1 History

On July 29, 1967, the USS Forrestal was operating off the coast of North Vietnam. The Forrestal had been conducting combat operations for 4½ days, including a strike early that morning. At 10:52 am, the crew was beginning the second launch cycle of the day when a Zuni rocket from an F-4 Phantom was accidentally launched by stray voltage during the ignition of the Phantom. It soared across the deck hitting a parked and armed A-4 Skyhawk. The impact caused a rupture in the belly of the fuel tank on the Skyhawk, spilling fuel and causing a chain reaction fire of planes parked on the deck. The impact also caused a 1,000-pound bomb to fall off into the spreading fire. Within a minute and a half, the first bomb reached "cook-off[1]" and detonated, killing the flight deck chief and the first wave of firefighters. This initial detonation caused a massive chain reaction of explosions that engulfed half the airwing's aircraft and blew huge holes in the steel flight deck. Fed by jet fuel and ordnance from other aircraft that were armed and ready for the coming strike, the fire spread quickly. Many pilots and support personnel were trapped and killed. Meanwhile, fuel and bombs began spilling into the holes created by the exploding ordinance, spreading the fire further into the ship. The fire burned for thirteen hours before being extinguished.[2]

Once the fires were extinguished, the extent of the devastation was apparent. The most tragic event was the loss of the crew. One hundred thirty four men had lost their

[1] Cook-off: the temperature at which the ordnance detonates because of the thermal energy
[2] USS Forrestal Association, http://users.erols.com/routts/cv59.htm, 2001

lives, while an additional sixty were injured. This disaster remains the single worst loss of life on a navy vessel since the USS Franklin was bombed in WWII, while operating in the Pacific. The total damage was estimated to be $72 million dollars, as calculated in 1968. Following the accident the Navy was assigned the task of developing a remote firefighting system for Navel vessels.[3]

1.2 Customer Requirements

Eric Wilson, Materials Engineer, NAVAIR, China Lake, CA, provided requirements for the final prototype. The prototype must be smaller than a cubic foot, less than 10 pounds, be a single use device that has a shelf life of at least thirty years, and survive in a temperature of 1600°F, while maintaining a temperature of 150°F internally. These requirements are shown in the Table 1.1.

Specification	Requirement	Specification	Requirement
Size:	1 foot cubic	Activation temperature:	250°F
Weight:	less than 10lbs	Battery Life:	5V for 30 minutes
Life cycle (non-fire):	30 years	Shelf life:	30 years
Life cycle (fire):	single use	Frequency:	2.39 –2.45GHz
Max. External temperature:	1600°F	Transmit distance	100 meters
Max. Internal temperature:	150°F	Transmit data:	Live and preprogrammed data

Table 1.1: List of Customer Requirements

These customer requirements were the original goals for the final project. However, these goals were modified as restraints involving time and money were encountered. The modified goals are discussed further in Section 1.3 Project Description.

[3] USS Forrestal Museum, Inc., http://forrestal.org/fidfacts/page13.htm, 2001

1.3 Project Description

This project was a step towards an intelligent firefighting system for ordnance carrying aircraft. This design team was assigned the task of developing one part of this firefighting system. The device, being designed, will be placed within the ordnance on the aircraft and will send a signal to a central firefighting receiving station when a fire is present. It will also send the type of ordnance, inventory information, and the expected cook off temperature.

A heat detection unit was developed to identify temperatures that would be dangerous to shipboard ordnance. This device was to be capable of transmitting an RF signal through the hostile conditions of a JP8 fuel fire. A JP8 fuel fire produces temperatures as high as 1600^{o}F, in which the device must be able to survive for at least thirty minutes. The transmission signal must contain the following information: an ordnance authentication code, the ordnance type, the inventory data, the current temperature, and the heat flux. These requirements are shown below in Table 1.2 below.

Ordnance authentication code
Ordnance type
Inventory data
Current temperature
Heat flux

Table 1.2: Transmission requirements

In order to provide power to a mechanism with a thirty-year shelf life, the suggested power supply was a high temperature Peltier junction device. To ensure the survivability of the electronic components of the device, a highly insulating material must be used as protection.

The long-term goals are to develop a cost-effective device that will be small, durable, and lightweight. It will conform to the ordinance so that it does not affect the aerodynamics of the ordinance. The time and funding constraints imposed a change on this facet of the long-term project. The redefined goals were as follows: Study the feasibility of using a thermopile to provide power for the device. Develop a way to insulate the electronics of this device from the heat of the fire, while maintaining a maximum temperature of 150°F. Select a processor that is able to compile the necessary data and store it in a format that cannot be destroyed over time. Design a transmission system that can communicate at 2.39 – 2.45 GHz. These requirements are shown in Table 1.3 below.

- Study the feasibility of using a thermopile to provide power for the device
- Find a way to insulate the electronics of this device from the heat of the fire
- This insulation had to be designed to keep the temperature of the electronics at or below 150^{o}F
- Select a device capable of processing the necessary data
- This device must also be capable of storing inventory data in memory
- The device was to be capable of retrieving and transmitting the data
- Design this device so it could transmit through the thermal insulation
- The transmission signal must travel a distance of 100 meters
- The transmission signal must be between 2.39-2.45 GHz

Table 1.3: Design requirements

This project was broken up into two parts, an electrical section and a materials section. Both parts were completed simultaneously, with the final goal of integrating the parts together to build one working prototype.

Chapter 2.0 Technical Data

2.1 Power Supply

Basic type-K thermocouples wired together in series were originally considered as a possible power supply. This power supply would obtain power from the fire and have an indefinite shelf life. Research indicated that this would not be feasible as these and other types of thermocouples produce a very small voltage, on the millivolt scale (Omega Corp. 2000). Another consideration for the power supply was to run a Peltier Junction or Thermoelectric Module (TE) in reverse. A TE is a bimetallic or semiconductor module that performs as a heat pump when a voltage is applied across it. When operated in reverse, with a temperature, a voltage is produced. Most commercial TE modules are composed of a Bismuth Telluride semiconductor. Bismuth Telluride operates from below 0^{o}C to about 300^{o}C (www.ferrotec-usa.com, 2000). The second most common TE composition is of Lead Telluride, which operate at 500^{o}C. A second option was to use Thermoionics (TI) which also use the heat of the fire. However, TI is still under development but can be reconsidered in the future. The final decision was to use a long life lithium battery in conjunction with a thermo-mechanical switch to power the device. The thermal switch would be connected to a relay and the battery. When the switch closed, at the appropriate temperature, the relay was activated, keeping the circuit closed if the switch failed.

The major variables involved with a TE are heat flux through the TE, temperature difference (ΔT) between the hot and cold sides, operating temperature, and electrical and heat resistance of the TE. Under ideal conditions a TE can have a high efficiency at approximately 70% at a large ΔT (CRC Handbook, 1995), relatively low

temperatures, less than 200^{o}C, and a good heat sink to draw the heat away from the module are also required (CRC Handbook, 1995). For this project, several requirements were under consideration; the temperature on the outside of the device would rise rapidly to 1000^{o}C, while the internal temperature must stay at or below 150^{o}C. The hot side of the TE would be on the outside edge of the insulation, while the cold side would be imbedded into the insulation. While this would create a very large ΔT in the beginning, the ΔT would drop, as the cold side temperature of the TE would increase due heat flux from the hot side to the cold side of the device. This would cause two problems. First, the TE module would lose efficiency from the low ΔT. Secondly, the overall high temperatures would cause an even greater loss of efficiency and at a higher rate than the first. Essentially, the TE would only be able to produce the required power for a short time before failure. Connecting the TE to a bank of capacitors was considered a solution to this problem. However, a TE only produces a small amount of current making this solution unreasonable.

2.2 Casing

The casing for the prototype protected the internal electrical components from the worst conditions that the device might encounter onboard a navy aircraft, due to a JP8 jet fuel fire. These conditions were assumed to be a maximum temperature of 1600°F, see Appendix I, and a heat flux of 6-8 BTU/ft^{2}s (ares.ame.arizona.edu/materials/ysz.html, 2000). The true worst-case heat flux value was not available and could not be determined by our design team. In general, the heat flux from a fire has been found to be variable. Estimates of the variability of heat flux in a fire was quantified by NIST, and with more information on the

conditions of a fire the variability in heat flux may be determined using empirical equations found by NIST research (ares.ame.arizona.edu/materials/ysz.html, 2000).

Because the harshest conditions the prototype was designed to meet are thermal, the prototype casing must be composed of a thermally insulating material. Under the conditions outlined above, the insulation was to be able to protect the internal electronics for at least 30 minutes. In order to protect these components, the internal temperature of the prototype was to not exceed 150°F, see Appendix I.

The ideal characteristics for the insulating material are that the material has a minimal thermal conductivity and minimal density in order to minimize weight and size. Additionally, corrosion, UV, and weather resistance, as well as brittleness and strength were considered. All of these factors must be considered to chose the most favorable housing for the device, so that it will be durable.

The requirements for the insulation to meet were quite stringent when one considers that space shuttles, upon reentry into the atmosphere, must insulate against 2000°F for 9 minutes with an internal temperature not to exceed 350°F (www.spaceshuttletille.com/faq.html, 2001). For this reason, when considering insulating materials, both commercially available and non-commercial available insulating materials were considered.

Insulation materials must protect against a flux of heat. Heat transfer can occur by three basic means: conduction, convection, and radiation. Conduction is defined to occur when energized collide and transfer energy through a material or space. Convection is found as the transfer of heat by mixing. Finally, radiation is the

transfer of energy by electromagnetic waves. Heat flux by convection could be neglected for the given conditions. While radiation in a fire can be appreciable, and is treated by the NIST article on heat flux in fire (Womeldorf International Conference, 2001), this also was neglected for the purposes of this project. By considering conduction to be the method of heat transfer, Fourier's law, Q=k*A*dT/dx (Geankoplis, 1993) can be applied where Q is the heat flow, A is the cross-sectional area, dT/dx the change in temperature change in thickness, and k is the thermal conductivity.

This general equation was applied to non-steady state conditions, as it exists for the conditions of this project. Calculations using heat transportation equations are shown in Appendix II. These were used to determine size requirements for insulation casing.

Another class of material considered was silica ceramics. Silica has a low thermal conductivity. Very high purity silica is used to make the space shuttle tiles (www.spaceshuttletile.com/faq.html, 2001). While shuttle tiles are far too expensive for the price range requirements of this project, silica materials were considered for their good thermal insulation, see Table 2.1. Additionally, zirconia ceramics have been considered as they too have very good thermal insulation.

Foam and blanket materials, made from very porous or fibrous ceramic, were also considered. These take advantage, not only of the good thermal insulation of the ceramics they were composed of, but of air, which has an even better insulation and much lower density than ceramics.

Information on insulation materials that were researched was listed in Table 2.1. The materials are arranged from highest thermal conductivity to lowest. It should be noted that there are general trends found in the density and cost columns. With decreasing thermal conductivity, there was generally an increase in cost, i.e. the better the insulation the more expensive it was. Additionally, density generally decreased with decreasing thermal conductivity. This may be attributed to the fact that the large volume of air in porous materials both reduces the thermal conductivity and the density.

Product	Density (lb/ft^3)	Conductivity (btu/ft-hr-°F)	Cost ($/lb)
Cotronics Liquid[3] Silica	110	4	5.50
Thermal Ceramics Cerablanket[4]	8	2.05 at 1800°F	1.5 *
Yttria Stabilized Zirconia[13]	368	1.15	_____
Cotronics Rescor[3] Foam	40	1 at 2300°F	6.50
Fastblock 301[5]	39.9-44.9	0.07 at 400°F	40 *
Zirpor 1[14]	18.7 – 21.9	0.042 at 1472°F	21
Fastblock 800[5]	20.0	0.04 at 78°F	____
MicroTherm Moulded Form[15]	21.2-25.0	0.22 at 800°F	18
Space Shuttle Tiles[1]	9	.028 BTU/ft-hr-°F at 70°F	2200

Table 2.1: Materials Considered (* =bulk price)

The information obtained from research into these materials was used in choosing materials to test and the tests to be completed.

2.3 Temperature Sensor

A thermocouple is a device that can be used to measure temperature. It is composed of two metals that when attached at one end, produce a small voltage on the order of millivolts. There are many types of thermocouples composed of different metal and alloy combinations. These different types of thermocouples are effective over different temperature ranges.

A K-type thermocouple was found to be best for the requirements of this project, which required a temperature range of 250^{o}F to 1600^{o}F. A K-type thermocouple is rated up to 1800^{o}F. It is composed of Chromega and Alomega. For the purpose of this project, the end wires were welded together and left as an exposed junction. A K-type thermocouple is a general-purpose thermocouple, which is very dependable in the necessary range.

2.4 Central Processor

The project required a central processor to compile the data obtained from the temperature sensor and transform it into a useable form for transmission. This processor needed to have the ability to change a voltage into a binary stream, transmit and receive digital numbers, and compute basic math functions. These are the minimum requirements necessary for this project to meet specification.

It was decided to use a Programmable Integrated Circuit (PIC) from Acroname Incorporated, known as *The Brainstem,* to implement the needed functionality.[4] The

[4] See section 3.4.1 for selection criteria used in selection of the PIC

Brainstem is an integrated controller running of a 40 Mhz RISC[5] processor that has five ten-bit Analog-to-Digital Converters (A/D), five general purpose digital Input / Output (I/O) ports, and 10KB of EEPROM. EEPROM is on board memory that can only be erased electronically. This ensures that programs can be stored and they may be called at a later time.

2.5 Transmitter

The navy requires that shipboard transmission frequency be between 2.39 and 2.45 GHz. This is known as the UHF Band, or more specifically the S-Band. Tests preformed by the NAVAIR prior to the beginning of this project showed that JP8 fuel fires have little or no effect on signals transmitted at the specified frequency. This is stated in Appendix I.

This project required no receiver to be built. This limited the amount knowledge that can be provided regarding the transmitter. The power density required by a receiver remains unknown. There is also information needed about signal propagation onboard ship. Knowing this information and assuming line of sight, one can use Friis' Equation:

$$P_r = P_t G_t G_r (\lambda/[4\pi d])^2$$

The required power density of the signal can be calculated with this extra information.

[5] RISC: reduced Instruction Set Computer

3.0 Original Developments

3.1 Power Supply Research, Performed by Camden Mullen

Extensive research into the use of TE's was performed to determine the best TE module configuration. Experiments with a bismuth telluride module indicated that this type of module would not be suitable for a power supply. The module used had solder that melted at too low of a temperature. Research into the use of a Thermionic device suggested that this type of device would be more suitable. Dr. Harold Brown and Dr.Yan Kucherov of Eneco, Inc, Salt Lake City, UT were contacted regarding Eneco's "thermal diodes" which are solid state Thermoionic devices for converting heat-to-electricity. These devices are still in the development stages and were not available. A third consideration was the use of lead telluride TE generation modules. These produce more power and continue to work at a higher temperature than the bismuth telluride modules. Dan Allen at Hi-Z Corp, San Diego, CA, was contacted about the possibility of Hi-Z lending or selling at a reduced cost, one of their power generation modules. HI-Z declined but did send technical information on their products.

The requirement of using the heat from the fire to power the device was not met. In place of the TE device, a 5V Lithium battery was selected to supply the required power. This is a commercially available power source that has a shelf life of 30 years. A thermal-mechanical switch was connected to a relay to activate the entire device. Several small bimetallic switches were ordered from Allelectronics Corp, Van Nyes, CA, for integration and testing. These close at 212^{o}F (100^{o}C). When

imbedded under a small amount of insulation, the switches close and activate the unit at the activation temperature of 250^{o}F.

3.2 Insulation Material, Performed by Summer Rhodes

Research into standard thermal testing was completed so that thermal properties of materials could be determined. This could be used to determine these properties for materials where they were not available from the manufacturer. Several tests have been devised that adapt standard thermal conduction/insulation tests to the available equipment in the department (Speyer, 1994). ASTM standards test E1225 was to be the principle thermal conductivity test. This test method describes steady state techniques for thermal conductivity, λ, where the values are between 0.2-200 W/mK up to 1300K, or 1880^{o}F (ASTM Standards, 2000). Such testing was not needed, however, because thermal conductivity data was found for all materials under consideration. The tests may be used to verify thermal conductivity values given in the literature.

Using data for the insulating materials, modeling was conducted using principles of unsteady-state heat transfer (Cotronics Corp., 2000). The insulation was modeled to be a hollow hemisphere. The dimensions of the electronic components are 6.5cmX6.5cmX7cm, which, for simplicity, was assumed to be a hemisphere with radius of 4cm. It was determined what was the minimum thickness required to insulate in the worst case scenario, as described in Section 2.2, was determined. Table 3.1 lists the thickness required for each available material which can serve as the main thermal insulation.

Material	Thickness (m) (Appendix X)
Cotronics Liquid Silica 2700°F	2.0
Fastblock 301 castable ceramic	0.07
Fastblock 301 castable ceramic	0.21

Table 3.1: Material thickness needed to meet thermal specifications

As seen in Table 3.1, the Cotronics Liquid Silica would not be feasible because of the thickness required. Fastblock 301 is a more viable option because of its lower thermal conductivity.

Thermal and flame tests were conducted on the materials available for testing. These were conducted to test for failures such as melting, cracking or ignition and the resultsof flame tests are found in Table 3.2. Cerablanket, Sauereisen adhesive and Cotronics fused silica are ceramic materials. As expected there was no noted change in the materials when subjected to heat or flame. Pictures of these tests can be seen in Appendix VII.

Material	**Ignition time**	**Notes**
Cerablanket	N/A	Became less pliable following test
Sauereisen Adhesive	N/A	Thermal shocking on cooling caused fracture; brittle after heating
Fastblock 301	25 sec	Cracked; ignited; material expanded
Cotronics Silica	N/A	No noted changes in material

Table 3.2: Observations from flame testing

While Fastblock will not hold its shape during exposure to flame and high temperature, its low thermal conductivity still made it the most viable material for design. Sauerisen was also chosen to seal Fastblock 301 parts together. It was used, despite its thermal shocking, because testing showed that the adhesive did not thermally shock and rupture a junction during a thermal test as seen in Appendix V.

Figure 3.1: Setup for flame test

The internal temperature was plotted during the thermal test for the above insulation, where a propane torch was used to simulate fuel fire conditions. The plot is found in Figure 3.2.

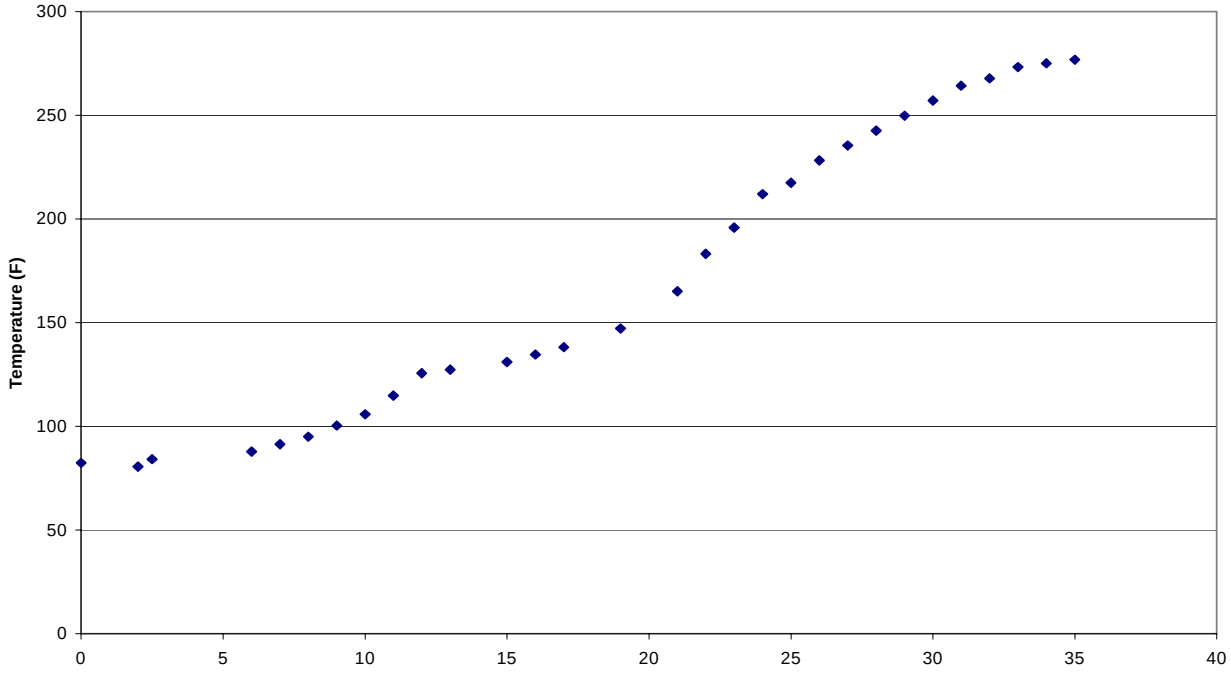


Figure 3.2: Thermal test for first insulation casing using Fastblock 301

The specified maximum internal temperature was 150°F that was reached 20 minutes into the test. While the temperature should not have reached this temperature until 30 minutes into the test, the results were promising, and changes to the design could be made in order to meet the specifications.

Using the results of the first prototype tested, a second test was designed. This prototype was designed with consideration for the electrical components that had to fit within the casing. The design was again a hemisphere as stated previously. The setup used for this test can be seen in Figure 3.1. Due to materials limitations the spherical section of the design was 1 inch thick while the bottom of the casing was 0.8 inches thick. Again, a thermocouple was fed into the center of the prototype and Cerablanket insulation was placed inside the sphere. The use of the Cerablanket was cost effective and drastically reduced the weight of the overall insulation, see Table 3.1. The two Fastblock 301 components were sealed while wet so that it would form its own seal. While Sauereisen had worked previously, its brittle nature did not provide a durable device casing. Sauereisen was used to seal the hole that the thermocouple was inserted into. Two thermocouples were attached, one to measure the internal temperature and one the external temperature. The internal temperature reading was fed to a computer where the data was plotted using Omega temperature acquisition software. The plots of the external and internal temperature versus time and of internal temperature versus time are found in Figures 3.3 and 3.4 respectively.

Temperature readings were taken every twenty seconds as seen in Figure 3.3. As can be expected from a fire, the external temperature had a large variability, however

it remained close to 800°C or 1472°F. The internal temperature gradually increased. The internal temperature can be seen in Figure 3.4.

There was an increase in temperature, as expected during the test. The temperature at thirty minutes was 85°C or 184.6°F, close to the specified temperature of 150°F. It was determined that this prototype was feasible, and it could be modified to fully meet the specifications.

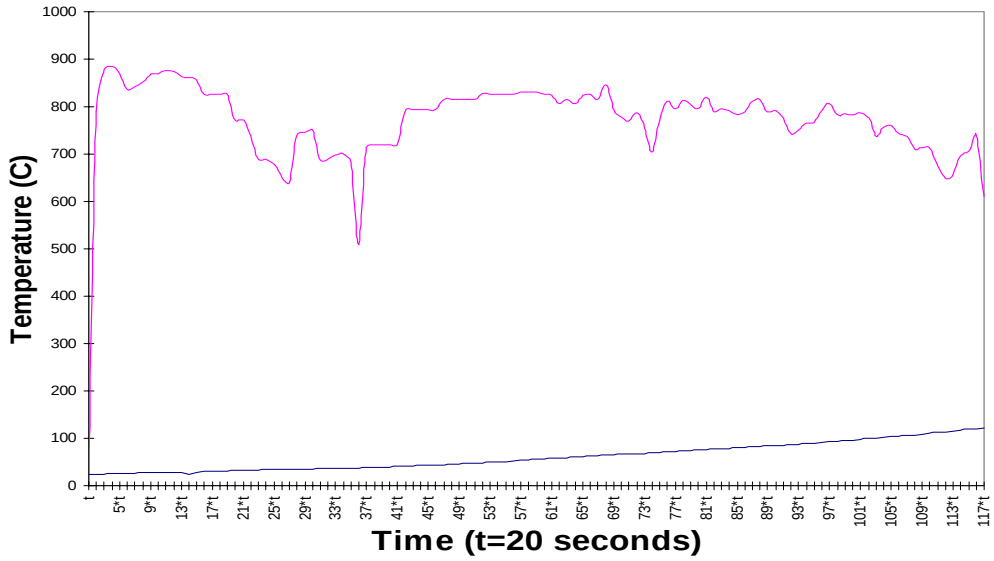


Figure 3.3: External temperature is plotted above and internal temperature below.

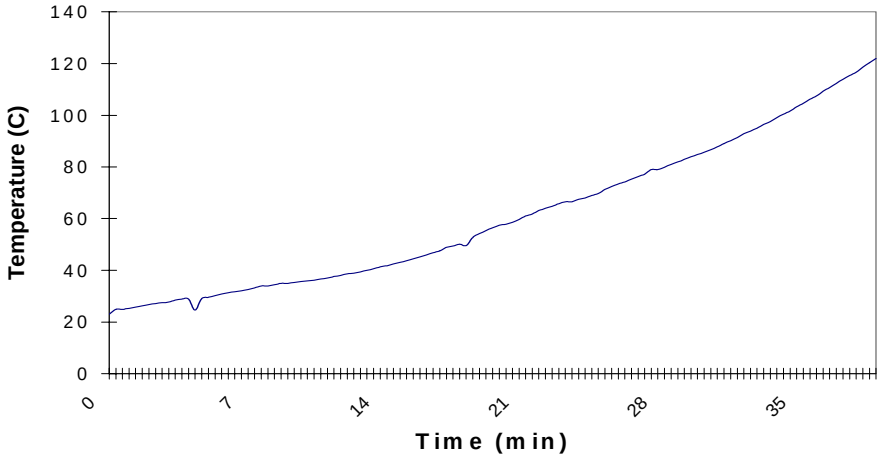


Figure 3.4: Internal temperature plot for insulation prototype

3.3 Temperature Sensor, by Stephen Stange

This project required the temperature sensor to be small, inexpensive, lightweight, and low power. The sensor also had to be able to reliably read temperatures in the full range of the customer requirements, 250°F to 1600°F, see Section 1.2. To meet these requirements, several types of temperature sensors were considered. The data that these temperature sensors output had to be transformed into a signal that could be read by the data processor inside the device.

3.3.1 Evaluation of Temperature Sensors

Several different types of temperature sensors were considered for this project. The first option considered was a thermocouple. Type "K" thermocouples, described in Section 2.3, have a temperature range that can accommodate the requirements of this project (250^{o}F to 1600^{o}F). Another option considered was a RTD, resistance temperature device. Also considered for this project were infrared radiators and thermistors. The project requirements were thoroughly considered, while trying to choose a viable option for the temperature sensor.

One option for the temperature sensor that was considered was to use type-K thermocouples, like the ones originally intended for the Peltier junction power supply. A thermocouple "consists of two dissimilar metals joined together at one end, known as a junction, that produces a small thermoelectric voltage when the junction is heated" (Omega Engineering, Inc., 2000). A thermocouple thermometer measures the electromotive force (emf) between two different materials and translates it into a temperature.

Resistance temperature devices, or RTD's, work, because as temperature changes, the resistance of the RTD material changes. Resistance of an RTD rises relatively linearly with temperature. RTD's have a moderately wide temperature range, from –270^{o}C to 850^{o}C. RTD's were the most stable kind of sensors considered. They have specifications to be off by only 0.2^{o}C after 10,000 hours at the maximum temperature. The main problem with using an RTD for this project is the tendency of RTD's to self-heat. This not only leads to inaccuracies, but also the internal circuit for this project needs to be kept at a temperature of 150^{o}F. Therefore, any internal heat due to self-heating would be damaging.

Infrared radiators work by measuring the thermal radiation of an element. These were strongly considered because they do not require physical contact and are good at measuring large surfaces. The long-range goal of this project is to have a disposable product that will only be used in a fire once. The price of such sensors, that would meet the temperature requirements of this project, was excessive.

Thermistors are a specific type of RTD. Thermistors work because a change in temperature, which causes a resistance change in a ceramic semiconductor. The resistance drops nonlinearly with temperature rise. These devices are very accurate. The limited temperature range, typically between –40^{o}C and 150^{o}C, was not enough for the required temperature range of this project.

It was necessary to choose between these of temperature sensors to find the one that best suits the needs of this project. Table 3.3 lists the pros and cons of the different types of temperature sensors considered.

Device	PROS	CONS
Thermocouples	Wide temperature range; Inexpensive; Rugged	Not as accurate as others; Not as stable as others
RTD's	Stable; Accurate; Fairly wide temperature range	Not as rugged as other sensors; Subject to self-heating
Infrared Radiators	Good at measuring large surfaces; Do not require physical contact; Range includes high temperatures	Not as accurate as other sensors; Sensitive to surface radiation; Expensive
Thermistors	Very accurate	Limited temperature range; Subject to self-heating; Highly nonlinear

Table 3.3: Comparison of Temperature Sensors

Using the information contained in Table 3.3, the final choice was to use a K-type thermocouple. The thermocouple had a temperature range that best met the needs of this project. When this consideration was combined with cost effectiveness, K-type thermocouples were the only choice. Originally, there was concern that the large temperature range of a thermocouple, typically between -270^{o}C and 2300^{o}C, would limit the resolution that could be provided to the internal electronics. The customer for this project determined that the resolution of the temperature was not as important as being able to use the full range of the original project goals and design was done accordingly.

3.2.2 Temperature Sensor Signal Conversion

Data acquisition systems that convert the output of a thermocouple into a temperature are commercially available from companies such as Omega. These units take the millivolt output of a thermocouple and use a conversion factor to change this into a temperature. However, these units would not meet the requirements of this project because they cost between $200 and $2000. This cost makes these products unreasonable for this project. The conclusion drawn from this information meant that

a circuit had to be developed to translate the millivolt output of a K-type thermocouple into a voltage range that could be measured by the electronic processor.

The first step of this process was to determine the output voltage range of the thermocouple when it is exposed to the full temperature range of this project, 250^{o}F to 1600^{o}F. A K-type thermocouple was acquired for this project from the New Mexico Tech Instrument Room. Thermal testing was preformed on the K-type thermocouple to determine the voltage range it put out when exposed to the extremes of the required temperature range. These tests involved using a torch to heat one end of the thermocouple while the temperature and the voltage at the output end where measured. It was determined to yield a maximum output of 60mV (0.06 Volts).

The next step was to use the output range of the K-type thermocouple and convert it to a voltage signal that the internal electronic processor could use. Calculations to determine the gain can be seen in Appendix VI. This gain was determined to be 67.

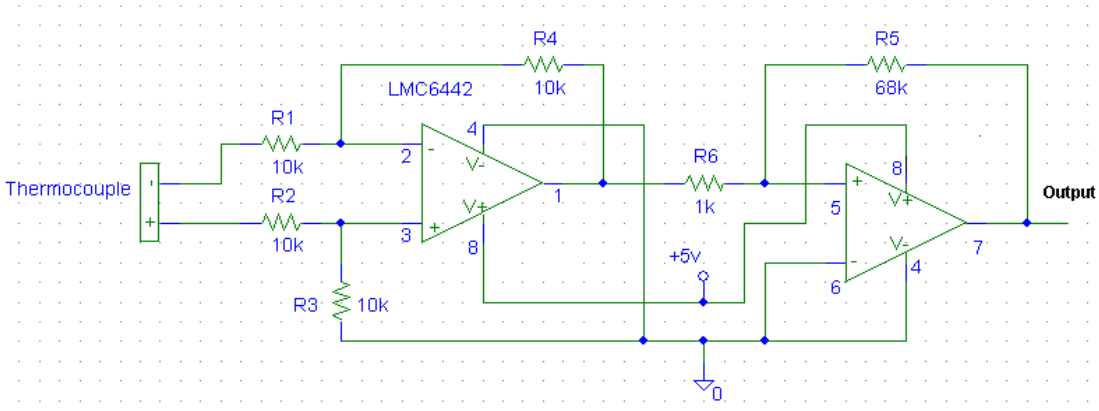


Figure 3.5: Schematic of Temperature Sensor Circuit

The next step was to implement the circuit. The circuit was designed using a two-stage op-amp circuit. The first stage was a difference circuit, which took the difference between two signals. The two signals were provided by the two different metals in the thermocouple, as shown in the schematic in Figure 3.5. The second stage was a non-inverting amplifier with a gain of 67.

The requirements of this project limited the choice of operational amplifiers, or op-amps, that could be implemented. The source voltages in the device were limited to ground and 5 volts. This meant that a single source op-amp had to be used. There were also limitations due to the nature of the signal that was being amplified. The thermocouple output was on the order of millivolts, which a Rail-to-Rail op-amp had to be used to accommodate this. A Rail-to-Rail op-amp is an op-amp that can have input signal voltages that are equal to the source voltages. The model LMC6442 op-amps were donated by National Semiconductor.

All of the components used in this circuit are rated to work at temperatures up to 85^{o}C. This temperature is equal to182^{o}F, which exceeds the internal temperature of 150^{o}F, required by the original proposal, see Appendix I.

3.4 Central Processor, by Scott Dearie

The central processor was implemented to transform the voltages sent for the temperature sensor and change it into a useful form. The data was transformed by the processor into a binary stream that was sent to the transmitter. The transmission protocol was an asynchronous stream that had to automatically synchronize the transmitter and the receiver station.

3.4.1 Selection of the Microprocessor

It was determined than any processor used in this project must meet a specific set of requirements. These requirements included that the processor have a nominal current of 100 mA, a start-up current of less than 300 mA, and an overall voltage of less than 5 volts. These values were chosen such that the device would be low power. It was also necessary that it have minimal noise impact on the transmission form the transmitter. It was also determined that the PIC must have at least 2 A/D ports, 5 digital I/O ports, and 128 KB of EEPROM. These specifications were selected to ensure that all the necessary functions of receiving and transmitting data could be achieved. In addition to the power and electrical requirements, it was determined that the device should be pre-mounted or non-surface-mounted, and be able to execute programs on power up. The requirements are summarized in Table 3.4 below.

Nominal current:	<100 mA
Start up current:	<300 mA
Voltage requirement:	5V
Ports:	2A/D, 5 IO
Memory:	128KB of EEPROM
Characteristics:	non-surface-mount or pre-mounted Executes on power up

Table 3.4: Minimum requirements for central processor

Three processors considered met these requirements, the *Brainstem* form Acroname Inc., the OOPIC from Savage Innovations, and the 1800 series PIC from Microchip. Samples of the *Brainstem* and the OOPIC were received and tested. The 1800 series PIC was not tested due to the fact it is not pre-mounted, and would require extensive modification to become a viable solution. It was determined that if the OOPIC or the Brainstem worked, more time could be devoted to learning how to operate the PIC rather than trying to modify it.

Tests on the *Brainstem* and the *OOPIC* revealed that the two processors were substantially different, even though they both met the design requirements. The *Brainstem* is able to run on a variety of platforms that include Windows, Linux, Unix, Windows CE, and Palm operating systems. The OOPIC must be run on Windows 98. The OOPIC is programmed in Object Oriented ANSI C whereas the *Brainstem* is programmed using TEA[6] packets in a C style language[7]. The *Brainstem* runs at 75mA compared to 300 mA on the OOPIC. The OOPIC has 24 digital I/O ports and 10 6 bit A/D ports contrasted by the 5 digital I/O ports and 5 10 bit A/D ports on the *Brainstem*. In addition to these shared characteristics, each device has its own unique features. The OOPIC has an onboard prototype area that allows for expansion and also has developed objects to simplify the programming process. The *Brainstem* is able to run four programs concurrently, a small package size, and a very active customer support system. These benefits and detriments are listed in Table 3.5.

[6] TEA: (Tiny Embedded application) small programs that are used concurrently to minimize program size
[7] C style….: see Appendix VII

Device	Pro's	Con's
OOPIC	Object Oriented	Must use WIN 98
	ANSI standard	High Power Requirement
	Prototype Area	No User Support
BrainStem	Universal Platform	Non ANSI C
	10 Bit A/D	No Prototype Area
	Active User Support	Limited Digital Ports
	Low Power Requirement	
	Able to Run Programs Concurrently	

Table 3.5: Comparison of Processors

The Brainstem was selected for the project due to its universal developmental platform, low power requirements, and high level of customer support. These factors made this device stand out from the OOPIC because they were determined to be vital to the overall success of the project. In testing, it was determined that the processor can be programmed with any machine ranging from Windows XP to Linux or Palm to Windows CE. This provided a preferred developmental platform for testing and loading code. With the *OOPIC,* one must use Windows 98 as the developmental and launch platform. The overall theme of our project is a low power design and this product draws 1/3 the power of the *OOPIC.* This is crucial because any power that can be saved does not have to be created form the fire. Finally, the *BrainStem* has the backing of full customer support from the manufacturer. The *OOPIC* is a side project of a very small company and does not provide any customer support.

3.4.2 Data acquisition, transmission, and reception

The overall data acquisition system is broken down into two parts; the transmitter and the receiver. The transmitter takes the data from the temperature

sensor and them transmits a raw A/D value to the receiver. In conjunction to transmitting the A/D, value the transmitter will send a value that indicates the type of ordnance that is on fire. This is done by using the data space that corresponds to the temperatures below the activation temperature.[8] The transmitter also sends a 2 bit number to the receiver that indicates the next frequency of transmission.

The receiver obtains the value from the transmitter, converts it to Celsius, and compares it to the previous value; these values are displayed to a console window. In addition to displaying the temperatures, the receiver will indicate what type of ordnance is on fire every 5th sample and indicate if the external temperature exceeds the cook-off temperature for that ordnance. The method of loading programs onto the *Brainstem* is found in Appendix VII.

3.4.2.1 Transmission protocol

The onboard A/D of the *Brainstem* is used for data acquisition. After executing the necessary setup commands, the device reads a value from the A/D port as a 10-bit number at a set sampling rate. After the value has been acquired, the processor cues the receiver by sending out a series of 10 1's followed by a zero, then a value every clock cycle. The cue of 10 ones is used because this pattern will never occur in the transmitted data. After the cue has been sent, the processor sends the value from the A/D bit by bit least significant bit first. This continues every clock cycle until the all ten bits have been sent. After the tenth bit is sent, the processors send zeros every clock cycle until the new sample is

[8] data space…. Activation temperature: approximately the numbers from 0 to 75 allowing for 75 different ordnances to be used.

taken. The simple program structure is shown in the flow chart in Figure 3.6 below. The code for the transmitter can be found in Appendix VIII.

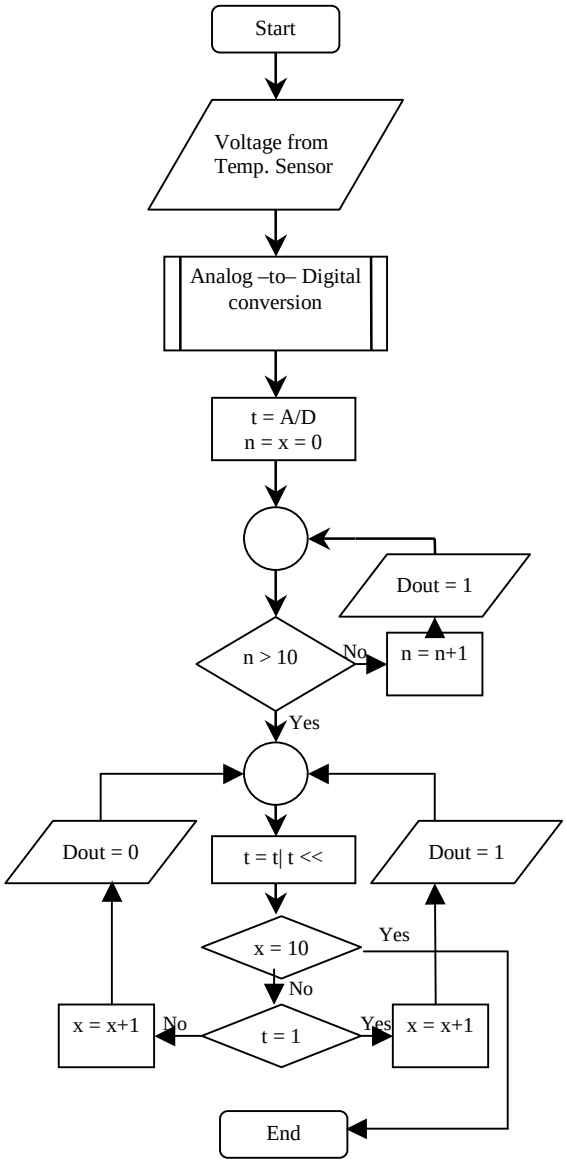


Figure 3.6: Basic flow chart for data acquisition and transmission

3.4.2.2 Receiver protocol

The receiver side of the system was more complicated. This was attributed to the fact that the receiver and the transmitter did not share a clock or a method of synchronizing off each other. To accommodate for this, the receiver synchronized with the transmitter. The receiver remains in an idle state until it received a cue from the transmitter. The cue was seen as a series of 9 ones instead of a series of 10 ones, witch was the standard protocol. This was done to ensure that the receiver had to wait for the zero bit that was transmitted at the end of the cue. After the receiver had received 9 ones, it waited for a zero to be transmitted. When the zero was received it waits for one and a half clock cycles and then received the 10 bits from the transmitter. The timing of the receiver was shown in Figure 3.7 below.

After the receiver had received the data stream, it performed several

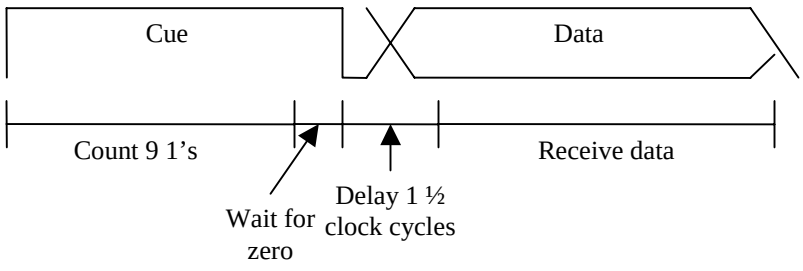


Figure 3.7: Timing diagram for the receiver

checks on the number. If the number is between 0 and 75 it was recognized as the inventory data for the ordnance and the ordnance data is displayed on the console. If the data is greater than 75 it was interpreted as a temperature and was multiplied by a scaling factor to convert it to Celsius, and it was compared with the previous data value. The current temperature and change in temperature was

displayed on the console. The current temperature is also compared to the cook-off temperature of the ordnance, if it was within a specified margin, a warning message was sent.

The choice was made to calculate the actual temperature on the receiver side instead of computing it on the transmitter. This was done to minimize the amount of data sent through the fire. It was clear that the smaller the packet sent the better, and there is a less of a chance that the fire would corrupt it. Rather than sending the inventory data, it was decided to store the inventory data on the receiver side. This had two benefits; it freed the transmitter to send more data and it kept all secure data from being transmitted. If the transmission were intercepted, the only thing received would be a number between 0 and 75 or the possibility of an abstract number. This was a more secure method that allowed for the random generation ordnance code numbers. The code for the receiver is in Appendix IX. The following is the flowchart for the receiver. In Figure 3.8 below, is a sample of what is displayed to the screen when the receiver was active.

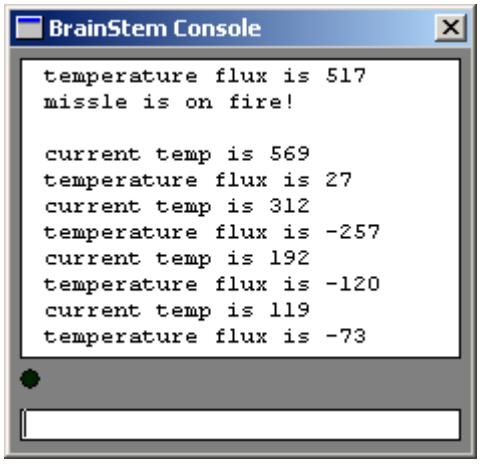


Figure 3.8: Sample of receiver display

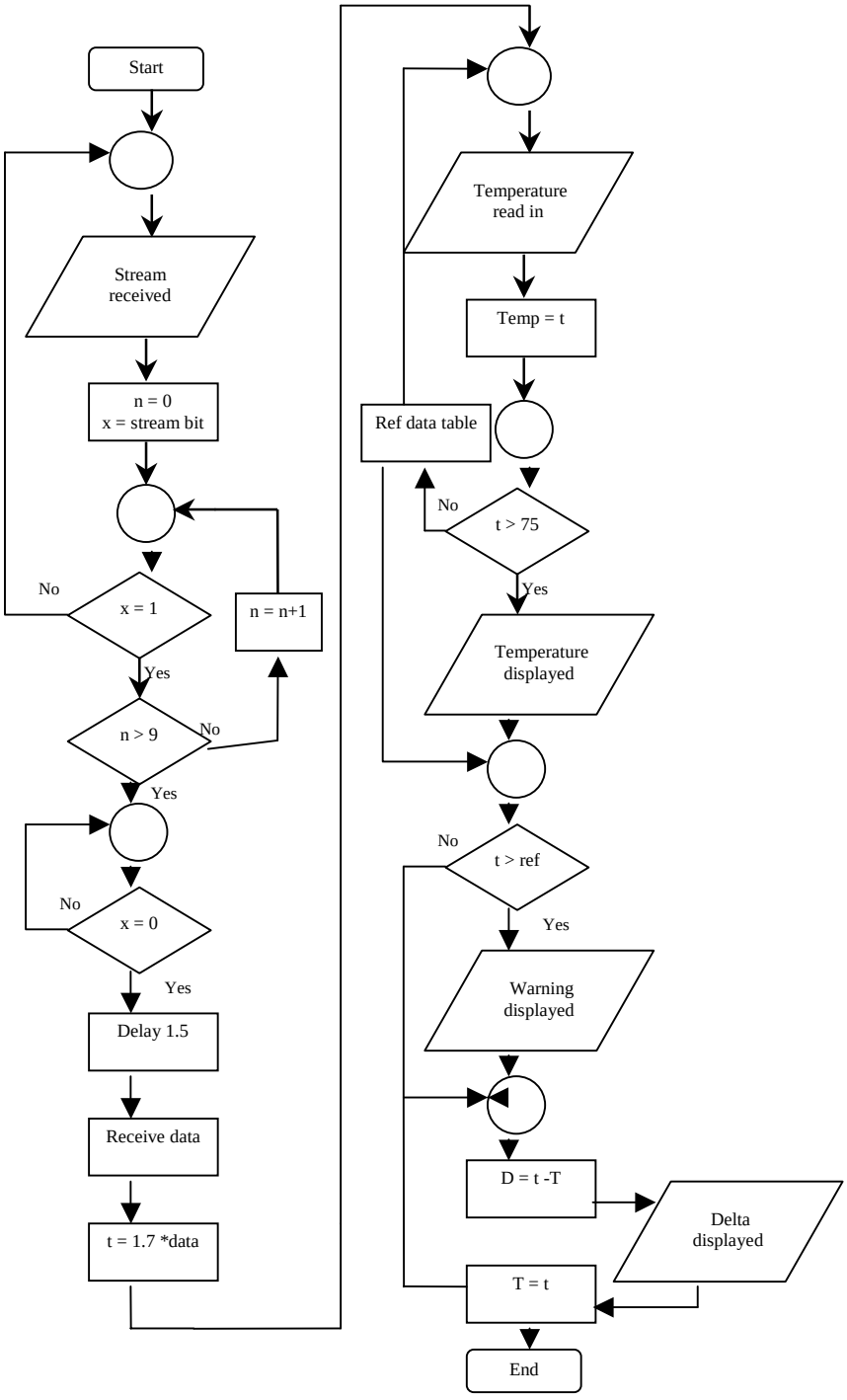


Figure 3.9: Flow chart for receiver

3.5 Transmitter Development, by Stephen Stange

The transmitter for this project was required to be low power, small, inexpensive, and lightweight. It was also needed to transmit the signal at a frequency of 2.39 to 2.45GHz. A block diagram demonstrating the manor in which the transmitter components were configured is shown in Figure 3.10.

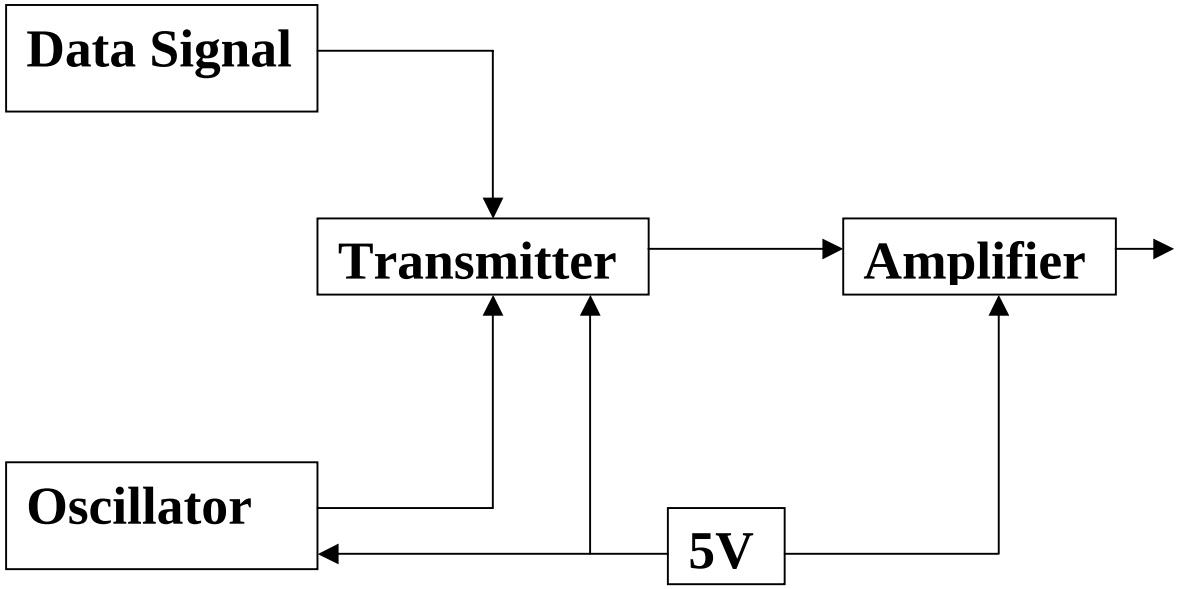


Figure 3.10: Block Diagram for Transmitter Circuit

3.5.1 Transmitter Design

Maxim/Dallas, Dallas, TX, provided a quadrature transmitter for this project. The transmitter was a prepackaged chip, the MAX2721. This chip included a LO Doubler, local oscillator doubler (this means that the oscillator used to tune the frequency of the transmitter needed to oscillate at half the desired frequency) and an integrated I/Q modulator, see Section 3.4.2.1 for transmission protocol. One of the main advantages of this chip is that a low input voltage is required, where typical operation requires 3V. Another advantage of using this transmitter was

that has been tested to work at temperatures up to 85°C with only a 1.5dB loss. To ensure that the required distance of 100 yards line of site was met an amplifier was added into the final transmitter package.

The oscillator used for this project was provided by Mini-Circuits, Brooklyn, NY. The oscillator was a surface mount voltage controlled oscillator, model number ROS-1121V. Since, this oscillator was voltage controlled it could be tuned to half of the desired frequency. One advantage of this oscillator was that it was able to function at temperatures up to 85°C. Another advantage of this packaged oscillator was its power supply requirement of only 5V.

The third main component of the transmitter was the amplifier. The amplifier used was also provided by Mini-Circuits, it is model number GALI-6F. This amplifier was one of the few found that provided a 10X gain at 2.2GHz, with only a 5V source. The power gain graph for this amplifier can be seen in Figure 3.11. A main advantage of this amplifier was its cost-effective price of $1.29.

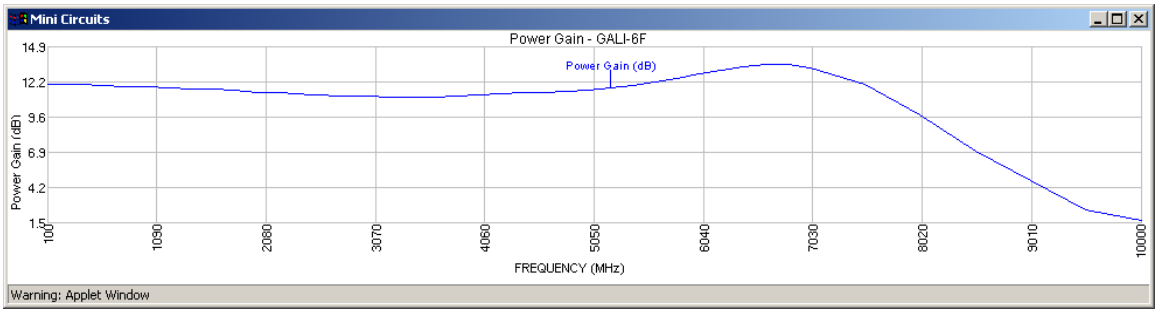


Figure 3.11: Power gain graph for amplifier

4.0 Conclusions

Research was conducted into each necessary aspect of the heat detection device. Testing of the material was completed and electronic components were chosen integrated and assembled. Feasibility was demonstrated for each of the original specifications, except for the requirement of scavenging all power from the fire. This one specification, that was not met, was researched for further physical development. While a single, fully integrated component was not created and destructively tested due to lack of funding and materials, it is clear that this full integration could be easily completed.

It was not demonstrated that the insulation maintains an internal temperature of 150°F while subjected to an intense fire for thirty minutes. However, components are available that have operating temperatures of up to 85^{o}C, or 182 ^{o}F. The insulation tested did demonstrate that it could keep the internal temperature below this temperature when subjected to the extreme temperatures for thirty minutes. The insulation prototype tested was of the appropriate size to house all of the current electronic components.

While the power is not being supplied in a manner consistent with the original specifications, the power supply system chosen meets all other specifications in terms of voltage, current, and shelf life. The device is activated when a switch is subjected to heat of 212°F, which is an improvement over the original specified activation temperature of 250°F. A latching relay ensures that regardless of temperature excursions following the activation of the device, the power supply remains constant.

The electronic components have been fully integrated and perform required functions. The power supply was integrated with each component. There is a K-type thermocouple temperature sensor integrated with an amplifier circuit, which the signal

can be accurately detected and read by the processor. The processor has been shown to encrypt stored data, live temperature, and temperature flux. This has been demonstrated by being hard-wired to a second “receiving” brainstem that decodes the encrypted signal and reads it properly. The data stream can also be sent via a RF signal. It has been demonstration that the transmitter is transmitting a signal of the correct frequency. When a quadrature receiver is available the signal can be decoded to ensure that the signal can be read.

This project was successful in nearly all the revised required specifications. The total developmental cost was $234.00 and cost of reproduction was $203.50. (See Appendix VIII) Using some of the suggestions discussed in the Future Considerations Section, seen in Section 6, can bring the prototype even further along.

5.0 Future Work

This project has many potential improvements. The first and most obvious improvement would be to place all electronic components into one integrated chip. The device should be integrated into the ordinance design to prevent aerodynamic interference. Insulation for the electronics must be minimized to aid in the aerodynamics of the device. A receiving station that has the ability to decode the device signal must be developed or purchased in order to evaluate the transmitted signal. This will also provide information about power characteristics that the signal needs to possess. Research and testing can be done to examine how a transmitted signal propagates onboard ship. Cost must further be reduced to make the device a feasible option.

Works Cited

ASTM “Standard Test Method for Thermal Conductivity of Refractories”, Annual Book of ASTM Standards, American Society for Testing and Materials, Philadelphia, PA.

Bryant,Rodney; Erik Johnsson; Thomas Ohlemille;Carole Womeldorf International Fire Science and Engineering Conference, 9th. Proceedings. September 17-19, 2001. Edinburgh, Scotland. 2001. Estimates of the Uncertainty of Radiative Heat Flux Calculated from Total Heat Flux Measurements.

Cotronics Corp. High Temperature Materials and Adhesives for use to 3000F Volume 01 Number 41 3379 Shore Parkway, Brooklyn, N.Y. 11235

CRC Handbook of Thermoelectrics, D.M.Rowe, Ed. CRC Press New York:1995

Ferrotec-USA Inc. www.ferrotec-usa.com

Geankoplis, Christie J., Transport Processes and Unit Operations 3rd Edition. Prentice Hall, New Jersey, 1993

Loveall, Kevin, Manager, Business Development and Planning; Kirkhill TA. Letter to the author. 20, November 2001.

Omega Temperture Handbook and Encyclopedia, Omega Corp., 2000

Space Shuttle Tile Company; http://www.spaceshuttletile.com/faq.html

Speyer, Robert F. Thermal Analysis of Materials. Marcel Dekker, Inc. New York. 1994 Radial Heat flow. Calorimiter methods.

Thermal Ceramics P.O. Box 923 Augusta Georgia 30903-0923
Kirkhill-TA 28065 Franklin Parkway; Valencia, CA 91355

3M http://www.3m.com/market/industrial/ceramics/pdfs/outerspace_apps_brochure.pdf

Appendix I: Customer Proposal

Appendix II: Power Supply Calculations

The calculations in this appendix illustrate a very simplified case. These calculations show how many modules are needed to meet the power requirements for a dT of 100C. This dT is used if we assume a 100 degree difference between the Devices internal and external temperatures. This would most likely be the dT at the "power on" temperature. A module consists of 127 junctions. The calculations show that the 3 modules are required to meet the power requirements. Other things in consideration are the internal resistance of the modules versus the load resistance. They should match, or be fairly close for maximum power output. There is also a calculation for overall efficiency at dT100. I think the short paragraph explains that 8 modules are needed for a dT of 50 and that these dTs are for relatively low temperatures. A dt of 50C from 800C to 750C is still dT of 50C but there is much more heat flow than the same dT from 100C to 150C. The equations below are simplified to allow for an ideal case.

Average Temperature: Tav= (Th-Tc)/2. Th= hot side temp, Tc= cold side temp.

Th=403K Tc=303K so Tav=353k Load resistance is R=V/I = 5V/1A= 5Ohms

Max Power: Pm= ((Sm*DT)^2)/4*Rm

Where Sm is the Seebeck coefficient 0.05544V/K; DT is the temperature difference between Th and Tc, 100K; Rm is the resistance of the module, 3.00994Ohm.
The result is that the module will produce 2.48 watts of power.
The minimum number of modules needed is Po/Pm, which, for our case equals 2 modules.

The maximum generating efficiency is when the resistance of the generator equals the resistance of the load:

Rgen=(NS*Rm)/Np

NS is the number of modules in series, 2, and Np is the number of modules in parallel, 1.
Rgen = 6.2ohms, not far off of 5ohms
The total heat input to the generator is

Qh=NS*[[((Sm*Th*I)/Np)-0.5]*[(I/2)^2]*Rm+Km*DT]

Qh for a our module set up is 167.2watts Generator efficiency, Eg is:

PO/Qh

PO is the power output from the module configuration.
Eg=2.99%
The heat transferred to the cold side is

Qc-Qh-PO

167.2watts -5watts=162.5 watts.

The equations and figures above only represent one DT. For our device, the TE modules would be embedded into the insulation. If we apply the above equations to our device, the 162.5watts of heat that is transferred to the cold side will have no place to dissipate to, as the insulation stops the heat from moving in that direction. Essentially over time, the heat builds up on the cold side and as a result the DT drops. This continues until DT is zero. At large temperatures more heat is produced, regardless of DT. The smaller the DT, and the higher the temperature, the more modules are needed to produce the same power, therefore, this design is not feasible at this time.

Appendix III: TEA Grammar Index

translation-unit:
  external-declaration
  translation-unit external-declaration

external-declaration:
  function-definition

function-definition:
  declaration-specifier* declarator compound-statement

declaration-specifier:
  type-specifier declaration-specifier*

type-specifier:
  void
  char
  int
  string

declarator:
  identifier
  declarator (parameter-list)

parameter-list:
  parameter-declaration
  parameter-list , parameter-declaration

parameter-declaration:
  declaration-specifier declarator

compound-statement:
  statement-list*

statement-list:
  statement-list statement

statement:
  asm-statement
  selection-statement
  jump-statement
  labeled-statement
  loop-statement
  expression-statement

labeled-statement:
  identifier : statement
  case constant : statement
  default : statement

selection-statement:
  if (expression) statement
  if (expression) statement else statement
  switch (expression) statement

asm-statement:
  asm { asm-list* }

asm-list:
  opcode
  labeled-statement

jump-statement:
  return expression* ;
  continue ;
  break ;
  goto identifier ;

expression:
  assignment-expression
  expression , assigment-expression

assignment-expression:
  conditional-expression
  unary-expression assignment-operator assigment-expression

assignment-operator:
  =

conditional-expression:
  logical-OR-expression
  logical-OR-expression ? expression : conditional-expression

logical-OR-expression:
  logical-AND-expression
  logical-OR-expression || logical-AND-expression

logical-AND-expression:
  inclusive-OR-expression

```
  logical-AND-expression && inclusive-OR-expression

inclusive-OR-expression:
  exclusive-OR-expression
  inclusive-OR-expression | exclusive-OR-expression

exclusive-OR-expression:
  AND-expression
  exclusive-OR-expression ^ AND-expression

AND-expression:
  equality-expression
  AND-expression & equality-expression

equality-expression:
  relational-expression
  equality-expression == relational-expression
  equality-expression != relational-expression

relational-expression:
  shift-expression
  relational-expression < shift-expression
  relational-expression > shift-expression
  relational-expression <= shift-expression
  relational-expression >= shift-expression

shift-expression:
  cast-expression
  shift-expression << cast-expression
  shift-expression >> cast-expression

additive-expression:
  multiplicative-expression
  additive-expression + multiplicative-expression
  additive-expression - multiplicative-expression

multiplicative-expression:
  cast-expression
  multiplicative-expression * cast-expression
  multiplicative-expression / cast-expression
  multiplicative-expression % cast-expression

cast-expression:
  unary-expression
  ( type-specifier ) cast-expression
```

unary-expression:
  postfix-expression
  ++ unary-expression
  -- unary-expression
  unary-operator cast-expression

unary-operator:
  -
  +
  !
  ~

postfix-expression:
  primary-expression
  postfix-expression ++
  postfix-expression --

primary-expression:
  identifier
  constant
  (expression)

loop-statement:
  while (expression) statement
  do statement while (expression) ;
  for (expression* ; expression* ; expression*) statement

Appendix IV: Code for transmission and data acquisition

```
/* this program will send the value on A2D #4 to Digital port 2 ones digit first    /
/    with a buffer of 10 1's folled by a zero.  It also sends the quad signal to the/
/    transmitter.  */

#include <aCore.tea>
#include <aDigIO.tea>

void main()
{
   int a=0,x=0,t = 0,m = 0,f = 1, mis = 0;
   int zz = 500;
   aDigital_Config(2, DIG_OUTPUT);
   aDigital_Config(3, DIG_OUTPUT);
   aDigital_Config(4, DIG_OUTPUT);
   aDigital_Config(0, DIG_INPUT);

while(1)
{
/* print value*/
a = aAnalog_Read_Int(0);
   display_int_dec(a);
   display_char('\n');
   sleep(zz);

/*cue the trasmitter freq*/
  if (f == 1)
  {
  aDigital_Write (3,0);
  aDigital_Write (4,0);
  }
  if (f == 2)
  {
  aDigital_Write (3,1);
  aDigital_Write (4,0);
  }
  if (f == 3)
  {
  aDigital_Write (3,0);
  aDigital_Write (4,1);
  }
  if (f == 4)
  {
  aDigital_Write (3,1);
  aDigital_Write (4,1);
```

```
  f = 0;
  }
/*cue the reciever*/
 while(m<10)
  {
 aDigital_Write(2,1);
 sleep(zz);
  m = m+1;
    }
  m = 0;

  aDigital_Write (2,0);
  sleep(zz);

 /*end cue start xmit */
  if (mis == 0)
  {
  a = 0x0001;
  mis = 5;
  }
  else
  {
  mis= mis - 1;
  }

  for (x = 0 ; x < 10; x= x +1)
 {
  if (a&1)
 {
  aDigital_Write (2,1);
  }
  else
  {
    aDigital_Write (2,0);
  }
  a = a >> 1;
  sleep(zz);
 }


/*end xmit*/
 while(m<10)
  {
 aDigital_Write(2,0);
 sleep(zz);
  m = m+1;
```

```
    }
  m = 0;
/*return to idle*/

 f = f+1;                    /*change freq*/
}
                                                    }
```

Appendix V: Code for the reciever

```
/* this prgram will decode a serial string of 10 bits sent with a buffer of 10 1's with a /
/  trailing zero ones digit first.  It also changes the transmitter number into a          /
/  temperature or references it to a data value.                                          */

#include <aCore.tea>
#include <aDigIO.tea>

void main()
{
  int a=0,x=0,t=0,s=0,m=0,i = 0, delta = 0, pos;
  int zz=500;
  display_string("Reciever is on \n");
  display_char('\n');
  sleep (200);
  while(1)
 {
  aDigital_Config(2, DIG_OUTPUT);
  aDigital_Config(0, DIG_INPUT);

  while(m < 9 )
  {
    a = aDigital_Read_Int(0);
    a = a & 0x0001;
    if ( a & 0x0001)
    {
    m = m+1;
    }
    else
    {
    m = 0;
     }
  sleep(zz);
  }

   if ( m== 9 )
   {
      a = aDigital_Read_Int(0);
      a = a & 0x0001;

      while (a == 1)
      {
        a = aDigital_Read_Int(0);
        a = a & 0x0001;
      }
```

```
  sleep (zz >> 1);
  sleep (zz);
  if (a == 0)
  {

/* start decode */
  x = 0;
  pos = 1;
  for( i = 0; i<10 ; i=i+1)
     {
       a = aDigital_Read_Int(0);
       a = a & 0x00001;
       if (a == 1) { x = x | pos; }
       pos = pos << 1;
       sleep(zz);
     }
   if ( x == 1)
   {
   display_string(" missle is on fire! \n");
   display_char('\n');
   sleep(zz);
   }
  else
  {
   t = x *17;
   t = t / 10;
   display_string(" current temp is ");
   display_int_dec(t);
   display_char('\n');
   sleep(zz);
   delta = t - delta;
   display_string(" temperature flux is ");
   display_int_dec(delta);
   display_char('\n');
   sleep(zz);
   delta = t;
   }
   x  = 0;
   i  = 0;
   a = 0;
   }
 }
else
{
m = 0;
}
```

```
 m = 0;
}
                                                    }
```

Appendix VI: How to compile and load programs onto the *Brainstem* and save then to EEPROM

1) Write the program as a text file and save it as *.tea in the "USER" directory of the Brainstem folder.
2) Open the console window and type the following to compile the program "steep "(file name)"
3) To save to program to the brainstem execute the following: With the Brainstem connected and powered up type the follow "load "(file name)" 2 X" where X is the location you want it stored in. You can choose a number from 0-3.
4) To launch a program enter the following: "launch 2 X" where X is the program location you want to launch.
5) To have a program launch from EEPROM execute the following script.
    - "2 18 X 0" where X is 15 – 18 for file location 0 – 3 i.e. 15 loads the program stored in slot 0.
    - "2 19" this stores the new values to the EEPROM
6) To stop a program from running there are two options
    - "2 24" executes a power up resent and stops all programs from running
    - "2 22 X" where X is the program number that is running terminates just that one program.

Appendix VIII: Budget

Table 5.1 compares the project development cost with the cost of reproduction. This information is important to demonstrate the feasibility of this project in teams of the long team goals.

Item	Distributor	Development Cost	Reproduction Cost
Operational Amplifier LMC6442	National Semiconductor	$0.00	$1.11
Voltage Controlled Oscillator ROS-1121V	Mini Circuits	$0.00	$15.95
Amplifier GALI-6F	Mini Circuits	$0.00	$1.29
Transmitter MAX2721	Maxim	$0.00	$3.23
K-type Thermocouple	NMT Instrument Room	$0.00	$0.10/foot
Fastblock 301	Kirkhill TA	$84.00	$90
Cerablanket	Thermal Ceramics	$0.00	$0.32
Brainstem	Acromane Inc	$64.00	$79.00
DC Relay	Radio Shack	$5.00	$5.00
Thermal Switch	Allelectronics	$8.00	$0.50
Lithium Battery	Radio Shack	$66.00	$7.00
Total		$227.00	$203.50

Table 5.1: Total project budget

Laser Beam Dump and Power Meter

A Thesis
Presented to the Faculty of
the Electrical Engineering Department
of
New Mexico Tech
By

Josef Hart
Ernest Jim
Daniel Rodriguez

In partial fulfillment of the requirements
for the course
EE-482 Senior Design Project II

May, 2002

Abstract

High-energy lasers are used for a variety of experimental purposes. In a laboratory, the laser beam produced must be stopped and absorbed by means of a laser beam dump. Applying such a beam dump can also be useful in monitoring the energy that is left after them beam passes through its target. The given problem is to design and build a beam dump that can withstand and measure the energy directly from a laser as well as a beam that has passed through a target. This project was sponsored by Dr. Daniel Eastman from Boeing SVS and the faculty advisor for the project was Dr. Scott Teare from New Mexico Tech.

This thesis provides a description of the project from beginning to end. It illustrates the steps taken when approaching the problem, the choices made while seeking a solution, and the results from testing the final design.

This document also provides contributions from the individual group members. To expedite the progress of the project, it was divided up into equal parts among the three group members. The combination of their time and efforts results in the device that this document outlines and describes.

Acknowledgements

First we would like to thank Dr. Scott Teare of New Mexico Tech Electrical Engineering Department, who was our class instructor and faculty advisor. Dr. Teare did a great job advising us.

Dr. Daniel Eastman of Boeing SVS in Albuquerque New Mexico for presenting this project to the New Mexico Tech Electrical Engineering Department. And also for supplying the laser and facilities used to test our project.

Dr. Deidre Hirschfeld of the New Mexico Tech Materials Department for directing us in the proper path on the materials portion of this project.

Norton Euart of the use of the machine shop in Workman Center. Without use of the equipment we would have not been able to construct the laser beam dump. And also for supplying us with the components needed in this project.

Table of Contents

List of Tables **Page**

List of Figures **Page**

1. Laser Beam Dump and Power Meter Introduction

1.1 Project Description

The project our group was assigned was to design a high-energy laser beam dump (beam dump) and power meter with a given configuration to meet given specifications. When operating, the laser beam is directed into the "V" shaped segments and the plates absorb its energy. Temperature sensors that are mounted to the plates are used to measure the temperature rise of the plates and the total incident power. The total incident power is computed as a function of this temperature rise.

1.2 Definitions

HEL- High Energy Laser

Specific Heat- The quantity of heat required to raise the temperature of a given mass of a substance. (Serway, 323-342)

Protel 99- A Windows based printed circuit board (PCB) design tool.

1.3 Literature Review

To begin this project, we started by searching the internet and library for sources that concern our project. Since our project was based on a device that already existed, we tried to find previously written documentation about it. The problem was that there was none, and even our customer had trouble finding anything regarding our topic. While we found many sources for commercially available laser beam dumps, there was little about beam dumps that were integrated with power meters. The available literature was useful for the design of the beam dump. It gave us an idea of what materials to use and the geometry of portion that the laser beam strikes, but the power meter design was left to us to develop.

In addition to design ideas, other literature searches helped us find what hardware to use for power measurement. These sites were primarily manufacturer's websites that gave us ideas of how to take commercially available parts and implement them to solve our given problem. Our reference section outlines the sources we reviewed while solving our design problem.

1.4 History

The history of the project comes from the existence of a similar device. The problem was that there was little documentation for the high-energy laser power meter system. It consists of two circles of trapezoidal plates whose specific heat and thermal conductivity are known. It looks similar to the hexagonal form shown in the viewgraphs (Figures here) except instead of a cone to direct the energy azimuthally around the wedge plates they used a rotating mirror spun at high speed. The two surfaces of the hexagonal beam dump form a "V" facing inward.

2. Technical Background

The problem to be solved consists of three major details. The first was to consider the specifications given in table one and design a power meter prototype that could be used

Specifications
-Plates scaled to accept a 0.25 to 2.0 inch laser beam
-Plate material, edge treatment and thickness designed to accept a 10-watt laser beam for 50 seconds operating at a wavelength of 1.3 microns.
-Plate material and thickness chosen so that the temperature rise is sufficient to measure the incident power to 2% yet low enough so that the material and treatment is not ablated or destroyed so that multiple shots can be measured (N<1000)
-Plate angle and surface treatment to be determined with a design goal to absorb 99.9% of the incident energy

Table 1-Project Specifications

to measure the incident power propagated by a laser. The next was to investigate commercially available beam dumps to determine the optimal material and angle needed to meet the given specifications yet use commercially available materials and surface treatments. Then we were to build a prototype capable of measuring the power output from a given laser and with a laser furnished by the customer demonstrate its capability.

2.1 Subsystem Solutions

The device consists of two rectangular plates subtending an angle determined by the designer. The designer also determined the material, thickness and surface treatment. Several temperature sensors are attached to the outer surfaces, and the temperature sensors are linked to an analog to digital board (A/D board).

2.1.1 Laser Beam Dump Design

Designing and constructing the laser beam dump. This involved determining what material to use, what dimensions the beam dump would have, and the availability of each material.

First several materials were investigated. Materials with high melting points and good thermal conductivity were selected. The properties (melting point, thermal conductivity, specific heat, mass, reflectivity) of each material were documented for later calculations. Since most of these materials were highly reflective, it was decided that we should apply a dark coating to the inner surface layer of the material. This would increase the absorption of the laser beam. Two selections were black anodized and black paint. Black anodized was very durable, but it could only be used on aluminum and copper. It also had to be applied to the surface of the material through a chemical process and could only be applied after the beam dump parts were machined to the correct size. Black paint could be used on any material, it was easy to apply, and it was available the locally. We decided to use the black paint because applying it did not involve a chemical process like black anodized involved.

Next the dimensions of the wedge plates had to be determined. The basic shape of the wedge was a V as seen in figure XX. The wedge consisted of two identical

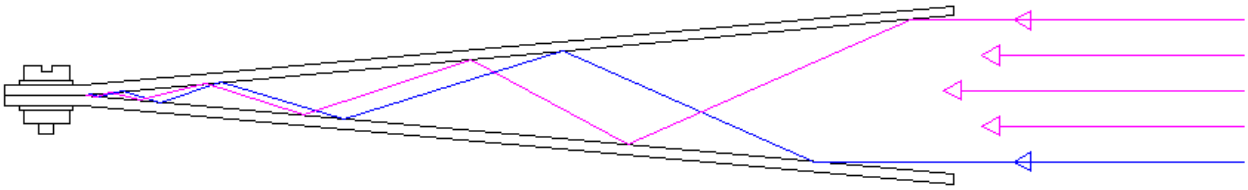


Figure 1. Beam Dump with Laser Beams entering

plates that were bolted together. The opening of the V would be one inch and each plate would be one and one eighth of an inch wide. The length and thickness of each plate were calculated using specific heat, mass, laser power, and expected change in temperature of each material. Which basically comes from the equation of specific heat ($c = Q/m*\Delta T$). Now that the dimensions of the beam dump for each material were determined, this further narrowed the selection of material type.

Finally when all calculations were complete the availability of each material was checked into. Aluminum was the only material that was available locally. It could be purchased at the local hardware store in the thickness and width determined in our calculations. This meant that machining would be made easier. Therefore we decided to use aluminum because it was easier and faster to get a hold of and since it came in some of the dimensions we wanted. The final size and dimensions of the beam dump for aluminum are shown in figure 2.

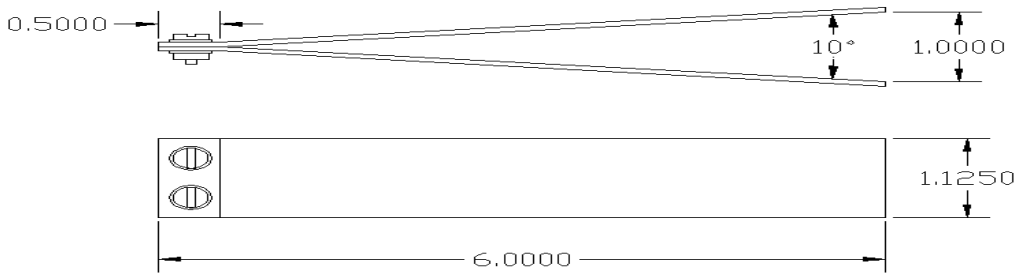


Figure 2. Final dimensions of the beam dump

2.1.2 Heat Sensing Hardware Design

Measuring the temperature rise in the plates was an important part in the laser beam dump design. At first glance there was a wide range of surface temperature sensing hardware to choose from. The options that were explored were thermocouples, thermistors and IC temperature sensors. Thermocouples are interchangeable, have standard connectors and can measure a wide range of temperatures. Their response time is in the milliseconds. One of their weak aspects is that they require special connectors, and test equipment, which would significantly increase the cost per unit. The prices for these sensors are about $13.00 each. Thermistors have some of the same advantages as thermocouples. The draw back of using thermistors is their slow response time. Their response time can be in the range of seconds. The price for this type of sensor is about $15.00 each. IC temperature sensors are inexpensive, fairly precise, and can easily integrated into other circuits.

