May 16, 2006

CALVIN

MINDS IN THE MAKING

Calvin College Engineering
Senior Design 2006

Blúgate Final Design Report

Team 6

Silence Is Golden

Matt Cosnek
Nick Ellens
Jared Heys
Ryan Smith

Table of Contents

Table of Figures

Table of Tables

Executive Summary

The purpose of this design project was to design a system for silencing cellular phones within a specified area. This system was to be constructed in two parts: a client to be run on the cellular phone and a base station to activate/deactivate the client within a given proximity. The client was to be responsible for controlling the ring setting of the phone. The connection between the phone and base station was to be via Bluetooth technology. The device was to be designed to be placed in an entrance/exit. The first pass through would trigger the client to save the current ring tone settings and deactivate the ringer. The second pass through would cause the client to return the ring tone settings to their previous state.

1. Introduction

In modern society the prevalence of mobile devices is increasing almost daily. Personal Data Assistants are used by many to help keep busy schedules organized, cellular phones are almost a necessity, and newer 'smart phones' combine the functionality of both. These devices travel with their users wherever they go and often find themselves in places where an alert tone would be inappropriate – such as in a movie theater. Advances in mobile devices have also fueled advances in mobile communications. The development of Bluetooth technology -- a secure, short-ranged wireless communications protocol – provides for the first time a viable method of communication with these mobile devices. This technology is also the keystone of the BlúGate product.

1.1 Acronyms and Definitions

Table 1: Table of Acronyms and Definitions

TERM	DEFINITION
ActiveScheduler	SymbianOS method of performing asynchronous actions
Application Framework	Basic structure of a program, usually OS specific
BGA	Ball Grid Array
BlúGate	Both the product system and the client GUI
BlúGateServer	Client process running in background
BlúStation	Base station program
BOM	Bill of Materials
CAD	Computer Aided Design
C++	Object oriented programming language
DB9	Nine pin serial connector
HCI	Host Controller Interface
IC	Integrated Circuit
I/O	Input/Output
IR	Infra-Red
IP	Internet Protocol
LED	Light Emitting Diode
PCB	Printed Circuit Board
Piconet	Localized Bluetooth network
PPFS	Project Proposal and Feasibility Study
RAM	Random Access Memory
RF	Radio Frequency
RFCOMM	Radio Frequency COMMunication
ROM	Read Only Memory
RS232	Serial data communications protocol
SDK	Software Development Kit

SymbianOS	Mobile device operating system
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

2. Problem Specification

This section lays out the problem to be solved as well as the requirements that need to be met to effectively solve the problem.

2.1 Description of Challenge

The challenge is to design a process to automatically change a cell phone's ring settings using a Bluetooth wireless connection. The process shall include two parts, a base station and client program. The base station shall be responsible for continuously scanning an area for phones with Bluetooth technology. When a connection is first made with a mobile phone, the client program shall save the phones current settings and turn it to vibrate mode. When a mobile phone passes past the base station a second time, the client program shall restore the previous phone settings.

2.2 Project Requirements

This section lists the requirements for both the base station and the client program. Each list is laid out arbitrarily.

2.2.1 Base Station

The base station was broken up into two separate requirements. There are requirements for the base hardware, and there are requirements for the base software.

2.2.1.1 Base Station Hardware

1) Base station shall affect a minimum range of a 5 foot radius, and no more than 33 feet
2) The base station hardware shall be capable of communication via a Bluetooth connection
3) The base production design of the enclosure height, depth, and width shall be limited to 4" x 4" x 2"
4) The base station weight shall be limited to 1lb
5) The cost of the production design shall be no more than $100
6) The wattage of the power supply shall be limited to 10 watts for the prototype and two watts for the production design
7) The voltage output from the power supply shall be limited to five volts
8) The base station shall be meet temperature requirements of ICs
9) The base station shall operate within a specified temperature range of 0°C to 50°C

10) The base station shall have a power Light Emitting Diode (LED)
11) The base station shall meet all FCC requirements for such a device
12) A base station unit shall not disrupt operation of another base unit if they are located within range of one another
13) The base station shall be capable of withstanding the transient effects of a normal power outages, if not battery operated
14) If the device is power by batteries, the batteries shall last at least one week before needing to be changed

2.2.1.2 Base Station Software

1) The base station software shall be capable of communication via a Bluetooth connection.
2) The base station software shall be capable of connecting to 7 clients simultaneously. This is the maximum number of slaves a host can support in a piconet [18]
3) The base station software shall have the ability to be customized in production, but it shall not be accessible to the user

2.2.2 Client Program

1) The client program shall take up no more than 1Mb space in memory, based on the typical free space available on most cellular phones
2) The client program shall be capable of storing ring tone settings prior to encountering a base station
3) The program shall be capable of restoring the stored ring tone settings when leaving past a base station
4) The client program shall automatically accept the Bluetooth connection from a base station as a trusted device
5) The client program shall have the ability to be customized in production, but it shall not be accessible to the user
6) The client program shall provide the option for silent or vibrate mode
7) The client program shall be available as a stand-alone program
8) The client program shall be protected against unauthorized modification
9) The client program shall be responsible for comparing its current status with a base station signal and determining the appropriate course of action.

10) A mobile phone that does not leave a base station's range will still be silenced and restored properly
11) The phone shall not recognize the difference between different base stations at different entrances/exits. Therefore, the client shall be activated and deactivated properly by any base station connection
12) The user shall be able to manually restore their ring settings at any time in case the settings fail to restore automatically
13) The user shall also be able to disable and re-enable the client program if they desire

3. Proposed Solution

3.1 Project Management

The design team has two ways of managing the project. The college course that this design is associated with keeps the project moving in the right direction and on the right pace. Within the team, tasks have been divided among the members as the project progresses along a schedule.

3.1.1 Team Organization

Silence is Golden is a team project that is part of a year long engineering capstone course at Calvin College. The purpose of the course is to gather everything learned in a student's college career and integrate them into a project that will simulate an industrial experience. The senior engineering class was divided into teams of four or five people to work on a project of their choosing. While this course is a little more open-ended than most courses at Calvin, it still requires about a dozen deliverables to make sure each team stays on track.

Each team was assigned an industrial consultant for analyzing the team's design work. These consultants have an engineering background, but are not currently affiliated with Calvin College. Consultants provide teams with an outside perspective on how the team is progressing. Silence is Golden was assigned Tim Theriault from Smiths Aerospace. The team met with him twice over the two semesters. He was able to provide advice on topics that were a priority to the team at that point in time, as well as point out a few things that should be thought about later in the semester.

Each team was also assigned a professor as an advisor/manager for their project. Silence is Golden's advisor is Steven VanderLeest. The design team communicates with Professor VanderLeest through weekly status reports, meetings, and various other deliverables.

Silence is Golden also has organization within itself. Over the first semester, the organization was a little vague. The team would see what deliverables were due in the near future, and either schedule a time for a meeting to work on the item, or split up the work amongst the members to get done on their own. Coming into the second semester, it was necessary to define more specific roles for each of the team members. It was thought that we would divide the team and two and have one group work on the base station, while the other worked on the client software. However, as the design started to come together, it was necessary to split up the groups in terms of hardware and software. Nick Ellens and Ryan Smith chose to work on the software, while Jared Heys and Matt Cosnek worked on the Hardware. Ellens started out working together with Smith on coding for the client software. After they both got a general understanding of the coding for the client, Ellens starting working on the base station coding. Smith continued to enhance the client code. At the same time, Cosnek and Heys were working diligently on the hardware side of the design. They worked together to get a basic understanding of the parts that were used, and then Heys concentrated on designing PCBs to connect the parts for our prototype, while Cosnek concentrated on design a production model.

The team never had a rigid schedule for when to work on the project. Generally, the two groups in the team would decide when it was best for them to get together. It was found that working side by side with another member yielded more efficiency than if everyone were to work on their own. This was because it allowed members to ask questions amongst themselves and get answers right away, instead of waiting for an answer in an e-mail or seeing another team member the next day. At least once a week, the entire team would arrange a meeting in which the two groups would discuss what they had done that week, and what they planned on working on the following week. This allowed both groups to know what the others were working on, and when to expect certain aspects of the design to be finished. This time was also used to send a weekly status report to our team's advisor.

Team conflicts have also been encountered as the design process has progressed. On many occasions, team members have had different ideas on how a certain aspect of the system will work. As conflicts such as these arrive, the team sits down and discusses the matters at hand. Design decisions are discussed until all team members agree.

3.1.2 Work Breakdown Structure

In the middle of our first semester of this course, each team was asked to come up with list of tasks that would need to be completed for the design and estimate the number of hours it would take to complete the tasks. At that point in time, it was unclear as to the exact tasks that would need to be completed. However, the team came up with a list that was reasonable for what was known about the design at the time. It was decided to budget a generous amount of hours for most tasks, just in case some tasks took longer than expected. Some design decisions were made during the second semester that made some of the tasks obsolete as well as adding new tasks. At the same time that weekly status reports were sent, the team would update the task list to show where the hours for that week were spent. The completed task list can be seen in Appendix A.

There are a couple of things that can be noted about the completed task list. First of all, it becomes apparent that the team was not thought of some of the tasks that would be needed. The most noticeable is the section that was added for the base station software. This was budgeted zero hours because it was not thought of when the list was first generated. Also, the coding in general was under budgeted, as the team worked on the coding for over 300 hours and it was only budgeted 100 hours (and that was just for the client program). The individual testing was also under budgeted, but the team allotted extra hours for full system testing. This is because it ended up taking longer to get the individual parts working, and there was not much time to spend on full system testing.

Overall, the team had budgeted about 1400 hours between four people for two semesters worth of work. At the end of the first semester, the team had only logged approximately 300 hours, which left 1100 hours for interim and spring semester. The plan was that a lot of work would be done over the interim to catch up a little, and then to continue on a more regular basis for the rest of the semester. However, it turned out that very little got done of the interim due to a larger work load than expected. The team did pick up the slack though, and ended up with almost 1500 hours logged by the end of second semester.

3.1.3 Schedule

Along with the task list, the team came up with a projected schedule that the project would follow over the two semesters. This was primarily based off of the initial task list, which, as described above, was not perfect. However, it gave the team a good idea of what needed to be done, and to think about what smaller parts needed to be completed before moving on to other things. See Appendix B for project schedule.

The first semester was filled with preliminary design and the tasks associated with the start of a new design project. This included things such as creating a task list, a project schedule, and generating a PPFS. In making the PPFS, the team chose the way to pursue a solution to the problem. This included many hours of research and discussion amongst team members and various others. After it was decided that Bluetooth was the way the project would head, the team started preliminary design.

After not making much progress over interim, there was quite a bit left to accomplish in the fall semester. First of all, the team had to make up for all the things that did not get done in the interim. Not only that, but early in the fall semester it was found out that the onboard processor of the Bluetooth module that we were using would not be accessible, and that a separate processor and surrounding circuitry would be needed for the project. By this point, the team was way off the projected schedule. Also, the time it took to write the code for the project was underestimated. However, the testing and coding were integrated more than was thought. Instead of writing the whole program and then testing it, it was decided to write pieces of the code at a time, and testing the functionality of the code as we added it. The prototype hardware was designed to be integrated with the Coldfire development board that was used by a previous senior design team. This process yielded several different PCBs to make it compatible.

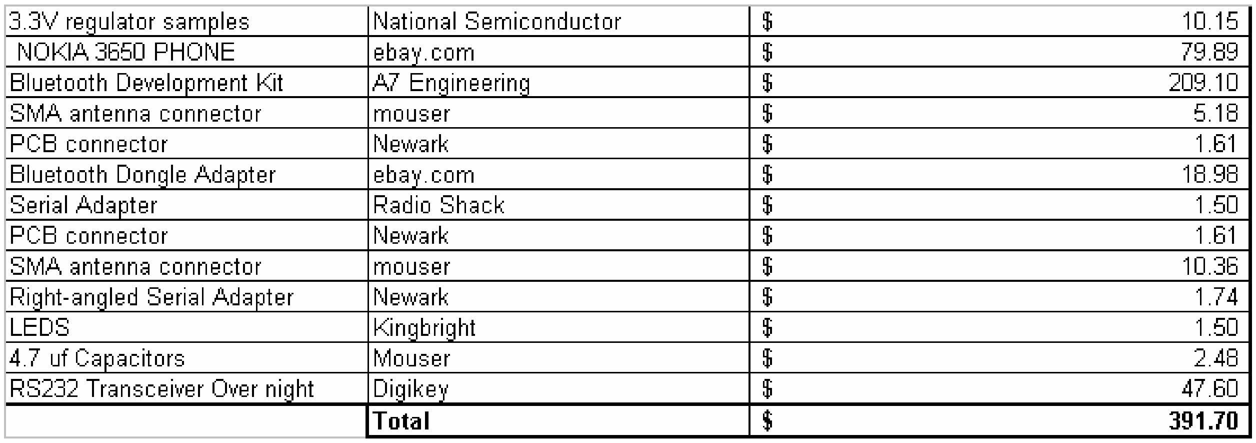
And finally at the end of the semester the team put together the working pieces of hardware and software to be able to show a prototype of or design. The design is not complete, but we were able to communicate between the cell phone and a laptop by using the Bluetooth circuit we designed. The final week was spent generating final documentation for the project.

3.2 Project Budget

3.2.1 Prototype

Initially the team was given a budget of $300 for the project and prototype. This budget was later increased to $500 after some initial parts were ordered. Many of the parts needed for the prototype module were found in the electrical engineering labs on campus. The Coldfire development board that was used to test and simulate the base station design was available to the team free of charge because a senior design team had purchased and used it two years ago. To stay within the $500 budget, the team used as many parts that could be found on campus as possible, and the team searched many places for the best price for an item. The cost of the Bluetooth development kit was split with another senior design team that also needed it. The team received an educational sponsorship from PCB express that amounted to $350 of PCB production. A summary of the project budget is shown in Table 2 below.

Table 2 Project Budget Summary



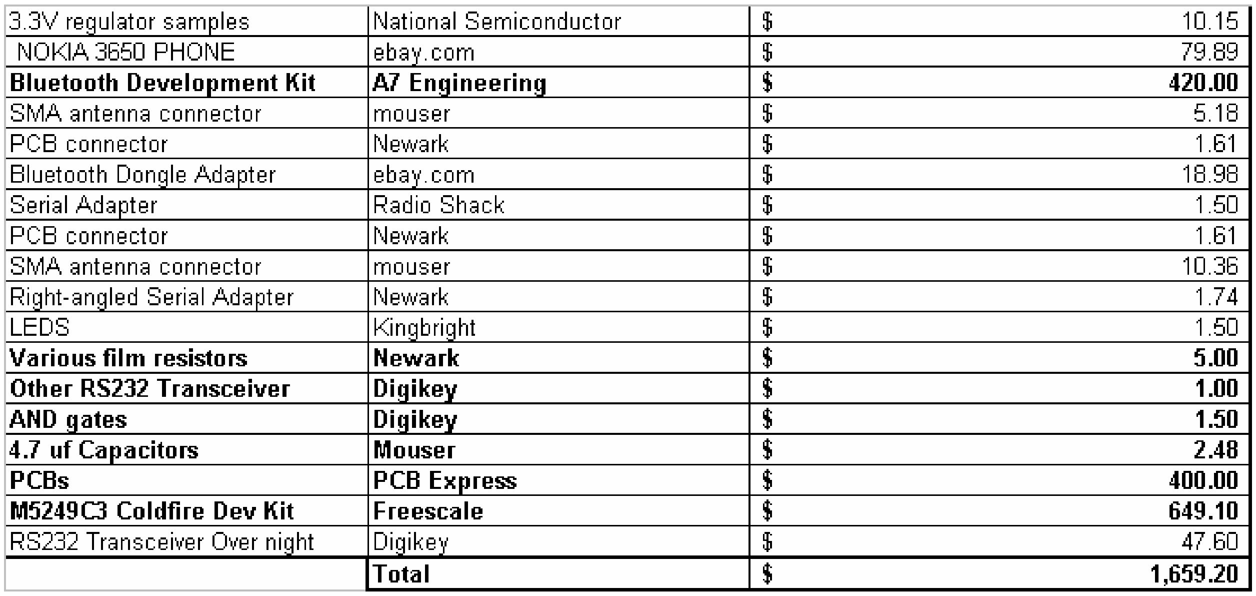
3.3V regulator samples	National Semiconductor	$	10.15
NOKIA 3650 PHONE	ebay.com	$	79.89
Bluetooth Development Kit	A7 Engineering	$	209.10
SMA antenna connector	mouser	$	5.18
PCB connector	Newark	$	1.61
Bluetooth Dongle Adapter	ebay.com	$	18.98
Serial Adapter	Radio Shack	$	1.50
PCB connector	Newark	$	1.61
SMA antenna connector	mouser	$	10.36
Right-angled Serial Adapter	Newark	$	1.74
LEDS	Kingbright	$	1.50
4.7 uf Capacitors	Mouser	$	2.48
RS232 Transceiver Over night	Digikey	$	47.60
	Total	**$**	**391.70**

The cost of the Bluetooth development kit was $400. It was split evenly between the two teams that needed it. The prototype mobile device used was a Nokia 3650 cellular phone. This phone was purchased used and without a plan on eBay for $80.00, which is significantly less than the retail price. Two RS232 transceivers were purchased at $8.80 apiece. Due to the late recognition of their necessity, an overnight shipping charge of $30 was also added to the cost of these modules, bringing the total to $47.

The Coldfire development kit that was used, free of charge, also helped keep the project under the budget. On the manufacturer's website the price of this kit is listed as $649.10 [21]. Calvin College has limited PCB manufacturing capabilities on campus that is free for the students to use. The amount of PCBs that were made in house by the team would have cost about $400 if ordered from PCB express.

Without these major donations and sponsorships the budget for the team would have had to be considerably higher. In Table 3 below, a full cost budget summary for the entire project including the prototype, is shown.