The solution to the heat sensing hardware was to use an IC temperature sensor. The sensor chosen was the LM34CAZ. The LM34CAZ is a precision integrated-circuit temperature sensor. The output of the sensor is linearly proportional to the Fahrenheit temperature. The sensor gives out a 10mV change per degree Fahrenheit. An example of this is at 83.2 degrees the sensor gives out .832 volts. The accuracy of the sensor is plus or minus one and one-half degrees Fahrenheit. The price for these sensors is $6.64 each. These factors make this temperature sensor a good alternative to higher price temperature sensors.

2.1.3 Data Collection System Design

The data collection subsystem collected the data from the temperature sensors, passed the values into a computer, and then performed calculations on the data to return a useful value. The temperature sensors returned a temperature value in the form of a voltage, so that voltage number had to be calculated with other known values to produce a power value.

To solve this problem we chose the DI-194 A/D board from Dataq Instruments. This board did not provide the level of precision our customer desired, but due to budget constraints it served its purpose very well. The DI-194 is a four-channel, eight bit A/D board. It connected very simply to the external serial port of a computer and it received its power from this port. The A/D board was then interfaced with Microsoft Excel through the use of a macro written in visual basic (Huang). A copy of this program can be found in the appendix on page twenty-three. We chose to use Excel because we were already familiar with the software package and this allowed us to stay on schedule.

3. Individual Contributions

3.1 Josef Hart

Josef was the group's point of contact, group leader, and responsible for interfacing the laser beam dump hardware with a data collection system.

As the point of contact, Josef kept in contact with both Dr. Scott Teare, class instructor and the group's faculty advisor, and Dr. Daniel Eastman, the customer from Boeing-SVS. Josef then passed on the information to the rest of his group so that they were aware of all communications and project developments.

The responsibilities of the group leader coincided with the duties of point of contact. In addition to keeping everyone's work on track, Josef also arranged meetings and scheduled tests at the customer's facilities.

The task of interfacing the laser beam dump hardware with a data collection system consisted of two main areas. The first was to connect the sensors to a computer through an analog to digital (A/D) board and the second was to process the data with a software package that would calculate and return values that were useful in our experiment. The A/D board was a four-channel board and connected to a computer through the serial port. It took its power from the computer, and therefore required no external power source. The board also did not require any programming, which made set-up simple. The vendor that we bought the board from also provided methods for interfacing the A/D board with commonly used software packages such as Microsoft Excel (Excel). By downloading and installing a program (reference and code) from the vendor that enabled Excel to receive data from the computer's serial port, the collection

and calculation of the data was as easy as using Excel to perform mathematical calculations.

3.2 Ernest Jim

Ernest was in charge of constructing the laser beam dump. This involved determining what type of material the beam dump would be made from and what shape it would have to best fit the project needs. The material was determined by the amount of energy that the beam dump needed to absorb over time. This gave the information needed to determine the mass, length, thickness and width of the beam dump wedges. Once the dimensions were calculated the plates were cut, bent, drilled and assembled in the machine shop. To increase absorption of the beam dump the inside of the wedge plates was coated with black paint. The wedge plates were long, so the angle between the plates was small. This enabled the beam dump to keep less energy from escaping. Calculations for the wedge were made for several different types of materials: copper, alumina, and steel.

3.3 Daniel Rodriguez

Daniel was responsible for designing and developing the heat sensing hardware system of the project.

First, Daniel performed a literature search on the commercially available temperature sensors. The result was the available thermocouples, thermistors and IC temperature sensors. Thermocouples were the first considered elements to be used. The high sensitivity and fast output reading were the main characteristics that led to the choice of these sensors. The weak aspect of the thermocouples was that they were expensive, and due to budget constraints Daniel had to consider another device. So,

following a suggestion from our costumer Daniel Eastman, Daniel Rodriguez decided to use an IC temperature sensor. The final sensor chosen was the LM34CAZ. The sensitivity was lower then the thermocouples and the response time was slower but the cost was much cheaper.

After selecting and purchasing the temperature sensors, Daniel could begin testing the temperature sensors. The initial testing was done on the breadboard. A heated knife placed against the sensor was used as a stimulus to the sensors so the response from the sensors could be measured. Later testing showed that a gain would be required so that reliable data could be sent to the A/D board. Using the information provided by Ernest Jim on the degree change that would be seen and the limitations of the A/D board, the sensor output was amplified by six.

Once the temperature sensor circuit was designed, Daniel etched a printed circuit board (PCB) for the temperature sensors. He used Protel 99 to create a simple, easy to use design. The use of one percent resistors, LM 741 operational amplifiers, and one-way connectors made the temperature sensor package not only durable but also highly precise. The Schematic for the sensor is given in Figure 1.

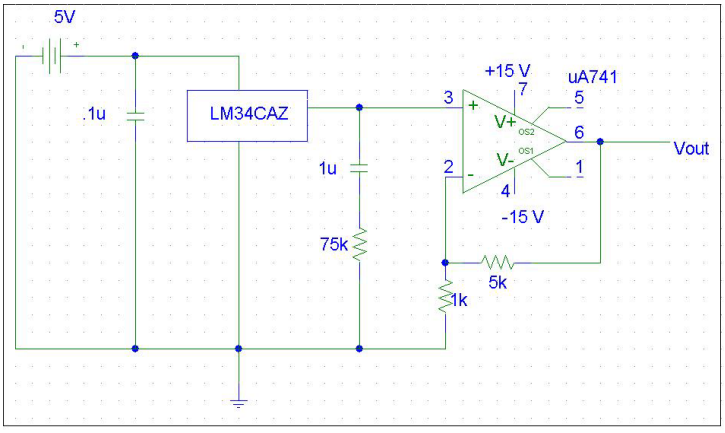


Figure 3. Schematic of the IC temperature sensor and op-amp

When the temperature sensor circuit was complete, Daniel constructed a wiring harness to connect the sensors to the A/D board. The wiring harness consisted of one-way connectors for the sensor PCB in order to avoid confusion between ground and power leads on the sensor package. The wires were also clearly labeled so the ground, power, and sensor output would be connected correctly.

With the sensor hardware fabricated, Daniel was able to attach the sensors to the body of the beam dump. He attached the sensors to the aluminum plates with a non-conductive epoxy. Besides not conducting electricity, the epoxy also could withstand high temperatures so the sensors were held firmly to the beam dump during the entire test.

4. Summary and Results

Overall this project was successful. We performed four trials with an HEL up at Boeing-SVS in Albuquerque, New Mexico on April 11, 2002. The trial results are shown in table X. The front panel of the laser read that it was propagating an eight-watt beam

Trial Results	
Trial 1	3.23 Watts
Trial 2	2.53 Watts
Trial 3	2.53 Watts
Trial 4	2.33 Watts

Table 2- Trial Results

which at first was discouraging since our power meter was only returning values in the 2.3 to 3.3 watt range. After the trials, the laser operator placed a power meter in front of the beam to take an actual power reading. The result from this measurement was 2.42 watts. When compared to our data, we had a 9.7 percent error. Not only were we pleased with the results but the customer was also. A copy of the data can be found in appendices A.4 through A.7.

4.1 Discussion of Results

The results were pretty much what we were expecting. The plates heated up and cooled down as the charts in appendix A.3 show. The heat up time lasts from zero to fifty seconds. After fifty seconds, the laser was turned off, and the cool down phase began. The series one and two lines represent the sensors that were on the back third of the beam dump. They appeared to heat up the most. The other two sensors were on the

out third of the beam dump and did not heat up quite as much as the first two. All of the sensors appeared to cool down to the same value between trials.

The actual power value was only computed from the first fifty cells which were when the laser was on. From those cells, we found the overall change in temperature, multiplied it by the mass and specific heat of aluminum, and then divided it by the time, or fifty seconds. The final values can be seen in either Table 2, or in appendices A.4 through A.7.

4.2 Laser Beam Dump Summary

The body of the beam dump worked like we thought it would. We did not have the means or time to test it individually, but when the sensors were attached and laser energy absorbed, the beam dump performed according to calculations. Given more time for design and with the laser we would have worked harder to ensure that all light and energy was absorbed. Due to our results, we can assume that all of the energy and power were absorbed, but we could not see if any light was reflecting out of the beam dump.

4.3 Heat Sensing Hardware Summary

The use of the temperature sensor package helped tie in the actual beam dump with the data collection. The simple reliable package made it possible to measure the heat exchange in the aluminum. This measurement was then feed into the A to D board for data collection and manipulation. Without the sensor package one of the main goals of this project could not have been possible. The actual LM34 temperature sensor in the sensor package worked as specified in the factory specifications for these sensors.

4.4 Data Collection System Summary

The data collection system worked exactly like we thought it would. The A/D board was very easy to connect and interface with a computer. Interfacing the A/D board with other software packages was also simple since such resources were provided by the manufacturer (Huang). Starting and stopping the program was a simple push button that was inserted into an Excel spreadsheet. The data was collected directly by Excel and we were able to perform the necessary calculations in another cell. Excel proved to be a straight-forward, easy to use method of collecting and calculating the data.

4.5 Budget

The following was our expenditure for this project. Additional items that were donated are marked with zero in the price column.

Item	Qty	Unit Price	Total
1% Resistors (1K ohm)	4	$0.10	$0.40
1% Resistors (75K ohm)	4	$0.10	$0.40
1% Resistors (5K ohm)	4	$0.10	$0.40
Capacitor .1uF	4	$0.00	$0.00
Capacitor .1uF	4	$0.00	$0.00
Printed Circuit Board (copper)	1	$5.00	$5.00
Small Connectors	8	$0.60	$4.80
Ring Connectors	11	$0.12	$1.32
Large Ring Connectors	2	$0.08	$0.16
Tap	4	$0.15	$0.60
Flex Block	4	$0.41	$1.64
Solder	1	$2.40	$2.40
Cable Ties	4	$0.04	$0.16
UA741 (op-amp)	4	$0.00	$0.00
LM34CAZ	5	$6.64	$33.20
A/D Board kit	1	$24.95	$24.95
Sheet Aluminum	1	$3.00	$3.00
Temperature Resistant Paint	1	$3.00	$3.00
		Total:	**$81.43**

Table 3. Table of total expenditures in the building of the Laser Beam Dump

5. Conclusions

The goal of this project was design and develop a high power laser beam dump and power meter. This was not the type of project we had expected to see due to the interdisciplinary aspects. Having a strong foundation in electrical components and data collection, we were left to learn material properties as well. After learning about and choosing a material that could be set in front of a laser, the rest of the project was a matter of applying our electrical engineering skills with our newly acquired material knowledge. The result was a successful project that returned values very close values shown by a commercial laser power meter.

This project was a very valuable experience. Up until now our practical lab work has been routine and predictable. This project still required the skills learned in those labs, but the final answer was not known. The successful completion of this project illustrates our ability to come together as a team and solve a problem that is anything but routine.

References

"Contact Temperature Sensors " Temperature Sensors. 1996-2002 <http://www.temperatures.com/csensors.html> (2 September 2001)

"LM34 Precision Fahrenheit Temperature Sensors" Datasheet. 30 November 2000 <http://www.national.com/pf/LM/LM34.html> (2 September 2001)

"Temperature" Products By Handbook. 5 January 2001 <www.omega.com> (2 September 2001).

"DI-194 Starter Kit" Dataq Instruments. 28, March 2002. <http://www.dataq.com/products/startkit/di194rs.htm> (29 April 2002).

Huang, Manna. "How to Use ActiveX in Excel." Ulima Serial. 17 May 2001. <http://www.geocities.com/ultimaserial/exceltutor.html> (29 May 2002).

Serway, R.A. and Faughn, J.S. College Physics, 4th edition. Harcourt Brace College Publishers, 1995.

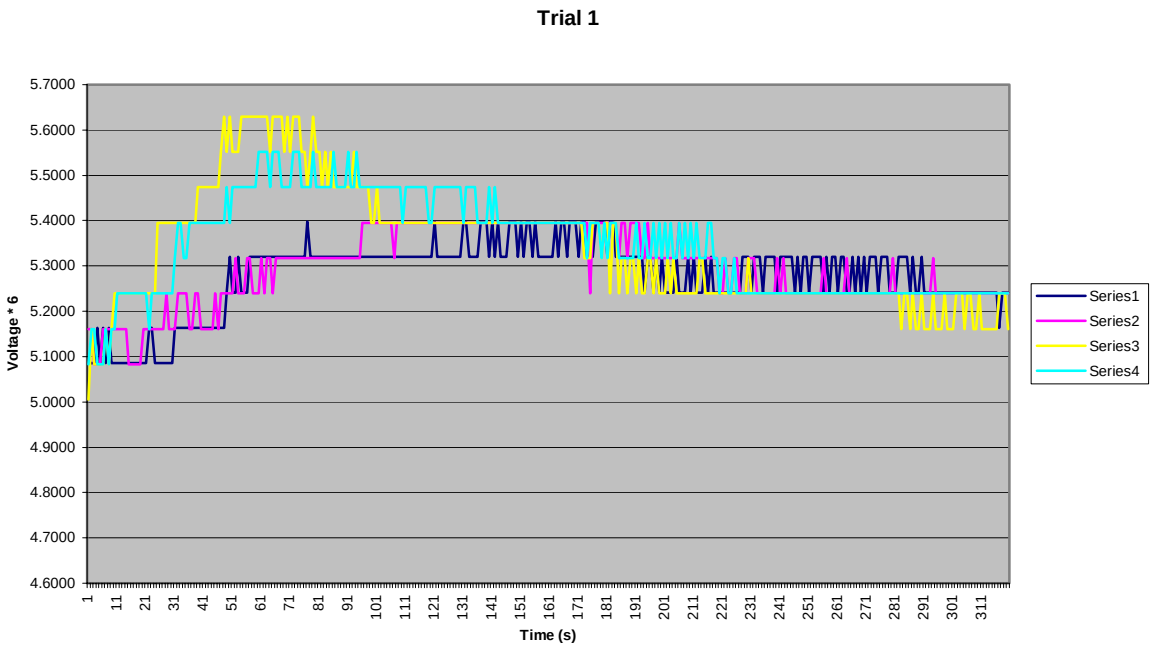
A.1: Computer Code

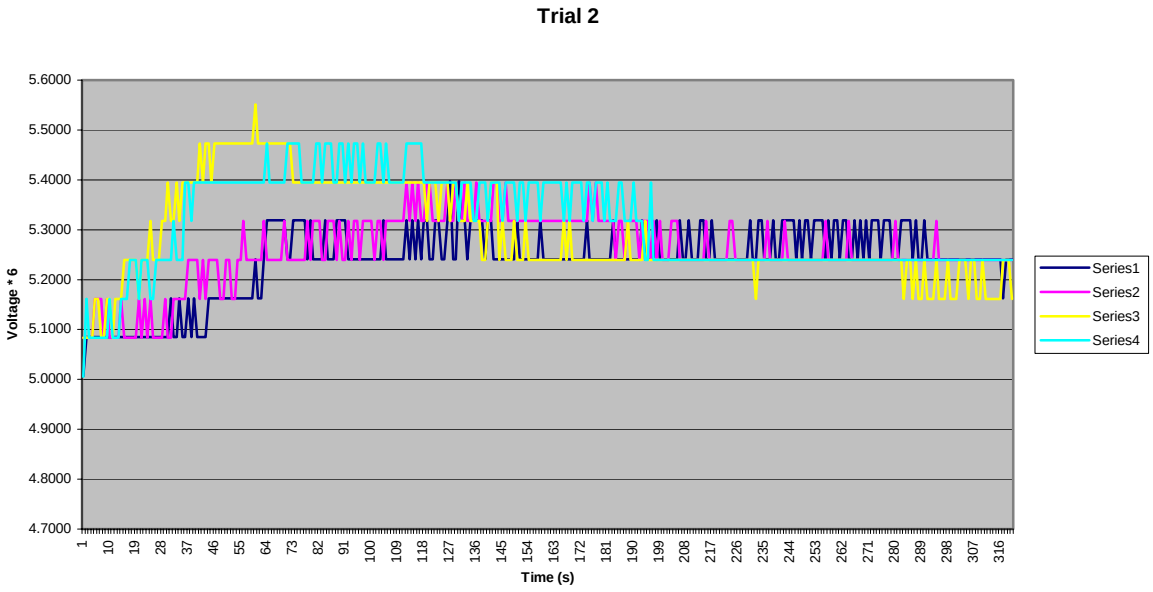
```
Dim cellindex As Integer
Private Sub CommandButton1_Click()
UltimaSerial1.CommPort = 1
UltimaSerial1.Device = 194
UltimaSerial1.SampleRate = 100
UltimaSerial1.EventLevel = 0
UltimaSerial1.ChannelCount = 4
UltimaSerial1.Key = variable
variable = "35FFD0BD"
UltimaSerial1.Start
IeTimer1.Interval = 1000
IeTimer1.Enabled = -1
For cellindex = 1 To 120
ActiveSheet.Cells(cellindex, 1) = ""
ActiveSheet.Cells(cellindex, 2) = ""
ActiveSheet.Cells(cellindex, 3) = ""
ActiveSheet.Cells(cellindex, 4) = ""
ActiveSheet.Cells(cellindex, 5) = ""
Next
cellindex = 1
End Sub
Private Sub CommandButton2_Click()
IeTimer1.Enabled = ValFalse
UltimaSerial1.Stop

End Sub

Private Sub IeTimer1_Timer()
i = UltimaSerial1.AvailableData
ActiveSheet.Cells(cellindex, 2) = Format$(UltimaSerial1.AnalogInput(Ch1) / 3276.8,
"0.00")
ActiveSheet.Cells(cellindex, 3) = Format$(UltimaSerial1.AnalogInput(Ch2) / 3276.8,
"0.00")
ActiveSheet.Cells(cellindex, 4) = Format$(UltimaSerial1.AnalogInput(Ch3) / 3276.8,
"0.00")
ActiveSheet.Cells(cellindex, 5) = Format$(UltimaSerial1.AnalogInput(Ch4) / 3276.8,
"0.00")
ActiveSheet.Cells(cellindex, 1) = Time$

cellindex = cellindex + 1
If cellindex > 120 Then cellindex = 1
End Sub
```

A.2: Gantt Chart

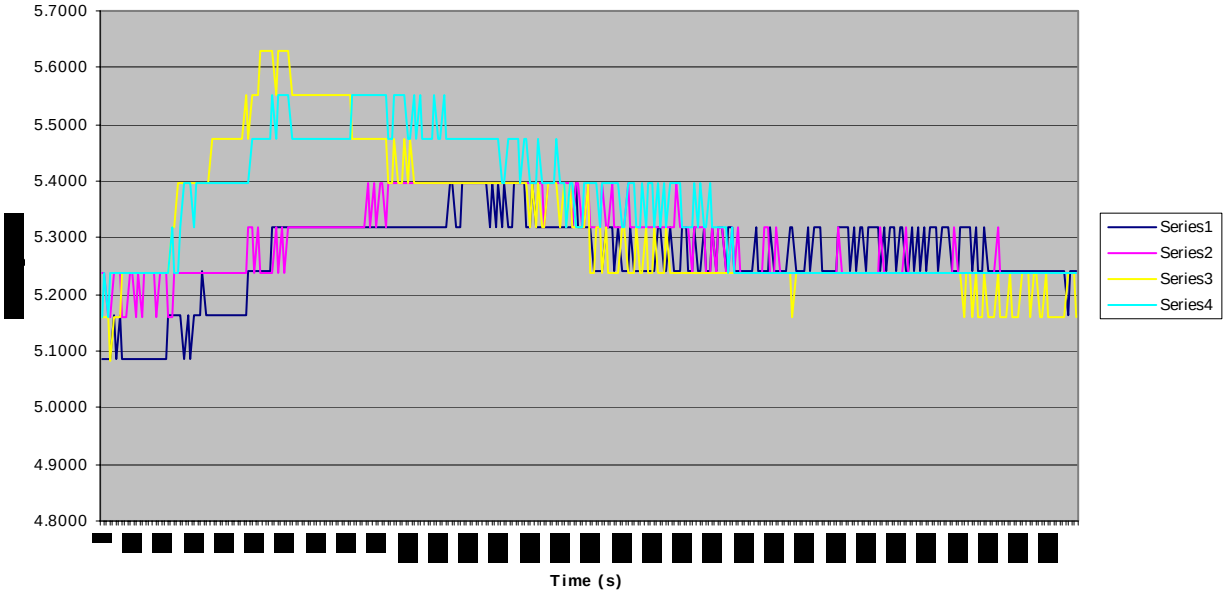
Gantt Chart		Sep-01				Oct-01				Nov-01				Dec-01				Jan-02				Feb-02				Mar-02				Apr-02				May-02			
	Responsible Person	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Statement of work	all																																				
Research of Materials	Ernest Jim																																				
Research of Thermocouples	Daniel Rodriguez																																				
Research of A/D boards	Joe Hart																																				
Design Status Report	all																																				
Build prototype wedge	Ernest, Joe																																				
Test Optics	Ernest, Joe																																				
Test Temperature Sensors	Daniel Rodriguez																																				
Connect and test TS with A/D board	all																																				
Integration of temperature sensors	all																																				
System test	all																																				
Final Documentation	all																																				

A.3: Trial Data Charts

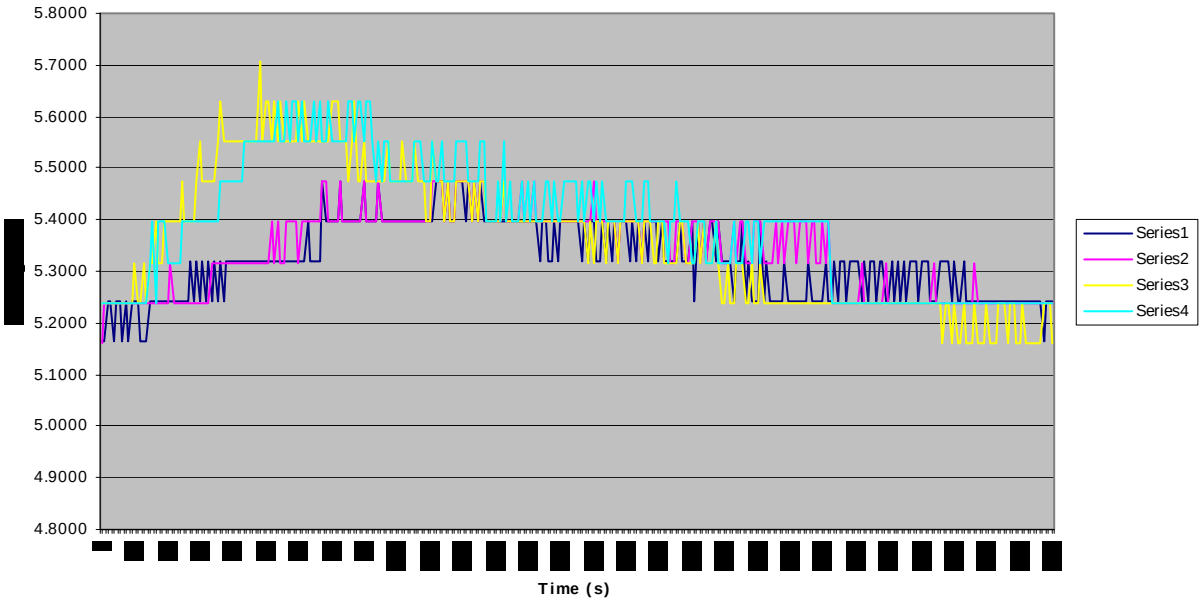




Trial 3



Trial 4



A.4: Trial 1 Data

14:27:40	5.0851	5.1611	5.0049	5.0830					
14:27:41	5.0851	5.1611	5.1611	5.1611					
14:27:42	5.0851	5.0830	5.0830	5.1611					
14:27:43	5.1633	5.0830	5.0830	5.0830					
14:27:44	5.0851	5.0830	5.0830	5.0830					
14:27:45	5.1633	5.1611	5.0830	5.0830	Max Values from rows 1-50				
14:27:46	5.0851	5.1611	5.1611	5.1611	5.3195	5.2393	5.6299	5.4736	
14:27:47	5.1633	5.1611	5.0830	5.0830					
14:27:48	5.0851	5.1611	5.1611	5.1611	Min Values from rows 1-50				
14:27:49	5.0851	5.1611	5.2393	5.1611	5.0851	5.0830	5.0049	5.0830	
14:27:50	5.0851	5.1611	5.2393	5.2393					
14:27:51	5.0851	5.1611	5.2393	5.2393	Avg Value of Maxes	F		C	
14:27:52	5.0851	5.1611	5.2393	5.2393	5.4156	90.25958		32.36644	
14:27:53	5.0851	5.1611	5.2393	5.2393					
14:27:54	5.0851	5.0830	5.2393	5.2393	Avg Value of Mins	F		C	
14:27:55	5.0851	5.0830	5.2393	5.2393	5.0640	84.4		29.11111	
14:27:56	5.0851	5.0830	5.2393	5.2393					
14:27:57	5.0851	5.0830	5.2393	5.2393	Av. Max - Av. Min	F		C	
14:27:58	5.0851	5.0830	5.2393	5.2393	0.3516	5.859583		3.255324	
14:27:59	5.0851	5.1611	5.2393	5.2393					
14:28:00	5.0851	5.1611	5.2393	5.2393	Average Temp change in Fahrenheit				
14:28:01	5.1633	5.1611	5.2393	5.1611	5.859583				
14:28:02	5.1633	5.1611	5.2393	5.2393					
14:28:03	5.0851	5.1611	5.2393	5.2393	Temperature Change in Celcius				
14:28:04	5.0851	5.1611	5.3955	5.2393	3.255324				
14:28:05	5.0851	5.1611	5.3955	5.2393					
14:28:06	5.0851	5.1611	5.3955	5.2393	Power Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:28:07	5.0851	5.2393	5.3955	5.2393	3.232146	Watts	time		
14:28:08	5.0851	5.1611	5.3955	5.2393					
14:28:09	5.0851	5.1611	5.3955	5.2393	Energy Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:28:10	5.1633	5.1611	5.3955	5.3174	161.6073	Joules			
14:28:11	5.1633	5.2393	5.3955	5.3955					
14:28:12	5.1633	5.2393	5.3955	5.3955					
14:28:13	5.1633	5.2393	5.3955	5.3174					
14:28:14	5.1633	5.2393	5.3955	5.3174					
14:28:15	5.1633	5.1611	5.3955	5.3955					
14:28:16	5.1633	5.1611	5.3955	5.3955					
14:28:17	5.1633	5.2393	5.3955	5.3955					
14:28:18	5.1633	5.2393	5.4736	5.3955					
14:28:19	5.1633	5.1611	5.4736	5.3955					
14:28:20	5.1633	5.1611	5.4736	5.3955					
14:28:21	5.1633	5.1611	5.4736	5.3955					
14:28:22	5.1633	5.1611	5.4736	5.3955					
14:28:23	5.1633	5.1611	5.4736	5.3955					
14:28:24	5.1633	5.2393	5.4736	5.3955					
14:28:25	5.1633	5.1611	5.4736	5.3955					

14:28:26 5.1633 5.2393 5.5518 5.3955
14:28:27 5.1633 5.2393 5.6299 5.3955
14:28:28 5.2414 5.2393 5.5518 5.4736
14:28:29 5.3195 5.2393 5.6299 5.3955
14:28:30 5.2414 5.2393 5.5518 5.4736
14:28:31 5.2414 5.3174 5.5518 5.4736
14:28:32 5.3195 5.2393 5.5518 5.4736
14:28:33 5.2414 5.2393 5.6299 5.4736
14:28:34 5.2414 5.2393 5.6299 5.4736
14:28:35 5.2414 5.3174 5.6299 5.4736
14:28:36 5.3195 5.3174 5.6299 5.4736
14:28:37 5.3195 5.2393 5.6299 5.4736
14:28:38 5.3195 5.2393 5.6299 5.4736
14:28:39 5.3195 5.2393 5.6299 5.5518
14:28:40 5.3195 5.3174 5.6299 5.5518
14:28:41 5.3195 5.2393 5.6299 5.5518
14:28:42 5.3195 5.3174 5.6299 5.5518
14:28:43 5.3195 5.3174 5.5518 5.4736
14:28:44 5.3195 5.2393 5.6299 5.5518
14:28:45 5.3195 5.3174 5.6299 5.5518
14:28:46 5.3195 5.3174 5.6299 5.5518
14:28:47 5.3195 5.3174 5.6299 5.4736
14:28:48 5.3195 5.3174 5.5518 5.4736
14:28:49 5.3195 5.3174 5.6299 5.4736
14:28:50 5.3195 5.3174 5.5518 5.4736
14:28:51 5.3195 5.3174 5.6299 5.5518
14:28:52 5.3195 5.3174 5.6299 5.5518
14:28:53 5.3195 5.3174 5.6299 5.5518
14:28:54 5.3195 5.3174 5.5518 5.4736
14:28:55 5.3195 5.3174 5.5518 5.4736
14:28:56 5.3976 5.3174 5.4736 5.4736
14:28:57 5.3195 5.3174 5.5518 5.4736
14:28:58 5.3195 5.3174 5.6299 5.5518
14:28:59 5.3195 5.3174 5.5518 5.4736
14:29:00 5.3195 5.3174 5.5518 5.4736
14:29:01 5.3195 5.3174 5.4736 5.4736
14:29:02 5.3195 5.3174 5.5518 5.4736
14:29:03 5.3195 5.3174 5.4736 5.4736
14:29:04 5.3195 5.3174 5.5518 5.4736
14:29:05 5.3195 5.3174 5.4736 5.5518
14:29:06 5.3195 5.3174 5.4736 5.4736
14:29:07 5.3195 5.3174 5.4736 5.4736
14:29:08 5.3195 5.3174 5.4736 5.4736
14:29:09 5.3195 5.3174 5.4736 5.4736
14:29:10 5.3195 5.3174 5.4736 5.5518
14:29:11 5.3195 5.3174 5.4736 5.4736
14:29:12 5.3195 5.3174 5.5518 5.4736
14:29:13 5.3195 5.3174 5.4736 5.5518

14:29:14 5.3195 5.3174 5.4736 5.4736
14:29:15 5.3195 5.3955 5.4736 5.4736
14:29:16 5.3195 5.3955 5.4736 5.4736
14:29:17 5.3195 5.3955 5.4736 5.4736
14:29:18 5.3195 5.3955 5.3955 5.4736
14:29:19 5.3195 5.3955 5.3955 5.4736
14:29:20 5.3195 5.3955 5.4736 5.4736
14:29:21 5.3195 5.3955 5.3955 5.4736
14:29:22 5.3195 5.3955 5.3955 5.4736
14:29:23 5.3195 5.3955 5.3955 5.4736
14:29:24 5.3195 5.3955 5.3955 5.4736
14:29:25 5.3195 5.3955 5.3955 5.4736
14:29:26 5.3195 5.3174 5.3955 5.4736
14:29:27 5.3195 5.3955 5.3955 5.4736
14:29:28 5.3195 5.3955 5.3955 5.4736
14:29:29 5.3195 5.3955 5.3955 5.3955
14:29:30 5.3195 5.3955 5.3955 5.4736
14:29:31 5.3195 5.3955 5.3955 5.4736
14:29:32 5.3195 5.3955 5.3955 5.4736
14:29:33 5.3195 5.3955 5.3955 5.4736
14:29:34 5.3195 5.3955 5.3955 5.4736
14:29:35 5.3195 5.3955 5.3955 5.4736
14:29:36 5.3195 5.3955 5.3955 5.4736
14:29:37 5.3195 5.3955 5.3955 5.4736
14:29:38 5.3195 5.3955 5.3955 5.3955
14:29:39 5.3195 5.3955 5.3955 5.3955
14:29:40 5.3976 5.3955 5.3955 5.4736
14:29:41 5.3195 5.3955 5.3955 5.4736
14:29:42 5.3195 5.3955 5.3955 5.4736
14:29:43 5.3195 5.3955 5.3955 5.4736
14:29:44 5.3195 5.3955 5.3955 5.4736
14:29:45 5.3195 5.3955 5.3955 5.4736
14:29:46 5.3195 5.3955 5.3955 5.4736
14:29:47 5.3195 5.3955 5.3955 5.4736
14:29:48 5.3195 5.3955 5.3955 5.4736
14:29:49 5.3195 5.3955 5.3955 5.4736
14:29:50 5.3976 5.3955 5.3955 5.3955
14:29:51 5.3976 5.3955 5.3955 5.4736
14:29:52 5.3195 5.3955 5.3955 5.4736
14:29:53 5.3195 5.3955 5.3955 5.4736
14:29:54 5.3195 5.3955 5.3955 5.4736
14:29:55 5.3195 5.3955 5.3955 5.3955
14:29:56 5.3976 5.3955 5.3955 5.3955
14:29:57 5.3976 5.3955 5.3955 5.3955
14:29:58 5.3976 5.3955 5.3955 5.3955
14:29:59 5.3195 5.3955 5.3955 5.4736
14:30:00 5.3976 5.3955 5.3955 5.3955
14:30:01 5.3195 5.3955 5.3955 5.4736

14:30:02 5.3976 5.3955 5.3955 5.3955
14:30:03 5.3195 5.3955 5.3955 5.3955
14:30:04 5.3195 5.3955 5.3955 5.3955
14:30:05 5.3195 5.3955 5.3955 5.3955
14:30:06 5.3976 5.3955 5.3955 5.3955
14:30:07 5.3976 5.3955 5.3955 5.3955
14:30:08 5.3976 5.3955 5.3955 5.3955
14:30:09 5.3195 5.3955 5.3955 5.3955
14:30:10 5.3976 5.3955 5.3955 5.3955
14:30:11 5.3195 5.3955 5.3955 5.3955
14:30:12 5.3976 5.3955 5.3955 5.3955
14:30:13 5.3976 5.3955 5.3955 5.3955
14:30:14 5.3195 5.3955 5.3955 5.3955
14:30:15 5.3976 5.3955 5.3955 5.3955
14:30:16 5.3195 5.3955 5.3955 5.3955
14:30:17 5.3195 5.3955 5.3955 5.3955
14:30:18 5.3195 5.3955 5.3955 5.3955
14:30:19 5.3195 5.3955 5.3955 5.3955
14:30:20 5.3195 5.3955 5.3955 5.3955
14:30:21 5.3195 5.3955 5.3955 5.3955
14:30:22 5.3976 5.3955 5.3955 5.3955
14:30:23 5.3195 5.3955 5.3955 5.3955
14:30:24 5.3976 5.3955 5.3955 5.3955
14:30:25 5.3976 5.3955 5.3955 5.3955
14:30:26 5.3195 5.3955 5.3955 5.3955
14:30:27 5.3976 5.3955 5.3955 5.3955
14:30:28 5.3976 5.3955 5.3955 5.3955
14:30:29 5.3976 5.3955 5.3955 5.3955
14:30:30 5.3195 5.3955 5.3955 5.3955
14:30:31 5.3976 5.3955 5.3955 5.3955
14:30:32 5.3976 5.3955 5.3174 5.3955
14:30:33 5.3195 5.3955 5.3174 5.3174
14:30:34 5.3195 5.2393 5.3174 5.3955
14:30:35 5.3195 5.3955 5.3955 5.3955
14:30:36 5.3976 5.3955 5.3955 5.3955
14:30:37 5.3976 5.3955 5.3955 5.3955
14:30:38 5.3976 5.3955 5.3174 5.3174
14:30:39 5.3976 5.3955 5.3955 5.3955
14:30:40 5.3195 5.3955 5.3955 5.3174
14:30:41 5.3976 5.3955 5.2393 5.3955
14:30:42 5.3976 5.3955 5.3955 5.3955
14:30:43 5.3195 5.3955 5.3955 5.3955
14:30:44 5.3195 5.3174 5.2393 5.3174
14:30:45 5.3195 5.3955 5.3174 5.3174
14:30:46 5.3195 5.3955 5.3174 5.3174
14:30:47 5.3195 5.3174 5.2393 5.3174
14:30:48 5.3195 5.3955 5.3174 5.3174
14:30:49 5.3195 5.3955 5.3174 5.3174

14:30:50 5.3195 5.3955 5.2393 5.3955
14:30:51 5.3195 5.3955 5.3174 5.3174
14:30:52 5.3195 5.3174 5.2393 5.3174
14:30:53 5.2414 5.3174 5.2393 5.3955
14:30:54 5.3195 5.3955 5.3174 5.3174
14:30:55 5.3195 5.3174 5.3174 5.3174
14:30:56 5.2414 5.3174 5.2393 5.3955
14:30:57 5.3195 5.3174 5.3174 5.3174
14:30:58 5.2414 5.3174 5.2393 5.3955
14:30:59 5.3195 5.3174 5.2393 5.3174
14:31:00 5.2414 5.3174 5.2393 5.3955
14:31:01 5.2414 5.3174 5.3174 5.3174
14:31:02 5.2414 5.3174 5.2393 5.3955
14:31:03 5.2414 5.3174 5.3174 5.3174
14:31:04 5.3195 5.3174 5.2393 5.3174
14:31:05 5.2414 5.3955 5.2393 5.3955
14:31:06 5.2414 5.3174 5.2393 5.3174
14:31:07 5.2414 5.3174 5.2393 5.3955
14:31:08 5.3195 5.3174 5.2393 5.3174
14:31:09 5.2414 5.3174 5.2393 5.3955
14:31:10 5.3195 5.3174 5.2393 5.3174
14:31:11 5.2414 5.3174 5.2393 5.3955
14:31:12 5.2414 5.3174 5.3174 5.3174
14:31:13 5.2414 5.3174 5.3174 5.3174
14:31:14 5.3195 5.3174 5.2393 5.3174
14:31:15 5.2414 5.3174 5.2393 5.3955
14:31:16 5.3195 5.3174 5.2393 5.3955
14:31:17 5.2414 5.3174 5.2393 5.3174
14:31:18 5.2414 5.3174 5.2393 5.3174
14:31:19 5.2414 5.2393 5.2393 5.2393
14:31:20 5.2414 5.2393 5.2393 5.3174
14:31:21 5.2414 5.3174 5.2393 5.3174
14:31:22 5.2414 5.2393 5.2393 5.2393
14:31:23 5.2414 5.2393 5.2393 5.2393
14:31:24 5.2414 5.3174 5.2393 5.3174
14:31:25 5.2414 5.3174 5.2393 5.2393
14:31:26 5.2414 5.2393 5.2393 5.2393
14:31:27 5.3195 5.2393 5.2393 5.2393
14:31:28 5.3195 5.2393 5.2393 5.2393
14:31:29 5.3195 5.3174 5.3174 5.2393
14:31:30 5.2414 5.3174 5.2393 5.2393
14:31:31 5.3195 5.3174 5.2393 5.2393
14:31:32 5.3195 5.2393 5.2393 5.2393
14:31:33 5.3195 5.2393 5.2393 5.2393
14:31:34 5.2414 5.2393 5.2393 5.2393
14:31:35 5.3195 5.2393 5.2393 5.2393
14:31:36 5.3195 5.2393 5.2393 5.2393
14:31:37 5.3195 5.2393 5.2393 5.2393

14:31:38 5.3195 5.2393 5.2393 5.2393
14:31:39 5.2414 5.3174 5.2393 5.2393
14:31:40 5.3195 5.2393 5.2393 5.2393
14:31:41 5.3195 5.3174 5.2393 5.2393
14:31:42 5.3195 5.2393 5.2393 5.2393
14:31:43 5.3195 5.2393 5.2393 5.2393
14:31:44 5.3195 5.2393 5.2393 5.2393
14:31:45 5.2414 5.2393 5.2393 5.2393
14:31:46 5.3195 5.2393 5.2393 5.2393
14:31:47 5.2414 5.2393 5.2393 5.2393
14:31:48 5.3195 5.2393 5.2393 5.2393
14:31:49 5.3195 5.2393 5.2393 5.2393
14:31:50 5.2414 5.2393 5.2393 5.2393
14:31:51 5.3195 5.2393 5.2393 5.2393
14:31:52 5.3195 5.2393 5.2393 5.2393
14:31:53 5.3195 5.2393 5.2393 5.2393
14:31:54 5.3195 5.2393 5.2393 5.2393
14:31:55 5.2414 5.3174 5.2393 5.2393
14:31:56 5.3195 5.2393 5.2393 5.2393
14:31:57 5.2414 5.2393 5.2393 5.2393
14:31:58 5.3195 5.2393 5.2393 5.2393
14:31:59 5.3195 5.2393 5.2393 5.2393
14:32:00 5.2414 5.2393 5.2393 5.2393
14:32:01 5.3195 5.2393 5.2393 5.2393
14:32:02 5.3195 5.2393 5.2393 5.2393
14:32:03 5.2414 5.3174 5.2393 5.2393
14:32:04 5.2414 5.2393 5.2393 5.2393
14:32:05 5.3195 5.2393 5.2393 5.2393
14:32:06 5.2414 5.2393 5.2393 5.2393
14:32:07 5.3195 5.2393 5.2393 5.2393
14:32:08 5.2414 5.2393 5.2393 5.2393
14:32:09 5.3195 5.2393 5.2393 5.2393
14:32:10 5.2414 5.2393 5.2393 5.2393
14:32:11 5.3195 5.2393 5.2393 5.2393
14:32:12 5.3195 5.2393 5.2393 5.2393
14:32:13 5.3195 5.2393 5.2393 5.2393
14:32:14 5.2414 5.2393 5.2393 5.2393
14:32:15 5.3195 5.2393 5.2393 5.2393
14:32:16 5.3195 5.2393 5.2393 5.2393
14:32:17 5.3195 5.2393 5.2393 5.2393
14:32:18 5.2414 5.2393 5.2393 5.2393
14:32:19 5.2414 5.3174 5.2393 5.2393
14:32:20 5.2414 5.2393 5.2393 5.2393
14:32:21 5.3195 5.2393 5.2393 5.2393
14:32:22 5.3195 5.2393 5.1611 5.2393
14:32:23 5.3195 5.2393 5.2393 5.2393
14:32:24 5.3195 5.2393 5.2393 5.2393
14:32:25 5.2414 5.2393 5.1611 5.2393

14:32:26 5.3195 5.2393 5.2393 5.2393
14:32:27 5.2414 5.2393 5.1611 5.2393
14:32:28 5.2414 5.2393 5.1611 5.2393
14:32:29 5.3195 5.2393 5.2393 5.2393
14:32:30 5.2414 5.2393 5.1611 5.2393
14:32:31 5.2414 5.2393 5.1611 5.2393
14:32:32 5.2414 5.2393 5.1611 5.2393
14:32:33 5.2414 5.3174 5.2393 5.2393
14:32:34 5.2414 5.2393 5.1611 5.2393
14:32:35 5.2414 5.2393 5.1611 5.2393
14:32:36 5.2414 5.2393 5.1611 5.2393
14:32:37 5.2414 5.2393 5.2393 5.2393
14:32:38 5.2414 5.2393 5.1611 5.2393
14:32:39 5.2414 5.2393 5.1611 5.2393
14:32:40 5.2414 5.2393 5.1611 5.2393
14:32:41 5.2414 5.2393 5.2393 5.2393
14:32:42 5.2414 5.2393 5.2393 5.2393
14:32:43 5.2414 5.2393 5.2393 5.2393
14:32:44 5.2414 5.2393 5.1611 5.2393
14:32:45 5.2414 5.2393 5.2393 5.2393
14:32:46 5.2414 5.2393 5.2393 5.2393
14:32:47 5.2414 5.2393 5.1611 5.2393
14:32:48 5.2414 5.2393 5.1611 5.2393
14:32:49 5.2414 5.2393 5.2393 5.2393
14:32:50 5.2414 5.2393 5.1611 5.2393
14:32:51 5.2414 5.2393 5.1611 5.2393
14:32:52 5.2414 5.2393 5.1611 5.2393
14:32:53 5.2414 5.2393 5.1611 5.2393
14:32:54 5.2414 5.2393 5.1611 5.2393
14:32:55 5.2414 5.2393 5.1611 5.2393
14:32:56 5.1633 5.2393 5.2393 5.2393
14:32:57 5.2414 5.2393 5.2393 5.2393
14:32:58 5.2414 5.2393 5.2393 5.2393
14:32:59 5.2414 5.2393 5.1611 5.2393

A.5: Trial 2 Data

14:41:53	5.0070	5.0830	5.0830	5.0049					
14:41:53	5.0851	5.0830	5.0830	5.1611					
14:41:54	5.0851	5.0830	5.0830	5.0830					
14:41:55	5.0851	5.0830	5.0830	5.0830					
14:41:56	5.0851	5.1611	5.1611	5.0830					
14:41:57	5.0851	5.1611	5.1611	5.0830	Max Values from rows 1-50				
14:41:58	5.0851	5.1611	5.0830	5.0830	5.1633	5.2393	5.4736	5.3955	
14:41:59	5.0851	5.0830	5.0830	5.0830					
14:42:00	5.0851	5.1611	5.1611	5.0830	Min Values from rows 1-50				
14:42:01	5.0851	5.0830	5.1611	5.1611	5.0070	5.0830	5.0830	5.0049	
14:42:02	5.0851	5.0830	5.0830	5.0830					
14:42:03	5.0851	5.0830	5.1611	5.0830	Avg Value of Maxes	F		C	
14:42:04	5.0851	5.0830	5.1611	5.0830	5.3179	88.63208		31.46227	
14:42:05	5.0851	5.1611	5.1611	5.1611					
14:42:06	5.0851	5.0830	5.2393	5.1611	Avg Value of Mins	F		C	
14:42:07	5.0851	5.0830	5.2393	5.1611	5.0445	84.07458		28.93032	
14:42:08	5.0851	5.0830	5.2393	5.2393					
14:42:09	5.0851	5.0830	5.2393	5.2393	Av. Max - Av. Min	F		C	
14:42:10	5.0851	5.0830	5.2393	5.2393	0.2735	4.5575		2.531944	
14:42:11	5.0851	5.1611	5.1611	5.1611					
14:42:12	5.0851	5.0830	5.2393	5.2393	Average Temp change in Fahrenheit				
14:42:13	5.0851	5.1611	5.2393	5.2393	4.5575				
14:42:14	5.0851	5.0830	5.2393	5.2393					
14:42:15	5.0851	5.1611	5.3174	5.1611	Temperature Change in Celcius				
14:42:16	5.0851	5.0830	5.2393	5.1611	2.531944				
14:42:17	5.0851	5.0830	5.2393	5.2393					
14:42:18	5.0851	5.0830	5.2393	5.2393	Power Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:42:19	5.0851	5.0830	5.3174	5.2393	2.513917 Watts	time			
14:42:20	5.0851	5.1611	5.3174	5.2393					
14:42:21	5.0851	5.0830	5.3955	5.2393	Energy Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:42:22	5.1633	5.0830	5.3174	5.2393	125.6959 Joules				
14:42:23	5.0851	5.1611	5.3174	5.3174					
14:42:24	5.0851	5.1611	5.3955	5.2393					
14:42:25	5.1633	5.1611	5.3174	5.2393					
14:42:26	5.0851	5.1611	5.3955	5.2393					
14:42:27	5.0851	5.1611	5.3955	5.3955					
14:42:28	5.1633	5.2393	5.3955	5.3955					
14:42:29	5.0851	5.2393	5.3955	5.3174					
14:42:30	5.1633	5.2393	5.3955	5.3955					
14:42:31	5.0851	5.2393	5.3955	5.3955					
14:42:32	5.0851	5.1611	5.4736	5.3955					
14:42:33	5.0851	5.2393	5.3955	5.3955					
14:42:34	5.0851	5.1611	5.4736	5.3955					
14:42:35	5.1633	5.2393	5.4736	5.3955					
14:42:36	5.1633	5.2393	5.3955	5.3955					
14:42:37	5.1633	5.2393	5.4736	5.3955					

14:42:38 5.1633 5.2393 5.4736 5.3955
14:42:39 5.1633 5.1611 5.4736 5.3955
14:42:40 5.1633 5.1611 5.4736 5.3955
14:42:41 5.1633 5.2393 5.4736 5.3955
14:42:42 5.1633 5.2393 5.4736 5.3955
14:42:43 5.1633 5.1611 5.4736 5.3955
14:42:44 5.1633 5.1611 5.4736 5.3955
14:42:45 5.1633 5.2393 5.4736 5.3955
14:42:46 5.1633 5.2393 5.4736 5.3955
14:42:47 5.1633 5.3174 5.4736 5.3955
14:42:48 5.1633 5.2393 5.4736 5.3955
14:42:49 5.1633 5.2393 5.4736 5.3955
14:42:50 5.1633 5.2393 5.4736 5.3955
14:42:51 5.2414 5.2393 5.5518 5.3955
14:42:52 5.1633 5.2393 5.4736 5.3955
14:42:53 5.1633 5.2393 5.4736 5.3955
14:42:54 5.2414 5.3174 5.4736 5.3955
14:42:55 5.3195 5.2393 5.4736 5.4736
14:42:56 5.3195 5.2393 5.4736 5.3955
14:42:57 5.3195 5.2393 5.4736 5.3955
14:42:58 5.3195 5.2393 5.4736 5.3955
14:42:59 5.3195 5.2393 5.4736 5.3955
14:43:00 5.3195 5.2393 5.4736 5.3955
14:43:01 5.3195 5.3174 5.4736 5.3955
14:43:02 5.2414 5.2393 5.4736 5.4736
14:43:03 5.2414 5.2393 5.4736 5.4736
14:43:04 5.3195 5.2393 5.3955 5.4736
14:43:05 5.3195 5.2393 5.3955 5.4736
14:43:06 5.3195 5.2393 5.3955 5.4736
14:43:07 5.3195 5.2393 5.3955 5.3955
14:43:08 5.3195 5.2393 5.3955 5.3955
14:43:09 5.2414 5.3174 5.3955 5.3955
14:43:10 5.3195 5.2393 5.3955 5.3955
14:43:11 5.2414 5.3174 5.3955 5.3955
14:43:12 5.2414 5.3174 5.3955 5.4736
14:43:13 5.2414 5.3174 5.3955 5.4736
14:43:14 5.2414 5.2393 5.3955 5.3955
14:43:15 5.3195 5.2393 5.3955 5.4736
14:43:16 5.2414 5.3174 5.3955 5.4736
14:43:17 5.2414 5.3174 5.3955 5.4736
14:43:18 5.2414 5.3174 5.3955 5.3955
14:43:19 5.3195 5.2393 5.3955 5.3955
14:43:20 5.3195 5.3174 5.3955 5.4736
14:43:21 5.3195 5.2393 5.3955 5.4736
14:43:22 5.3195 5.2393 5.3955 5.3955
14:43:23 5.2414 5.3174 5.3955 5.4736
14:43:24 5.2414 5.2393 5.3955 5.3955
14:43:25 5.2414 5.3174 5.3955 5.4736

14:43:26 5.2414 5.3174 5.3955 5.4736
14:43:27 5.2414 5.2393 5.3955 5.3955
14:43:28 5.2414 5.3174 5.3955 5.4736
14:43:29 5.2414 5.3174 5.3955 5.3955
14:43:30 5.2414 5.3174 5.3955 5.3955
14:43:31 5.2414 5.3174 5.3955 5.3955
14:43:32 5.2414 5.2393 5.3955 5.3955
14:43:33 5.2414 5.3174 5.3955 5.4736
14:43:34 5.2414 5.3174 5.3955 5.4736
14:43:35 5.3195 5.2393 5.3955 5.3955
14:43:36 5.2414 5.3174 5.3955 5.4736
14:43:37 5.2414 5.3174 5.3955 5.3955
14:43:38 5.2414 5.3174 5.3955 5.3955
14:43:39 5.2414 5.3174 5.3955 5.3955
14:43:40 5.2414 5.3174 5.3955 5.3955
14:43:41 5.2414 5.3174 5.3955 5.3955
14:43:42 5.2414 5.3174 5.3955 5.3955
14:43:43 5.3195 5.3955 5.3955 5.4736
14:43:44 5.2414 5.3174 5.3955 5.4736
14:43:45 5.3195 5.3955 5.3955 5.4736
14:43:46 5.2414 5.3174 5.3955 5.4736
14:43:47 5.3195 5.3955 5.3955 5.4736
14:43:48 5.2414 5.3174 5.3955 5.4736
14:43:49 5.3195 5.3174 5.3955 5.3955
14:43:50 5.3195 5.3955 5.3174 5.3955
14:43:51 5.2414 5.3955 5.3955 5.3955
14:43:52 5.2414 5.3174 5.3955 5.3955
14:43:53 5.3195 5.3174 5.3955 5.3955
14:43:54 5.3195 5.3174 5.3174 5.3955
14:43:55 5.2414 5.3174 5.3955 5.3955
14:43:56 5.2414 5.3174 5.3955 5.3955
14:43:57 5.3195 5.3955 5.3955 5.3955
14:43:58 5.3976 5.3174 5.3174 5.3955
14:43:59 5.2414 5.3174 5.3955 5.3955
14:44:00 5.2414 5.3955 5.3955 5.3955
14:44:01 5.3976 5.3174 5.3174 5.3174
14:44:02 5.3195 5.3174 5.3174 5.3955
14:44:03 5.3195 5.3955 5.3174 5.3955
14:44:04 5.2414 5.3174 5.3955 5.3955
14:44:05 5.3195 5.3174 5.3174 5.3955
14:44:06 5.3195 5.3174 5.3174 5.3174
14:44:07 5.3195 5.3955 5.3174 5.3174
14:44:08 5.3195 5.3174 5.3174 5.3955
14:44:09 5.3195 5.3174 5.2393 5.3955
14:44:10 5.2414 5.3174 5.2393 5.3955
14:44:11 5.3195 5.3174 5.3174 5.3174
14:44:12 5.3195 5.3174 5.3955 5.3955
14:44:13 5.2414 5.3955 5.3955 5.3955

14:44:14 5.2414 5.3955 5.3955 5.3955
14:44:15 5.2414 5.3174 5.2393 5.3955
14:44:16 5.2414 5.3174 5.3174 5.3174
14:44:17 5.2414 5.3955 5.2393 5.3955
14:44:18 5.2414 5.3174 5.2393 5.3955
14:44:19 5.2414 5.3174 5.2393 5.3955
14:44:20 5.2414 5.3174 5.3174 5.3955
14:44:21 5.3195 5.3174 5.2393 5.3174
14:44:22 5.2414 5.3174 5.2393 5.3955
14:44:23 5.2414 5.3174 5.2393 5.3955
14:44:24 5.2414 5.3174 5.3174 5.3174
14:44:25 5.2414 5.3174 5.2393 5.3955
14:44:26 5.2414 5.3174 5.2393 5.3955
14:44:27 5.2414 5.3174 5.2393 5.3955
14:44:28 5.2414 5.3174 5.2393 5.3955
14:44:29 5.3195 5.3174 5.2393 5.3174
14:44:30 5.2414 5.3174 5.2393 5.3955
14:44:31 5.2414 5.3174 5.2393 5.3955
14:44:32 5.2414 5.3174 5.2393 5.3955
14:44:33 5.2414 5.3174 5.2393 5.3955
14:44:34 5.2414 5.3174 5.2393 5.3955
14:44:35 5.2414 5.3174 5.2393 5.3955
14:44:36 5.2414 5.3174 5.2393 5.3955
14:44:37 5.2414 5.3174 5.3174 5.3174
14:44:38 5.2414 5.3174 5.2393 5.3955
14:44:39 5.2414 5.3174 5.3174 5.3174
14:44:40 5.2414 5.3174 5.2393 5.3955
14:44:41 5.2414 5.3174 5.2393 5.3955
14:44:42 5.2414 5.3174 5.2393 5.3955
14:44:43 5.2414 5.3174 5.2393 5.3955
14:44:44 5.2414 5.3174 5.2393 5.3174
14:44:45 5.3195 5.3174 5.2393 5.3955
14:44:46 5.2414 5.3955 5.2393 5.3955
14:44:47 5.2414 5.3174 5.2393 5.3174
14:44:48 5.2414 5.3955 5.2393 5.3955
14:44:49 5.2414 5.3174 5.2393 5.3955
14:44:50 5.2414 5.3174 5.2393 5.3955
14:44:51 5.2414 5.3174 5.2393 5.3174
14:44:52 5.2414 5.3174 5.2393 5.3955
14:44:53 5.2414 5.3174 5.2393 5.3174
14:44:54 5.3195 5.3174 5.2393 5.3174
14:44:55 5.2414 5.2393 5.2393 5.3174
14:44:56 5.2414 5.3174 5.2393 5.3955
14:44:57 5.2414 5.3174 5.2393 5.3955
14:44:58 5.2414 5.2393 5.2393 5.3174
14:44:59 5.2414 5.3174 5.3174 5.3174
14:45:00 5.2414 5.3174 5.2393 5.3174
14:45:01 5.2414 5.3174 5.2393 5.3955

14:45:02 5.2414 5.3174 5.2393 5.3174
14:45:03 5.2414 5.2393 5.2393 5.3174
14:45:04 5.3195 5.3174 5.2393 5.3174
14:45:05 5.2414 5.3174 5.3174 5.2393
14:45:06 5.2414 5.3174 5.2393 5.2393
14:45:07 5.2414 5.2393 5.2393 5.3955
14:45:08 5.2414 5.3174 5.2393 5.2393
14:45:09 5.3195 5.2393 5.2393 5.2393
14:45:10 5.2414 5.3174 5.2393 5.2393
14:45:11 5.2414 5.2393 5.2393 5.2393
14:45:12 5.2414 5.2393 5.2393 5.2393
14:45:13 5.2414 5.2393 5.2393 5.2393
14:45:14 5.2414 5.3174 5.2393 5.2393
14:45:15 5.2414 5.3174 5.2393 5.2393
14:45:16 5.2414 5.3174 5.2393 5.2393
14:45:17 5.3195 5.2393 5.2393 5.2393
14:45:18 5.2414 5.2393 5.2393 5.2393
14:45:19 5.2414 5.2393 5.2393 5.2393
14:45:20 5.3195 5.2393 5.2393 5.2393
14:45:21 5.2414 5.2393 5.2393 5.2393
14:45:22 5.2414 5.2393 5.2393 5.2393
14:45:23 5.2414 5.2393 5.2393 5.2393
14:45:24 5.3195 5.2393 5.2393 5.2393
14:45:25 5.3195 5.2393 5.2393 5.2393
14:45:26 5.2414 5.3174 5.2393 5.2393
14:45:27 5.2414 5.2393 5.2393 5.2393
14:45:28 5.3195 5.2393 5.2393 5.2393
14:45:29 5.2414 5.2393 5.2393 5.2393
14:45:30 5.2414 5.2393 5.2393 5.2393
14:45:31 5.2414 5.2393 5.2393 5.2393
14:45:32 5.2414 5.2393 5.2393 5.2393
14:45:33 5.2414 5.2393 5.2393 5.2393
14:45:34 5.2414 5.3174 5.2393 5.2393
14:45:35 5.2414 5.3174 5.2393 5.2393
14:45:36 5.2414 5.2393 5.2393 5.2393
14:45:37 5.2414 5.2393 5.2393 5.2393
14:45:38 5.2414 5.2393 5.2393 5.2393
14:45:39 5.2414 5.2393 5.2393 5.2393
14:45:40 5.2414 5.2393 5.2393 5.2393
14:45:41 5.3195 5.2393 5.2393 5.2393
14:45:42 5.2414 5.2393 5.2393 5.2393
14:45:43 5.2414 5.2393 5.1611 5.2393
14:45:44 5.3195 5.2393 5.2393 5.2393
14:45:45 5.3195 5.2393 5.2393 5.2393
14:45:46 5.2414 5.2393 5.2393 5.2393
14:45:47 5.2414 5.3174 5.2393 5.2393
14:45:48 5.2414 5.2393 5.2393 5.2393
14:45:49 5.3195 5.2393 5.2393 5.2393

14:45:50 5.2414 5.2393 5.2393 5.2393
14:45:51 5.2414 5.2393 5.2393 5.2393
14:31:40 5.3195 5.2393 5.2393 5.2393
14:31:41 5.3195 5.3174 5.2393 5.2393
14:31:42 5.3195 5.2393 5.2393 5.2393
14:31:43 5.3195 5.2393 5.2393 5.2393
14:31:44 5.3195 5.2393 5.2393 5.2393
14:31:45 5.2414 5.2393 5.2393 5.2393
14:31:46 5.3195 5.2393 5.2393 5.2393
14:31:47 5.2414 5.2393 5.2393 5.2393
14:31:48 5.3195 5.2393 5.2393 5.2393
14:31:49 5.3195 5.2393 5.2393 5.2393
14:31:50 5.2414 5.2393 5.2393 5.2393
14:31:51 5.3195 5.2393 5.2393 5.2393
14:31:52 5.3195 5.2393 5.2393 5.2393
14:31:53 5.3195 5.2393 5.2393 5.2393
14:31:54 5.3195 5.2393 5.2393 5.2393
14:31:55 5.2414 5.3174 5.2393 5.2393
14:31:56 5.3195 5.2393 5.2393 5.2393
14:31:57 5.2414 5.2393 5.2393 5.2393
14:31:58 5.3195 5.2393 5.2393 5.2393
14:31:59 5.3195 5.2393 5.2393 5.2393
14:32:00 5.2414 5.2393 5.2393 5.2393
14:32:01 5.3195 5.2393 5.2393 5.2393
14:32:02 5.3195 5.2393 5.2393 5.2393
14:32:03 5.2414 5.3174 5.2393 5.2393
14:32:04 5.2414 5.2393 5.2393 5.2393
14:32:05 5.3195 5.2393 5.2393 5.2393
14:32:06 5.2414 5.2393 5.2393 5.2393
14:32:07 5.3195 5.2393 5.2393 5.2393
14:32:08 5.2414 5.2393 5.2393 5.2393
14:32:09 5.3195 5.2393 5.2393 5.2393
14:32:10 5.2414 5.2393 5.2393 5.2393
14:32:11 5.3195 5.2393 5.2393 5.2393
14:32:12 5.3195 5.2393 5.2393 5.2393
14:32:13 5.3195 5.2393 5.2393 5.2393
14:32:14 5.2414 5.2393 5.2393 5.2393
14:32:15 5.3195 5.2393 5.2393 5.2393
14:32:16 5.3195 5.2393 5.2393 5.2393
14:32:17 5.3195 5.2393 5.2393 5.2393
14:32:18 5.2414 5.2393 5.2393 5.2393
14:32:19 5.2414 5.3174 5.2393 5.2393
14:32:20 5.2414 5.2393 5.2393 5.2393
14:32:21 5.3195 5.2393 5.2393 5.2393
14:32:22 5.3195 5.2393 5.1611 5.2393
14:32:23 5.3195 5.2393 5.2393 5.2393
14:32:24 5.3195 5.2393 5.2393 5.2393
14:32:25 5.2414 5.2393 5.1611 5.2393

14:32:26 5.3195 5.2393 5.2393 5.2393
14:32:27 5.2414 5.2393 5.1611 5.2393
14:32:28 5.2414 5.2393 5.1611 5.2393
14:32:29 5.3195 5.2393 5.2393 5.2393
14:32:30 5.2414 5.2393 5.1611 5.2393
14:32:31 5.2414 5.2393 5.1611 5.2393
14:32:32 5.2414 5.2393 5.1611 5.2393
14:32:33 5.2414 5.3174 5.2393 5.2393
14:32:34 5.2414 5.2393 5.1611 5.2393
14:32:35 5.2414 5.2393 5.1611 5.2393
14:32:36 5.2414 5.2393 5.1611 5.2393
14:32:37 5.2414 5.2393 5.2393 5.2393
14:32:38 5.2414 5.2393 5.1611 5.2393
14:32:39 5.2414 5.2393 5.1611 5.2393
14:32:40 5.2414 5.2393 5.1611 5.2393
14:32:41 5.2414 5.2393 5.2393 5.2393
14:32:42 5.2414 5.2393 5.2393 5.2393
14:32:43 5.2414 5.2393 5.2393 5.2393
14:32:44 5.2414 5.2393 5.1611 5.2393
14:32:45 5.2414 5.2393 5.2393 5.2393
14:32:46 5.2414 5.2393 5.2393 5.2393
14:32:47 5.2414 5.2393 5.1611 5.2393
14:32:48 5.2414 5.2393 5.1611 5.2393
14:32:49 5.2414 5.2393 5.2393 5.2393
14:32:50 5.2414 5.2393 5.1611 5.2393
14:32:51 5.2414 5.2393 5.1611 5.2393
14:32:52 5.2414 5.2393 5.1611 5.2393
14:32:53 5.2414 5.2393 5.1611 5.2393
14:32:54 5.2414 5.2393 5.1611 5.2393
14:32:55 5.2414 5.2393 5.1611 5.2393
14:32:56 5.1633 5.2393 5.2393 5.2393
14:32:57 5.2414 5.2393 5.2393 5.2393
14:32:58 5.2414 5.2393 5.2393 5.2393
14:32:59 5.2414 5.2393 5.1611 5.2393

A.6: Trial 3 Data

14:49:05	5.0851	5.2393	5.1611	5.1611					
14:49:06	5.0851	5.2393	5.1611	5.2393					
14:49:07	5.0851	5.1611	5.1611	5.1611					
14:49:08	5.0851	5.1611	5.0830	5.2393					
14:49:09	5.1633	5.2393	5.1611	5.2393					
14:49:10	5.0851	5.2393	5.1611	5.2393	Max Values from rows 1-50				
14:49:11	5.1633	5.2393	5.1611	5.2393	5.2414	5.3174	5.5518	5.4736	
14:49:12	5.0851	5.1611	5.2393	5.2393					
14:49:13	5.0851	5.1611	5.2393	5.2393	Min Values from rows 1-50				
14:49:14	5.0851	5.2393	5.2393	5.2393	5.0851	5.1611	5.0830	5.1611	
14:49:15	5.0851	5.2393	5.2393	5.2393					
14:49:16	5.0851	5.1611	5.2393	5.2393	Avg Value of Maxes	F		C	
14:49:17	5.0851	5.2393	5.2393	5.2393	5.3961	89.93417		32.18565	
14:49:18	5.0851	5.1611	5.2393	5.2393					
14:49:19	5.0851	5.2393	5.2393	5.2393	Avg Value of Mins	F		C	
14:49:20	5.0851	5.2393	5.2393	5.2393	5.1226	85.37625		29.65347	
14:49:21	5.0851	5.2393	5.2393	5.2393					
14:49:22	5.0851	5.2393	5.2393	5.2393	Av. Max - Av. Min	F		C	
14:49:23	5.0851	5.1611	5.2393	5.2393	0.2735	4.557917		2.532176	
14:49:24	5.0851	5.2393	5.2393	5.2393					
14:49:25	5.0851	5.2393	5.2393	5.2393	Average Temp change in Fahrenheit				
14:49:26	5.0851	5.2393	5.2393	5.2393	4.557917				
14:49:27	5.1633	5.1611	5.2393	5.2393					
14:49:28	5.1633	5.1611	5.3174	5.3174	Temperature Change in Celcius				
14:49:29	5.1633	5.2393	5.3174	5.2393	2.532176				
14:49:30	5.1633	5.2393	5.3955	5.2393					
14:49:31	5.1633	5.2393	5.3955	5.3174	Power Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:49:32	5.0851	5.2393	5.3955	5.3955	2.514147	Watts	time		
14:49:33	5.1633	5.2393	5.3955	5.3955					
14:49:34	5.0851	5.2393	5.3955	5.3955	Energy Absorbed	(Mass*Specific Heat*Change in Temp (C))			
14:49:35	5.1633	5.2393	5.3955	5.3174	125.7073	Joules			
14:49:36	5.1633	5.2393	5.3955	5.3955					
14:49:37	5.1633	5.2393	5.3955	5.3955					
14:49:38	5.2414	5.2393	5.3955	5.3955					
14:49:39	5.1633	5.2393	5.3955	5.3955					
14:49:40	5.1633	5.2393	5.3955	5.3955					
14:49:41	5.1633	5.2393	5.4736	5.3955					
14:49:42	5.1633	5.2393	5.4736	5.3955					
14:49:43	5.1633	5.2393	5.4736	5.3955					
14:49:44	5.1633	5.2393	5.4736	5.3955					
14:49:45	5.1633	5.2393	5.4736	5.3955					
14:49:46	5.1633	5.2393	5.4736	5.3955					
14:49:47	5.1633	5.2393	5.4736	5.3955					
14:49:48	5.1633	5.2393	5.4736	5.3955					
14:49:49	5.1633	5.2393	5.4736	5.3955					
14:49:50	5.1633	5.2393	5.4736	5.3955					

14:49:51 5.1633 5.2393 5.4736 5.3955
14:49:52 5.1633 5.2393 5.5518 5.3955
14:49:53 5.2414 5.3174 5.4736 5.3955
14:49:54 5.2414 5.3174 5.5518 5.4736
14:49:55 5.2414 5.2393 5.5518 5.4736
14:49:56 5.2414 5.3174 5.5518 5.4736
14:49:57 5.2414 5.2393 5.6299 5.4736
14:49:58 5.2414 5.2393 5.6299 5.4736
14:49:59 5.2414 5.2393 5.6299 5.4736
14:50:00 5.2414 5.2393 5.6299 5.4736
14:50:01 5.3195 5.2393 5.6299 5.5518
14:50:02 5.3195 5.3174 5.5518 5.4736
14:50:03 5.3195 5.2393 5.6299 5.5518
14:50:04 5.3195 5.3174 5.6299 5.5518
14:50:05 5.3195 5.2393 5.6299 5.5518
14:50:06 5.3195 5.3174 5.6299 5.5518
14:50:07 5.3195 5.3174 5.5518 5.4736
14:50:08 5.3195 5.3174 5.5518 5.4736
14:50:09 5.3195 5.3174 5.5518 5.4736
14:50:10 5.3195 5.3174 5.5518 5.4736
14:50:11 5.3195 5.3174 5.5518 5.4736
14:50:12 5.3195 5.3174 5.5518 5.4736
14:50:13 5.3195 5.3174 5.5518 5.4736
14:50:14 5.3195 5.3174 5.5518 5.4736
14:50:15 5.3195 5.3174 5.5518 5.4736
14:50:16 5.3195 5.3174 5.5518 5.4736
14:50:17 5.3195 5.3174 5.5518 5.4736
14:50:18 5.3195 5.3174 5.5518 5.4736
14:50:19 5.3195 5.3174 5.5518 5.4736
14:50:20 5.3195 5.3174 5.5518 5.4736
14:50:21 5.3195 5.3174 5.5518 5.4736
14:50:22 5.3195 5.3174 5.5518 5.4736
14:50:23 5.3195 5.3174 5.5518 5.4736
14:50:24 5.3195 5.3174 5.5518 5.4736
14:50:25 5.3195 5.3174 5.5518 5.4736
14:50:26 5.3195 5.3174 5.5518 5.4736
14:50:27 5.3195 5.3174 5.4736 5.5518
14:50:28 5.3195 5.3174 5.4736 5.5518
14:50:29 5.3195 5.3174 5.4736 5.5518
14:50:30 5.3195 5.3174 5.4736 5.5518
14:50:31 5.3195 5.3174 5.4736 5.5518
14:50:32 5.3195 5.3955 5.4736 5.5518
14:50:33 5.3195 5.3174 5.4736 5.5518
14:50:34 5.3195 5.3955 5.4736 5.5518
14:50:35 5.3195 5.3174 5.4736 5.5518
14:50:36 5.3195 5.3955 5.4736 5.5518
14:50:37 5.3195 5.3955 5.4736 5.5518
14:50:38 5.3195 5.3174 5.4736 5.5518

14:50:39 5.3195 5.3955 5.3955 5.4736
14:50:40 5.3195 5.3955 5.3955 5.4736
14:50:41 5.3195 5.3955 5.4736 5.5518
14:50:42 5.3195 5.3955 5.3955 5.5518
14:50:43 5.3195 5.3955 5.3955 5.5518
14:50:44 5.3195 5.3955 5.4736 5.5518
14:50:45 5.3195 5.3955 5.3955 5.4736
14:50:46 5.3195 5.3955 5.4736 5.4736
14:50:47 5.3195 5.3955 5.3955 5.5518
14:50:48 5.3195 5.3955 5.3955 5.4736
14:50:49 5.3195 5.3955 5.3955 5.5518
14:50:50 5.3195 5.3955 5.3955 5.4736
14:50:51 5.3195 5.3955 5.3955 5.4736
14:50:52 5.3195 5.3955 5.3955 5.4736
14:50:53 5.3195 5.3955 5.3955 5.4736
14:50:54 5.3195 5.3955 5.3955 5.5518
14:50:55 5.3195 5.3955 5.3955 5.4736
14:50:56 5.3195 5.3955 5.3955 5.4736
14:50:57 5.3195 5.3955 5.3955 5.5518
14:50:58 5.3195 5.3955 5.3955 5.4736
14:50:59 5.3976 5.3955 5.3955 5.4736
14:51:00 5.3976 5.3955 5.3955 5.4736
14:51:01 5.3195 5.3955 5.3955 5.4736
14:51:02 5.3195 5.3955 5.3955 5.4736
14:51:03 5.3976 5.3955 5.3955 5.4736
14:51:04 5.3976 5.3955 5.3955 5.4736
14:51:05 5.3976 5.3955 5.3955 5.4736
14:51:06 5.3976 5.3955 5.3955 5.4736
14:51:07 5.3976 5.3955 5.3955 5.4736
14:51:08 5.3976 5.3955 5.3955 5.4736
14:51:09 5.3976 5.3955 5.3955 5.4736
14:51:10 5.3976 5.3955 5.3955 5.4736
14:51:11 5.3976 5.3955 5.3955 5.4736
14:51:12 5.3195 5.3955 5.3955 5.4736
14:51:13 5.3976 5.3955 5.3955 5.4736
14:51:14 5.3195 5.3955 5.3955 5.4736
14:51:15 5.3976 5.3955 5.3955 5.4736
14:51:16 5.3195 5.3955 5.3955 5.3955
14:51:17 5.3976 5.3955 5.3955 5.3955
14:51:18 5.3195 5.3955 5.3955 5.4736
14:51:19 5.3195 5.3955 5.3955 5.4736
14:51:20 5.3976 5.3955 5.3955 5.4736
14:51:21 5.3976 5.3955 5.3955 5.4736
14:51:22 5.3976 5.3955 5.3955 5.3955
14:51:23 5.3976 5.3955 5.3955 5.4736
14:51:24 5.3195 5.3955 5.3955 5.4736
14:51:25 5.3195 5.3955 5.3174 5.3955
14:51:26 5.3976 5.3955 5.3955 5.3955

14:51:27 5.3195 5.3955 5.3174 5.3955
14:51:28 5.3195 5.3955 5.3955 5.4736
14:51:29 5.3195 5.3955 5.3174 5.3955
14:51:30 5.3195 5.3174 5.3174 5.3955
14:51:31 5.3195 5.3955 5.3955 5.3955
14:51:32 5.3195 5.3955 5.3955 5.3955
14:51:33 5.3195 5.3955 5.3955 5.3955
14:51:34 5.3195 5.3955 5.3955 5.4736
14:51:35 5.3195 5.3955 5.3174 5.3955
14:51:36 5.3195 5.3955 5.3955 5.3955
14:51:37 5.3195 5.3955 5.3174 5.3174
14:51:38 5.3195 5.3955 5.3955 5.3955
14:51:39 5.3195 5.3174 5.3174 5.3955
14:51:40 5.3976 5.3955 5.3174 5.3174
14:51:41 5.3195 5.3955 5.3174 5.3174
14:51:42 5.3195 5.3174 5.3174 5.3174
14:51:43 5.3195 5.3174 5.3174 5.3955
14:51:44 5.3195 5.3955 5.3955 5.3955
14:51:45 5.3195 5.3174 5.2393 5.3955
14:51:46 5.2414 5.3174 5.2393 5.3955
14:51:47 5.2414 5.3174 5.3174 5.3955
14:51:48 5.2414 5.3174 5.3174 5.3174
14:51:49 5.2414 5.3955 5.2393 5.3955
14:51:50 5.3195 5.3174 5.3174 5.3955
14:51:51 5.3195 5.3174 5.2393 5.3955
14:51:52 5.3195 5.3955 5.2393 5.3955
14:51:53 5.2414 5.3174 5.2393 5.3955
14:51:54 5.3195 5.3174 5.2393 5.3955
14:51:55 5.2414 5.3174 5.3174 5.3174
14:51:56 5.2414 5.3174 5.3174 5.3174
14:51:57 5.2414 5.3955 5.2393 5.3955
14:51:58 5.3195 5.3174 5.2393 5.3955
14:51:59 5.2414 5.3174 5.2393 5.3955
14:52:00 5.2414 5.3174 5.3174 5.3174
14:52:01 5.2414 5.3174 5.2393 5.3174
14:52:02 5.2414 5.3174 5.2393 5.3955
14:52:03 5.2414 5.3174 5.3174 5.3174
14:52:04 5.2414 5.3174 5.2393 5.3955
14:52:05 5.2414 5.3174 5.2393 5.3955
14:52:06 5.2414 5.3174 5.3174 5.3174
14:52:07 5.3195 5.3174 5.2393 5.3955
14:52:08 5.3195 5.3174 5.2393 5.3174
14:52:09 5.2414 5.3174 5.2393 5.3955
14:52:10 5.2414 5.3174 5.3174 5.3174
14:52:11 5.3195 5.3174 5.2393 5.3955
14:52:12 5.2414 5.3174 5.2393 5.3955
14:52:13 5.2414 5.3955 5.2393 5.3955
14:52:14 5.2414 5.3174 5.2393 5.3955

14:52:15 5.3195 5.3174 5.2393 5.3174
14:52:16 5.3195 5.3174 5.2393 5.3174
14:52:17 5.2414 5.3174 5.2393 5.3174
14:52:18 5.2414 5.2393 5.2393 5.3174
14:52:19 5.3195 5.3174 5.2393 5.3955
14:52:20 5.2414 5.3174 5.2393 5.3174
14:52:21 5.2414 5.3174 5.2393 5.3955
14:52:22 5.3195 5.3174 5.2393 5.3174
14:52:23 5.2414 5.2393 5.2393 5.3174
14:52:24 5.2414 5.2393 5.2393 5.3955
14:52:25 5.3195 5.3174 5.2393 5.3174
14:52:26 5.2414 5.2393 5.2393 5.3174
14:52:27 5.3195 5.3174 5.2393 5.3174
14:52:28 5.3195 5.3174 5.2393 5.3174
14:52:29 5.2414 5.2393 5.2393 5.3174
14:52:30 5.3195 5.2393 5.2393 5.2393
14:52:31 5.3195 5.3174 5.2393 5.3174
14:52:32 5.2414 5.2393 5.2393 5.2393
14:52:33 5.2414 5.3174 5.2393 5.2393
14:52:34 5.2414 5.2393 5.2393 5.2393
14:52:35 5.2414 5.2393 5.2393 5.2393
14:52:36 5.2414 5.2393 5.2393 5.2393
14:52:37 5.2414 5.2393 5.2393 5.2393
14:52:38 5.2414 5.2393 5.2393 5.2393
14:52:39 5.3195 5.2393 5.2393 5.2393
14:52:40 5.2414 5.2393 5.2393 5.2393
14:52:41 5.2414 5.2393 5.2393 5.2393
14:52:42 5.2414 5.3174 5.2393 5.2393
14:52:43 5.2414 5.3174 5.2393 5.2393
14:52:44 5.3195 5.2393 5.2393 5.2393
14:52:45 5.2414 5.2393 5.2393 5.2393
14:52:46 5.2414 5.3174 5.2393 5.2393
14:52:47 5.2414 5.2393 5.2393 5.2393
14:52:48 5.2414 5.2393 5.2393 5.2393
14:52:49 5.2414 5.2393 5.2393 5.2393
14:52:50 5.3195 5.2393 5.2393 5.2393
14:52:51 5.3195 5.2393 5.1611 5.2393
14:52:52 5.2414 5.2393 5.2393 5.2393
14:52:53 5.2414 5.2393 5.2393 5.2393
14:52:54 5.2414 5.2393 5.2393 5.2393
14:52:55 5.2414 5.2393 5.2393 5.2393
14:52:56 5.3195 5.2393 5.2393 5.2393
14:52:57 5.2414 5.2393 5.2393 5.2393
14:52:58 5.3195 5.2393 5.2393 5.2393
14:52:59 5.3195 5.2393 5.2393 5.2393
14:53:00 5.3195 5.2393 5.2393 5.2393
14:53:01 5.2414 5.2393 5.2393 5.2393
14:53:02 5.2414 5.2393 5.2393 5.2393

14:53:03 5.2414 5.2393 5.2393 5.2393
14:53:04 5.2414 5.2393 5.2393 5.2393
14:53:05 5.2414 5.2393 5.2393 5.2393
14:31:41 5.3195 5.3174 5.2393 5.2393
14:31:42 5.3195 5.2393 5.2393 5.2393
14:31:43 5.3195 5.2393 5.2393 5.2393
14:31:44 5.3195 5.2393 5.2393 5.2393
14:31:45 5.2414 5.2393 5.2393 5.2393
14:31:46 5.3195 5.2393 5.2393 5.2393
14:31:47 5.2414 5.2393 5.2393 5.2393
14:31:48 5.3195 5.2393 5.2393 5.2393
14:31:49 5.3195 5.2393 5.2393 5.2393
14:31:50 5.2414 5.2393 5.2393 5.2393
14:31:51 5.3195 5.2393 5.2393 5.2393
14:31:52 5.3195 5.2393 5.2393 5.2393
14:31:53 5.3195 5.2393 5.2393 5.2393
14:31:54 5.3195 5.2393 5.2393 5.2393
14:31:55 5.2414 5.3174 5.2393 5.2393
14:31:56 5.3195 5.2393 5.2393 5.2393
14:31:57 5.2414 5.2393 5.2393 5.2393
14:31:58 5.3195 5.2393 5.2393 5.2393
14:31:59 5.3195 5.2393 5.2393 5.2393
14:32:00 5.2414 5.2393 5.2393 5.2393
14:32:01 5.3195 5.2393 5.2393 5.2393
14:32:02 5.3195 5.2393 5.2393 5.2393
14:32:03 5.2414 5.3174 5.2393 5.2393
14:32:04 5.2414 5.2393 5.2393 5.2393
14:32:05 5.3195 5.2393 5.2393 5.2393
14:32:06 5.2414 5.2393 5.2393 5.2393
14:32:07 5.3195 5.2393 5.2393 5.2393
14:32:08 5.2414 5.2393 5.2393 5.2393
14:32:09 5.3195 5.2393 5.2393 5.2393
14:32:10 5.2414 5.2393 5.2393 5.2393
14:32:11 5.3195 5.2393 5.2393 5.2393
14:32:12 5.3195 5.2393 5.2393 5.2393
14:32:13 5.3195 5.2393 5.2393 5.2393
14:32:14 5.2414 5.2393 5.2393 5.2393
14:32:15 5.3195 5.2393 5.2393 5.2393
14:32:16 5.3195 5.2393 5.2393 5.2393
14:32:17 5.3195 5.2393 5.2393 5.2393
14:32:18 5.2414 5.2393 5.2393 5.2393
14:32:19 5.2414 5.3174 5.2393 5.2393
14:32:20 5.2414 5.2393 5.2393 5.2393
14:32:21 5.3195 5.2393 5.2393 5.2393
14:32:22 5.3195 5.2393 5.1611 5.2393
14:32:23 5.3195 5.2393 5.2393 5.2393
14:32:24 5.3195 5.2393 5.2393 5.2393
14:32:25 5.2414 5.2393 5.1611 5.2393

14:32:26 5.3195 5.2393 5.2393 5.2393
14:32:27 5.2414 5.2393 5.1611 5.2393
14:32:28 5.2414 5.2393 5.1611 5.2393
14:32:29 5.3195 5.2393 5.2393 5.2393
14:32:30 5.2414 5.2393 5.1611 5.2393
14:32:31 5.2414 5.2393 5.1611 5.2393
14:32:32 5.2414 5.2393 5.1611 5.2393
14:32:33 5.2414 5.3174 5.2393 5.2393
14:32:34 5.2414 5.2393 5.1611 5.2393
14:32:35 5.2414 5.2393 5.1611 5.2393
14:32:36 5.2414 5.2393 5.1611 5.2393
14:32:37 5.2414 5.2393 5.2393 5.2393
14:32:38 5.2414 5.2393 5.1611 5.2393
14:32:39 5.2414 5.2393 5.1611 5.2393
14:32:40 5.2414 5.2393 5.1611 5.2393
14:32:41 5.2414 5.2393 5.2393 5.2393
14:32:42 5.2414 5.2393 5.2393 5.2393
14:32:43 5.2414 5.2393 5.2393 5.2393
14:32:44 5.2414 5.2393 5.1611 5.2393
14:32:45 5.2414 5.2393 5.2393 5.2393
14:32:46 5.2414 5.2393 5.2393 5.2393
14:32:47 5.2414 5.2393 5.1611 5.2393
14:32:48 5.2414 5.2393 5.1611 5.2393
14:32:49 5.2414 5.2393 5.2393 5.2393
14:32:50 5.2414 5.2393 5.1611 5.2393
14:32:51 5.2414 5.2393 5.1611 5.2393
14:32:52 5.2414 5.2393 5.1611 5.2393
14:32:53 5.2414 5.2393 5.1611 5.2393
14:32:54 5.2414 5.2393 5.1611 5.2393
14:32:55 5.2414 5.2393 5.1611 5.2393
14:32:56 5.1633 5.2393 5.2393 5.2393
14:32:57 5.2414 5.2393 5.2393 5.2393
14:32:58 5.2414 5.2393 5.2393 5.2393
14:32:59 5.2414 5.2393 5.1611 5.2393

A.7: Trial 4 Data

15:02:19	5.1633	5.1611	5.2393	5.2393					
15:02:20	5.1633	5.2393	5.2393	5.2393					
15:02:21	5.2414	5.2393	5.2393	5.2393					
15:02:22	5.2414	5.2393	5.2393	5.2393					
15:02:23	5.1633	5.2393	5.2393	5.2393					
15:02:24	5.2414	5.2393	5.2393	5.2393	Max Values from rows 1-50				
15:02:25	5.2414	5.2393	5.2393	5.2393	5.3195	5.3174	5.6299	5.5518	
15:02:26	5.1633	5.2393	5.2393	5.2393					
15:02:27	5.2414	5.2393	5.2393	5.2393	Min Values from rows 1-50				
15:02:28	5.1633	5.2393	5.2393	5.2393	5.1633	5.1611	5.2393	5.2393	
15:02:29	5.2414	5.2393	5.2393	5.2393					
15:02:30	5.2414	5.2393	5.3174	5.2393	Avg Value of Maxes	F		C	
15:02:31	5.2414	5.2393	5.2393	5.2393	5.4547	90.91083		32.72824	
15:02:32	5.1633	5.2393	5.2393	5.2393					
15:02:33	5.1633	5.2393	5.3174	5.2393	Avg Value of Mins	F		C	
15:02:34	5.1633	5.2393	5.2393	5.2393	5.2008	86.67917		30.37731	
15:02:35	5.2414	5.2393	5.3174	5.3174					
15:02:36	5.2414	5.2393	5.3174	5.3955	Av. Max - Av. Min	F		C	
15:02:37	5.2414	5.2393	5.3955	5.2393	0.2539	4.231667		2.350926	
15:02:38	5.2414	5.2393	5.3174	5.3955					
15:02:39	5.2414	5.2393	5.3174	5.3955	Average Temp change in Fahrenheit				
15:02:40	5.2414	5.2393	5.3955	5.3955	4.231667				
15:02:41	5.2414	5.2393	5.3955	5.3174					
15:02:42	5.2414	5.3174	5.3955	5.3174	Temperature Change in Celcius				
15:02:43	5.2414	5.2393	5.3955	5.3174	2.350926				
15:02:44	5.2414	5.2393	5.3955	5.3174					
15:02:45	5.2414	5.2393	5.3955	5.3174	Power Absorbed	(Mass*Specific Heat*Change in Temp (C))			
15:02:46	5.2414	5.2393	5.4736	5.3955	2.334187	Watts	time		
15:02:47	5.2414	5.2393	5.3955	5.3955					
15:02:48	5.2414	5.2393	5.3955	5.3955	Energy Absorbed	(Mass*Specific Heat*Change in Temp (C))			
15:02:49	5.3195	5.2393	5.3955	5.3955	116.7094	Joules			
15:02:50	5.2414	5.2393	5.3955	5.3955					
15:02:51	5.3195	5.2393	5.4736	5.3955					
15:02:52	5.2414	5.2393	5.5518	5.3955					
15:02:53	5.3195	5.2393	5.4736	5.3955					
15:02:54	5.2414	5.2393	5.4736	5.3955					
15:02:55	5.3195	5.2393	5.4736	5.3955					
15:02:56	5.2414	5.3174	5.4736	5.3955					
15:02:57	5.3195	5.3174	5.4736	5.3955					
15:02:58	5.2414	5.3174	5.5518	5.3955					
15:02:59	5.3195	5.3174	5.6299	5.4736					
15:03:00	5.2414	5.3174	5.5518	5.4736					
15:03:01	5.3195	5.3174	5.5518	5.4736					
15:03:02	5.3195	5.3174	5.5518	5.4736					
15:03:03	5.3195	5.3174	5.5518	5.4736					
15:03:04	5.3195	5.3174	5.5518	5.4736					

15:03:05 5.3195 5.3174 5.5518 5.4736
15:03:06 5.3195 5.3174 5.5518 5.4736
15:03:07 5.3195 5.3174 5.5518 5.5518
15:03:08 5.3195 5.3174 5.5518 5.5518
15:03:09 5.3195 5.3174 5.5518 5.5518
15:03:10 5.3195 5.3174 5.5518 5.5518
15:03:11 5.3195 5.3174 5.5518 5.5518
15:03:12 5.3195 5.3174 5.7080 5.5518
15:03:13 5.3195 5.3174 5.5518 5.5518
15:03:14 5.3195 5.3174 5.6299 5.5518
15:03:15 5.3195 5.3174 5.6299 5.5518
15:03:16 5.3195 5.3955 5.5518 5.5518
15:03:17 5.3195 5.3174 5.6299 5.5518
15:03:18 5.3195 5.3955 5.5518 5.6299
15:03:19 5.3195 5.3174 5.6299 5.5518
15:03:20 5.3195 5.3174 5.5518 5.5518
15:03:21 5.3195 5.3955 5.5518 5.6299
15:03:22 5.3195 5.3955 5.5518 5.5518
15:03:23 5.3195 5.3955 5.5518 5.6299
15:03:24 5.3195 5.3955 5.5518 5.6299
15:03:25 5.3195 5.3174 5.6299 5.5518
15:03:26 5.3195 5.3955 5.5518 5.6299
15:03:27 5.3195 5.3955 5.6299 5.5518
15:03:28 5.3976 5.3955 5.5518 5.5518
15:03:29 5.3195 5.3955 5.5518 5.5518
15:03:30 5.3195 5.3955 5.5518 5.6299
15:03:31 5.3195 5.3955 5.5518 5.5518
15:03:32 5.3195 5.3955 5.5518 5.6299
15:03:33 5.4758 5.4736 5.5518 5.5518
15:03:34 5.3976 5.4736 5.5518 5.5518
15:03:35 5.3976 5.3955 5.5518 5.6299
15:03:36 5.3976 5.3955 5.6299 5.5518
15:03:37 5.3976 5.3955 5.6299 5.5518
15:03:38 5.3976 5.3955 5.6299 5.5518
15:03:39 5.4758 5.4736 5.5518 5.5518
15:03:40 5.3976 5.3955 5.5518 5.5518
15:03:41 5.3976 5.3955 5.5518 5.5518
15:03:42 5.3976 5.3955 5.4736 5.6299
15:03:43 5.3976 5.3955 5.5518 5.6299
15:03:44 5.3976 5.3955 5.6299 5.5518
15:03:45 5.3976 5.3955 5.4736 5.6299
15:03:46 5.3976 5.3955 5.4736 5.6299
15:03:47 5.4758 5.4736 5.5518 5.5518
15:03:48 5.3976 5.3955 5.4736 5.6299
15:03:49 5.3976 5.3955 5.4736 5.6299
15:03:50 5.3976 5.3955 5.4736 5.5518
15:03:51 5.3976 5.3955 5.4736 5.4736
15:03:52 5.4758 5.4736 5.4736 5.5518

15:03:53 5.3976 5.3955 5.4736 5.4736
15:03:54 5.3976 5.3955 5.4736 5.5518
15:03:55 5.3976 5.3955 5.5518 5.5518
15:03:56 5.3976 5.3955 5.4736 5.4736
15:03:57 5.3976 5.3955 5.4736 5.4736
15:03:58 5.3976 5.3955 5.4736 5.4736
15:03:59 5.3976 5.3955 5.4736 5.4736
15:04:00 5.3976 5.3955 5.5518 5.4736
15:04:01 5.3976 5.3955 5.4736 5.4736
15:04:02 5.3976 5.3955 5.4736 5.4736
15:04:03 5.3976 5.3955 5.4736 5.4736
15:04:04 5.3976 5.3955 5.5518 5.5518
15:04:05 5.3976 5.3955 5.4736 5.5518
15:04:06 5.3976 5.3955 5.4736 5.5518
15:04:07 5.3976 5.3955 5.4736 5.4736
15:04:08 5.3976 5.3955 5.3955 5.4736
15:04:09 5.3976 5.3955 5.3955 5.4736
15:04:10 5.3976 5.4736 5.5518 5.5518
15:04:11 5.4758 5.4736 5.4736 5.4736
15:04:12 5.4758 5.4736 5.4736 5.4736
15:04:13 5.4758 5.4736 5.4736 5.5518
15:04:14 5.3976 5.3955 5.3955 5.4736
15:04:15 5.4758 5.4736 5.4736 5.4736
15:04:16 5.3976 5.3955 5.3955 5.4736
15:04:17 5.3976 5.3955 5.3955 5.4736
15:04:18 5.4758 5.4736 5.4736 5.5518
15:04:19 5.4758 5.4736 5.4736 5.5518
15:04:20 5.4758 5.4736 5.4736 5.5518
15:04:21 5.3976 5.4736 5.4736 5.5518
15:04:22 5.4758 5.4736 5.4736 5.4736
15:04:23 5.3976 5.3955 5.3955 5.4736
15:04:24 5.4758 5.4736 5.4736 5.4736
15:04:25 5.3976 5.3955 5.3955 5.4736
15:04:26 5.4758 5.4736 5.4736 5.5518
15:04:27 5.3976 5.4736 5.4736 5.5518
15:04:28 5.3976 5.3955 5.3955 5.3955
15:04:29 5.3976 5.3955 5.3955 5.3955
15:04:30 5.3976 5.3955 5.3955 5.3955
15:04:31 5.3976 5.3955 5.3955 5.3955
15:04:32 5.3976 5.3955 5.3955 5.4736
15:04:33 5.3976 5.3955 5.3955 5.3955
15:04:34 5.3976 5.4736 5.4736 5.5518
15:04:35 5.3976 5.3955 5.3955 5.3955
15:04:36 5.3976 5.3955 5.3955 5.4736
15:04:37 5.3976 5.3955 5.3955 5.3955
15:04:38 5.3976 5.3955 5.3955 5.3955
15:04:39 5.3976 5.3955 5.3955 5.3955
15:04:40 5.3976 5.4736 5.3955 5.4736

15:04:41 5.3976 5.3955 5.3955 5.3955
15:04:42 5.3976 5.3955 5.3955 5.4736
15:04:43 5.3976 5.3955 5.3955 5.3955
15:04:44 5.3976 5.4736 5.3955 5.4736
15:04:45 5.3976 5.3955 5.3955 5.3955
15:04:46 5.3195 5.3955 5.3955 5.3955
15:04:47 5.3976 5.3955 5.3955 5.3955
15:04:48 5.3976 5.3955 5.3955 5.4736
15:04:49 5.3195 5.3955 5.3955 5.4736
15:04:50 5.3195 5.3955 5.3955 5.3955
15:04:51 5.3976 5.3955 5.3955 5.4736
15:04:52 5.3195 5.3955 5.3955 5.3955
15:04:53 5.3976 5.3955 5.3955 5.3955
15:04:54 5.3976 5.3955 5.3955 5.4736
15:04:55 5.3976 5.3955 5.3955 5.4736
15:04:56 5.3976 5.3955 5.3955 5.4736
15:04:57 5.3976 5.3955 5.3955 5.4736
15:04:58 5.3976 5.3955 5.3955 5.4736
15:04:59 5.3976 5.3955 5.3955 5.3955
15:05:00 5.3195 5.3955 5.3955 5.4736
15:05:01 5.3976 5.3955 5.3955 5.3955
15:05:02 5.3195 5.3955 5.3174 5.4736
15:05:03 5.3976 5.3955 5.3955 5.4736
15:05:04 5.3976 5.4736 5.3174 5.3955
15:05:05 5.3195 5.3955 5.3955 5.4736
15:05:06 5.3195 5.3955 5.3955 5.3955
15:05:07 5.3976 5.3955 5.3955 5.4736
15:05:08 5.3195 5.3955 5.3174 5.3955
15:05:09 5.3976 5.3955 5.3955 5.3955
15:05:10 5.3195 5.3955 5.3955 5.3955
15:05:11 5.3976 5.3955 5.3955 5.3955
15:05:12 5.3195 5.3174 5.3174 5.3955
15:05:13 5.3976 5.3955 5.3955 5.3955
15:05:14 5.3976 5.3955 5.3955 5.3955
15:05:15 5.3976 5.3955 5.3955 5.4736
15:05:16 5.3195 5.3955 5.3955 5.4736
15:05:17 5.3976 5.3955 5.3955 5.4736
15:05:18 5.3195 5.3955 5.3955 5.3955
15:05:19 5.3976 5.3955 5.3955 5.3955
15:05:20 5.3195 5.3174 5.3174 5.3955
15:05:21 5.3976 5.3955 5.3955 5.4736
15:05:22 5.3195 5.3955 5.3955 5.4736
15:05:23 5.3976 5.3955 5.3955 5.3955
15:05:24 5.3976 5.3955 5.3174 5.3955
15:05:25 5.3976 5.3955 5.3174 5.3955
15:05:26 5.3195 5.3955 5.3955 5.3955
15:05:27 5.3976 5.3955 5.3955 5.3955
15:05:28 5.3195 5.3955 5.3174 5.3955

15:05:29 5.3195 5.3955 5.3174 5.3174
15:05:30 5.3195 5.3174 5.3174 5.3955
15:05:31 5.3195 5.3174 5.3174 5.3955
15:05:32 5.3195 5.3955 5.3174 5.4736
15:05:33 5.3976 5.3955 5.3955 5.3955
15:05:34 5.3195 5.3955 5.3955 5.3955
15:05:35 5.3195 5.3174 5.3174 5.3955
15:05:36 5.3195 5.3174 5.3174 5.3955
15:05:37 5.3976 5.3955 5.3174 5.3174
15:05:38 5.2414 5.3955 5.3174 5.3174
15:05:39 5.3976 5.3955 5.3955 5.3955
15:05:40 5.3195 5.3174 5.3174 5.3955
15:05:41 5.3976 5.3955 5.3955 5.3955
15:05:42 5.3195 5.3174 5.3174 5.3174
15:05:43 5.3976 5.3955 5.3174 5.3174
15:05:44 5.3976 5.3955 5.3174 5.3174
15:05:45 5.3195 5.3955 5.3955 5.3955
15:05:46 5.3976 5.3955 5.3174 5.3174
15:05:47 5.3195 5.3174 5.2393 5.3174
15:05:48 5.3195 5.3174 5.2393 5.3174
15:05:49 5.3195 5.3174 5.3174 5.3174
15:05:50 5.3195 5.3174 5.2393 5.3174
15:05:51 5.2414 5.3955 5.2393 5.3955
15:05:52 5.3195 5.3174 5.3174 5.3174
15:05:53 5.3195 5.3955 5.3174 5.3174
15:05:54 5.3976 5.3955 5.3174 5.3174
15:05:55 5.2414 5.3174 5.2393 5.3955
15:05:56 5.3195 5.3174 5.2393 5.3955
15:05:57 5.2414 5.3174 5.3174 5.3174
15:05:58 5.2414 5.3955 5.2393 5.3955
15:05:59 5.2414 5.3955 5.2393 5.3955
15:06:00 5.3976 5.3955 5.3174 5.3174
15:06:01 5.2414 5.3174 5.2393 5.3955
15:06:02 5.3195 5.3174 5.2393 5.3955
15:06:03 5.2414 5.3174 5.2393 5.3955
15:06:04 5.2414 5.3174 5.2393 5.3955
15:06:05 5.2414 5.3955 5.2393 5.3955
15:06:06 5.2414 5.3174 5.2393 5.3955
15:06:07 5.2414 5.3955 5.2393 5.3955
15:06:08 5.3195 5.3174 5.2393 5.3955
15:06:09 5.2414 5.3955 5.2393 5.3955
15:06:10 5.2414 5.3955 5.2393 5.3955
15:06:11 5.2414 5.3955 5.2393 5.3955
15:06:12 5.2414 5.3174 5.2393 5.3955
15:06:13 5.2414 5.3955 5.2393 5.3955
15:06:14 5.2414 5.3955 5.2393 5.3955
15:06:15 5.2414 5.3955 5.2393 5.3955
15:06:16 5.3195 5.3174 5.2393 5.3955

15:06:17 5.2414 5.3955 5.2393 5.3955
15:06:18 5.2414 5.3955 5.2393 5.3955
15:06:19 5.2414 5.3174 5.2393 5.3955
15:06:20 5.2414 5.3174 5.2393 5.3955
15:06:21 5.2414 5.3955 5.2393 5.3955
15:06:22 5.3195 5.3174 5.2393 5.3955
15:06:23 5.2414 5.3955 5.2393 5.3955
14:31:45 5.2414 5.2393 5.2393 5.2393
14:31:46 5.3195 5.2393 5.2393 5.2393
14:31:47 5.2414 5.2393 5.2393 5.2393
14:31:48 5.3195 5.2393 5.2393 5.2393
14:31:49 5.3195 5.2393 5.2393 5.2393
14:31:50 5.2414 5.2393 5.2393 5.2393
14:31:51 5.3195 5.2393 5.2393 5.2393
14:31:52 5.3195 5.2393 5.2393 5.2393
14:31:53 5.3195 5.2393 5.2393 5.2393
14:31:54 5.3195 5.2393 5.2393 5.2393
14:31:55 5.2414 5.3174 5.2393 5.2393
14:31:56 5.3195 5.2393 5.2393 5.2393
14:31:57 5.2414 5.2393 5.2393 5.2393
14:31:58 5.3195 5.2393 5.2393 5.2393
14:31:59 5.3195 5.2393 5.2393 5.2393
14:32:00 5.2414 5.2393 5.2393 5.2393
14:32:01 5.3195 5.2393 5.2393 5.2393
14:32:02 5.3195 5.2393 5.2393 5.2393
14:32:03 5.2414 5.3174 5.2393 5.2393
14:32:04 5.2414 5.2393 5.2393 5.2393
14:32:05 5.3195 5.2393 5.2393 5.2393
14:32:06 5.2414 5.2393 5.2393 5.2393
14:32:07 5.3195 5.2393 5.2393 5.2393
14:32:08 5.2414 5.2393 5.2393 5.2393
14:32:09 5.3195 5.2393 5.2393 5.2393
14:32:10 5.2414 5.2393 5.2393 5.2393
14:32:11 5.3195 5.2393 5.2393 5.2393
14:32:12 5.3195 5.2393 5.2393 5.2393
14:32:13 5.3195 5.2393 5.2393 5.2393
14:32:14 5.2414 5.2393 5.2393 5.2393
14:32:15 5.3195 5.2393 5.2393 5.2393
14:32:16 5.3195 5.2393 5.2393 5.2393
14:32:17 5.3195 5.2393 5.2393 5.2393
14:32:18 5.2414 5.2393 5.2393 5.2393
14:32:19 5.2414 5.3174 5.2393 5.2393
14:32:20 5.2414 5.2393 5.2393 5.2393
14:32:21 5.3195 5.2393 5.2393 5.2393
14:32:22 5.3195 5.2393 5.1611 5.2393
14:32:23 5.3195 5.2393 5.2393 5.2393
14:32:24 5.3195 5.2393 5.2393 5.2393
14:32:25 5.2414 5.2393 5.1611 5.2393

14:32:26 5.3195 5.2393 5.2393 5.2393
14:32:27 5.2414 5.2393 5.1611 5.2393
14:32:28 5.2414 5.2393 5.1611 5.2393
14:32:29 5.3195 5.2393 5.2393 5.2393
14:32:30 5.2414 5.2393 5.1611 5.2393
14:32:31 5.2414 5.2393 5.1611 5.2393
14:32:32 5.2414 5.2393 5.1611 5.2393
14:32:33 5.2414 5.3174 5.2393 5.2393
14:32:34 5.2414 5.2393 5.1611 5.2393
14:32:35 5.2414 5.2393 5.1611 5.2393
14:32:36 5.2414 5.2393 5.1611 5.2393
14:32:37 5.2414 5.2393 5.2393 5.2393
14:32:38 5.2414 5.2393 5.1611 5.2393
14:32:39 5.2414 5.2393 5.1611 5.2393
14:32:40 5.2414 5.2393 5.1611 5.2393
14:32:41 5.2414 5.2393 5.2393 5.2393
14:32:42 5.2414 5.2393 5.2393 5.2393
14:32:43 5.2414 5.2393 5.2393 5.2393
14:32:44 5.2414 5.2393 5.1611 5.2393
14:32:45 5.2414 5.2393 5.2393 5.2393
14:32:46 5.2414 5.2393 5.2393 5.2393
14:32:47 5.2414 5.2393 5.1611 5.2393
14:32:48 5.2414 5.2393 5.1611 5.2393
14:32:49 5.2414 5.2393 5.2393 5.2393
14:32:50 5.2414 5.2393 5.1611 5.2393
14:32:51 5.2414 5.2393 5.1611 5.2393
14:32:52 5.2414 5.2393 5.1611 5.2393
14:32:53 5.2414 5.2393 5.1611 5.2393
14:32:54 5.2414 5.2393 5.1611 5.2393
14:32:55 5.2414 5.2393 5.1611 5.2393
14:32:56 5.1633 5.2393 5.2393 5.2393
14:32:57 5.2414 5.2393 5.2393 5.2393
14:32:58 5.2414 5.2393 5.2393 5.2393
14:32:59 5.2414 5.2393 5.1611 5.2393

A.8: Josef Hart's Resume

HOME ADDRESS 12112 Glen Canyon Rd NE
Albuquerque, New Mexico 87111
(505) 293-3098

SCHOOL ADDRESS Post Office Box 2083
Socorro, New Mexico 87801
(505) 835-6853
e-mail: jhart@nmt.edu

OBJECTIVE: Seeking permanent employment in the area of electrical engineering

EDUCATION: B.S. Electrical Engineering, New Mexico Institute of Mining and Technology
Expected Date of Graduation: May 2002
GPA: 3.14/4.0 after 118 graded hours

WORK
EXPERIENCE: **Summer Intern, Utilities and Energy Management Team (UEMT), United States Department of Energy, Albuquerque, New Mexico, 2001.** Provided clerical support for the UEMT. Assisted studying of legal briefs. Reviewed and commented on proprietary solicitation proposals. Attended and participated in meetings with the Public Service Company of New Mexico, Public Regulation Committee of New Mexico, and Los Alamos County Power Pool. Quality checked and assured construction drawings for accuracy in estimated material needs. "L" security clearance being processed.

Summer Intern, Environmental Analysis, Intelligent Systems and RoboticsCenter, Sandia National Laboratories, Albuquerque, New Mexico, 1999, 2000. Compiled soil sample data; Analyzed maps for metal concentrations; Generated data for nuclear fall-out absorption; Marked sample locations using a GPS receiver; Entered chemical data for website database; Collected and analyzed data using MATLAB for a proprietary project.

Lab Assistant & Grader, New Mexico Tech Digital Lab, Socorro, New Mexico, January 1998-present. Maintained lab equipment; Entered data; Monitored lab activity; Prepared materials for lab classes; Graded papers and recorded grades for courses previously completed.

ACTIVITIES: New Mexico Tech Rugby Team

References available upon request

A.9: Ernest Jim's Resume

PO Box 128
Shiprock, NM 87420

Phone (505) 838-0145
Msg. (505)368-5908
E-mail ejim800@hotmail.com

Objective

Seeking employment in the electrical engineering field that will allow me to utilize previous employment experience and skills obtained from my educational background. Areas such as hardware, analog/digital design, controls and testing.

Education

1998 – 2002 New Mexico Tech Socorro, NM

Electrical Engineering major to receive BS in May 2002

- GPA 3.01
- Honor Roll Fall 2001

Employment

5/01- 8/01 Arizona Public Service(4-Corners Power Plant) Fruitland, NM

Electrical Engineering Intern

- Assisted in designing UPS(back-up power system) for server network.
- Responsible for cost estimates of several projects.
- Designed circuitry for several systems.

5/00- 8/00 BHP Minerals Fruitland, NM

5/99- 8/99

Electrical Engineering Intern

- Revised and updated Drag-Line electrical schematics.
- Assisted with electrical tribulations and solution development.

5/98- 8/98 Lawrence Livermore National Laboratory Livermore, CA

5/97- 8/97

Engineering Intern

- Assisted in design and fabrication of targets for two-stage gas gun.
- Designed database for data collected from experiments.
- Prepared weekly reports based on performance and development.

Qualifications

- Knowledge of computer aided programs – AutoCad, C, Microsoft products, Altera
- One Year + work experience from summer internships.
- College Course and Lab experience with digital/analog electronics, linear systems, digital signal processing, communication systems, micro-controllers, PC-interfacing, electro-magnetics, 3-phase power systems and transmissions lines.

References

Available upon request

A.10: Daniel Rodriguez's Resume

Campus Address
P.O. Box 3154
Socorro, NM 87801
(505) 835-6302
dmr78@hotmail.com

Home Address
1106 David Ct S.W.
Albuquerque, NM 87105
(505) 877-251
dmr78@hotmail.com

Objective

To obtain a position for full time employment in the Electrical Engineering industry that will allow me to apply the educational training in a commercial environment.

Education

Bachelor of Science in Electrical Engineering, New Mexico Institute of Mining and Technology, May 2002 (2.95 GPA – Major)

Experience

Energetic Materials Research and Testing Center, Socorro, NM
Office Assistant; Fall and Spring Quarters, 1997 to 2002
Assisted in data entry for the Purchasing Department
Filed and maintained purchasing accounts

The City of Albuquerque; Family and Community Services Dept., Albuquerque, NM; Special Events Assistant Summer Quarters, 1999 to 2000
Developed and maintained badge system of community centers.
Supervised special events at community centers

Bob's Burgers; Albuquerque, NM
Cashier/Cook/Assistant Manager 1993-1997
Worked the cash register
Managed night shift

Related Course Work
Microcontrollers
Electromagnetic Wave Transmission and Radiation
Intermediate Control Theory
Analog Electronics Adobe Photoshop

Computer Skills
LabView Matlab 6 Altera Max+plus II C C++ Adobe Photoshop MSWord MSExcel MSAccess Protel 99

Honors
Placed 7th in the Trinity College Fire Fighting Home Robot Contest

Wireless Vehicle Signaling System

A Thesis

Presented to the Faculty of

the Electrical Engineering Department

of

New Mexico Tech

By

David Byrd, Brandon Lattimore, Ivan Olguin

In partial fulfillment of the requirements

for the course

EE-482 Senior Design II

May 2002

Abstract

A major disadvantage to conventional trailer hookups is that they rely on hardwiring the tow vehicle wiring to trailer wiring. The Wireless Signaling System will eliminate the need for consumers to struggle with faulty wiring and bad grounds common on trailers today. The Wireless Signaling System Team is responsible for designing a universal system to bypass the normal trailer wire assemblies on current trailers and replace them with an innovative RF wireless control and LED array system. The system is portable and can be installed with simple hand tools, allowing the average consumer to simplify the amount of wiring necessary to setup a trailer signaling system.

After completing an extensive literature search on the necessary components, the prototype circuit was developed and preliminary testing was done on all the subsystems. These designs were then transferred onto a professionally made printed circuit board, which was also designed by the group. The sensors, which are used to determine if the brake or turn signals are being activated, were then integrated into the complete system. The overall results of the project were a success and the components have been tested inside vehicles during real world road testing.

Acknowledgements

We would like to thank our sponsors Jason Vandehey and Jeff Duyck for the opportunity that they have given us. This opportunity is one that has not only benefited us this semester but will also be a great help in our success as electrical engineers. We would also like to thank Jason for his continued support and knowledge that he shared with us.

We would like to give a special thanks to the faculty of the Electrical Engineering Department at New Mexico Tech for the continued support and knowledge. Thanks to Dr. William Rison for his support and contribution of the programmable chips and equipment. Special thanks to our faculty advisor, Dr. Ronald Thomas, for his guidance and support along with all the extra time he set aside for us out of his busy schedule. We would also like to thank Dr. Richard Scott and Dr. Robert Bond for the support and knowledge about RF transmission.

We would also like to offer our appreciation for all the support and guidance that Dr. Scott W. Teare has given us both in and out of class. Dr. Teare has given us knowledge on how to succeed in the real world and how to develop our engineering skills.

Table of Contents

Chapter 1

This chapter gives a background and overview to the Wireless Vehicle Signaling System

1.1 Overview of Project

This document presents an outline to our approach, reviews the requirements for the Wireless Vehicle Signaling System, and states the design processes taken to develop a successful end product. It shows all the steps taken to develop an innovative remote signaling system for automobiles.

1.2 Introduction

A major disadvantage to conventional trailer hookups is that they rely on hardwiring the tow vehicle wiring to trailer wiring. The Wireless Signaling System will eliminate the need for consumers to struggle with faulty wiring and bad grounds common on trailers today. The Wireless Signaling System Team is responsible for designing a universal system to bypass the normal trailer wire assemblies on current trailers and replace them with an innovative RF wireless control and LED array system. The system is portable and can be installed with simple hand tools, allowing the average consumer to simplify the amount of wiring necessary to setup a trailer signaling system.

1.3 Project Goals

The objective for this project is to build a wireless vehicle signaling system to eliminate having to run a cable from the vehicle to plug into the trailer. We were given several requirements from the project sponsor to include in the project.

Two separate transmission systems needed to be researched, tested, and built capable of meeting the Wireless Vehicle Signaling System's requirements. A huge requirement was that the whole product must be universal. It had to have the capability of working in most makes of vehicles. The two systems needed to be compared and evaluated in order to select the best one to take into the final design. A main requirement was that it be user friendly in all aspects. It needed to be set up by the average consumer in minutes with no needed training. To ensure this, there could be no direct wiring to the vehicle electrical system. We had to design and develop several sensors capable of detecting the intentions of the driver since we were unable to tie into the vehicle's wiring. The sensors needed to be reliable and sturdy. The whole system would need to be laid out and then designed in PCB design software. The board would then be sent out to a manufacturing house to be professionally etched. The system also required a set of prototyped LED arrays and light driver circuitry to power the arrays. The whole system would need to cost less than $150 while in full in production and a $200 goal for early production.

1.4 Deliverables for the Wireless Vehicle Signaling System

The project started on September 10th, 2001 and was completed May 1st 2002. The group created a Gantt chart for each of the first and second semesters of the project (APPENDIX M). The tasks were divided up and distributed to the three group members depending on interest and the skills that each possessed.

One of the most important tasks that was handled was the documentation of project design. Bi-weekly reports were written to give the faculty and project sponsor a heads up on the progress and a summary of the successes and issues that were arising in the project. Also, more formal work plans and project summary reports were required giving the customer a good look at what work was being done, and that the project was going as scheduled. By reporting on any issues the group had come across, the team was able to see all the issues that arose early and was able to tackle them in a timely manner.

The following figure lays out the items that had to be presented to our customer during the project. On May 3rd the project was fully completed and the complete prototype system was ready to be handed over to the customer. Refer to Appendix L for a list of the web sites containing information we referenced for the project.

Deliverables
- Investigate several types of input sensors and RF transmitter/receivers.
- Choose several different sensor options and 2 transmitter options.
- Design sensors that seem feasible past the elementary stage and test them to see if they work well for this application.
- Build prototype LED array for system testing.
- Report stating how the sensor selections and working- end of October.
- Report of how the transmitter and decoder work is going –end of November.

- Bi-weekly status reports.
- Prototype RF circuitry and encoding/decoding system.
- Design PCB's for system and combine into an integrated system.
- Documentation of project including a users guide.
- Testing of our final product.
- Final document stating the functionality of system.

Figure 1: Chart listing of the main deliverable requirements

1.5 Design Specifications

The following section states the necessary specifications of the transmitter, receiver, input sensor, and LED array subsystems.

The transmitter needs to interface with the vehicle by connecting through the cigarette lighter plug. This is the only direct connection to the vehicle for the entire system. There needs to be built in noise-immunity and surge protection on the control box. The system must contain override switches on control box to manually signal driver intentions, along with status LED's to display when the sensors are tripped or the manual switches are set. The transmission setup must be encoded to safeguard against cross talk. The worst-case transmission distances is 100ft and the entire system should be designed with industry standard parts.

The receiver side of the setup will be battery powered and be designed to consume low power. The system will run on a user supplied power source of between 10-14 volts. There is a decoder on the receiver side to ensure that the received signals are coming from the correct tow vehicle. The system will include light driver circuitry that sets the LED arrays to run at a low level for continuous

level lighting, along with a full intensity level when the brakes are applied. Logic circuitry is needed to control the flashing/braking of LED arrays. The brake input state must override the hazard state.

The light emitting diode arrays must consist of red high intensity LED's. These LED's must meet or exceed the brightness of current production trailers. The bulbs must have a long life, and consume low power. The prototype board built will be able to give the customer a general idea of how bright the LED arrays are, along with a sample of how the braking and turning functions are output from the rear signaling box.

Chapter 2: Technical Work

This Chapter describes in detail all the work that was needed to put together a successful end product. It is broken up into sections detailing the transmitter and receiver subsystems along with Protel layout and design work involved.

2.1 Work Overview

The project's first technical research was to decide on a transmitter and receiver pair. We decided to choose two different pairs (alpha and beta). Each pair was tested and evaluated for simplicity, transmission distance, and cost.

The alpha transmitter and receiver pair was not pre-wired to any other components. This allowed us to have full flexibility on both the transmitter and receiver. This system allowed the group to add all external logic and a light driver circuit to meet exact specifications. This system will work with a variety of encoders where the beta system will only work with a specific encoder because of the prefabricated receiver.

The beta transmitter and receiver pair came in a package that only needed an encoder to make a completely working system. The receiver was enclosed in a box that included the decoder and relays to power an output maximum of 5 amps. The problem that we ran into with this system was that it would not allow us to have the flexibility to add our own specific requirements. The beta pair would also use up excess power to run the relay switches. Another important decision was to find an encoder and decoder, which would work with our transmitter and receiver and also allow us to meet the specific

requirements of our customer. After making our decision we then separated the project into two parts: the transmitter and receiver boards.

2.2 Transmitter Circuit

The transmitter board consists of several chips, such as: transmitter, encoder, voltage regulator, and several other components that were necessary to get these chips to work together. Due to the simplicity of the board there was only the need for one input logic gate to transmit our signals. We first decided to use an OR gate as a way to control when the encoder would transmit, however after further evaluation we decided to make an OR gate out of diodes. This not only cut down on cost but it also used less space on the PC board. Refer to Appendix C for the entire transmitter schematic. The main requirements for the transmitter board were that it had to be powered by a cigarette lighter. We also added a hazard switch and override switches for both the turn signals and brake signal. Another main task was to add a tri-state dipswitch to our encoder, which would allow us to exceed the requirement of at least 256 addressing. The 256 addressing would also prevent any possibility of cross talk or interference from another signal. The four channels on the encoder were used for the brake signal, left and right turn signal, and the hazard signal. There were several stages that we took in building the transmitter board. The first stage was to connect up the transmitter and encoder and verify that a signal could be transmitted properly. The entire transmitter system is shown in Figure 2.

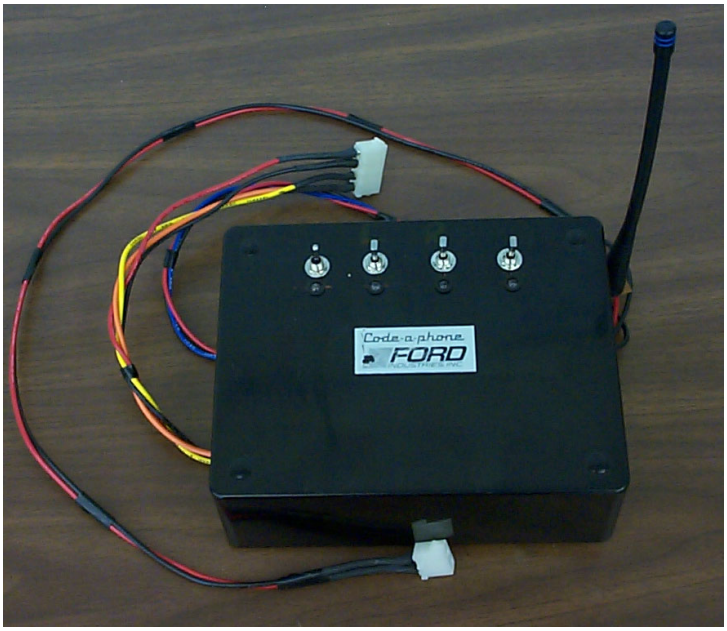


Figure 2: Entire transmitter circuit

2.3 Receiver Circuit

The receiver board was a lot more complex because it also included the output logic necessary to control our LED arrays and the relays. The main components in the receiver board are the receiver, decoder, programmable logic chip (PAL), 556 timers, Voltage regulator, BJT power transistors, and relay switches. Refer to Appendix D for a schematic of the receiver system. On the receiver board we were required to use a battery for our main power source. The battery had to have a long life and it also had to be rechargeable. The battery selection was done after we completed the board. The receiver board used so little power that it allowed us to use two small Ni-Cad batteries in series to get the 14 volts necessary to power our circuit.

The receiver board was done in several steps. First we built and tested the receiver and decoder. This was done simultaneously with the transmitter

side, so that testing on interference and distance could be checked. The entire receiver circuit is shown in Figure 3.

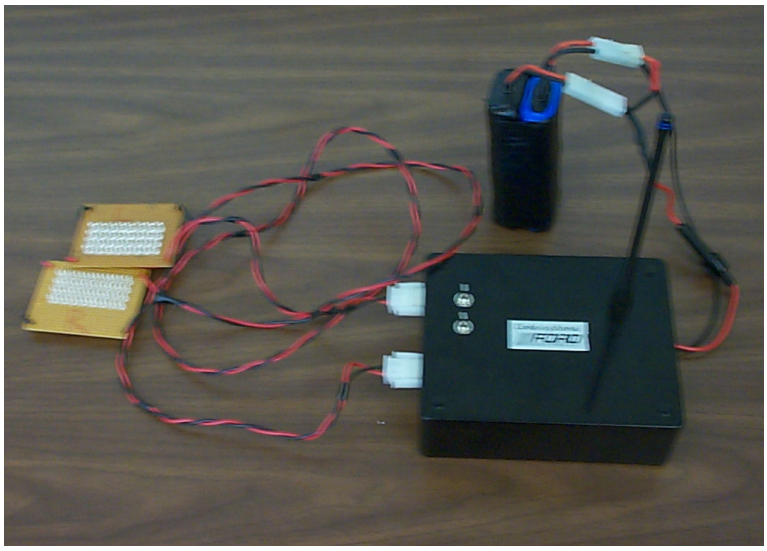


Figure 3: Entire receiver circuit

2.4 Output Logic

After we were confident that the receiver was working correctly we started the design of the output logic. The output logic was necessary so that we could control not only the intensity of the LED's but also the hierarchy of each of the sensor inputs. Our first approach to the output logic was successful. However, the first approach used individual IC's to accomplish this task. We felt the individual IC's took up too much room on our final PC board design. After completely testing the output logic on the receiver end we decided it would be more beneficial to look into using a programmable chip for that logic. Since an Altera chip is easily accessible to us we decided to implement our design on the Altera chip. Although the design worked flawlessly we felt that the Altera chip was a little overkill for our specific design. This forced us to do further research

on other possible programmable chips that would not only better suit our design but also be more cost effective. We got a hold of an old style programmer from BP Micro and several programmable chips called PAL's from a professor and looked into getting it to work. That involved downloading some software for the programmer from BP Micro and also compiling software called PALASM. All of this was new to the group so we spent time learning the new software and how to implement our logic in the form needed to compile. After weeks of struggling, we were finally able to successfully program a PAL (AMD348217) chip. The use of this chip allows for future expansion, as one just needs to reprogram the chip the way they want it. We added a dip-switch to the unused inputs so that they could be used for more combinations in the future. Adding the AMD348217 simplified the entire system greatly as it reduced the total chip count by four chips. To implement the flashing of the LED arrays for left and right turn purposes and for continuous lights we used a LM 556, which has dual 555 timers.

2.5 Relays

As an added feature we gave the user an option of using the power transistors or the relays. This gives the user an opportunity to use the LED arrays or normal incandescent light bulbs. The user can accomplish this by positioning the jumper that is located at the inside of the box to the correct setting. The LED arrays were designed in house and made specifically for our setup. Each of the LED arrays is powered by a power transistor and requires only one input line and a ground wire. The incandescent light bulbs are from a

normal utility trailer and a relay switch powers each of the bulbs. This setup requires the use of additional incandescent bulb on each side to run as continuous lights.

2.6 Protel

Once both the transmitter and receiver boards were tested we decided to use the PCB making software Protel to lay out the circuit design. The reason why we decided to use Protel was because the group was vaguely familiar with it and it was easily accessible to us in the lab. To lay the board out quicker we simultaneously laid the transmitter and receiver out on different computers. The layout took a little longer than expected, because we only knew the basics of the software. We laid out the entire transmitter and receiver systems and then began to design the PCB part of it. In order to save money on the PCB manufacturing we decided to put both the transmitter and receiver on the same board, and just leave space so that we could cut them in half.

During the PCB process we assumed that once we had the PCB layout of each of systems complete we could just copy one into the other. That was not the case. The whole system has to be complete in the schematic to update the PCB so we wasted quite a bit of time not knowing this. We put the whole system in the schematic and updated the PCB to find over 135 errors. Refer to Appendix E for the PCB layout. This obviously took some time to debug. Most of the errors involved resistors and capacitors being named the same and the wrong footprints for the parts. Those were relatively easy problems to solve. The

hardest problems were ones involving footprints that we had to make for the part that were not available in Protel. For those parts, we had to make our own footprint that matched the real part exactly. After all this was done and all the errors were fixed we printed out the lay out to make sure the holes for all the parts matched. Everything looked successful so we continued on with the PCB process and sent the files to advanced circuits for manufacturing. The printed circuit board is shown in Figure 4.

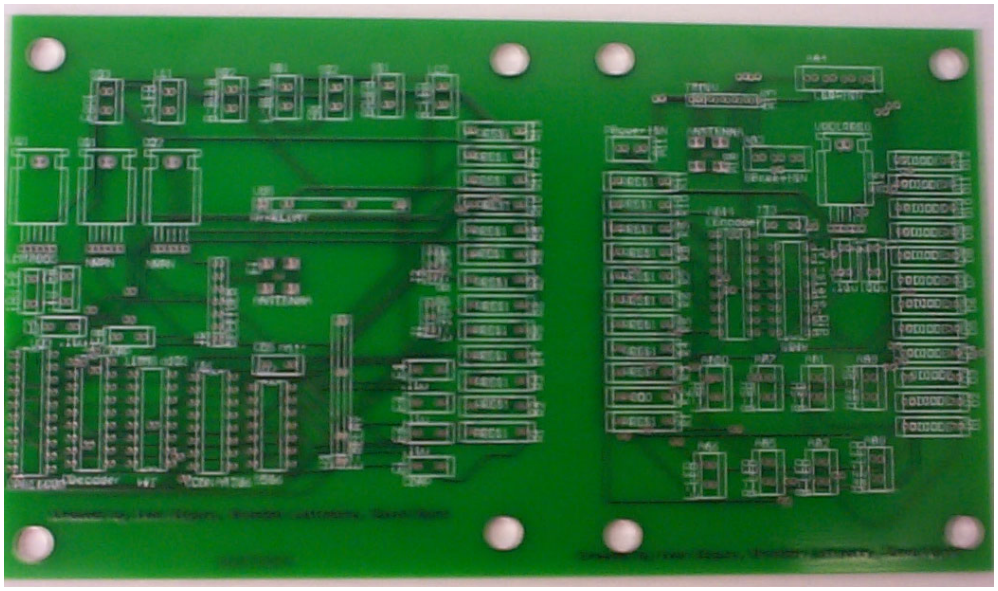


Figure 4: Printed circuit board

Upon receiving the board, we found errors immediately. Some of the hole sizes for the parts were too small. The reason for us not seeing the problem from the printed layout was that it was just too close to see with the eye. The resistor holes were tight but they fit. The 7805 and transistor holes were too small, but after we re-drilled the holes they fit perfectly. The problem was with

the connectors. We had ordered .156 mil spacing for all our connectors so we had to design the footprints for all the connectors. We got the spacing correct but failed to make the holes larger not noticing that there was a difference between our connectors and the standardized Protel connector pads. This affected about 15 connectors. To fix this we had to grind down the pins on all the connectors so they would fit in the board. Resizing the current connectors was the only option, as for the vendors we found only used pins of at least 45 mils wide for connectors with .156 spacing. The PC board was a success and worked perfectly even with these few setbacks.

Chapter 3: Final Sensors Status

For the wireless transmitter project, the driver's brakes and turning intentions needed to be sensed without tapping into any of the wires. To do this, sensors were needed to sense the driver's intentions of two different functions: turn signal and brake. We researched and tested several possibilities for each task using some of the sensors to try and accomplish both functions by applying it in a different way to accomplish the needed task. This chapter is broken up into two sections: a brake sensor system and a turn signal system.

3.1 Brake Sensor System Development

The brake pedal sensor would need to sense when the brake pedal is being pushed but without any interference to the normal driving habits and safety of driver in any way. The sensor would need to be kept away from the driver's foot and not hinder the foot reaching the brake pedal in any way.

3.1.1 Sharp GP2D120:

Description: This is a general purpose IR distance measuring sensor. The output voltage on the sensor increases as an object gets closer.

Application: The first application involved placing the sensor directly above the brake pedal so that it would see the driver's foot on the brake pedal. This wasn't

very feasible for a couple of reasons. It needed to be placed within 3 inches of the path of the foot for an accurate reading, which meant it could easily be kicked off. Also, since it was looking directly for the foot it might see the foot just resting by the pedal or under it and assume that braking is intended. A more feasible way of using this sensor was to set it up to look for the movement of the brake pedal shaft. We had to add some extra circuitry to make this sensor work for this application. The output voltage of the sensor ranges from about 1.3 volts to 1.5 volts when an object is within 3.5 to 6 inches. This sensor needed external circuitry since the encoder needs 3.7 volts to switch levels. To fix this we added an adjustable schmitt trigger to take inputs of 1.3 volts to 1.5 volts and output 5 volts. With a few calculations we were able to figure the exact resistor values for the schmitt trigger to implement this. The sensor could now sense when an object was within range and then output 5 volts. Refer to Appendix I for a printed circuit board layout and schmitt trigger circuit details.

Final Status: This is proving to be a reliable sensor when a good mounting place is found. This sensor is depicted in figure 5 below. It works in a good variety of vehicles but it will be up to the customer to mount the sensor 3.5 to 6 inches from the firewall and then adjust the switching level by adjusting the 10k multi-turn pot until the schmitt trigger is giving a logic low. We found that the R2 value on the schmitt trigger needed to be increased in order to work at distances above 6 inches. The current generation of full-size Ford pickups has the firewall set back further than the other tested vehicles

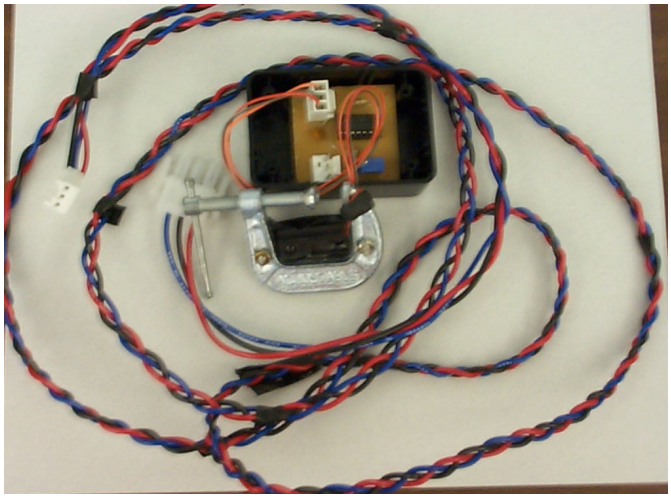


Figure 5: Distance Sensor

3.1.2 Push Button Switch:

Description: This is a small, discrete, general-purpose push button switch. This sensor is a button that outputs 5 volts when pushed.

Application: The hardest thing with the sensor is how to mount it without it interfering with the driver. We created housing for the push button out of thick plexi-glass to protect it from damage from the foot. This block also enabled us to put sandpaper on the face of the block so that the foot wouldn't slip off the block. The plexi-glass block is placed over the entire brake pedal with the button being in the center of the brake.

Final Status: This sensor is in working condition and has been tested on a vehicle. It is a little awkward at first, but after a while it feels a lot like the original brake pedal. It is a very reliable sensor. This sensor is depicted in Figure 6.

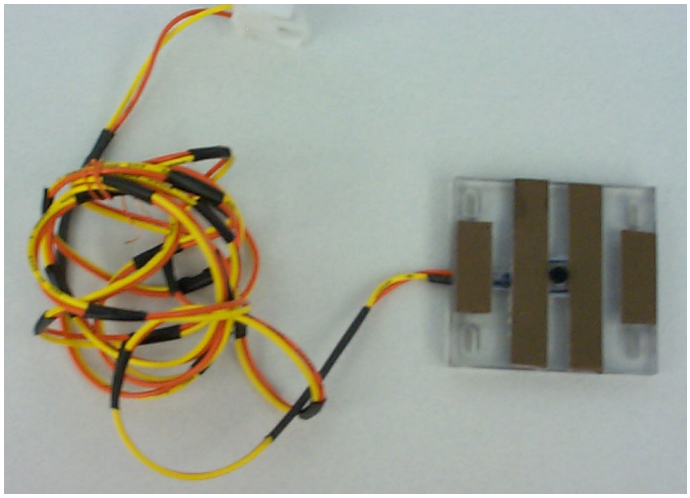


Figure 6: Push button switch

3.1.3 Lever Switch*:*

Description: This is a basic lever switch with a decent size lever on it. Five volts is outputted when the lever is pressed.

Application: The idea behind this one was to mount the base on the shaft of the brake pedal and it would be pushed up against the back firewall as the brake pedal was pushed. This is a good application because it is totally out of the way of the driver's foot.

Final Status: We had problems locating a good place on the brake pedal shaft where the switch would make contact when the pedal was pushed due to the severe differences in vehicles brake mounting among different models. Because of this issue this switch didn't make it past the initial selection.

3.1.4 Hall-Effect Sensor Hallogic OH090U:

Description: This is a Hallogic hall-effect sensor that detects the presence or absence of a magnetic field.

Application: The best way to incorporate this into the design is to mount a magnet onto the brake lever arm, while mounting a hall-effect sensor board in a fixed location near the magnet. When the magnet is within 3/8 inches to 5/8 inches depending on the intensity of the magnetic field, the transistor switches, and logic low is seen on the output of the sensor. As the brake pedal is depressed 1/8 of an inch or more, the magnetic field is not strong enough to switch the internal transistor of the hall-effect sensor and logic high is seen on the hall effect output. To mount this sensor, zip ties, or Velcro can be used and sensor adjustment consists of changing the distance between the magnet mounted on the brake pedal and the sensor body. The Hallogic sensor can sink up to 30mA so this sensor is more than capable of driving the encoder chip. Refer to Appendix G for a printed circuit board layout and the circuitry involved.

Final Testing: The sensor is designed to work in harsh environments and there is an internal voltage regulator along with built in histeresis internal to the chip. In-vehicle testing proved the sensor worked flawlessly with no false signals being seen. For testing in the current generation of full size Fords, a bracket was made to mount into factor holes underneath the dash to extend the sensor over the brake lever assembly. This sensor is shown in Figure 7.

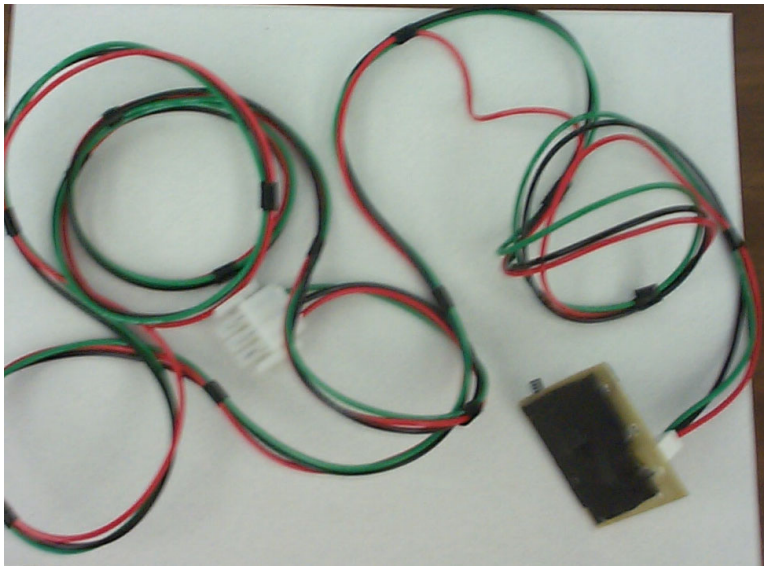


Figure 7: Hall-effect sensor

3.1.5 Current Sensor

Description: This is a relatively simple sensor that senses the presence of current in a wire.

Application: Our first approach to using this sensor was to search for one we could wrap around the brake wire that would sense the presence of current while the brake pedal is compressed. This would be easy to implement if we could find the correct sensor and then locate the correct wire.

Final Status: We finally found a clamp on type sensor but for no less than a few hundred dollars, which was much more than we could allot for one sensor. Many vendors did carry a through-hole device that would work but with the stipulation of not being able to tamper with vehicle wiring we had to dismiss the idea. That would be hard to do without cutting the wire. Another problem we were running

into was that we would have to instruct future customers of this product on how to find the exact wire. That would be hard since the location of the wire is different on all vehicles. For these reasons this sensor didn't work out.

3.1.6 Touch Pad

Description: This is a simple momentary contact touch sensor that outputs five volts when touched and stays high until untouched.

Application: The idea behind this sensor was to find a very thin touch sensor that could be placed over the brake pedal and sense the touch of the foot. We wanted something very small and discrete that would not change the feel of the brake pedal drastically. This sensor in theory would be the best way to go.

Final Status: The problem we had with this was that we could not find a vendor that carries such a sensor to meet our needs. We later searched for the touch sensors similar to those used for the touch pad of a microwave. We were unsuccessful in finding this sensor.

3.1.7 Angle Sensor

Description: Analog devices ADXL202JQC angle sensor capable of seeing tilt relative to the X and Y planes.

Application: The sensor is mounted to the brake shaft and when the user applies pressure to the brakes, the shaft tilts and changes the sensor output a few tenths of a volt when a 5-10 degree tilt is seen on the brake lever. This change is fed into a Schmitt Trigger and the signal would be conditioned and sent to the transmitter box to be relayed out to the receiver box.

Final Status: The sensor initially looked good during initial prototype testing, but when a vehicle accelerates or there is weight in the back of the tow vehicle, the tilt seen by the sensor is great enough to emit false signals. This sensor had to be discarded and the other brake sensors were further developed.

3.2 Turn Signal Sensor Development

The turn signal sensor would need to sense whether a turn is being initiated and relay back in which direction the turn is intended. This sensor can in no way hinder the drivers usual driving habits or give the driver any added responsibility.

3.2.1 Toggle Switch:

Description: This is a basic three-way toggle switch with a very long shaft.

Application: The base of the switch is attached to the steering column while the shaft of the toggle switch is attached to the turn signal shaft. As the turn signal shaft is moved up and down the toggle switch moves with it. The correct signal

is then relayed to the control box indicating which direction the driver is turning. This in a sense would work exactly like the normal mechanical turn signal does.

Final Status: The shaft on the switch needs to be long enough so that it can move easily with the turn signal. This is what created the problem because we couldn't find a toggle switch that had a long enough shaft or with a sensitive enough switching mechanism that moves smoothly and keeps the base of the switch immobile. We tried using a toggle switch that we had that didn't have a very long shaft but it failed to move smoothly with the turn signal shaft.

3.2.2 Push Button Switch:

Description: This is a small, discrete, general-purpose push button switch. This sensor is a button that outputs 5 volts when pushed.

Application: This application would involve two general-purpose push button switches with one each being placed on the top and bottom of the shaft. When the driver operated the turn signal he would have to make sure his hand was placed on the appropriate button. The hope was that this sequence would become routine to the driver and not a hassle, although anything extra the driver has to do could become a distraction so this was thought to pose a problem from the beginning.

Final Status: It was later discovered through testing that in order for this sensor to work the driver would have to hold the button throughout the entire turning or process or hit the button again when the turn is complete. This sensor wouldn't work since we could not expect the driver to have to adjust his driving style in order for the system to fit his needs.

3.2.3 Phototransistor:

Description: A simple PN168 along with a driver circuit.

Application: We would mount these on the dash next to the turn signal indicator lights to relay back when one of the lights is blinking. We will need one of these for each the left and right turn signal light. We will also be looking for a prepackaged light indicator that might be easier to mount.

Final Status: The overall design of the phototransistor worked great. The only problem that may occur in the future is excess light given off by the dashboard. This may cause the phototransistor to be activated falsely. I have added a 10K-ohm adjustable resistor to the circuit, which will allow the user to make adjustments to the sensitivity of the phototransistor. Refer to Appendix H for a detailed schematic of the circuitry involved.

3.2.4 Lever Switch:

Description: This is a very small simple lever switch that outputs 5 volts when activated.

Application: Two lever switches are needed for the turn signal application, one each above and below the turn signal shaft. The lever switches are attached with Velcro just above and below the shaft with each lever just barely resting upon the turn signal shaft. When the driver pushes the turn signal shaft either up or down, the correct lever is activated and a high is sent to the control box indicating the correct turn.

Final Status: This sensor has proven to be very reliable and works well for this application in a large variety of vehicles due to the simplistic design and ease of mounting since turn signal levers and steering columns are reasonably uniform. This lever switch is shown in Figure 8.

Figure 8: Lever switches

3.2.5 Sharp GP2D120:

Description: This is a general purpose IR distance measuring sensor. The output voltage on the sensor increases as an object gets closer.

Application: To apply this sensor to work for the turn signal there would need to be a sensor attached to both the bottom and top of the turn signal shaft. Each sensor would be angled toward the steering wheel column so that there will be no voltage reading when the turn signal shaft is at the idle position. When a turn is indicated, the sensor would then be at such an angle that it reflects off of the steering column indicating whether it is a left or right turn.

Final Status: This was a little too complicated of a design to implement in such a space-constrained place. Using two distance sensors made the cost of this sensor fairly considerable. We also ran into mounting issues with this design because the sensors themselves were too bulky to place anywhere so they would not interfere with the driver and normal operation of the turn signal lever. This sensor worked for the brake but wasn't a good approach for the turn signal application. Another issue that this sensor brings up is that the sensory circuitry that needs to be added to bring the outputs to appropriate levels would have to be extremely precise. This is because many turn signal shafts have an extremely small range of motion thus the GP2D120 would have issues picking up those small angle changes.

Chapter 4: Individual Contributions

This chapter is broken up into sections describing each team member's individual contributions to the overall success of the project.

4.1 Tasks Accomplished by Brandon Lattimore

The first task I was given was to do the sensor research and selection. I was to research several possibilities for sensors to be used for both the brake and turn signals. They would need to be affordable, reliable, and compact. They also could not interfere with the driver's normal habits. The customer also requested that we try out several different ways to do each task with the possibility of giving the customer the option to pick and choose the sensor that would work best for his vehicle. For the brake sensor I looked for sensors that would not get in the way of the driver's foot. Along the same lines, I chose sensors for the turn signal that would not interfere with the driver's usual turn signal routine. I searched through Digi-key, online web sites, and through previous work to determine which sensors to order. I ordered a total of about nine sensors that were wired up and tested for feasibility. David Byrd found a few more sensors to add onto a later order we had. I helped wire up all of the sensors on a breadboard to see which ones would be good to try out on a vehicle. From there, we chose which sensors to take out to the vehicle and test. After testing all of the sensors, six were chosen as ones that would work for the final product. Those six needed to be wired up in a way that was convenient to mount and pleasing to eye so we divided the sensors equally. I was given the two lever switches to mount and wire up.

I also worked on the interface logic to take the signals from the driver's inputs and make them do what was necessary, i.e. brake, blink. I developed a series of AND and OR gates to implement the necessary patterns on the tail light system. After that was finished I worked with Dave to simplify either his design or mine. After many brutal hours we were able to simplify my design down to a reasonable number of chips. It involved four logic chips so we decided to look at a programmable logic chip to put all of that logic onto one chip. I got the programmer from Dr. Rison and downloaded all the necessary software to run it and compile the code. I wrote the code (Appendix A) for the logic using a truth table (Appendix B) and successfully programmed it onto a PAL. I tested the PAL into the entire circuit and it worked exactly like it was suppose to. This saved both money and space on the board.

I worked with Ivan to get the alpha transmitter/receiver pair from Glolab to work. We ran into a few problems at the beginning trying to get a complicated encoder working, but ended up using a simpler Holtek encoder/decoder pair to get the Glolab transmitter/receiver to receive signals.

The major task I completed for this project was the board layout in Protel. We decided to implement the transmitter board and receiver board onto one to save on cost of manufacturing the board. I developed the receiver board in Protel while Ivan did the transmitter board. Once we laid out the circuits in Protel, I combined the two into one board and placed all the parts in the PCB part of Protel. I had to make certain that no traces from the transmitter side were creeping over to the receiver side. After the board was laid out the way we

wanted it, Dave sent it off to the manufacturer. I was in charge of soldering all the parts onto the receiver board and getting it to work once it came in. I then rigged up a plastic box so that it would hold the circuit, switches, wires, and antenna. Once all of this was done in correlation with the other members' work, we were able to test the entire project on a vehicle.

Along with the work on the project, I also played a big part in the writing of the papers. I was given every third progress report to write and helped write every other paper that was due.

4.2 Tasks Accomplished by David Byrd

One of my main initial tasks consisted of investigating several receiver/ transmitter pairs along with encoding/decoding options. In order to make a decision upon which transmission pair to use, I compared spec sheets from several vendors to narrow the selection down to just a few choices. In this phase, I also spoke with the Linx Technical Department to verify compatibility of their prefab receiver with the Holtek encoding I was planning on using for the design.

Another task I worked on was the interface circuitry for the Linx Module. In this stage I looked at different ways to interface the Linx transmitter and receiver with the input sensors and taillights. Two separate designs with the linx module were developed. They incorporated either two arrays per side or additional logic after the pre-fab receiver and thus were thrown out after the preliminary design phase. Along with the interface research, I looked into a manner in which to send a PWM signal through the transmitter for the turn signal flashing. After the

group decided upon using the Glolab transmission system and going with one LED array on each side, the Linx prefab receiver was set aside and investigation began more on the way our output logic would be attacked.

Brandon Lattimore and I spent several days working on finding the most efficient way to simplify the logic the group initially created. Appendix F shows the simplest design that we came up with. The four-chip design was prototyped on a breadboard and was viable, but an inexpensive programmable logic device would allow for more options and expansion in the future, so that was decided upon for our final design.

Another task I worked on was the creation of the flashing times for the LED arrays. The lights need to flash similar to the rate and intensity that normal trailer lights flash. So the turn signals were set to flash about every one-third a second. The flash levels were set by adjusting the resistors on the 555 timers chip and with the lights being off for more time than on.

Brake sensor investigation and development were two key roles I took on the second semester. Some of the ideas that I implemented were a tilt sensor, a Hall effect sensor, and a adjustable IR distance sensor. I created printed circuit boards for the Hall effect and distance sensor circuits, and used a previously developed PC board for the tilt sensor. Other sensors I looked into that were feasible would be a different IR emitter and detector pair, a current sensor, or a pressure sensitive device that would mount directly onto the pedal itself.

After the prototype brake sensors were developed, I proceeded to test the initial designs, refine them, work on mounting issues, and verify the correct

operation of the sensors. I found that the best way to verify the designs was to actually mount the sensors in several different makes and years of trucks to get a feel for whether the sensor would work in a real world application.

A final task that I was in charge of was taking care of all the purchasing and budget issues. This task allowed me to work with many different vendors and allow me to interact with many different people in industry. The documentation was divided among the group members so I also worked on several progress reports along with the development of the Gantt charts and work plans.

4.3 Tasks Accomplished by Ivan

I was assigned to work on four major parts of our project. The first part was to research the FCC regulations that may effect or limit the use of our transmitter/ receiver pairs. This was a very important part of our project, because we needed to make sure that we purchased a transmitter/receiver pair that would be within FCC regulations. Even though we did not find any transmitter/receiver pairs that were already FCC accepted, we did find several that laid within the frequency range that is acceptable to the FCC. According to the FCC the transmitter /receiver pairs that we picked could only transmit a signal within a range of 300 feet. Anything over 300 feet would be considered illegal and must be FCC approved in order to be used. I was also chosen to work with Brandon on the alpha transmitter/receiver pair. The alpha transmitter and receiver are from Glolab and are part of a kit. The kit was to be interfaced with an encoder/decoder pair from either Holtek or Glolab. Brandon and I testing several

encoder/ decoder pair so that we would be able to determine which pair will work best. Performing these tests allowed us to pick an encoder/decoder pair that would not encounter any future interference. Testing was done to determine which encoder/decoder pair would work best with our transmitter/receiver from Glolab. We decided to use an 18 pin encoder/decoder pair from Holtek (HT694). This encoder/decoder pair turned out to work best for our application. The Holtek was not only simple to interface but it also gave us an acceptable amount of encoding possibilities. The range of our transmitter/receiver pair is about 250 feet.

My second assignment was to design the taillight (LED arrays) as depicted in figure 9 that would be mounted to the trailer. I have done research on both the pre-manufactured LED arrays and also on a design of my own. Due to the high expense of the pre-manufactured LED arrays I decided to design my own. I have done some simple calculations that will give us maximum light intensity with minimal power consumption. At the beginning of the spring semester we were asked to add on the possibility for the use of regular incandescent light bulbs. This had no major effect on the LED array output. Instead we used a jumper to access either the LED's or the light bulbs.