Table 3 Full Cost Project Budget



3.3V regulator samples	National Semiconductor	$	10.15
NOKIA 3650 PHONE	ebay.com	$	79.89
Bluetooth Development Kit	**A7 Engineering**	**$**	**420.00**
SMA antenna connector	mouser	$	5.18
PCB connector	Newark	$	1.61
Bluetooth Dongle Adapter	ebay.com	$	18.98
Serial Adapter	Radio Shack	$	1.50
PCB connector	Newark	$	1.61
SMA antenna connector	mouser	$	10.36
Right-angled Serial Adapter	Newark	$	1.74
LEDS	Kingbright	$	1.50
Various film resistors	**Newark**	**$**	**5.00**
Other RS232 Transceiver	**Digikey**	**$**	**1.00**
AND gates	**Digikey**	**$**	**1.50**
4.7 uf Capacitors	**Mouser**	**$**	**2.48**
PCBs	**PCB Express**	**$**	**400.00**
M5249C3 Coldfire Dev Kit	**Freescale**	**$**	**649.10**
RS232 Transceiver Over night	Digikey	$	47.60
	Total	**$**	**1,659.20**

The items that were adjusted or added are in bold so that they can be easily seen and compared to the actual project budget. Table 3 shows that the actual amount for the project without subsidization would have been almost $1700. Through creative purchasing, donations and using the available on campus resource, the team was able to reduce the cost to just under $400; well within the $500 budget the team was given.

3.2.2 Production

If base stations were to be marketed, the production of base stations shall stay under the projected cost of 100 United States Dollars (USD). The production includes all parts, labor, and distribution.

3.2.2.1 Estimate Cost of Parts, Labor, Distribution

The cost of parts can be seen in the Bill of Materials in section 3.4.4.4. An important part that is left out of the BOM is the enclosure. The cost of the enclosure was estimated at $0.75 per base station. This estimate was based on information from Team 11, who had purchased a plastic enclosure for their project. For labor it was assumed that it would take 20 minutes to hand craft a base station and get the software loaded on. If an employee is paid $12.00 per hour, then it costs $4.00 per base station for labor. As for distribution, a third party would be paid to transport base stations to customers. United States Postal Service can ship in 11” x 8.5” x 5.5” boxes for $8.10. Most customers will probably buy base stations in quantities of four or more. Four base stations could easily fit in an 11” x 8.5” x 5.5”, so the price per base station to transport would be no more than $2.03 USD.

3.2.2.2 Final Retail Cost

Using the BOM section and the costs for labor and distribution, base stations will cost $60.90. According to market surveys, however, the retail price would be much higher, see Figure 1.

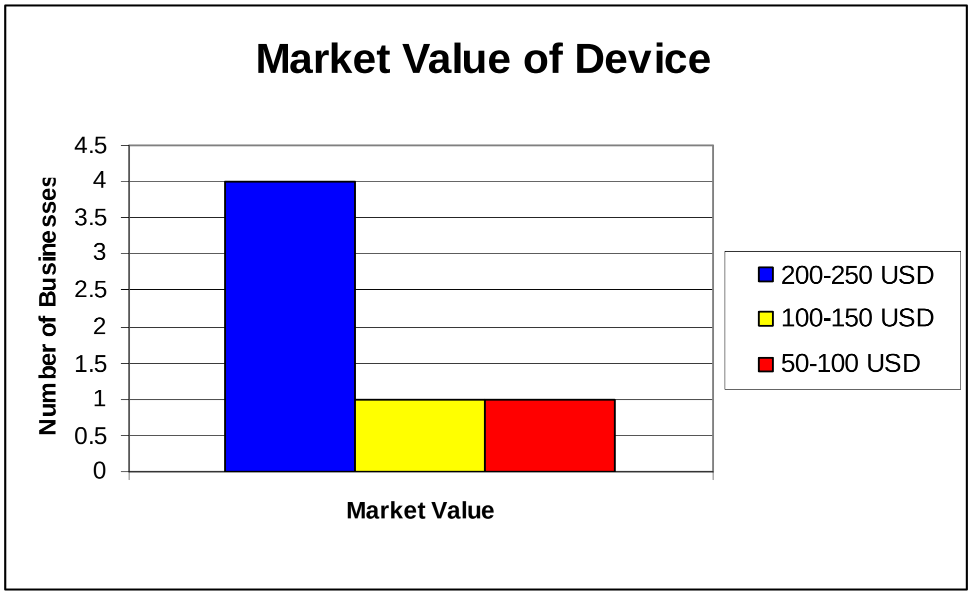


Figure 1: Market Value of Base Station

According to Figure 1, base stations would probably be sold in the range of 200 to 250 United States Dollars (USD). These prices do not take into account competition, so the actual retail price may be a bit lower depending on the level of competition.

3.2.2.3 Estimate Cost of Parts, Labor, Distribution

3.2.2.4 Final Retail Cost

3.3 Preliminary Research

There were several areas of research. First, similar products were researched to determine the project's feasibility and to ensure no copyrights were infringed. As stated before, the two major components of the system are the base station and the client program running on the mobile phone. In order to make educated design decisions, research was done for the different components involved in the construction of the base station. The base device and the mobile phone need to connect to each other. Therefore, various connection methods were researched as well.

3.3.1 Similar Products

The idea of silencing cellular phones in certain areas is not a new concept. In fact there are other products that already exist or are in development. Two companies in particular have developed, or are in the process of developing specific products to silence cellular phones in similar ways as the proposed solution.

Cell Block Technologies, Inc. has developed a product (patent-pending as of 2000) that achieves the desired goal, which is to silence cellular phones in a limited area [1]. The way that they have chosen to do this is by using "Quiet Cell" technology which they developed. This consists of a small device about the size of a smoke detector called the "Quiet Cell Control Unit." This device is recognized by cellular phones at a radius of about 2 meters (about 100 square feet of floor space). Once the device is recognized, it tells the phone to change to a channel that is not active, which keeps the phone from sending or receiving information from its original base station (tower). It is also possible to get control units that cover a larger area, up to 10,000 square feet. In the United States, however, this technology is considered a cell phone jammer, which is illegal. Because of this, Cell Block Technologies is concentrating their efforts outside of the United States, in places where the technology is not considered illegal. This device is not available to small volume retail customers. Their primary market consists of prisons, religious institutions, and foreign embassies [1].

BlueLinx, a North Carolina-based corporation, is also developing a way to silence cellular phones in a confined area [2]. They do this through the use of Q-Zone technology. Q-Zone technology consists of a small device that communicates to cellular phones in a predetermined area through a Bluetooth connection. Since Bluetooth is not on most cellular phones in use today, the device is not commercially available, and only the general concepts are released to the public. The technology has two parts: the base station and the software on the phone. The base station sends out a Bluetooth signal to all Bluetooth enabled devices in its range. The range is not strictly specified, but they claim to be able to cover sizes from small conference rooms to a multiple screen movie theatre. This area is called the Quiet Zone. When a Bluetooth device enters this zone, the Q-Zone device initiates a brief communication with the Bluetooth device telling it to be quiet. The Bluetooth device will then shift to quiet mode, based on the type of device (i.e. PDA, cellular phone, etc.). For example, one phone might turn to vibrate mode, while another would just turn down the volume to a very low level. As long as the mobile phone is in the Quiet Zone, these settings will hold, but upon leaving the area, the original settings will be restored. The goal behind the Q-Zone technology is not to interrupt service to the phones, but to reduce or eliminate the distractions that could be caused by these devices. Since the communication is through Bluetooth, only devices with this technology and the software can be controlled. The software is currently only available on a royalty-free licensing to handset manufacturers.

BlueLinx also holds a patent for “politeness zones for wireless communication devices.” [3] This patent seemed to be exactly what the Silence is Golden team was planning on implementing. However, upon closer inspection of the patent it was noted that the “politeness zone transmitter” was defining the politeness zone. Team 6’s idea is to make a device that acts similar to a gate; that is when a cellular device passes in proximity of a base station, the settings will be changed to a silent or vibrate mode. The device will remain in the silent or vibrate state until it passes through the proximity of a base station for a second time, or a time limit has been reached.

3.3.2 Base Station

The major components involved in creating the base station, were a Bluetooth Integrated Circuit (Bluetooth IC), and the surrounding circuitry. The surrounding circuitry involved research into processors, memory, and supporting components like resistors and capacitors.

3.3.2.1 Bluetooth IC

Research for the base device primarily consists of researching a Bluetooth IC. Initially a goal of the project was to design as much of the circuit as possible, but research into the operation of Bluetooth and the components required to implement Bluetooth found that it would be nearly impossible to design the circuit at this level. The operation of Bluetooth requires many components that had to interact in a very precise way at high frequencies. Some of these components are a microcontroller interacting with a memory management system, and code from memory executing and sending data to radio components that are operating at 2.5GHz. In light of this information, the focus shifted from designing a complicated

Bluetooth IC from scratch to designing an IC that had Bluetooth and most of the essential components included. It was not difficult to find these types of ICs, but most of the manufacturers provided limited datasheets and information.

The BlueCore2 IC was found at a reasonable price of $54 for a quantity of five [4]. BlueCore3 and BlueCore4 ICs are also available at variable prices. The specifications concerning the decision between these ICs are package size, price, and whether the IC has the ability to run software without an external host. CSR is the manufacturer of this IC, and their support website provides a lot of helpful information [5]. The site provides in depth datasheets, application notes, example circuit designs, and much more.

Other ICs were also researched to compare to the BlueCore2 IC. A decision matrix is shown in Table 1. The three Bluetooth modules selected are the BlueCore2 External, the LMX9820A from National Semiconductor, and the CXN1000 from Sony.

Table 4: Decision matrix for Bluetooth IC

Device	Package	Features	Price	Availability	Documentation	Total
Weighting	20	15	5	25	35	100
BlueCore2 External	5	12	4	23	30	74
LMX9820A	2.5	11	2	10	20	45.5
CXN1000	10	14	0	0	12	36

The first criterion in the decision matrix was the package, which included the type, size, and pin configuration of the IC. The weighting for this criterion received a 20 because a good package allowed the IC to be mounted easily to the circuit board. The BlueCore2 External received a 5 because the package was very small with a 96 pin ball grid array configuration [6]. The LMX9820A received a 2.5 because it was available in a package very similar to the BlueCore2 External, but it had 116 pins [7]. The CXN1000 received a 10 because it was available in a slightly larger package with 36 pins [8].

The second criterion was the features included on the IC. This criterion has a weighting of 15 because more features provided an easier design. The BlueCore2 External received a 12 because it included the essential Bluetooth components, and it allowed software to be executed without a host [6]. An 11 was given to the LMX9820A because detailed information of the features was not found, but it was advertised to include all the necessary Bluetooth components [7]. The CXN1000 received a 14 because it had all the features of the BlueCore2 External, but it also included a radio filter and voltage regulators onboard [8].

The price of the IC was the third criterion. This received a weighting of 5 because all the other factors were more important than the price. The BlueCore2 External was available in quantities of five for $54. This price fit within the provided budget. Therefore, it received a 4 in the matrix. The LMX9820A was available at about $35 each from Arrow Electronics. This fit in the team's budget as well but is much more expensive than the BlueCore2 External, so it received a 2. A price for the CXN1000 could not be found and therefore it was given a 0.

The availability of the IC was also an important criterion in the decision matrix. If the IC was not available then it couldn't be used, so this criterion received a 25. The BlueCore2 External received a 23, because there was no lead time specified at the given price [4]. There was a six week lead time at Arrow Electronics for the LMX9820A. This was not a major problem because if the decision were made to purchase this device the module would still arrive in time to implement. However, if something went wrong and another LMX9820A needed to be purchased later, the lead time could have presented a problem with the May deadline. For these reasons, the LMX9820A received a 10 for this criterion. Since no price was found for the CXN1000, the IC did not seem to be available in the US, so it was given a 0.

The last and most important criterion was the documentation provided with the IC. This criterion received a 35 because the documentation provides the team with the information needed to design using the particular IC. The BlueCore2 proved to be the best choice on documentation because the datasheets that were provided clearly outlined four modes of operation that pertain to software. One mode allows software to be run entirely onboard without the need for an external host. This means that the BlueCore IC can operate as a processor to execute the client program written by Team 6 [6]. The BlueCore2 External's manufacturer provided vast amounts of information for its products, but there was little information found that describes the software execution of the IC. These reasons gave the BlueCore2 External a 30 for documentation. National Semiconductor did not give a free detailed data sheet for the LMX9820A, but it looked like more information was available to members of the Simply Blue development page. Since there is probably a lot of information that the team cannot access without purchasing the IC, the LMX9820A received a 20 for the documentation. The CXN1000 had a 24 page datasheet that had some good information, but no other information was found relating to this product. Therefore, the CXN1000 received a 12 for the documentation.

The weighted totals in the decision matrix show that the BlueCore2 was clearly the best choice for the design team to purchase. As can be seen from table 1, documentation was the leading factor in the decision matrix. This was a major reason for considering the BlueCore2 IC over the other options. The remaining weighted totals in the decision matrix showed that the BlueCore2 was clearly the best choice for purchase.

The research for the remaining hardware components revolved around the selection of the Bluetooth IC. Since the BlueCore ICs looked like the best choice, research continued, so that other components needed to operate the BlueCore IC could be found. Manufacturers were searched for voltage regulators, resistors, capacitors, antennas and filters. These components depended on the final circuit design, so the next step was to create a final schematic for the prototype.

3.3.2.2 Supporting Hardware Components

It was later discovered that the BlueCore IC would be unable to work without supporting circuitry. Although the BlueCore IC had an onboard processor and memory, it was impossible to access these devices without the assistance of expensive equipment. In order to use the BlueCore IC for prototyping and production, more hardware would be needed. This included a microprocessor, flash Read Only Memory (ROM), and Random Access Memory (RAM). The major issue was finding components that were compatible with each other.

3.3.2.2.1 Prototype

To address the problem of prototyping, a coldfire development board from a previous design team was investigated. It was discovered that the development board could perform all the needed functionality to support the BlueCore IC. Due to lack of funds and time, the coldfire board was the only choice to use with prototyping, so a decision matrix wasn’t even constructed.

Another challenge presented was interfacing the BlueCore IC with the coldfire development board. The BlueCore IC package came with a Universal Serial Bus (USB) development board that could be used with a USB port. The USB development board was used with a computer and a laptop to simulate the base station software, however, the coldfire board did not have a USB port. The interfacing mechanisms available on the coldfire board were: a serial port, a data bus, and I^2C. The BlueCore IC didn’t have the ability to use I^2C, so only the serial port and the data bus could have potentially been used. A decision matrix was used to find out the best means of interfacing the BlueCore chip, see Table 5.

Table 5: Decision Matrix for Interfacing Bluetooth IC

Device	Compatibility	Familiarity	Ease	Documentation	Total
Weighting	20	25	30	25	100
Serial	20	25	20	25	90
Data Bus	15	10	15	10	50

The first criterion was compatibility. Compatibility encompasses the devices ability to even work with the BlueCore IC. It was discovered through research that the BlueCore IC could be integrated by means of a Universal Asynchronous Receiver/Transmitter (UART) controller. This is also known as serial transmission, so it was deduced that the BlueCore IC was fully compatible with a serial port. The serial port got the full 20. There was less information regarding the data bus, but since the data bus is used to control every device on a computer, it was assumed the BlueCore IC was compatible. The data bus was given a 15.

The second criterion was familiarity. Everyone on Team 6 was pretty familiar with serial ports and how they work. So the serial port got a 25. As for the data bus there was less familiarity, so it got a 10. Ease of integration was the third criterion. It was assumed the BlueCore IC would be hard to integrate with both devices. The serial port got a 20, because a RS232 plug could be purchased, to put into the serial port on the coldfire board. If the data bus was used, wires would have to be attached directly to the board itself, so the data bus got a 15.

The last criterion was documention. There was a lot more documentation regarding the use of the serial port. In addition, UART and RS232 are standardized, so it got a 25. There was some documentation in the coldfire manual on the data bus, but online documentation was a little confusing. The data bus received a 10.

A picture of what the prototype might look like is shown in Figure 2. The attachment on the left of the figure is the BlueCore IC interface plug into the serial port.

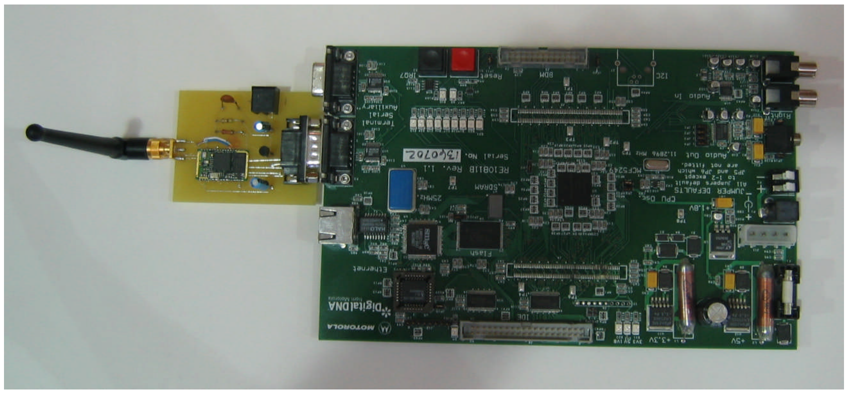


Figure 2: Desired Coldfire and BlueCore IC Prototype

3.3.2.2.2 Production Model

Although funding permitted the full implementation of a base station, it was still important to create a model with parts that could function together. The most important component to find was a microprocessor, and a decision matrix was created, see Table 6. The microprocessor is responsible for controlling everything in the integrated system.

Table 6: Decision Matrix for Microprocessor

Microprocessor						
PART	**Price**	**Ease of Integration**	**Compatibility**	**Speed**	**Documentation**	**Total**
	20	**10**	**40**	**10**	**20**	**100**
MCF5249LPV140	18	10	40	5	15	88
NG80960JT100	15	8	20	3	20	66
MCF5249LPV120	20	8	35	5	15	83
MCF5470ZP200	12	10	35	10	15	82

According to the decision matrix the MCF5249LPV140 microprocessor was chosen for the production design. Interestingly, this was the ColdFire processor that was on the development board used in prototyping. The major design criterion was compatibility. Since we knew the BlueCore IC could work on the development board we knew that the processor was compatible. Therefore, the MCF5249LPV140 microprocessor got a 40.

The next component that needed to be found was the flash ROM. The flash ROM would be the medium where the base station software would be stored. A decision matrix for the flash ROM can be seen in Table 7.

Table 7: Decision Matrix for Flash ROM

Flash ROM					
PART	**Price**	**Ease of Integration**	**Size**	**Documentation**	**Total**
	40	**20**	**10**	**30**	**100**
M28W320CT90N6	25	20	10	30	85
LHF00L12	23	18	10	5	56
M29F032D70NGT	12	10	10	30	62
JS28F640P30B85	15	20	10	10	55
AT49BV160C-70TI	40	20	5	30	95
AT49BV320-11T1	2	20	10	30	62

All of the flash ROM's in the decision matrix were compatible with the MCF5249LPV140 microprocessor. So compatibility was not a major issue. The amount of storage was directly dependent to the size of the base station software. Since the base station software was small, the size of the storage was not important either. The most important component that went into flash ROM was price and documentation. The AT49BV320-11T1 flash ROM did well in both of those categories, and it ended up being the best choice based on the decision matrix.

The final major component for the supporting circuitry was the RAM. The RAM allows for fast access to the base station program. A decision matrix is shown in Table 8.

Table 8: Decision Matrix for RAM

SDRAM							
PART	**Price**	**Ease of Integration**	**Size**	**Speed**	**Documentation**	**Lead Free**	**Total**
	30	**25**	**15**	**15**	**15**	**Yes = 1**	**100**
MT48LC4M16A2P-75:G TR	30	25	12	15	13	1	96
MT48LC4M16A2TG-75:G TR	30	25	12	15	13	0	95
MT48LC8M16A2P-75:G TR	20	25	15	15	13	1	89
MT48LC8M16A2TG-75:G TR	20	25	15	15	13	0	88

Similar to the flash ROM the price was the most important criterion. Ease of integration was also important, but all four parts could be easily integrated with the other components. Two devices actually tied, so the deciding factor was the fact that the MT48LC4M16A2P-75:G TR RAM was lead free. Lead can pose it poses toxic effects to the environment, as well as be a harmful poison to human beings [7]. Therefore, it was ethically appropriate to choose the lead free device.

3.3.3 Mobile Phone

The client program will run on a mobile phone. Therefore, in order to properly prototype the client program, a compatible mobile phone must be found. The mobile phone research was focused in two areas: cellular device and software development kit. The cellular device must be researched to determine if it can connect the base station, and if it has the capabilities of running the chosen program. This also means it must be compatible with the chosen programming language. Secondly, available SDK's must be researched to ensure that the programming language used can be used on the cellular device.

3.3.3.1 Cellular Device

In order to use a mobile phone in the proposed system, the phone had to meet a few important requirements. Most importantly, the phone had to have Bluetooth Technology in order for it to communicate with the base station. Secondly, a programming language that can be used to implement the client program must also be available on the phone.

There were several programming languages that could have been used to control the cell phone. However, the programming language chosen directly impacted the type of phone that could be used. The three programming language candidates were Java, C++, and Python, which is very similar to C++. The most promising language was Java, because it was more familiar to the team than Python, and it was believed it would provided the desired functionality to implement the client program. Also, since the program was to be optionally downloaded by a user, then Java had an advantage over the other languages. Java games and other programs can already be downloaded on Java-enabled phones, so downloading a Java programming to a Java-Enabled phone was very feasible. If the program was written in Java, the phone being used must be Java-enabled. Java-enabled phones are new, so cost becomes an important factor. Also, it was found out that Java did not provide easy low level access to phone settings.

A less powerful alternative was C++. C++ is not a native programming language for a large number of phones that are on the market today, but is limited to smart phones. Also, unlike Java, C++ provided low level access to phone settings. C++ was the most familiar programming language to the design team, so it would require a minimal learning process. Another programming language that is based of C++ was python. Python was unfamiliar and would require extra time to learn the programming language. Like C++, though, Python would not require a Java-enabled phone.

Bluetooth technology, the type of programming language, and cost were the most critical requirements when purchasing a cellular phone for testing, but there were other properties that were researched as well. The amount of free memory available on the phone had to be considered, even though most phones provided plenty of memory for storage. The method of transferring the program to the phone was also an issue. Most phones, like the Java enabled phones, can transfer programs and information via a Universal Serial Bus (USB) cable. Finally, the program interaction with the phone was considered. It was important that the functionality of the phone (namely the ring settings) could be accessed. After meeting with our consultant, Tim Theriault from Smiths Aerospace, it was determined that we should meet with several cell phone vendors to discuss how the settings on the phone can be saved and how the phone can/will run the client program.

This advice turned out to be very valuable. Very few phones would allow third parties to access the lower functionality of their phones, like ring settings. One of the few phones that would allow access to the ring functionality was the Nokia 3650. The Nokia 3650 ran on the Symbian Operating System (OS), which was well documented online. The Symbain Operating System also allowed for a free downloadable development environment. The Nokia 3650 also had Bluetooth Technology and supported the C++ programming language. Table 9 shows a design matrix that assisted in deciding the mobile phone.

Table 9: Decision Matrix for Mobile Phone

Mobile Phone							
PART	**Price**	**Programming Language**	**Memory**	**Bluetooth Enabled**	**Method of Transfer**	**Feature Access**	**Total**
	10	**10**	**5**	**40**	**5**	**30**	**100**
Nokia 3650	10	10	1	40	3	25	**89**
Motorola MPx220	1	8	5	40	5	5	**64**
Motorola RAZR V3	2	8	3	40	4	5	**62**
Nokia 3660	4	10	1	40	3	25	**83**

These cell phones had to be compatible with a Software Development Kit (SDK). SDK's will be discussed in the next section.

3.3.3.2 SDK Research

Researching and deciding on a Software Development Kit (SDK) was directly dependent on the type of cell phone that was used. A lot of SDKs are designed specifically for one cell phone model. Therefore, it was important to know what type of cell-phone as well as the SDK that was to be used in prototyping.

In the case of the Python mobile phones, the most promising SDK was the Python Series 60 SDK [13]. Only Nokia series 60 phones can be used with this SDK [14]. The SDK could provide all the functionality that was needed, as long as the phone was a series 60. The Python Series 60 SDK is also free, so that eliminates an item that would have to be purchased

In the case of C++, the Symbian OS C++ SDK was to be the software development kit. Like the Python Series 60 SDK, the Symbian OS C++ SDK can provide all the functionality needed for the program.

The SDK that would have been used with a Java phone would have been the Java 2 Standard Edition (J2SE) [15]. Java 2 Platform, Micro Edition (J2ME) can be used to provide the environment to run J2SE on the cell phone. J2ME was designed to work on cell phones, plus other packages could have been downloaded to implement new technologies like Bluetooth [16]. J2ME and J2SE are both free to download.

As stated before, the chosen phone would directly impact which SDK could be used. Since the Nokia 3650 was determined as the best phone for prototyping, the Symbian OS C++ SDK was chosen by default.

3.4 Design

3.4.1 Design Considerations and Criteria

When creating any product, there are many facets of design that must be considered and, as with any problem to be solved, these must be approached from various angles. The design of the BlúGate system is no exception. The most prevalent consideration throughout the design process was that of minimal user interaction. The usefulness of the system stems from the removal of responsibility from the user. Keeping this in mind, the issue of invasion of privacy is called to mind. Does a system have the right to manipulate the personal mobile device of an individual? This question was kept in mind when designing the basic functionality and interactions of the system. The size and appearance of the production base station model was also a consideration, as the appearance of a product can sometimes determine its marketability regardless of functionality.

3.4.2 Design Alternatives and Analysis

The team had limited ability to design and simulate alternate designs. The preliminary research in section 3.3 above showed the complexity of the Bluetooth ICs and that they are very expensive. All of the alternative Bluetooth ICs would have been impossible for the team to solder, since a typical IC contains about 96 pins in a BGA in under one square centimeter. Also, these IC would have required an even more expensive development kit. For example, the CSR Bluetooth ICs were the best option for the team to use, but this IC required a $3000 development kit[22][23]. A7 Engineering provided the best solution for the team's project, which was an affordable development kit that included the Bluetooth ICs on a solder able breakout.

The alternate connection methods would have been too expensive for the team to test and simulate. The Bluetooth connection method was decided based on decision matrices in section 3.4.3 Based on the preliminary Bluetooth research, there were no reasonable alternatives to the A7 Engineering breakout board. However, alternatives were considered for other components of the prototype. The power supply for the prototype was originally designed to be battery powered. To stay within the project budget, the team placed cost at higher priority than power efficiency.

3.4.2.1 Testing

The battery power supply is shown in Figure 3 below. This circuit contained a low battery indicator LED. The team decided that the base station required a low battery indicator so that the user would know when the battery needed to be changed.

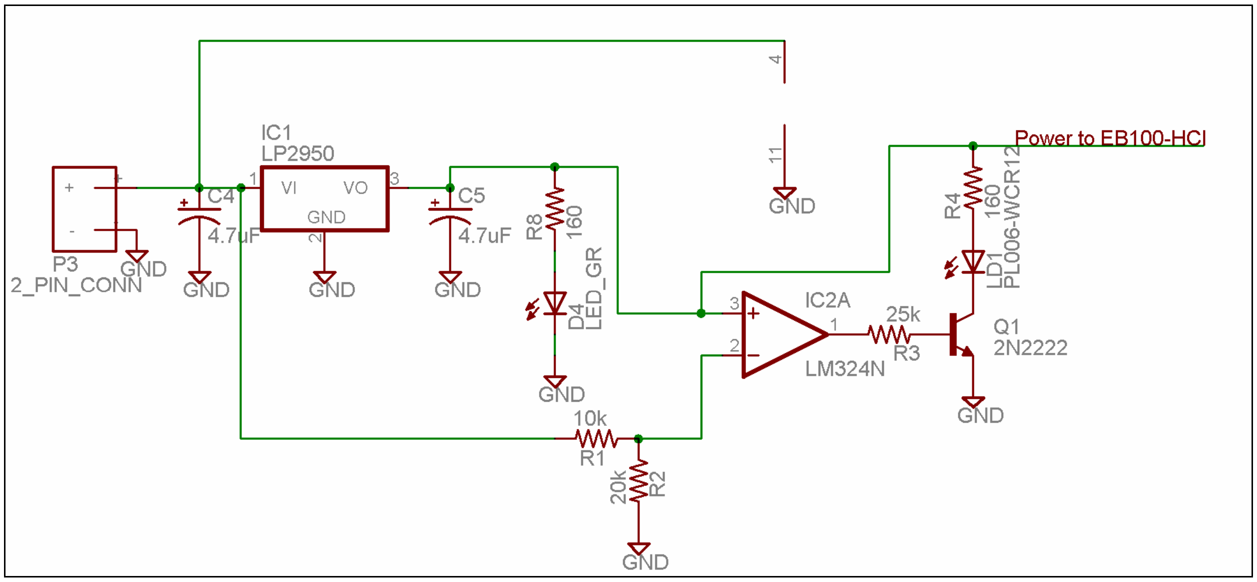


Figure 3 Battery Power Supply Circuit

This circuit contains a simple comparator to monitor the battery power and turn on the LED when the battery voltage drops below a certain value. The op amp is powered by the unregulated battery voltage. The green LED, D4, is the power LED that indicates when power is supplied to the base station. The red LED is the low battery LED. It turns on when the battery voltage drops below 5V. This allows the red LED to be on for a little while before the circuit shuts down due to low battery voltage. The 3.3V regulator, IC1, can operate down to just over 3.3V input. The voltage divider, R1 and R2, was configured to provide 3.3V to the negative input of the op amp when the battery voltage is equal to 5V. When the battery voltage drops below 5V, the voltage divider will provide a voltage less than 3.3V. This will cause the op amp comparator to output the same voltage that is currently coming from the battery. Resistor R3 was chosen to cause transistor Q1 to be saturated when the op amp is outputting high voltage. The transistor in saturation has a near zero voltage drop from collector to emitter. Resistor R4 was selected to limit the current through the LED to 10mA so that the LED will function properly. This circuit was not be used because too much power is required for the components used in the final circuit. A battery would not last long enough to be a practical power source for the prototype. See section 3.4.2.2.

3.4.2.2 Hand Analysis

The power requirements for the prototype circuit did not allow for sufficient battery life to make battery power a good choice. The total current for the prototype circuit powered by the battery circuit was about 60mA. Table 10 below, shows how the total current was calculated.

Table 10 Total current in battery powered prototype

Part	mA
4 LEDs	40
opamp output	0.172
opamp supply	1
Regulator quiescent	5
EB100 input	3
Transceiver supply	10
Voltage divider	0.3
Total	**59.472**

A typical 9V battery has an operating life of 8 hours at 60mA [24]. This is not an acceptable operating time because the team specified in the project requirements that the base station should operate for at least two weeks with a single 9V battery if batteries were selected as the power source.

3.4.3 Decisions

This project required many design decisions for both the hardware and software sections. In addition to the decisions for the design, some decisions were made for the prototype to ease development.

In an effort to determine the best method of connection for the design, a decision matrix was made to weigh the different options against the criteria, see Table 11. The criteria were selected to reflect the most important aspects being sought in the connection method. The amount of user interaction required was the most important factor. The connection method should require the least amount of work on the part of the user. Due to this, the user interaction criterion was weighted at 60 percent. The four other criteria were considered to be of equal weight because they could all be worked around if the other criteria prove to be strong. In the decision matrix, higher numbers demonstrate better performance.

Table 11: Connection Method Decision Matrix

Device	User Interaction	# of Connections	Implementation complexity	Speed	Availability	Total
Weighting	60	10	10	10	10	100
Infrared	20	2	10	5	10	47
Docking Station	10	2	5	5	10	32
Cellular Network	5	10	2	5	10	32
Bluetooth	60	8	5	8	5	86

Infrared (IR): The user interaction of this connection method received a 20 because it required the user to hold the device's IR port in line of sight. IR is capable of only one connection at a time, lending this criterion a two. There was, of course, the possibility of using multiple IR ports, but this would still require too much user interaction for our application. IR is a simple standard of communication that has been used for many years and would be easy to implement in hardware. The complexity of implementation was therefore given a 10. The speed with which this connection could be created was dependent upon the user presenting their phone and properly aligning it. This lengthy process, from the perspective of modern electronics, gave IR speed a value of five. Since IR has been used for many years, the components for its implementation are easily obtained and it is used on many modern phones. The IR method therefore received a 10 for this category.

Docking Station: The user interaction of this connection method received a 10 because it required the user to place their phone in a docking cradle – arguably a difficult task. The docking station would be capable of only one connection at a time without becoming cluttered, lending this criterion a two. Designing a docking station that would accept all models of cellular phone would have been extremely difficult, as each phone manufacturer has a different connector design. The complexity of implementation was therefore given a five. The speed with which this connection could be created was dependent upon the user presenting their phone and properly inserting it into the docking cradle. This lengthy process, from the perspective of modern electronics, gave the docking cradle speed a value of five. Since docking cradles have been used for many years, the components for implementation are easily obtained as all phones are designed for some form of cradle interface. The docking station method therefore received a 10 for this category.

Cellular Network: The user interaction of this connection method received a five because it required the user to provide their personal cellular phone number, as well as accept any call or data transfer sent to the phone from the base. The cellular network is capable of contacting all of the required phones at the same time, lending this criterion a 10. The cellular network is well established technology. This method would require hardware to interface with this network and contact the phones. However, creating the capability to change settings on the phone via this network would present insurmountable challenges, as this form of interaction would infringe upon proprietary phone features. The complexity of implementation was therefore given a two. The speed with which this connection could be created is dependent upon the cellular network and the speed with which the user accepted the incoming call. This lengthy process, from the perspective of modern electronics, gave the cellular network speed a value of five. Since every cellular phone requires use of the cellular network, availability receives a 10.

Bluetooth: The user interaction of this connection method received a 60 because it required no user interaction. Bluetooth is capable of up to seven simultaneous client connections, lending this criterion an eight. Bluetooth is a relatively new technology, but is also relatively contained. The Bluetooth standard is contained within its IC implementations. Due to this, the hardware design needed only to incorporate this IC. The IC packages, however, are difficult to interface without sophisticated component mounting equipment. The complexity of implementation was therefore given a five. The speed with which this connection could be created was dependent upon the paging time of the Bluetooth connection – 1.804

seconds. [17] This is the fastest connection speed that could be expected from any of the considered connection methods and was given a value of eight. Since Bluetooth is still a relatively new technology, it is not on every cellular phone on the market today. Bluetooth has been well received, however, and is being included in most of the new phones being produced. Bluetooth has strong potential for the future, but due to its current lack of availability, this criterion received a five. Based upon the decision matrix, Bluetooth was chosen as the connection method.

While the initial scope of this project specified Java as the programming language for the client program, further research done after the writing of the specifications document showed MIDP, the Java protocol for mobile devices, to have only superficial hardware control. Due to this, a new programming language was needed, and C++ provided this. The initial intent with choosing Java as the coding language was to allow for maximum portability between various mobile platforms. The switch to C++ reduced this cross-portability, and a cellular phone had to be found that would allow access to lower level functionality through user-written C++ programs. The phone that met the criteria and was also not too expensive for the prototyping budget was the Nokia 3650 running on the SymbianOS S60. Unfortunately, a side issue to coding in C++, and to accessing such low-level functionality as ringer settings, is the loss of generalization. The code for the prototype is specialized to the device for which it was developed. Therefore more changes, and in some case significant changes, may be required to port the program to other mobile devices.

The base station functionality was also initially specified to be written in the Java programming language. The Bluetooth chip that was selected had a virtual machine built in and processing capabilities. However, the breakout board that was chosen, the EB100 form A7 Engineering, did not allow access to this functionality of the CSR BlueCore chip off of which it was based. Due to this, it was necessary to have a host processor. It was decided to use embedded with this processor for three reasons: 1) Linux is free; 2) Linux is well known and documentation help exists; 3) Linux has several Bluetooth software stacks available. The stack chosen was BlueZ upon the suggestion of Bryan Hall from A7 Engineering.

When searching for a Bluetooth chip, it was discovered that they are normally only available from the manufacturer in BGA packages, usually measuring less than a centimeter square. The capacity for soldering such packages does not currently exist at Calvin College, so it became necessary to find a breakout board. After searching for sockets and breakout boards, the EB-100 from A7 Engineering was found. This board provided the functionality of the Bluetooth chip in a more easily solderable package. However, as discussed previously, some issues did exist regarding communication and documentation with A7 Engineering as to what functionality was accessible. Despite this lack of documentation, it was decided to proceed with the use of this board due to time constraints and the apparent lack of other options.