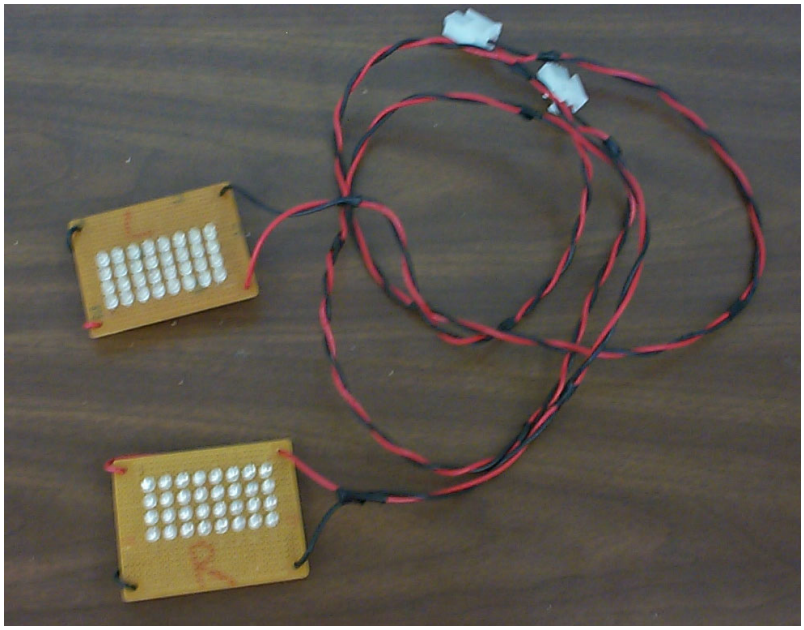


Figure 9: LED taillight arrays

The third part I was assigned to design was a PC board that would consist of all the necessary hardware to have a complete prototype by the end of the spring semester. Brandon and I were in charge of different parts the PC board design. We decided to use Protel to layout our design since both Brandon and I were familiar with it. I was assigned to work on the transmitter board layout. Once we each had our section laid out in Protel we then incorporated our designs into one package, which was then sent out for etching. After receiving our board from Advanced Circuits I mounted and soldered all necessary connectors, resistors, diodes, and chips to the board. Next, Brandon and I preformed testing on our individual boards and then we tested them together as a pair.

The fourth part that I was assigned to was sensor development. Each of us was required to choose several sensors for the turn signal or the brake pedal.

I decided to use a push button for the brake pedal. It was a simple design but it required that a housing be built for the sensor to prevent it from being damaged. I used a square piece of Plexiglas to house the sensor. This allowed the sensor to be easily activated without the pressure of the foot damaging it or the wiring. I used Velcro to attach the sensor to the brake pedal. For the turn signal I decided to use a phototransistor to detect the presents of light from the dashboard. The phototransistor does not give off enough voltage to activate our transmitter circuit, so I added an amplifier from Maxim to increase the voltage. The mounting of this sensor only required a piece of double stick tape.

Chapter 5: Conclusion

The following chapter is broken down into four sections that include the project highlights, budget summary, final sensor functionality, and overall project functionality.

5.1 Project Highlights

The wireless signaling system incorporates many features that make this product a success. Robust keyed connectors are used in the design to insure that the user is able to easily hook up various sensors and does not have to worry about a difficult installation. The system incorporates an encoding and decoding system that allows several of the same devices to operate in a close proximity without crosstalk. Another key feature of the system is that the output logic is developed using a programmable logic device, so additional features can be added in the future if necessary. The sensors developed can be used on many different vehicle applications. A final highlight is that the transmission system and sensors have worked well throughout the initial testing process.

5.2 Budget Summary

The Wireless Signaling System team was given $740 to build up the prototype system. The team developed two wireless transmission systems with one being chosen to take into the full prototype stage. Other items that the team developed include LED arrays and sensors. The entire budget was spent with a

good portion going to the two transmission systems. Spare parts and shipping and handling fees also added to the final budget. Appendix J lists how the project funds were dispersed.

The team was given constraints on how much the project should cost in the developmental stage and also if the product ever reached full-scale production. With the sensors that the team built and the parts selected, the final product is under the $200 initial prototype budget that the requirements specify. The transmitter, receiver and sensors run about $160 dollars, and a set of LED arrays would run about $40-50 to buy or manufacture. The following figure shows the prototype cost of the entire system. Appendix K contains tables which state the parts purchased along with tables that lay out the costs associated in building the individual assemblies and sensors.

CATEGORY	COST
Transmitter Assembly	$54
Receiver Assembly	$85
IR distance sensor	$15
Momentary contact lever switch	$4.5
Hall-effect sensor	$3.5
Push Button Sensor	$3
Phototransistor sensor	$2.5
TOTAL System Cost	$146-160

Fig. 10 Total prototype system cost. (final cost is the sum of the transmitter and receiver, along with one brake and one turn signal sensor)

5.3 Final Sensor Functionally

The five sensors that we took to the final prototype status have worked well in the limited time we have had to test them. The input sensors also tested very well in a variety of vehicle styles, years and makes. The sensors were tested on two full-size Ford trucks, along with a full-size Chevy and Dodge truck. The only issue that may be a problem is adapting the sensors to vehicles that were not a part of the test cases, and thus would require modifications to the sensors mounting to work properly in other applications. One example of this would be the need to change the resistor values in the schmitt trigger to accommodate vehicles that have more distance to the firewall than the test cases. Another example would be the creation of a mounting bracket to meet a specific application where the sensor could not be mounted easily using the methods that were successful in the test cases.

5.4 Final Project Status

The entire prototype system has been completed and is in a full working state. The System was tested inside a vehicle and functioned flawlessly. The system operated perfectly at 200 feet, so the 100 feet requirement has been met with flying colors. The LED arrays have a brighter intensity than current production vehicles so they also meet our requirements. All the initial tests of the sensors seem to make the product look viable, as the front signaling box performs well using both the sensors and the manual switches on the control box. The receiver box containing the output logic and light driver circuitry also is

in full functioning order. Further testing of the product is recommended before it is put into unrestricted highway usage.

APPENDICIES

Appendix A

PALASM code that was used to program the PAL with the output logic.

```
                ************************************
                *     PALASM PARSER LISTING        *
                ************************************
LINE #   |----+----1----+----2----+----3----+----4----+----5----+----6----+
  1      |;PALASM Design Description
  2      |
  3      |;---------------------------------- Declaration Segment ------------
  4      |TITLE    Trial 1
  5      |PATTERN
  6      |REVISION
  7      |AUTHOR   us
  8      |COMPANY
  9      |DATE     02/17/02
 10      |
 11      |CHIP  _trial1  PALCE16V8
 12      |
 13      |;---------------------------------- PIN Declarations ---------------
 14      |PIN  2   L                                ;Leftturn
 15      |PIN  3   B                                ;Brake
 16      |PIN  4   R                                ;Rightturn
 17      |PIN  5   C                                ;Continuous
 18      |PIN  6   A                                ;CLOCK2
 19      |PIN  14  RF                                ;RIGHTFET
 20      |PIN  15  LF                                ;LEFTFET
 21      |
 22      |;---------------------------------- Boolean Equation Segment ------
 23      |EQUATIONS
 24      |LF = (B * /L) + (A * L) + C + (L * R * B)
 25      |RF = (B * /R) + (A * R) + C + (L * R * B)
 26      |;---------------------------------- Simulation Segment ------------
 27      |SIMULATION
 28      |TRACE_ON L B R C A RF LF
 29      |
 30      |SETF A /L /B /R /C
 31      |CHECK /RF /LF
 32      |
 33      |SETF A /L /B /R C
 34      |CHECK RF LF
 35      |
```

```
36  |SETF A /L /B R /C
37  |CHECK RF /LF
38  |
39  |SETF A /L /B R C
40  |CHECK RF LF
41  |
42  |SETF A /L B /R /C
43  |CHECK RF LF
44  |
45  |SETF A /L B /R C
46  |CHECK RF LF
47  |
48  |SETF A /L B R /C
49  |CHECK RF LF
50  |
51  |SETF A /L B R C
52  |CHECK RF LF
53  |
54  |SETF A L /B /R /C
55  |CHECK /RF LF
56  |
57  |SETF A L /B /R C
58  |CHECK RF LF
59  |
60  |SETF A L /B R /C
61  |CHECK RF LF
62  |
63  |SETF A L /B R C
64  |CHECK RF LF
65  |
66  |SETF A L B /R /C
67  |CHECK RF LF
68  |
69  |SETF A L B /R C
70  |CHECK RF LF
71  |
72  |SETF A L B R /C
73  |CHECK RF LF
74  |
75  |SETF A L B R C
76  |CHECK RF LF
77  |
78  |TRACE_OFF
```

Appendix B

Truth table of the inputs and outputs used for PALASM code.

Design 1 with flash and brake: 5 chips and 12 gates

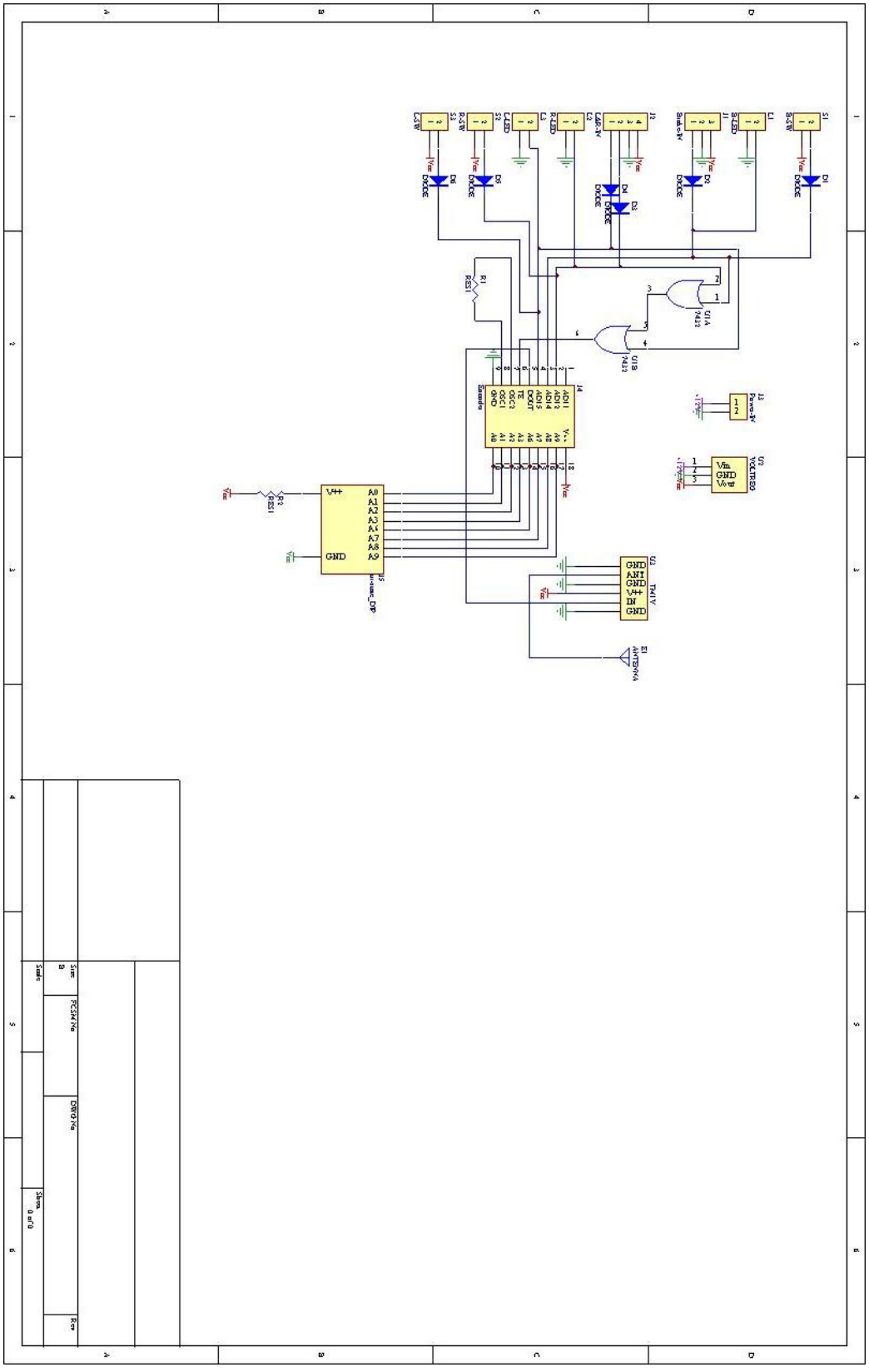
	INPUTS				OUTPUTS	
States	Left	Brake	Right	Continuous	LEFT	RIGHT
0	0	0	0	0	X	X
1	0	0	0	1	C	C
2	0	0	1	0	X	F
3	0	0	1	1	C	F
4	0	1	0	0	B	B
5	0	1	0	1	B	B
6	0	1	1	0	B	F
7	0	1	1	1	B	F
8	1	0	0	0	F	X
9	1	0	0	1	F	C
10	1	0	1	0	F	F
11	1	0	1	1	F	F
12	1	1	0	0	F	B
13	1	1	0	1	F	B
14	1	1	1	0	B	B
15	1	1	1	1	B	B

LEGEND
C=continous intensity
F=flash
B=Brake intensity
X=Off

Here is the design with Brake over-riding the flash
This design adds a few gates and chips to the previous design.

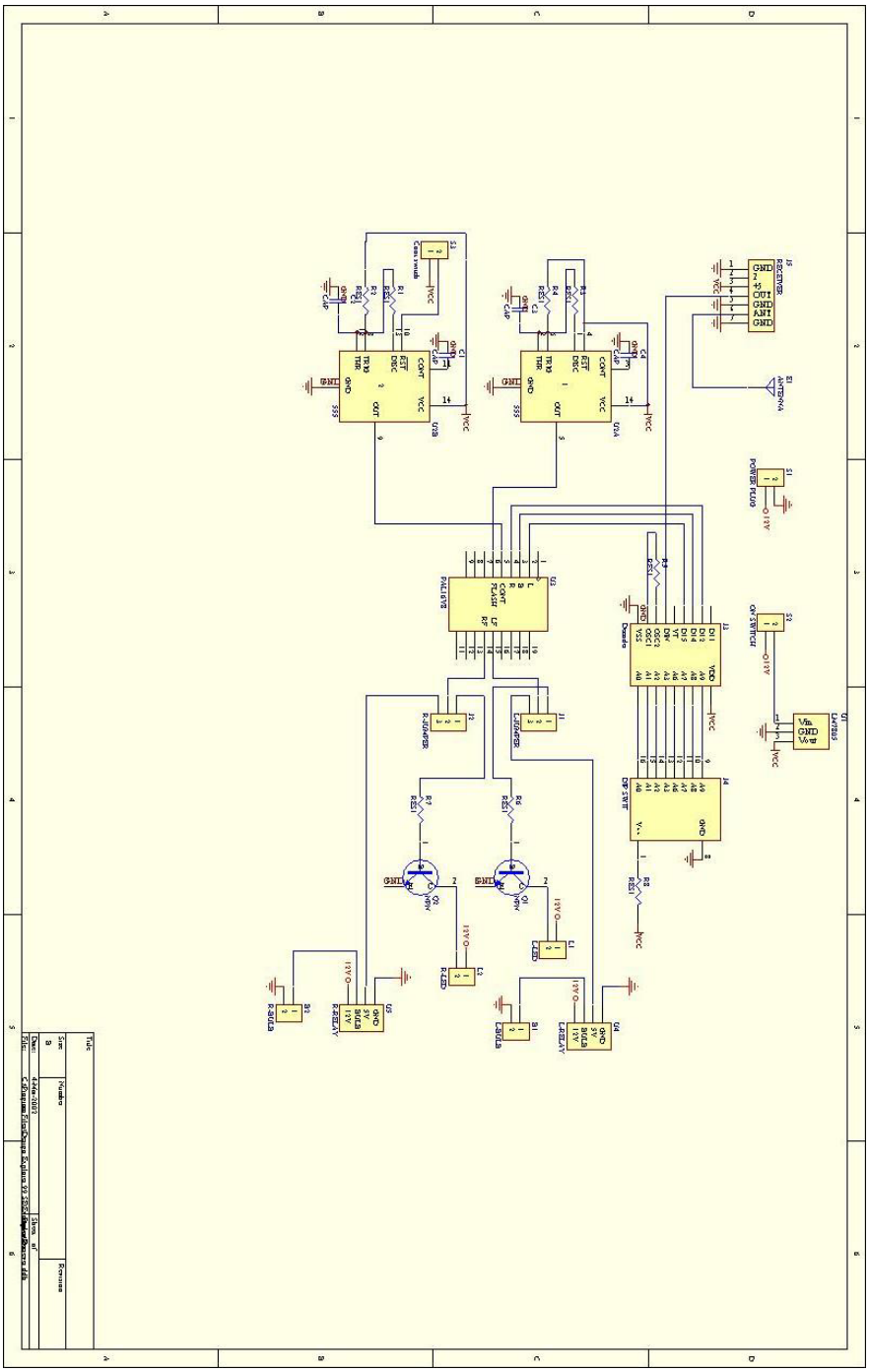
Appendix C

Transmitter circuit implemented in Protel.



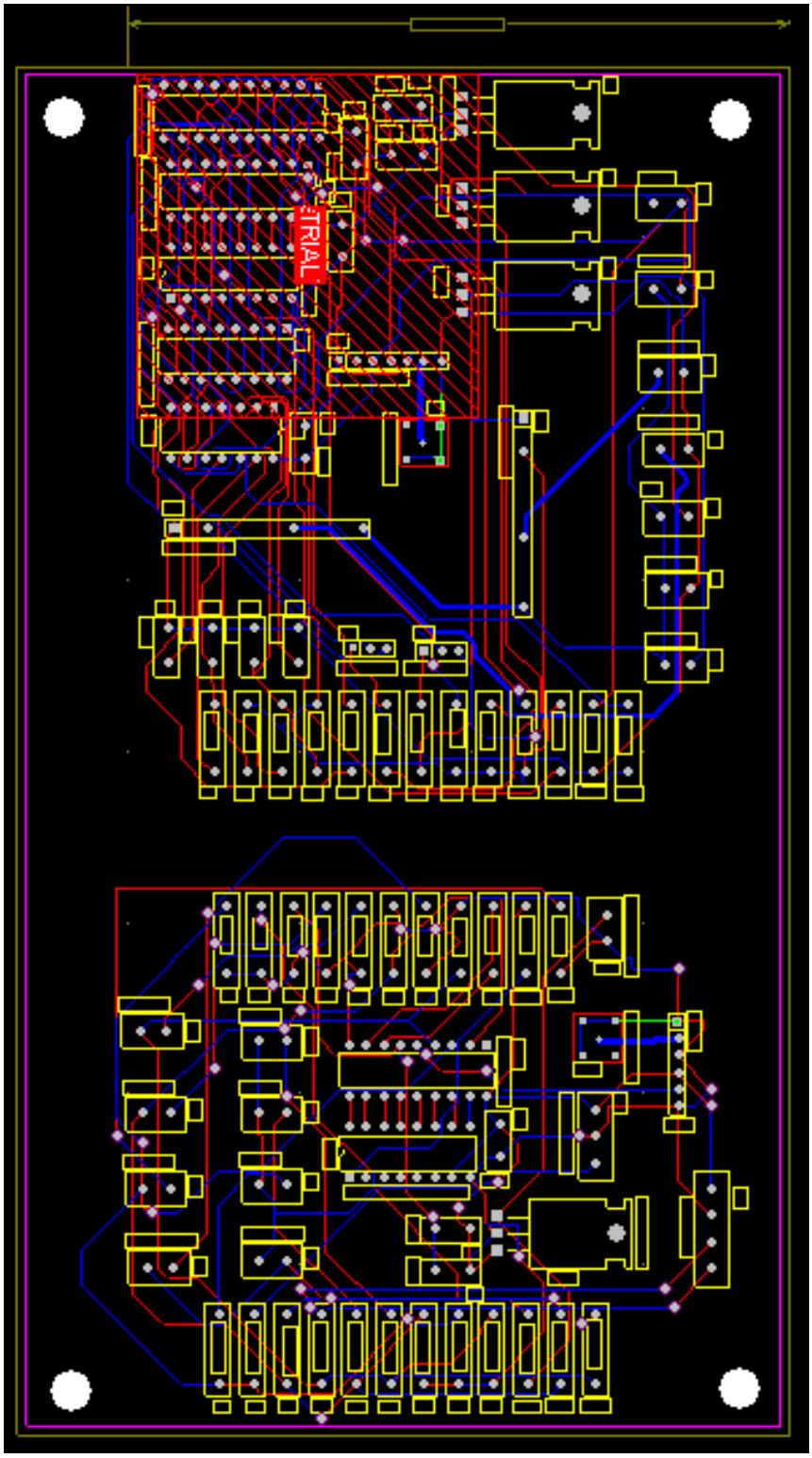
Appendix D

Receiver circuit implemented in Protel.



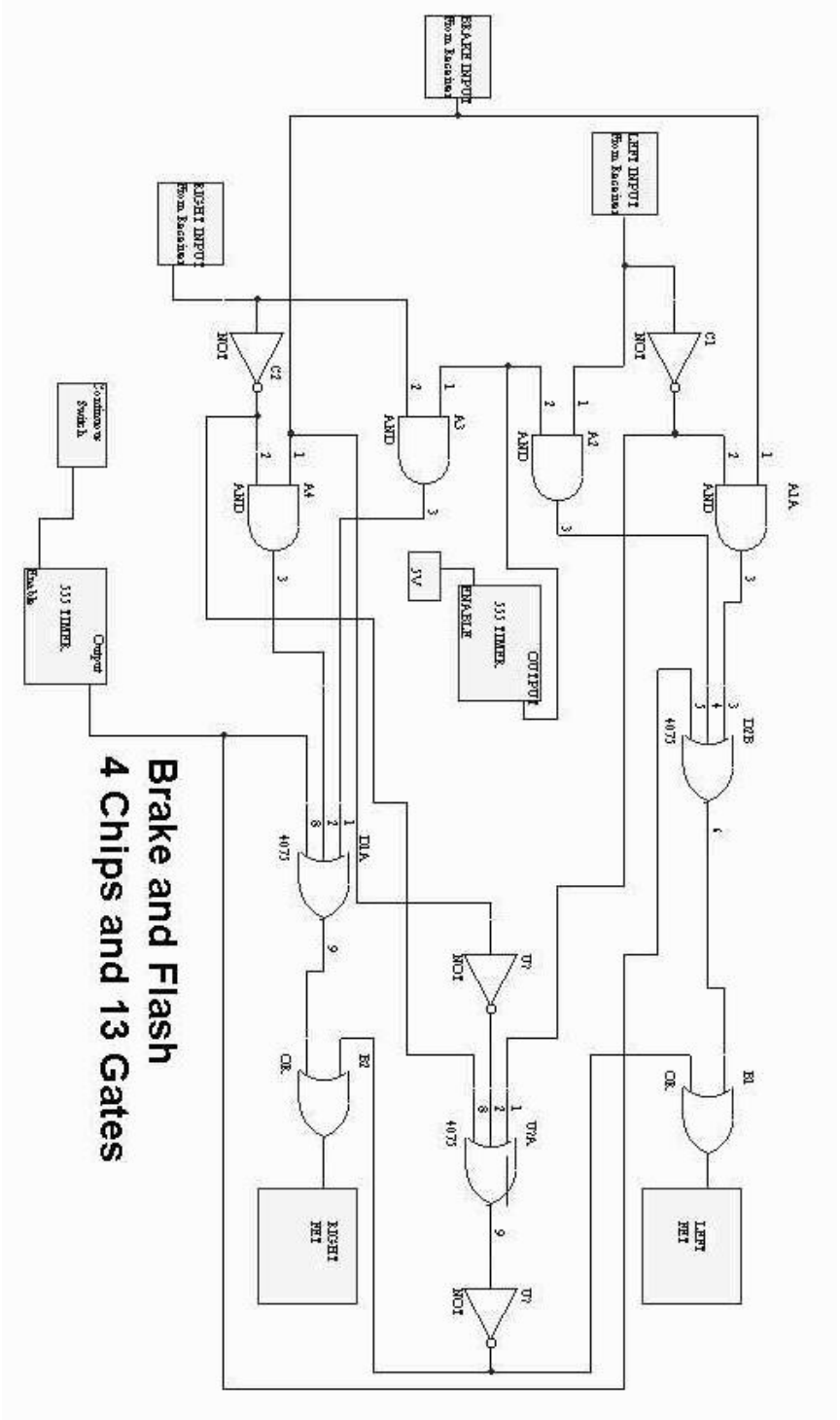
APPENDIX E

PCB layout of entire system. This layout comprises of the transmitter and receiver boards.



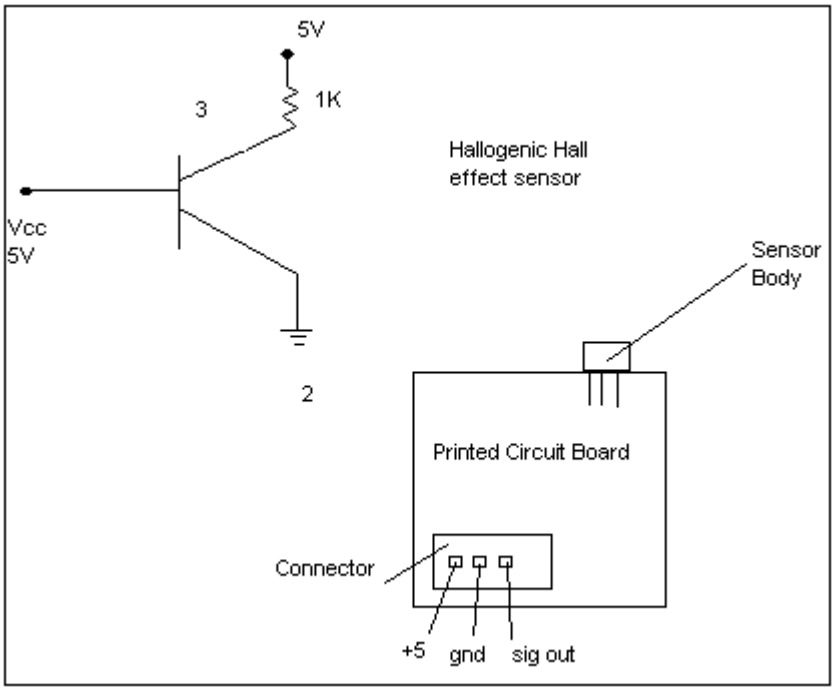
Appendix F

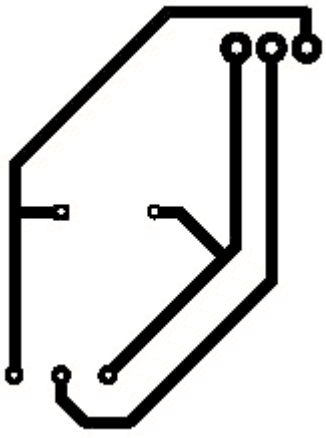
Output Logic that was later implemented on programmed chip.



APPENDIX G

Schematics of Hall-effect sensor circuit and board layout.

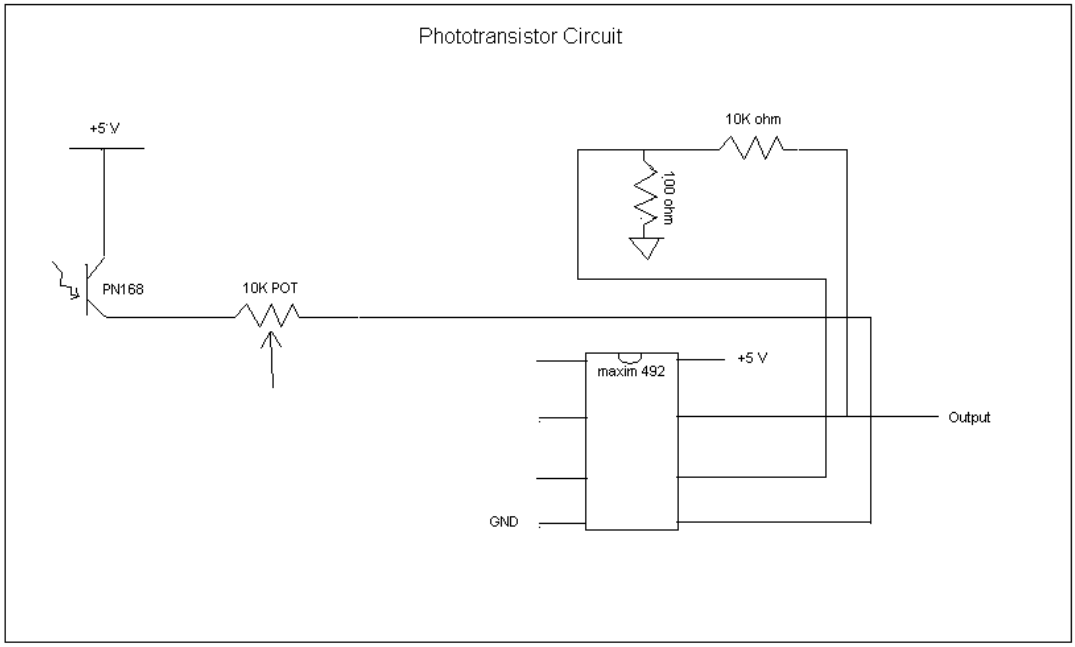




Printed circuit board for hall effect sensor

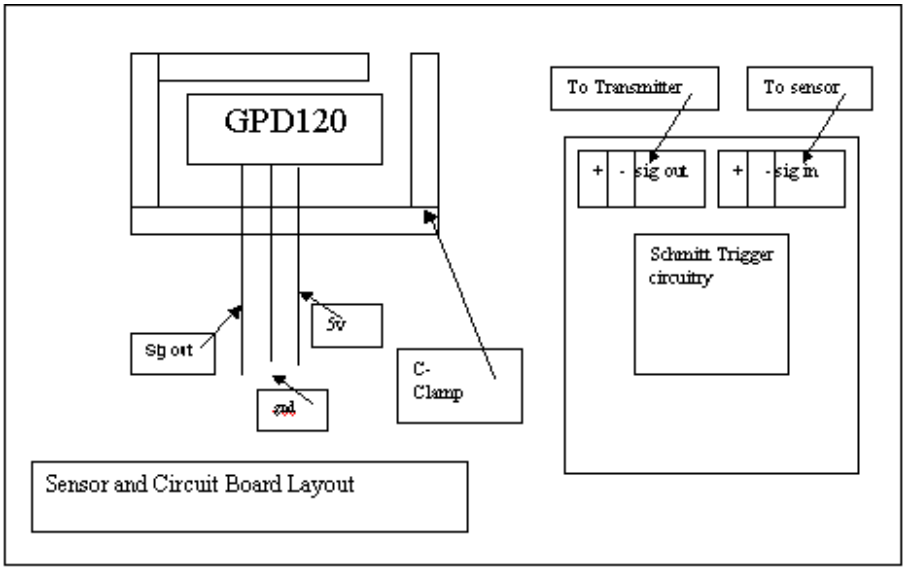
APPENDIX H

Phototransistor circuit schematic that includes a Max 492 Op-Amp and a phototransistor.

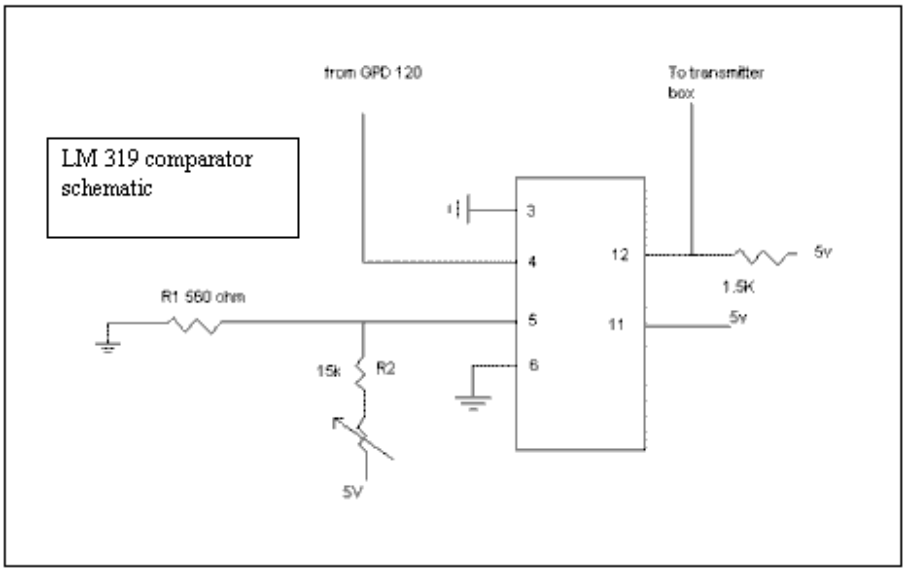


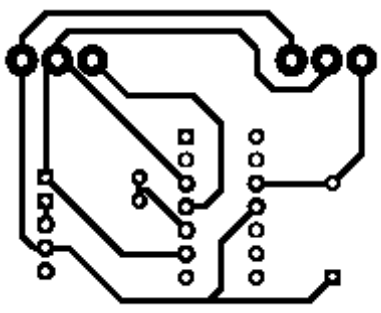
APPENDIX I

Distance sensor layout along with the schmitt trigger circuit layout and printed circuit board design.



Sensor and Circuit Board Layout





Printed Circuit Layout for IR Schmitt trigger

APPENDIX J

Spreadsheet of all parts purchased for the project, including vendors and costs.

Vender	Part #	Description	Qty	Price Ea	Total
Glolab	TM1V	Transmitter module, 418MHz	1	16.4	16.4
Glolab	RM1V	Receiver module, 418 MHz	1	23.75	23.75
Glolab	GL116A	Encoder / Decoder	4	12	48
Glolab	VDR	4 volt voltage detector reset	4	0.6	2.4
Glolab	CR4	4 MHz ceramic resonator	4	0.5	2
Glolab	Holtek HT-680	Encoder	6	2.87	17.22
Glolab	Holtek HT-694	Decoder	4	2.59	10.36
Glolab	WA418	Flexible Whip Antenna	2	7	14
Glolab	HT418	Helical Loop antenna	2	1	2
Glolab	RE200B	Pyroelectric Infrared Sensor.	2	4	8
Glolab	FL65	Infrared Fresnel lens.	1	4	4
Glolab	S211FL	Machined enclosure	1	9	9
Newark	50N267	Toggle switch	1	4.72	4.72
Newark	07F036	Lever switch snap action	2	5.51	11.02
Newark	24F209	Button Contact switch	1	4.23	4.23
Newark	04F1089	Snap dome Push button switch	3	2.25	6.75
Newark	83F5353	Detection switch d3k series	4	1.35	5.4
Newark	46F2946	8-position DIP 2 state	5	1.22	6.1
Newark	08F2612	Hallogenic hall effect OH09OU	2	1.58	3.16
Newark	93F6655	hall effect TO92	2	1.81	3.62
Newark	50F5176	hall effect TO92	2	1.92	3.84
Newark	16N8553	tri-state dip switches	3	3.04	9.12
Newark	65F2133	14 pin dip socket	6	0.18	1.08
Newark	51F1543	Header st. post	10	0.27	2.7
Newark	05B7184	pwr Mosfet	8	0.67	5.36
Newark	95B1530	Snap action switches	2	1.97	3.94
Digi-Key	FCTN-RLY4XX-433	Reciever	1	59.85	59.85
Digi-Key	TXM-433-LC	LC transmitter	2	6.9	13.8
Digi-Key	ANT-433-PW-QW	Quarter wave Antenna	1	6.7	6.7
Digi-Key	296-6504-5-ND	IC DUAL TIMER 14-DIP	3	0.75	2.25
Digi-Key	ED3120-ND	IC SOCKET 20PIN MS	2	0.81	1.62
Digi-Key	CC1205-ND	I/O MODULE 60VDC OUTPUT 3A	2	7.33	14.66
Digi-Key	K4A16-ND	TRIM POT 1M 8MM	5	0.31	1.55
Digi-Key	36G15-ND	TRIM POT 100K OHM CERMET	3	0.56	1.68
Digi-Key	A4027-ND	CONN SMB PLUG NICK RG174	2	5.05	10.1
Digi-Key	A4042-ND	CONN SMB JACK VERTICAL	2	3.2	6.4
Digi-Key	WM3204-ND	6 ckt pin header reciever socket	2	0.81	1.62
Digi-Key	WM3205-ND	7 ckt pin header trans socket	2	0.92	1.84
Digi-Key	A1490-ND	2 CT PRE-TIN PCB SOCKET hdr	6	0.35	2.1

Digi-Key	A1492-ND	3 CT PRE-TIN PCB SOCKET hdr	4	0.44	1.76
Digi-Key	A-1494-ND	4 CT PRE-TIN PCB SOCKET hdr	2	0.37	0.74
Digi-Key	A1448-ND	CONN PLUG IN-LINE 2 CIRCUIT	6	0.17	1.02
Digi-Key	A1550-ND	CONN PLUG IN-LINE 3 CIRCUIT	4	0.16	0.64
Digi-Key	A1552-ND	CONN PLUG IN-LINE 4 CIRCUIT	2	0.17	0.34
Digi-Key	A1436-ND	CONN PIN 24-18 AWG MALE	30	0.102	3.06
Digi-Key	A1437-ND	BRASS/PRE-TIN SOCKET	30	0.095	2.85
Digi-Key	WM4640-ND	HEADER 2 POS .156 FRIC LOCK	16	0.26	4.16
Digi-Key	WM4641-ND	HEADER 3 POS .156 FRIC LOCK	6	0.37	2.22
Digi-Key	WM4642-ND	HEADER 4 POS .156 FRIC LOCK	4	0.46	1.84
Digi-Key	WM2111-ND	CONN HOUS 2 POS .156"	16	0.22	3.52
Digi-Key	WM2112-ND	CONN HOUS 3 POS .156"	6	0.31	1.86
Digi-Key	WM2113-ND	CONN HOUS 4 POS .156"	4	0.37	1.48
Digi-Key	WM2300-ND	CRIMP TERMINAL .156	70	0.06	4.2
Digi-Key	WM5738-ND	CONN BARRIER STRIP 2POS	1	2.36	2.36
Digi-Key	365-1001-ND	SENSOR HALLOGIC HALL EFF	3	2.94	8.82
Digi-Key	10W-2-ND	RES 10 OHM 2W 5% METAL	2	0.23	0.46
Digi-Key	20W-2-ND	RES 20 OHM 2W 5% METAL	2	0.23	0.46
Digi-Key	39W-2-ND	RES 39 OHM 2W 5% METAL	2	0.23	0.46
Digi-Key	82W-2-ND	RES 82 OHM 2W 5% METAL	2	0.23	0.46
Digi-Key	1N5237BDICT-ND	DIODE ZENER 8.2V 500MW 5%	2	0.36	0.72
Digi-Key	1N4738ADICT-ND	DIODE ZENER 8.2V 1W 5%	2	0.36	0.72
Digi-Key	67-1610-ND	1500 MCD Red LED	50	0.24	12
Digi-Key	67-1612-ND	2800 MCD Red LED	50	0.45	22.5
Digi-Key	STA-KIT-ND	ASSORTED SHRINK TUBING	1	10.95	10.95
Linx-Technologies	HT-640	Holtek encod for surf. mt trans	3	3	9
Sharp	GPD120	IR Distance Sensor	1	Est 10	10
Radio Shack	Misc	2 sided copper board	1	4.02	4.02
Radio Shack	270-1017	1/4 Amp fuses	4	0.5	1.99
Radio Shack	270-1281	Inline fuse holder	2	1.79	3.58
Radio Shack	278-1655	Nylon Zip Ties	30	.13	4.00
Radio Shack	270-1801	3x 2x1” project enclosure	1	1.99	1.99
Radio Shack	Misc	Magnets	2	0.99	1.98
E-Switch	SM0850501F025V1A	Contact Switches Green	2	sample	x.xx
E-Switch	100SP3T6B11M1QE	3 way toggle switch	6	sample	x.xx
E-Switch	KAT1108E	8 pin tri state dips	3	sample	x.xx
ACE Hardware	Misc.	c-clamps for mounting	2	1.75	3.5
Walmart	Misc.	mini fuse holder	2	1.97	3.94
Walmart	Misc.	Velcro 2 rolls	2	3	6
TCC	Misc.	Color print charges	10	0.5	5
Tech Library	Misc.	Transparancies	20	0.3	6
Analog Devices	Adxl202jqc	Tilt sensor	1	24.94	24.94
		SUB_TOTAL			640.23
		Shipping/ handling/ taxes			89.19
		GRAND TOTAL			729.42

APPENDIX K

Build sheet for transmitter and receiver subsystems, along with sensor build sheets.

	Transmitter Parts List			
TM1V	Transmitter module, 418MHz	1	16.4	16.4
Holtek HT-680	Encoder	1	2.87	2.87
WA418	Flexible Whip Antenna	1	7	7
A4027-ND	CONN SMB PLUG NICK RG174	1	5.05	5.05
A4042-ND	CONN SMB JACK VERTICAL	1	3.2	3.2
WM3205-ND	7 ckt pin header trans socket	1	0.92	0.92
A1492-ND	3 CT PRE-TIN PCB SOCKET hdr	1	0.44	0.44
A-1494-ND	4 CT PRE-TIN PCB SOCKET hdr	1	0.37	0.37
WM4640-ND	HEADER 2 POS .156 FRIC LOCK	5	0.26	1.3
WM4641-ND	HEADER 3 POS .156 FRIC LOCK	1	0.37	0.37
WM4642-ND	HEADER 4 POS .156 FRIC LOCK	1	0.46	0.46
WM2111-ND	CONN HOUS 2 POS .156"	5	0.22	1.1
WM2112-ND	CONN HOUS 3 POS .156"	1	0.31	0.31
WM2113-ND	CONN HOUS 4 POS .156"	1	0.37	0.37
WM2300-ND	CRIMP TERMINAL .156	10	0.06	0.6
270-1017	¼ Amp fuses	1	0.5	0.5
270-1281	Inline fuse holder	1	1.79	1.79
100SP3T6B11M1QE	Toggle switches	4	est. 2.00	8
KAT1108E	8 pin tri state dips	1	est 2.50	2.5
LM7805	5v regulator	1	0.3	0.3
10uf cap	Electrolytic cap	1	0.05	0.05
.01uf cap	Cap	2	0.05	0.1
1n914	Diodes	12	0.02	0.24
Resistors	Various resistors	11	0.02	0.22
	TOTAL			54.46

	Receiver Parts List			
RM1V	Receiver module, 418 MHz	1	23.75	23.75
Holtek HT-694	Decoder	4	2.59	10.36
WA418	Flexible Whip Antenna	1	7	7
296-6504-5-ND	IC DUAL TIMER 14-DIP	1	0.75	0.75
ED3120-ND	IC SOCKET 20PIN MS	1	0.81	0.81
CC1205-ND	I/O MODULE 60VDC OUTPUT 3A	2	7.33	14.66
A4027-ND	CONN SMB PLUG NICK RG174	1	5.05	5.05
A4042-ND	CONN SMB JACK VERTICAL	1	3.2	3.2
WM3204-ND	6 ckt pin header reciever socket	1	0.81	0.81
A1490-ND	2 CT PRE-TIN PCB SOCKET hdr	2	0.35	0.7

A1448-ND	CONN PLUG IN-LINE 2 CIRCUIT	2	0.17	0.34
A1436-ND	CONN PIN 24-18 AWG MALE	4	0.102	0.4
WM4640-ND	HEADER 2 POS .156 FRIC LOCK	7	0.26	1.82
WM2111-ND	CONN HOUS 2 POS .156"	7	0.22	1.6
WM2300-ND	CRIMP TERMINAL .156	14	0.06	0.74
270-1017	¼ Amp fuses	1	0.5	0.5
270-1281	Inline fuse holder	1	1.79	1.79
100SP3T6B11M1QE	Toggle switches	2	est. 2.00	4
KAT1108E	8 pin tri state dips	2	est 2.50	5
LM7805	5v regulator	1	0.3	0.3
10uf cap	Electrolitic cap	1	0.05	0.05
.033uf cap	Cap	4	0.05	0.2
.068uf cap	Cap	4	0.05	0.2
Resistors	Various resistors	13	0.02	0.26
TIP121	NPN power transistor	2	0.3	0.6
	TOTAL			84.89

	IR Distance Sensor			
A1550-ND	CONN PLUG IN-LINE 3 CIRCUIT	1	0.16	0.16
A1436-ND	CONN PIN 24-18 AWG MALE	3	0.102	0.33
WM4641-ND	HEADER 3 POS .156 FRIC LOCK	2	0.37	0.74
WM2112-ND	CONN HOUS 3 POS .156"	2	0.31	0.62
WM2300-ND	CRIMP TERMINAL .156	6	0.06	0.36
GPD120	IR Distance Sensor	1	Est 10	10
misc.	c-clamps for mounting	1	1.75	1.75
Resistors	5.4k, 16k	2	0.02	0.04
.1uf cap	Cap	1	0.05	0.05
10k Adj Pot	Adjustable pot	1	est .5	0.5
LM319	Comparator	1	est .5	0.5
	TOTAL			15.05

	Brake Contact Pad			
04F1089	Snap dome Push button switch	1	2.25	2.25
A1436-ND	CONN PIN 24-18 AWG MALE	2	0.102	0.22
A1550-ND	CONN PLUG IN-LINE 3 CIRCUIT	1	0.16	0.16
xx	Plexiglass pad	1	.5 est	0.5
	TOTAL			3.13

	Turn Signal Contact Switch			
SM0850501F025V1A	Contact Switches Green	2	est 2.00	2
A1552-ND	CONN PLUG IN-LINE 4 CIRCUIT	1	0.17	0.17
A1436-ND	CONN PIN 24-18 AWG MALE	4	0.102	0.4
	TOTAL			4.57

	Hall Effect Sensor			
08F2612	Hallogenic hall effect OH09OU	1	1.58	1.58
A1550-ND	CONN PLUG IN-LINE 3 CIRCUIT	1	0.16	0.16
A1436-ND	CONN PIN 24-18 AWG MALE	3	0.102	0.33
WM4641-ND	HEADER 3 POS .156 FRIC LOCK	1	0.37	0.37
WM2112-ND	CONN HOUS 3 POS .156"	1	0.31	0.31
WM2300-ND	CRIMP TERMINAL .156	3	0.06	0.18
xx	Single sided circuit board 3x2"	1	0.25	0.25
xx	Resistor 1k	1	0.05	0.05
	TOTAL			3.55

	Phototransistor Sensor			
PN 168	PHOTOTTRANSISTOR	2	0.05	0.1
MAX 492	MAX 492 OP-AMP	2	est 0.50	1
A-1494-ND	4 CT PRE-TIN PCB SOCKET hdr	1	0.37	0.37
A1436-ND	CONN PIN 24-18 AWG MALE	4	0.102	0.4
xx	Board layout material 2" x 4"	1	0.5	0.5
	TOTAL			2.37

APPENDIX L

Literature review of major web sites referenced for parts, schematics and project information.

---

http://www.holtek.com/pdf/consumer/3_18e.PDF
-Encoder

http://www.linxtechnologies.com/ldocs/f_prod.html
-Transmitter/Receiver pair

http://www.glolab.com/modules/module1.html
-Transmitter/Receiver pair

http://www.digikey.com
-Parts supplier

http://www.e-switch.com
-Parts supplier

http://www.newark.com
-Parts supplier

http://www.analogdevices.com
-Tilt sensor

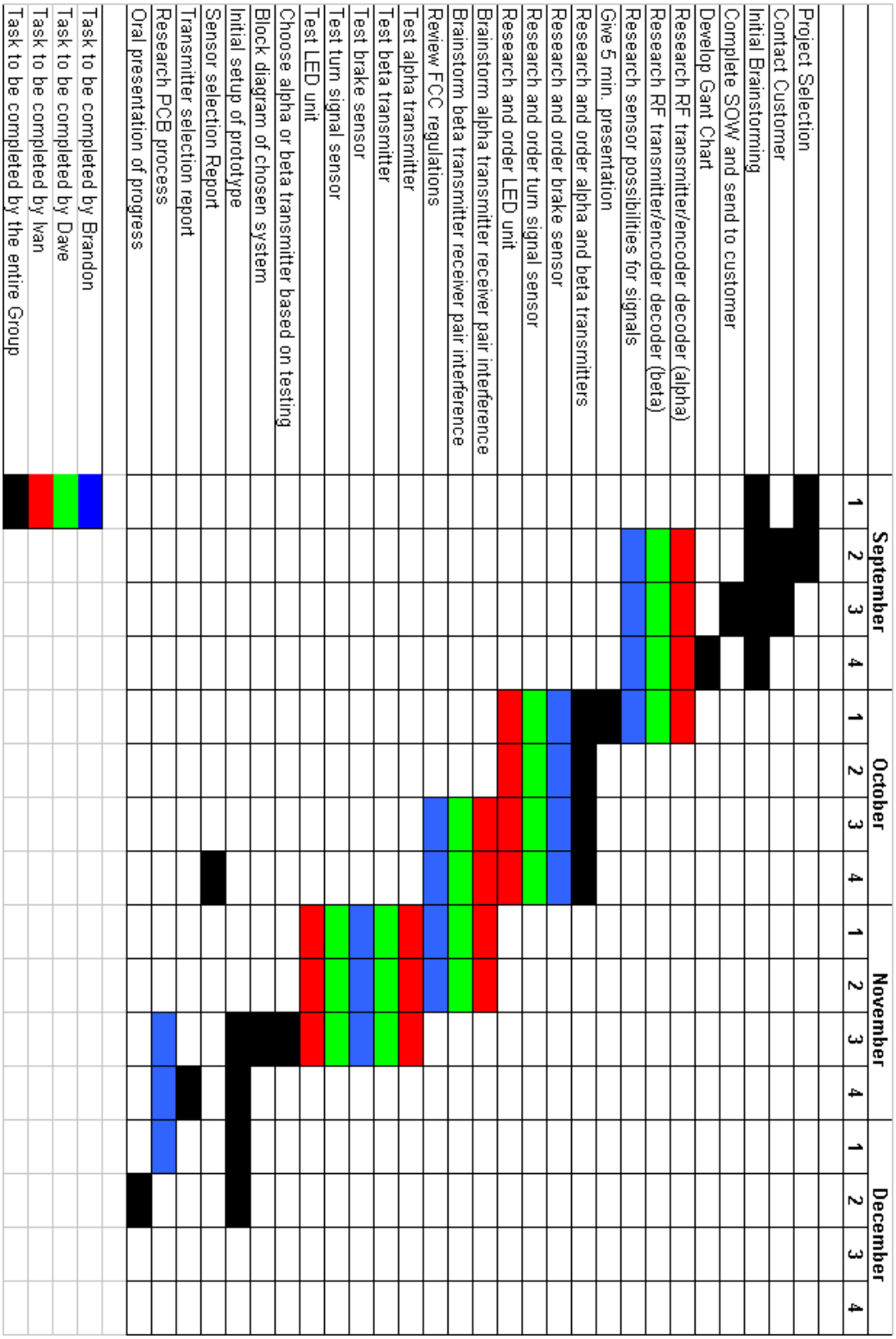
http://www.optekinc.com/book/pdf/oh090u.pdf
-Hallogenic Hall effect sensor OH090U

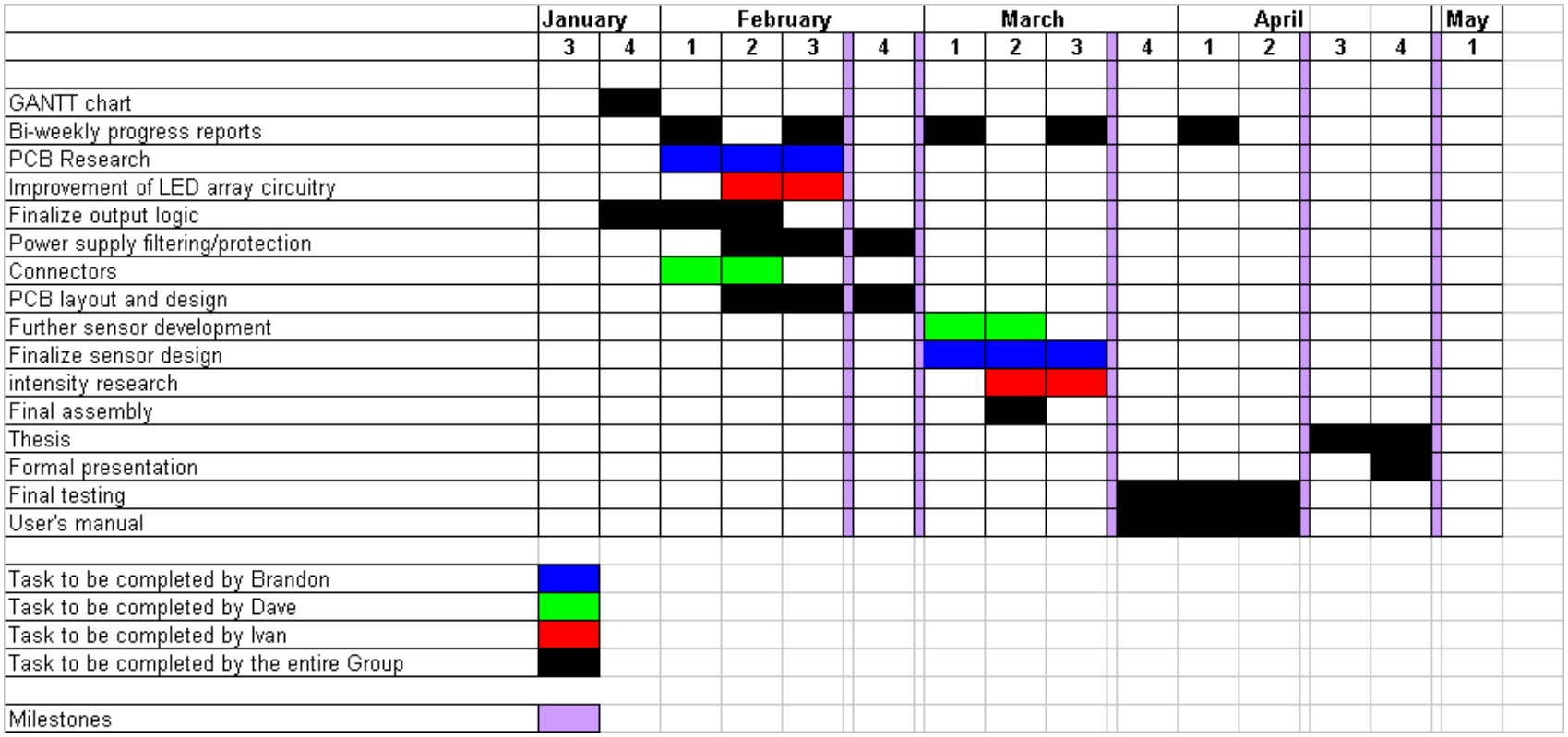
http://www.ee.nmt.edu/~wedeward/EE382/SP01/ee382.html
-GP2D120-distance sensor

http://www.fcc.gov
-FCC regulations literature

APPENDIX M

Gantt charts for the first and second semesters.




January
February
March
April
May
3 4 1 2 3 4 1 2 3 4 1 2 3 4 1
GANTT chart
Bi-weekly progress reports
PCB Research
Improvement of LED array circuitry
Finalize output logic
Power supply filtering/protection
Connectors
PCB layout and design
Further sensor development
Finalize sensor design
intensity research
Final assembly
Thesis
Formal presentation
Final testing
User's manual
Task to be completed by Brandon
Task to be completed by Dave
Task to be completed by Ivan
Task to be completed by the entire Group
Milestones

Small Radio Telescope Project

A Thesis

Presented to the Faculty of

the Electrical Engineering Department

of

New Mexico Tech

By

Gerald Bivens, Sam Field, and Brian Rajala

In partial fulfillment of the requirements

for the course

EE-482 Senior Design Project II

May, 2002

Abstract

The National Radio Astronomy Observatory (NRAO) is a publicly run institution under the direction of the National Science Foundation. NRAO runs a number of world-class radio astronomy research facilities. One of these is the Very Large Array (VLA), located in the San Augustin Plains west of Socorro NM.

The VLA has been receiving record numbers of visitors and because of this, public education about radio astronomy has risen in importance for NRAO. For this reason, Smal Radio Telscope (SRT) is being built for NRAO for use at the VLA Visitors Center, where it will play a major role as an interactive display and educational tool to explain radio astronomy to the public.

The goal of the SRT project is to build a small radio telescope, which will be installed for use by visitors to the VLA. Project requirements include building and testing the telescope kit and designing an interface, which is both easy to use and educational. The system must provide visitors with a choice of objects to observe and educate them in the basics of radio astronomy. At the same time, it must require minimal maintenance and supervision, and must be resistant to abuse the public is likely to inflict.

To complete the project, we divided up major tasks among team members. The main areas of work involved developing educational material, arranging for installation of the antenna, developing a control program, and implementing a touchscreen interface. Our team has achieved all of the SRT project's goals except for completing final installation at the VLA site. There has been a delay in obtaining a kiosk to house the control computer. Arrangements have been made to complete installation within two weeks of the date of this thesis.

Acknowledgments

First, we would like to thank the National Radio Astronomy Observatory for giving us the opportunity to work on such a noteworthy project.
In addition, Clint Janes, with whom we worked in obtaining invaluable resources for this project, earns special mention because he offered advice and guidance throughout the development of this work.
Additionally, Robyn Harrison played a major role in shaping the final product because even though she initiated the idea, our team was given free reign to develop the product as we could conceptualize it.
The following is a list of persons who all deserve our thanks and we ask at this point to forgive us if we have mistakenly forgotten anyone. Thank you.

With NRAO
Clint Janes- Electronics Head, Senior design projects implementation.
Robyn Harrison – NRAO Education Liaison
Marie Glendenning- computer
Sheila Reasner shipments Purchasing agent
Pat Van Buskirk-Java and network consultation
Phil Dooley, Intranet-Weather Station
Lothar Dahlmeyer-Password control
Richard Rupp- delivery of kit to site, helping with kit installation
James Robnett-consultation about timeserver

With VLA
Jon Spargo (NRAO/VLA Safety Officer)
Bob Broilo (VLA Electrical Division Head)
Guy Stanzione (Grounds Projects)
Jaime Montero (Lead Electrician)

Other Key Personnel
Jim Shepherd-Outsourced Carpenter for kiosk
Steve Wasson- programming consultation
Physics Department chair, Dr. Minschwaner-permission for SRT antenna
Dr. Hasan M. Shanechi (Faculty Project Advisor) and laboratory space
Dr. Scott W. Teare (Senior Project Coordinator)
Dan Jones (NMT Architect, Physical-Plant) Consultation for installation
Chris Kaiser: contact at MassWorks
8401 Washington Pl. N.E.
Albuquerque, New Mexico 87113
Chris Pauli -EE Dept. Network Administrator
Carrol Teel-EE Dept. Secretary
Pete Martinez-communications consultation
Dr. Dan Klinglesmith, NMT Physics, Astronomy,
Basic Astronomy Science Information consultation

Table of Contents

List of Figures

1 Introduction

This paper is in partial fulfillment of course requirements for the New Mexico Institute of Mining and Technology Department of Electrical Engineering's Senior Design course. This is a detailed description of the work accomplished by Gerald Bivens, Sam Field, and Brian Rajala on the National Radio Astronomy Observatory's Small Radio Telescope project between August 2001 and May 2002.

1.1 **NMT Senior Design Course**

As part of its requirements for graduation, the Electrical Engineering Department at New Mexico Tech offers a two-semester design course. At the beginning of the course, seniors in the department are offered a number of projects by outside entities representing a wide variety of public and private enterprises. Students are placed in groups of two to four members based on their project selections. Each team is responsible for meeting with their customer to draft a statement of work outlining the customer's requirements for the project. The rest of the year is spent working to meet customer demands. In addition to work for the customer, regular presentations and progress reports are required. This work culminates in a presentation of group achievements to Electrical Engineering faculty and the department's corporate advisory board, as well as written documentation of the project. Grades are based on customer satisfaction, regular written and oral communication, and overall team performance.

1.2 The National Radio Astronomy Observatory

Among the customers offering projects was the National Radio Astronomy Observatory (NRAO). The NRAO is a publicly funded research facility operated by Associated Universities Incorporated (AUI) for the National Science Foundation (NSF). Based in Charlottesville, Virginia, NRAO operates a number of world-class radio telescopes including the Very Large Array (VLA) located west of Socorro, New Mexico. Composed of 27 antennas in the San Augustin plains, the VLA is a one of a kind instrument used by scientists from around the world to conduct cutting edge research.

The VLA has a visitors center, which NRAO is currently in the process of upgrading. One of the plans in the expansion is to put a small, single dish radio telescope in the Visitor Center where the public can access it. The radio telescope would be one of the first stops on the self guided tour and would educate people about radio astronomy so that they could get the most out of their visit to the VLA.

1.3 The Small Radio Telescope

The project offered by NRAO involved the adaptation of an existing radio telescope kit intended for students and amateur astronomers for use by the public at the VLA Visitors Center. The Small Radio Telescope (SRT) kit was developed by the Massachusetts Institute of Technology (MIT) as a fully functional radio telescope and is sold for use primarily by universities and amateur astronomers. The kit consists of a commercially available satellite dish, originally made for satellite television systems, along with a custom azimuth/elevation drive, L-band receiver, and a controller which allows the telescope to be interfaced with a PC. The control software is furnished as open source Java code and provides full control of the SRT through a simple graphical user interface (GUI).

The SRT (Figure 1.3.1) collects data from the universe in the radio part of the electromagnetic spectrum. The radio spectrum ranges from 1mm to 30 meters. Data can be collected in both the day and night. The emissions that are measured by the SRT come from the astronomical objects that are in the universe. The radio telescope uses a large parabolic metal dish to reflect radio waves to the sub-reflector. The sub-reflector then directs the radiation to the feed at the center of the reflector. Then the signal is amplified through the receiver. A computer then processes the amplified radio signal. The SRT will be tracking, for example, the Sun, Cygnus A, and Cass A., which are strong radio sources.

Figure 1.3.1 Small Radio Telescope on top of Workman Center

NRAO ordered an SRT kit with the intention of installing it at the VLA Visitors Center for use by the public as an interactive tutorial on radio astronomy. As furnished by MIT, the kit is not suitable for this application for a number of reasons. First, the control software is not intuitive and requires a significant amount of time to learn. Second, a mouse and keyboard are needed to run the telescope, both of which are subject to theft and damage and allow too much access to the computer running the telescope. Third, the software is intended for users with significant knowledge of radio astronomy, and is therefore of little value in educating the general public. Therefore, our project was to modify the kit to make it suitable for NRAO's needs. This involved making it easy for the public to use. The intent of the project is to demonstrate the functional operation of a radio telescope and provide a tutorial on radio astronomy for the public.

2 Overview of Radio Astronomy Project

2.1 Customer requirements

In order to find out what the project was all about we made contact with Clint Janes, Electronics Division Head and Robyn Harrison, Educational Liaison for NRAO. We first met with Robyn Harrison and discussed NRAO's needs. We found that we would be furnished with a kit to build a radio telescope, a development computer, and access to NRAO facilities. We would have to manipulate the software to track an assortment of astronomical objects and display telescope data in a manner that is both educational and easy to understand for the general public. We would have to implement an interface that would be foolproof and easy to use. The interface had to control the telescope without a keyboard or mouse, but would still allow public interaction. It was suggested that this be done with a touchscreen. The top priority of the entire project was to educate the public about radio astronomy.

The Project budget was briefly discussed and we were told that a distinct ceiling was not set, but all expenses must be approved by the NRAO. This would also include installation of "things like support piers, conduits, [and] electrical" (see e-mail Appendix-A page 63).

2.2 Assignment of Work

A clear idea for this open-ended project was obtained when we presented our initial ideas attending a meeting with Visitor Center Committee. From this meeting, we were also able to finalize the Statement Of Work contract. After outlining the customer's requirements for the SRT Project, we had to decide how to break up the project among the team. The logical divisions of work seemed to be in the areas of control software, touchscreen interface, educational tutorials and final installation design. In addition, some tasks were determined to be immediate needs that the entire group would address. These included:

- Set up development computer with network capabilities.
- Decide on control software options.
- Download NRAO web links and VLA Weather Station program in design.
- Acquire the Java compiler the open source Java code for SRT tracking software.
- Acquire Microsoft Office for educational slideshow presentations in PowerPoint.
- Obtain a touchscreen for input and control.
- Communicate vehemently with the vendors to remind them to send the SRT kit.
- Obtain permission for testing location.

  Viable locations discussed and debated.

Once we had a breakdown of the individual tasks we could assign them according to our levels of expertise in each area. Sam would work on the Java software, Brian would work on the touchscreen interface, and Gerald would be responsible for developing educational material and coordinating installation at the VLA.

3 Discussion of Individual and Group Accomplishments

After initial contact with our client, we devised methods of putting together the tasks that the project entailed. The first task was to break up the work among the group members. This was accomplished by evaluating our strengths and weaknesses. In doing this we tried to consider each group member's strengths. Sometimes our group was lacking any knowledge of a subject, but we plunged into research of the topic anyway, and found some valuable information. The following sections are our individual personal accountings of the SRT project, followed by a section describing work that performed as a group.

3.1 Gerald

From the start of this Senior Design project my tasks fell under two main categories: communication and education. It seemed that more effort went into communication with people to pool information and to organize the many tasks that would bring the SRT Project to completion. Many people were involved in completing this project and gave it a professional polished look. The education aspect was certainly a challenge, as I had no prior experience with radio or any other astronomy. The idea for educational tutorials from PowerPoint slides had its basis in many courses I have taken in my undergraduate studies at NMT.

Initial communication and coordinating efforts started with some personal contacts I had at Tech's Physical-Plant. I took the task of finding out about putting the antenna on a roof at Tech. We thought our hardware was due to arrive soon and getting an area laid out before the equipment arrived would be a good idea. At Tech's Physical Plant, I was put in touch with the architect for Tech. Together we surveyed the roof of the Workman building for installation of the SRT and discussed possible alternatives. We found we could run the cables down through the existing conduit. We checked the blueprints for access points for the cables. Since the Workman building was designed for experiments to be mounted on top of it along with penetrations for power, control and data cables to

various points in the building, the Workman roof seemed to be perfect. Roof mounting would avoid obstacles from the SRT looking at the sky. Another reason for roof-mounting was to reduce the possibilities of tampering with the dish since locked doors control the roof access.

We went through some more brainstorming of ideas with our project advisor and supervisor on topics such as, permissions for NRAO to have the antenna on Workman or another place on Tech property, liabilities, other design planning, other politics and even other possible sites like Etscorn Observatory and my house. We needed to find out if we would need administrative permission (Brown Hall), or if we would be infringing on any positions of Union workers from the P-Plant or other government workers and whether we would need work orders to complete our installation. Further tasks were assigned and

I got to find out about mechanical, cabling, and power requirements.
When we finally obtained permission for the roof-mounting, we had to decide whether or not to order the base for it. I also investigated the other technical aspects from data sheets for the SRT. From my literature search, I decided we did not need a base as the antenna would fit over the existing stanchion as in Figure 3.1.1.

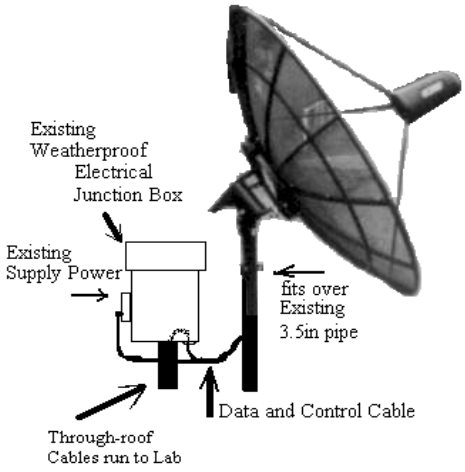


Figure 3.1.1. Concept Design for Mounting SRT

A literature search confirmed that since the parabolic dish is of a metal mesh construction, wind loading would not be a problem. The architect's only concern was for the dish to be secure. He was assured that it would be, and as a follow-up, he was contacted again when the dish was installed to confirm it was a secure installation.

Since the antenna was to be controlled from a lab on the second floor of the Workman building via a personal computer, about a hundred feet of control and signal cables had to be run between the antenna and the lab. It was necessary to find out which labs the cables would go through and for access to conduits on the third floor and obtain permission to run the cable in those existing conduits. One professor on the third floor would not allow access to the conduit run in his lab (due to sensitivity of his experiments), so an alternative route was needed and permission for routing the cable there was obtained. The Closed Circuit TV video cable was run through a different set of conduits and permission to run that cable was also obtained. I devised a bracket assembly for temporary mounting of the camera to the roof stanchions and hooked up a power cord.
Based on how the project goals were to be divided up, I went ahead and made a Gant chart to be approved by everyone. From this further division of tasks I was to work on the watchdog, which was required to restart the system in the event of a power failure. I had no idea how to find a watchdog, and after extensive literature searches and interviewing the network system administrator, I found out how we could restart our system without human interaction and without additional costs.

For the responsibility of education I started to educate myself in the basics of radio astronomy from websites and other reference material recommended by Robyn Harrison to develop the instructional displays. Preliminary ideas, like Physics and Radio Astronomy, for the tutorials were initiated and in the second semester I started compiling literature search information. I finally decided on three slide shows for tutorials and one slideshow for the project itself. The largest slide show is The Science of Radio Astronomy, which tells about some of the complex topics of radio astronomy and breaks them down to a basic high school grade level. How the SRT Works is the next largest

slide show, which describes the major parts and pieces of this small radio telescope. The Tracking slide show, which runs automatically and gives time for the antenna to start tracking a source before the viewer gets bored, tells of what the SRT is doing and what will be seen. The final slide show, The SRT Project, was developed for presentations.

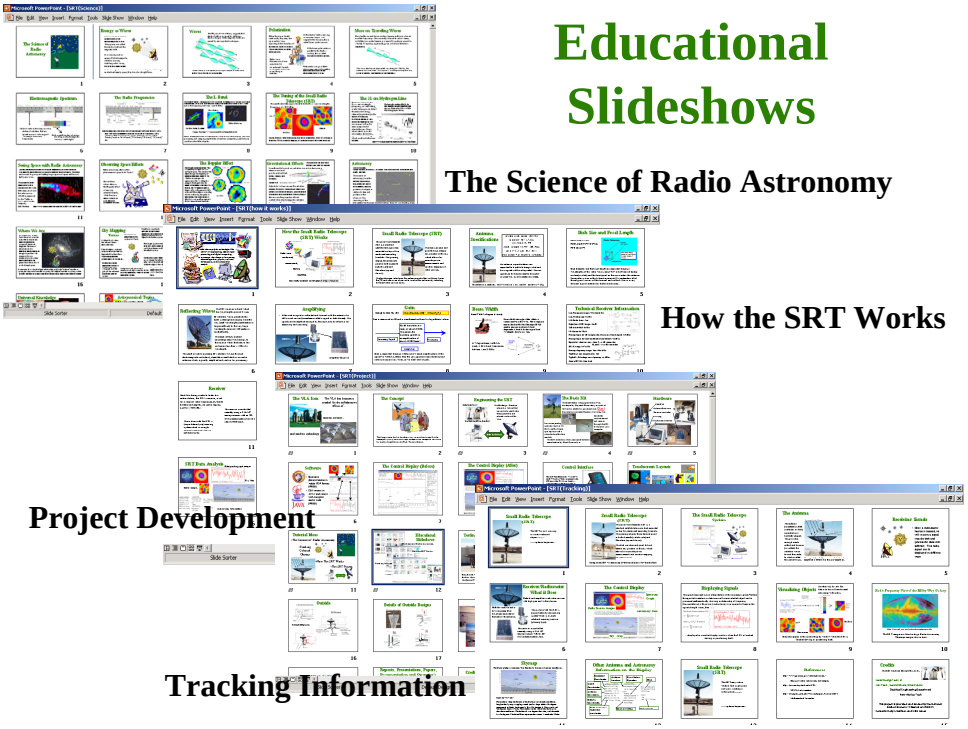
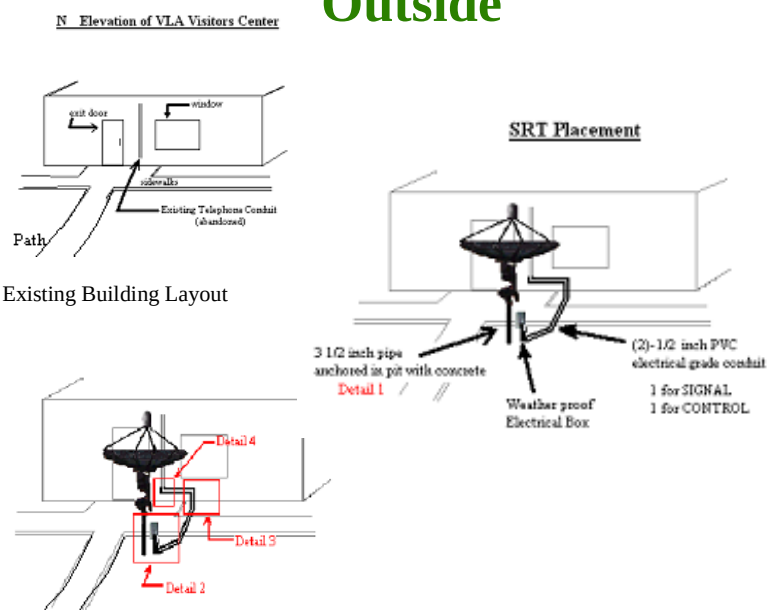


Figure 3.1. 1 Tutorial Slide shows

In addition, because of my background as an electrician (please see resume in appendix), I designed and submitted for approval the power and signal conduit runs for cables run inside of conduit when the entire project is moved for use at the VLA Visitor's building (Figures 3.1.3. and 3.1.4.). Final design approval from VLA personnel and the NRAO safety officer was needed before construction could begin.

Outside



Existing Building Layout

Figure 3.1. 2 Preliminary Installation

Details of Outside Designs

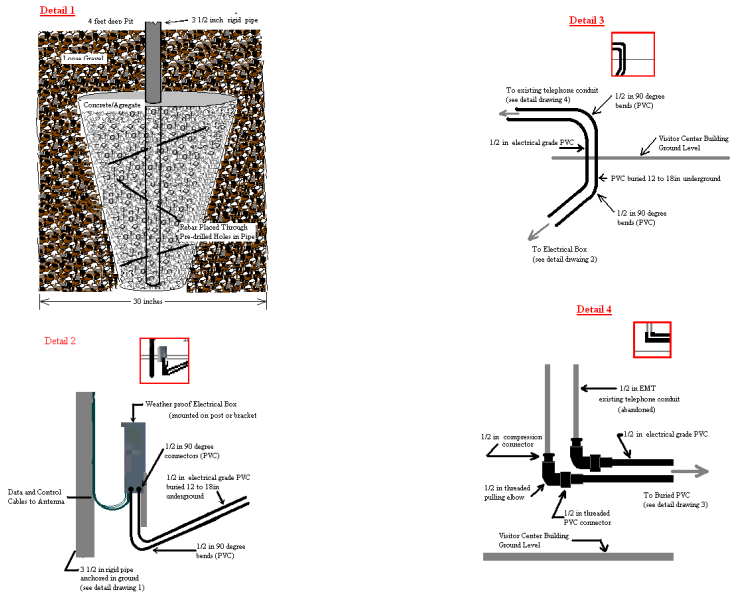


Figure 3.1. 3 Details of Installation

Coordination efforts with the key personnel for the final SRT installation had a few issues that had to be resolved. While the concept designs were generally accepted (as well as the preliminary parts lists), by the time February rolled around several e-mails were received regarding some confusion in these installation plans for the SRT. Personal contact with the lead people overseeing the different construction phases the SRT Project installation was needed "to get everyone on the same page." I had to address specification issues like location, materials, parts, installation design, grounding concerns, lightning protection, and a fence for protection. Talking personally to the Grounds Projects Coordinator, the Head Electrical Engineer and the Lead Electrician, seemed to clear up the idea of the entire project. This saved a lot of time and confusion in emails, phone calls, and inter-department memos. Next, my conceptual design for the housing cabinet for touchscreen and monitor and computer and controller was submitted for approval and revisions (Figure 3.1.5.). The Safety officer for the VLA, Jon Spargo, was contacted, to address his concerns on the SRT project. He was assured that the National Electric Code regulations for power and signal voltages were observed. For warehouse parts that might be needed at the site we had to have authorization for dispersion, so I contacted appropriate personnel for those authorizations. In the beginning of March, I came up with a physical design for implementing the lightning protection devices. I consulted with Communications installation technician Pete Martinez of the Information Services Department of NMT about lightning protection. For the end of this project, I will oversee the final installation of this lightning protection equipment that was not a part of our original statement of work.