The new requirement of a host processor for the base station due to the use of the A7 Engineering board was not received until later in the second semester. At this time, Professor Steven VanderLeest suggested the use of the Motorola ColdFire processor, as a development kit was readily available for use. Further research showed that this processor would indeed fulfill the functionality of a host processor, and the design proceeded with this processor.

3.4.4Implementation

3.4.4.1 Detailed Engineering Drawings

3.4.4.2 Assembly Drawings

3.4.4.3 Circuit Diagrams

Circuit diagrams were made to help design and make the circuits for the project. Eagle Layout Editor was used to make the schematics for the base station prototype. This tool allows for custom parts to be easily made and used in the schematic. The schematic for the production design was made using Microsoft Visio. The final prototype schematic is shown in Appendix C.

3.4.4.3.1 Prototype

The Prototype circuit consisted of three major parts. The power supply, Bluetooth interface, and RS232 transceiver were all designed separately then connected together in a single schematic. Some of the prototype PCB production was outsourced to PCB Express, and some of the PCB production was done with the available on campus facilities. All of the PCBs were populated by the team.

3.4.4.3.1.1 Power Supply

A simple power supply was used for the prototype. See Figure 4. The regulator is a simple three pin plastic can IC. This device is made for battery applications so it has a low input current requirement. It can take an input up to 30V. The two pin connector allowed for an easy connection from a power supply to the circuit board. A filter capacitor was placed on the input and output of the regulator. The regulator's datasheet specified at least a 2.2uF capacitor on the output. No 2.2uF capacitor could be found on campus, but the 4.7uF capacitors were available for free on campus, and the 4.7uF still meets the requirements given in the datasheet.

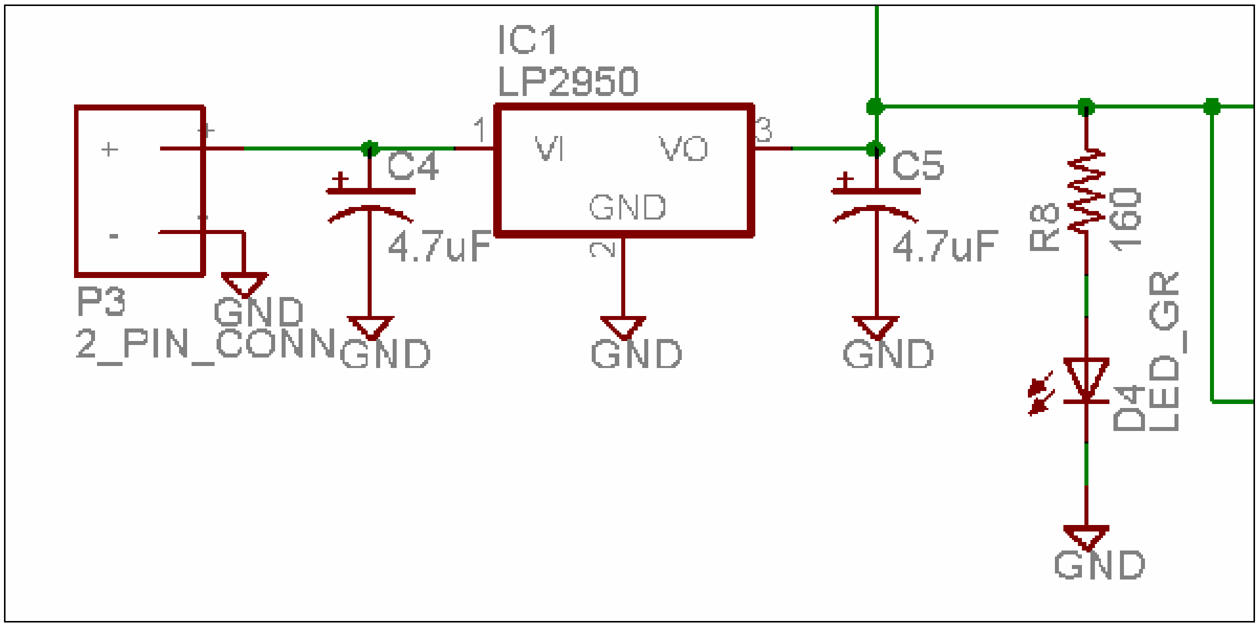


Figure 4 Prototype Power Supply

The capacitor on the input of the regulator was added after initial testing because there was some oscillation from the power supply that caused the output of the regulator to be above 3.3V. The green LED lights when the regulator is receiving an input. Resistor R8 limits the current through the LED to be 10mA so that the LED shines brightly.

3.4.4.3.1.2 Bluetooth Module

The Bluetooth module did not require a lot of surrounding circuitry. The module contained most of the required components onboard for the Bluetooth IC to function. This included memory, oscillator, regulators, and some passive components. Figure 5 shows the Bluetooth module with surrounding circuitry.

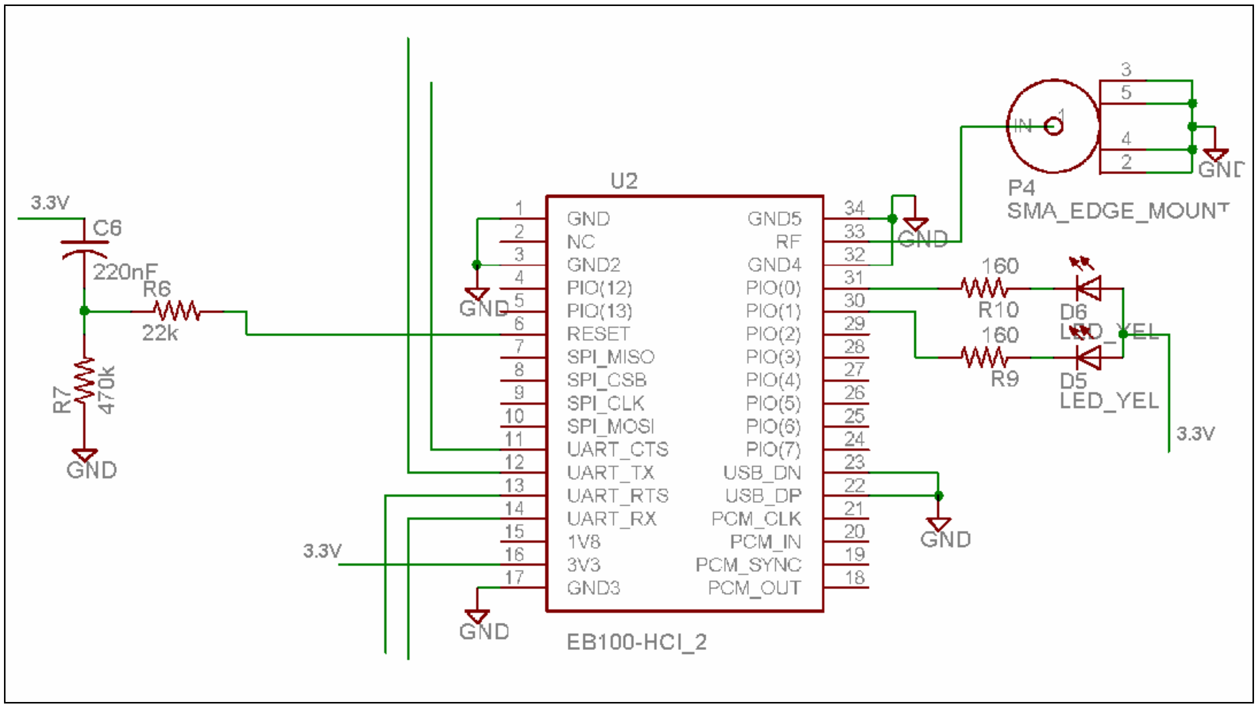


Figure 5 Bluetooth module and surrounding circuitry

All of the ground pins (1, 3, 17, 32, and 34) have been grounded. Pin 2 is not supposed to be connected to anything. Pins 4 and 5 are used for analog input/output, but these were not needed for this project. The reset, pin 6, is connected to an RC circuit. This circuit holds the Bluetooth module in reset for 5ms while the power supply is reaching its target output voltage. This is needed because the memory on the module needs 3.3V and the microprocessor needs 1.8V. If the microprocessor receives 1.8V before the memory has 3.3V, then the microprocessor will try to fetch data from the memory before the memory is ready. Pins 7 through 10 are used to program the onboard memory during the manufacturing of the Bluetooth module. The team did not use these pins for the project. The UART interface, pins 11 through 14, were connected to the RS232 transceiver, which will be explained more in sections 3.4.4.3.1.2 and 3.4.4.3.1.3. The Bluetooth module CTS pin was connected to the RTS pin on the base station host processor, and the Bluetooth module RTS pin was connected to CTS on the host processor. The Bluetooth module TX and RX pins were connected to the RX and TX pins, respectively, on the host processor. This connection configuration allows for hardware flow control between the module and the host processor. Pin 15 is a 1.8V output from the regulator that supplies the 1.8V to the microprocessor. The output of the 3.3V regulator, described in section 3.4.4.3.1.1, was connected to pin 16. This supplies power to the Bluetooth module. Pins 18 through 21 are used for audio applications and were not needed for this project. Both of the USB interface pins, 22 and 23, were grounded since the UART was used. This was done on the suggestion of Bryan Hall from A7 Engineering. Programmable IO pins 24 through 29 were not used in this project. Pins 30 and 31 are programmable IO pins that were used to indicate when the module is transmitting and receiving data on the antenna. The LEDs turn on when data is being sent and received. The diode on pin 30 will blink when data is being transmitted, and the diode on pin 31 will blink when data is being received. Resistors R9 and R10 are used to limit the current through the LEDs so that they will function properly. Pin 33 is connected to a 50 ohm antenna. This is an edge mounted sma antenna.

3.4.4.3.1.3 RS232 Transceiver

The RS232 Transceiver was needed because the UART outputs from the Bluetooth module are 3.3V TTL signals. This means that the signals are asserted either 3.3V or 0V to transmit data. The RS232 standard uses 5V to -5V signals. The -5V corresponds to the 3.3V TTL from the module and 5V corresponds to 0V TTL from the module. The RS232 transceiver converts the voltages so that these two devices can communicate with each other. The prototype connects to the host computer with a DB9 serial port. This is a common 9 pin connector available on most computers. The RS232 Transceiver is shown in Figure 6

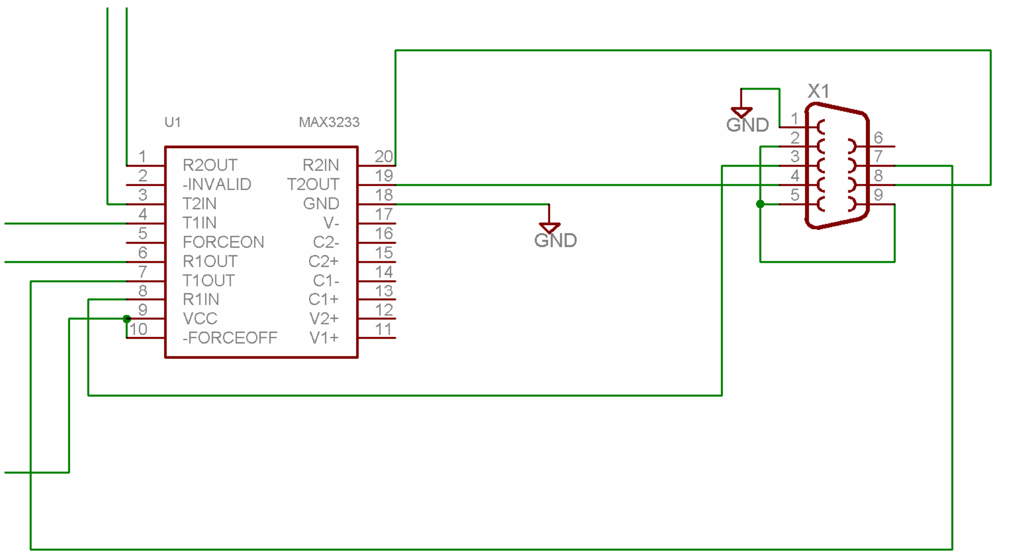


Figure 6 RS232 Transceiver

The pins of the transceiver were connected as described in section 3.4.4.3.1.2. Pin 1, R2OUT, was connected to the CTS pin of the Bluetooth module. Pin 2, INVALID, is an active low output, and will output high if a valid RS232 signal is present on a receiver. Pin 2 was not used in this project. Pins 3 and 4 were connected to the TX and RTS pins, respectively, on the Bluetooth module. Pin 5 can be driven high to force the receivers and transmitters to be active, but it was not used for this project. Pin 6 was connected to the RX pin on the Bluetooth module. Pins 7 and 8 were connected to the CTS and TX, respectively, on the host processor. Pin 9 is VCC for the transceiver and was connected to the 3.3V output of the regulator for the prototype. Pin 10 is used to shut down the receivers and transmitters and is active low; driving this pin high prevents the transmitters and receivers from being shut down. Pins 11 through 17 are outputs from internal capacitors and charge pumps, and these were not connected. Pin 18 is the ground pin for the IC. Pin 19 was connected to RX on the host processor. Pin 20 was connected to the RTS pin on the host processor.

3.4.4.3.1.4 PCB Layout

The PCB layout of the prototype was done using the Eagle Layout Editor. This program allows a PCB to be generated from a schematic. This is a much easier method than placing the parts and drawing traces manually. A schematic allows for easier debugging and changing of connections. After the traces were generated by the software, some modification was required to make the PCB easier to solder and debug. Some of the traces were moved for easier access so that they could be probed with a DMM. The PCB layout in Figure 7 is not the PCB that was constructed for the project night.

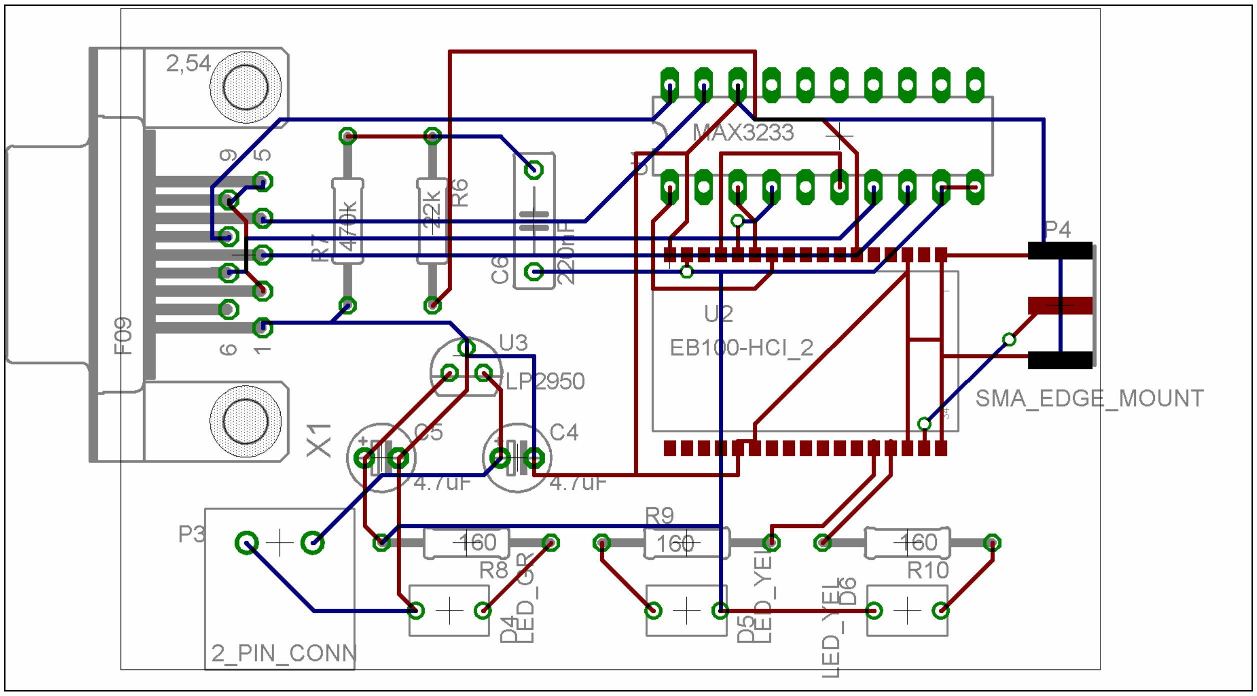


Figure 7 Prototype PCB Layout.

This latest PCB layout includes the RS232 transceiver on the same board as the rest of the circuitry. For project night the RS232 transceiver arrived after the PCB from PCB Express was ordered, so a board was made on campus to connect to the board from PCB Express. The PCB layout in Figure 7, above, is what the next PCB order might look like for future work on the project.

3.4.4.3.2 Production Model

The Production model was designed so that the team would have an initial design if mass production of the base station were to be considered. In Figure 8, the major components of the production design can be seen.

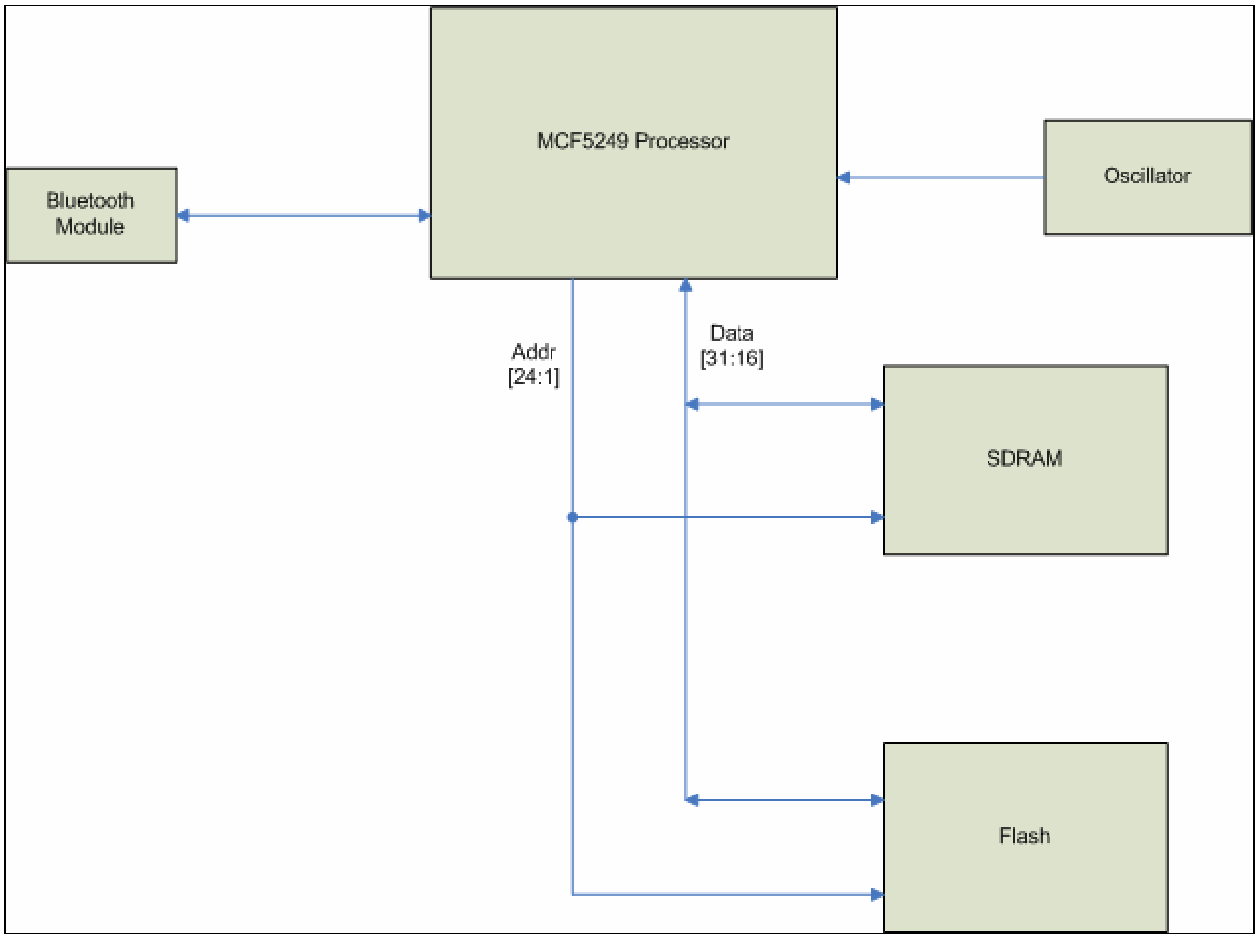


Figure 8 Production Block Diagram

The base station is similar to a small computer. There is a CPU, Flash memory, which is like a hard drive, RAM, a power supply and an oscillator. The base station also has a Bluetooth module. A more detailed system diagram can be found in appendix D. This diagram has all the signals labeled and denotes which subsystem they belong to. The directions of these signals are also shown in the diagram.

3.4.4.3.2.1 CPU

The MCF5249 Coldfire V2 was the selected processor for the production base station. The pins on the processor that were used can be seen in Figure 9.

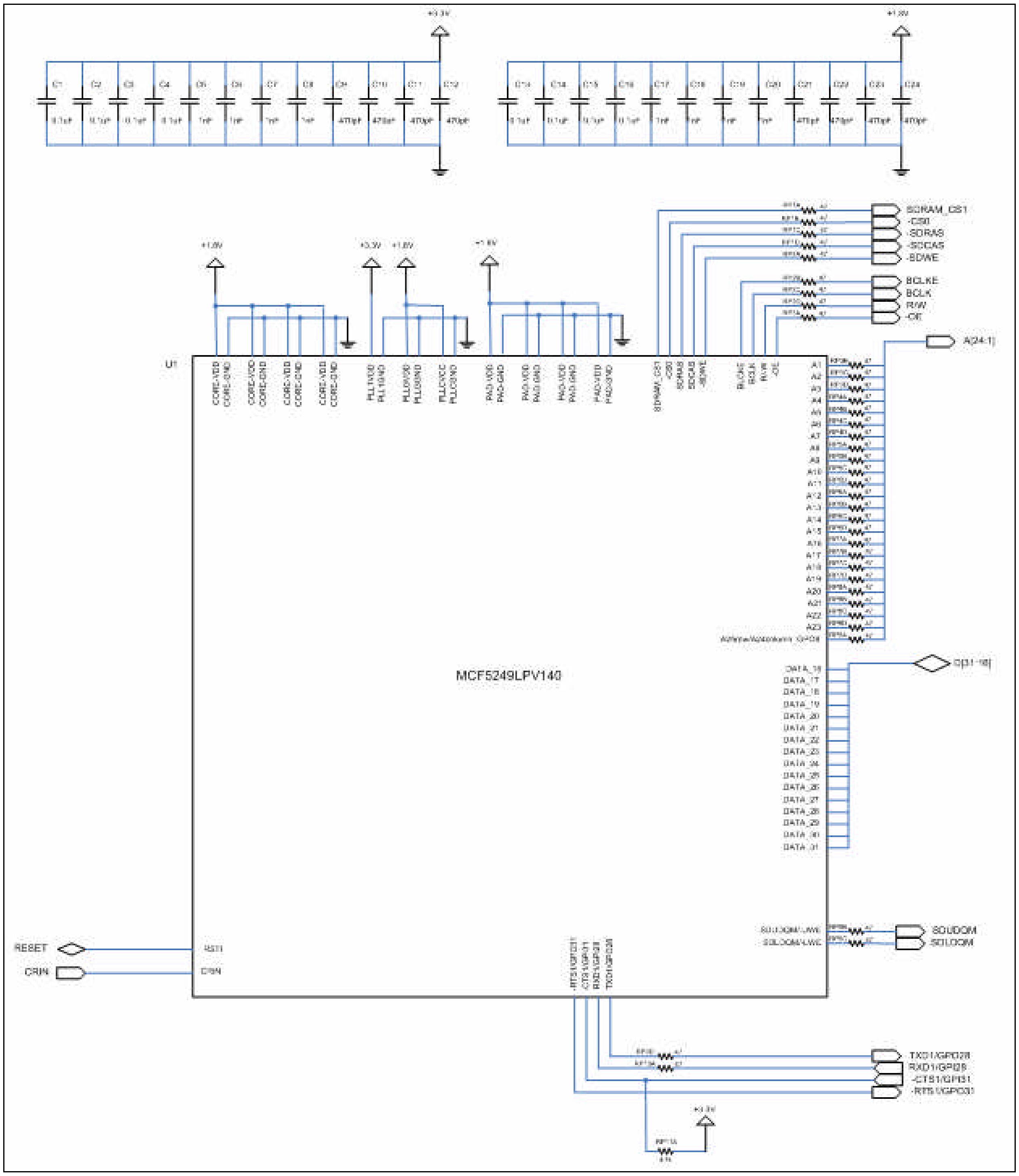


Figure 9 Base Station Processor

The array of capacitors, at the top of Figure 9, were used to decouple the voltages to the processor. These capacitors must be placed as close to the processor as possible. The CRIN pin on the processor was connected to the oscillator to give the processor the required clock speed to run. The processor is power with 1.8V and 3.3V. The other signals are discussed in the following sections.

3.4.4.3.2.2 Memory

There are two memory devices in the base station. A flash memory module which is has a capacity of 1M x 16 or 16 megabits. The SDRAM has a capacity of 1M x 16 x 4 banks which is equal to 64 megabits. The memory interface to the processor is shown in Figure 10.

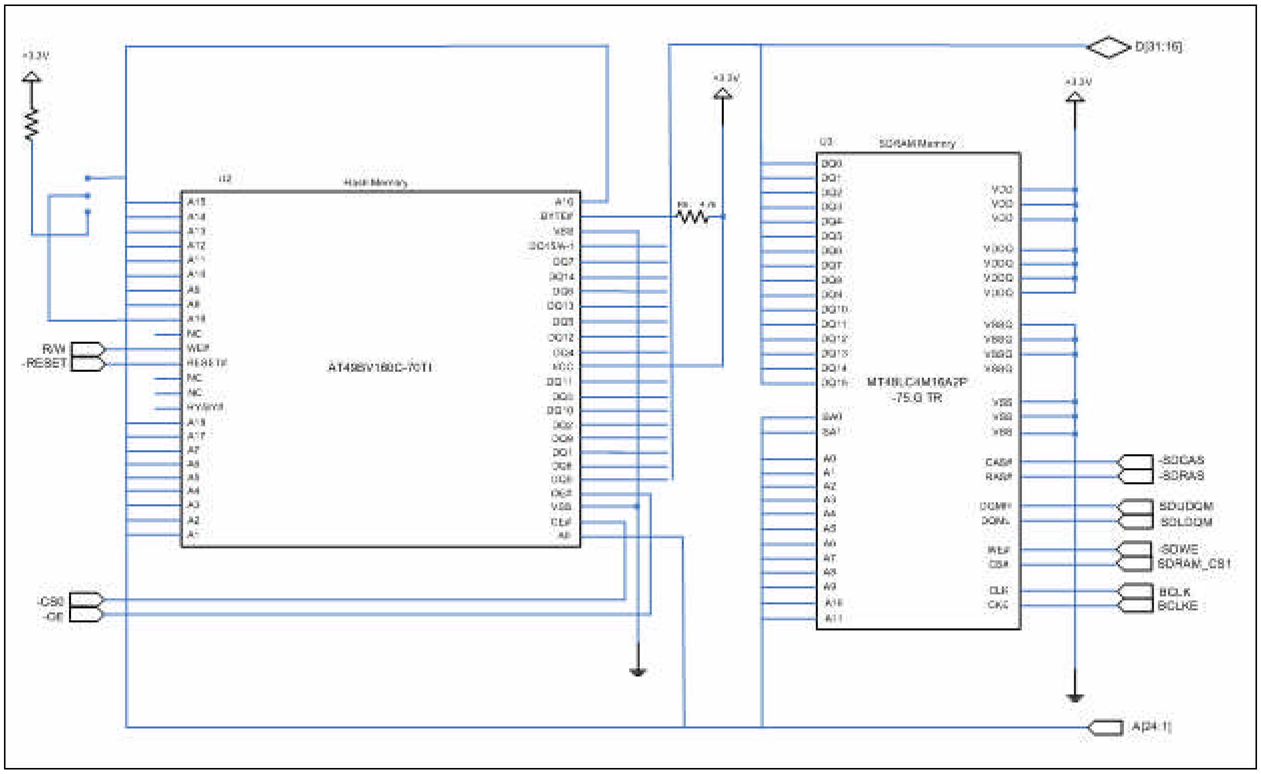


Figure 10 Production Model Memory Interface

The Flash memory is nonvolatile storage for the base station software. The address bus from the processor is a single direction signal that the memory takes as an input. The data bus is a bidirectional multiplexed bus. The R/W signal on the flash module is read/write control for the data pins of the module. The –CSO signal is the chip select signal. This allows the processor to specify which chip it is talking to. The data output is enabled by the –OE pin. The flash memory is power with 3.3V

The SDRAM holds the operating system while the base station is running so that it can be executed quickly. The same address and data busses were used for the SDRAM as were used for the flash memory. This is why they are called multiplexed busses. The –SDCAS and –SDRAS signals indicate when a valid column and row address is present. The SDUDQM signal indicates when the high bit has been written during a write cycle, and SDLDQM indicates when the low bit has been written during a write cycle. The –SDWE signal is low for write enable and high for read enable. The SDRAM_CS1 is the chip select for the SDRAM. BCLK is the clock input for the SDRAM so that it can be synchronized with the processor. The clock enable is BCLKE. When this signal is low the memory will enter a self refresh mode. The SDRAM is power with 3.3V.

3.4.4.3.2.3 Power Supply and Oscillator

The power supply circuits provide 3.3V and 1.8V for the components in the base station. This was done with simple three pin regulators and filter capacitors. The components in this subsystem can be seen in Figure 11.

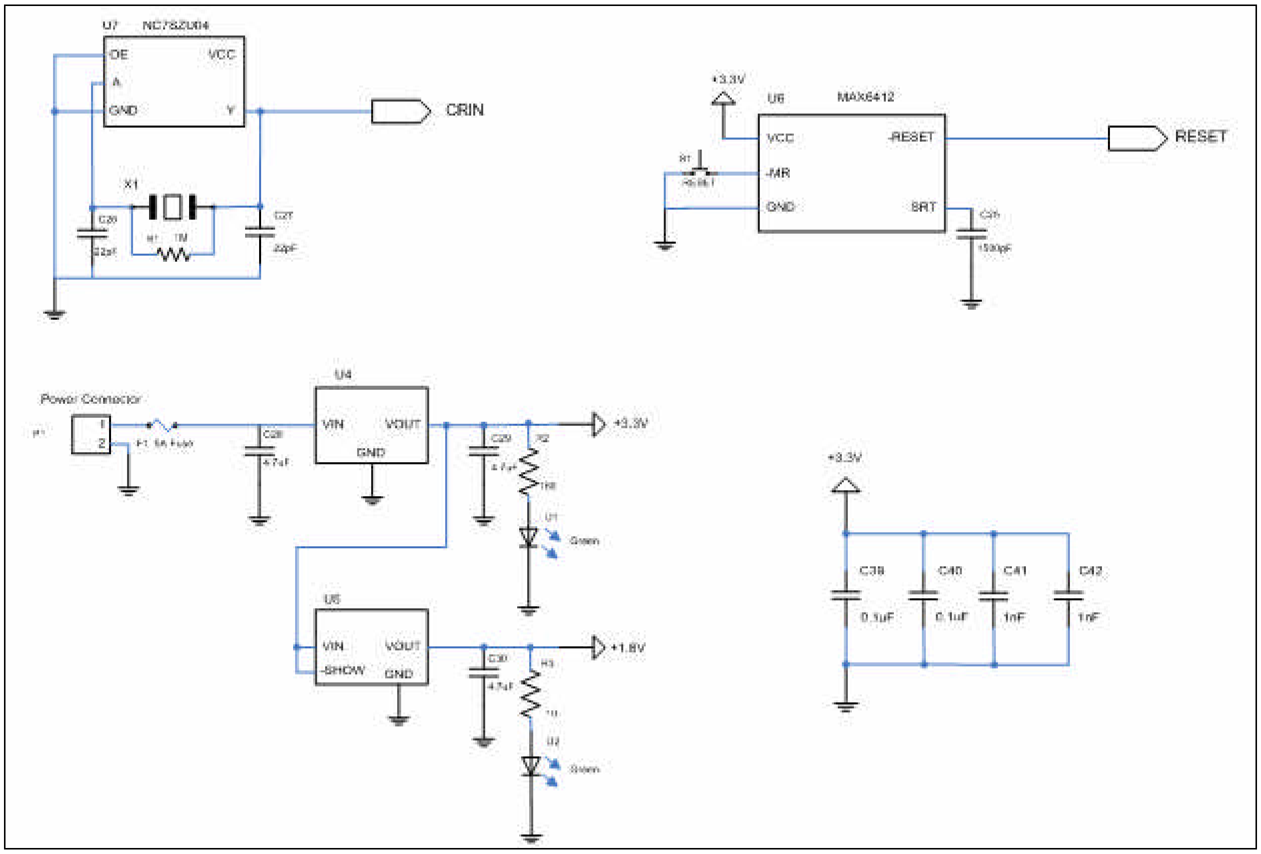


Figure 11 Power Supply and Oscillator Components

There is a power LED for each regulator to indicate that each of the regulators is functioning. There are a few decoupling capacitors for noise reduction. The reset circuit IC is U6 in Figure 11. This reset circuit monitors the 3.3V output and will hold the entire system in reset until the supply voltage is stable. This circuit also allows for a manual reset to be connected to a button as shown in the figure. The crystal oscillator, denoted as X1, is an 11.2896MHz quartz crystal. It was connected to an inverter, U7, to drive a more stable signal.

3.4.4.3.2.4 Bluetooth Module

The Bluetooth module interface is less complex than the interface on the prototype. The UART pins on the Bluetooth module could be connected directly to the UART pins on the processor. This would bypass the need for an RS232 transceiver. The Bluetooth interface circuit is shown in Figure 12

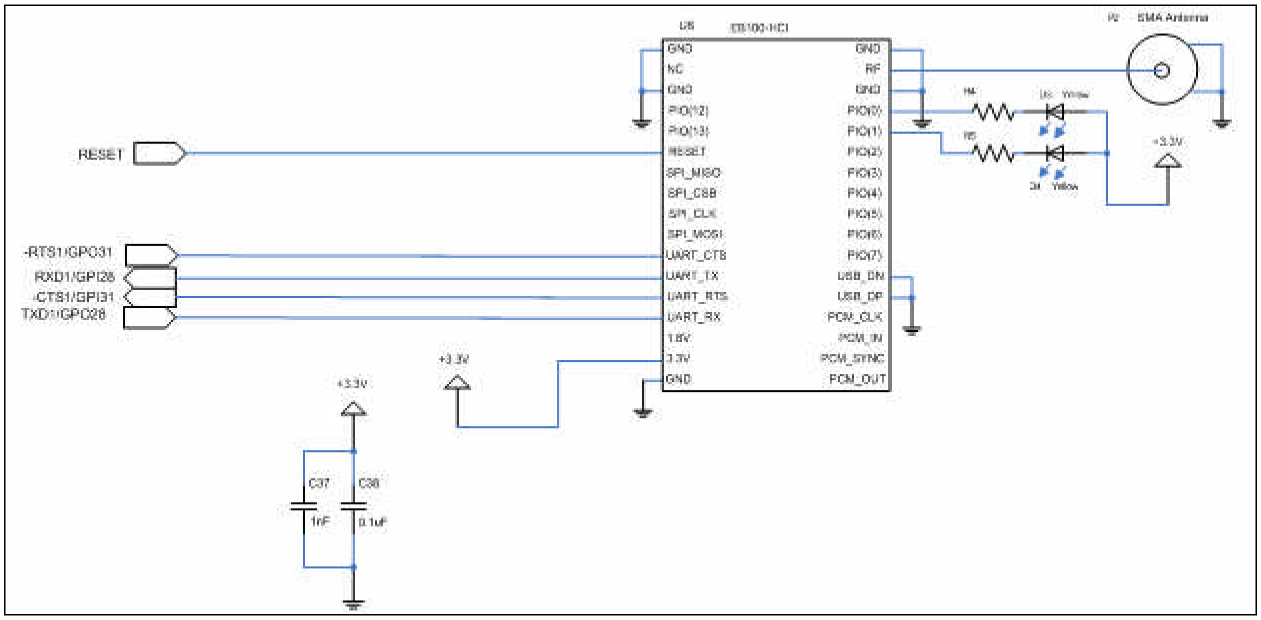


Figure 12 Production Bluetooth Interface

The pins were connected in the same way as the prototype. This means that TX and RX on the Bluetooth module were connected to RX and TX, respectively, on the processor. Also, CTS and RTS on the Bluetooth module were connected to RTS and CTS, respectively, on the processor. There are also some decoupling capacitors that need to be placed near the module. This design has the same indicator diodes to show transmitting and receiving data wirelessly. The antenna is also the same as in the prototype design.