Figure 3.1. 4 Visitor Center Kiosk Concept

3.2 Sam

My work on the SRT project focused on developing software which would control the telescope and provide a display of useful information for users. This is a detailed description of the work I accomplished between August 2001 and May 2002.

While we were in the initial stages of determining what needed to be done and how to split up the work, one of the tasks that came up was the implementation of some sort of control/data processing/data display software that would run the dish and display the information it gathered in a manner suitable for educating the public about radio astronomy. There were a number of options available, including software such as TheSky by Software Bisque. TheSky would have provided a very nice graphical user interface (GUI) but would have taken work to interface with the SRT. We decided early on to use the software that MIT ships with SRT.

This software is shipped in the form of Java code which must be compiled by the user to run the telescope. We had a number of good reasons to use this software. First, it would save us the work of figuring out how to calculate azimuth and elevation coordinates of sources, how to point the telescope, and how to process information coming in from the receiver. In addition, the fact that the software was open-source made it very easy to modify to suit NRAO's needs. Thus, the decision was made to use SRT's existing control software. I volunteered to do the programming because I had the most programming experience in the group, having written much of the code for my junior design robot. Still, I had not done any Java programming or worked with a program of this magnitude before.

The SRT control software consists of about 5400 lines of code divided into 12 classes. Their names and functions are given in Table 3.2.1.

Class Name	Primary Function
Cat	parse srt catalog file
checkey	watch mouse and keyboard
disp	define functions for graphics and text
geom	compute az/el pointing coordinates
global	store and provide access to global variables
map	contour 25 point map
outfile	write output file
plots	plot skymap, spectra, and power vs time displays
procs	coordinate procedures in other classes
sport	communicate with controller via serial port
Srt	main class: initialize software and telescope
time	calculate time in all the various formats SRT uses

Table 3.2.1: SRT Classes and Their Functions

When the SRT program starts, it begins in the class named srt. This class initializes all variables and runs the telescope to its mechanical stops so that it has a base position (referred to by the software as the stow point). This is necessary because the azimuth and elevation drives do not have absolute encoders. Every time the software is shut down the computer forgets where the dish is pointed. After finding the stow position, it starts an endless while loop which hands over control of the dish to other classes. At this point, the user is free to issue commands using a combination of control buttons at the top of the screen and a text input box at the bottom. Sources are selected by clicking on them on the skymap. Figure 3.2.1 shows the SRT control display in its original configuration.

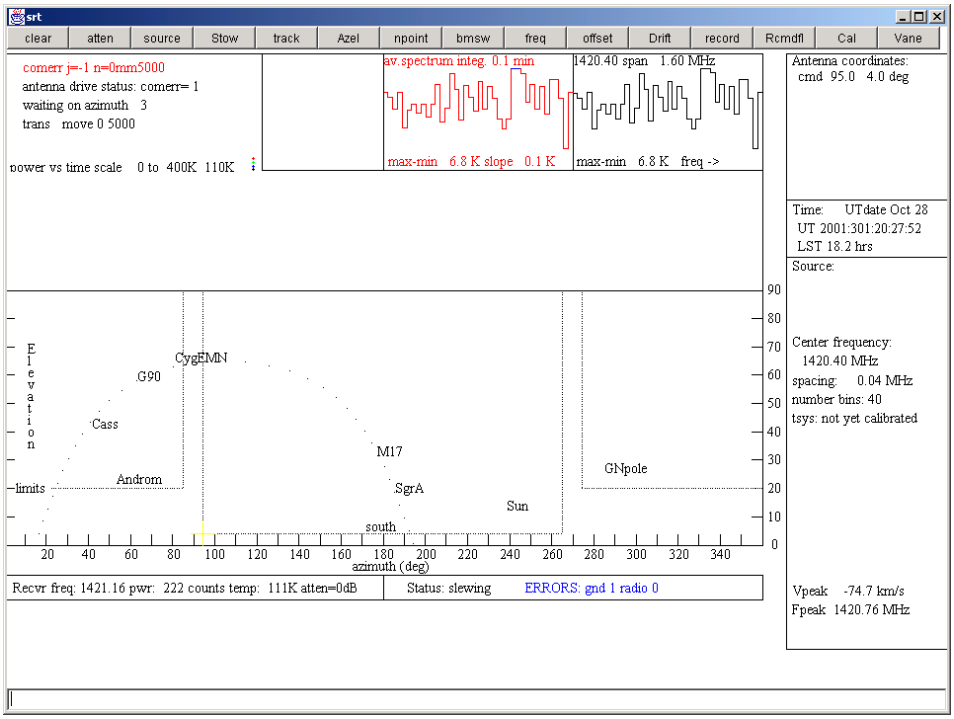


Figure 3.2.2 Original SRT Control Panel

There were a number of changes that had to be made to the software. First and foremost, the display as it arrives is fairly confusing for people with a technical background, let alone people with little or no knowledge about astronomy. I had to read through the manual a few times to understand how to use the display to control the SRT. It is unreasonable to expect visitors to read 15 pages of instructions before using the SRT, so the display had to be changed. Also, NRAO specified a "bulletproof" system which would be safe to leave in the hands of unsupervised visitors. This ruled out leaving the mouse and keyboard accessible to the public. Input would be accomplished via a touchscreen, so the software had to be modified to allow this.

Before I could modify the software, I had to learn how it worked. The code had minimal documentation, something on the order of one useful comment every few hundred lines, and many of the variables did not have names reflecting their use. I spent over a week reading through the program and marking the parts I could recognize before modifying any of the code.

My first modifications were to the display. The display is drawn by the class procs, which calls methods defined in plots. Plots contains methods for drawing the skymap, the spectrum plots, and the power vs time graph. I started by temporarily removing all of the display elements. This proved to be a time consuming task, as numerous classes contribute to the display. Tracking down all of the drawn elements and commenting them out of the code took many hours.

Starting with a clean slate, I began work on the spectrum plots, specifically working to make them movable and scalable to fit our needs. This work began in October 2001 and continued into March 2002. The SRT program generates two spectrum plots, as shown in Figure 3.3.1. The one on the right is redrawn with every sweep of the receiver across the observing band. The one on the left is the average over time of the individual sweeps. We chose to include the integrating plot in the final display, for the reason that it is less susceptible to noise and provides a cleaner spectrum. To move the plot around on the screen, the programmer has only to specify the upper left and lower right corners of the plot. Either of the plots can be turned on or off using a switch in procs, which calls the method that draws the spectra.

During this time I also worked on the skymap, changing it in much the same way as the spectra plots. The skymap is now easy to move and resize by setting the upper left and lower right corners in three classes. The outlines and mechanical limit lines are drawn in plots, the sources are drawn in geom, and sport draws the crosshairs which indicate the antenna position.

I encountered several scaling problems in altering the spectra and skymap, mostly due to the fact that SRT uses methods in many classes to draw these objects. The drawing commands pass through a number of scaling routines before actually being drawn, and I had to keep close track of where scaling happened to get the items to display properly.

The second phase of modifying the display involved deciding where to put all of the display elements, as well as making them clearer and more attractive than in the original.

With artistic direction from Gerald and Brian, I placed the skymap in the lower left corner of the display, and the spectrum in the upper right. In addition, we superimposed the skymap on a picture of the VLA landscape so that zero degrees elevation lies along the horizon. To further clarify the function of the skymap, I wrote an explanation which is to the right of the image. A final touch was to replace the crosshairs SRT uses to denote the active source with a small transparent background GIF image. When the sun is being tracked, an image of the sun appears on the skymap. For other sources, an image of a star marks the source in the sky. I also wrote an explanation of the spectrum, and placed information on parameters such as source coordinates, telescope coordinates, receiver frequency, and time on the display. In addition, there are two pictures of the active source in the upper left corner of the display. One is produced using SRT's mapping function, and the other is produced by the VLA. The map function on SRT is very impressive, and we would have liked to have made it available to the public, but it takes a very long time, so the image shown on the display is from a previously produced file. The main purpose of these two images is to show visitors the quality of the work taking place at the VLA.

Figure 3.2.2 is a screenshot of the new display panel as it would look tracking the sun. It shows all of the elements mentioned above.

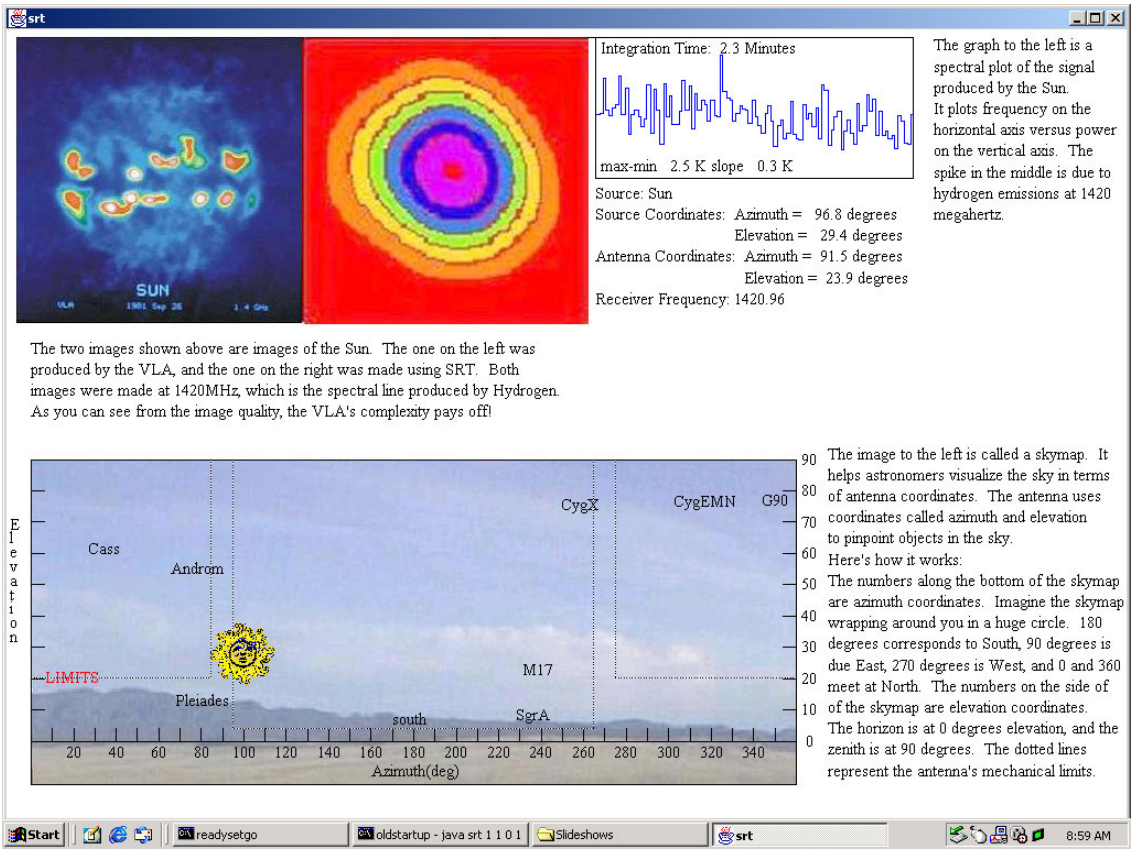


Figure 3.2.2 Modified SRT Display Panel

My other primary modification to the software involved controlling it. Without a mouse and keyboard, SRT would be useless, so the software has to interface with the touchscreen. There are a number of ways of doing this. The best would be to implement a driver which would interface the program directly with the touchscreen. Brian and I investigated this possibility by speaking to Pat Van Buskirk, one of NRAO's programmers. She referred us to another programmer who did not have time to help us. We looked in a book and also checked a few internet references and decided against trying to write a driver for the touchscreen, mainly due to time constraints. Also, the touchscreen that Brian had selected came equipped with some powerful software which lets it emulate sequences of keystrokes, mouse movements, and other functions. He had set it up to run the software through a DOS prompt for testing, so we decided to continue along that route.

As it arrived, the SRT software uses three command line arguments. These serve as switches for az/el drive simulation, receiver simulation, and maintenance mode. To keep things as simple as possible, we decided to simply add another command line argument which would specify the source. This works with the SRT catalog file so that the number entered in the fourth argument corresponds to the order of the desired source in the catalog file. This proved to be a quick way to select sources using the touchscreen. The rest of the necessary commands are hard coded. When the touchscreen starts the program, it passes the desired source through the fourth command line argument. The software takes note of this and goes about its routine of moving the dish to the stow position. I had to add a mechanism to wait while the dish moves, due to the fact that the software moves much faster than the dish. The original software depends on the user waiting until the dish stows before making commands. Adding this waiting mechanism turned out to be a lengthy process, but it is in place. Once the dish is stowed, commands are issued within the program to select the previously noted source and begin tracking. There are a few other settings that occur, but this is the basic routine.

While I would by no means claim ownership of the new version of the SRT software, there are some new functions that I implemented that did not exist in the old version. The routine that waits until the dish stows before issuing commands did not exist. It was also fairly difficult to implement due to the lack of information on dish status available within the program. This is because the two Basic Stamp microcontrollers in the SRT control module handle the most basic drive tasks. Interfacing the program with the touchscreen is another accomplishment I made. While very simple, this ability was also not present in the original software. A third contribution I made to the program is a method for drawing images. While it sounds simple, I had never used images in a program before, and I had to figure out how to do it. I found Google's usenet archives invaluable for this, along with the many other aspects of working with this program that I did not understand at the start of the project. Also, I did my best to add comments to the code, hopefully clearing it up a bit for future work. Commenting (and coding in general) became much easier one night in February when Dr. Stephen Bruder walked into the lab and asked why I was using Wordpad. He suggested downloading a Java Integrated Development Environment

(IDE). I downloaded and installed the RealJ IDE that night and life immediately became much easier. I did not use many of its features, but the color coding in RealJ helped immensely. My version of the code can be found on the accompanying CD.

In summary, I would describe my work as reshaping the visible parts of the program for NRAO, rather than rebuilding it in any way. The things I would certainly not claim in the program are all of the major program functions such as calculating source coordinates, calculating dish coordinates, low level dish control, receiver control, data processing, and time calculations. I also did not do any work on the serial interface the program uses.

While the program consumed the bulk of my time on this project, I had other tasks. One of these was figuring out how to update the computer clock automatically and accurately, as well as determining the dish's physical location. This is necessary because of the manner in which the software calculates source coordinates. The coordinates are specified in the catalog file in right ascension/declination form, and to transform these into SRT's native azimuth/elevation coordinates, accurate knowledge of current time is absolutely necessary. In addition, it is necessary to specify the coordinates of the telescope's location on earth. With knowledge of its position, both physically and temporally, the telescope can figure out where to point. Physical location would be easy to determine. A GPS reading at the dish location will be more than accurate enough to point the dish.

Deciding how to update the clock required much more work. One option was a radio receiver which would plug into the computer's serial port and update the clock using the signal broadcast by the National Institute of Standards and Technology (NIST) in Fort Collins, Colorado. I ruled this out due to the fact that we could not test it. The receiver was specified for use in wood frame buildings, and it seemed highly unlikely that it would work in our lab in Workman. Another option was software which would log in to one of NIST's time servers over the internet and update the clock. This was attractive because it would be cheap and would require no additional hardware. NIST distributes a

very nice program for free which seemed to suit our needs perfectly. It could be set to update the clock as often as once per hour, more than enough to keep SRT pointed accurately. This did not work, however, due to the EE Department's firewall. I explored many options, downloading and testing numerous freeware and shareware internet time update programs. None of them worked through the firewall. I checked with Chris Pauli and found that changing the firewall settings was not an option. One thing I noticed about the NIST software was that it used TCP port 13, with no provision for changing this setting. I downloaded the source code for the program and changed it to use port 80, but found that this would not work. The time server the program accesses only listens to port 13. As a last resort I talked to James Robnett, one of NRAO's computer experts. He told me that NRAO maintains two time servers on its network for the specific purpose of updating PC and workstation clocks. He said that connecting our PC, which runs Windows 2000, would be no trouble at all once it was on the NRAO network. The problem was solved, but I spent about two weeks on it when all it took was a five minute conversation. This was quite a learning experience.

3.3 Brian

My first task was to set up the NRAO computer. The computer did not have a java compiler so I downloaded one on the internet. A java compiler is required for the SRT software. In order to compile a program the instruction "set CLASSPATH =" must be typed in the directory with the file before the program can be compiled. I tried to add the directory with the java code to the CLASSPATH variable but this did not work. Once the computer was set up I installed Microsoft Office with PowerPoint. We used PowerPoint to make educational slideshows for the public. Our customer wanted a touchscreen for the user interface. I found a touchscreen supplier in Albuquerque called MassWorks. They sell a small color LCD touchscreen that connects to the computer via a USB port connection. The layout software that comes with the touchscreen can be integrated with any computer application. At this time I have set up the touchscreen to run the VLA

weather software, the VLA web pages, the educational slideshows, and the tracking software with the PowerPoint presentation.

I set up three main layouts because some objects are not visible at certain times of the year (figure 3.3.1). The layout on figure 3.3.1a has the Sun, Cygnus A, and Cassiopeia available for tracking. This layout will be used from November through May. The layout on figure 3.3.1b has only the Sun and Cassiopeia available. Cygnus A is not on this layout because it is only visible in the northern hemisphere from June through October. The layout on figure 3.3.1c will not have any objects available to track. This layout will be used if the SRT must be inactive due to weather conditions or technical difficulties. People will only have access to the slideshows, web links, and VLA weather. A technician will have to change layouts at the appropriate time.

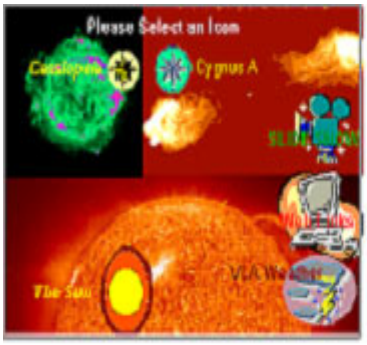


Figure 3.3.1a

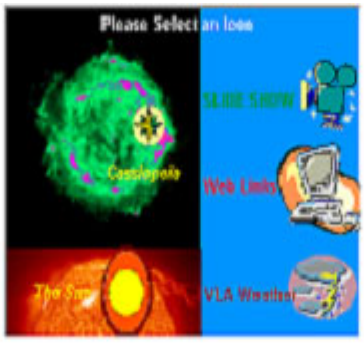


Figure 3.3.1b

Figure 3.3.1c

Figure 3.3.1 The Main Layouts

There are three slideshow presentations. One slideshow will run when the Sun, Cygnus A, or Cassiopeia is selected on the touchscreen. The SRT takes about five minutes to track a celestial object. This slideshow is set up to keep people interested during this tracking down time. To run two programs at the same time two dos prompts must be open. The mouse pointer is first positioned on a DOS prompts to run the java program. Then mouse pointer is then positioned on the other DOS prompt to run the PowerPoint presentation. The mouse pointer is positioned using the touchscreen software. Once the PowerPoint presentation is finished the java software will run until a user selects the exit

button on figure 3.3.2. The exit button will exit the java program and load the main layout.

Figure 3.3.2 The Exit Layout

The other two slideshows are accessed when a user selects slide show on one of the main layouts. Both "How the SRT works" and "The Science of Radio Astronomy" buttons on figure 3.3.3 will run the PowerPoint presentation and load the layout on figure 3.3.4a. The back to main button will sent the user back to the main layout. The layout on figure 3.3.4a will allow the user to automatically go through the presentation or go to the next slide. The automatic button on figure 3.3.4a goes through the selected slideshow at a timed interval. The next slide button on figure 3.3.4a will change slides and load the layout on figure 3.3.4b. This layout will not have the automatic feature. The user will only have access to the next slide button. At any time when a slideshow is selected the user will have the ability to exit the slideshow. The exit buttons on figure 3.3.4a and figure 3.3.4b will exit the slideshow and send the user back to the layout on figure 3.3.3. The exit button must be selected to go back to the layout on figure 3.3.3 if next slide is selected. The automatic button will load the layout on figure 3.3.3 after the slideshow is completed.

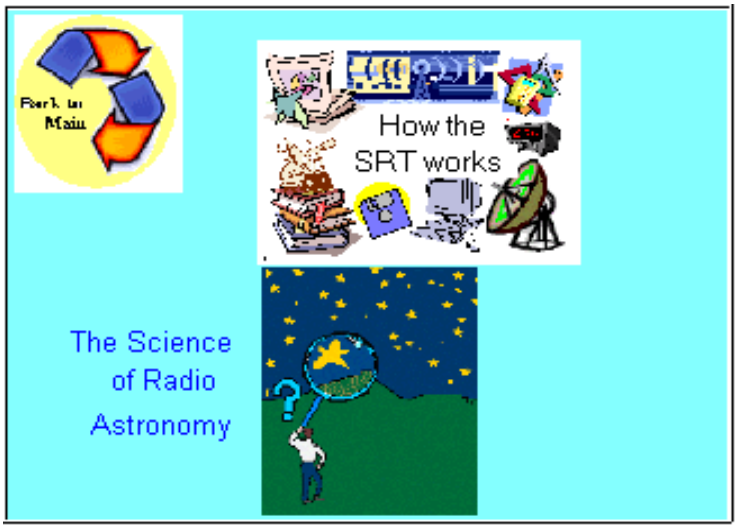


Figure 3.3.3 Slideshow Presentation Layout

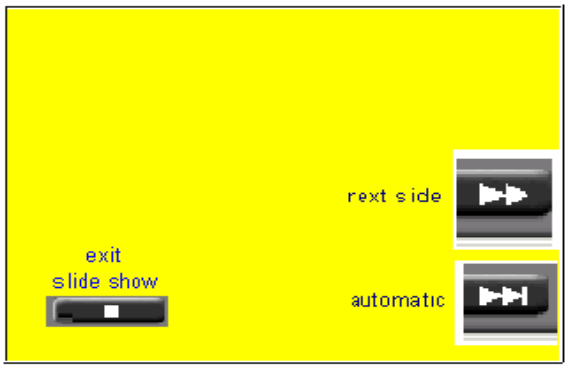


Figure 3.3.4a

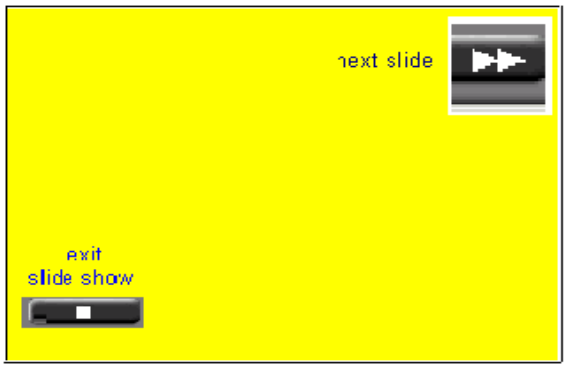


Figure 3.3.4b

The VLA weather button will load a NRAO weather program to display on the monitor and the layout on figure 3.3.5. The weather program will tell people what the current weather conditions are at the VLA site. The back to main button will exit the weather program and send the user back to the main layout. The VLA weather program looks like the layout on the touchscreen. Data is received from the VLA control center via the network.

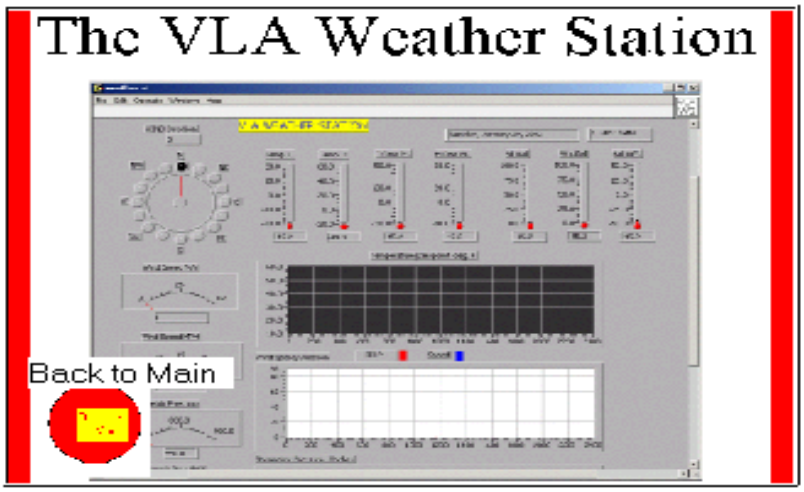


Figure 3.3.5 The VLA Weather Station Layout

When the web links icons are selected on Figure 3.3.3 the layout will change layouts to figure 3.3.6. The user will have access to three NRAO web pages. The FAQ web page has answers to common questions that people have about the VLA. The Careers at NRAO web page has the current job openings at the NRAO. The Radio Astronomy Images web page displays celestial images that the VLA produced. The back to main button will exit the opened web page and send the user back to the main layout. The page up and page down buttons will allow the user to scroll up and down the web pages.

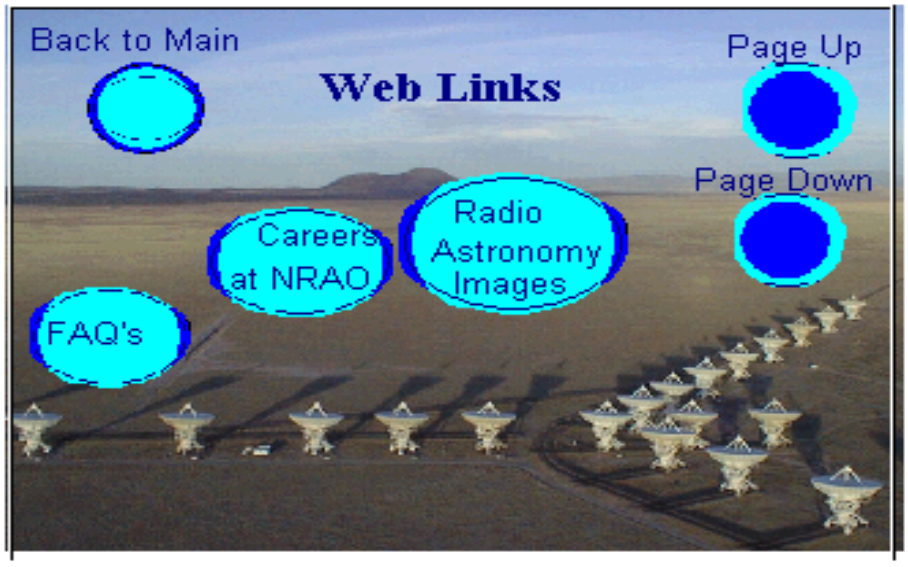


Figure 3.3.6 The VLA Web Links Layout

This feature was necessary because the web pages could not fit on one screen. The user will only have access to these three web pages. The web pages are accessed on the computer as a .htm extension file. This will increase the reliability because it does not depend on the network.

Each main layout had its own identical set of corresponding sub-layouts. This was necessary to link the layouts together because the sharing of layouts would create errors when trying to control the programs.

One problem with the touchscreen was when a button was pushed which had a command to change layouts. If the button on the second layout were in the same place as the button on the first layout, the button on the second layout would also be selected. This led to the problem of choosing an option that was unwanted. I tried to fix this problem by adding a delay after each layout. I thought a delay would give the touchscreen time to change before the selection becomes available, but this did not work so I was forced to have buttons in different locations when a layout changed. I think this is a flaw with the touchscreen software.

3.4 Group Efforts

In addition to weekly meetings in which we met with our clients and faculty advisor for updates, there were some tasks that we performed as a group.
The first task we accomplished as a group was finding a place to mount the SRT antenna for testing and development. We went through some brainstorming of ideas with our project advisor and supervisor on topics such as permissions for NRAO to have the antenna on Workman or another place on Tech property, liabilities, other design planning, other politics and even other possible sites like Etscorn Observatory and Gerald's house. We needed to find out if we would need administrative permission (Brown Hall), or if we would be infringing on any positions of Union workers from the

P-Plant or other government workers and whether we would need work-orders to complete our installation.

Because there are stanchions on the roof, which are designed to securely mount rooftop experiments such as weather sensors and antenna hardware, the idea of roof mounting the SRT on top of the Workman Building was appealing. Other reasons it was an appealing location was its accessibility and ability to be secured. Another factor that made Workman attractive was that our faculty advisor, Dr. Shanechi, graciously agreed to let us set up the development computer in his lab on the second floor. This saved us time, as we did not have to run the cables all the way down to the senior design lab on the first floor. We finally obtained permission to mount the SRT on the roof of Workman from the Physics Department chair, Dr. Minschwaner, after Clint Janes spoke with him. We met with the university architect, Dan Jones, to verify that the mounts on Workman were suitable for this project.

In early November, when the SRT kit finally arrived, our team got together to assemble and mount it on the roof of Workman. When we received the electronics and cable assemblies a short time later, we again got together to run them from the roof to the second floor lab. We all became familiarized with the antenna control program and the educational slide show demonstration. We also obtained a closed circuit television camera and got together to run the cable, again from the roof to the lab. The camera served two purposes. First, it provided us with visual feedback as to the dish's motion. Also, it provided a measure of safety by letting us know when people were on the roof or near the antenna.

In March, we took a trip to the VLA and saw the completed antenna mount and the installation of the conduit runs for grounding, control and signal wires along with a spare conduit that might be used for power. Communication with the carpenters about the design of the Monitor cabinet led to the conclusion that the cabinet would have to come from somewhere else. A decision was agreed upon as to the size of the fence that would enclose the SRT for safety considerations and to protect it from abuse. This information

was passed on to appropriate personnel. Then, tentative final delivery and installation dates were scheduled.

After development of the antenna control programs and educational tutorials, the team got together to deliver the kit to the VLA site in April. Together the team got the antenna and mounts taken apart, the cables pulled out, and everything loaded into the NRAO's panel-truck. We had a backup plan to store everything in the warehouse in case the installation went awry. At the site, we reassembled the kit and mounted it on the provided post in the enclosure built by VLA site personnel (Figure 3.4.1). Contact was made with the electricians to borrow a wire pulling-fish, ladder and some other tools. We managed to pull the signal wire into the installed conduit between the enclosure and the Visitor Center building and assemble the antenna without any problems. When we were trying to pull in the control cable, it got stuck because of too many 90 degree bends that were put into the conduit run. The VLA electricians were called to expedite cutting the conduit in the middle and to install an extra pulling elbow. The control cable was finally pulled through to the junction box at the base of the dish. The cables were pulled through the outer wall of the Visitor Center, but the junction box there was not completed. In addition there was a Coke vending machine in the way.

Figure 3.4.1 Reassembling the SRT at the VLA Site

Once the installation was complete, we connected the development computer and controller to verify that the telescope still worked. We were able to control the SRT with no difficulties, but since the kiosk wasn't ready we brought the computer back with us. Appropriate VLA building and maintenance personnel were asked to move the cabinets and displays that might be in the way of the SRT's kiosk. The Coke machine authorities would also have to be notified. We have since arranged for a couple more needed parts like a multi-strip outlet and grounding lugs. When the kiosk arrives, the space needed for it will be clear and we can schedule final installation and testing. We have also made arrangements to pass on documentation of the project to Phil Dooley, a VLA engineer.

4 Final State of Project

Figure 4.1 SRT at Its Final Location at the VLA

4.1 Final Testing

Initial development and testing was an on-going process. With the touch screen we tested operation and functionality every time we made a change or added a new feature. Proceeding in this manner we could ensure the system worked both in simulation runs and actual tracking of selected objects. With the final version of the software we have tested the system and it has worked every time. The system was tested on the site using the original software. This test worked and verified that the wiring was re-installed properly. From this we, have every reason to believe that it will work with our developed software.

We plan to meet with Phil Dooley an engineer at the VLA May 9th, after this paper is due. The purpose of this meeting will be to pass on responsibility for the project and give Mr. Dooley the knowledge he will need to set up the system when the last of the needed parts are assembled and put in place.

4.2 Functionality

Using a touchscreen a person can select to operate the SRT to track celestial objects. It will take approximately five minutes to initialize the system to start tracking. During this time an educational slide show will run explaining what the viewer is about to watch. There are also tutorials on The Science of Radio Astronomy and How the SRT Works. Additional features include NRAO Web pages and The VLA Weather Station, a LabVIEW program used by VLA astronomers.

5 Summary and Recommendations

This section describes effect of our work on this project. In addition, we have outlined some possibilities for further improvement of the SRT and some recommendations for maintaining it.

5.1 Summary of Current Work

Our work has enable NRAO to provide a sophisticated instrument that the general public can access and use to learn about radio astronomy. The SRT kit designed by MIT is suitable for astronomy students or scientists. We have adapted it to be suitable for people who do not necessarily have any technical knowledge.

Some other possibilities for applications of our work include using our simplified interface and program to benefit science hub schools. In addition, the idea of making CD's of the slide shows available to generate revenue has been raised.

5.2 Recommendations For Future Work

A nice addition to this project would be to have the java program stay open all the time to keep the SRT running and still have a choice of selecting other educational demonstrations. This will cut down on the time it takes to track an object. As it is, the SRT must find its motor limits every time the java program starts up. Furthermore, the java program must shut down to change celestial objects because the objects are selected through the command line. The program must be restarted every time the user wants to select a different object. Further work with the touch screen interface and the program's command structure would eliminate this requirement.

Automatic features in software could be used to stow the dish (pointed up at the zenith) when inactive or under severe weather conditions. If we had more time we would like to incorporate the VLA weather station program to flag on a high wind indicator and the

SRT program would cue make the antenna automatically stow and load the touchscreen layout which will inform the public that it is inoperable.

Another idea would be to open two programs at the same time with one dos prompt. Currently when a celestial object is selected two dos prompts are needed because two programs are used. One prompt will run the tracking software and the other will run the PowerPoint presentations. At the present time, mouse positioning is used to change dos prompts. This is not the best way to do this because over time the mouse can get out of position. A Visual C++ program could be created which will open both programs using one prompt. This will eliminate the need to position the mouse creating a cleaner running program. A copy of Visual C++ would need to be installed on the computer and we would need to learn how to use it to accomplish this task.

More radio sources could be made available for selection including terrestrial sources and geosynchronous satellites. This would be a simple addition to the program, requiring minimal changes to code.

An interesting Senior design project would be to upgrade the SRT controller. Possible improvements would be to add a better microcontroller and a DSP chip to perform signal processing that the SRT program is not capable of.

More information about the VLA could be made available in the SRT display like incorporating VLA observation schedules. The SRT project slideshow could also be made to run along with the educational slide shows.

Another possible addition is having a Web cam available to allow access from the web. Its website could be configured to control the SRT from a remote location.

Our project has inspired the NMT Physics Department to begin work on an array of multiple SRT antennas. This project had been planned previously, but they needed incentive to initiate production.

A printer could be made available for screen shots of the control display (easy to implement) to generate revenue for the VLA Visitor Center from it.

5.3 Recommendations For Maintaining the System

In providing for some minimal maintenance a technician should grease it at the provided fittings, check bolts for tightness, and verify alignment of vertical axis chassi. A technician should run the SRT program once upon startup to verify operation. A technician should Change layouts in November and June when objects to be selected are not visible (please see Touchscreen Operator's Manual in appendix).

Appendix:

Touchscreen Operators Manual

1. Left click on the start icon
2. Select programs/Massworks/ID-75 Layout Software to start the layout program
3. To open a layout , click on file and open>>Program Files/MassWorks ID-75/Layouts. In the Layouts folder there are three folders that the SRT uses.

   stow – Used when the SRT is inactive.

   cass,sun&cygnus – Used when all three sources are visible (Cass, Sun and Cygnus). Used from June through October.

   cass&sun – Used when only two sources are visible (Cass and Sun). Used from November through May.

Each of these folders has three main layouts.

(stow) – The main layout is stow.lay. Open this by choosing file and open. Follow steps 8 and 9. Once the main layout has been set all other corresponding layouts will load automatically.

(cass,sun&cygnus) – The main layout is cass2.lay. Open this by choosing file and open. Follow steps 8 and 9. Once the main layout has been set all other corresponding layouts will load automatically.

(cass &sun) – The main layout is cass2noCygnus.lay. Open this by choosing file and open. Follow steps 8 and 9. Once the main layout has been set all other corresponding layouts will load automatically.

4. Once a layout is open a buttons is created by right mouse clicking on an empty space on the layout and selecting insert and choosing Macro. A "macro" can be a list of keyboard or mouse actions or events to perform when the button is pressed or released. The event is received from the controller by the button in the layout. Each button can have many macros of actions to carry out.

5. A button is modified by right mouse clicking on it and selecting properties. All of the macros can be modified from this window.
6. For a more detailed method of creating or editing buttons select the help menu.
7. Text can be added to the layout by selecting insert and choosing Text. This will not create more than one line of text at a time.
8. A layout is sent to the device by left mouse clicking on the yellow lightning bolt icon on the toolbar.
9. The message "The layout has been sent to the device" will appear, click ok.
10. Slideshow timing can be modified. The time between slides can be changed by opening up the corresponding layouts that have an automatic button. There are three layouts for each slideshow with an automatic button that will need to be changed if any one is changed.

    Program Files/MassWorks ID-75/Layouts/stow/stowsrt(science).lay
    Program Files/MassWorks ID-75/Layouts/stow/stowsrt(works).lay
    Program Files/MassWorks ID-75/Layouts/cass&sun/stowsrt(science).lay
    Program Files/MassWorks ID-75/Layouts/cass&sun/stowsrt(works).lay
    Program Files/MassWorks ID-75/Layouts/cass,sun&cygnus/stowsrt(science).lay
    Program Files/MassWorks ID-75/Layouts/cass,sun&cygnus/stowsrt(works).lay

    To do this count which slide needs to be modified. Right click on automatic button and select properties. The delay for each slideshow change is before the corresponding space bar command. Double click on the chosen delay to change the timing. (example: 22 seconds = 22000) Change this same line in each of the other two automatic button layouts for the slideshow.
11. Close the program so the window will not hamper the operation.
12. Do a test run by selecting the VLA weather button on the layout and then press the back to main button once the weather station is up.

For more information go to the website at http:/ /www.massworks.com. This website will contain help on using the touchscreen.

Compiling and Running a Java Program

1. Change directories to the one with the Java program that you want to compile.
2. Type “set CLASSPATH=” on the dos prompt.
3. Type javac filename.java.
4. If the program compiled with no errors filename.class will be created.
5. Run the program by typing filename.java.

Kit Specs

(Schematics located on CD)

Radiometer Characteristics

L.O. Frequency range	1370-1800 MHz
L.O. Tuning steps	40 kHz
L.O. Settle time	<5 ms
Rejection of LSB image	>20 dB
3 dB bandwidth	40 kHz
I.F. Center	40 kHz
6 dB I.F. range	10-70 kHz
Preamp frequency range	1400-1440 MHz
Typical system temperature	150K
Typical L.O. leakage out of preamp	-105dBm
Preamp input for dB compression from out of band signals	-24 dBm
Preamp input for intermodulation interference	-30 dBm
Square law detector max.	4000 K a 0 dB attenuation 40,000 K at 10 dB attenuation
Control	RS-232 2400 baud

Antenna Specifications

Manufacturer	Kaul-Tronics Inc.
Model Number	(S-7.5)
Diameter	90" (2.3m)
F/D Ratio	0.375
Focal Length	33.75" (85.7cm)
Gain @ 4.2 GHz	38.1 dBi
Gain @ 1.4 GHz	
Weight with mount	160 lbs
Beam Width	7.0 Degrees (L-band)

Gant Chart Explanations

In the following diagrams, as can be seen, the individual tasks of the overall project are placed on the left hand side while the time line (and deadlines) extends out to the right. The assignment of these tasks is distinguished by color-code (the key is in the bottom right hand corner) along with the coding for the tasks that everyone (black) will participate in. The longest tasks (paths) have been divided up according to experience and preference. The shorter paths have been assigned intuitively and thought of as challenges.

Some tasks have already been accomplished while some tasks, which were scheduled for later, were moved up and started on already. Still, there are some tasks that had to be "bumped" later because these time constraints have been relying on outside sources that have bogged down. Following the two full versions (Updated and Previous) of the time line Gant charts are listed short explanations of the individual tasks.

FIRST SEMESTER Tasks

Build and Mount SRT: Assemble the SRT kit and mount on Workman for testing

Run Cables: Pull cables through existing conduit to second floor lab

Setup Development PC: Install necessary software on PC provided by NRAO

Network Address: Connect development PC to Tech network

Java Compiler: Install Java Compiler on development PC

Simulate Dish Operation: Use SRT software to simulate tracking and data collection

Develop Public Use System: Set up interface that the public can use to control SRT and learn about radio astronomy.

Research/Order Parts: Find out about necessary components, particularly a touchscreen

Touchscreen Implementation: Set up touchscreen for public use with intuitive commands and attractive appearance

Interface Receiver Data: Process raw data coming from the receiver into a display (e.g. spectrum) that demonstrates VLA operation

Time and Location Update: Implement software or hardware to update computer clock for accurate tracking. Survey site to find exact dish location.

Implement Tracking Software: Look into using LabVIEW, TheSky, STS control system, or existing SRT software to track astronomical objects.

Calibration of SRT: Calibrate receiver to obtain accurate data

Electronic Noise: Method of calibration, more accurate than motorized vane

Motorized Vane: Alternative method of calibrating receiver

Watchdog Reset: Set up computer so that it and the display program restart without human intervention (in the event of power failure)

SECOND SEMESTER Tasks

Control Software Implementation: further development and use of existing SRT software for tracking astronomical objects and deadlines to reject other control software.

Research/order parts: Deadline for other controls software such as LabVIEW, TheSky, or STS control system.

Develop Public Use System-further development of touchscreen interface that the public can use to control SRT and learn about radio astronomy. As part of implementing the touchscreen for public use with intuitive commands and that has an attractive appearance

Research/order parts: this might be to get together with someone to design a counter to house the computer components that denies public access to the main computer except for the monitor and touchscreen

Data Processing: source incoming signals to appropriate displays

Spectrum Analysis: source incoming signals to appropriate displays

Educational Graphics: completion of educational displays

Other Displays: integration of other data files like weather or observation schedules and explanations of their relevance for public knowledge

Complete Installation: physically install the SRT at the VLA Visitors Center

Research/order parts: figure out what parts are needed for the installation

Planning/Engineering/Contacts: design the final installation layout and make contacts with appropriate personnel to get the jobs completed

Reports/Papers/Presentation Material: ongoing production of paperwork

Final Documentation: write this SRT operation manual complete with diagrams

Compile Ongoing Diary: Putt together all email and correspondence as a chronological history of the SRT project

Client E-mails

****

Date: Wed, 17 Oct 2001 14:25:37 -0600 (MDT)
From: Clint Janes <cjanes@aoc.nrao.edu>
To: brajala@nmt.edu, gbivens@nmt.edu, sfield@nmt.edu
Subject: project

My apologies for not making it to the meeting yesterday. Sam showed
me what has been going on and I'm really impressed, thanks!

Clint

PS: I'll try harder to make it next Tuesday.
Clint

****

Date: Wed, 17 Oct 2001 16:05:10 -0600 (MDT)
From: Clint Janes <cjanes@aoc.nrao.edu>
To: rharriso@zia.aoc.NRAO.EDU
Cc: brajala@nmt.edu, gbivens@nmt.edu, sfield@nmt.edu
Subject: Project

Hi Robyn:

I am so pleased at the progress so far of the student project.
I suggest having the Visitor Center Committee meet at Tech next
time so that the students can demo and get additional suggestion.
Let me know if you agree and I can set up. Tuesday afternoon is
a good time for them.

Clint

****

Date: Wed, 10 Apr 2002 17:18:29 -0600

From: Robyn Harrison <rharriso@cv3.cv.nrao.edu>

To: Clint Janes <cjanes@zia.aoc.NRAO.EDU>, Sam Field <sfield@nmt.edu>,
Gerald Bivens <gbivens@nmt.edu>, Brian <brajala@nmt.edu>,
Dr. Shanechi <shanechi@ee.nmt.edu>

Subject: SRT installation at site

Hi,
The fence went out to the site today. I think Charley plans to put it up either tomorrow, Friday, or Monday at the latest so it should be ready for you to move the dish itself out next week. We can't install the internet cable until we have an actual cabinet to put the (I forget what Fred told me goes between the end of the cable and the computer) in, but anyway, it would be great if we could go ahead and test it w/o the fiber link, though I know we need to update the clock, etc. (Let me know and we can plan a time--we can store the computer in the projector room if we need to). The cabinet builder gets back April 24. I expect we'll have the cabinet within a few days thereafter.
And Fred will probably connect the cable shortly thereafter, or the same day--I'll be sure he gets it ordered before hand.
I'm sure I've forgotten something. If you think of it, let me know.
Robyn

Date: Tue, 30 Apr 2002 11:24:40 -0600

From: Robyn Harrison <rharriso@cv3.cv.nrao.edu>

To: Clint Janes <cjanes@zia.aoc.NRAO.EDU>, Sam Field <sfield@nmt.edu>,
Gerald Bivens <gbivens@nmt.edu>, Brian <brajala@nmt.edu>,
Dr. Shanechi <shanechi@ee.nmt.edu>

Subject: kiosk

Okay. This one was my screw up. I evidently miscommunicated what I needed to the builder of the kiosk--at least as far as when we needed it. I just talked to him and he thought I was still waiting to have it approved. At any rate, he'll work on it this week and try to have it done first of next week. What do you want to do?

Robyn

--

****

Robyn Harrison
Education and Public Outreach
National Radio Astronomy Observatory
Socorro, New Mexico 87801
505-835-7243

Date: Tue, 30 Apr 2002 13:16:28 -0600 (MDT)

From: Clint Janes <cjanes@aoc.nrao.edu>

To: brajala@nmt.edu, cjanes@zia.aoc.NRAO.EDU, gbivens@nmt.edu, rharriso@cv3.cv.nrao.edu, sfield@nmt.edu, shanechi@ee.nmt.edu

Subject: Re: kiosk

The students plan to meet with Phil middle of next week. We can take it out if it is ready.

Clint

Real Time Linux On A Small Mobile Robot

A Thesis

Presented to the Faculty of

the Electrical Engineering Department

of

New Mexico Tech

By

Heather Bitsoi

Kyle Lamb

Bill Willems

In partial fulfillment of the requirements

for the course

EE-482 Senior Design Project II

May, 2002

Table of Contents

List of Figures

List of Tables

List of Abbreviations

A	Amps
A/D	Analog to Digital
AC	Alternating Current
BIOS	Basic Input/Output System
BMP	Bitmap
°C	degrees Celsius
ccw	counterclockwise
CMOS	Complementary Metal Oxide Semiconductor
COM	Communications Port
CPU	Central Processing Unit
cw	Clockwise
D/A	Digital to Analog
DC	Direct Current
DIO	Digital Input/Output
DMA	Direct Memory Access
FIFO	First-In-First-Out
I/O	Input and Output
IDE	Integrated Drive Electronics
IrDA	Infrared Data Association
IRQ	Interrupt Request
ISA	Industry Standard Architecture
ISR	Interrupt Service Routine
KB	kilobyte = 1,000 bytes
LBA	Logical Block Addressing
LED	Light Emitting Diode
mA	milliamps = 0.1 Amps
MB	MegaByte = 1,000,000 bytes
Mbps	Mega bits per second
MHz	Megahertz = 1,000,000 hertz
OS	Operating System
PC	Personal Computer
PC/104	Personal Computer Interface – 104-pin connector
PCI	Peripheral Component Interconnect
PS/2	Serial Port – 2nd Generation
PWM	Pulse Width Modulation
RAM	Random Access Memory
RGB	Red Green Blue
ROM	Read Only Memory
RT	Real Time
SBC	Single Board Computer
SDRAM	Synchronous Dynamic Random Access Memory
USB	Universal Serial Bus
V	Volts

Introduction

This project is the basis of our capstone course at New Mexico Tech. We selected this project based on our interests and our skills at the beginning of this year. The purpose of this project is to design and implement a Robot, which would use RT Linux on a Single Board Computer to bring in sensor data and respond to the environment based on the sensor data. Our client has also asked that we do some image processing using two USB cameras such as vertical and horizontal line detection. On the robot, a wide variety of sensors would be used to get a good sense of the environment and the robot could respond to these variables accordingly. We were also to use the SBC along with wireless networking in order to have feedback from the robot as it ran its different tasks.

A possible future use of the work we completed is to design robots that can collaborate efforts over a network, and possibly control the robot over the Internet. Along with the work, collaborating robots the Breakout Board that we constructed can be used in many applications possibly used as a reconfigurable data processor.

Customer Requirements

The requirements set forth by our customer were to develop a package using RT Linux that would run on the Prometheus Single Board Computer to access and respond to multiple sensors. We were also required to do some Image Processing to find vertical and horizontal lines. We were provided with an assembled Robot Kit, that the customer asked us to add some sensors to as well as add a better battery. We were also provided with a Prometheus SBC, which due to its lack of Timer/Pulse Accumulators needed an external board to interface with the motors on the robot. As far as the external board went we were told we could either design our own, or order a prefabricated board. We decided it would be better for us to design our own board, but this unfortunately ended up taking most of our time.

Background

RedHat Linux

Linux is an operating system that was initially created as a hobby by a young student, Linus Torvalds, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The current full-featured version is 2.4 (released January 2001) and development continues.

Linux is developed under the GNU General Public License and its source code is freely available to everyone. This however, does not mean that Linux and its assorted distributions are free -- companies and developers may charge money for it as long as the source code remains available. Linux may be used for a wide variety of purposes including networking, software development, and as an end-user platform. Linux is often considered an excellent, low-cost alternative to other more expensive operating systems.

Due to the very nature of Linux's functionality and availability, it has become quite popular worldwide and a vast number of software programmers have taken Linux's source code and adapted it to meet their individual needs. At this time, there are dozens of ongoing projects for porting Linux to various hardware configurations and purposes. (Linux Online 2002)

Real Time Linux

RTLinux is a hard realtime operating system that coexists with the Linux OS. With RTLinux, it is possible to create realtime POSIX.1b threads that will run at precisely specified moments of time. The basic premise underlying the design of RTLinux is that it is not feasible to identify and eliminate all aspects of kernel operation that lead to unpredictability. These sources of unpredictability include the Linux scheduling algorithm (which is optimized to maximize throughput), device drivers, uninterruptible system calls, the use of interrupt disabling and virtual memory operations. The best way to avoid these problems is to construct a small, predictable kernel separate from the Linux kernel, and to make it simple enough such that operations can be measured and shown to have predictable execution. This has been the course taken by the developers of RTLinux. This approach has the added benefit of maintainability - before the development of RTLinux, every time new device drivers or other enhancements to Linux were needed, a study would have to be performed to determine that the change would not introduce unpredictability.

Figure 1 shows the basic Linux kernel without hard realtime support. You will see that the Linux kernel separates the hardware from user-level tasks. The kernel has the ability to suspend any user-level task, once that task has outrun the “slice of time” allotted to it by the CPU. Assume, for example, that a user task controls a robotic arm. The standard Linux kernel could potentially preempt the task and give the CPU to one that is less critical (e.g. one that boots up Netscape). Consequently, the arm will not meet strict timing requirements. Thus,

in trying to be “fair” to all tasks, the kernel can prevent critical events from occurring.

Figure 1 also shows a Linux kernel modified to support hard realtime. An additional layer of abstraction - termed a “virtual machine” in the literature - has been added between the standard Linux kernel and the computer hardware. As far as the standard Linux kernel is concerned, this new layer appears to be actual hardware. More importantly, this new layer introduces its own fixed-priority scheduler. This scheduler assigns the lowest priority to the standard Linux kernel, which then runs as an independent task. Then it allows the user to both introduce and set priorities for any number of realtime tasks.

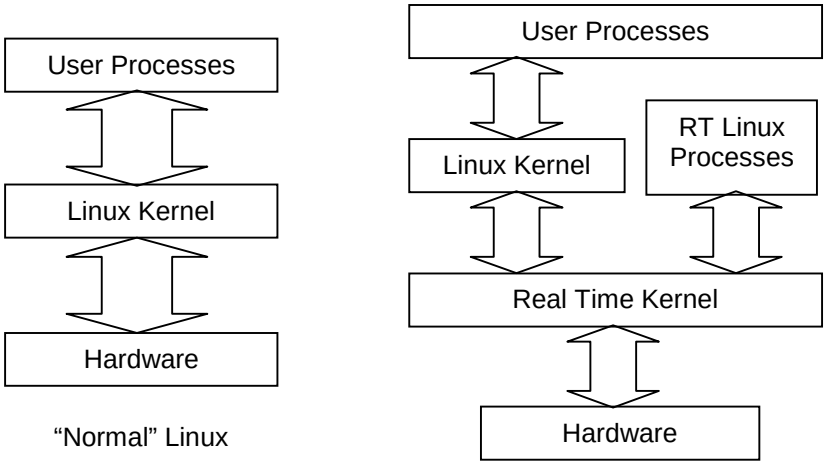


Figure 1: “Normal” Linux Vs. RT Linux

The abstraction layer introduced by RTLinux works by intercepting all hardware interrupts. Hardware interrupts not related to realtime activities are held and then passed to the Linux kernel as software interrupts when the RTLinux kernel is idle and the standard Linux kernel runs. Otherwise, the appropriate realtime ISR is run. The RTLinux executive is itself nonpreemptible. Unpredictable delays within the RTLinux executive are eliminated by its small

size and limited operations. Realtime tasks have two special attributes: they are “privileged” (that is, they have direct access to hardware), and they do not use virtual memory. Realtime tasks are written as special Linux modules that can be dynamically loaded into memory. They are not expected to execute Linux system calls. The initialization code for a realtime tasks initializes the realtime task structure and informs RTLinux of its deadline, period, and release-time constraints. Non-periodic tasks are supported via the use of interrupts.

In contrast with some other approaches to realtime, RTLinux leaves the Linux kernel essentially untouched. Via a set of relatively simple modifications, it manages to convert the existing Linux kernel into a hard realtime environment without hindering future Linux development (Barabanov 2001).

RT-FIFOs

Realtime FIFOs are Linux character devices with the major number of 150. RT-FIFOs are First-In-First-Out queues that can be read from and written to by Linux processes and RTLinux threads. FIFOs are uni-directional - you can use a pair of FIFOs for bi-directional data exchange.

Prometheus SBC

The Prometheus Single Board Computer (SBC) is an embedded PC/104 CPU that integrates three separate circuits onto a single compact board: CPU, Ethernet, and Analog I/O. The Prometheus conforms to the PC/104 standard, an embedded standard that is based on the ISA and PCI buses and provides a compact, rugged mechanical design for embedded systems. PC/104 modules feature a pin and socket connection system in place of card edge connectors, as

well as mounting holes in each corner. The result is an extremely rugged computer system fit for mobile and miniature applications. PC/104 modules stack together with 0.6” spacing between boards (0.662” pitch including the thickness of the PCB).

Prometheus uses the PCI bus internally to connect the ethernet circuit to the processor. It uses the ISA bus internally to connect serial ports 3 and 4, as well as the data acquisition circuit, to the processor. Only the ISA bus is brought out to expansion connectors for the connection of add-on boards (Diamond Systems 2002). The features found on the Prometheus SBC are outlined in Appendix E.

Breakout Board

The Prometheus SBC has only two timer / pulse accumulators. One of these timers is already dedicated to the onboard I/O module – the other, due to a flaw in the programming of the controller chip, is also unavailable. In order to control a single motor in a closed loop, we need one timer to generate a PWM signal to control speed; and one accumulator to read the encoder data generated by the motor. We are using two motors creating the need for a minimum of four timer / pulse accumulators. Due to the lack of timer / pulse accumulators provided by the Prometheus, it was clear that we would need to either buy a third party timer / pulse accumulator board, or build our own.

Each choice has advantages as well as disadvantages: A third party board means we do not have to spend the time to design our own. However, designing our own board allows us to tailor it to suit our specific needs as well as the ability

to easily change such features should those needs change. After much deliberation, it was decided that we would design and build our own timer / pulse accumulator board. The details of this design will be discussed later.

The Breakout Board makes use of the PC/104 interface bus to communicate with the Prometheus. This interface is electrically identical to the ISA interface found on the common PC. The key differences between these two architectures are some additional ground pins in the PC/104 interface for the added ruggedness inherent to embedded systems design and a different form factor. A brief overview of the PC/104 bus and our interface to that bus are outlined in Appendix A.

Image Processing

Image Processing is the convergence of image data and digital signal processing. The concept and procedures do not come first hand to everyone. Research was conducted using the Internet, books, simple code examples and finally peer communication. There were several resources for Image Processing encountered; a book provided by the client was the primary resource. The different types of methods used to manipulate images are summarized in Table 1 below.

Operation	What it could be used for?	How it can be applied to this project?
Mutiplication	Used to brighten an image	If the image from camera is too dark
Division	Used to darken an image	If the image from camera has too much light
Subtraction	Used to determine whether there is a change in two image.	Multi-frame line tracking with some additional manipulation
Addition	Image morphing or adding noise	Not applicable at this time
Bit Operations		
AND	Crop and image and get the part you are needing	These operations were not incorporated in the final Image Processing algorithm. It could be of later use in Advanced Image Processing.
OR	Get rid of the additional image attached after AND and getting the cropped image alone	
NOT	Change the image to a negative image	

Table 1 – Useful Image Processing Operations

The Image Processing is highly dependent upon the convolution process. The convolution process functions with the use of 3x3 matrices. These matrices vary significantly and depend primarily on the application needs. In the Image Processing, the matrix used was the Sobel operator. There are many different operators (i.e. Prewitt operator, Kirsch, Robinson, Laplacian operators, Frei-Chen, etc.) used in image processing and the more complicated it gets to implement in C programming. The easiest to implement was the Sobel operator, which successfully executed the conversion of images and produced satisfactory output. The Roberts operator was used to find the difference in the color of the pixels. (Umbaugh 1998)

The code for the Image Processing was written in C. The program depended highly on structures. They were easy to access and cut the time to process each image down. The members of the project knew a significant amount of C programming which made it an easier way to implement with.

Discussion

Image Processing – Heather Bitsoi

In the image processing part of the project, our primary objective is to take an image from a basic web camera, store this image, and apply some edge finding algorithms. There are different ways to accomplish this task. The most appropriate and efficient processing is based upon the application needs. The need for image processing is to be able to find a line in the surrounding environment and perform wall following using cameras instead of the IR distance sensors. There are many more applications, which the customer wishes to explore, which are unfortunately beyond the scope of this project. We have some limitations to the image processing procedure, one of which is the limited processing power of the Prometheus SBC, which is the source of the surrounding information. Another consideration we had to acknowledge was the placement of the cameras. There were two cameras provided by the customer. The positions of the cameras were going to allow us to determine the distance to objects via depth perception. The results from the image processing would be used to enhance and create a more accurate perception of the robot's surroundings.

Image Processing Algorithm – Heather Bitsoi

The Image Processing algorithm consists of five different parts. The first part is reading a file that is saved to memory on the SBC. This file is a 24-bit bitmap (BMP) image. The image is determined to be a bitmap image if the file extension is ".bmp". The image is retrieved from the IBM web camera, which is mounted to

the robot. The retrieval of this image takes, on average, 4.5 seconds per image. The format of the image is as follows:

- a bitmap-file header: this contains information about the type, size, and layout of a device-independent bitmap file.
- a bitmap-information header which specifies the dimensions, compression type, and color format for the bitmap.
- The color table is not present for bitmaps with 24 color bits because each pixel is represented by 24-bit red-green-blue (RGB) values in the actual bitmap data area.
- an array of bytes that defines the bitmap bits. These are the actual image data, represented by consecutive rows, or "scan lines," of the bitmap. Each scan line consists of consecutive bytes representing the pixels in the scan line, in left-to-right order.

All BMP files contain RGB data. In our case we have:

- 24-bit: 16777216 colors, mixes 256 tints of Red with 256 tints of Green and Blue

The set of the first 55 positions in the image is the image header. First, we extract the image width, followed by the height, the depth, the compression, and finally the compressed size. After reading all this information into a buffer (the buffer is a character array that is allocated to store the bitmap information and retrieve it for later use). The next section of the bitmap is the image data. The image data is put into another buffer. This information would be altered and

manipulated. The original data is untouched and remains the same for comparison.

Now that the information in the BMP is stored, the next procedure is to use the convolution process to highlight the horizontal and vertical lines in the image. The convolution process constitutes of two processes, one is defining the horizontal lines and the other is defining the vertical lines. Each process uses a different mask (the mask is a 3x3 matrix with the values that make the lines more prominent). The mask for horizontal lines is called the row mask and for vertical lines, it is called the column mask. The masks used for the convolution process are depicted in Figures 2 and 3. These matrices are also known as the Sobel convolution mask.

3	2	3
0	0	0
-3	-2	-3

Figure 2 - Row Mask

3	0	-3
2	0	-2
3	0	-3

Figure 3 - Column Mask

The convolution process takes each of the mask and is virtually placed over the shaded area depicted in Figure 4 (actual image data). The data in the image is the pixel information. The mask values [3,2,3,0,0,0, -3, -2, -3] are multiplied by the pixel values of the corresponding value when the mask is in placed, after the multiplication the results are summed together and the results of this operation is stored in X (Figure 5). The mask is moved to the right by one pixel. The same operation is performed on the corresponding pixel values. The buffer holds the result of the convolution process. This process continues until the mask covers the entire image and the mask is at the lower right corner.

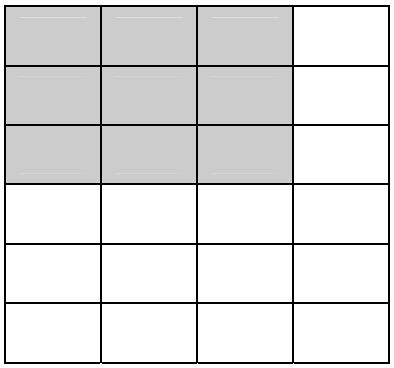


Figure 4 – Image Data

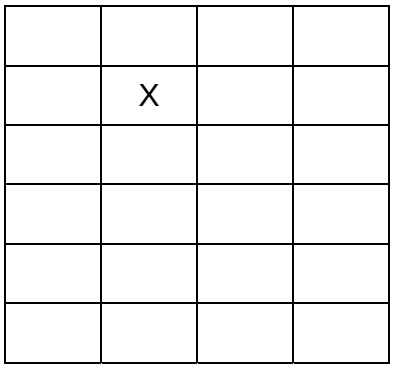


Figure 5 – Buffer

The following pictures (in Figure 6) are the actual results from the convolution process.

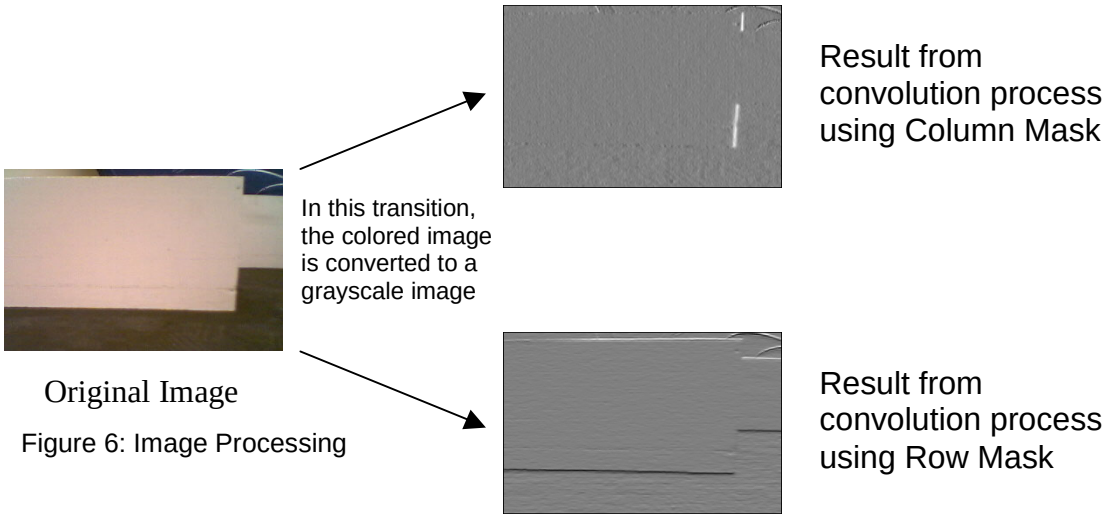


Figure 6: Image Processing

In each of the resulting images, the information it contains is more then we need for our application. To extract the unnecessary information and retain the useful information we apply an edge point equation. The equation we used is:

$$X = |D - A| + |B - C|$$

A, B, C, and D are the pixel values in the image. The shaded area is the area the operation is applied to (Figure 7). The result is placed in a buffer at the location that corresponds to the D location in the original image, as seen in Figure 8.

A	B		
C	D		

Figure 7 – Image Data

	X		

Figure 8 – Result Placement

This procedure continues until the entire image is covered, by shifting the shaded area over one pixel to the right.

The equation checks to see if the pixel at position D is an edge. If the color or the difference in the values are not equal to zero it is possible that there is an edge. If this equation were directly applied to the colored image then you would get an output pictured in Figure 9.

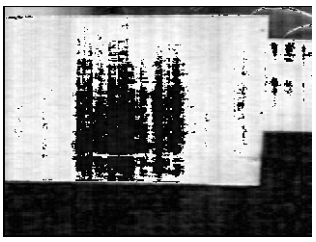


Figure 9 – Edge Equation on Color Image

This resulting image is not a clean and precise output as to the following mono images in Figures 10 and 11.

Figure 10 – Vertical Lines

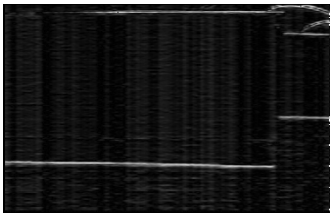


Figure 11 – Horizontal Lines

Figure 10 is the result of the edge equation on the output image from the convolution process using the column mask. Figure 11 is the output from the edge equation on the output from the convolution process using the row mask. These mono images show exactly where the line is and whether or not the line is vertical or horizontal.

After this edge finding equation is applied, the data in the result buffer is then saved as a bitmap file. The file header will be attached, followed by the data in the buffer. Although the newly saved images are only viewable in certain image viewing applications, some of which are Microsoft Paint for Windows and XV for Linux. The reason for this problem was not solved or researched any further. This would be on enhancement for this algorithm. Another programming constraint was each process, the row mask and column mask process, had to be executed independently. If some linear algebra operations were conducted on them to obtain one matrix to convert the image. The resulting image after the convolution process, with this new matrix, would not be same as the images shown in this paper. However, the time for the program to execute completely and write the data to files and save them is on average 1.5 seconds. Which is fairly good for this application. The processing is faster than the amount of time it takes to grab an image with the web camera.

Breakout Board Wiring – Kyle Lamb

The Breakout Board was constructed using a PC/104 board kit, which was ordered from Diamond Systems. An Altera chip was selected and placed on the

board, followed by the many sensor connections and the 50-pin ribbon cable connector which connected to the Prometheus' I/O.

The Altera chip was programmed to interface to the PC/104 bus, as well as generate a PWM signal for each motor to control the speed, and two pulse accumulators, which were used to tell motor position and derive motor speed. The Altera chip was connected to the PC/104 bus and two four-pin connectors for motor encoder input, and two two-pin connectors for PWM and direction output to the motors. All the connections were wire wrapped when available, and soldered otherwise. The Altera connections included connections to the clock from the PC/104 bus, which was divided down for use on the Altera Chip, as well as connections to the Address and Data Bus lines on the PC/104 bus. Since we were using 16-bit addressing connections to the 16-bit addresses on the PC/104 bus had to be added, which meant connecting up the C and D rows of the PC/104 connector, as shown in the schematic in Appendix D.

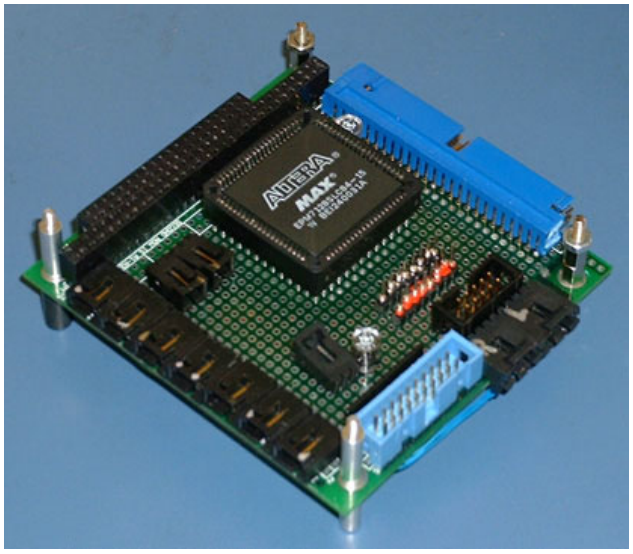


Figure 12: This is our Breakout Board, isn't it beautiful?

All sensor connections were ran to the Breakout Board including distance sensors, flame sensors, the white line sensor, the ultra sonic distance sensor, and the connection to the relay for the fan. For the sensor connections, I used

audio connectors wherever possible, so that there could be no bad connections that would cook the sensors. All the sensor data lines were fed to the appropriate pins on the 50-pin connector and in cases where a 5V supply was needed the power and ground connections were made to the sensor connectors. This made for a clean and easy way in which to connect all of the sensors up and to interface the sensor data to the Prometheus.

When I began, I decided to bring our power in through the PC/104 bus and use this to power the sensors. I looked at the specifications for the PC/104 bus and saw a place were it stated that the bus was capable of supplying up to 500mA. I decided to put a 250mA fuse on the board between the bus power and all of our connections as a safety precaution. When I had everything connected up I discovered that I weren't getting enough power to the sensors, so I decided that the PC/104 bus wasn't supplying enough current despite what the specifications said. I decided to bring in our own power supply to the Breakout Board also using a 250mA fuse and connect all of the sensors up to it while leaving the Altera power connections separate on the PC/104 bus. This worked well and made sure that I was not trying to draw too much current from the PC/104 bus.

After all connections were made and tested, our customer asked us to draw up a schematic to show all the connections. Our customer is planning to have a board made that will use the functionality, which I implemented and which can be changed around to meet the customer needs. It will also be used as the basis of an embedded systems platform.

Behavior Control – Kyle Lamb

The behavior control algorithms that were utilized were designed to use the RT threads developed by Bill. First off, the sensors were characterized, and a math function was developed that would give back a distance in centimeters based on the readings from the analog to digital conversions. The calculations work very well for distances between 3cm and 20cm. Since I was planning to have the robot follow the wall from about 9cm away, these values worked very well.

The wall following function first uses the distance equation to determine distance errors, which are used by a proportional controller, that will speed up or slow down the motors based on the distance. The basic layout of the proportional controller for left wall following is like this.

Left_PWM = Speed_Desired - (Distance – Distance_Desired)*K

Right_PWM = Speed_Desired + (Distance – Distance_Desired) *K

The Speed_Desired is the desired speed in terms of the PWM. I take the distance, which is an error measurement and multiply it times a K to get the desired change in speed to the motor. Notice that the left side subtracts the error, while the right side adds. This is so that when you are getting close to a wall and your distance is larger than your distance desired then the PWM will be a larger number and the left motor will speed up while the right does the opposite and you turn away from the wall. Conversely, when the distance is large the left motor slows down while the right motor speeds up and the robot turns in to the wall. Determining the K can be a little tricky, if it is to small the robot wont correct

well enough, and if it is too big the robot will oscillate and never get the desired distance. The K was determined by computing the constant that would make the Left PWM 10, which is very small, when the robot was a maximum distance away from the wall. Our maximum distance was 20cm due to nonlinearities with the distance sensors so I was able to calculate a K that would work well.

In order to right wall follow I just swapped which sensor I were reading from and changed the plus and minus signs on the PWM calculations. Front wall detection was also implemented, which used the front distance sensor to check if the robot was going to hit anything in front of it. If the front sensor detected a wall the robot would turn in place to the right and continue left wall following, or turn in place to the left and continue right wall following depending on which wall was being followed.

There are separate functions written for both right and left wall following and are included in the code in Appendix C.

Ultra Sonic Distance Sensor – Kyle Lamb

The Ultra Sonic distance sensor was a bit of a mystery throughout the year, and it was not until the end that I took it upon myself to figure out how to properly connect the sensor. The sensor required greater than 9V in order to operate properly, and in all the tests, which were performed earlier the voltage supply was always connected to 5V. Once the sensor was connected to the proper 9V supply it worked beautifully and was mounted and connected up to the Breakout Board post haste.

Bump Sensors – Heather Bitsoi

The basis for this project was to see if the Prometheus SBC was feasible for a mobile robot project and to test the capability of the processing power it has. The customer decided to add additional sensors to the provided robot. One of these sensors were the bump sensors.

The bump sensors are momentary switches that were placed around the base of the robot. The four switches were provided by the client. The wire use to connect these switch was a wire coat hanger. The coat hanger was difficult to bend, making it more difficult to get a perfect half circle. The switch had small rollers on metal that made contact. This roller had to be grinded down for the wire coat hanger to fit in between the roller and lever. The switches were mounted as it is depicted in Figure 13.

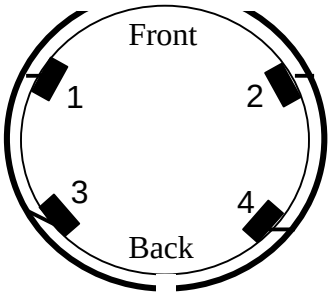


Figure 13 – Bump Sensors

The wire was not extended to the front of the robot because the switches would not work properly. It was a redundant process to have the wire cover the front of the robot in addition to the Sharp GP2D12 IR distance sensor. The removing of the wire allowed for better positioning of the wire and more consistent results. The wire did extend to the back of the robot since there is no way of knowing if we made contact. When the wire made contact with a surface surrounding the robot the output of logic high was sent to the Prometheus's digital input. The difference in the logic depends on the area of contact. The contact areas are depicted in color in Figure 14.

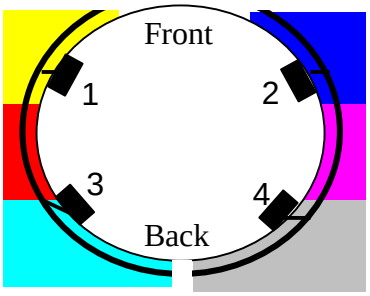


Figure 14 – Color Code for Contact

Switch	Outputs from Contact Switches 1 = Logic High 0 = Logic Low					
1	1	1	0	0	0	0
2	0	0	0	1	1	0
3	0	1	1	0	0	0
4	0	0	0	0	1	1

Table 2 – Output Logic for Contact Areas in Fig. 14

The outputs from the switches are shown in Table 2. The bump sensor design worked consistently and there was no significant difference in the output. The positioning of the wire played the major role in maintaining consistence outputs.

The wire used was the main problem in this design. That was the only difficulty encountered. The overall design was successful.

Hardware Connection – Kyle Lamb

The Robot came pre-assembled so there were not a lot of hardware connections that needed to be made, but a new power supply needed to be added to supply power to the Prometheus, and the variety of sensors that I would be using. In order to meet the power requirements of the robot I decided to go with a 12V lead acid battery, which gave us a surplus of power, and made charging simple. The battery was strapped to the robot using Velcro, which made it easy to take off the battery if so needed. Along with the 12V battery, I added a 3A DC-to-DC converter in order to supply 5V to the Prometheus and the sensors. I also added a 3A fuse and a connector that is used to recharge the battery using an AC adapter. Connections to the various sensors had to be

added, as well as wiring connections to the H-Bridges for motor control, and power connections to the Prometheus SBC, and the Breakout Board.

Altera Programming for Breakout Board – Bill Willems

The Altera code within the Breakout Board has three functions: Position Accumulators to gather encoder data from the motors, PWM generation to control the speed of the motors, and a PC/104 interface to get accumulator data to the Prometheus as well as PWM commands from the Prometheus. PWM Generation was discussed previously and the complete schematic and Altera code can be found in Appendices B & D.

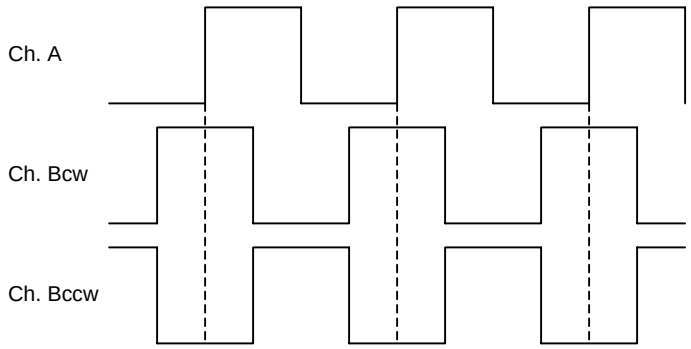


Figure 15: Quadrature Signal from the motor.