3.4.4.4 Bill of Materials

Table 12 shows the bill of materials for the prototype. This however doesn't include prices. Since all of the parts listed were from the ColdFire development board, many of the parts could not be found to get a price.

Table 12: BOM for Prototype

Prototype

Item	Qty	Reference	Part	Description
1	2	C1, C2	4.7uF 35V	Through hole Electrolytic capacitor
2	1	C3	220nF	Through hole capacitor
3	1	D1	Kingbright W17A2GT	Green through hole LED
4	2	D2, D3	Kingbright W17A2YT	Yellow through hole LED
5	2	IC1, IC2	SN7408	Through hole 4x AND gate
6	1	IC3	LM324	General purpose opamp
7	1	IC4	National LP2950	3.3V voltage regulator TO-92
8	1	P1	Newark 39870-0702	2 pin molex connector
9	1	P2	Mouser 142-0701-801	Edge mount sma connector
10	3	R1, R2, R5		Through hole 160 ohm resistor

11	1	R3		Through hole 22k ohm resistor
12	1	R4		Through hole 470k ohm resistor
13	1	U1	A7 ENG EB100-HCI	SMT Bluetooth module
14	1	U2	MAX3233	RS232 Transceiver
15	1	X1	Newark DE9P-FRS	Male right angle DB9 through hole connector

Table 13 shows the bill of materials for the production model. This includes all of the parts and prices that would be needed to construct a base station.

Table 13: BOM for Production Model

Production Model					
Item	**Qty**	**Reference**	**Part**	**Description**	**Price**
1	11	C1, C2, C3, C4, C13, C14, C15, C16, C38, C39, C40	0.1uF 25V	SMT decoupling Capacitor	$0.0360
2	9	C5, C6, C7, C8, C17, C18, C19, C20, C37, C41, C42	1nF 50V	SMT decoupling Capacitor	$0.0090
3	8	C9, C10, C11, C12, C21, C22, C23, C24	470pF 50V	SMT decoupling Capacitor	$0.0190
4	1	C25	1500pF	SMT capacitor	$0.0101
5	2	C26, C27	22pF 50V	SMT capacitor	$0.0070
6	3	C28, C29, C30	4.7uF	SMT capacitor	$0.0400
7	2	D1, D2	Kingbright W17A2GT	Green through hole LED	$0.1000
8	2	D3, D4	Kingbright W17A2YT	Yellow through hole LED	$0.1000
9	1	F1	Multicomp MCHTE-15M	Fuse	$1.5892
10	1	P1	Newark 39870-0702	2 pin molex connector	$0.2120
11	1	P2	Mouser 142-0701-801	Edge mount SMA connector	$2.3400
12	10	RP1, RP2, RP3, RP4, RP5, RP6, RP7, RP8, RP9, RP10	Philips ARC241-47R	4x 47 ohm SMT resistor pack	$0.0190
13	1	RP11	Philips ARC241-4K7	4x 4.7k ohm SMT resistor pack	$0.0190
14	1	R1		SMT 1M ohm resistor	$0.0024
15	3	R2, R4, R5		SMT 160 ohm resistor	$0.0024
16	1	R3		SMT 10 ohm resistor	$0.0024
17	1	R6		SMT 4.7k ohm resistor	$0.0024
18	1	S1	C&K KS11R23CQD	Red Switch	$0.5733
19	1	U1	Motorola MCF249LPV140	ColdFire V2 CPU - 140 MHz	$10.3000
20	1	U2	AT49BV160C-70TI	Flash memory	$1.7100
21	1	U3	Micron MT48LC4M16A2P-75:G TR	DRAM	$3.0320
22	1	U4	National LP2950CDT-3.3	SMT 3.3V regulator	$0.4250
23	1	U5	Linear LT3020-1.8	SMT 1.8V regulator	$1.7900
24	1	U6	Maxim MAX6412	Reset controller	$0.9100
25	1	U7	National NC7SZU04	Inverter driving oscillator circuit	$0.0750
26	1	U8	EB100-HCI	SMT Bluetooth Module	$20.0000
27	1	X1	HC49US case	11.2896MHz quartz crystal	$0.6640
				Part TOTAL	$45.0172
				20 sq. in. Printed Circuit Board (P	$9.1000
				Base Station Total	**$54.12**

3.4.4.5 Code

During the first semester, an initial program flow for the software was developed (see Appendix E). However, this was designed without much knowledge of how the cellular phone ran programs as well as how Bluetooth devices connected to each other. However, the final coding was similar to these initial program flows.

The final code that was written for this project can be broken down into three parts: BluStation, BluGateServer, and BluGate. The software that was written to run on the base station is referred to as the BluStation. BluGateServer and BluGate were both written for the cell phone. BluGate was an application for the phone. This was primarily the GUI, but also contained some functionality. This is what can be opened directly from the Applications window. BluGateServer was an executable for the phone. This could not be opened directly, but rather is started by the application or started on boot. This program ran as a background process and took care of the Bluetooth connection for the phone. The following sections will provide a brief description of the code, but for more detailed documentation see Appendix F and the full code can be found in Appendix G.

3.4.4.5.1 BlúGateServer

As mentioned above, this was a program designed to run in the background of the phone. This means that it would do all its work without outputting anything to the screen unless certain conditions were met. A basic program flow can be seen in Figure 13.

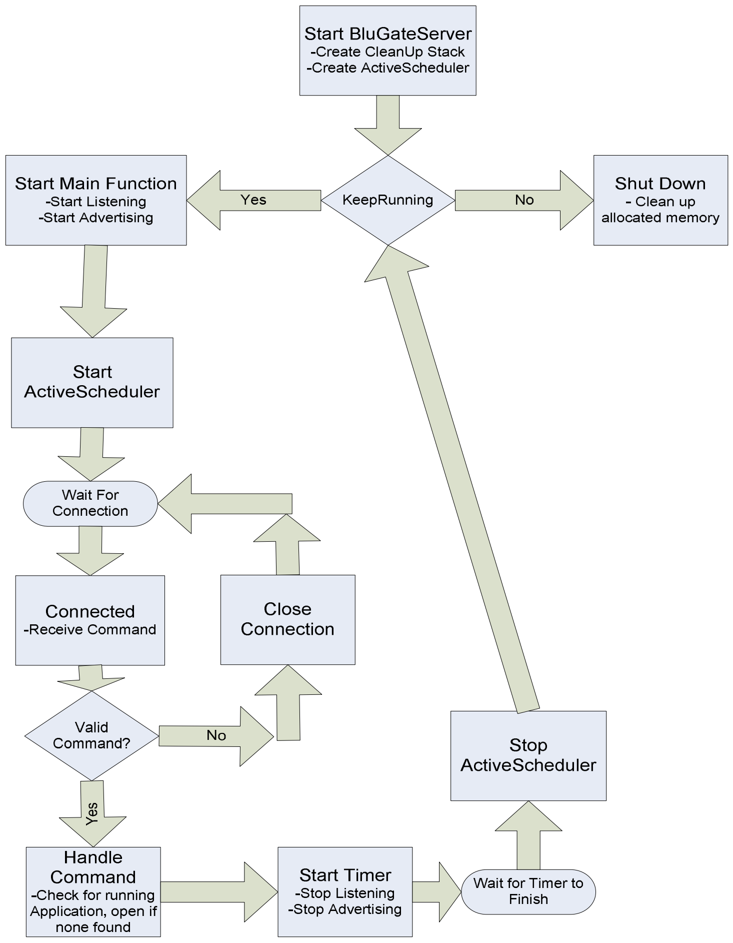


Figure 13 – BluGateServer basic program flow

When the server is started, the first thing that it does is create an ActiveScheduler. This is a Symbian specific way of handling asynchronous commands. After that is created, it decides whether the server should keep running or not. If it detects another server process running or detects that the user has turned the status to be off, then the program will exit the main function and cleanup any memory it has allocated. If it detects that it should keep running, then it will open a Bluetooth channel (port), set security settings so that no authorization or authentication is required for that channel, and listen for an RFCOMM connection on that channel. It will also edit the phones SDP database to show that it is offering the BluGate service on the channel it is listening on. After these items are taken care of, the program starts the ActiveScheduler. This basically puts the program to sleep until the conditions of the active object (in this case the active object is of type BluGate_Listener, which is waiting for a Bluetooth connection on the channel it opened) have been met.

When a Bluetooth connection has been made, the server waits for a message. It will then take that message and determine if it is a valid BluGate command. If it is not, it will close the connection, stop advertising, stop listening, and stop the ActiveScheduler. If it is a valid command, it will check for an open BluGate application. If one isn't found it will open the application (the application will take care of toggling profiles). Then it will stop listening and advertising and set and outstanding request to the ActiveScheduler to wake up the server in two minutes. This effectively stops the cell phone from being connected to more than once (if it received a valid command) every two minutes. When the two minutes is up, the ActiveScheduler is stopped. Whenever the ActiveScheduler is stopped, the server will determine again if it should keep running.

3.4.4.5.2 BlúGate

BluGate is the program that is seen by the user. It can be opened directly from the application menu of the phone. It contains the GUI where users can change the settings of the phone. It also is the program that is in charge of changing the profile of the phone. A basic program flow can be seen in Figure 14.

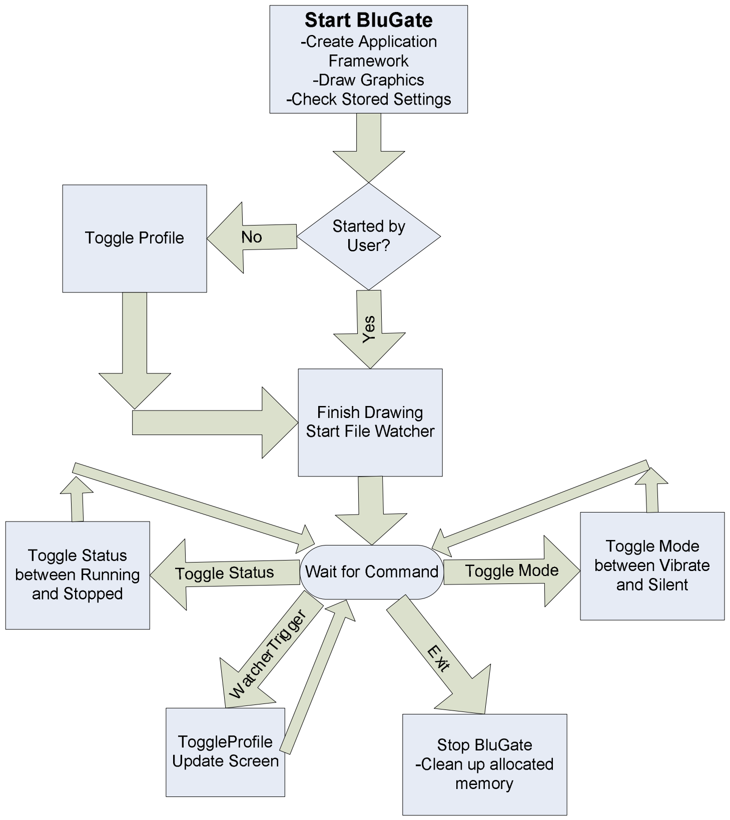


Figure 14 – BluGate basic program flow

When the BluGate application is started, the first action is to create the application framework. This is Symbian's way of making it a little bit easier for developers of Symbian applications. The main thing to note about this framework is that it has a built in ActiveScheduler to handle asynchronous actions.

After the framework is created, the program draws the graphics of the application. It checks to see if it was opened by the server, and if it was, then the profile is toggled. If not it will continue to draw the rest of the graphics. The application stores the settings (Status, Mode, and Active settings) in its own private file, and references them when the application is started. This allows the application to know the current status of the overall BluGate program, as well as save settings between phone power cycles. During the start of the program, the watcher is also started. This is a custom class object that is used to detect if BluGateServer receives a valid connection while the application is running.

After the GUI is drawn, and settings are shown, the application waits for a command. From the menu, the user currently has 3 options: Toggle Status, Toggle Mode, and Exit. The Toggle Status command will change the status from between Running and Stopped. If the status is Stopped, the phone will not change profiles when passing a device running BluStation. The Toggle Mode command allows the user to switch between Silent or Vibrate. This is the profile that the phone will be changed to when a device running BluStation is detected. The last user option is Exit, which exits the application. For a more detailed description of the GUI, see the User's Manual in Appendix H.

Another command that the application is waiting for is the WatcherTrigger. This is a process that is activated when BluGateServer has accepted a valid command from a base station. This will cause the program to toggle profiles and update the screen to show that it received a BluGate connection.

3.4.4.5.3 BlúStation

The BlúStation is a program responsible for finding mobile devices, determining if they have the BlúGate program, and toggling them if they do, see Figure 15. This was accomplished through the use of three functions: search(), sdpbrowse(), and rfconnect(). Search() is responsible for searching out all Bluetooth devices in the area. Sdpbrowse() is responsible for browsing the SDP databases of all devices found by search(). Rfconnect() is responsible for connecting to the devices shown by sdpbrowse() to be running the BlúGateServer program. These three functions are called within a forever loop in the main() function of the BlúStation program. This allows the program to run indefinitely. Each function is passed a pointer array deviceList of type deviceInfo. Device info is a struct containing address and service channel information.

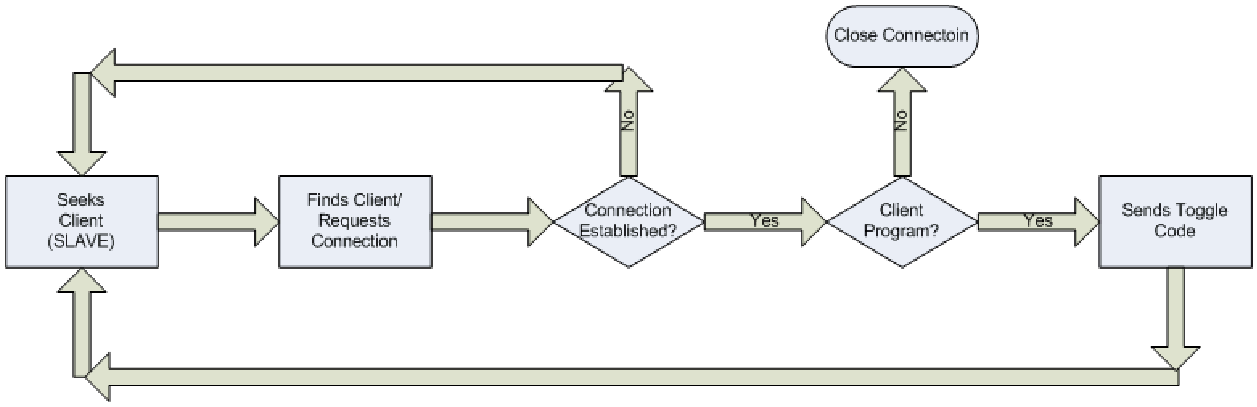


Figure 15: Base Station Program Flow

At the beginning of the forever loop, the array device array is initialized to zero for the channel and cleared memory for the address character array. Sdpbrowse() updates the channel value with the channel obtained during the SDP service search. If the BlúGate service is not found, the channel remains zero. Rfconnect() checks the channel value prior to attempting to connect to a device. If the device channel is non-zero, it will connect to it, otherwise that address will be ignored. Both sdpbrowse() and rfconnect() utilize validAddress() to check if the data in btaddr[] of a device is formatted correctly to be a Bluetooth IP address. If it is not, the array entry will be ignored. Complete BluStation Code can be found in Appendix G.

3.4.5 Functionality

The Hardware functionality is evident in the Bluetooth RS232 interface. The device can be plugged into a serial port and used to make wireless connections to Bluetooth devices. This is important, because the production design depends upon the BlueCore IC being used serially. The Bluetooth RS232 interface proves that using a serial connection is feasible.

The interface is equipped with three LED's. One of the LED's is used to show power; the other two LED's are used to show data transfers. The interface however, must be powered from an outside source. A 2-pin molex (or 12 volt power connector) is provided on the board to supply the voltage needed.

The last major piece of the Bluetooth RS232 interface is the voltage transceiver. The voltage transceiver changes the voltage between the RS232 plug and the Bluetooth module. These to devices operate at different voltage levels, so communication would be impossible without the voltage transceiver.

The other main piece of hardware was the base station production model. It, however, exists in theory, and was constructed based on research and team knowledge. Therefore, there is no real functionality to this device, yet.

The BlúGate base station has an operational radius of approximately thirty feet in unobstructed areas. This allows enough time for the base station code to inquire for Bluetooth addresses, scan the addresses for the BlúGate client, connect to each one, and send the toggle code. The system has only been tested with a single mobile device (Nokia 3650) due to the lack of multiple available compatible mobile devices. However, given the time taken for the single phone, it is reasonable to assume that the flux of people passing through the area in a typical movie theater – assumed here to be approximately thirty people per minute – would not overwhelm the system.

The BlúGate client program allows for automatic connection via Bluetooth to a BlúGate base station if the current status is "Running". When toggle code is received from the base station, the profile of the phone will be changed based on the mode setting. The program has the ability to save the users settings (before being toggled) and restoring them upon a second pass by a base station. It also has the ability to start itself when the phone is turned on (it will load approximately 30 seconds after the phone has booted).

The BlúGate client program also has a graphical user interface that displays the active state of the phone, the current status, and the current mode. In this GUI, the user can change the settings of the phone. The status can be switched to Running or Stopped, and the mode can be changed to Silent or Vibrate.

4. Conclusion

4.1 Lessons Learned

As in most projects it is difficult to predict many issues and problems that can arise. In Team 6’s case, a major problem was overestimations of part functionality. It was assumed that the BlueCore2 External development kit would provide all of the needed functionality to construct a base station. Unfortunately, that was not the case, and the team had to scramble to find a solution. Therefore, it is importation to always have several plans, so that it is not overly difficult to change if needed.

Also, Team 6 learned to learn. The team had no previous knowledge regarding Bluetooth, since Bluetooth is a new technology. The team also had to learn to design Printed Circuit Boards (PCB) in Eagle and make them in the lab, program in a Linux environment, and access phone functionality in the Symbian OS.

Communication was the last lesson learned. The team was broken up into two groups: two members worked on hardware and two members worked on software. Although the hardware and software groups communicated well in themselves, full team communication was lacking. Sometimes it was difficult for the groups to explain to each other what they had accomplished. Also, sometimes one group was dependent on the other team’s progress, and if the other group did not have what was needed, it caused frustration and inefficiency.

4.2 Future Work

The current performance of this design is not to the standard that would be necessary for a production version. It must be kept in mind, however, that this design is still in the prototype stages. At the time of this writing, the base station code takes approximately ten seconds from beginning of device scan to start of next device scan. The operational radius of the base station is, at present, approximately 30 feet (as specified for such a class 2 Bluetooth device [25]). These figures interact well with one another and allow for proper operation of the base station. However, in order to reduce overlap from one base station to another, the operational radius would need to be reduced for most applications; for example, movie theaters. With a reduced radius, the base station code would need to be streamlined for faster operation to allow the same flux of people through the now reduced area.

In addition to increasing the speed, the base station code may also be augmented with additional functionality in future development. The use of the Linux operating system for the base station allows for great expandability. One suggested possibility is the use of a database to track devices as they go past the gate. With the ability to recognize specific devices, the base station would be capable of more intelligent functionality. The possibility exists to dynamically set timer lengths on the device, as well as track whether the device is coming or going through the gate, so as to trigger the correct setting. This method would be preferable to the current method of forcing the device dormant by the client program for a pre-specified amount of time after receiving the base station signal as this greater flexibility would increase the robustness of the system.

The base station hardware is rather large for the prototype and could be incorporated into a single board design in the future. Such a design is included in our deliverables for this semester, but a working model could be made. However, more memory (both volatile and non-volatile) would have to be added to the included design if the base station would be responsible for databaseing. The amounts would vary based upon the increase in code requirements.

As of the time of this writing, the client program is programmed only for SymbianOS S60 1v0 cellular phones -- specifically the Nokia 3650. Future development of this system would need to port the program functionality to other operating systems. Viable systems for this to be ported to would be other versions of the SymbianOS, the PalmOS, or WindowsCE. These are the current major mobile platforms. In addition, the code could be ported to various proprietary operating systems, however this would most likely require working with the phone manufacturers.

5. Acknowledgements

The following people and organizations have helped Silence is Golden with this design project. Silence is Golden would like to thank everyone listed for helping to make this a successful project.

Chuck Holwerda for helping with PCB manufacturing, and other areas of the project.

Jeremy Andrus a Calvin alumnus who gave tips and advice for the Coldfire processor.

Professor Randall Brouwer for giving advice and answering questions relating to the project

Professor Steven VanderLeest for being the team’s advisor, and giving suggestions and hints when the team hit some road blocks.

Team 10 for giving advice and help with Linux and Bluetooth.

PCB Express for giving the team $350 worth of their services as a student sponsorship program.

6. Bibliography

1. Cell Block Technologies Inc. *Cell Phone Security Systems: Non Jamming Cell Phone Silencer To Prevent Incoming Calls*. http://www.cell-block-r.com/ (9 December 2005).

2. BlueLinx, Inc. *B L U E L I N X, Inc.* . http://www.bluelinx.com/index.html (9 December 2005).

3. Slettengren, Sven K., and Alex K. Raith. "Politeness zones for wireless communication devices." *United States Patent:6,898,445*. United States Patent and Trademark Office. http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/srchnum.htm&r=1&f=G&l=50&s1=6,898,445.WKU.&OS=PN/6,898,445&RS=PN/6,898,445 (9 December 2005).

4. BTDesigner Online Store. *BlueCore ICs*. http://www.mangocommerce.com/btdesigner/bluecore2extics.cfm (9 December 2005).

5. Cambridge Silicon Radio. *CSR Support*. http://www.csrsupport.com (9 December 2005).

6. Cambridge Silicon Radio. "BlueCore2-External Product Data Sheet." *CSR Support*. http://www.csrsupport.com/download/83/BlueCore2-External%20Data%20Sheet%20BC212015-ds-001Pi.pdf (9 December 2005).

7. National Semiconductor. *Simply Blue Product Family*. http://www.national.com/appinfo/wireless/simply_blue.html (9 December 2005).

8. Sony. *Sony Global - Radio Communication*. http://www.sony.net/Products/SC-HP/pro/radio/bluetooth.html (9 December 2005).

9. Nokia. *Nokia: Forum Nokia*. http://www.forum.nokia.com/main/0,,034-4,00.html (9 December 2005).

10. Mobiledia Corp. *Mobiledia*. http://www.mobiledia.com/shop/search/compare.php (9 December 2005).

11. Nokia. *Nokia: Web Specials*. http://nokia.letstalk.com/product/promo.htm?pgId=100&setZip=15001&model=6256i (9 December 2005).

12. Amazon.com. *Amazon.com*. http://www.amazon.com/exec/obidos/subst/home/home.html/103-4341911-1514268 (9 December 2005).

13. Nokia. *Python for Series 90*. http://www.forum.nokia.com/python (9 December 2005).

14. All About Group. *AAS Gallery :: Series 60 Phones*. http://www.allaboutsymbian.com/gallery/Series-60-Phones (9 December 2005).

15. Sun Microsystems. *J2SE 1.4.2*. http://java.sun.com/j2me/download.html (9 December 2005).

16. Sun Microsystems, Inc. *Java 2 Platform, Micro Edition (J2ME) Overview*. http://java.sun.com/j2me/overview.html (9 December 2005).

17. Woodings, Ryan, Derek Joos, Trevor Clifton, and Charles Knutson. "Rapid Heterogeneous Connection Establishment:." Brigham Young University. http://faculty.cs.byu.edu/~knutson/publications/IrDA_Assisted_BT_Discovery.pdf (9 December 2005).

18. HSW Media Network. *Howstuffworks* . http://electronics.howstuffworks.com/Bluetooth.htm (9 December 2005).

19. Karty, Steve. "Bluetooth Personal Area Network Technology." *Office of the Manager National Communications System Techinical Notes* 7, no. 3 (2000). http://www.ncs.gov/library/tech_notes/tn_vol7n3.pdf (9 December 2005).

20. Levine, Bernard, “Why Go Lead Free?”, Priority Press Edition No. 7, Environmental and Occupational Risk Management, September 2003. http://www.eorm.com/ezine/pp7/leadfree.asp.

21. Freescale Semiconductor. M5249C3 Product Summary Page. http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=M5249C3&parentCode=MCF5249&nodeId=0162468rH3YTLC00M93094 (May 11, 2006)

22. Cambridge Silicon Radio. CSR store: Development tools. www.csr.com/store/category.php?catID=28 (May 11, 2006)

23. Cambridge Silicon Radio. “BlueCore2-External Product Data Sheet.” *CSR Support*. P 32. http://www.csrsupport.com/download/83/BlueCore2-External%20Data%20Sheet%20BC212015-ds-001Pi.pdf (August 2004)

24. Energizer. “Energizer 522 Product Datasheet.” http://data.energizer.com/PDFs/522.pdf (May 11, 2006)

25. Bluetooth SIG. “Bluetooth Basics” http://www.bluetooth.com/Bluetooth/Learn/Basics/ (May 16, 2006)

7. Appendices

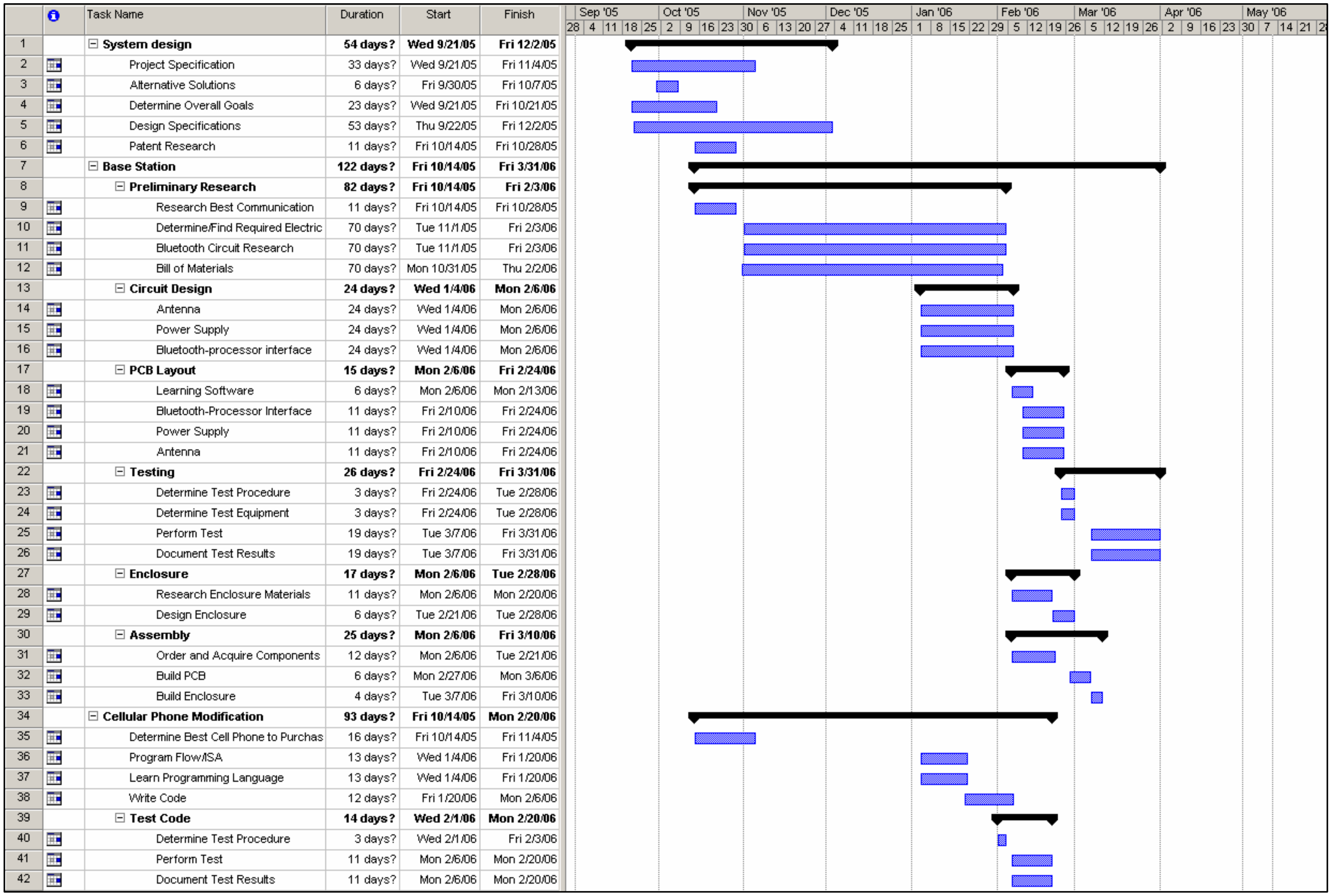
A. Task List

			Hours logged			
Task name	**Hours Budgeted**	**Hours Logged**	**Matt: Hours**	**Nick: Hours**	**Jared: Hours**	**Ryan: Hours**
System design	68.00	39.50	9.00	13.00	9.00	8.50
Project Specification	10.00	9.00	2.00	3.00	3.00	1.00
Alternative Solutions	10.00	6.00	2.00	2.00	1.00	1.00
Determine Overall Goals	8.00	6.00	1.00	2.00	1.00	2.00
Design Specifications	20.00	16.00	4.00	4.00	4.00	4.00
Patent Research	20.00	2.50		2.00		0.50
Base Station	463.00	466.50	125.00	156.50	160.00	25.00
Preliminary Research	91.00	42.00	16.00	4.50	18.50	3.00
Research Best Communication	5.00	1.00		1.00		
Determine/Find Required Electrical Components	50.00	22.00	14.00		7.00	1.00
Bluetooth Circuit Research	20.00	15.00		3.50	11.50	
Bill of Materials	16.00	4.00	2.00			2.00
Circuit Design	150.00	90.00	44.00	3.00	43.00	0.00
Antenna	15.00	2.00		2.00		
Power Supply	35.00	15.00		1.00	14.00	
Bluetooth-processor interface	100.00	73.00	44.00		29.00	
PCB Layout	100.00	33.00	5.00	4.00	24.00	0.00
Learning Software	8.00	10.00		2.00	8.00	
Bluetooth-Processor Interface	79.00	17.00	5.00	2.00	10.00	
Power Supply	5.00	3.00			3.00	
Antenna	8.00	3.00			3.00	
Testing	66.00	80.00	24.00	20.00	26.00	10.00
Determine Test Procedure	4.00	4.00	2.00		2.00	
Determine Test Equipment	2.00	0.50	0.50			
Perform Test	40.00	72.00	20.00	20.00	22.00	10.00
Document Test Results	20.00	3.50	1.50		2.00	
Enclosure	16.00	0.00	0.00	0.00	0.00	0.00
Research Enclosure Materials	4.00	0.00				
Design Enclosure	12.00	0.00				
Assembly	40.00	84.50	36.00	0.00	48.50	0.00
Order and Acquire Components	4.00	4.00	2.00		2.00	
Build PCB	26.00	80.50	34.00		46.50	
Build Enclosure	10.00	0.00				
Base Station Software (Added)	0.00	137.00	0.00	125.00	0.00	12.00
Learn OS	0.00	15.00		15.00		
Code Base Station	0.00	110.00		100.00		10.00
Test Code	0.00	12.00		10.00		2.00
Cellular Phone Modification	177.00	327.50	26.00	113.00	0.00	188.50
Determine Best Cell Phone to Purchase/Dev Kit	8.00	30.00	26.00	3.00		1.00
Program Flow/ISA	25.00	0.00				
Learn Programming Language	20.00	30.00		22.00		8.00
Write Code	100.00	221.00		75.00		146.00
Test Code	24.00	46.50	0.00	13.00	0.00	33.50
Determine Test Procedure	4.00	12.00		2.00		10
Perform Test	16.00	31.50		10.00		21.50
Document Test Results	4.00	3.00		1.00		2.00

Task List continued

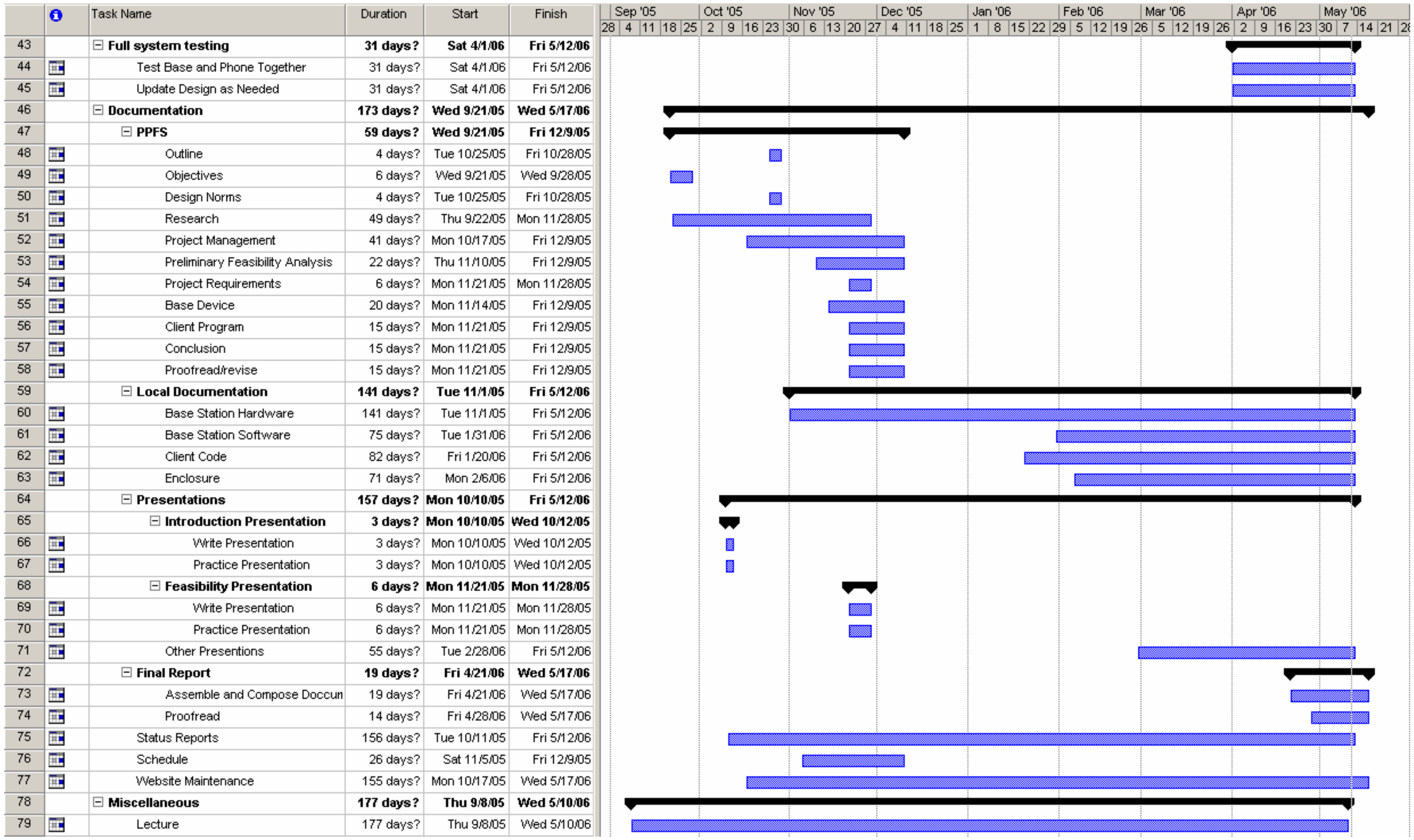
Task name	Hours Budgeted	Hours Logged	Matt: Hours	Nick: Hours	Jared: Hours	Ryan: Hours
Full system testing	144.00	40.00	0.00	20.00	0.00	20.00
Test Base and Phone Together	96.00	20.00		10.00		10.00
Update Design as Needed	48.00	20.00		10.00		10.00
Documentation	358.00	349.50	79.50	94.50	107.50	68.00
PPFS	71.00	113.00	21.00	28.50	37.50	26.00
Outline	8.00	8.00	2.00	2.00	2.00	2.00
Introduction	3.00	2.50	0.50		2.00	
Objectives	1.00	1.50			1.50	
Design Norms	1.00	3.00		3.00		
Research	7.00	17.50	6.00	1.00	6.50	4.00
Project Management	4.00	4.00				4.00
Preliminary Feasibility Analysis	8.00	9.00		6.00	3.00	
Project Requirements	2.00	3.00			3.00	
Base Device	15.00	10.50		4.00	6.50	
Client Program	15.00	3.50		3.50		
Conclusion	2.00	1.50	1.50			
Appendix	4.00	2.00				2.00
Proofread	12.00	47.00	11.00	9.00	13.00	14.00
Local Documentation	105.00	80.00	20.00	20.00	20.00	20.00
Base Station Hardware	33.00	45.00	20.00	5.00	20.00	
Base Station Software	33.00	15.00		15.00		
Client Code	33.00	20.00				20.00
Enclosure	6.00	0.00				
Presentations	90.00	75.50	21.50	26.00	23.50	4.50
Write Presentation	24.00	39.00	11.00	14.50	12.00	1.50
Practice Presentation	56.00	25.50	7.50	7.50	8.50	2.00
Exectuive Summaries	10.00	11.00	3.00	4.00	3.00	1.00
Final Report	36.00	56.00	14.00	14.00	14.00	14.00
Assemble and Compose Doccuments	24.00	40.00	10.00	10.00	10.00	10.00
Proofread	12.00	16.00	4.00	4.00	4.00	4.00
Status Reports	40.00	15.50	2.00	2.50	10.00	1.00
Schedule	6.00	7.00	1.00	1.00	2.50	2.50
Website Maintenance	10.00	2.50		2.50		
Miscellaneous	200.00	270.50	70.50	68.50	67.00	64.50
Lecture	150.00	199.00	46.00	51.00	51.00	51.00
Miscellaneous (consultant meeting)	50.00	60.50	20.50	14.50	14.00	11.50
Marketting	15.00	11.00	4.00	3.00	2.00	2.00
Total Hours	**1410.00**	**1493.50**	**310.00**	**465.50**	**343.50**	**374.50**
Total Days	**58.75**	**62.23**	**12.92**	**19.40**	**14.31**	**15.60**