We receive data from the motors in the form of a quadrature signal as depicted above in Figure 15. A quadrature signal consists of two channels (A & B); as the motor spins clockwise, we get the signals depicted above by Ch. A and Ch. Bcw. As the motor spins counterclockwise, we get the signals depicted above by Ch. A and Ch. Bccw. As indicated, the direction in which the motor is spinning is easily determined by looking at the state of Ch. B on the rising edge of Ch. A. We decided not to take full advantage of the quadrature encoding for two reasons: 1) The Prometheus is fast enough to read the position

accumulators several times before they overflow and 2) our Altera design barely fit on the chip as it is. Thus, the Position Accumulators are simple up-down 16bit counters.

These counters assume that the signal from the motors is a clean square wave. As this is a physical system, the quadrature signal is prone to noise. To combat this interference, we designed a digital filter with a 6-state state machine depicted below in Figure 16. This filter was applied to both channels of both motors.

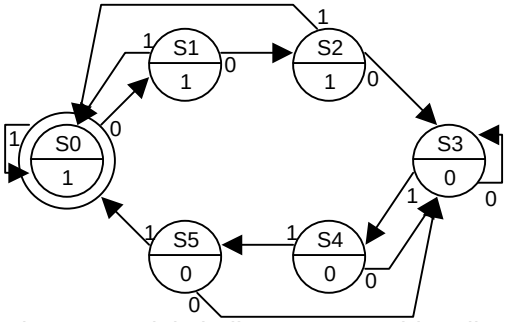


Figure 16: Digital Filter State Machine diagram.

Due to the layout of the motors on the robot chassis, forward rotation (relative to the robot) is clockwise on one motor but counterclockwise on the other. This could be solved in software by negating the value read from one of the accumulators but we decided to solve this in hardware by simply inverting Ch. B of one motor as it comes out of the digital filter.

Altera Code for PWM Generation – Kyle Lamb

In order to control the motor speed we needed to have a Pulse Width Modulation signal incorporated into our Breakout Board, since the Prometheus did not have any PWM generators. The PWM signal was easily incorporated into the Altera code as a counter, which would count up to a certain value sent over

the PC/104 Bus, and remain high as long until the value was reached, and then send out a digital low. Bill and I decided to send over a signed 8-bit number that would use the most significant bit to determine the direction desired out of the motor. Any number over 64 would be considered to be negative, and hence send a signal to the direction on the PWM and reverse the motor direction. Another problem that we encountered was setting the frequency of the PWM signal that we wanted. From previous experience in Junior Design, I knew that a frequency of about one kilohertz would work great. So in order to get the PWM function to work we needed a clock signal to come in that was about 128 kilohertz because we were counting between 0 and 128. This required that we have a divided down clock based on the PC/104 bus clock signal. With the divided down clock it was easy to implement the counter and have the PWM and direction bits connected to an output from the Altera.

Altera Code for 8-bit PC/104 Interface – Bill Willems

The Prometheus utilizes the PC/104 interface bus and so does our Breakout Board (convenient, huh?). There are several ways to utilize this bus: DMA, IRQ-driven, and simple memory mapping. We decided to go with memory mapping the registers we needed, as it is the simplest to implement. The Base Address of the Breakout Board is 0x340. This address was decided because it is below the 1MB boundary (making driver programming easier) and this address was not in use on neither the development machine nor the Prometheus. The Position Accumulators are at addresses 0x340 and 0x342 (16-bit addresses) and the PWM command registers are at 0x344 and 0x345 (8-bit addresses).

Altera Code for 16-bit PC/104 Interface – Kyle Lamb

When we were working on the PC/104 bus connections on the Breakout Board we ran into troubles with Altera timing issues using the 16-bit reads. I was able to write a nested if statement in Altera, which solved the timing issue, by bringing the IOCS16 line low before IOR or IOW went low. With the 16-bit reads fixed we were able to continue with the development of the PC/104 bus, and enable read in 16-bit data from the motor pulse accumulators.

Installation of RT Linux – Bill Willems

We are using RedHat Linux 7.2 with the 2.4.19-pre6 kernel and RTLinux V3.1. We chose to use such a bleeding edge kernel because of several USB controller fixes available only in this kernel. Due to the limited size of our Compact Flash drive, we had to find a way to get a minimal set of instructions. To do this, we installed the “Swiss Army Knife of Embedded Linux”, BusyBox version 0.60.2.

We started out with the 2.4.4 kernel because RTLinux was designed around this kernel. We upgraded to a newer kernel to support devfs and upgraded again to the 2.4.19-pre6 kernel to solve our USB controller issues. The first thing we needed to do was to compile the kernel in such a way that it would fit in the boot sector of the Compact Flash drive. Once that was complete, we designed the root filesystem using BusyBox. We then wrote a script to reformat the Flash disk, create the ext3 filesystem, copy our filesystem source tree and then ran LILO from the Flash disk to correctly write the boot sector. This does not sound like much but it took months to perfect the technique. Once RTLinux was

up-and-running on the Prometheus, we were then able to test our Breakout Board, the corresponding driver, and control algorithms.

Breakout Board Driver – Bill Willems

In order to read and write data to and from the Breakout Board and the Prometheus I/O module, we needed to write a driver for it. Our module sets up threads to gather data from the hardware and pass that to the Behavior Control algorithm. This bi-directional communication is established using RT-FIFOs. There are three types of FIFOs: a Command FIFO used by the user-space control algorithm to start and stop threads as well as to pass PWM commands to the hardware. Several Data FIFOs from which the user reads the data the threads have received from the hardware and a Command FIFO that is used by the module to pass user-space commands to the appropriate thread. The complete code for this driver can be found in Appendix C and uses five subroutines: `init_module`, `cleanup_module`, `my_handler`, `intr_handler`, and `thread_code`.

This driver is designed to be a loadable kernel module. When a module is loaded into kernel memory, the code that is executed to initialize the module is `init_module`. This subroutine sets up any hardware the module needs to use, memory spaces, pointers to functions, etc. Our module accesses our Breakout Board and the Prometheus I/O module using threads and RT-FIFOs. First, we need to destroy the FIFOs we will be using in case they are already in use by another process. We then create the FIFOs to get the data to and from the user-space Behavior Control algorithm as well as the threads to gather all of the data.

Next, we setup the Prometheus I/O (we do not need to setup the Breakout Board as it is simple memory-mapped registers). We enable Digital interrupts and disable Analog I/O and Timer interrupts as well as DMA operations. We then configure the Analog I/O module by setting the gain to a unipolar 0-5V as well as setting the FIFO depth to one. Next, we configure the Digital I/O module by setting Ports A & B to be input and the lower byte of Port C to output. To finish the setup of the Prometheus I/O module we enable the interrupt the I/O module uses (IRQ 5) and assign an ISR (`intr_handler`). Finally, to complete the initialization of the module we assign a device file handler (`my_handler`) for our Control FIFO. This code is demonstrated below in Figure 17.

```
int init_module(void)
{
  pthread_attr_t attr;
  struct sched_param sched_param;
  char temp;

  /* make sure the RT-FIFOs we want aren't already in use.  */
  rtf_destroy(CONTROL_FIFO);
  rtf_destroy(R_POS_FIFO);
            ...
  /* create our FIFOs */
  rtf_create(CONTROL_FIFO, CMD_FIFO_BUFF_SIZE);
  rtf_create(R_POS_FIFO, FIFO_BUFF_SIZE);
            ...
  /* create threads */
  pthread_attr_init (&attr);
  sched_param.sched_priority = POS_PRIORITY;
  pthread_attr_setschedparam (&attr, &sched_param);
  pthread_create (&tasks[R_POS_FIFO], &attr, thread_code, /
  (void *)R_POS_FIFO);
  pthread_attr_init (&attr);
  sched_param.sched_priority = POS_PRIORITY;
  pthread_attr_setschedparam (&attr, &sched_param);
  pthread_create (&tasks[L_POS_FIFO], &attr, thread_code, /
  (void *)L_POS_FIFO);
            ...
  /* setup Prometheus I/O */
  /* Reset FIFO depth to 0, clear all the int. flags. */
  CLR_INTS;
  /* disable Analog Input Interrupts */
  temp = rtl_inb(INT_DMA_COUNT_CTL);
  temp &= ~ENABLE_AIO_INTS;
```

```
  rtl_outb(temp, INT_DMA_COUNT_CTL);
  /* Set appropriate Gain */
  rtl_outb(UNIPOLAR_GAIN_0_5_V, AD_GAIN_SCAN_SET);
  /* FIFO depth to 1 */
  rtl_outb(0x01, FIFO_THRESH);

  /* Configure the DIO */
  temp = rtl_inb(DIO_CTL); // get current config
  temp |= (DIR_PORTA_INPUT | DIR_PORTB_INPUT);  // PORTs A&B are
                                                // input
  temp &= ~(DIR_PORTC_3_0_INPUT | DIOCTR); // PORT Cl to output
                                           // and DIOCTR to 0
  rtl_outb(temp, DIO_CTL);

  /* Enable DIO interrupts */
  temp = rtl_inb(INT_DMA_COUNT_CTL);
  temp |= ENABLE_DIO_INTS;
  rtl_outb(temp, INT_DMA_COUNT_CTL);

  /* enable the Prometheus IRQ and assign a handler */
  rtl_request_irq(PROMETHEUS_IRQ, intr_handler);

  /* Disable timer interrupts and DMA operation */
  temp = rtl_inb(INT_DMA_COUNT_CTL);
  temp &= ~(ENABLE_DMA | ENABLE_TIMER_INTS);
  rtl_outb(temp, INT_DMA_COUNT_CTL);

  /* assign a device file handler for the Control FIFO */
  rtf_create_handler(CONTROL_FIFO, &my_handler);

  return 0;
}
```

Figure 17: `init_module` example code.

Just as a module can be loaded, it can also be unloaded. For this, we need a routine to be executed that frees any memory we allocated, shuts down the hardware, etc. In our case, the routine cleanup_module is executed to destroy the FIFOs we are using, free our threads, release the Prometheus IRQ, and send a PWM of zero to stop the motors. This code is demonstrated below in Figure 18.

```
void cleanup_module(void)
{
  /* let go of fifos */
  rtf_destroy(CONTROL_FIFO);
  rtf_destroy(R_POS_FIFO);
            ...
  /* cancel threads */
```

```
  pthread_cancel (tasks[R_POS_FIFO]);
  pthread_join (tasks[R_POS_FIFO], NULL);
  pthread_cancel (tasks[L_POS_FIFO]);
  pthread_join (tasks[L_POS_FIFO], NULL);
            ...
  /* free the Prometheus IRQ */
  rtl_free_irq(PROMETHEUS_IRQ);

  /* make sure the motors have stopped and the fan is off */
  rtl_outb((char) 0, (short) R_PWM_ADDR);
  rtl_outb((char) 0, (short) L_PWM_ADDR);
  rtl_outb(0, DIO_PORTC);
}
```

Figure 18: `cleanup_module` example code.

When the user-space control algorithm writes to the Control FIFO the function `my_handler` is executed. This handler reads the structure (depicted below in Figure 19) from the Control FIFO. `my_handler` then wakes up the appropriate thread.

```
struct my_msg_struct
{
  int command;
  int task;
  unsigned int period;
  int rw; /* read = 1; write = 0 -- only used with PWM fifos */
  char data; /* only used with PWM fifos writes – PWM command */
};
```

Figure 19: `my_msg_struct`

When the Prometheus generates an interrupt, our ISR (`intr_handler`) is executed. Similar to `my_handler`, this routine wakes up the appropriate thread but also clears the interrupt.

Now we come to the workhorse of our module – `thread_code`. This is the code that actually reads the data from the hardware (as well as writing PWM values to the motors). Once a thread is created, it begins executing. Thus, the first thing our thread needs to do is go to sleep until woken up by the user-space control algorithm. When it does wake up, it goes into an infinite while loop where

the first thing it does is check the Command FIFO for start/stop instructions. The start instruction sets up how the thread will function. That is, is this thread periodic or non-periodic (interrupt driven threads are a special case of non-periodic threads)? The stop command simply puts the thread to sleep – in the case of a PWM thread, we send a PWM command of 0 to stop the motors before it goes to sleep.

Next, the thread must gather the data. To read the Position Accumulators, we simply read a word (two bytes) of data from the appropriate address using the Linux function `inw(short address)`. To write to the PWM command registers, we use the RTLinux function `rtl_outb(char data, short address)`. The variable `data` is a `signed char` data type. This is an 8-bit value where the most significant bit (the sign bit) determines the direction (relative to the robot – forward or reverse) we wish the motors to spin. The remaining seven bits represent the speed that the motors should spin. A value of 0 represents a speed of 0% (stopped) and a value of 127 represents a speed of 100% (full speed).

Any Analog I/O data we need to gather (IR Distance Sensors, Ultrasonic Distance Sensor & IR Flame Sensors) makes use of the Prometheus A/D module, which was setup in the `init_module` routine. The first thing we need to do is to clear the A/D interrupt and reset the A/D FIFO depth to zero by writing a 0x11 to the Command Register (0x280) for the Prometheus Data Acquisition Circuitry. We then reset the FIFO Threshold to 1 by writing a 0x01 to the FIFO Threshold Register (0x285) and set which channel to scan by writing to the A/D Channel Register (0x282). Next, we trigger the A/D Scan by writing a 0x80 to the

Command Register (0x280) and wait for the scan to finish by waiting for bit 7 of the Analog Input Status Register (0x283) to go low. Finally, we must read the value of the A/D conversion (a 16-bit value) by reading the least significant byte from address 0x280 and the most significant byte from address 0x281. To make the code easier to read, we wrote macros for each of these steps – these macros are defined in prometheus.h found in Appendix C.

To read Digital I/O values (bump sensors and the white line sensor) we simply read from the appropriate DIO register: 0x288 for Port A and 0x289 for Port B. To write to the Digital I/O (the fan) we write a 1 to the appropriate bit of Port C (0x28A).

The Prometheus has only one interruptible Digital I/O line. The documentation of this DIO Interrupts implies that an interrupt could be generated on the falling edge of the input line. We took this to mean that we could setup this interrupt to be generated whenever any subset of the 24 available digital input lines goes low. Unfortunately, this is not the case – the DIO Interrupt has nothing to do with the 24 digital I/O lines. This interrupt is only generated when a separate pin (the External Trigger line) goes low. If we want an interrupt to be generated when any of the DIO lines go high, we would have to tie them to the Digital I/O lines as well as the External Trigger line (through an N-channel NOR Gate). Instead of doing this, we only tied the tone-decoder digital I/O line to the External Trigger. When we catch this interrupt, we simply place a default value into the appropriate RT-FIFO.

Now that all of the data has been gathered, we need to place it into the appropriate RT-FIFO for the user-space control algorithm to read. We do this with the RT-Linux function `rtf_put(int fifo, char *data, int size)`.

Finally, the thread must go to sleep. If the thread is periodic, the thread must sleep until the next period using `pthread_wait_np()`. This function tells the thread to wait until the next period or until the user-space control algorithm wakes it up. If the thread is non-periodic, the thread is put to sleep by executing `pthread_suspend_np(pthread_self())`, which tells the thread to sleep until it is explicitly woken up by the control algorithm.

Hindsight is 20/20

Breakout Board – Bill Willems

If, given the chance to restart this project knowing what I know now, there are a few things I would do differently. For one, I would have gone with buying a third-party timer / pulse accumulator board instead of designing and building our own. This aspect of our project occupied all of my time during the first semester and became a “project-within-a-project”. If we used a third party board, we would only have to convert (or rewrite) the manufacturer-supplied driver to a Realtime driver. This would have allowed more time to be devoted to other aspects of the project.

There are also other, smaller changes I would have made. Had I known that our client was waiting for the go-ahead from us to order our Prometheus, I would have ordered it sooner. I would have given the task of interfacing to the Prometheus I/O to Kyle because he was waiting for me to finish it before he could develop his Behavior Control algorithms. I wouldn’t have nuked the WindowsNT boot partition on our development system. I would also have setup our weekly team meetings (outside of our weekly meetings with our advisor) during the first semester.

Despite these grievances, I am satisfied with the end-result of this project. Had we gained the time lost to development of our Breakout Board, this project could have achieved so much more. However, our client is very happy that we did design and build our own expansion board and he has big plans for it – I guess I can’t complain.

Future Work

Breakout Board and Driver – Bill Willems

With the reconfigurable Altera chip on the Breakout Board, the possibilities for improvement are endless. USB, for instance, is very CPU intensive – if a USB controller could be placed on the Altera chip, it would ease the overhead that now rests with the CPU. If some image processing firmware could be developed, imagine what we could do with that! The Altera chip has a JTAG programming interface that allows it to be programmed through the parallel port. The Prometheus has a parallel port – so, theoretically, the Altera chip could be reprogrammed on the fly.

Before the driver for our Breakout Board is released into the public domain, it needs to become more robust. The reason it works so well for us is because we wrote it and, therefore, we know exactly how to interface to it. I am sure there is room to improve the efficiency of the code as well as some error checking. Such improvements in efficiency could include breaking up the one version of `thread_code` routine into several “flavors”: one for the Prometheus A/D interface, the Digital interface, and the Breakout Board; One could also break them up into periodic and non-periodic threads. Future error checking could include code to ensure that the Prometheus Data Acquisition Circuitry, as well as the Breakout Board, is out there.

Image Processing – Heather Bitsoi

For the continuation of the image processing here are a few details of where the project ends and how to continue.

To compile	**To run the program**	**Outputs**
gcc –o <filename> <filename>.c	root:~$ ./<filename> <image filename w/o extension>	The altered images from program

The commands above will compile the program. The current program finds vertical and horizontal lines. There is additional code provided which was used for testing purposes. In the testing algorithm, the test was conducted to determine the convolution matrix that was most efficient and effective for finding edges.

There are many more applications that could be added to the image processing part of the project. Applications like determining how far the line is relative to the robot, the magnitude and orientation of the line, determining whether a line has moved or not using the subtraction of images. This method would be an approach for multi-frame line tracking. Other applications like object recognition would be possible. For instance, the furniture in the maze for the firefighting robot is yellow. The color value of yellow can be compared to the pixels in the image to determine whether a furniture object is in the image or not. The image can be re-configured to a 2-bit, 4-bit, or 8-bit image. Making it possible to determine the difference in colors easier being that the number of colors is less than the 24-bit image obtained from the web camera. Reducing the number of colors in the image could make determining the objects in the field of

view easier to recognize. The walls could be one color and the background or door could be another. All in all, the future of image processing is endless. The procedure on how to get there is the hard part, which the same attribute faced in this project.

Summary & Conclusion

The final product of our labors is a robot that is using the Prometheus SBC along with RT Linux to read and respond to sensor data in real time. We loaded RT Linux onto a small 128MB Flash disk and were able to boot from it, as well as interface to our breakout board and the Prometheus I/O. The robot is also able to network and use remote login capabilities, which allow the user to connect with the robot over great distances. A wide variety of sensors have been connected to the robot, all of the existing sensors, as well as bump sensors and the ultrasonic distance sensor. The Breakout Board we built is fully functional, and is highly adaptable so that it can be changed to meet customer desires. Along with the Breakout Board hardware, we have supplied a driver for the breakout board that can be easily modified to add more functionality. We have also accomplished some Image Processing and are able to modify an image to show only the vertical and horizontal lines. We can also use the Prometheus SBC to capture images from the USB cameras, which can then be used by our image processing algorithms. As far as behavior control goes, we have programmed the robot to left and right wall-follow, as well as respond to front sensor input. Code has been written to interface to and check for updates on the rest of the sensors as well. We were unable to implement Image Processing into the functionality of the robot due to time and work constraints.

References

PC/104 Embedded Consortium; “PC/104 Specification Version 2.4”; August 2001; http://www.pc104.org

Messmer, Hans-Peter; The Indispensable PC Hardware Book 3rd Edition, Addison-Wesley, 1997

Diamond Systems Corporation; “PROMETHEUS TM High Integration PC/104 CPU with Ethernet and Data Acquisition Models PR-Z32-E-ST, PR-Z32-EA-ST User Manual V1.35” http://www.diamondsystems.com

Barabanov, Michael; “Getting Started with RTLinux”, Finite State Machine Labs, Inc. September 26, 2001

Linux Online, Inc. “What Is Linux?”, March 2002, http://www.linuxonline.com

Umbaugh, Scott E; Computer Vision and Image Processing, Prentice Hall, 1998

Appendix A: Our Interface to the PC/104 Bus

The PC/104 Interface is electrically identical to the ISA bus found on the common PC with a few minor differences. Figure 20 below shows the PC/104 connector and the pins we are interested in. All of the Altera code to handle the interface to the PC/104 bus can be found in Appendix B.

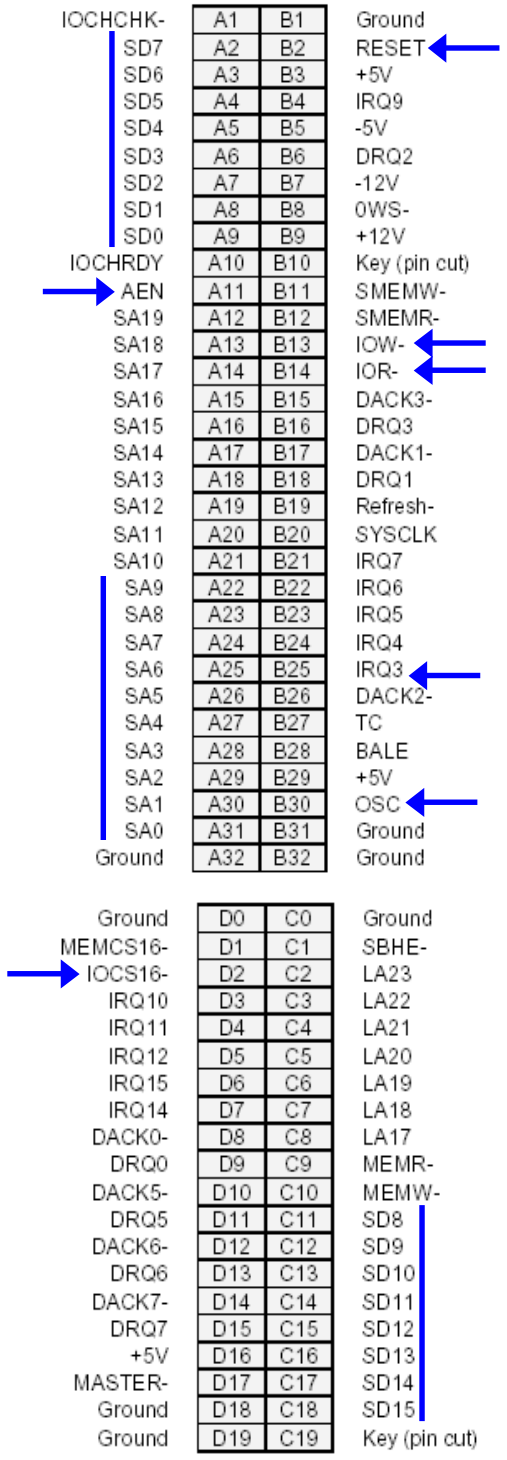


Figure 20: PC/104 Connector

SD0-SD15: Data Bus

SA0-SA9: Address Bus – this bus is actually 24-bits wide but, due to the backwards compatibility requirement of the ISA bus, we can get away with using only ten.

RESET: System Reset

IOR/IOW: Read/Write signal lines – when this line goes low, the system is reading (or writing) to the ISA device.

OSC: System Clock

IOCS16: Signal line to indicate a 16-bit read or write.

BALE: Address Latch Enable – when this line goes high, the data on the address line has had enough time to settle and is valid.

AEN: Address Enable line. This line indicates whether or now the address that is now on the address bus pertains to a read (or write) from memory. When this line is high, the SBC is reading from memory. When it is low, the SBC is reading from the ISA (or PCI bus).

The Position Accumulators are simple 16-bit memory-mapped up-down counters at address 0x340 and 0x342. When the SBC requests this information, it places the address on the address bus and brings AEN low. We need to recognize this action and bring IOCS16 low (because this is a 16-bit read or write) before the SBC brings the IOR or IOW line low. Once the IOR line is low, we can place the data onto the data bus. If the IOW line goes low, we do not do anything because the Position Accumulators are Read-Only registers. Once our address disappears from the address bus, we let go of the data bus. This process is depicted below in Figure 20.

Figure 20: Timing Diagram for a 16-bit Read & Write on the PC/104 Bus.

The PWM command registers are 8-bit memory-mapped registers at addresses 0x344 and 0x345. When the SBC requests this information, it places the address on the address bus and brings AEN and IOR or IOW low. If IOR goes low, we place the data onto the data bus. If the IOW line goes low, we read

the value from the data bus and store it in an internal register. Once our address disappears from the address bus, we let go of the data bus. This process is depicted below in Figure 21.

Figure 21: Timing Diagram for an 8-bit Read & Write on the PC/104 Bus.

Appendix B: Altera Code

```
% pc104inter.tdf                                  %
% Interface to PC/104 Bus for Senior Design:     %
% RT Linux On A Small Mobile Robot               %
% Written for Diamond Systems Prometheus SBC     %

% Bill Willems 1/3/02 - Written for 8-bit interface   %
% Bill Willems 1/13/02 - updated for 16-bit interface;%
%                        made PWM regs R/W            %

% Constants %
CONSTANT    OUR_ADDRESS = B"110100"; % We can use 0x340 - 0x34F %
CONSTANT    CTRL_R_POS = B"0000"; % Position Accumulators need  %
CONSTANT    CTRL_L_POS = B"0010"; % 16 bit transfers.           %
CONSTANT    CTRL_R_PWM = B"0100"; % PWM Registers only require  %
CONSTANT    CTRL_L_PWM = B"0101"; % only 8 bit transfers.       %

SUBDESIGN pc104inter
(
      % NOTE: _n indicates line is active LOW %

      % PC/104 Bus specific input lines %
      RESET             :     INPUT;
      addr[9..0]        :     INPUT; % address bus-there are 20 lines %
                                     % but we can get away with 10    %
      IOW_n, IOR_n      :     INPUT; % I/O R/W lines %
      AEN               :     INPUT; % Address Enable-used with DMA   %
      BALE              :     INPUT; % Address Latch Enable - when is %
                                     % the address valid? %

      % Robot specific input lines %
      R_pos[15..0]      :     INPUT;
      L_pos[15..0]      :     INPUT;

      % PC/104 specific output lines %
      IOCS16            :     BIDIR; % Asserted by us to allow a      %
                                     % 16bit transfer instead of 2    %
                                     % 8bit transfers                 %

      % Robot specific output lines %
      R_pwm[7..0]       :     OUTPUT;
      L_pwm[7..0]       :     OUTPUT;

      % Bidirectional Lines %
      data_bus[15..0]   :     BIDIR; % PC/104 Data Bus %
)

VARIABLE
      R_pwm_latch[7..0] :     DFF;  % PWM counters compare to these %
      L_pwm_latch[7..0] :     DFF;
      addr_latch[9..0]  :     DFF;  % latch what's on the addr. Bus %
      data[15..0]       :     NODE; % Internal Data Bus %
      thats_a_read      :     NODE; % Is the SBC reading from us? %
      hey_thats_us      :     NODE; % Are you talkin' ta me?? %
      latch_R_pwm       :     NODE; % use these to clock the PWM %
```

```
      latch_L_pwm       :     NODE; % latches.                    %
      enable16          :     NODE; % tri-state enable for IOCS16 %
      shitbrick         :     NODE; % tri-state enable for the upper %
                                    % half of the data bus during    %
                                    % 16-bit transfers %

BEGIN
      DEFAULTS
            latch_R_pwm = VCC;
            latch_L_pwm = VCC;
            thats_a_read = GND;
            hey_thats_us = GND;
            enable16 = GND;
            shitbrick = GND;
      END DEFAULTS;

      % Setup clocks and clears %
      R_pwm_latch[].clk = latch_R_pwm;
      L_pwm_latch[].clk = latch_L_pwm;
      addr_latch[].clk = BALE;
      R_pwm_latch[].clrn = !RESET;
      L_pwm_latch[].clrn = !RESET;
      addr_latch[].clrn = !RESET;

      % wire latch inputs %
      R_pwm_latch[].d = data_bus[7..0];
      L_pwm_latch[].d = data_bus[7..0];
      addr_latch[].d = addr[];

      % wire pwm latch outputs %
      R_pwm[] = R_pwm_latch[].q;
      L_pwm[] = L_pwm_latch[].q;

      % you talkin' ta me?? - Robert DeNiro kicks a$$ %
      IF (addr_latch[9..4].q == OUR_ADDRESS) AND (AEN == GND) THEN
            hey_thats_us = VCC;
      END IF;

      % Put/Get stuff on/from the pc/104 bus %
      % reads %
      IF (hey_thats_us == VCC) AND
         (addr_latch[3..0].q == CTRL_R_POS) THEN
            enable16 = VCC;
            IF (IOR_n == GND) THEN
                  thats_a_read = VCC;
                  shitbrick = VCC;
                  data[] = R_pos[];
            END IF;
      END IF;
      IF (hey_thats_us == VCC) AND
         (addr_latch[3..0].q == CTRL_L_POS) THEN
            enable16 = VCC;
            IF (IOR_n == GND) THEN
                  thats_a_read = VCC;
                  shitbrick = VCC;
                  data[] = L_pos[];
            END IF;
```

```
        END IF;
        IF (hey_thats_us == VCC) AND (addr_latch[3..0].q == CTRL_R_PWM)
           AND (IOR_n == GND) THEN
              thats_a_read = VCC;
              data[7..0] = R_pwm_latch[7..0].q;
              data[15..8] = H"55";
        END IF;
        IF (hey_thats_us == VCC) AND (addr_latch[3..0].q == CTRL_L_PWM)
           AND (IOR_n == GND) THEN
              thats_a_read = VCC;
              data[7..0] = L_pwm_latch[7..0].q;
              data[15..8] = H"55";
        END IF;
        IF (hey_thats_us == VCC) AND (addr_latch[3..0].q > CTRL_L_PWM)
           AND (IOR_n == GND) THEN
              thats_a_read = VCC;
              data[] = H"55";
        END IF;

        % writes %
        IF (hey_thats_us == VCC) AND (addr_latch[3..0].q == CTRL_R_PWM)
           AND (IOW_n == GND) THEN
              latch_R_pwm = GND;
              data[] = H"55";
        END IF;
        IF (hey_thats_us == VCC) AND (addr_latch[3..0].q == CTRL_L_PWM)
           AND (IOW_n == GND) THEN
              latch_L_pwm = GND;
              data[] = H"55";
        END IF;

        % Tri-state the data bus %
        IOCS16 = TRI(B"0", enable16);
        data_bus[15] = TRI(data[15], shitbrick);
        data_bus[14] = TRI(data[14], shitbrick);
        data_bus[13] = TRI(data[13], shitbrick);
        data_bus[12] = TRI(data[12], shitbrick);
        data_bus[11] = TRI(data[11], shitbrick);
        data_bus[10] = TRI(data[10], shitbrick);
        data_bus[9] = TRI(data[9], shitbrick);
        data_bus[8] = TRI(data[8], shitbrick);
        data_bus[7] = TRI(data[7], thats_a_read);
        data_bus[6] = TRI(data[6], thats_a_read);
        data_bus[5] = TRI(data[5], thats_a_read);
        data_bus[4] = TRI(data[4], thats_a_read);
        data_bus[3] = TRI(data[3], thats_a_read);
        data_bus[2] = TRI(data[2], thats_a_read);
        data_bus[1] = TRI(data[1], thats_a_read);
        data_bus[0] = TRI(data[0], thats_a_read);

END;
```

```
% dig_filter.tdf       %
% Digital Filter       %
% Bill Willems 1/3/02  %

SUBDESIGN Dig_Filter
(
      clk   :     INPUT;
      In    :     INPUT;
      Out   :     OUTPUT;
)

VARIABLE

      Digital_Filter_Machine : MACHINE
                               WITH STATES (s0, s1, s2, s3, s4, s5);

BEGIN

      Digital_Filter_Machine.clk = clk;
      CASE Digital_Filter_Machine IS
            WHEN s0 =>
                  Out = VCC;
                  IF In == GND THEN
                        Digital_Filter_Machine = s1;
                  ELSE
                        Digital_Filter_Machine = s0;
                  END IF;
            WHEN s1 =>
                  Out = VCC;
                  IF In == GND THEN
                        Digital_Filter_Machine = s2;
                  ELSE
                        Digital_Filter_Machine = s0;
                  END IF;
            WHEN s2 =>
                  Out = VCC;
                  IF In == GND THEN
                        Digital_Filter_Machine = s3;
                  ELSE
                        Digital_Filter_Machine = s0;
                  END IF;
            WHEN s3 =>
                  Out = GND;
                  IF In == VCC THEN
                        Digital_Filter_Machine = s4;
                  ELSE
                        Digital_Filter_Machine = s3;
                  END IF;
            WHEN s4 =>
                  Out = GND;
                  IF In == VCC THEN
                        Digital_Filter_Machine = s5;
                  ELSE
                        Digital_Filter_Machine = s3;
                  END IF;
            WHEN s5 =>
                  Out = GND;
```

```
                IF In == VCC THEN
                      Digital_Filter_Machine = s0;
                ELSE
                      Digital_Filter_Machine = s3;
                END IF;
      END CASE;
END;

% div_clk.tdf                                                  %
% Divide the Clock Down                                       %
% The PC/104 Bus has a 14.31818 MHz clock.                    %
% Divide it down to get ~1 Mhz and ~128 kHz clocks            %
% Bill Willems 1/3/02                                         %

SUBDESIGN div_clk
(
      clk_in      :     INPUT;
      pwm_clk     :     OUTPUT;
      filter_clk  :     OUTPUT;
)

VARIABLE
      div_2       :     DFF;
      div_4       :     DFF;
      div_8       :     DFF;
      div_16      :     DFF;
      div_32      :     DFF;
      div_64      :     DFF;
      div_128     :     DFF;

BEGIN
      div_2.clk = clk_in;
      div_4.clk = div_2.q;
      div_8.clk = div_4.q;
      div_16.clk = div_8.q;
      div_32.clk = div_16.q;
      div_64.clk = div_32.q;
      div_128.clk = div_64.q;

      div_2.d = !div_2.q;
      div_4.d = !div_4.q;
      div_8.d = !div_8.q;
      div_16.d = !div_16.q;
      div_32.d = !div_32.q;
      div_64.d = !div_64.q;
      div_128.d = !div_128.q;

      pwm_clk = div_128.q;    % 111.86 kHz %
      filter_clk = div_8.q;   % 1.7857 MHz %

END;
```

```
% pos_pulse_accumulator.tdf                        %
% Pulse Accumulator for Position Registers        %
% Accumulator value is latched in pc104inter      %
% Bill Willems 1/3/02                             %
SUBDESIGN Pos_Pulse_Accumulator
(
      Ch_A              :     INPUT; % clock for accumulators %
      Ch_B              :     INPUT;
      RESET             :     INPUT;
      ACC_OUT[15..0]    :     OUTPUT;
)

VARIABLE
      Accumulator[15..0]      :     DFF;  % we gotta put the count %
                                          % somewhere... %

BEGIN
      % Set clocks and clears %
      Accumulator[].clk = Ch_A;
      Accumulator[].clrn = !RESET;
      ACC_OUT[] = Accumulator[].q;

      % So count already! %
      IF Ch_B == VCC THEN
            Accumulator[].d = Accumulator[].q + 1;
      ELSE
            Accumulator[].d = Accumulator[].q - 1;
      END IF;
END;

% pwmgen.tdf                                                          %
% PWMgen - Generate PWM with a duty cycle based on input value.       %
% Kyle Lamb 1/3/02                                                    %
subdesign PWMgen
(
      clock             :     INPUT;
      PWM_cmd[7..0]     :     INPUT;
      RESET             :     INPUT;

      DC_out            :     OUTPUT;
      dir_out           :     OUTPUT;
)
VARIABLE
      count[6..0]       :     DFF;
BEGIN
      count[].clk = clock;
      count[].clrn = !RESET;

      dir_out = PWM_cmd[7];

      count[].d = count[].q +1;

      IF (PWM_cmd[6..0] <= count[].q) THEN DC_out = GND;
      ELSE DC_out = VCC;
      END IF;
END;
```

Appendix C: Breakout Board Driver

Note: bb_mod.c, bb.h, prometheus.h, bb.c, image.c, and Makefile are not included in text here. You should have received a copy of these files in electronic format with this text.

Appendix D: Breakout Board Schematic

Note: The schematic file is not included in text here. You should have received a copy of this file in electronic format with this text.

Appendix E: Features of the Prometheus SBC

System Features	
Processor	
486-DX2 processor running at 100MHz with co-processor	Pentium class platform including burst-mode SDRAM and PCI-based IDE controller and USB
32MB SDRAM system memory	50MHz memory bus for improved performance
2MB 16-bit wide integrated flash memory for BIOS and user programs	8KB unified level 1 cache
I/O	
4 serial ports, 115.2kbaud max	2 ports 16550-compatible, 2 ports 16850-compatible with 128-byte FIFOs
2 full-featured powered USB ports	1 ECP-compatible parallel port
Floppy drive connector	IDE drive connector (44-pin version for notebook drives)
Accepts solid-state flashdisk modules directly on board	10BaseT full-duplex PCI bus mastering Ethernet (10Mbps)
IrDA port (requires external transceiver)	PS/2 keyboard and mouse ports
Speaker, LEDs	
Other System Features	
Plug and play BIOS with IDE autodetection, 32-bit IDE access, and LBA support	Built-in fail-safe boot ROM for system recovery in case of BIOS corruption
User-selectable COM2 terminal mode	On-board lithium backup battery for real-time-clock and CMOS RAM
ATX power switching capability	Programmable watchdog timer
Power surge monitor for fail-safe operation	Zero wait-state capability for flash memory and PC/104 bus
+5V-only operation	Extended temperature range operation (-40 to +85°C)
Data Acquisition Subsystem	
Analog Input	
16 single-ended / 8 differential inputs, 16-bit resolution	100KHz maximum aggregate A/D sampling rate
Programmable input ranges/gains with maximum range of ±10V / 0-10V	Both bipolar and unipolar input ranges
5 ppm/°C drift accuracy	Internal and external A/D triggering
48-sample FIFO for reliable high-speed sampling and scan operation	
Analog Output	
4 analog outputs, 12-bit resolution	±10V and 0-10V output ranges
Simultaneous update	Adjustable output range (optional)
Digital I/O	
24 programmable digital I/O, 3.3V and 5V logic compatible	Enhanced output current capability: –8/+12mA max
Counter / Timers	
1 24-bit counter/timer for A/D sampling rate control	1 16-bit counter/timer for user counting and timing functions
Programmable gate and count enable	Internal and external clocking capability

Beowulf Cluster

Research and Implementation

A Thesis

Presented to the Faculty of the

Electrical Engineering Department

of

New Mexico Tech

By

Michael Berg

Kevin Fisher

Anthony Montoya, Jr.

In partial fulfillment of the requirements

for the course

EE-482 Senior Design Project II

May, 2002

Beowulf Cluster Research and Implementation

Michael Berg

Kevin Fisher

Anthony Montoya, Jr.

Electrical Engineering Department

New Mexico Tech

ABSTRACT

A Beowulf cluster is a method of building a super-computer by connecting numerous machines together and getting them to work in parallel on a single task. The design and construction of a Beowulf cluster is still a highly technical activity, requiring a proficiency in operating systems, networking, and system design. The primary goal of this project is to provide the customer with a functional and easy to use Beowulf cluster constructed from surplus computers. A secondary goal of this project it to allow existing parallel computing techniques to be more easily applied to future Beowulf cluster work at New Mexico Tech and elsewhere. This project promotes the use of surplus computers – readily available to university environments – in Beowulf clusters; and will simplifies the process of constructing a Beowulf cluster. This makes Beowulf clusters more accessable to researchers in other disciplines, such as physics and biology.

Acknowledgments

First we would like to thank Dr. Scott W. Teare for sponsoring this project. Dr. Kevin Wedeward was our faculty advisor, and we would like to thank him for advising our team throughout the year. Dr. David Westpfahl, Jr. was a great help by providing the surplus computers used in the construction of this Beowulf cluster. Denis Oesch provided the project with a copy of his Matlab "100 inch mirror coating" simulation as a real world application for porting to the ZPL programming language and executing on the finished Beowulf cluster. Finally, we would like to thank everyone who has contributed to the Linux Documentation Project. The background information, HOWTOs, and troubleshooting guided contained in that archive were the foundation for the work that was accomplished in this project.

Table of Contents

List of Figures

1 Project Background and Goals

1.1 Project History

The first Beowulf cluster was built by Donald Becker in 1994 at NASA's Goddard Space Flight Center. Since then, further research has been done by other national labs, universities, researchers, businesses, and hobbyists. This Beowulf cluster project is based on this existing body of work with the goal of simplifying the process for application to small scale university projects.

1.2 Project Description

The design and construction of a Beowulf cluster is still a highly technical activity, requiring a proficiency in operating systems, networking, and system design. The goal of this project is to simplify this process so that the existing techniques can more easily be applied to future Beowulf cluster work at New Mexico Tech and elsewhere. This project targets using surplus computers that are readily available to university environments simplifies the construction of a Beowulf cluster for researchers in other disciplines such as physics and biology.

This completed project delivers a Linux-based Beowulf cluster consisting of eight functional nodes. At the time of this document, several nodes were not functioning properly and have been excluded from the final cluster. This issue will be addresses as well as possible with the next week. This cluster runs the ZPL parallel computing software in conjunction with the user's choice of PVM or MPI parallel libraries.

Software implementing a parallel solution to a real world computation problem of interst to the client is also provided. A final collection of documentation is included. The documentation describes the results of out investigation into other Beowulf cluster implementations and why we made certain design decisions on this Beowulf cluster. The documentation includes setup and usage instructions, scalability analysis of the hardware and number of nodes, and supporting performance benchmarks results for the cluster.

1.3 Definitions

MPI: Message Passing Interface, a specification for implementing message passing parallel libraries for use in parallel programming. There are multiple MPI implementations, the two most common on Linux being LAM/MPI (Local Area Multicomputer) and MPICH (MPI CHameleon) .

PVM: Parallel Virtual Machine, the de-facto standard message passing parallel library. It predates the MPI specification and is used in many older parallel applications.

ZPL: An array-based parallel programming language intended to support engineering and scientific applications.

NFS: Network File System. The de-facto standard for exporting file systems for use on a network consisting of UNIX or Linux computers.

NIS: Network Information Services. NIS handles user authentication in a network environment and is commonly used in conjunction with NFS to decide when a user can access a shared file system.

RSH: Remote Shell. This program allows the execution of commands on a remote computer and is used to start parts of parallel programs by the PVM and MPI parallel libraries.

2 Technology Review and Design Decisions

Affordability, simplicity of construction and maintenance, and reasonable performance were the major factors considered in the design and construction of this Beowulf cluster. The design decisions that were available to this project and the final decisions made will now be discussed.

2.1 Local vs. Network Boot

Several different file system configurations were considered in the design of this Beowulf cluster. Network booting, local booting with the root file system served over NFS, and local booting and local file system with the /home partition and some configuration files served over NFS were all considered.

Network booting

In a network boot configuration, the nodes in the clusters do not have hard drives. Instead, a boot-rom is used on the network card to locate a bootp server and load the operating system image across the network from that server. The primary advantage of this method is that a group of computers with similar hardware, a single operating system image can be maintained. This simplifies updates and configuration changes to the nodes in the cluster. Another advantage of this method is that hard drives eventually fail. In a large cluster, the mean time between failures of the hard drives will lead to the failure of a hard drive somewhere in the cluster every few weeks. If the nodes do not have hard drives this problem is eliminated as the only hard drives are in the bootp server. The disadvantages are the need for network cards that support network booting and the

associated boot-roms.

Local booting with NFS root

In a local boot with NFS root configuration, each node boots the Linux kernel from a floppy, hard drive, or cdrom, and then mounts the root file system from the server. This method has the advantage of a single operating system image as in the network boot, but has the disadvantages of boot media that can fail (floppies and cdroms are especially prone to failure over extended periods of use) and needing a custom kernel and a complicated setup for the exported root file system.

Local booting and root with NFS /home

In the local boot and root with NFS /home configuration, each node has its own hard drive containing a complete Linux system. The node boots and loads programs from this drive. The server node exports the /home directory and several cluster-wide configuration files which each of the client nodes then mount over NFS. The primary advantage of this method is the simplicity of setting up the cluster as a nearly default Linux install can be used. In a configuration with varied surplus hardware, this method is also simpler than construction of a tagged network image for each computer. The disadvantages are the difficulties in upgrades and the potential for hard drive failures in nodes of the cluster.

Decisions made for this cluster

After considering the options, the decision was made to use the local boot and local file system with an NFS mounted /home. Since this project aims at simplifying the use of

surplus computers in a cluster, the network boot and local boot with NFS root methods both have complications. Many surplus computers are already equipped with networking hardware, but most of this network hardware does not currently support boot-rom chips and booting from the network. Even if it does support a boot-rom, locating or making a functional boot-rom will likely be beyond the skill level of the targeted end user and contribute an extra cost to building the cluster. The NFS root method is also probably beyond the target skill level as it involves the creation of root file system images and building custom Linux kernels that support NFS root file systems. The local boot and file system method has several other attractive features. Surplus or off-the-shelf computers usually come equipped with a functional hard drive, so this resource should not just be discarded. This allows an almost default Linux install to the hard drive, with only slight modifications made for mounting the /home partition across NFS and user authentication with NIS. Installing Linux on each node has the added bonus of the install having the opportunity to detect and configure the different types of hardware that are present in the different computers that will be used since not all the nodes are guaranteed to be identical. Surplus computers also tend to have limited memory (RAM), and in such cases, the hard drive provides valuable swap space (disk based virtual memory) to compensate for this limitation.

2.2 Overview of the NAS MG benchmark used to evaluate the cluster

The NASA Advanced Supercomputing (NAS) Parallel Benchmarks are a collection of eight benchmarks used to evaluate the performance of “highly parallel supercomputers” (http://www.nas.nasa.gov/Research/Software/swdescription.html#NPB). Among these eight benchmarks is the Multigrid Array (MG) benchmark. This particular benchmark

solves Poisson's equation (calculates how a charge distribution will attempt to reach equilibrium) over a three dimensional array. The NAS MG benchmark was packaged with ZPL, and is what was used to locate some bottlenecks of the Beowulf Cluster.

2.3 10Base-T vs. 100Base-T Networks

It was theorized that moving from a 10Base-T network to a 100Base-T network would bring great improvement in the Beowulf Cluster's performance. While great improvement was observed in the NAS MG benchmark using Class S and Class W data sets, the improvement observed was moderate for the larger, more computation intensive tasks – such as the NAS MG benchmark using Class A data set and the mirror resurfacing simulation program.

Readily observable from Figures 1 & 2 below is the substantial reduction in the time required to complete the NAS MG benchmark using the Class S data set (Figure 1) and using the Class W data set (Figure 2) on a 100Base-T network relative to the time required on a 10Base-T network. The Class S and Class W data sets are substantially smaller than the Class A data set or the set of data manipulated in the mirror resurfacing program. Hence, a greater percentage of the time required to complete the NAS MG benchmark program is devoted to communication between the cluster nodes. This translates to an increase in network traffic with the addition of each new cluster node. Moving from a 10Base-T network to a 100Base-T network drastically increases the speed of such network traffic, thus substantially decreasing the time required to complete the program.

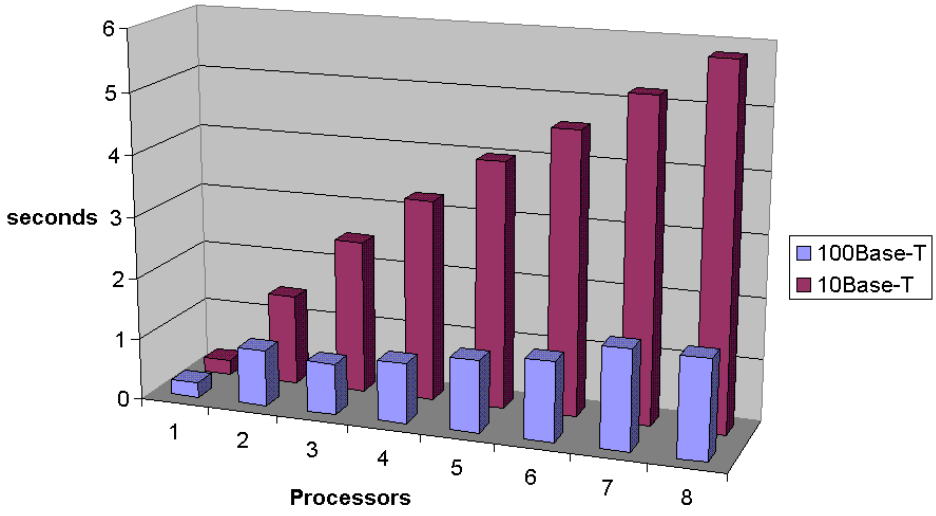


Figure 1 -- Comparison of the NAS MG benchmark using the Class S data set on 10Base-T and 100Base-T networks

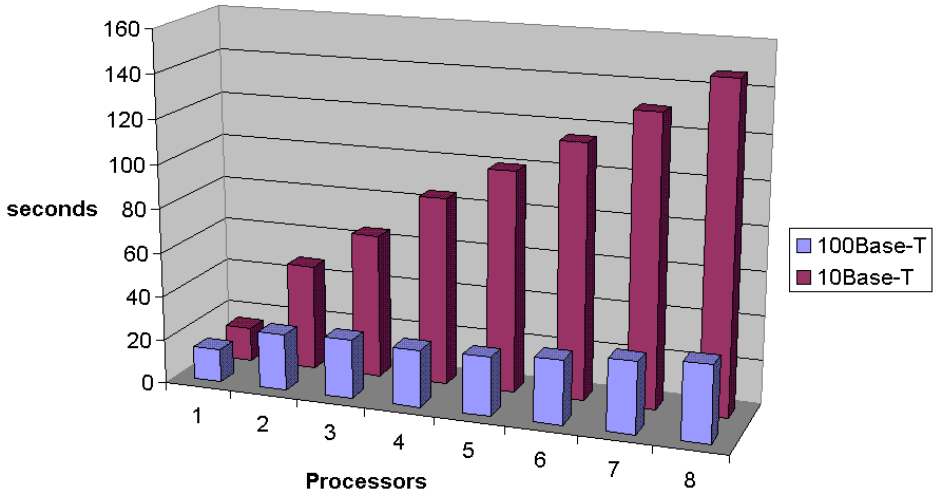


Figure 2 -- Comparison of the NAS MG benchmark using the Class W data set on 10Base-T and 100Base-T networks

A comparison of Figures 3 & 4, the performance of the mirror resurfacing simulation program on a 10Base-T network and a 100Base-T network, respectively, will show a moderate performance increase with the 100Base-T network. The mirror resurfacing simulation program, unlike the NAS MG benchmark program, outputs the time required for I/O and the actual computation time. Also observable in Figures 3 & 4 is the increase in time spent on I/O activities with the addition of each new cluster node.

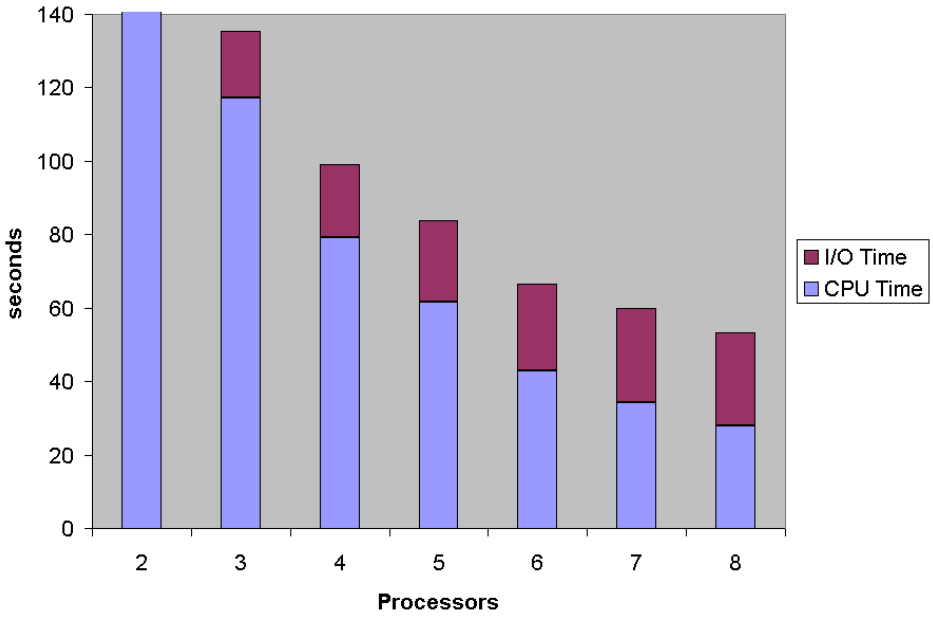


Figure 3 -- Time required to complete the mirror resurfacing simulation program on a 10Base-T network

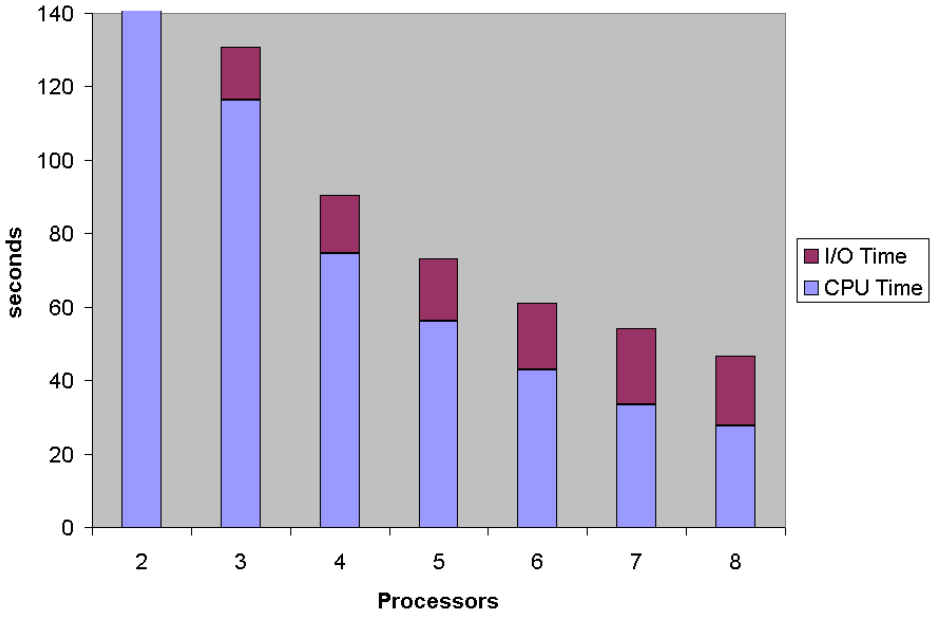


Figure 4 -- Time required to complete the mirror resurfacing simulation program on a 10Base-T network

Figure 5 shows a moderate performance increase of the NAS MG benchmark using a Class A data set. This performance increase is similar to that observed with the mirror resurfacing simulation program.

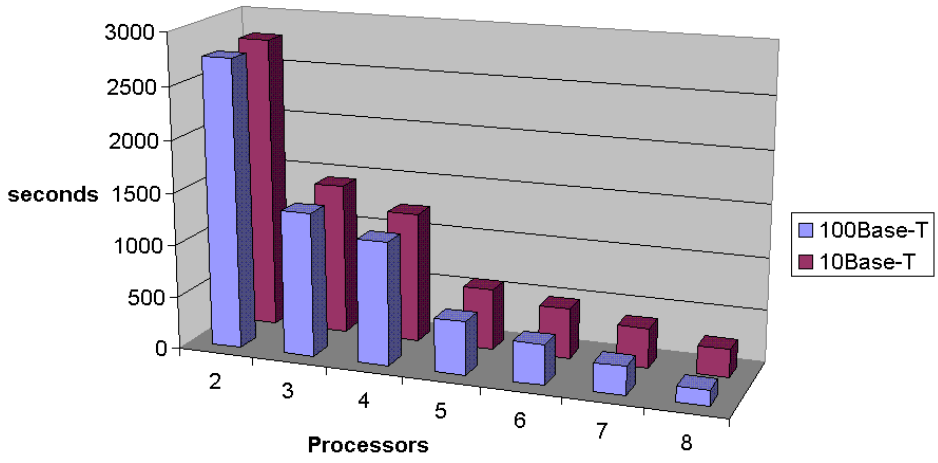


Figure 5 -- Comparison of the NAS MG benchmark using the Class A data set on 10Base-T and 100Base-T networks

2.4 Overview of the ZPL Parallel Programming Language

ZPL is described as follows on the ZPL main page:

> "ZPL is an array programming language designed from first principles for fast execution on both sequential and parallel computers. Because ZPL benefits from recent research in parallel compilation, it provides a convenient high-level programming medium for supercomputers with efficiency comparable to hand-coded message passing. Users with scientific computing experience can generally learn ZPL in a few hours. Those who have used MATLAB or Fortran 90 may already be acquainted with array programming style" (ZPL main page 2002).

This Beowulf cluster supports ZPL as one of the design requirements, and is able to utilize ZPL with the user's choice of the PVM, MPICH, or sequential libraries.

2.5 MPI vs. PVM Parallel Libraries

The MPI (Message Passing Interface) and PVM (Parallel Virtual Machine) libraries are the most common message passing parallel libraries in use today. PVM is the older of the two, and is the de-facto standard. PVM has a user shell that is used to configure the cluster interactively. MPI is the newer of the libraries and is backed by a formal industry standard. MPI is configured by a default configuration file, but the user can specify a different configuration file to use on the command line. There are several implementations of the MPI standard available under the Linux operating system. The two most common are LAM (Local Area Multicomputer) and MPICH (MPI CHameleon).

The MPI and PVM libraries both make use of the rsh program to connect to the nodes in the cluster and begin the execution of the parallel program on that node. The parallel libraries then manage the network communication between the processes on each node.

Figures 6 and 7 illustrate the results of the PVM and MPI benchmarks performed on the Beowulf cluster constructed. Figure 6 illustrates the PVM and MPI message passing overhead. Since the ‘S’ data class is a small data set, the predominant time factor is the communication between nodes. From these results, MPI has a lower communication overhead on the test cluster. Figure 7 involve more computations relative to network

communication and thus shows little difference between the performance of PVM and MPI. Though in the majority of the test results, MPI held a slight advantage. Figures 6 also shows increasing execution time as the number of processors involved increases (for the same amount of computations). This is due to the large overhead of network message passing in relation to the small data set that calculations are performed on. Figure 7 involves 512 times the number of calculations, which overwhelms the network overhead, and the benefits of the parallel computing come into play. The Class S data in Figure 6 is a 32x32x32 element data set, and the Class A data in Figure 7 is 256x256x256 element data set. From the results in Figures 6 and 7, the following guidelines for this cluster can be made: data sets of fewer than approximately 125,000 total elements will in general perform better on a single processor than on the cluster. Larger data set will reap the benefits of the parallel execution.

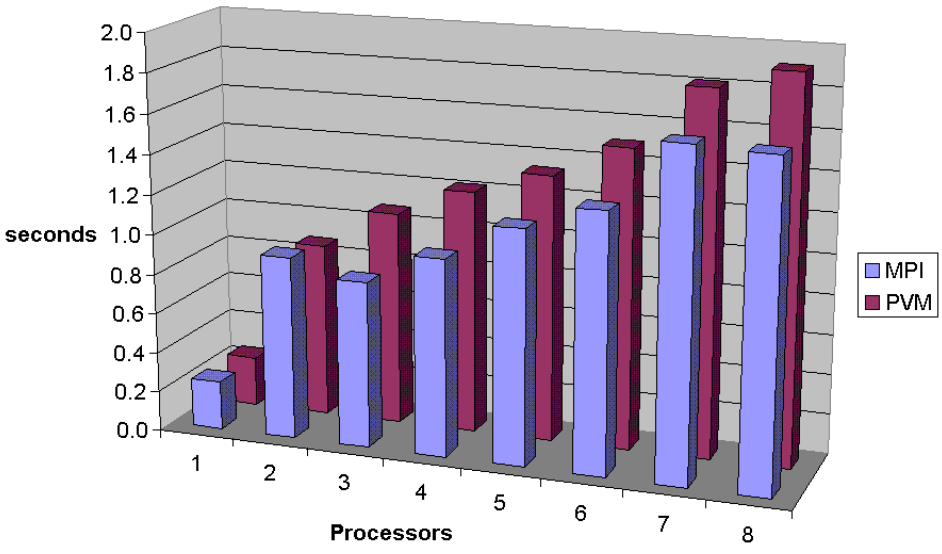


Figure 6 -- Performance comparison of NAS MG benchmark with the Class S data set using the MPI and PVM libraries

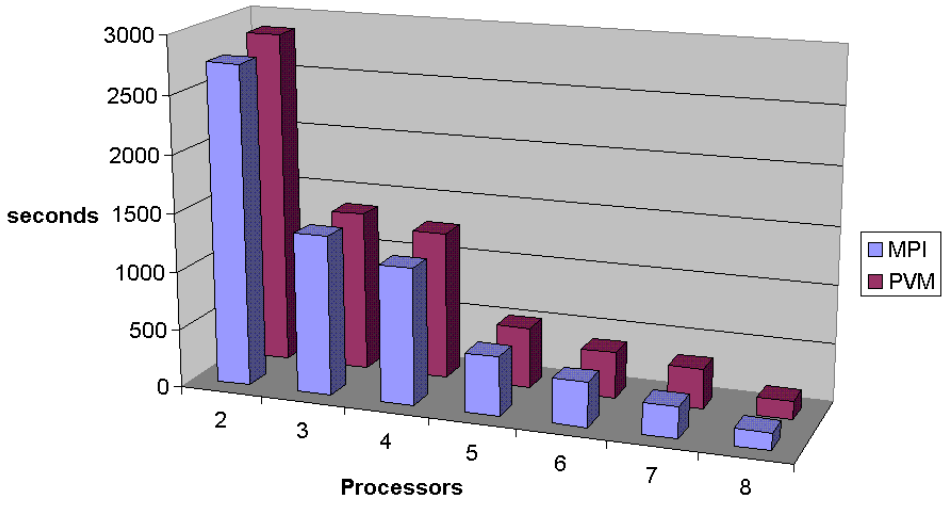


Figure 7 -- Performance comparison of NAS MG benchmark with the Class A data set using the MPI and PVM libraries

Figures 8 and 9 illustrate the performance of the MPI and PVM libraries in conjunction with the 100 inch mirror simulation program written in ZPL (see Appendix B). Since this simulation is a computation intensive task, the differences between MPI and PVM were not readily visible. However, another valuable suggestion is visible in Figures 8 and 9. As more processors are involved in the execution of the program, the execution time reduces, but the I/O time to reassemble the data and output the result file on the server node increases. The end user is advised to only output data to the final result file that is critical to the analysis of the results. If unnecessary data is collected and output, the network and disk overhead in gathering and writing this data will eventually cancel the performance benefits of the parallel execution.

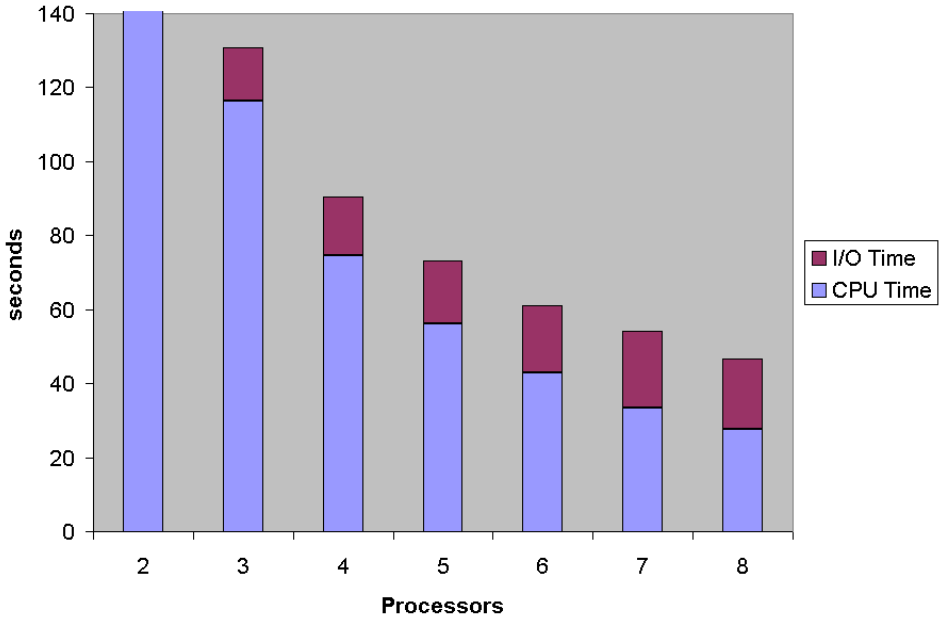


Figure 8 -- Performance of mirror resurfacing simulation program using the MPI library

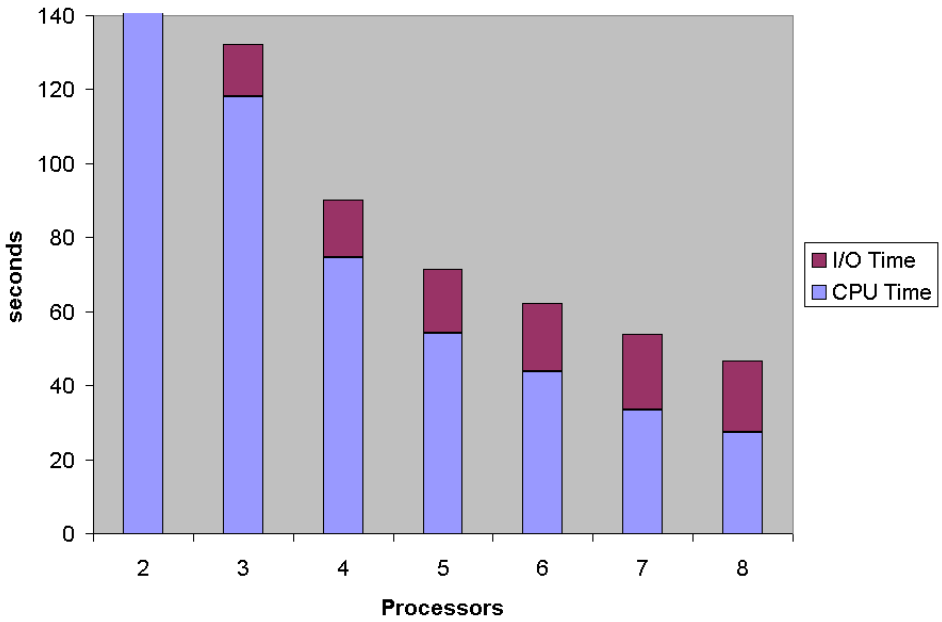


Figure 9 -- Performance of mirror resurfacing simulation program using the PVM library

3 Team Member Contributions and Finished Product

3.1 Michael Berg

My first contribution was a basic set of instructions to follow while installing Linux on the nodes. This allowed all of the nodes to have almost identical configurations for development and testing, and it made sure that the libraries and tools that we would need were available for use. Kevin then took over modifying this list as we made changes to the cluster.

My next contribution was obtaining and installing ZPL. ZPL comes in several variants. For example, under Linux there are ZPL downloads for use with the PVM library, the MPICH library, the LAM/MPI library, and for single computer (sequential) use. The documentation that comes with ZPL only describes how to install one of these variants, and we were interested in at least installing all of the PVM and MPI variants. So I compared all of the files in each download variant and found that there is a common ZPL core set of binaries and libraries that then compile ZPL code against separate directories for the parallel or sequential library being used. We were then able to safely install all of the ZPL variants that we were interested in, in order to facilitate easily switching between parallel libraries when compiling for testing purposes.

Once this was completed, I had to configure the PVM and MPICH parallel libraries so that ZPL could make use of them. This involved editing several configuration files for the libraries and setting shell environment variables in the user's login configuration files.

This step went smoothly except for some problems with the RSH configuration in the cluster that prevented PVM and MPI from successfully starting programs on the other nodes in the cluster. Once the team had solved the RSH problem, the parallel libraries were functioning properly and we could use ZPL in conjunction with all of the parallel libraries that we had installed.

My next contribution was porting Denis Oesch's "100 inch mirror coating" simulation (see Appendix A) from Matlab to ZPL so that it could be run on the cluster. The completed conversion from Matlab to ZPL is in Appendix B. This provided a real world test case to use in our performance benchmarks as well as providing a valuable example for the professors and graduate students at New Mexico Tech for converting existing Matlab simulation and data processing programs to ZPL.

My final contribution was the python and shell scripts I worked on to simplify the cluster configuration process for the end user (scripts contained in Appendix C). These scripts automate the generation of several system and parallel library configuration files and allow the user to easily switch ZPL between the available parallel libraries on the system. The scripts are also part of the installation/recovery procedure.

3.2 Kevin Fisher

At the onset of this project, most of the machines to be used as cluster nodes had a limited amount of RAM; others had absolutely no RAM. Dr. Scott Teare purchased many RAM DIMMS, and I installed this memory in the cluster nodes. Care was taken to

blow out any dust present in the memory sockets before installing the DIMMS. To ensure that all of the new DIMMS functioned properly, the program 'memtest86' was run on all machines for at least a twenty-four hour period. I assisted in this memory testing process.

Midway through the project, we concluded that the upgrade to RedHat Linux 7.2 was required for further advancement in the development of the Beowulf Cluster. To accomplish this task, I downloaded ISO image files of the RedHat 7.2 OS, burned these images to compact disc, installed the OS on every machine in the cluster, and set up an initial OS configuration that would facilitate the overall goals of the project. To ensure that working cluster configuration files under the old OS were not lost during the upgrade, I upgraded half of the machines to RedHat 7.2, stored backup copies of all relevant configuration files to these machines, then upgraded the other half of the machines.

Realizing that the final Beowulf Cluster product would include a recovery method that would instruct the user in repairing nodes on the cluster, I recorded all steps taken in the RedHat 7.2 installation process. This measure has ensured that no information vital to the OS installation process was forgotten when the actual creation of a recovery method took place.

To identify possible performance improvements in the Beowulf Cluster design, extensive

benchmark data had to be compiled. I was in charge of running all benchmark programs, compiling all benchmark data, and coming up with easy to read graphs of this data for inclusion in the final Beowulf Cluster user manual. Each benchmark program was run three times with the same parameters (data set, number of processors, etc.), and these three results were averaged to arrive at a final benchmark value.

I wrote many miscellaneous shell scripts to automate the benchmark process to a small degree. These scripts would run through each benchmark program, incrementing the number of processors utilized in the benchmark programs, and writing the output data from each program to separate and organized files. These scripts allowed for the running of many benchmarks in sequence without the need of a human operator to monitor the process and input commands, thus allowing the running of benchmarks overnight. The scripts also greatly reduced the probability of user error in setting up the benchmark parameters on the command line.

Near the end of the project, the Beowulf Cluster was moved from a 10Base-T network to a 100Base-T network for performance comparison purposes. I was in charge of this move, replacing the Beowulf Cluster's 20-port 10Base-T hub with one 8-port 100Base-T hub.

3.3 Anthony Montoya, Jr.

Hardware Evaluation

The initial phase of the project consisted of an evaluation of the hardware we received from our customer, Dr. Teare. I aided in the testing of the individual machines and the cluster's internal network. This included adding memory to the machines and testing the existing and upgraded memory in all of the machines. We also verified the working condition of the hub and its ports, as well as the existing and newly manufactured Cat5 cables.

Hardware Construction

I made and tested several Cat5 Cables that were used in the networking of the cluster. Because we were only provided with 5 working Cat5 cables, 6 additional cables were created to complete the network portion of the hardware evaluation.

Hardware Installation

I handled the purchase and assisted in the installation of a CD-RW to assist in completion of the Emergency System Recovery Plan and add to the system's backup and archiving capability.

Research Other Beowulf Clusters

Current designs of Beowulf clusters were researched with the goal of finding the best techniques to apply to our Beowulf cluster. Also, in this capacity, historical information and general background knowledge was sought for presentation and question aspects.

Person of Contact

I had the responsibility of communicating messages or changes to the project between the team, customer, advisor, and instructor for the duration of the project. The lines of communication between our group and the customer were the most important to keep strong and clear in order to ensure that we were progressing through the project in the right direction. I also was responsible for ensuring communications to our advisor Dr. Wedeward. We were required to report to our advisor on a weekly basis to report progress and seek advice if needed for the project. I also received and shared information that was passed to me from the instructor pertaining to changes in assignments or due dates, as well as any other modification in the class.

Documentation

I assisted in the documentation pertaining to the Beowulf cluster's configuration, operation, maintenance, User's Manual, as well as weekly and formal reports. Within this area, I also assisted in the creation of material used for the presentations throughout the year.

4 Summary and Recommendations

This project has produced a functional Beowulf cluster for Dr. Teare's use. A user's manual and series of scripts have also been produced. The scripts produced automate the configuration of the systems in the cluster and to configure the parallel libraries and ZPL for the user. These features lower the entry level to expanding the provided cluster or to building a new cluster with a similar configuration.

As a result of the benchmarks performed on the finished Beowulf cluster, the following recommendations are provided for areas of future improvement in the cluster configuration.

Memory improvements

The most beneficial upgrade that could be made to the cluster at this point is adding more RAM to the cluster nodes. Each node is currently equipped with 64MB of RAM and a 256MB swap partition. When running the 100 inch mirror coating simulation, S=165 (resulting in a 331x331 grid mapped to the mirror surface), the total memory in the nodes was over 90% used (according to the "top" program on the involved nodes). Adding more RAM to the nodes in the cluster will increase the limit on the maximum sized simulations that can be run. If the operating systems on the node computers are ever reinstalled, increasing the swap space to 512MB or higher would also raise this limit, although swap space is slower than RAM.

Network improvements

As was seen in Figures 1 and 2, the switch from a 10Base-T hub to a 100Base-T hub

significantly reduced the performance penalty of the network overhead. However, even with a 100Base-T hub, when results were being gathered to the server node or during periods of high inter-node communication when processing, the collision light on the hub would indicate a frequent network packet collisions. Packet collisions require the packets to be resent, wasting bandwidth and decreasing network performance. Moving to a 100Base-T switched network would eliminate the network inefficiencies due to packet collisions and allow simultaneous communication between different sets of nodes in the cluster, increasing total network throughput.

References

The Beowulf Project

http://www.beowulf.org/

Parallel-Processing-HOWTO

http://www.tldp.org/HOWTO/Parallel-Processing-HOWTO.html

Beowulf-HOWTO

http://www.tldp.org/HOWTO/Beowulf-HOWTO.html

ZPL main page 2002

http://www.cs.washington.edu/research/zpl/

NASA Advanced Supercomputing Software Descriptions (NAS Parallel Benchmarks)

http://www.nas.nasa.gov/Research/Software/swdescription.html#NPB

PVM resource list

http://www.csm.ornl.gov/pvm/pvm_home.html

PVM FAQ

http://www.netlib.org/pvm3/faq_html/node1.html

Appendices

Appendix A: 100 inch mirror coating simulation (Matlab)

```
% by Denis Oesch

clear all;
%setting all parameters
Ri = 63.5; %radius of inner track of filaments - centimeters

Ro = 127;  %radius of outer track of filaments - centimeters

M = 0.065; %mass per clip - grams

rho = 2.7; %density of the material - grams per cubic centiemter

H = 70.51675; %height of filament plane above the mirror's center -
centimeters

S = 25;    %"radius" of  the grid to cover the mirror - integer


%inner clips

ci = 4;      %number of clips per inside filament(0-6)

%inner clip positions - 1 if a clip is placed in that position, 0 if not

Cip = [0,1,1,1,1,0];


%outer clips

co = 5;       %number of clips per outside filament(0-6)

%outer clip positions - 1 if a clip is placed in that position, 0 if not

Cop = [1,1,1,1,1,0];


N = 12*ci+24*co;      %total number of clips used


%setting up filament tracks - positions of the clips on each filament
for later shifting

Cx = [2.0,1.2,0.4,-0.4,-1.2,-2.0];


%setting up the filaments positions


F = zeros(36,2);



%inner vectors to center of filaments in cartesian coordinates


for i = 1:1:12;
  F(i,1)=Ri*cos((pi/6)*(i-1)+pi/24);
```

```
  F(i,2)=Ri*sin((pi/6)*(i-1)+pi/24);
end

%vectors to center of outer filaments in cartesian coordinates


for j = 1:1:24;
  F(j+12,1)=Ro*cos((pi/12)*(j-1)+pi/16);
  F(j+12,2)=Ro*sin((pi/12)*(j-1)+pi/16);
end


%now we want to convert those centers of filaments into a line segment
%that will place the point sources along the filaments correctly

P = zeros(N,3);

%inner filaments
for n = 1:1:12
  no=1;
  for w = 1:1:6
    if Cip(w) == 1
                    P(ci*n-ci+no,1) = F(n,1)-Cx(w)*F(n,2)/Ri;
      P(ci*n-ci+no,2) = F(n,2)+Cx(w)*F(n,1)/Ri;
      no = no+1;
    end
  end
end


%outer filaments
for m = 1:1:24
  no=1;
  for w = 1:1:6
    if Cop(w) == 1
                    P((12*ci)+(co*m-co)+no,1) = F(m+12,1)-
                                Cx(w)*F(m+12,2)/Ro;
      P((12*ci)+(co*m-co)+no,2) = F(m+12,2)+Cx(w)*F(m+12,1)/Ro;
      no = no + 1;
    end
  end
```

```
end

clear('F');

% creating a plot of the clip positions above the mirror

figure(1);
for v=1:1:N;
plot(P(v,1),P(v,2),'*');
  hold on;
end

%setting up the position vectors to describe the mirror as coordinates
%since it will be a grid we need only assign points to a single row
%the x and y terms matching
XY = zeros(2*S+1,1);
XY2 = zeros(2*S+1,1);

for k = 1:1:(2*S+1)
  XY(k,1)=-128.27+(256.54/(2*S))*(k-1);
  XY2(k,1)=((XY(k,1))^2/(5130.8))-0.5*H;
end

%setting up an array to hold the position vectors
%for each point on the mirror grid

G = zeros(2*S+1,2*S+1,3);
D = zeros(2*S+1,2*S+1,2);

for q = 1:1:(2*S+1)
  G(q,:,1)=XY;
  G(:,q,2)=XY;
  D(q,:,1)=XY2;
  D(:,q,2)=XY2;
end

G(:,:,3)=sum(D,3);

clear('XY2');
clear('D');
```

```
%now we have an array, P, that lists the positions of the N point
%sources in vector form and another array, G, that lists the positions
%of the points on the grid placed across the mirror's surface

%now to convert all this position information into an array of distance
%information it will consist of N "plates" of arrays that are
%3-D vectors arranged in a 2*S+1 x 2*S+1 matrix

temp1 = zeros(2*S+1,3);
temp2 = zeros(2*S+1,2*S+1,3);
delta = zeros(2*S+1,2*S+1,N,3);

for t = 1:1:N
  for a = 1:1:2*S+1
    temp1(a,:) = P(t,:);
  end
  for b = 1:1:2*S+1
    temp2(b,:,:) = temp1;
  end
  delta(:,:,t,:) = G - temp2;
end

temp3 = delta.^2;

rsqrd = sum(temp3,4);

clear('temp1');
clear('temp2');
clear('temp3');
clear('P');
clear('delta');

%at this point we need to convert these distances over to a thickness
%contribution from each filament so we will have a new array of plates
%but this one will consist of N layers of the thickness contributed
%from each individual point source(clip) over the 2*S+1 x 2*S+1 grid
%covering the mirror

layers = zeros(2*S+1,2*S+1,N);
```

```
layers = ((5/4)*M/(4*pi*rho.*rsqrd));

clear('rsqrd');

%setting up the thickness on the mirror from this arrangement
Coating = zeros(2*S+1,2*S+1);
Coatingtemp = zeros(2*S+1,2*S+1);
Mirror = zeros(2*S+1,2*S+1);
temp4 = ones(2*S+1,2*S+1);
temp5 = zeros(2*S+1,2*S+1);


Coatingtemp = sum(layers,3);

%setting a limiting value to compensate for the square grid and
%the round mirror beyond the mirror's edge

Edge = Coatingtemp(S+1,2*S+1);

surface = zeros(2*S+1,2*S+1);
surface(S+1,:)=1;
surface(:,S+1)=1;

for e = 1:1:S
  for u = 1:1:S
    if (e-S-1)^2 + (u-S-1)^2 > S^2
      surface(e,u)=0;
      surface(2*S+2-e,u)=0;
      surface(e,2*S+2-u)=0;
      surface(2*S+2-e,2*S+2-u)=0;
    else
      surface(e,u)=1;
      surface(2*S+2-e,u)=1;
      surface(e,2*S+2-u)=1;
      surface(2*S+2-e,2*S+2-u)=1;
    end
  end
end
```

```
temp5 = Edge.*(temp4 - surface);

Coating = temp5 + Coatingtemp.*surface;

Mirror =(Coatingtemp + G(:,:,3)).*surface +(G(S+1,2*S+1,3)).*(temp4 -
surface);

%Plotting the resulting coating
figure(2);
mesh(XY,XY,10^8*Coating);
XLABEL('Mirror Position(cm)');
YLABEL('Mirror Position(cm)');
ZLABEL('Coating Thickness(angstroms)');
TITLE('Perfect run - no miss-fires');

% determining the range of thicknesses and other values as output
[A,B] = min(Coating(S+1,:));
[C,D] = max(Coating(S+1,:));

ci
co
Minimum = A*10^8
minradius = abs(26-B)*5.1308

Maximum = C*10^8
maxradius = abs(26-D)*5.1308

Var =(Maximum - Minimum)

Ave = 10^8*mean(mean(Coating))

% writting soem of the data to files
wk1write('e-glass.xls',G(:,S+1,3),2,2);
wk1write('e-mirror.xls',Mirror(:,S+1),2,2);
wk1write('position.xls',XY,2,2);

%curve fitting for use in zemax optical analysis
gc = polyfit(XY,G(:,S+1,3),7);
wk1write('glass cross-section curve fit.xls',gc);
mc = polyfit(XY,Mirror(:,S+1),7);
wk1write('mirror cross-section curve fit.xls',mc);
```

```
coatfit = polyfit(XY,Coating(:,S+1),7);
wk1write('coating cross-section curve fit.xls',coatfit);
```

Appendix B: 100 inch mirror coating simulation (ZPL + Matlab Display)

ZPL Computation Code

```
program mirror;

config var
          /* setting all parameters */
          output_filename : string = "data.m";
          dotiming : boolean = false;

          Ri : double = 63.5;  -- radius of inner track of filaments -
centimeters
          Ro : double = 127.0; -- radius of outer track of filaments -
centimeters
          M : double = 0.065;  -- mass per clip - grams
          rho : double = 2.7;  -- density of the material - grams per
cubic centiemter
          H : double = 70.51675;         -- height of filament plane
above the mirror's center - centimeters
          S : integer = 25;    -- "radius" of  the grid to cover the
mirror - integer

          /* which clips fire and which fail */
          Ci01 : boolean = true;
          Ci02 : boolean = true;
          Ci03 : boolean = true;
          Ci04 : boolean = true;
          Ci05 : boolean = true;
          Ci06 : boolean = true;
          Ci07 : boolean = true;
          Ci08 : boolean = true;
          Ci09 : boolean = true;
          Ci10 : boolean = true;
          Ci11 : boolean = true;
          Ci12 : boolean = true;

          Co01 : boolean = true;
          Co02 : boolean = true;
          Co03 : boolean = true;
          Co04 : boolean = true;
          Co05 : boolean = true;
          Co06 : boolean = true;
          Co07 : boolean = true;
          Co08 : boolean = true;
          Co09 : boolean = true;
          Co10 : boolean = true;
          Co11 : boolean = true;
          Co12 : boolean = true;
          Co13 : boolean = true;
          Co14 : boolean = true;
          Co15 : boolean = true;
          Co16 : boolean = true;
          Co17 : boolean = true;
          Co18 : boolean = true;
          Co19 : boolean = true;
          Co20 : boolean = true;
          Co21 : boolean = true;
          Co22 : boolean = true;
          Co23 : boolean = true;
          Co24 : boolean = true;

          /* inner clips */
```

```
          ci : integer = 4;    -- number of clips per inside
filament(1-6)
          -- inner clip positions - "true" if a clip is placed in that
position, "false" if not
          Cip1 : boolean = false;
          Cip2 : boolean = true;
          Cip3 : boolean = true;
          Cip4 : boolean = true;
          Cip5 : boolean = true;
          Cip6 : boolean = false;

          /* outer clips */
          co : integer = 5;    -- number of clips per outside
filament(1-6)
          -- outer clip positions - "true" if a clip is placed in that
position, "false" if not
          Cop1 : boolean = true;
          Cop2 : boolean = true;
          Cop3 : boolean = true;
          Cop4 : boolean = true;
          Cop5 : boolean = true;
          Cop6 : boolean = false;

          N : integer = (12 * ci) + (24 * co);      -- total number of
clips used

region
          R_XY    = [1..(2*S+1), 1..1];
          R_P     = [      *,     1..N,   *, 1..3];
          R_G     = [1..(2*S+1), 1..(2*S+1),   *, 1..3];
          R_D     = [1..(2*S+1), 1..(2*S+1),   *, 1..2];
          R_delta  = [1..(2*S+1), 1..(2*S+1), 1..N, 1..3];
          R_rsqrd  = [1..(2*S+1), 1..(2*S+1), 1..N, 1..1];
          R_mirror = [1..(2*S+1), 1..(2*S+1), 1..1, 1..1];

var
          output_file : file;
          timervalue : double;

          Cip : array [1..6] of boolean;
          Cop : array [1..6] of boolean;

          Ci  : array [1..12] of boolean;
          Co  : array [1..24] of boolean;

          Cx : array [1..6] of double;

          Fi : array [1..12, 1..2] of double;
          Fo : array [1..24, 1..2] of double;

          XY : [R_XY] double;

          P : [R_P] double;
          temp2 : [R_G] double;
          temp3 : [R_delta] double;
          delta : [R_delta] double;
          rsqrd : [R_rsqrd] double;

          layers  : [R_rsqrd] double;

          G : [R_G] double;
          D : [R_D] double;

          Edge   : [R_mirror] double;
```

```
            surface : [R_mirror] integer;
            temp5  : [R_mirror] double;
            temp6  : [R_mirror] double;
            temp7  : [R_mirror] double;
            Coating : [R_mirror] double;
            Coatingtemp : [R_mirror] double;
            Mirror  : [R_mirror] double;

            n  : integer;
            m  : integer;
            ni : integer;
            no : integer;
            wi : integer;
            wo : integer;
            t  : integer;
            b  : integer;

            temp : double;


procedure mirror ();
begin
            write (" S = %d":S, " (%dx":(2*S+1), "%d mirror
grid)\n":(2*S+1));

            ResetTimer();  /* Clear the timer for benchmarking or runtime
info */

            /* Open the output file for writing */
            output_file := open (output_filename, "w");

            Cx[1] := 2.0;
            Cx[2] := 1.2;
            Cx[3] := 0.4;
            Cx[4] := -0.4;
            Cx[5] := -1.2;
            Cx[6] := -2.0;


            /* Specify whether each set of clips fires or not */
            Ci[1]  := Ci01;
            Ci[2]  := Ci02;
            Ci[3]  := Ci03;
            Ci[4]  := Ci04;
            Ci[5]  := Ci05;
            Ci[6]  := Ci06;
            Ci[7]  := Ci07;
            Ci[8]  := Ci08;
            Ci[9]  := Ci09;
            Ci[10] := Ci10;
            Ci[11] := Ci11;
            Ci[12] := Ci12;

            Co[1]  := Co01;
            Co[2]  := Co02;
            Co[3]  := Co03;
            Co[4]  := Co04;
            Co[5]  := Co05;
            Co[6]  := Co06;
            Co[7]  := Co07;
            Co[8]  := Co08;
            Co[9]  := Co09;
            Co[10] := Co10;
            Co[11] := Co11;
```

```
          Co[12] := Co12;
          Co[13] := Co13;
          Co[14] := Co14;
          Co[15] := Co15;
          Co[16] := Co16;
          Co[17] := Co17;
          Co[18] := Co18;
          Co[19] := Co19;
          Co[20] := Co20;
          Co[21] := Co21;
          Co[22] := Co22;
          Co[23] := Co23;
          Co[24] := Co24;


          /* Initialize the Cip and Cop arrays from the config vars */
          Cip[1] := Cip1;
          Cip[2] := Cip2;
          Cip[3] := Cip3;
          Cip[4] := Cip4;
          Cip[5] := Cip5;
          Cip[6] := Cip6;

          Cop[1] := Cop1;
          Cop[2] := Cop2;
          Cop[3] := Cop3;
          Cop[4] := Cop4;
          Cop[5] := Cop5;
          Cop[6] := Cop6;


          /* vectors to center of inner filaments in cartesian
coordinates first */

          for n := 1 to 12 do
                    Fi[n,1] := Ri * cos ((M_PI/6) * (n - 1) +
(M_PI/24));
                    Fi[n,2] := Ri * sin ((M_PI/6) * (n - 1) +
(M_PI/24));
          end;


          /* vectors to center of outer filaments in cartesian
coordinates first */

          for m := 1 to 24 do
                    Fo[m,1] := Ro * cos ((M_PI/12) * (m - 1) +
(M_PI/16));
                    Fo[m,2] := Ro * sin ((M_PI/12) * (m - 1) +
(M_PI/16));
          end;


          /* now we want to convert those centers of filaments into a
line segment
            that will place the four or five point sources along the
filament */

          [R_P] P := 0;

          /* inner filaments */
          for n := 1 to 12 do
                    if (Ci[n] = true) then
                              ni := 1;
```

```
                                for wi := 1 to 6 do
                                        if (Cip[wi] = true) then
                                                [*, ci*n-ci+ni, *,
1] P := Fi[n,1] - (Cx[wi] * Fi[n,2] / Ri);
                                                [*, ci*n-ci+ni, *,
2] P := Fi[n,2] + (Cx[wi] * Fi[n,1] / Ri);
                                                ni := ni + 1;
                                        end;
                                end;
                        end;
                end;

                /* outer filaments */
                for m := 1 to 24 do
                        if (Co[m] = true) then
                                no := 1;
                                for wo := 1 to 6 do
                                        if (Cop[wo] = true) then
                                                [*, (12*ci)+(co*m-
co)+no, *, 1] P := Fo[m,1] - (Cx[wo] * Fo[m,2] / Ro);
                                                [*, (12*ci)+(co*m-
co)+no, *, 2] P := Fo[m,2] + (Cx[wo] * Fo[m,1] / Ro);
                                                no := no + 1;
                                        end;
                                end;
                        end;
                end;


                /* setting up the position vectors to describe the mirror as
coordinates
                  since it will be a grid we need only assign points a single
row
                  the x and y terms matching */


                /* setting up an array to hold the position vectors
                  for each point on the mirror grid */

                [1..(2*S+1), 1..(2*S+1), *, 1] G := -128.27 + ((256.54 /
(2*S)) * (Index2 - 1));
                [1..(2*S+1), 1..(2*S+1), *, 2] G := -128.27 + ((256.54 /
(2*S)) * (Index1 - 1));

                [1..(2*S+1), 1..(2*S+1), *, 1] D := ((G)^2 / (5130.8)) - (0.5
* H);
                [1..(2*S+1), 1..(2*S+1), *, 2] D := ((G)^2 / (5130.8)) - (0.5
* H);

                [1..(2*S+1), 1..(2*S+1), *, 3] G := +<< [R_D] D;


                /* This line if for simplifying output of Matlab compatable
results */
                [R_XY] XY := -128.27 + ((256.54 / (2*S)) * (Index1 - 1));


                /* at this point we need to convert these distances over to a
thickness
                  contribution from each filament so we will have a new array
of plates
                  but this one will consist of N layers of the thickness
contributed
                  from each individual point source(clip) over the 2*S+1 x
```

```
2*S+1 grid
            covering the mirror */

          for t := 1 to N do
                    [R_G] temp2 := >>[1, t, *, 1..3] P;

                    [1..(2*S+1), 1..(2*S+1), t, 1..3] delta :=
                              >>[1..(2*S+1), 1..(2*S+1), 1, 1..3] (G -
temp2);
          end;


          [R_delta] temp3 := (delta)^2;

          [1..(2*S+1), 1..(2*S+1), 1..N, 1] rsqrd := +<< [R_delta]
temp3;

          /* You must use 5.0 and 4.0 instead of 5 and 4, or else some
integer math
            is done which is incorrect and produces the wrong results
*/
          [R_rsqrd] layers := ( ((5.0 / 4.0) * M) / (4.0 * M_PI * rho *
rsqrd) );


          /* setting up the thickness on the mirror from this
arrangement */

          [1..(2*S+1), 1..(2*S+1), 1, 1..1] Coatingtemp := +<<
[R_rsqrd] layers;


          /* setting a limiting value to compensate for the square grid
beyond
            the round mirror's edge */

          /* Flood Coatingtemp[S+1, 2*S+1, 1, 1] into Edge */
          [R_mirror] Edge := >>[(S+1), (2*S+1), 1..1, 1..1]
Coatingtemp;

          /* This makes "surface" a circle filled with ones and zeros
outside the circle */
          [R_mirror] surface := !( ((Index1 - (S+1))^2) + ((Index2 -
(S+1))^2) > S^2 );

          [R_mirror] temp5 := Edge * (1.0 - surface);
          [R_mirror] Coating := temp5 + Coatingtemp * surface;


          /* Flood G[S+1, 2*S+1, 1, 3] into temp6 */
          [R_mirror] temp6 := >>[1..(2*S+1), 1..(2*S+1), 1, 3] G;

          /* Flood G[S+1, 2*S+1, 1, 3] into temp7 */
          [R_mirror] temp7 := >>[(S+1), (2*S+1), 1, 3] G;

          [R_mirror] Mirror := (Coatingtemp + temp6) * surface + (temp7
* (1.0 - surface));


          /* All the computational work is now done, get the total
runtime
            Do it here since the file I/O below shouldn't be in the
            computational runtime benchmarks */
```

```
          timervalue := CheckTimer();
          if (dotiming) then
                    write (" Computation Time in seconds\t= %12.4f\n":
timervalue);
          end;


          /* Write out all the data needed in Matlab for
            post-analysys and graphing the results */

          ResetTimer();  /* reset the timer for I/O timing */

          /* Clear all prior data in Matlab */
          write (output_file, "clear all;\n\n");

          write (output_file, "fprintf('Loading data file, please
wait... ');\n\n");

          write (output_file, "S  = ",  S, ";\n");
          write (output_file, "ci = ", ci, ";\n");
          write (output_file, "co = ", co, ";\n");
          write (output_file, "N  = ",  N, ";\n");
          write (output_file, "\n");

          [R_P] write (output_file, "P = [", "%.20f\t": P, "];\n\n");

          [R_XY] write (output_file, "XY = [", "%.20f\t": XY,
"];\n\n");

          [R_mirror] write (output_file, "Coating = [", "%.20f\t":
(10^8 * Coating), "];\n\n");
          /* Coating is scaled up by 10^8 in the data file, scale it
back down */
          write (output_file, "Coating = Coating / 10^8;\n\n");

          [R_mirror] write (output_file, "Mirror  = [", "%.20f\t":
Mirror, "];\n\n");

          /* When printing more than 2 dimensions, ZPL does not put any
kind of break
            between the higher dimensions, so I have to print it like
this */
          write (output_file, "G = zeros (", (2*S+1), ",", (2*S+1),
",", "3);\n");
          for n := 1 to 3 do
                    [1..(2*S+1), 1..(2*S+1), 1, n] write (output_file,
                              "G(:,:,%d) = [": n, "%.20f\t": G,
"];\n\n");
          end;

          write (output_file, "fprintf('done.\\n');\n");

          /* Close the output file */
          close (output_file);

          /* All the computational work is now done, get the total file
I/O time */
          timervalue := CheckTimer();
          if (dotiming) then
                    write (" File I/O Time in seconds\t= %12.4f\n":
timervalue);
          end;
end;
```

Matlab Display Code

```
% You must run the appropriate data output file from the ZPL simulation
% before running this file.

% The data file contains results from the ZPL code's run on the Beowulf
Cluster
% S, ci, ci, N, P, XY, G, Mirror, and Coating are all dumped to this
file


% Creating a plot of the clip positions above the mirror

figure(1);
for v=1:1:N;
plot(P(v,1),P(v,2),'*');
  hold on;
end
axis('square');
xlabel('Mirror Position (cm)');
ylabel('Mirror Position (cm)');
title('Clip Firing Pattern');


%Plotting the resulting coating

figure(2);
mesh(XY,XY,10^8*Coating);
view(2);
axis('square');
colorbar('vert');
xlabel('Mirror Position (cm)');
ylabel('Mirror Position (cm)');
zlabel('Coating Thickness (Angstroms)');
title('Coating Thickness (Angstroms)');

figure(3);
mesh(XY,XY,10^8*Coating);
axis('auto');
xlabel('Mirror Position (cm)');
ylabel('Mirror Position (cm)');
zlabel('Coating Thickness (Angstroms)');
%title('Perfect run - no miss-fires');


% determining the range of thicknesses and other values as output
[A,B] = min(Coating(S+1,:));
[C,D] = max(Coating(S+1,:));

ci
co
Minimum = A*10^8
minradius = abs(26-B)*5.1308

Maximum = C*10^8
maxradius = abs(26-D)*5.1308

Var = (Maximum - Minimum)

Ave = 10^8*mean(mean(Coating))


% writing some of the data to files
```

```
wk1write('e-glass.xls',G(:,S+1,3),2,2);
wk1write('e-mirror.xls',Mirror(:,S+1),2,2);
wk1write('position.xls',XY,2,2);

% curve fitting for use in zemax optical analysis
gc = polyfit(XY,G(:,S+1,3),7);
wk1write('glass cross-section curve fit.xls',gc);
mc = polyfit(XY,Mirror(:,S+1),7);
wk1write('mirror cross-section curve fit.xls',mc);
coatfit = polyfit(XY,Coating(:,S+1),7);
wk1write('coating cross-section curve fit.xls',coatfit);
```

Appendix C: Parallel library and system configuration scripts

Python auto-configuration script

The following script (parallel_config.py) is a python program that should be placed in a .parallelrc directory in the user's account. This program handles writing shell files that contain the necessary environment variables to use ZPL and the parallel libraries. The shell wrapper that follows in the next section allows the functions in they Python program to be called from the users shell. In order to support other shells such as csh (C-shell), follow the examples in this script where the Bourne shell data is exported to files and do the same for csh equivalents.

```
#!/usr/bin/python

import sys
import os
import re

user_home_dir = os.environ['HOME']
user_parallel_setting_dir = os.path.join (user_home_dir, '.parallelrc')

# The following are the available ZPLCOMMLAYERs for the
# supported ZPLTARGET platforms listed on the zpl website.
# This dictionary will require periodic updates.

valid_zpl_settings = { }
# Workstation Settings
valid_zpl_settings['alpha-osf'] = ['seq']
valid_zpl_settings['hppa-hpux'] = ['seq']
valid_zpl_settings['mips-irix'] = ['seq', 'pvm']
valid_zpl_settings['powerpc-aix'] = ['seq']
valid_zpl_settings['sparc-solaris'] = ['seq', 'mpi-mpich', 'pvm']
valid_zpl_settings['x86-freebsd'] = ['seq']
valid_zpl_settings['x86-linux'] = ['seq', 'mpi-lam', 'mpi-mpich',
'pvm']

# Parallel System Settings
valid_zpl_settings['enterprise'] = ['seq', 'mpi']
valid_zpl_settings['origin'] = ['seq', 'mpi', 'pvm']

# Platform Settings
valid_zpl_settings['sp2'] = ['seq', 'mpi']
valid_zpl_settings['t3e'] = ['seq', 'mpi', 'pvm', 'shmem']


# Convert 32-bit integer form of an IPv4 address to dotted-quad form
def int_to_ip4(ip4_addr_num):
    octet = (((ip4_addr_num & 0xff000000) >> 24), \
             ((ip4_addr_num & 0x00ff0000) >> 16), \
             ((ip4_addr_num & 0x0000ff00) >> 8),  \
             ((ip4_addr_num & 0x000000ff)))

    ip4_addr_str = '%s.%s.%s.%s' % octet
```

```
    return ip4_addr_str
# end of int_to_ip4


# Convert dotted-quad form of an IPv4 address to 32-bit integer form
def ip4_to_int(ip4_addr_str):
    octet = re.split('\.', ip4_addr_str)
    ip4_addr_num = (long(octet[0]) << 24) + (long(octet[1]) << 16) + \
                   (long(octet[2]) << 8)  + (long(octet[3]))

    return ip4_addr_num
# end of ip4_to_int


def pvm_setup ():
    # Common PVM install locations, extend this list as needed.
    common_pvm_root_dirs = ['/usr/share/pvm3', '/usr/lib/pvm3',
'/opt/pvm3', \
                            '/usr/local/pvm3', '/usr/local/lib/pvm3']
    #common_pvm_root_dirs = []
    valid_pvm_rsh_programs = ['/usr/bin/rsh', '/usr/bin/ssh', \
                              '/usr/local/bin/rsh',
'/usr/local/bin/ssh']

    pvm_root = ''
    pvmgetarch_path = ''

    print '-------------------------------------------------------'
    print ' Configuring the PVM parallel library                  '
    print '-------------------------------------------------------'

    while not (os.path.isdir(pvm_root) and
os.path.isfile(pvmgetarch_program)):
        pvm_root = ''
        pvm_arch = ''
        pvmgetarch_program = ''
        default_pvm_root = 'none'

        possible_pvm_root_dirs = []
        for directory in common_pvm_root_dirs:
            if os.path.isdir(directory):
                possible_pvm_root_dirs.append(directory)

        print 'The following possible PVM root directories were found:'
        for directory in possible_pvm_root_dirs:
            print '\t%s' % directory

        print ''

        if len(possible_pvm_root_dirs) > 0:
            default_pvm_root = possible_pvm_root_dirs[0]

        # The character sequences at the end of the following two lines
        # suppress a newline from print and remove the newline from
readline
        print 'Which PVM root directory do you wish to use?'
```

```
        print '[default %s]: ' % default_pvm_root,;
        pvm_root = sys.stdin.readline()[:-1]
        print ''

        if (len(pvm_root) == 0) and (len(possible_pvm_root_dirs) > 0):
            pvm_root = possible_pvm_root_dirs[0]

        pvmgetarch_program = os.path.join(pvm_root, 'lib/pvmgetarch')
        if os.path.isfile(pvmgetarch_program):
            pvm_arch = '`%s`' % pvmgetarch_program
        else:
            print 'Could not find file %s, let\'s try this again...\n'
\
                  % pvmgetarch_program


    pvm_rsh = ''
    while not os.path.isfile(pvm_rsh):
        pvm_rsh = ''
        default_pvm_rsh = 'none'

        possible_pvm_rsh_programs = []
        for program in valid_pvm_rsh_programs:
            if os.path.isfile(program):
                possible_pvm_rsh_programs.append(program)

        print 'The following possible PVM RSH programs were found:'
        for program in possible_pvm_rsh_programs:
            print '\t%s' % program
        print ''

        if len(possible_pvm_rsh_programs) > 0:
            default_pvm_rsh = possible_pvm_rsh_programs[0]

        # The character sequences at the end of the following two lines
        # suppress a newline from print and remove the newline from
readline
        print 'Which PVM RSH program do you wish to use?'
        print '[default %s]: ' % default_pvm_rsh,;
        pvm_rsh = sys.stdin.readline()[:-1]
        print ''

        if (len(pvm_rsh) == 0):
            pvm_rsh = possible_pvm_rsh_programs[0]

    #print 'PVM_ROOT=%s' % pvm_root
    #print 'PVM_ARCH=%s' % pvm_arch

    pvm_sh_file = open(os.path.join(user_parallel_setting_dir,
'pvm.sh'), 'w')
    pvm_sh_file.write('export PVM_ROOT=%s\n' % pvm_root)
    pvm_sh_file.write('export PVM_ARCH=%s\n' % pvm_arch)
    pvm_sh_file.write('export PVM_RSH=%s\n'  % pvm_rsh)
    pvm_sh_file.write('export PATH=$PATH:%s\n' % os.path.join(pvm_root,
'bin', pvm_arch))
# end of pvm_setup
```

```
def zpl_setup_zplhome():
    common_zpl_home_dirs = ['/opt/zpl/zplhome', '/opt/zplhome', \
                            '/usr/local/zpl/zplhome',
'/usr/local/zplhome']
    zpl_home = ''

    while not (os.path.isdir(zpl_home)):
        zpl_home = ''
        default_zpl_home = 'none'

        possible_zpl_home_dirs = []
        for directory in common_zpl_home_dirs:
            if os.path.isdir(directory):
                possible_zpl_home_dirs.append(directory)

        print 'The following possible ZPL HOME directories were found:'
        for directory in possible_zpl_home_dirs:
            print '\t%s' % directory

        print ''

        if len(possible_zpl_home_dirs) > 0:
            default_zpl_home = possible_zpl_home_dirs[0]

        # The character sequences at the end of the following two lines
        # suppress a newline from print and remove the newline from
readline
        print 'Which ZPL HOME directory do you wish to use?'
        print '[default %s]: ' % default_zpl_home,;
        zpl_home = sys.stdin.readline()[:-1]
        print ''

        if (len(zpl_home) == 0) and (len(possible_zpl_home_dirs) > 0):
            zpl_home = possible_zpl_home_dirs[0]

    return zpl_home
#end of zpl_setup_zplhome


def zpl_setup_zpltarget():
    zpl_target = ''
    default_zpl_target = 'x86-linux'

    while not zpl_target in valid_zpl_settings.keys():
        zpl_target = ''
        print 'The choices for ZPLTARGET are:'
        for target in valid_zpl_settings.keys():
            print '\t%s' % target
        print ''

        print 'Which ZPLTARGET is appropriate for this configuration?'
        print '[default %s]: ' % default_zpl_target,;
        zpl_target = sys.stdin.readline()[:-1]
        print ''

        if (len(zpl_target) == 0):
```

```
            zpl_target = default_zpl_target

    return zpl_target
# end of zpl_setup_zpltarget


def zpl_setup_zplcommlayer(zpl_target):
    zpl_commlayer = ''

    while not zpl_commlayer in valid_zpl_settings[zpl_target]:
        zpl_commlayer = ''
        print 'ZPL can use the following libraries on %s:' % zpl_target
        for commlayer in valid_zpl_settings[zpl_target]:
            print '\t%s' % commlayer
        print ''

        print 'NOTE: the default on most platforms is seq
(sequential),'
        print 'but you will almost certainly want to choose one of the'
        print 'parallel libraries (such as mpi or pvm) in order to
make'
        print 'use of ZPL\'s parallel execution.'
        print ''

        print 'Which parallel library should ZPL use?'
        print '[default %s]: ' % valid_zpl_settings[zpl_target][0],;
        zpl_commlayer = sys.stdin.readline()[:-1]
        print ''

        if (len(zpl_commlayer) == 0):
            zpl_commlayer = valid_zpl_settings[zpl_target][0]

    return zpl_commlayer
# end of zpl_setup_zplcommlayer


def zpl_setup():
    print '-------------------------------------------------------'
    print ' Configuring the ZPL parallel programming language     '
    print '-------------------------------------------------------'

    zpl_home = zpl_setup_zplhome()
    zpl_target = zpl_setup_zpltarget()
    zpl_commlayer = zpl_setup_zplcommlayer(zpl_target)

    zpl_bin = os.path.join(zpl_home, 'bin', zpl_target)

    #print 'ZPLHOME=%s' % zpl_home
    #print 'ZPLTARGET=%s' % zpl_target
    #print 'ZPLCOMMLAYER=%s' % zpl_commlayer

    zplcommlayer_sh_file = open(os.path.join(user_parallel_setting_dir,
'zplcommlayer.sh'), 'w')
    zplcommlayer_sh_file.write('export ZPLCOMMLAYER=%s\n' %
zpl_commlayer)
    zplcommlayer_sh_file.close()
```

```
    zpl_sh_file = open(os.path.join(user_parallel_setting_dir,
'zpl.sh'), 'w')
    zpl_sh_file.write('export ZPLHOME=%s\n' % zpl_home)
    zpl_sh_file.write('export ZPLTARGET=%s\n' % zpl_target)
    zpl_sh_file.write('. ~/.parallelrc/zplcommlayer.sh\n')
    zpl_sh_file.write('export PATH=$PATH:%s\n' % zpl_bin)
    zpl_sh_file.close()

# end of zpl_setup


def generate_cluster_files():
    default_node_count = 2
    default_cpu_per_node = 1
    default_domain_name = 'clusternet'
    default_node_base_name = 'node'
    default_ip_base_addr = '172.16.0.1'

    print '-------------------------------------------------------'
    print ' Generating system configuration files                 '
    print '-------------------------------------------------------'

    print 'What domain name should this cluster use?'
    print '[default %s]: ' % default_domain_name,;
    domain_name = sys.stdin.readline()[:-1]
    print ''
    if (len(domain_name) == 0):
        domain_name = default_domain_name

    print 'What IP address should the cluster start at?'
    print '[default %s]: ' % default_ip_base_addr,;
    ip_base_addr = sys.stdin.readline()[:-1]
    print ''
    if (len(ip_base_addr) == 0):
        ip_base_addr = default_ip_base_addr

    print 'How many nodes are in the cluster?'
    print '[default %s]: ' % default_node_count,;
    node_count_str = sys.stdin.readline()[:-1]
    print ''
    if (len(node_count_str) == 0):
        node_count = default_node_count
    else:
        node_count = int(node_count_str)

    print 'How many CPUs (processors) are in each node?'
    print '(choose the lowest number that applies for all nodes)'
    print '[default %s]: ' % default_cpu_per_node,;
    cpu_per_node_str = sys.stdin.readline()[:-1]
    print ''
    if (len(cpu_per_node_str) == 0):
        cpu_per_node = default_cpu_per_node
    else:
        cpu_per_node = int(cpu_per_node_str)

    print 'What base name should be used for the node names?'
    print '[default %s]: ' % default_node_base_name,;
```

```
    node_base_name = sys.stdin.readline()[:-1]
    print ''
    if (len(node_base_name) == 0):
        node_base_name = default_node_base_name


    node_list = []
    ip_base_addr_num = ip4_to_int(ip_base_addr)
    for i in range(node_count):
        node_name = '%s%03d' % (node_base_name, i+1)
        node_ip = int_to_ip4(ip_base_addr_num + i)
        node = [node_ip, node_name]
        node_list.append(node)


    # output /etc/hosts
    file = open('hosts', 'w')
    file.write('# Do not remove the following line, or various
programs\n')
    file.write('# that require network functionality will fail.\n')
    file.write('127.0.0.1\tlocalhost.localdomain\tlocalhost\n')
    for node in node_list:
        file.write('%s\t%s.%s\t%s\n' % (node[0], node[1], domain_name,
node[1]))
    file.close()

    # output /etc/hosts.equiv
    file = open('hosts.equiv', 'w')
    for node in node_list:
        file.write('%s.%s\n' % (node[1], domain_name))
    file.close()

    # output machines.LINUX
    file = open('machines.LINUX', 'w')
    file.write('# Change this file to contain the machines that you
want to use\n')
    file.write('# to run MPI jobs on.  The format is one host name per
line, with either\n')
    file.write('#    hostname\n')
    file.write('# or \n')
    file.write('#    hostname:n\n')
    file.write('# where n is the number of processors in an SMP.  The
hostname should\n')
    file.write('# be the same as the result from the command
"hostname"\n')
    for node in node_list:
        if (cpu_per_node > 1):
            file.write('%s:%d\n' % (node[1], cpu_per_node))
        else:
            file.write('%s\n' % (node[1]))
    file.close()

# end of generate_cluster_files



if (sys.argv[1] == 'pvm'):
```

```
    pvm_setup()

if (sys.argv[1] == 'zpl_setup'):
    zpl_setup()

if (sys.argv[1] == 'zpl_lib'):
    zpl_target = os.environ['ZPLTARGET']
    if zpl_target in valid_zpl_settings.keys():
        zpl_commlayer = zpl_setup_zplcommlayer(zpl_target)
        zplcommlayer_sh_file =
open(os.path.join(user_parallel_setting_dir, 'zplcommlayer.sh'), 'w')
        zplcommlayer_sh_file.write('export ZPLCOMMLAYER=%s\n' %
zpl_commlayer)
        zplcommlayer_sh_file.close()
    else:
        print 'ZPL environment is not correctly configured.\n'
        print 'You must run \"zpl --setup\" before using ZPL\n'


if (sys.argv[1] == 'gen_cluster_files'):
    generate_cluster_files()
```

Bourne Shell wrapper for the Python configuration script

This script is a wrapper around the Python script above. It provides shell function that handle calling the Python script with the necessary arguments and then sourcing the resulting files to activate the changes. This file should be placed in the .parallel directory of the user's directory and then sourced from their .bashrc file. Similar scripts can be written for other shells such as csh (C-shell).

```
if [ -f ~/.parallelrc/zpl.sh ]; then
    . ~/.parallelrc/zpl.sh
else
    echo "ZPL had not been configured yet!"
    echo "You must run \"zpl --setup\" before using ZPL"
    echo ""
fi

if [ -f ~/.parallelrc/pvm.sh ]; then
    . ~/.parallelrc/pvm.sh
else
    echo "PVM had not been configured yet!"
    echo "You must run \"pvm_setup\" before using the PVM libraries"
    echo ""
fi


if [ -f ~/.parallelrc/mpi-mpich.sh ]; then
    . ~/.parallelrc/mpi-mpich.sh
fi

if [ -f ~/.parallelrc/mpi-lam.sh ]; then
```

```
    . ~/.parallelrc/mpi-lam.sh
fi


parallel_script_dir=~/.parallelrc

function zpl
{
    case "$1" in
          --setup)
              ${parallel_script_dir}/parallel_config.py zpl_setup
              source ~/.parallelrc/zpl.sh
              ;;
          --change-lib)
              ${parallel_script_dir}/parallel_config.py zpl_lib
              source ~/.parallelrc/zplcommlayer.sh
              ;;
          *)
              echo "Usage: zpl {--setup|--change-lib}"
              ;;
    esac
}


function pvm_setup
{
    ${parallel_script_dir}/parallel_config.py pvm
    source ~/.parallelrc/pvm.sh
}


function beowulf_cluster
{
    case "$1" in
          --setup)
              zpl --setup
              pvm_setup
              beowulf_cluster --gen-files
              ;;
          --gen-files)
              ${parallel_script_dir}/parallel_config.py
gen_cluster_files
              ;;
          *)
              echo "Usage: beowulf_cluster {--setup|--gen-files}"
              ;;
    esac
}
```

Appendix D: Beowulf Cluster User's Manual

Following this page is the User's Manual for the produced cluster.

Thunderstorm Data Analysis Progress
Exercises in Matlab with an Emphasis in Digital Signal Processing

A Thesis Presented to the Faculty of the
Electrical Engineering Department
of
New Mexico Tech
by

G. Martin
C. Reiten
S. Schuyler

In Partial Fulfillment of the requirements for the course
EE-482, Senior Design Project II
May 1, 2002

Abstract

The storm activity in clouds strips charges from air molecules and eventually generates a powerful electric field. This field is detected by an instrument called an electric field meter (EFM) which ascends into the storm and telemeters the data to ground where it is recorded. The scope of the senior design project is to extract the direction and magnitude of the electric field vector from the existing data.

Over the past 9 months we have worked to achieve just this. Different types of filtering and transforms were used to separate out relevant components. We have written an interactive Matlab program that analyzes flight data, allowing the user to select time ranges and phase angles. The output of this program provides 9 comprehensive plots that show the user the raw data, the horizontal and vertical components of both the electric and magnetic field relative to the instrument, and the strength of the magnetic field as the instrument ascends.

Acknowledgements

Ken Eack, NMT Physics Department
Set up receiver and decoder for EFM experiments.

Tim Hamlin, NMT Physics Department
Maintained computers, provided lab area for EFM experiments, and acted as advisor for software questions.

Paul Krehbiel, NMT Physics Department
Acted as technical advisor and director of project.

William Rison, NMT EE Department
Contributed his time and knowledge to assist in using Matlab tools.

W. David Rust, NSSL
Furnished EFM and Labview program for experiments. Invaluable resource of knowledge concerning project's history.

Bill Winn, NMT Physics Department
Provided documentation of past work on project.

Table of Contents

List of Figures

Chapter 1: A Brief History of Thunderstorm Data Analysis as Relates to Electric Field Meter Data Collection

Project Description:

Thunderstorm data analysis has been going for many years. This project came to us as the continuation of an older project intended to broaden the knowledge base of electric fields in thunderclouds to help gain an understanding of lightning. An instrument called an electric field meter was attached to a helium filled balloon and released into a cloud where it collected and transmitted important data regarding a mercury switch, the earth's magnetic field, and the electric field of the thunderstorm. This data was given to us as dimensionless lists of numbers indicating different parameters in each storm. The goal of our project was to assess this data, and from it, extract the 3-dimensional electric field vector at the instrument.

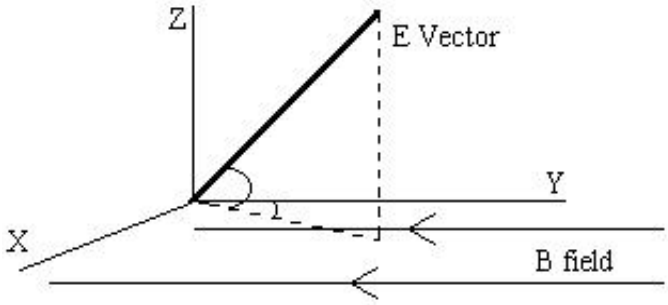


Figure 1: 3-D electric field vector

This was accomplished through several forms of data analysis including filtering, interpolating, demodulating, and a vast array of digital signal processing tools. Our final product is an interactive Matlab program that allows the user to select the flight and section of data they wish to study. It not only returns the 3-dimensional vector of the electric field in the cloud, but also displays other useful data in graphical format that is indicative of lightning strikes.

Definitions and Acronyms:

B-field – Earth's magnetic field as seen by the Hall Effect sensor in the EFM.

E-field – storm cloud's electric field as seen by the current induced across the conductive spheres of the EFM.

EFM - Electric Field Meter, the instrument used to collect the data for our project. Illustrated on page 9.

FIR – finite impulse response filter. Common digital signal filtering technique.

FFT - fast Fourier transform. Common digital signal spectrum analysis.

Hall Effect – The Hall effect occurs when the charge carriers moving through a material experience a deflection because of an applied magnetic field. This deflection results in a measurable potential difference across the side of the material that is transverse to the magnetic field and the current direction.

Hg switch – mercury switch. In one position, mercury flows to the end of a small tube where it touches two contacts, closing a switch and indicating a digital '1.' In the opposite position, the switch is open, indicating a digital '0.' This is used to determine up/down orientation of the EFM.

NOAA – National Oceanic and Atmospheric Administration, for more information visit http://www.nws.noaa.gov

NSSL – National Severe Storms Laboratories, for more information visit http://www.nssl.noaa.gov

Literature Review:

Books:

Applications of Linear Algebra, 2nd Edition

Chris Rorres and Howard Anton

Data Reduction and Error Analysis for the Physical Sciences

Philip R. Bevington

Digital Signal Processing: A Computer-Based Approach

Sanjit K. Mitra

Digital Signal Processing Using Matlab V. 4: A Bookware Companion Problems Book.

Vinay K. Ingle and John G. Proakis

Papers:

"Evaluation of a Standardized Sine Wave Fit Algorithm"

Peter Handel

"Thunderstorm on July 16, 1975, Over Langmuir Laboratory: A Case Study," Journal of Geophysical Research, June 20, 1978

William P. Winn, C.B. Moore, C.R. Holmes, and L.G. Byerley III

"Electric Field Structure in an Active Part of a Small, Isolated Thundercloud," Journal of Geophysical Research February 20, 1981

W.P. Winn, C.B. Moore, C.R. Holmes

History:

This project originated in the seventies. An instrument called a balloon-borne electric field meter (shown below) was attached to a helium filled balloon and released into a thunderstorm. Two conductive spheres rotated about a shaft as the device traveled through the clouds. As they rotated, the electric field component (E-field) perpendicular to this axis produced equal and opposite charges on the two spheres. The rotation caused the charges to vary sinusoidally which caused a small current. This current was detected by a charge amplifier. Inside one of the spheres was a mercury switch that indicated up/down orientation. Data was also gathered from a Hall Effect sensor, which detected the B-field. These devices have been used by several universities as well as the NSSL to collect and compile data.

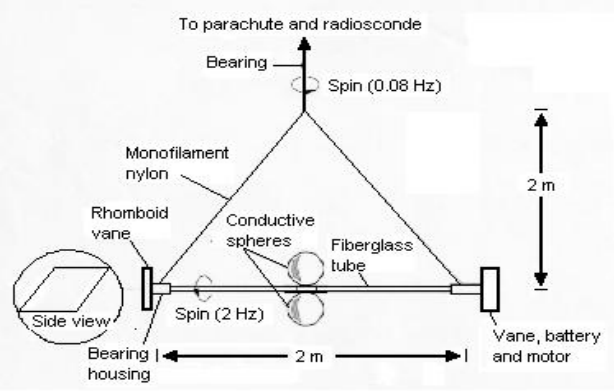


Figure 2: Electric field meter and rotational rates

Chapter 2: Past Findings on Electric Field Meter Data

For some years, our customer, Dr. Paul Krehbiel, and a team of physicists have been collecting data from thunderstorms with an electric field meter. Over the course of a thunderstorm, they recorded orientation of the instrument, the earth's magnetic field relative to the instrument, and the electric field relative to the instrument.

Our job is to analyze the data collected from the thunderstorms. Dr. Krehbiel wants a more accurate interpretation of the electric field. While the magnitude of the electric field has been extracted from the data, the direction remains a mystery. Using the magnetic field data as a reference, we hope to determine the three-dimensional electric field vector.

This project makes extensive use of digital signal processing. Our raw data are just numbers. At time t, the other data has values of x, y, and z. If we plot this data over a certain time interval and connect our points, we get waveforms.

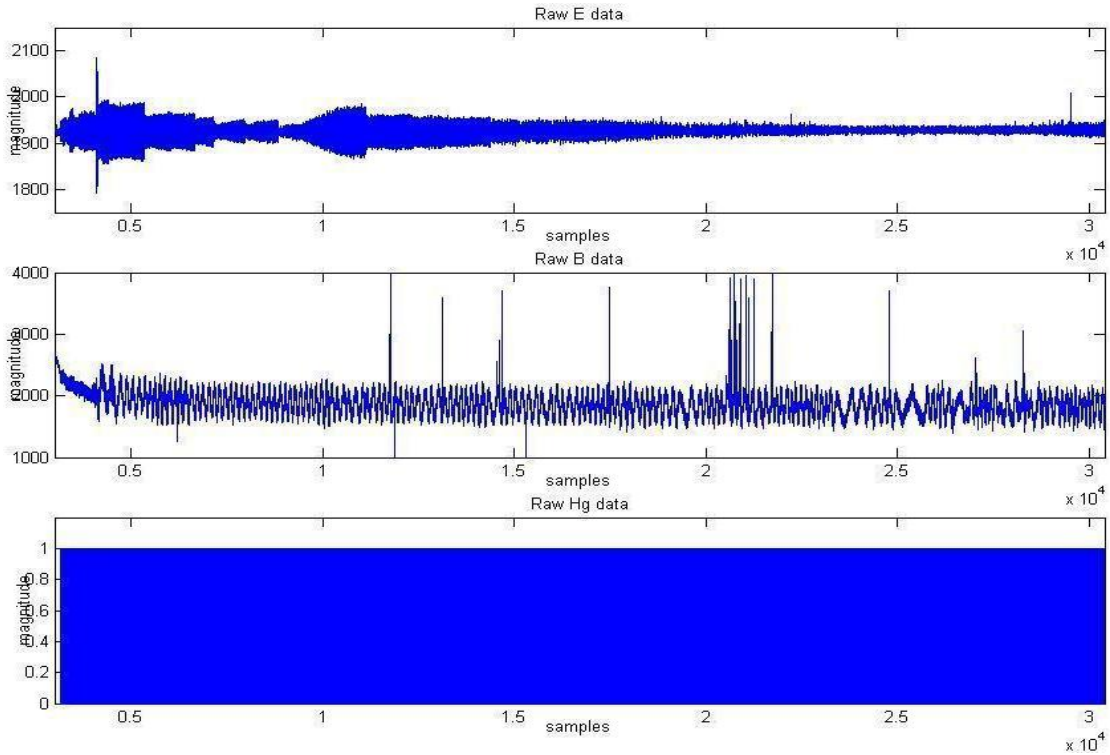


Figure 3: Waveforms produced by original data

There are data sets from several storms and over 40,000 individual data points per storm. It isn't practical to plot this by hand, so we are automating the process using Matlab. These waveforms show patterns of data behavior. We want to filter out different components of the waves such as high frequencies and low frequencies. We also want to isolate the fundamental frequencies of the data sets to compare phase shifts. Our goal is to detect patterns or correlations among the different data types. There is erroneous data that must be removed, such as data spikes. We will have to find where the actual data begins since the meter records on its way up to the storm as well as in it.

The physicists involved have analyzed and begun to filter the data, and, from this work, have gained significant understanding of the electric fields in a cloud. The vertical component, as well as the magnitude of the horizontal component, of the E-field has been extracted from the data by hand. This has been a slow process, and thus much of the analysis is still not done.

Ultimately we hoped to automate the analysis, so the knowledge buried in the data would be revealed more readily. This would greatly aid the NOAA and the NSSL in their life-protecting efforts by increasing their understanding of electric fields in thunderclouds.

Chapter 3: Steps Toward Development of Data Analysis Automation

Seth Schuyler: Menuing and User Interface

The initial idea was to all do the same work on the data independently. None of us were proficient with Matlab and we thought this division of labor would allow all of us the opportunity to gain the skills necessary for the completion of this project. As it turned out, however, each of us continued to find and explore new aspects of Matlab.

I bumped into the Filter and Design and Analysis Tool, and shortly there after found the Signal Processing Tool. Through a great deal of experimentation and study, I managed to isolate the fundamental frequency of the mercury switch data. Chris's work with the Z transform was a tremendous help. At this point I hit a limit. The data, being digital, does not yield sine waves. Instead the filtered signal was an obfuscated triangle wave. When this same filter was applied to the Hall effect sensor the output was interesting but non-intuitive. We later solved this problem by interpolating the data.

I invested a great deal of effort into a now-abandoned task. Initially, I created a program, written in the C programming language, which removed data spikes from both the B-field and E-field data sets. The E-field spikes, which were formally believed to be data anomalies, were closely inspected and determined to be lightening. Because of this, a decision was made to allow them to remain. We do not know the cause of the B-field data spikes, but they do not correspond with spikes in the E-field. Thus, it is impossible to use the B-field as a trigger for filtering E-field data anomalies. These discoveries were side effects of code rewrites and experiments.

As the data analysis portion of our project began to take form, the product became difficult to interact with. I worked with Ginny to build a menuing system for the data sets. This made starting a project a snap, but working within it was still cumbersome. To this

end, I programmed the subplots of any given panel to be interlaced. Now when one graph is re-scaled, the other graphs are automatically re-scaled to match. Our customer found this to be a very useful improvement.

I was reassigned to modify the data acquisition software to work with the new generation of EFM; however, it was later discovered that the new hardware would not be completed in time. I then assisted Ginny with making preparations to the old EFM for a forthcoming upgrade, but then that too was taken out of our domain. Finally, I worked with Chris on the FIR to phase shift the mercury switch. (See figure 9).

Chris Reiten: Matlab Coding and Digital Signal Processing

From the beginning of the project we all knew that there was going to be a lot of digital signal processing and electricity and magnetism involved. Fortunately I took Digital Signal Processing in the fall semester. This helped me to better understand some of the tools that would be very important in determining the magnitude and direction of the E-field vector.

An important primary step in extracting the E-field vector from the data is determining the relationship between the Hg switch data, the E-field data, and the B-field data. We all figured that this relationship is primarily a matter of the phase difference between the individual data components. Using the Fast Fourier Transform (FFT) command in Matlab, I was able to see all of the major frequencies that were present in each of the signals. As Seth mentioned above, the Hg switch contained several frequencies, the fundamental frequency and its harmonics. The harmonics are what make the signal a square wave instead of a sinusoid, so we decided to filter them out and isolate the fundamental frequency. (See Figure 4). I wrote a few lines of Matlab code that

would determine the peak value of the FFT and use that as the center for a FIR bandpass filter.

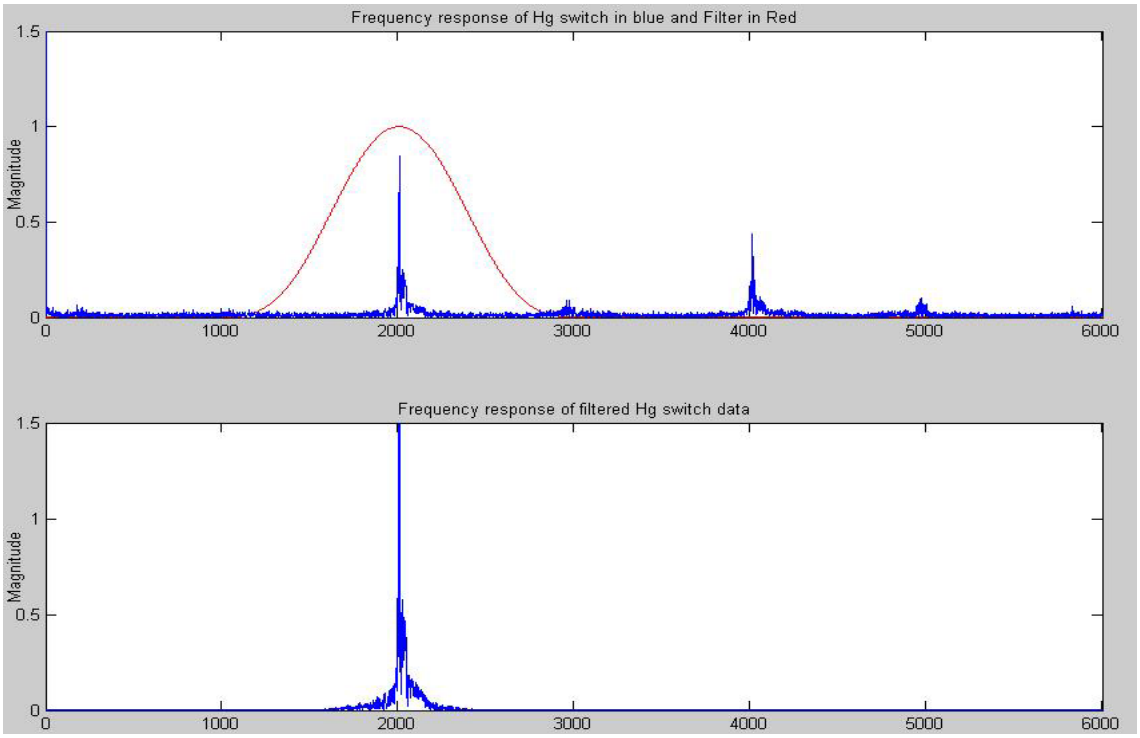


Figure 4: FFT and filtering of Hg

This effectively isolated the fundamental frequency. Although the fundamental frequency was isolated, the result was a sinusoid that was not very clean and had shifts in the amplitude where the original Hg switch data stuck. Because these amplitude shifts can affect some of the data processing, I wrote a for loop that re-filtered the data and got rid of many of those shifts. The loop took the sinusoid and any value that was greater than zero was turned into a '1' and any value less than zero became a '0'. This made a much nicer square wave without the data anomalies. This new square wave was filtered using the same technique as the original Hg switch data and a cleaner sinusoid was extracted. I then interpolated the filtered data using the interpft command to smooth the

sinusoid. For every existing point in the sinusoid, three new points were inserted. Figure5 shows the original square wave, the filtered Hg wave and the interpolated Hg wave.

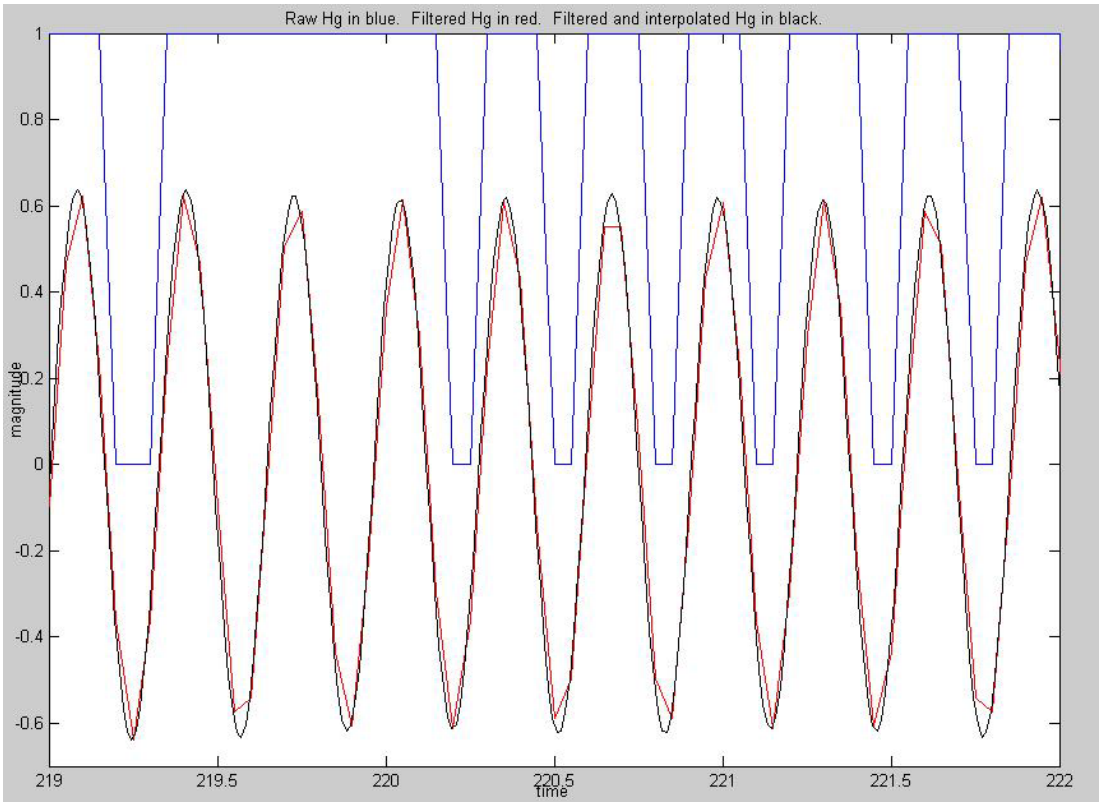


Figure 5: Original, filtered, and interpolated Hg data

The phase difference between the B-field data and the Hg switch data gives us the direction that the instrument is facing. This is very important for the final step of determining the direction of the E-field. Taking the FFT of the B-field data revealed two primary frequencies. (See Figure 6).

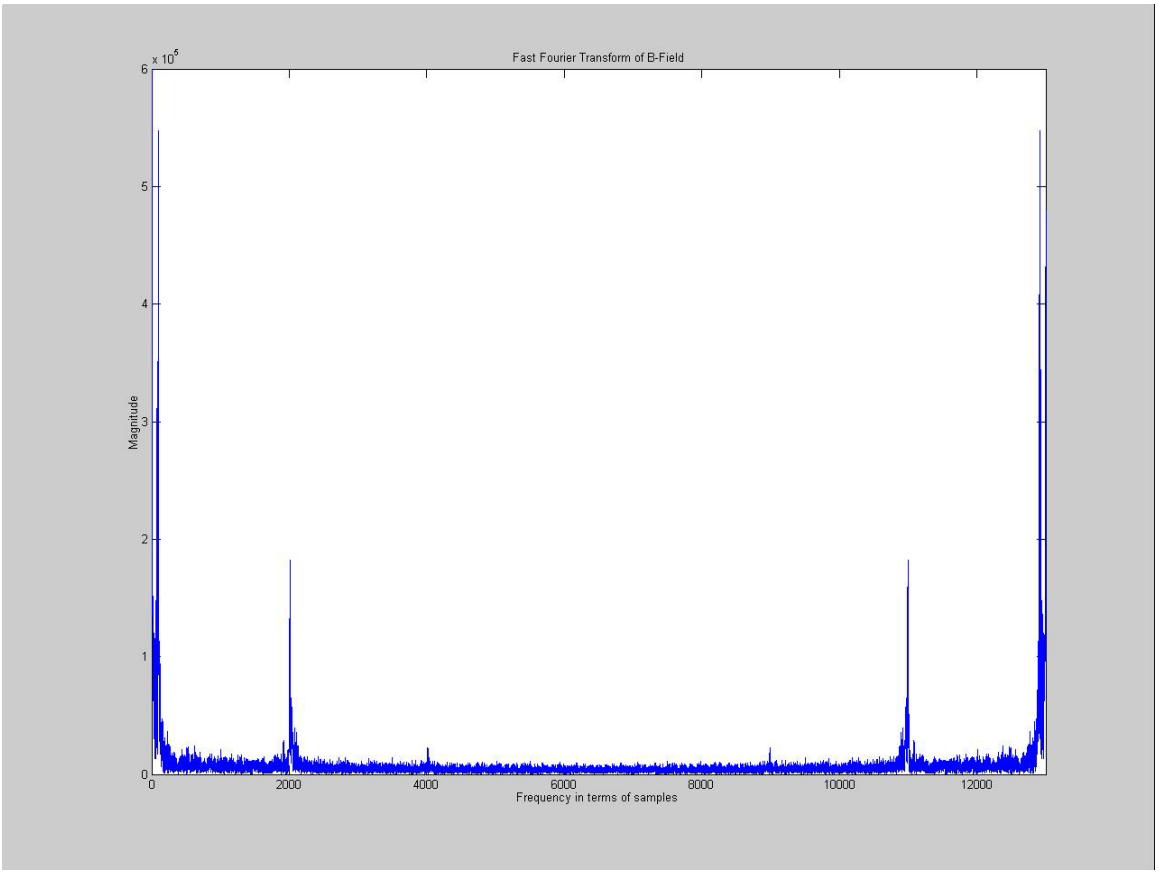


Figure 6: FFT of B-field

The higher frequency was due to the rotation of the spheres about the horizontal axis and the lower frequency was due to the precession about the vertical axis. For determining the phase between the B-field and the Hg switch, the higher frequency of the B-field had to be isolated. This was easy to implement since the filter for the Hg switch was already created. The filtered B-field data was then interpolated to smooth it out and keep it on the same time scale as the filtered and interpolated Hg switch data.

The phase relationship between the Hg switch and the B-field is found using quadrature demodulation. This method takes a sinusoid and then shifts it 90 degrees. (See figure 7).

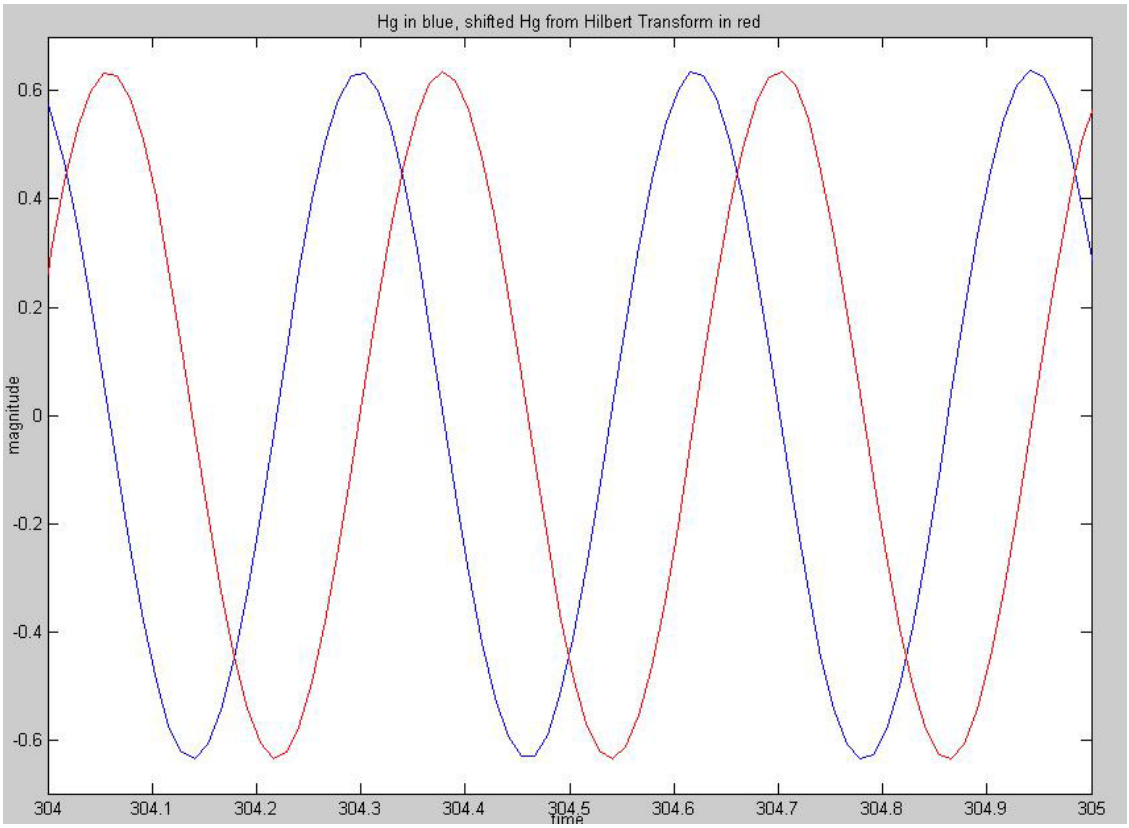


Figure 7: Hilbert transform

Each of these sinusoids is then multiplied by the sinusoid that contains the phase shift. This results in two different sinusoids that are twice the frequency of the original sinusoids plus a component that is the sine or cosine of the phase angle. Using a lowpass filter to get rid of the high frequency sinusoid, the sine and the cosine of the phase angle remains. By dividing the sine by the cosine and taking the arc tangent of the result, the exact phase angle is extracted. This is far more accurate than just using a cosine or a sine to determine the phase. See figure 8 for the lowpass filter that isolates the phase

component of the quadrature demodulation. Also see chapter 4 for more detail regarding this quadrature demodulation technique.

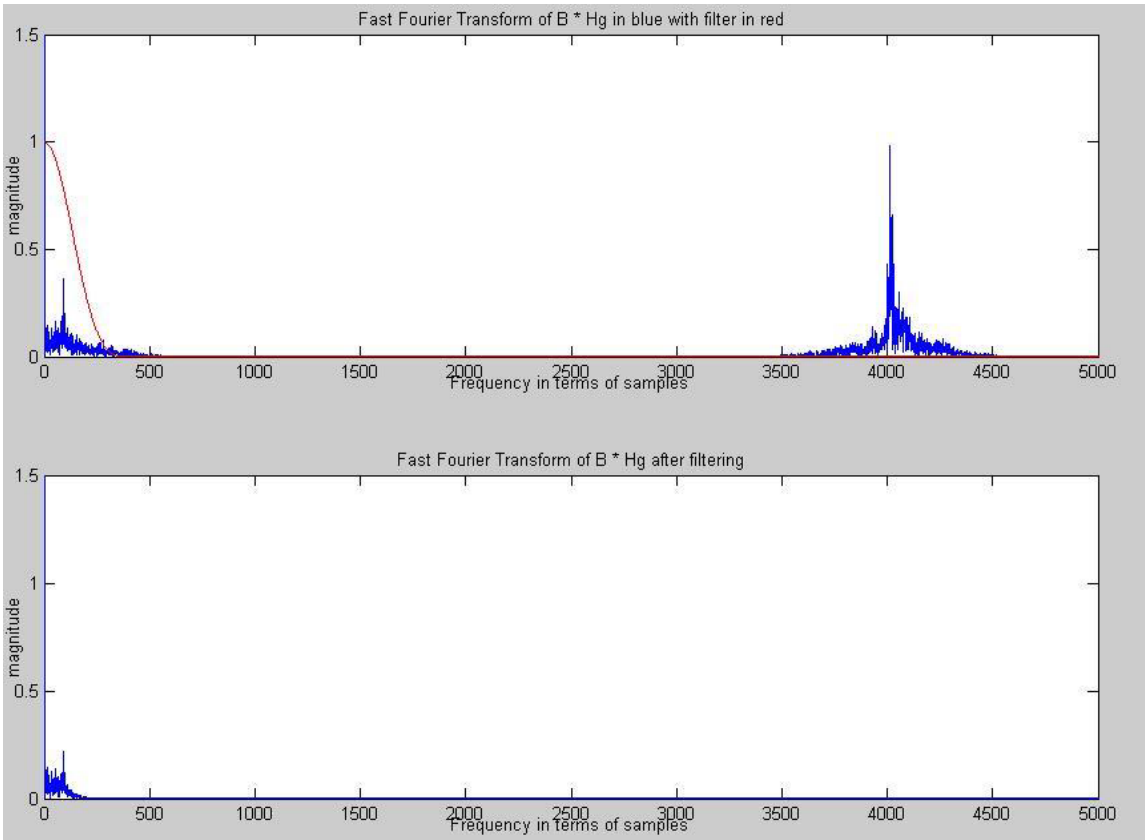
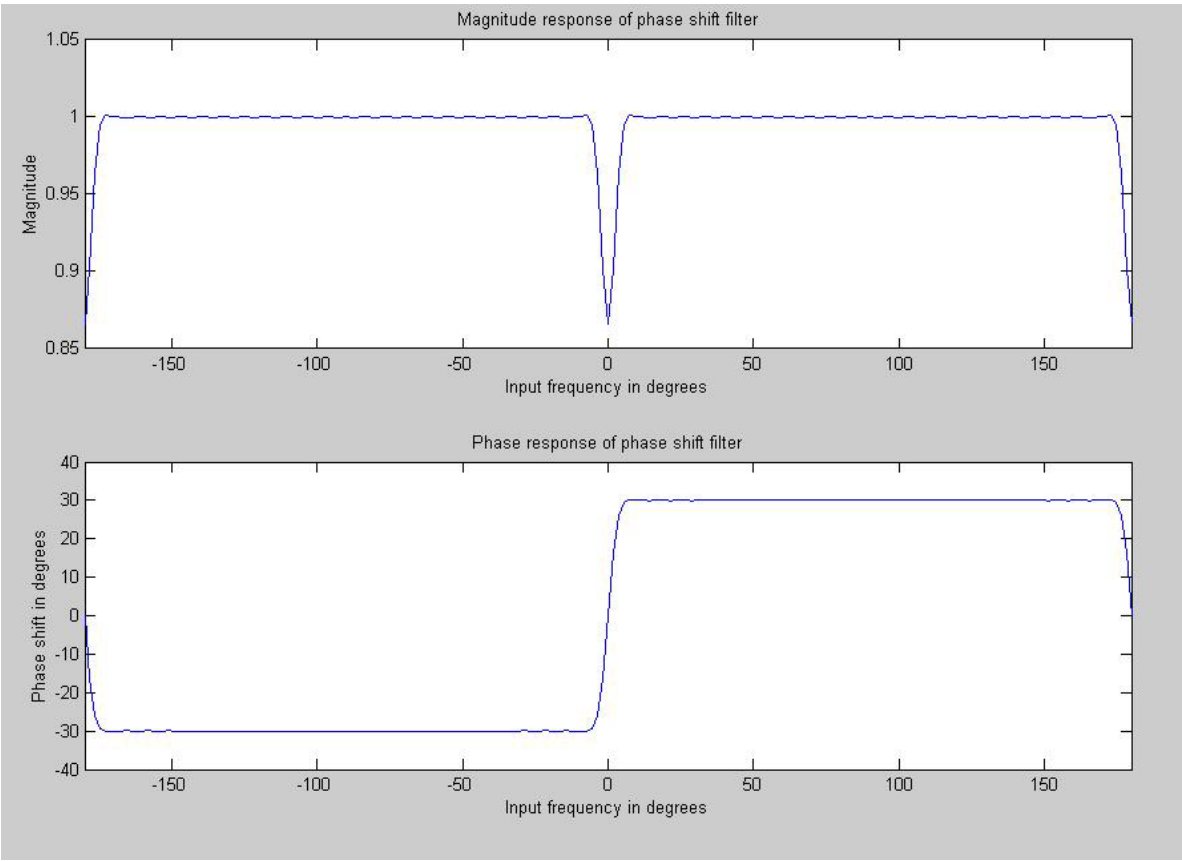


Figure 8: Lowpass filter for quadrature demodulation

In the program, the quadrature demodulation technique was implemented using the Hilbert transform function in Matlab. The Hilbert function shifts a sinusoid by 90 degrees. The filtered and interpolated Hg switch data was shifted using the Hilbert function, giving the two separate cosine and sine waves. This was then used to demodulate both the B-field data and the E-field data.

When demodulating the E-field, I had to keep the two-phase components separate. This told us the vertical component and the horizontal component of the E-field. The cosine component of the quadrature demodulation told us the horizontal E-field and the sine component of the quadrature demodulation told us the vertical

component of the E-field. Because the Hg switch changes before it is straight up and down it is not an exact description of when the spheres were vertical. To account for this we had to design a FIR filter that would pass all frequencies and have a constant phase shift. This would shift the Hg switch data by a certain phase angle to match up with the



B-field data. (See figure 9).

Figure 9: Constant phase shift FIR filter

With the help of Dr. Rison and Seth, I was able to create the filter in Matlab. The filter is set up so that the user can input a certain phase in degrees and the Hg switch data will be shifted by that amount. When the unshifted Hg data was used to demodulate the E-field horizontal and vertical components, the ripple from the horizontal component could be seen in the vertical component. When the Hg data was shifted, the ripple was gone,

effectively isolating the vertical and horizontal components. See figure 11 for the individual components.

My main role in the project was to write the majority of the code in Matlab that would do the data analysis. With the help of Ginny and Seth, our final product successfully met the requirements of our customer.

Ginny Martin: Experimentation and Communication

Initially, all three team members had the same task - understand the data. As it turns out, there an infinite number of solutions and we all took different paths. I haven't touched Matlab since my freshman year so I had a considerable amount of catch up to do. My team members graciously gave me a crash course on what they had learned in DSP.

Our customer gave us a diary file that one of his graduate students worked on years ago. I stepped through this file and learned much about the data. I broke it down in to workable pieces and added comment lines so that my other group members could use the tools I was uncovering.

The most useful tool I got from this diary was spike filtering. Though I had to do each point by hand, this made filtering go so much smoother and gave a foundation for the C-code that Seth wrote to automate flaw filtering.

There is a Matlab function called DISP that does partial derivatives between data points - meaning x2-x1, x3-x2, etc. I thought that perhaps I could add or multiply these functions to take each of these differences and somehow divide that value in half, and place it between the existing points to create a smoother curve. The answer turned out to be 'no,' but I spent quite a lot of time experimenting with this hypothesis.

I also tried to fit a sine wave to our B-field data. Our customer suggested the method of least squares. I searched the web and looked in the library and could only find

equations derived for linear fit and polynomial fit for least squares - no sine fit equations. I finally found the IEEE standard (IEEE-STD-1057) for sine fitting which involves matrix multiplication. This works wonderfully for small data sets, but our set contains 49000 points. Even a small section of our data would result in the multiplication of a 100x100 matrix. This would be cumbersome to compute and would possibly crash Matlab. Finally, I derived the equations based on the polynomial fit equations that I found.

I then applied these equations to a perfect sine wave to see if it would produce a perfect sine wave as it should. The answer was 'yes.' However, when these equations were applied to a pseudo-sinusoidal wave, which our real data turned out to be, the result was not regular enough to use as a sine wave. I tried many approaches and had several people check my logic and math. My conclusion was that with only a third degree polynomial, one degree being DC offset, we couldn't possibly have enough information to produce a sine wave.

Originally, we were going to collect data in an ideal environment using an EFM. I located the meters, talked to the appropriate powers, and coordinated acquiring one. This was no small task. I had to search the web for contact information and convince a stranger to let me borrow his expensive scientific equipment. I sent him special EFM boxes, which Federal Express promptly destroyed so we were back near square one.

Once we received the instrument, we still needed an area to set it up in, equipment to receive the transmitted data, and software to organize the data. I developed a mounting system to simulate the release of the instrument into a cloud. I also kept in close contact with the owner of the receiver equipment and learned how to operate it. I then devised a series of experiments that I wanted to perform. Unfortunately, I only got

to run one experiment before our customer decided that he would rather we focus on analyzing the data we had than collecting ideal data.

Before this element of the project was completely done away with, I helped Seth prepare the instrument for refurbishing. We met with Dr. Rison and gave suggestions on how the new sensors ought to be mounted in the EFM.

Quite a bit of hunting and collecting had to be done to relate true north and elevation for each flight as they were mostly done in different cities. I compiled a list of this information for use in determining the actual direction of the horizontal component of the B-field.

I worked with Seth to develop a menuing system and helped Chris debug the final Matlab program. Once we had an acceptable tool, I did some data sampling for customer to locate relevant data ranges within each flight.

Communication was my core strength in the group. I organized weekly meetings with our customer and advisor and kept everyone abreast of changes. Most of the year's writing and presentation preparations were also my responsibility.

Chapter 4: Final Product

We began with several sets of flight data, each of which contains digitized readings from a charge amplifier, Hall-effect sensor, and a mercury switch. To understand the three-dimensional E-field, the direction of the EFM must be extracted. The position of the conductive spheres must be known at any given moment, so the Hg switch data has to be compared to the B-field data. (See figure 10).

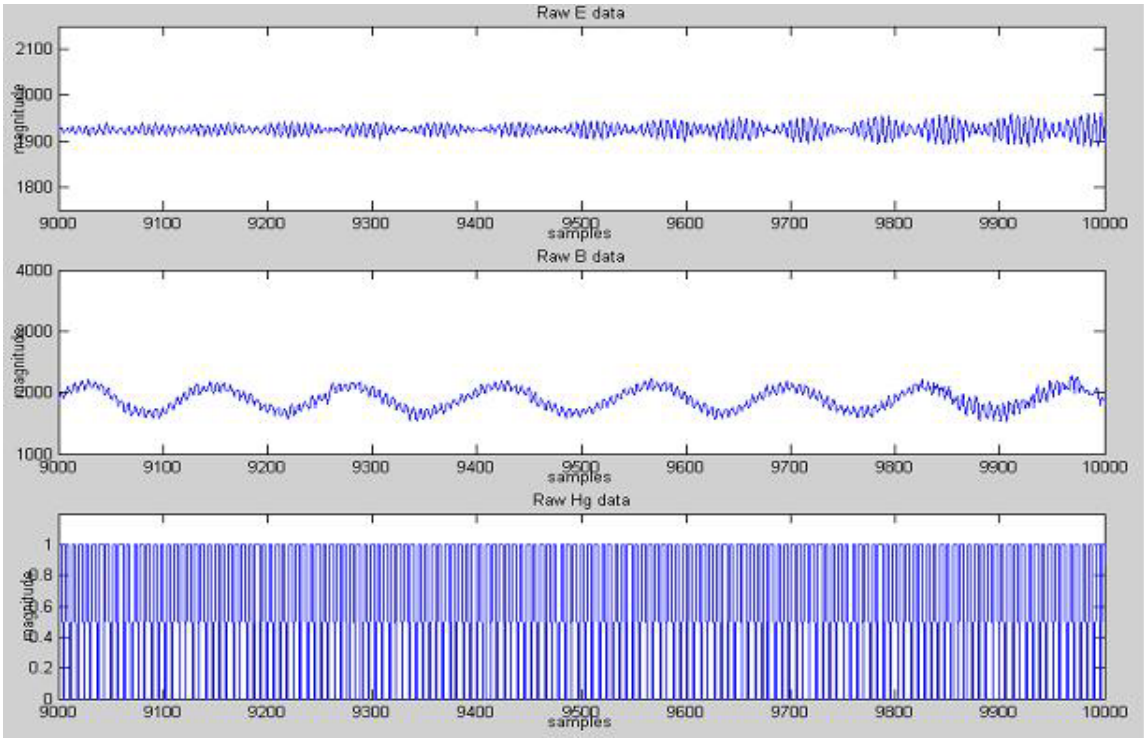


Figure 10: Small section of original data

To start, we took the fast Fourier transform of the mercury switch data to find the components of the frequency. Because the mercury switch data is similar to a square wave, the FFT has several harmonics of the fundamental frequency. Also, all of the frequencies are aliased by the FFT process. An initial filter is applied to the output of the Hg switch FFT to isolate the fundamental frequency from the spurious data. (See Figure 4). To clean up the filtered Hg data even more, we converted the filtered data, which was a rough sinusoid, back into a square wave by making any positive value a 1 and any

negative value a 0. This new cleaner square wave was then filtered again using the same filter from before. This accounted for slight changes in the amplitude of the filtered Hg switch data. We also interpolated the filtered Hg switch data by adding three more points for each existing point to make the sinusoid smoother. (See figure 5).

A phase shift was added to this filtered data to put it in phase with the B-field data. Once those were matched, the phase of this new sine wave was compared with the data from the charge amplifier and the Hall Effect sensor.

Often the electric field in a cloud is completely horizontal. By investigating the phase shift between the charge amplifier and the idealized mercury switch, the angle that the mercury switch needed to be shifted by could be determined. This is important to the quadrature demodulation technique we planned to use to separate the horizontal and vertical components of the electric field. This process finds the component of the charge amplifier data that is in phase with the mercury switch, and the component that is 90 degrees out of phase. Since we wanted these two components to represent horizontal and vertical, we had to correct the mercury switch data.

With the vertical and horizontal components of the E-field at the instrument known, just the direction of the instrument remained to be determined. Since the earth's magnetic field is basically constant across the instrument throughout the flight, the Hall-effect sensor's data was investigated for this information. There were two primary frequencies in the B-field data, the rotation and precession of the EFM. Only the low frequency is needed to determine direction, so the data is passed through a filter to extract it. This yields a sine wave that describes the direction of the EFM relative to the Earth's magnetic field.

Figure 11 shows 2 actual plots. One of the raw E-field and mercury switch data, and the other of the horizontal and vertical components of the E-field. The raw data is

plotted so one can monitor the E-field to see if any odd patterns in the components are due to lightning discharges. The vertical component was extracted by filtering out the high frequency component of the E-field and multiplying it by the mercury switch data. The horizontal component was derived by multiplying this same high frequency component of the E-field but by the mercury switch data shifted by 90 degrees.

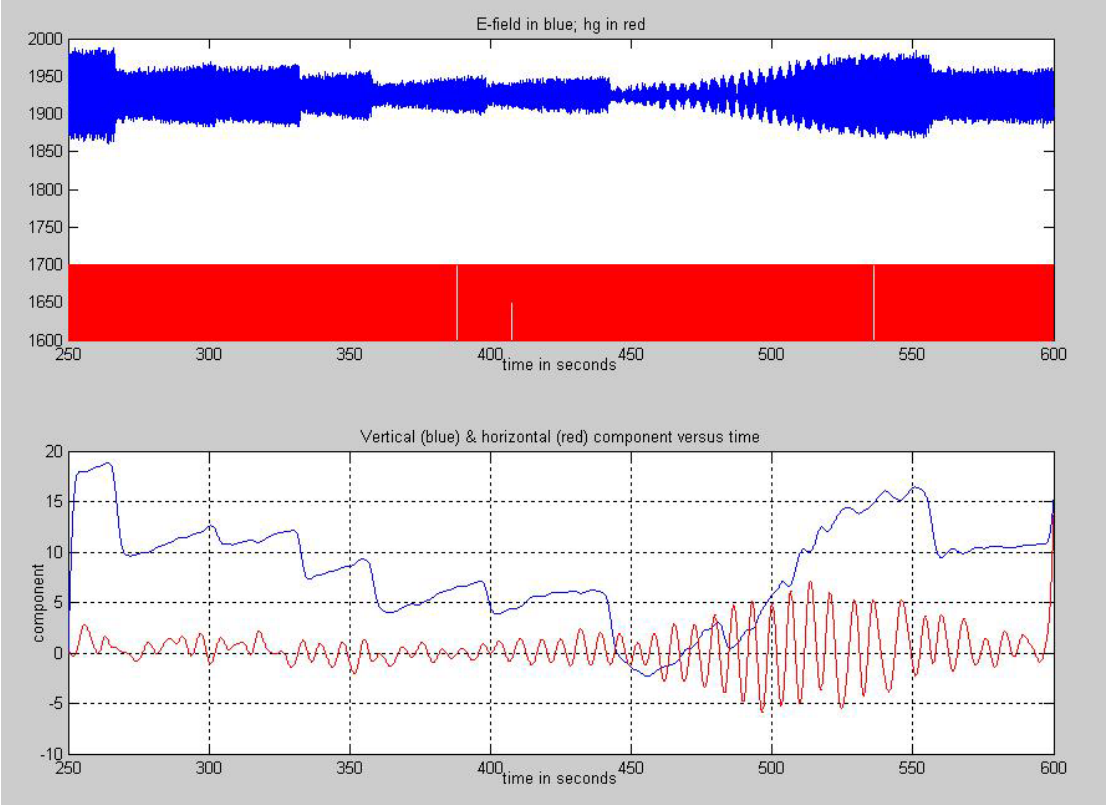


Figure 11: Horizontal and vertical components of E-field
First plot of finished product

Figure 12 shows 2 subplots also, the second is the exact same as in Figure 11. The first however, is of the raw B and mercury switch data. The relevance of the graph is to monitor the B-field as the components of the E-field change and also to show that the filtering is not affecting the vertical and horizontal components.

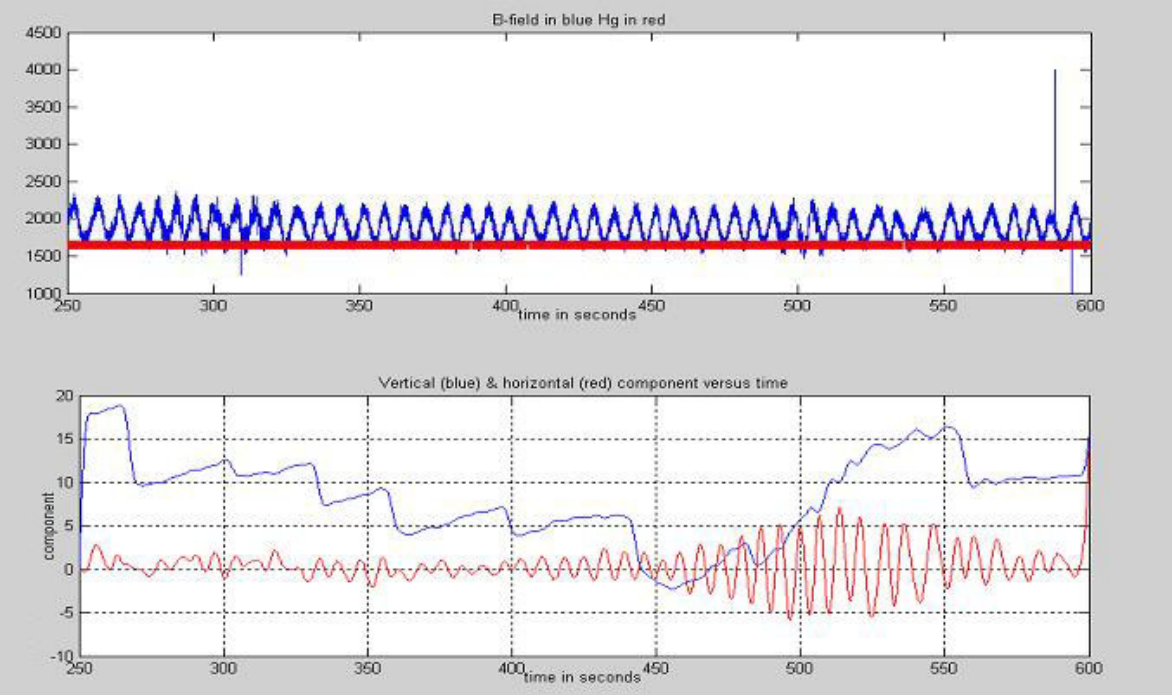


Figure 12: E-field directional components relative to B-field
Second plot of finished product

Figure 13 shows raw E-field and mercury switch data. The second plot shows the phase of the E-field versus the mercury switch. The relevance of this plot is that it shows if there is a switch in the direction of the E-field. If the phase shift changes, the E-field direction changes accordingly. If the phase shift remains constant, the direction of the E-field remains constant. This graph is produced by doing the quadrature demodulation of E-field data with the Hg switch data.

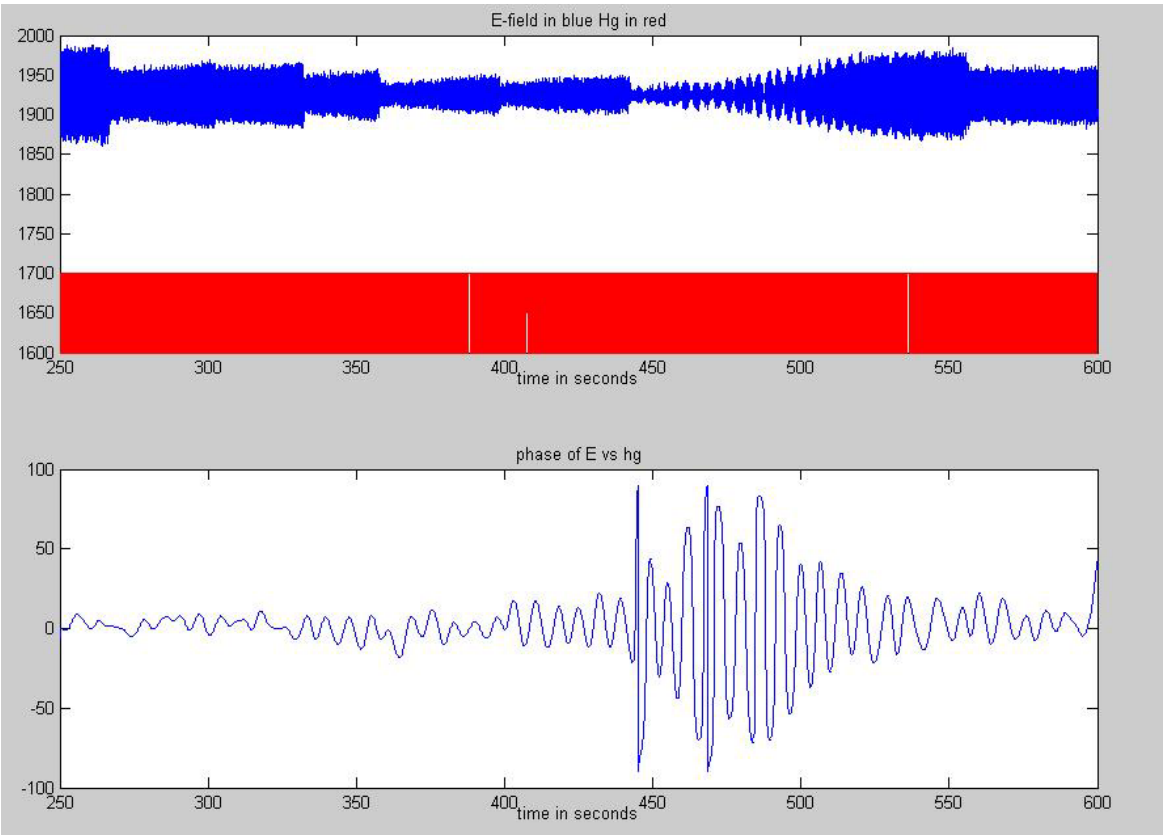


Figure 13: Angle of Hg switch to horizon
Third plot of finished product

We also use quadrature demodulation to determine the position of the instrument relative to the B-field. Using the Hilbert Transform, the Hg switch data is converted into a sine-cosine pair:

$$\text{Hilbert}\{A\cos(\omega t)\} \Rightarrow A\cos(\omega t) + iA\sin(\omega t)$$

Where the real part of the Hilbert transform is the original sinusoid and the imaginary part is the sinusoid shifted by 90 degrees. The two parts of this pair will be individually multiplied by the B-field data:

$$B\cos(\omega t + \phi) * A\cos(\omega t)$$

$$B\cos(\omega t + \phi) * A\sin(\omega t)$$

This results in two separate sinusoids as follows:

$$\tfrac{1}{2}AB[\cos(2\omega t + \phi) + \cos(\phi)]$$

$$\tfrac{1}{2}AB[\sin(2\omega t + \phi) - \sin(\phi)]$$

The results will be adjusted by a low-pass filter to remove the high frequencies incurred during this process. This will leave two values:

$$\tfrac{1}{2}AB\cos(\phi)$$

$$\tfrac{1}{2}AB\sin(\phi)$$

The two outputs will be divided by each other so the amplitudes will cancel out, and only tangent of the phase shift will remain

$$\frac{\tfrac{1}{2}AB\sin(\phi)}{\tfrac{1}{2}AB\cos(\phi)} = \tan(\phi)$$

Taking the arctangent of this result will reveal the phase difference. From this we know the degree of rotation between the instrument and the B-field. This information can be adjusted for minor deviations by shifting the Hg switch by a small phase amount.

Figure 14 shows the raw B-field data and the raw Hg switch data and a graph of the phase of the B-field versus the mercury switch. This data is found using the above quadrature demodulation technique. This shows the angle of the instrument from the horizontal of B. This should be a sine wave of relatively constant frequency.

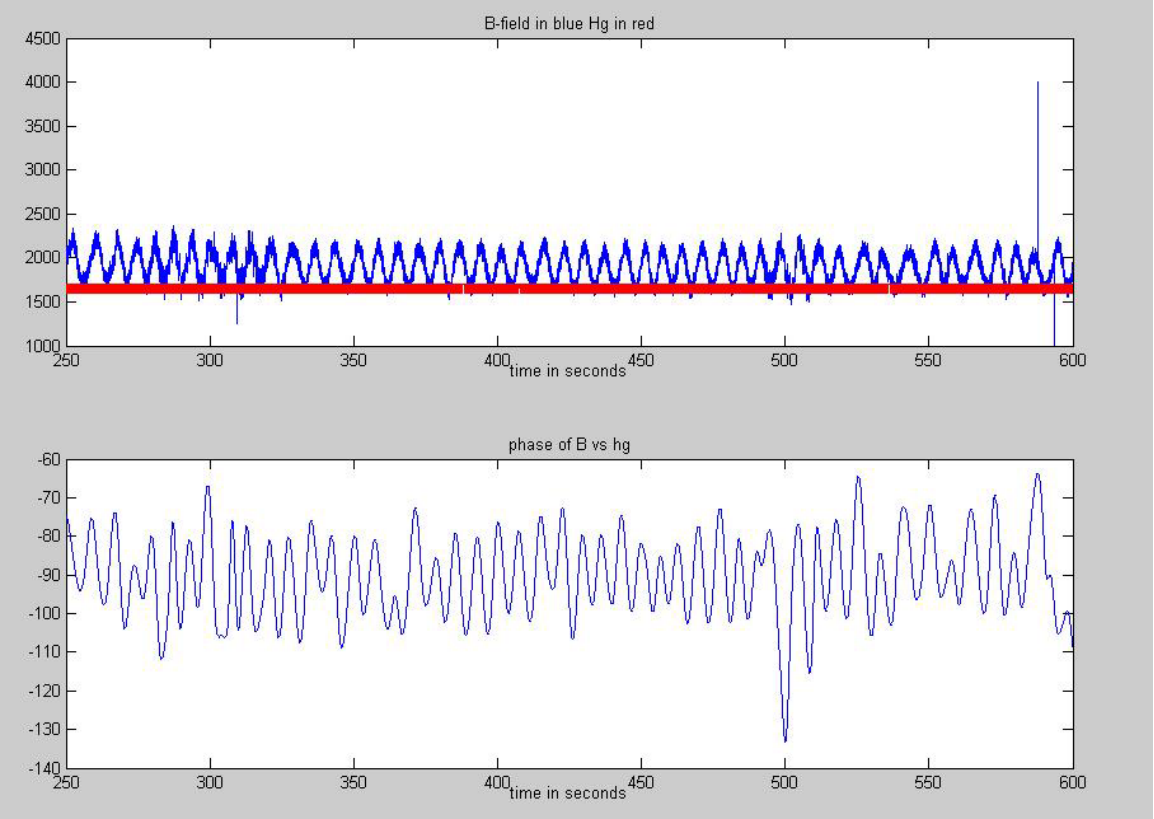


Figure 14: Orientation of instrument
Fourth plot of finished product

Figures 5-8 show the same things as graphs 1-4 respectively, but show the interpolated, smoothed data in the top plots instead of the raw data. This is relevant because the filtering takes place on the interpolated data. If the resulting vertical and horizontal components of the E-field, or the phase shifts between E, B and hg are not what it appears they should be, one can look back at the raw data and see if odd data anomalies are due to instrument failure or are real data points.

Figure 15 shows the individual components of the E-field vs. time; with time on the vertical axis and the components on the horizontal axis. As time passes, the instrument is rising. If it is assumed that the instrument is rising at a fairly constant rate, it can also be assumed that time and altitude are linearly related. Therefore, the graph can be interpreted as the E-field vs. altitude.

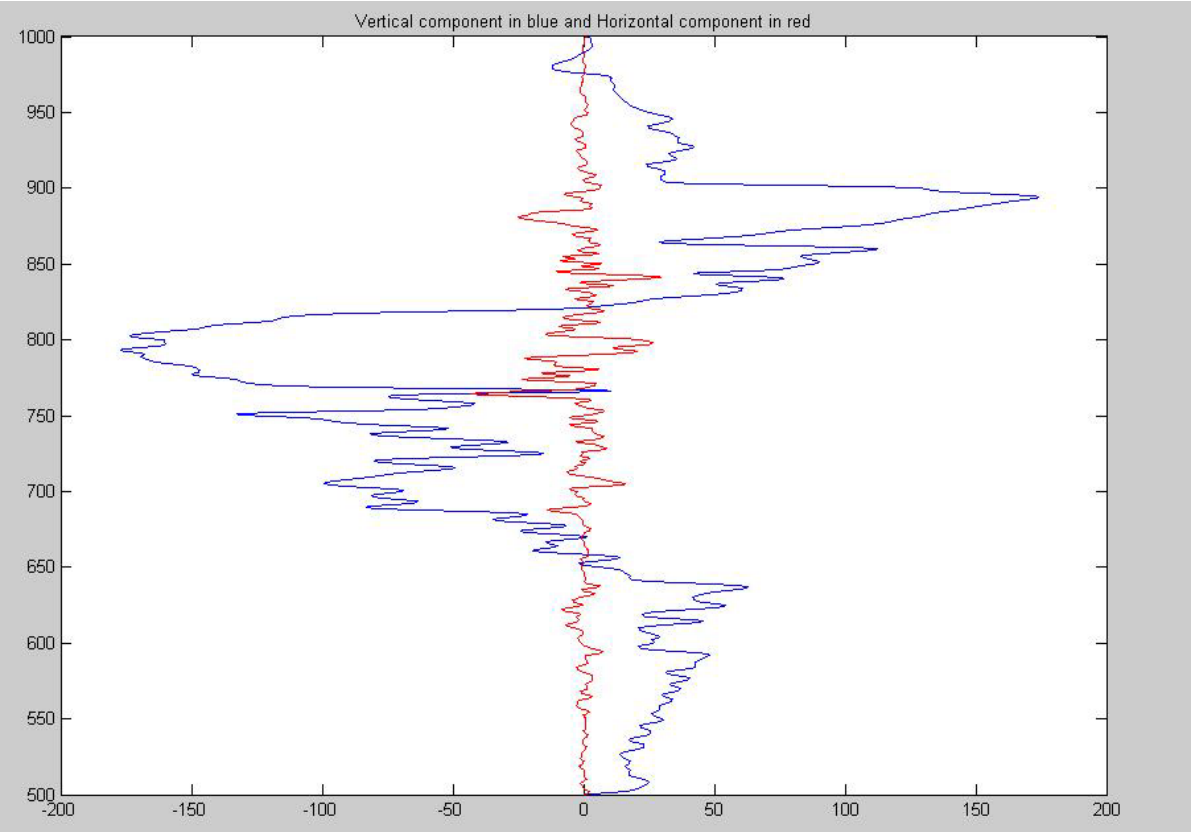


Figure 15: E-field components relative to altitude
Ninth plot of finished product

Using this, one can see how the fields change with altitude and when lightning discharges occur.

For implementation questions, please refer to the text version of the finished product in the appendix.

5.0 Future Work

The three dimensions have been extracted for a cylindrical coordinate system, but it would be preferable to compute them in the Cartesian coordinate system. This can be done by taking the phase shift of the low frequency component of the B-field with the horizontal component of the E-field. Once the electric field has been extracted in this form, the three-dimensional E-field can be displayed a function of time. Adding that feature would greatly increase ease of use. The phase shift that must be applied to the mercury switch is chosen by the user, and is found through trial and error. An automated system to find the phase would be preferable. These two additions would be large strides towards reaching our goal of furthering existing research.

Appendix

Reconfigurable Data Path Processor Verification

A Senior Thesis
Presented to the Faculty of the
Electrical Engineering Department
of New Mexico Tech

By
Steven Wasson
Thomee Wright

In Partial Fulfillment of the requirements for the course:
EE 482 -- Senior Design II

May 1, 2002

ABSTRACT

NASA's Institute for Advanced Microelectronics is in the process of developing a Reconfigurable Data Path Processor. A full design for the RDPP exists at this time but needs to be tested. This project verifies the design of the Processing Element, the workhorse component of the RDPP. This report first documents the implementation of the Processing Element in an Altera APEX20K200EFC chip and an interface to allow testing from a PC. The report next documents how the PC software interfaces with the Altera board to run test vectors. This report then documents how the test vectors were generated and why they provide sufficient test coverage. Finally, this document details how testing the device on the Altera board was unsuccessful.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the sponsor of this project, Dr. Greg Donohoe, for providing such a challenging project that contributed greatly to our learning experience. Also, Dr. Donohoe provided all of the equipment and materials that were needed to conduct this project when we began it. We would also like to acknowledge our faculty advisor for this project, Dr Kevin Wedeward, for the light-natured, humor-filled project meetings we had throughout the year. Finally, we would like thank all of the faculty of the Electrical Engineering Department at New Mexico Tech, to whom we are in great debt for the exceptional instruction they have provided us.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

The Reconfigurable Data Path Processor (RDPP) is a custom processor being developed by NASA's Institute for Advanced Microelectronics. It is being developed for high-throughput processing of streaming data aboard spacecraft. It's primary application will be image and signal processing (Donohoe 2002).

1.2 PROJECT OBJECTIVES

The objective of this senior design project was to verify the functionality of the RDPP. Verification, or functional testing, means to execute test inputs on the device under test (DUT) and verify that the output is functionally correct for each test input. Our customer, Dr. Greg Donohoe, specified that the RDPP be verified using an Altera Programmable Logic Device (PLD) that he supplied. Using a PLD for device verification has two advantages: 1) it's closer to the final chip layout than a software implementation but doesn't require that the device be fabricated and 2) design revisions can be easily implemented.

The end products of this project are twofold: first, this thesis documenting our design and reporting on the current state of the RDPP and second, a software and hardware package containing test procedures and tools for verification of future design revisions.

1.3 PROVIDED MATERIALS & EQUIPMENT

When we began this project, Dr. Donohoe provided us with the hardware design files for the RDPP written in VHDL. We were also provided with an Altera® Excalibur board containing the APEX20K200EFC Programmable Chip and a computer with the Altera® Quartus II software to program the Altera® chip.

CHAPTER 2: TECHNICAL BACKGROUND & DESIGN OVERVIEW

2.1 INTRODUCTION

This chapter will provide a brief technical background to this project. First, a basic description of the Reconfigurable Data Path Processor will be given. This will include schematics and definitions of components. Next, an overview of the Altera® Excalibur board and Quartus II software will be given. We'll then provide a brief introduction to the theory and practice of functional testing of devices. Finally, we will give an overview of our design for this project.

2.2 THE RECONFIGURABLE DATA PATH PROCESSOR

The Reconfigurable Data Path Processor (RDPP) is a custom processor being developed by NASA's Institute for Advanced Microelectronics. It is being developed for high-throughput processing of streaming data aboard spacecraft. The current RDPP design contains 16 reconfigurable Processing Elements (PEs) arranged in parallel. Figure 1 below is a block diagram of the current RDPP design.

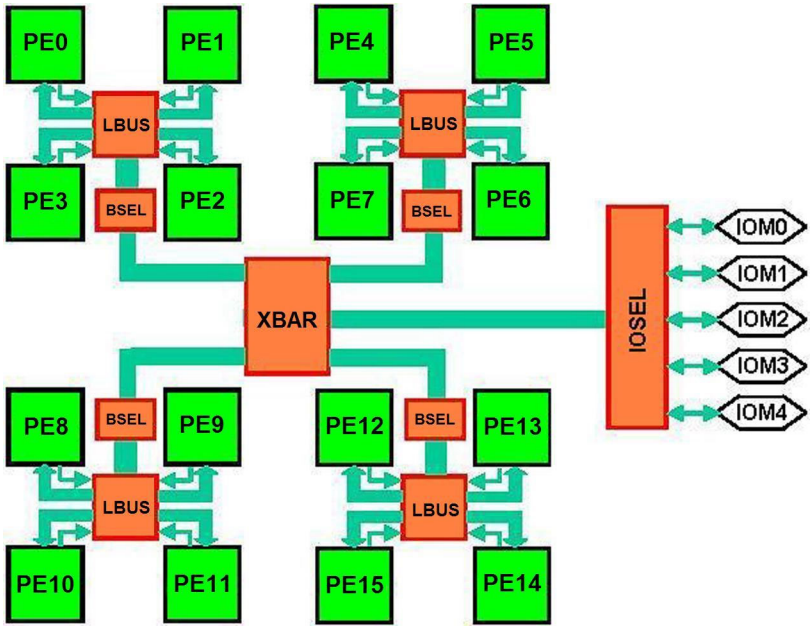


Figure 1: Block Diagram of the Reconfigurable Data Path Processor

Each Processing Element is identical and, other than five input data registers, contains purely combinational logic. There are three 24-bit data inputs to the PE that are stored in its internal registers. There are two 24-bit registers that are used for storing constants. There are 67 configuration bits that are used to configure the data paths within the PE for different computations. Finally, there are two 24-bit outputs. Figure 2 below is a block diagram of the Processing Element that highlights all of its I/O and control signals.

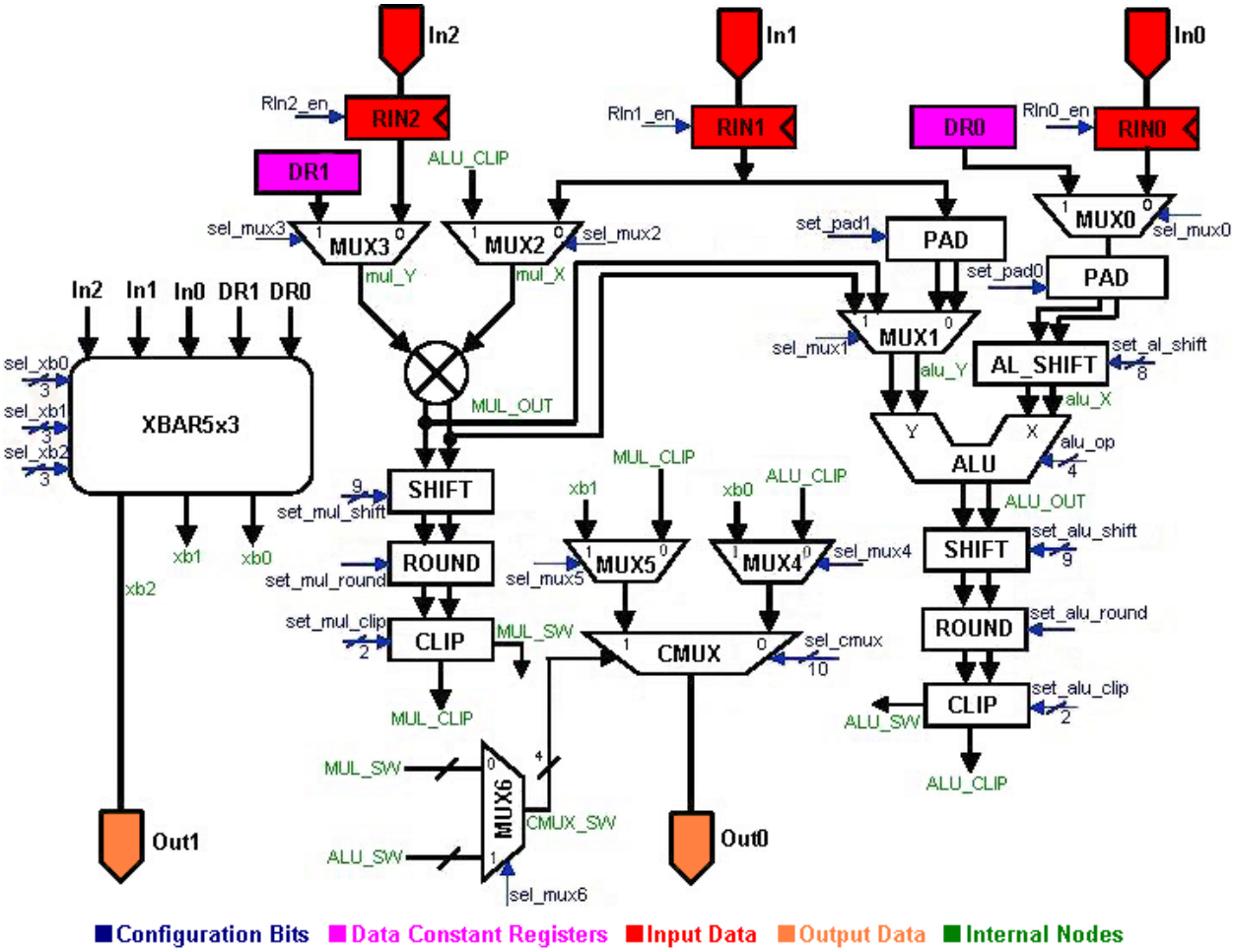


Figure 2: Block Diagram of Processing Element Highlighting I/O & Control Signals

There are three primary data paths within the PE: a delay path, a multiplication path, and an arithmetic path. Figure 3 on the next page highlights these various data paths.

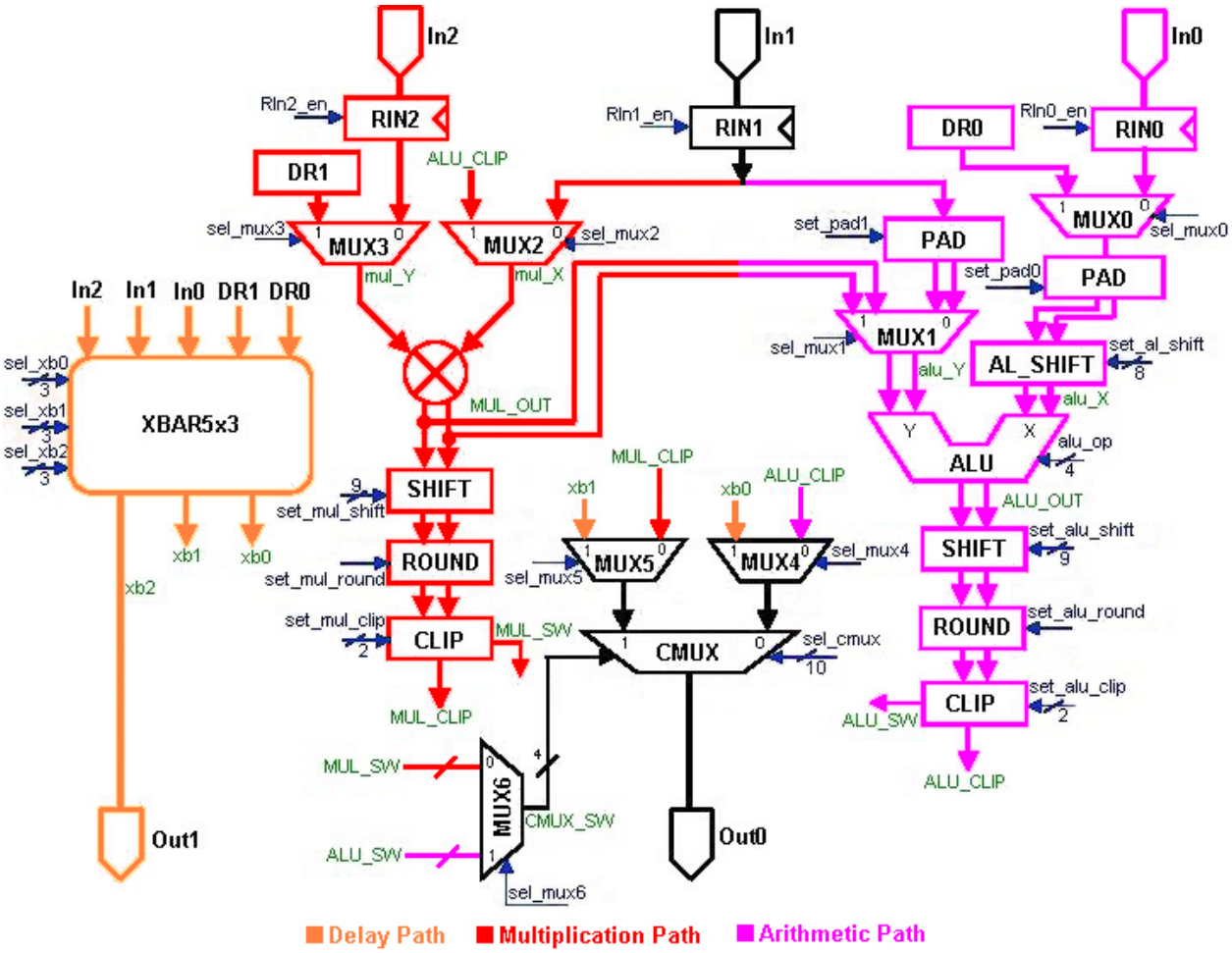


Figure 3: Block Diagram of Processing Element Highlighting the Data Paths

As shown in the above figure, there is a data path that goes from the multiplication path to the arithmetic path where the MUL_OUT node is an input to MUX1. This can be used to implement Multiply and Accumulate routines, which are useful in signal processing applications. There is also a path that goes from arithmetic to multiplication where the ALU_CLIP node is an input to MUX2.

There are a total of 10 different components in the Processing Element. Below are functional descriptions of the components.

ALU – This is the arithmetic logic unit. Based on the alu_op bits, it performs the following operations: addition, subtraction, and, nand, or, nor, xor, xnor, not, negate, pass X operand, pass Y operand.

AL_SHIFT – This component shifts 48 bits. It can shift both left and right. It can shift logically and arithmetically. The set_al_shift bits determine what kind of shift to do and how much the operand is shifted.

CLIP – This component clips the 48-bit operand to a 24-bit result. It also produces a four-bit condition word that represents if the result is negative or zero or if an overflow or underflow occurred.

CMUX – This component is a conditional multiplexer. It can either unconditionally select one of the two inputs or select one based on the condition word that is provided to it.

MUX – This component is a simple multiplexer that selects one of the two inputs and places it on the output depending on the value of the select line. The PE contains both 24-bit and 48-bit multiplexers and one 4-bit multiplexer for selecting the condition word from the multiplication or arithmetic path.

PAD – This component pads a 24-bit number with 0s or 1s to make it into a 48-bit number.

REGISTER – There are five 24-bit registers, two for the data constants (DR0 & DR1) and three for the data inputs (RIN0, RIN1, & RIN2).

ROUND – This component rounds its input or simply passes it through depending on the value of set_round.

SHIFT – This component is just like the AL_SHIFT except that it also has the capability of performing a circular shift.

XBAR5x3 – This component has five 24-bit inputs, three 24-bit outputs, and three 3-bit select lines. Each of the outputs can select one of the five inputs.

For a more thorough description of the RDPP and its operation, refer to "RDPP Overview" by Dr. Greg Donohoe.

2.3 ALTERA® EXCALIBUR BOARD AND QUARTUS II SOFTWARE

The Altera® Excalibur board, shown in Figure 4 on the next page, is designed to implement embedded systems such as NIOS processor design provided with the board. The APEX20K200EFC chip is a high-speed SRAM programmable chip. The board allows you to program this chip through a JTAG interface or by installing your configuration in

on-board flash memory. The Altera® Quartus II software, like MAX+plus, allows the user to design digital systems textually with a Hardware Description Language or graphically. It also serves as the means to program the PLD.

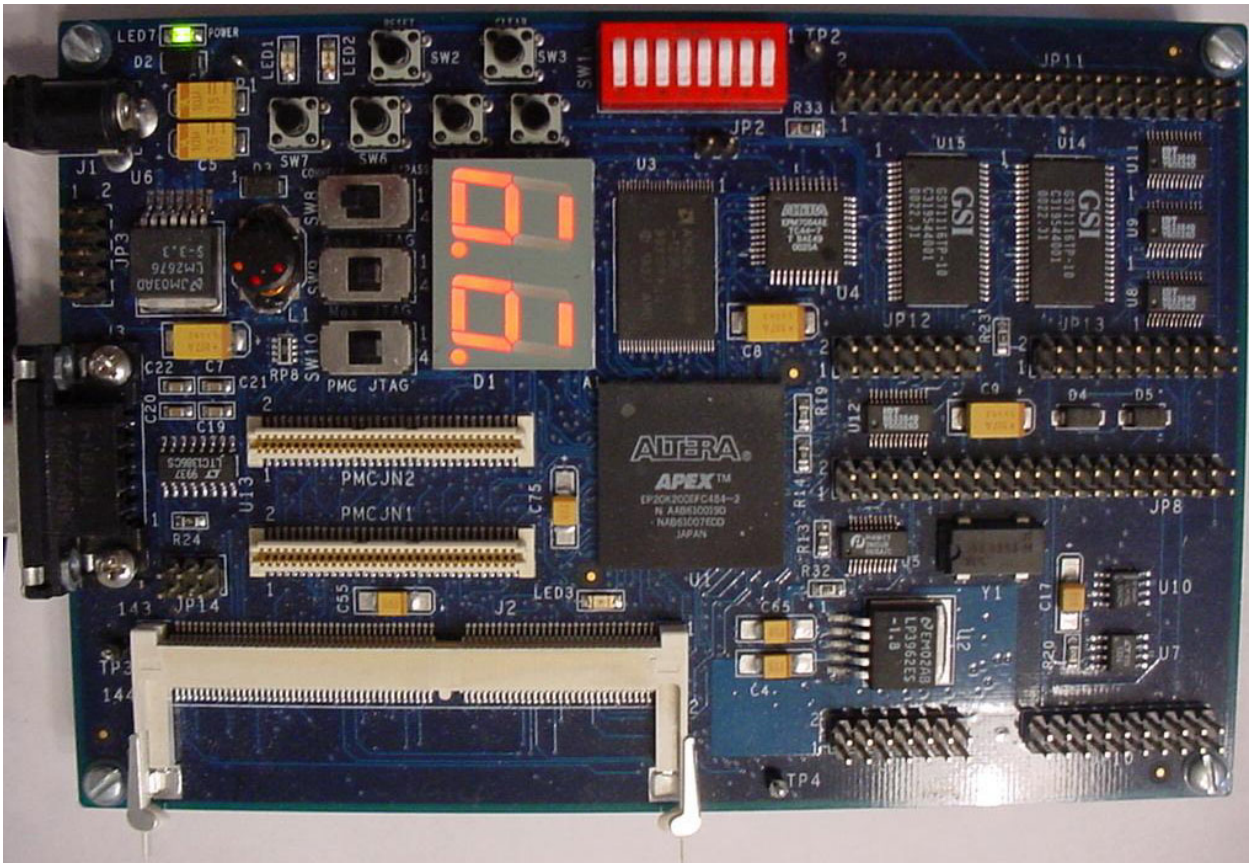


Figure 4: The Altera® Excalibur Board

2.4 FUNCTIONAL TESTING

A functional test is one of the many tests executed on a device as it is being developed. It answers the question, "Is this device actually doing what we are designing it to do?" To answer this question, test inputs or vectors are executed on the device and the output produced is compared with what the output should be according to the device specifications. If the actual and expected outputs do not equal each other then there is an error in the design. Ideally, one would like to test the device with every possible input combination. In reality, however, the device has so many input combinations that it is

impossible to test for every case. Consider the Processing Element of the RDPP. To test it would require the execution of 2^{187} test cases. Since it is impossible to test every case, the tester must derive a set of test vectors to execute that will convince him that the design is correct.

2.5 DESIGN OVERVIEW

To verify the RDPP, we decided to test only a single Processing Element. There were two factors involved in making this decision. First, earlier work done here at Tech on the RDPP showed that a single Processing Element takes up 57% of the APEX20K's resources, thus the entire RDPP cannot be emulated and verified in this chip. Second, all data manipulation occurs in the PE. Because of this, it is critical to test the PE for functional testing of the RDPP, while the other components are much less complex and easier to verify later.

We decided to develop an interface between our computer and the PE on the Altera® board. This interface was used to send and receive data between the PE and PC. We also decided to develop software that would dynamically generate test inputs to execute on the PE as well as their expected results. This software used the interface to the Altera® board to send and execute the test vectors on the PE and receive the results. The software also compared the actual results with the expected results and reported if any errors were detected.

Chapter 3: Altera Hardware Design

Steven Wasson

3.1 INTRODUCTION

This section covers all the Altera® hardware design necessary for this project. This includes the synthesis of the Processing Element as well as the design and implementation of the interface.

3.2 SYNTHESIZING THE PROCESSING ELEMENT

The first step we undertook to synthesize the Processing Element was to convert Dr. Donohoe's VHDL design files to Quartus II Block Symbol Files (BSF). These BSF files were then placed in a Quartus II Block Diagram File (BDF) and interconnected as specified by the block diagram of the Processing Element showed in Figures 1 and 2. Due to the magnitude of this file, it cannot be included in this report. It is, however, available on our project CD. We then compiled this BDF file with the APEX20K200E chip specified as the target device. This took about fifteen minutes on average and compiled correctly once various bugs were eliminated. The Processing Element used 57% of the chip's total resources.

3.3 INTERFACING TO THE PROCESSING ELEMENT

We decided to develop the interface between a computer and the Processing Element using the parallel port. We briefly considered using a Data Acquisition card but they are expensive as well as complicated to set up and use. The parallel port, on the other hand, exists on every computer, thus requiring no additional equipment costs and making our design portable. In addition to this, we are both intimately familiar with how to setup and

use the parallel port due to coursework in EE 352 – Microcomputer Interfacing offered at New Mexico Tech.

Figure 5 below shows a block diagram of the interface we designed between the computer and the Processing Element on the Altera® board.

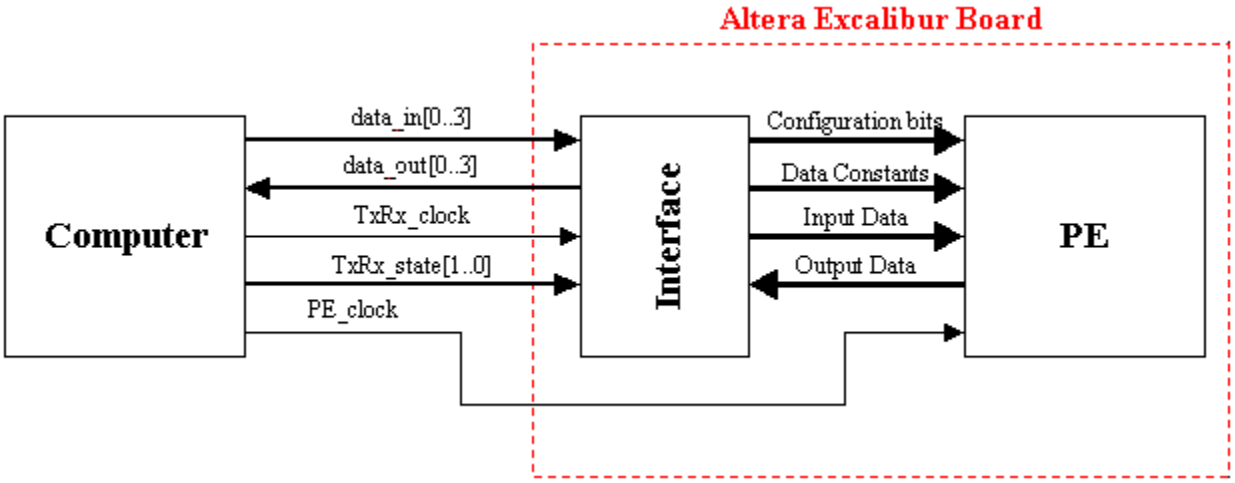


Figure 5: Block Diagram of PC Interface to the Processing Element

As shown, data is transferred to and from the Altera® board in 4-bit packets. When data is sent to the Altera® board, it is stored in temporary 4-bit registers. Once all the input data to the Altera® board has been sent it is transferred to the PE and clocked into it's internal registers. This immediately produces a new result. This result is sent back to the computer.

There are four different transmission states: send configuration bits, send data constants, send input data, and receive output data. Two bits, TxRx_state1 and TxRx_state0 are used to represent the transmission state. Table 1 on the next page shows the values of TxRx_state1 and TxRx_state0 that correspond to each transmission state.

TxRx_state1	TxRx_state0	Transmission State
0	0	Send Configuration Bits (PC→PE)
0	1	Send Data Constants (PC→PE)
1	0	Send Input Data (PC→PE)
1	1	Receive Output Data (PE→PC)

Table 1: Interface Data Transmission States

Finally, there are two clock signals that run from the computer to the Altera® board. The first clock signal, TxRx_clock, is used to control the 4-bit transmission and clock in the data that is being transferred to the Altera® board to the temporary 4-bit registers. The second clock signal, PE_clock, is used once all the data has been sent to the Altera® board to clock it in to the PE's internal registers.

In our initial design of the interface, we used simple counters and demultiplexers to route the incoming data to the appropriate register. Figure 6 below illustrates how this was done.

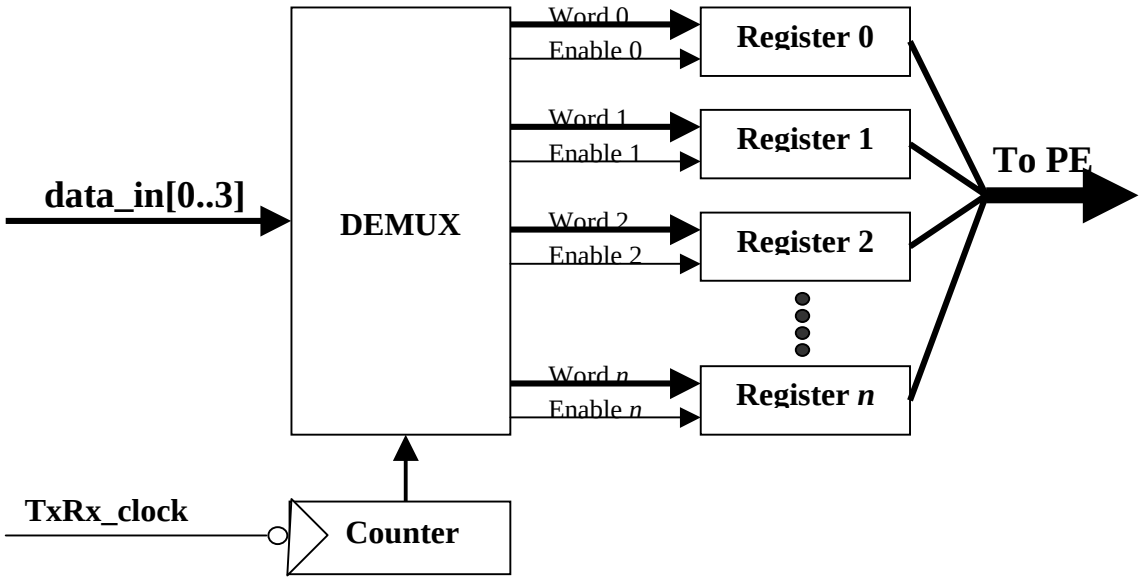


Figure 6: Initial Design to Route the Incoming Data

As shown, data is input to the demultiplexer. The value from the counter determines which output of the demultiplexer the input is placed on. The demultiplexer also provides an enable signal for each output. The outputs and enable signals go to different 4-bit registers. When the enable line for one of the registers is high and a rising edge of TxRx_clock occurs, the data is clocked into the register. On the falling edge of TxRx_clock, the counters are incremented and the input data is routed to the next register. Three of these demultiplexer and counter blocks for routing the incoming data were used for receiving the configuration bits, the data constants, and the input data. Only one of these three block were enabled at a time based on the value of the transmission state bits.

For sending the output from the Processing Element back to the computer, we used a mulitplexer and counter as shown in Figure 7.

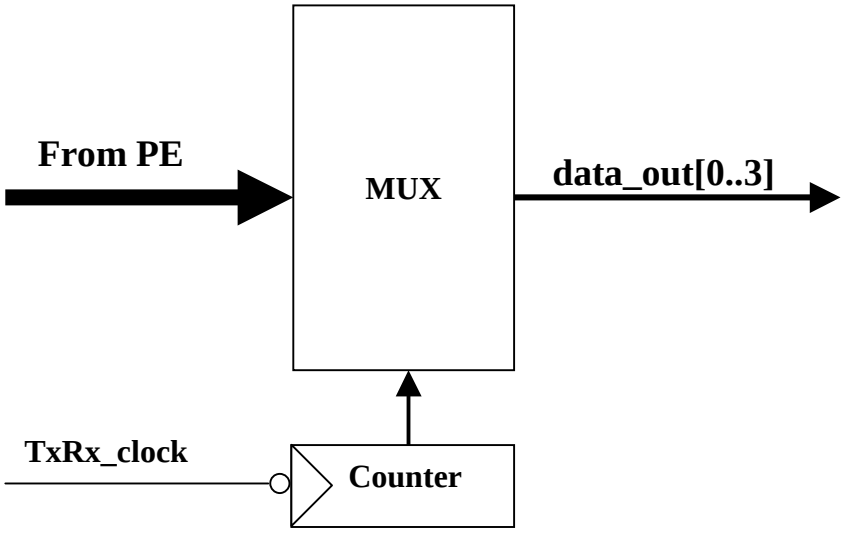


Figure 7: Initial Design for Sending Data to Computer

As shown, the output data is broken up into 4-bit packets and passed into the multiplexer. The counter determines which 4-bit piece is placed on the output and sent to the computer. The TxRx_clock increments the counter on the falling edge to select all of the 4-bit pieces sequentially. This whole block is enabled when the transmission state bits are both high,

corresponding to the transmission state in which data is being sent from the PE to the computer.

After implementing and testing this design we upgraded it to a more efficient design. We took out the counters and demultiplexers that were used for routing incoming data and connected the input data to all of the temporary 4-bit registers. We replaced the counters with a state machine that would drive a single enable line at a time for all of these registers. On the falling edge of TxRx_clock, the next enable line was driven high for the next register. Figure 8 below shows a block diagram of this design.

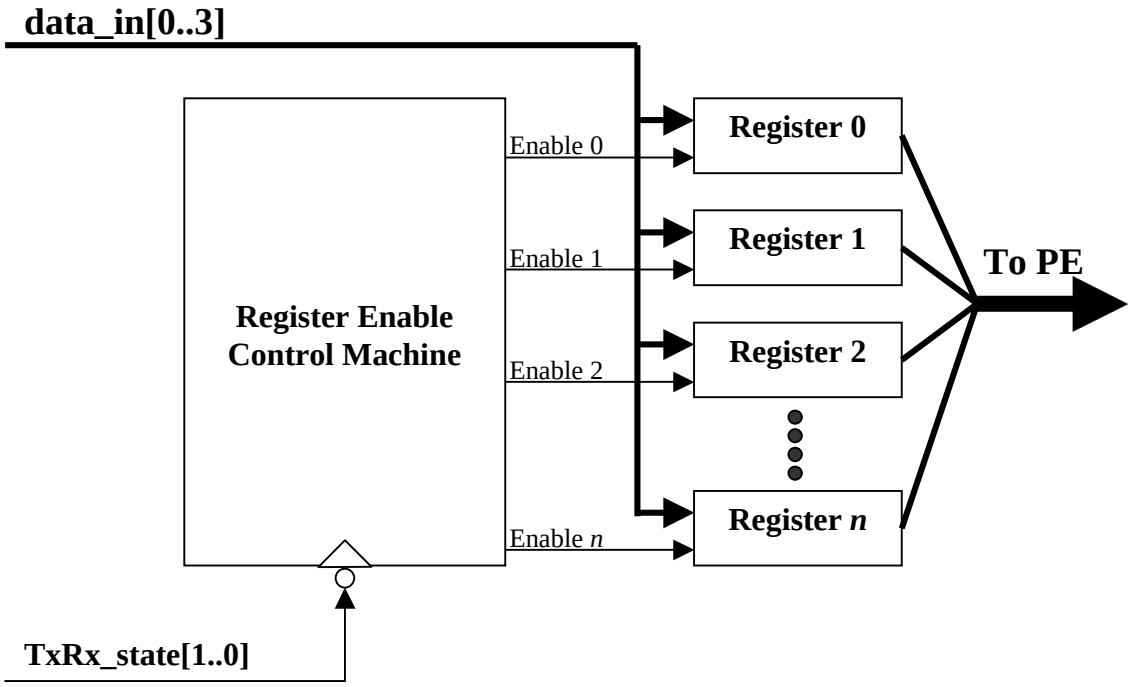


Figure 8: Final Design to Route the Incoming Data

We changed our design to send data back to the computer in a similar manner. The counter was removed and replaced with a state machine that drove singe select lines. We changed the multiplexer so that it took in individual lines to select which input to select rather than a string of bits encoding which input to select.

For the Altera® design files of our interface, refer to our project CD.

3.4 PROGRAMMING THE ALTERA® CHIP

With the Processing Element successfully compiled and our interface developed, we were ready to program the Altera® chip. Below is a diagram of the Altera® board with all the components that we used labeled.

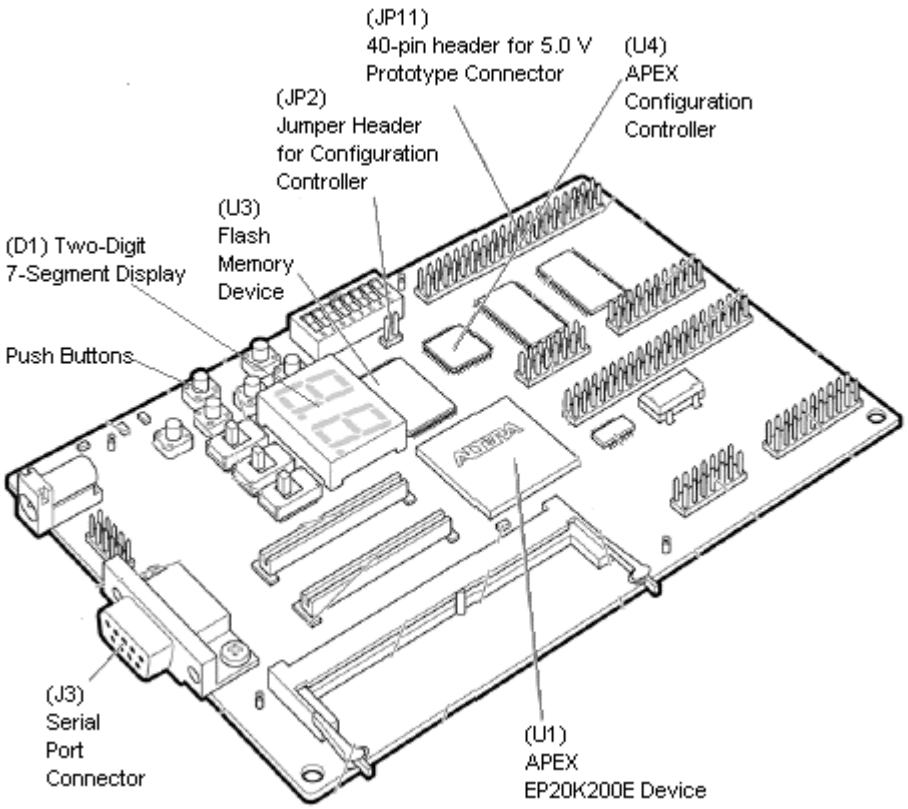
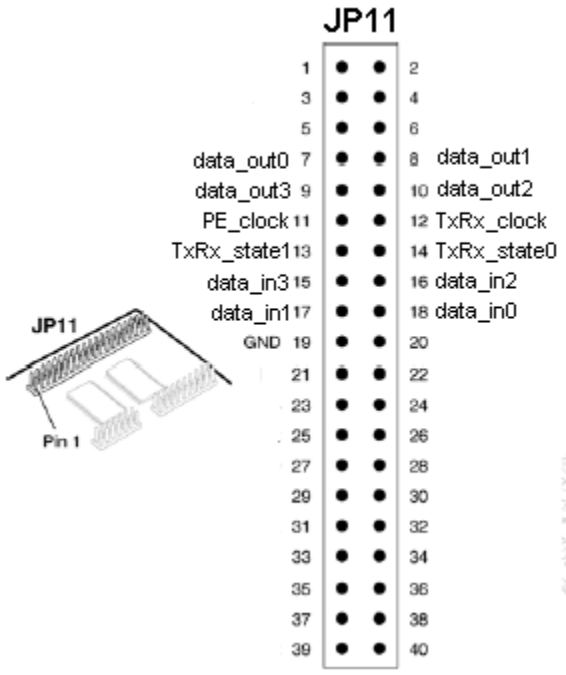


Figure 9: Altera® Board Diagram

In our program, we designated JP11 for the I/O to the computer. Figure 10 on the next page shows a pin out for the JP11 header. We set up the Two-Digit 7-Segment Display to display all of the I/O between the computer and the Processing Element for debugging purposes. Figure 11 on the next page shows which LED corresponds with each I/O signal. Finally, we used one of the Push Buttons for a reset (NIOS Embedded Processor Development Board 3-18).



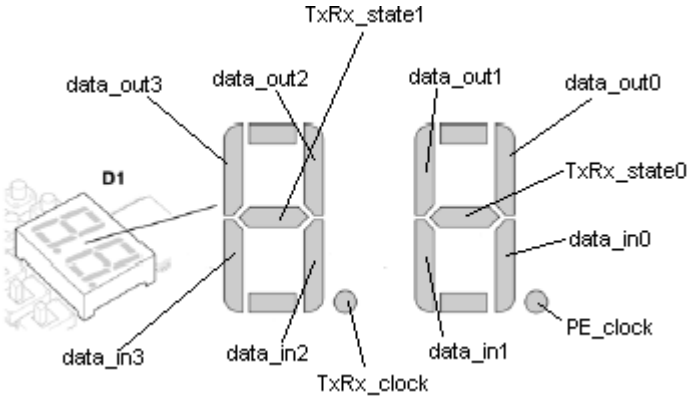


Figure 10: Pin Out of JP11 Header **Figure 11: Interface I/O on 7-Segment LED's**

The Altera® chip can be programmed either through the JTAG interface or through the configuration controller on the board. Since the chip is volatile, we would have to program every time we powered up the chip if we were to use the JTAG interface. The way that the configuration controller works, however, is that it loads a configuration from the flash memory on the board. To place our configuration in the flash memory we had to do several things. First, we set up our Altera® project so that it wrote the configuration in a hexadecimal format. We then used a program called "hexout2flash" developed by Altera® to convert the hexadecimal configuration file into a file that can be downloaded into the flash memory. Finally, with a jumper on JP2, we connected our development computer to the Altera® board through the serial port and used a program called nios-run to download our configuration into the flash memory. For more information on downloading configurations into the flash memory refer to the "NIOS Embedded Processor Software Development Reference Manual" from Altera®.

CHAPTER 4: TEST SOFTWARE DESIGN

Thomee Wright

4.1 INTRODUCTION

This section details the development of the PC side of the hardware and software. First, the available pins on a PC parallel port and their usage in this project is given. Second, the requirements and end solution for the design of the PC test software are given. Finally, a detailed explanation of the generation of test vectors is provided.

4.2 PC HARDWARE INTERFACE

The standard PC parallel port provides twelve output and five input pins. The output pins correspond to two eight bit hardware registers: an output byte and a control byte. Only the four low bits of the control byte are available as output pins. The five input pins are available as an eight bit status register. Only the high five bits correspond to actual input pins.

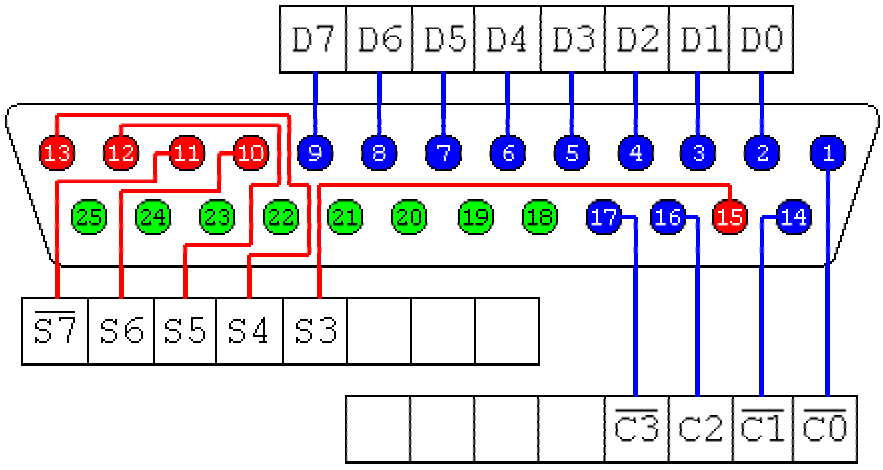


Figure 12: PC Parallel Port Diagram

Our project required only eight output bits, so we used only the data register. Bits 3..0 were used for data_in[3..0], bits 5..4 for TxRx_state, bit 6 for TxRx_clock, and bit 7 for PE_clock. The output from the Altera® interface, data_out[3..0] was read via the status

register bits 7..4. For a full wiring diagram of the Altera® to PC cable, refer to the project CD.

4.3 SOFTWARE DESIGN

The purpose of the PC test software is to run enough tests on the hardware to either find any problems with the design or to provide a reasonable assurance that there are no problems. Several factors influenced the design of the test software. The design needed to be easy to implement, portable, and fast.

In order to keep the implementation of the software as painless as possible and to make sure our customer would be able to use and modify it, it is implemented in Linux. This provided several advantages. It provides a stable multitasking environment capable of running long computational tasks without interruption. Access to the PC's parallel port is very easy in Linux. Most importantly, the operating system and a full set of development tools are available freely, ensuring that our customer will be able to use and modify the testing system without having to purchase any additional tools or worry about licensing restrictions.

The other primary consideration in designing the testing software was speed. In order to effectively test the device, it is necessary to run a vast number of test cases. In order to run enough test cases in a reasonable amount of time, the speed at which tests can be run is critical. The maximum speed at which data can be written to the parallel port is fixed, and the PC interface in the Altera® chip can easily receive at this speed. The time necessary to compute a result in the Processing Element is also fixed from the point of view of the PC software. This means that the time to run a test case on the Processing Element is a fixed constant. The remaining part of each test case is computing the expected result. In order to

test the device, it is necessary for the PC to independently compute the correct result so it can determine whether the response from the device is valid. It is in the computation of the expected result that timing is critical.

There were several options for computing the expected result. Our customer has a C++ object which simulates the Processing Element of the RDPP. It takes a configuration string and a set of inputs and will generate the expected output. The biggest advantage to this option is that it is a complete, ready to go solution. Unfortunately, though, its versatility and flexibility, which are its strongest points on its own, cause it to require significantly more computational overhead. This made it a poor solution for our project. The next option was to develop our own Processing Element simulator focussed on efficiency rather than flexibility. This option would require significant effort to develop, and probably would have a high computational overhead as well.

Because the test cases are broken down into groups of many tests with the same configuration, other methods of computing the expected results are possible. A function which computes the expected value for a test configuration given the inputs could be extremely efficient since it only has to perform the necessary calculations and doesn't have to spend time checking the configuration for what to do. One solution would be to generate these expected value functions at run time based on a configuration word. This process would not have to be as efficient, since it would only be done once per set of tests, allowing its effective time cost to be amortized over many test cases. While this would be a graceful and reasonably efficient solution, the language we chose for the project, C, does not have the ability to generate functions on the fly in a clean manner, so we can't use this solution. A less graceful but feasible alternative to this method is to write the expected value

function when writing the test configuration word. This adds to the software size, but keeps the execution speed of test cases down. This is the solution we used in our test software. For the full source code to the test software, refer to our project CD.

4.4 TEST CASE GENERATION

Because it is not possible to test every possible combination of inputs to the Processing Element, a subset of inputs which sufficiently test the device must be used. This can be accomplished by testing the interconnections between components, then testing each individual component. However, even using this approach, most components cannot be exhaustively tested. A subset of input cases for each component, which should insure full functionality, are used.

In order to fully test the Processing element, several types of tests must be run. The data path control mechanisms through the processor are tested first. The selection inputs on muxes are easy to test exhaustively, but only a subset of all possible data inputs are tested. Because each output bit depends on only a single input bit, 4 bit wide patterns are swept across the input. Next, the multiplier and ALU can be tested. Again, while all possible configuration inputs to the ALU can be easily tested, all possible data inputs are prohibitively large. Some operations of the ALU are tested the same as the data path, while others are tested with numbers in interesting areas. After a few general tests, the numerical boundary conditions are more thoroughly explored. The multiplier is tested in a similar manner. Following that, the link between the ALU and multiplier can be tested. Because this is simply a data connection performing no computation, the test cases which are used for the data path are sufficient. Finally, the conditional parts of the CMUX can be tested. This is a simple test because the data passing mechanisms of the CMUX are tested in the

data path tests, so all that remains is testing the conditional selection. This can be tested exhaustively.

The data path phase of the testing ensures that the components which guide data through the ALU function appropriately. These components include the muxes, shifters, and crossbar, as well as the interconnections between all the components. These tests check for paths which are stuck high, stuck low, stuck to an adjacent line, and for reversed busses. The test inputs for these configurations are generated by using each of 8 test patterns (1, 11, 101, 111, 1001, 1011, 1101, 1111) and shifting it through all possible positions in the input word, plus using the 16 repeated nibble words (111111, 222222, ..., FFFFFF). This yields roughly 200 test cases per path tested. Because test cases don't check the interaction between inputs, the number of test cases can be kept relatively small for each data path. There are 27 data paths tested here, producing roughly 6000 test cases. Each shifter is tested with the full set of data path values being shifted by one bit, producing 4000 test cases. Each shifter is then tested shifting a single bit through its full range of possibilities, producing roughly 530 thousand test cases.

A few assumptions are made in the data path testing which may make isolation of a problem difficult. It is assumed that multiplication by 1 on either input works, and that the ALU's pass X and pass Y functions work. Also, it is assumed that several data path components perform as expected before they are actually tested. These test ambiguities are unavoidable as any input must pass through several components between the input and output. This is not a problem for our project, as the goal is to determine whether problems exist, not where they are. Many of the tests will be obvious where the failure is, aiding in solving the problem, but this is not guaranteed.

The second group of tests check the Processing Element multiplier. The first group tests check using the multiplier as a shifter. They use the same 8 test patterns as the data path tests, shifted up to 47 bits left. This yields roughly 800 test cases. Following these tests, the multiplier is tested using numeric, rather than bit pattern, test cases. Three test groups, one at 0, one around the square root of the 24 bit max, and one around the 24 bit max, are used. Each of the two inputs are run through each of the three test ranges, yielding roughly 600 thousand test cases.

The third group of tests focus on the ALU. These tests must make the assumption that the data path is correctly routing the data, and thus are not run until the data path's integrity is verified. The tests also make the assumption that the multiplier works correctly, as it is used as a shifter to achieve some of the Y test values. The identity functions of the ALU (pass X, pass Y) are tested in the data path tests and are assumed to work in the ALU tests. The unary NOT operator is checked with test inputs the same as the data path. Because the test cases must be tested over the full 48 bit inputs on the ALU, this produces roughly 1000 tests for the NOT operator. The unary negation operator focuses on numerical values rather than bit patterns, and tests a group of inputs at the 48 bit negative maximum, a group centered around the 24 bit negative maximum, a group centered around 0, a group centered around the positive 24 bit maximum, and a group at the positive 48 bit maximum. Several other values scattered across the full input range are also tested. This yields roughly 6000 test cases for the negation operator. Like the data path tests, the unary operator tests benefit from a lack of interaction between inputs, keeping the number of test cases small.

The bitwise binary operators of the ALU are tested next. Each 4 bit nibble is run against each 4 bit nibble at all 44 positions, plus all the full word patterns of repeated nibbles are tested. These test cases are run for each of 6 bitwise operators, for a total of about 70000 test cases. The arithmetic binary operators are the last part of the ALU to be tested. Each of the two inputs are run through the ranges of inputs used in the unary negation test, for each of three operators X+Y, X-Y, Y-X), producing roughly 1.2 million test cases.

Once the multiplier and ALU have been successfully tested, the connection from the ALU to the multiplier can be tested. The connection from the multiplier to the ALU gets sufficiently exercised generating test inputs to the ALU that it doesn't need independent testing. The test cases passed from the ALU path to the multiplier go over the same range as the data path test cases, producing roughly 200 test cases.

The final stage of testing is the Conditional MUX. Its abilities to pass a constant value are tested in the data path section, leaving just the conditional switching to be tested. There are 8 conditions to test on each of two clippers, for 16 conditions to test. The inputs (via Mux4 and Mux5, then via the crossbar) are configured to use the two data constants, DR0 and DR1. This has the advantage that controlling input registers to create certain conditions only affects which output is selected, and doesn't change the possible output values. Hand written test cases are used to produce each of the desired conditions.

Running the full test suite runs roughly 2.5 million test cases, although when testing changes to the design, it will often require only a subset of the full test suite to be run to test that component. Preliminary timing results indicate that each test case can be run in roughly 0.2 milliseconds, requiring approximately 10 minutes to run the full test suite.

CHAPTER 5: RESULTS & CONCLUSIONS

The most important aspect of our project, the test cases, are a complete success. We have developed software which will generate a comprehensive set of tests and can be easily adapted to different interfaces. In addition, new sets of tests can easily be added, and blocks of tests can easily be removed for testing a single component of the Processing Element.

Our second task was to develop an interface between a PC and an Altera® board running the Processing Element. This also was a success, with both a hardware design for interfacing via a PC parallel port and software modules to communicate with it.

Our third task was to produce a report on the current state of the Processing Element design. Our initial set of data path tests fail on about half the test cases. When we tried to reproduce this error in simulation, the same test cases work flawlessly. All the synthesis settings in the Altera® software had no effect on our results. This leads to one of two conclusions. The first possibility is that a piece of the Processing Element design simulates well, but cannot be properly synthesized. If this is the case, then we have succeeded and saved our customer a significant amount of time and money by identifying the problem at this stage of the design, rather than it being discovered when the chip is manufactured. The second possibility is that there is a flaw in the Altera® synthesis software. If this is the case, it is out of our hands, but our testing system will still be a valuable tool for our customer once Altera® has corrected the problem. Our customer will further investigate the problem to determine whether it is a flaw in his design or a flaw in the Altera® software. The bottom line is that our project is a success and our customer is pleased with the final product.

References

Donohoe, Greg. Reconfigurable Data Path Processor Overview. Institute for Advanced Microelectronics. 2002.

Altera® Corporation. NIOS Embedded Processor Development Board. January 2002 <http://www.altera.com/literature/ds/ds_nios_devboard.pdf> (February 2002).

Altera® Corporation. NIOS Embedded Processor Software Development Reference Manual. January 2002 < http://www.altera.com/literature/manual/mnl_niossft.pdf> (February 2002).