B. Schedule



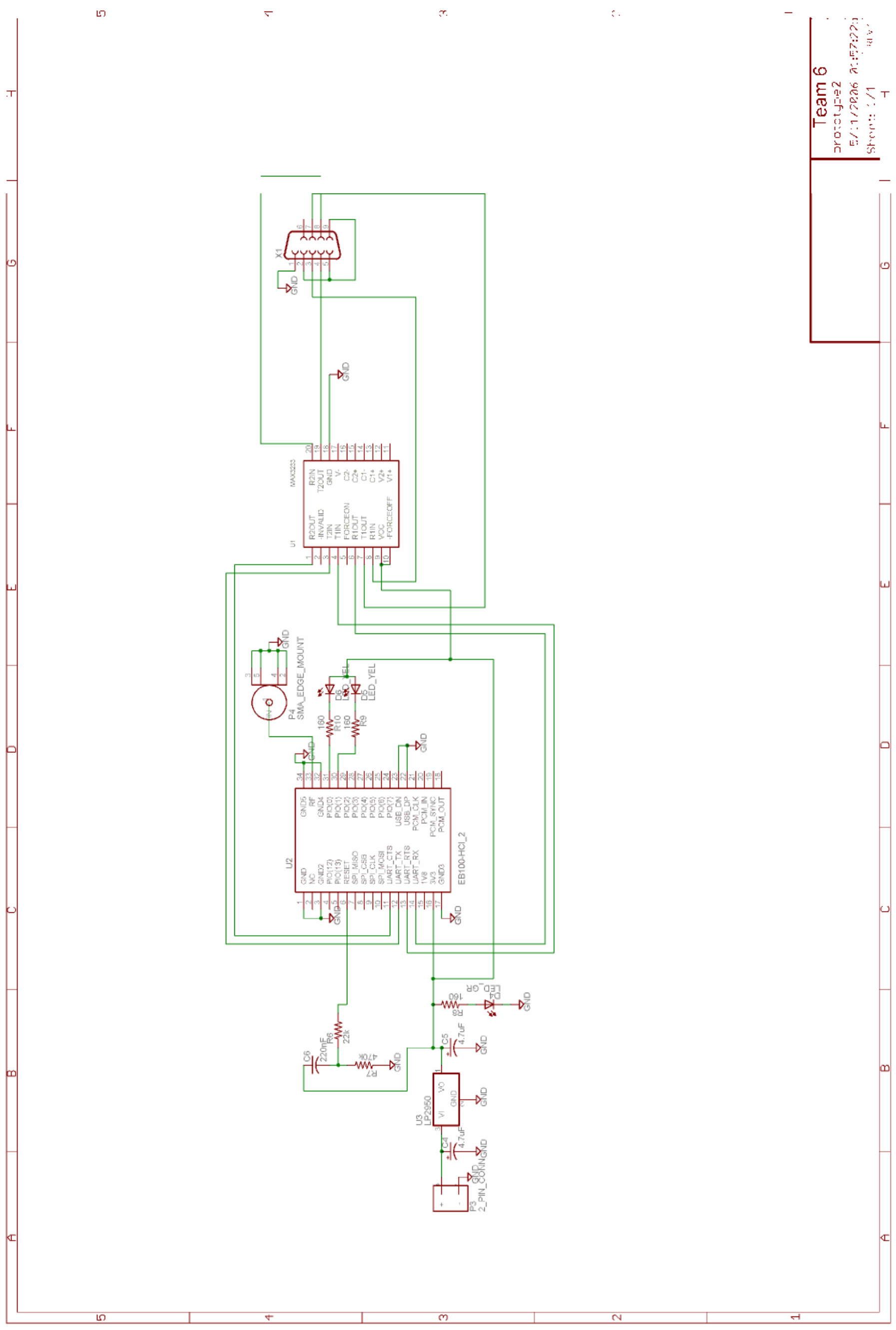
		Task Name	Duration	Start	Finish
1		⊟ **System design**	**54 days?**	**Wed 9/21/05**	**Fri 12/2/05**
2	▦	Project Specification	33 days?	Wed 9/21/05	Fri 11/4/05
3	▦	Alternative Solutions	6 days?	Fri 9/30/05	Fri 10/7/05
4	▦	Determine Overall Goals	23 days?	Wed 9/21/05	Fri 10/21/05
5	▦	Design Specifications	53 days?	Thu 9/22/05	Fri 12/2/05
6	▦	Patent Research	11 days?	Fri 10/14/05	Fri 10/28/05
7		⊟ **Base Station**	**122 days?**	**Fri 10/14/05**	**Fri 3/31/06**
8		⊟ **Preliminary Research**	**82 days?**	**Fri 10/14/05**	**Fri 2/3/06**
9	▦	Research Best Communication	11 days?	Fri 10/14/05	Fri 10/28/05
10	▦	Determine/Find Required Electric	70 days?	Tue 11/1/05	Fri 2/3/06
11	▦	Bluetooth Circuit Research	70 days?	Tue 11/1/05	Fri 2/3/06
12	▦	Bill of Materials	70 days?	Mon 10/31/05	Thu 2/2/06
13		⊟ **Circuit Design**	**24 days?**	**Wed 1/4/06**	**Mon 2/6/06**
14	▦	Antenna	24 days?	Wed 1/4/06	Mon 2/6/06
15	▦	Power Supply	24 days?	Wed 1/4/06	Mon 2/6/06
16	▦	Bluetooth-processor interface	24 days?	Wed 1/4/06	Mon 2/6/06
17		⊟ **PCB Layout**	**15 days?**	**Mon 2/6/06**	**Fri 2/24/06**
18	▦	Learning Software	6 days?	Mon 2/6/06	Mon 2/13/06
19	▦	Bluetooth-Processor Interface	11 days?	Fri 2/10/06	Fri 2/24/06
20	▦	Power Supply	11 days?	Fri 2/10/06	Fri 2/24/06
21	▦	Antenna	11 days?	Fri 2/10/06	Fri 2/24/06
22		⊟ **Testing**	**26 days?**	**Fri 2/24/06**	**Fri 3/31/06**
23	▦	Determine Test Procedure	3 days?	Fri 2/24/06	Tue 2/28/06
24	▦	Determine Test Equipment	3 days?	Fri 2/24/06	Tue 2/28/06
25	▦	Perform Test	19 days?	Tue 3/7/06	Fri 3/31/06
26	▦	Document Test Results	19 days?	Tue 3/7/06	Fri 3/31/06
27		⊟ **Enclosure**	**17 days?**	**Mon 2/6/06**	**Tue 2/28/06**
28	▦	Research Enclosure Materials	11 days?	Mon 2/6/06	Mon 2/20/06
29	▦	Design Enclosure	6 days?	Tue 2/21/06	Tue 2/28/06
30		⊟ **Assembly**	**25 days?**	**Mon 2/6/06**	**Fri 3/10/06**
31	▦	Order and Acquire Components	12 days?	Mon 2/6/06	Tue 2/21/06
32	▦	Build PCB	6 days?	Mon 2/27/06	Mon 3/6/06
33	▦	Build Enclosure	4 days?	Tue 3/7/06	Fri 3/10/06
34		⊟ **Cellular Phone Modification**	**93 days?**	**Fri 10/14/05**	**Mon 2/20/06**
35	▦	Determine Best Cell Phone to Purchas	16 days?	Fri 10/14/05	Fri 11/4/05
36	▦	Program Flow/ISA	13 days?	Wed 1/4/06	Fri 1/20/06
37	▦	Learn Programming Language	13 days?	Wed 1/4/06	Fri 1/20/06
38	▦	Write Code	12 days?	Fri 1/20/06	Mon 2/6/06
39		⊟ **Test Code**	**14 days?**	**Wed 2/1/06**	**Mon 2/20/06**
40	▦	Determine Test Procedure	3 days?	Wed 2/1/06	Fri 2/3/06
41	▦	Perform Test	11 days?	Mon 2/6/06	Mon 2/20/06
42	▦	Document Test Results	11 days?	Mon 2/6/06	Mon 2/20/06

Schedule Continued

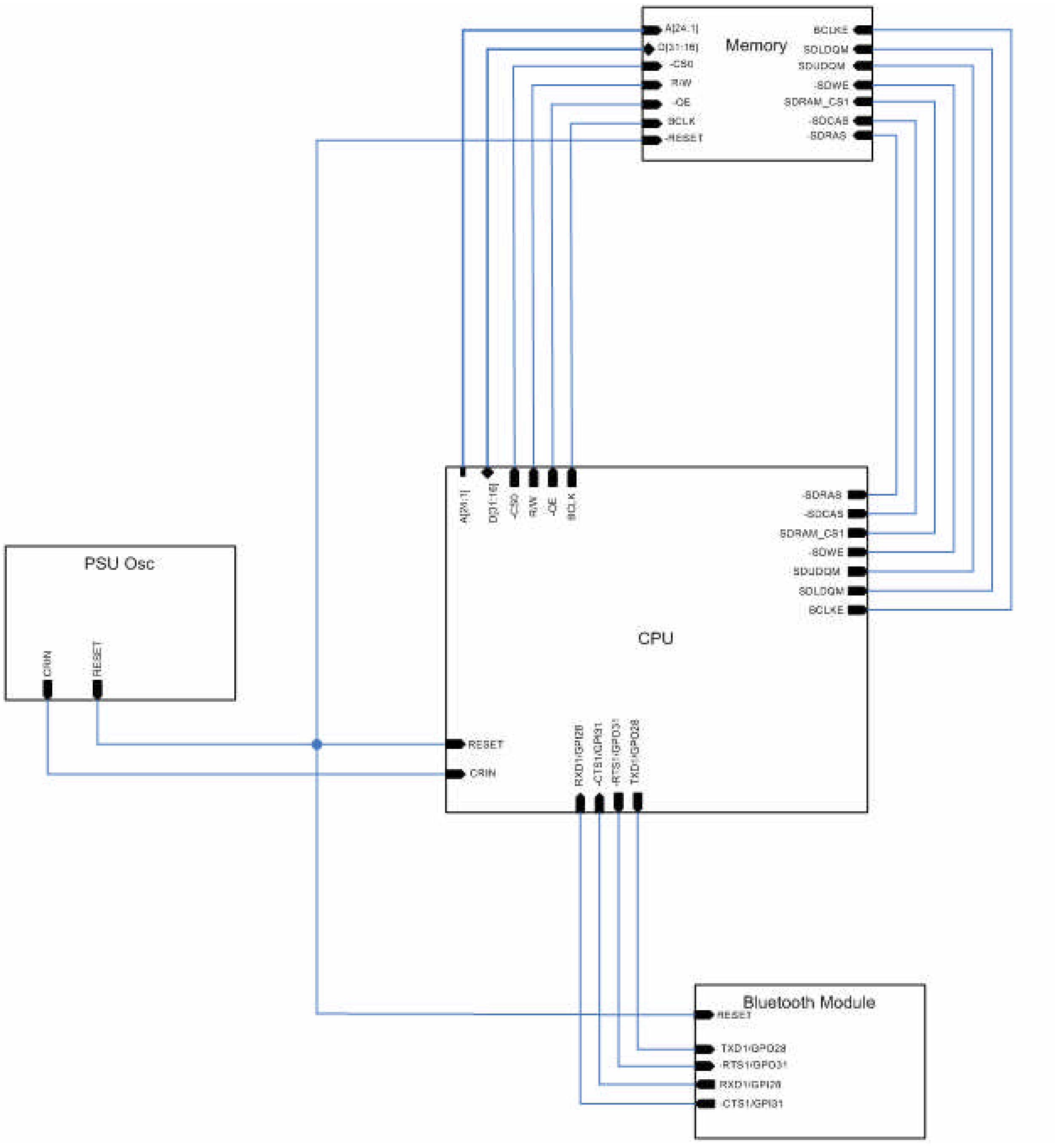


		Task Name	Duration	Start	Finish
43		⊟ **Full system testing**	**31 days?**	**Sat 4/1/06**	**Fri 5/12/06**
44	▦	Test Base and Phone Together	31 days?	Sat 4/1/06	Fri 5/12/06
45	▦	Update Design as Needed	31 days?	Sat 4/1/06	Fri 5/12/06
46		⊟ **Documentation**	**173 days?**	**Wed 9/21/05**	**Wed 5/17/06**
47		⊟ **PPFS**	**59 days?**	**Wed 9/21/05**	**Fri 12/9/05**
48	▦	Outline	4 days?	Tue 10/25/05	Fri 10/28/05
49	▦	Objectives	6 days?	Wed 9/21/05	Wed 9/28/05
50	▦	Design Norms	4 days?	Tue 10/25/05	Fri 10/28/05
51	▦	Research	49 days?	Thu 9/22/05	Mon 11/28/05
52	▦	Project Management	41 days?	Mon 10/17/05	Fri 12/9/05
53	▦	Preliminary Feasibility Analysis	22 days?	Thu 11/10/05	Fri 12/9/05
54	▦	Project Requirements	6 days?	Mon 11/21/05	Mon 11/28/05
55	▦	Base Device	20 days?	Mon 11/14/05	Fri 12/9/05
56	▦	Client Program	15 days?	Mon 11/21/05	Fri 12/9/05
57	▦	Conclusion	15 days?	Mon 11/21/05	Fri 12/9/05
58	▦	Proofread/revise	15 days?	Mon 11/21/05	Fri 12/9/05
59		⊟ **Local Documentation**	**141 days?**	**Tue 11/1/05**	**Fri 5/12/06**
60	▦	Base Station Hardware	141 days?	Tue 11/1/05	Fri 5/12/06
61	▦	Base Station Software	75 days?	Tue 1/31/06	Fri 5/12/06
62	▦	Client Code	82 days?	Fri 1/20/06	Fri 5/12/06
63	▦	Enclosure	71 days?	Mon 2/6/06	Fri 5/12/06
64		⊟ **Presentations**	**157 days?**	**Mon 10/10/05**	**Fri 5/12/06**
65		⊟ **Introduction Presentation**	**3 days?**	**Mon 10/10/05**	**Wed 10/12/05**
66	▦	Write Presentation	3 days?	Mon 10/10/05	Wed 10/12/05
67	▦	Practice Presentation	3 days?	Mon 10/10/05	Wed 10/12/05
68		⊟ **Feasibility Presentation**	**6 days?**	**Mon 11/21/05**	**Mon 11/28/05**
69	▦	Write Presentation	6 days?	Mon 11/21/05	Mon 11/28/05
70	▦	Practice Presentation	6 days?	Mon 11/21/05	Mon 11/28/05
71	▦	Other Presentions	55 days?	Tue 2/28/06	Fri 5/12/06
72		⊟ **Final Report**	**19 days?**	**Fri 4/21/06**	**Wed 5/17/06**
73	▦	Assemble and Compose Doccun	19 days?	Fri 4/21/06	Wed 5/17/06
74	▦	Proofread	14 days?	Fri 4/28/06	Wed 5/17/06
75	▦	Status Reports	156 days?	Tue 10/11/05	Fri 5/12/06
76	▦	Schedule	26 days?	Sat 11/5/05	Fri 12/9/05
77	▦	Website Maintenance	155 days?	Mon 10/17/05	Wed 5/17/06
78		⊟ **Miscellaneous**	**177 days?**	**Thu 9/8/05**	**Wed 5/10/06**
79	▦	Lecture	177 days?	Thu 9/8/05	Wed 5/10/06

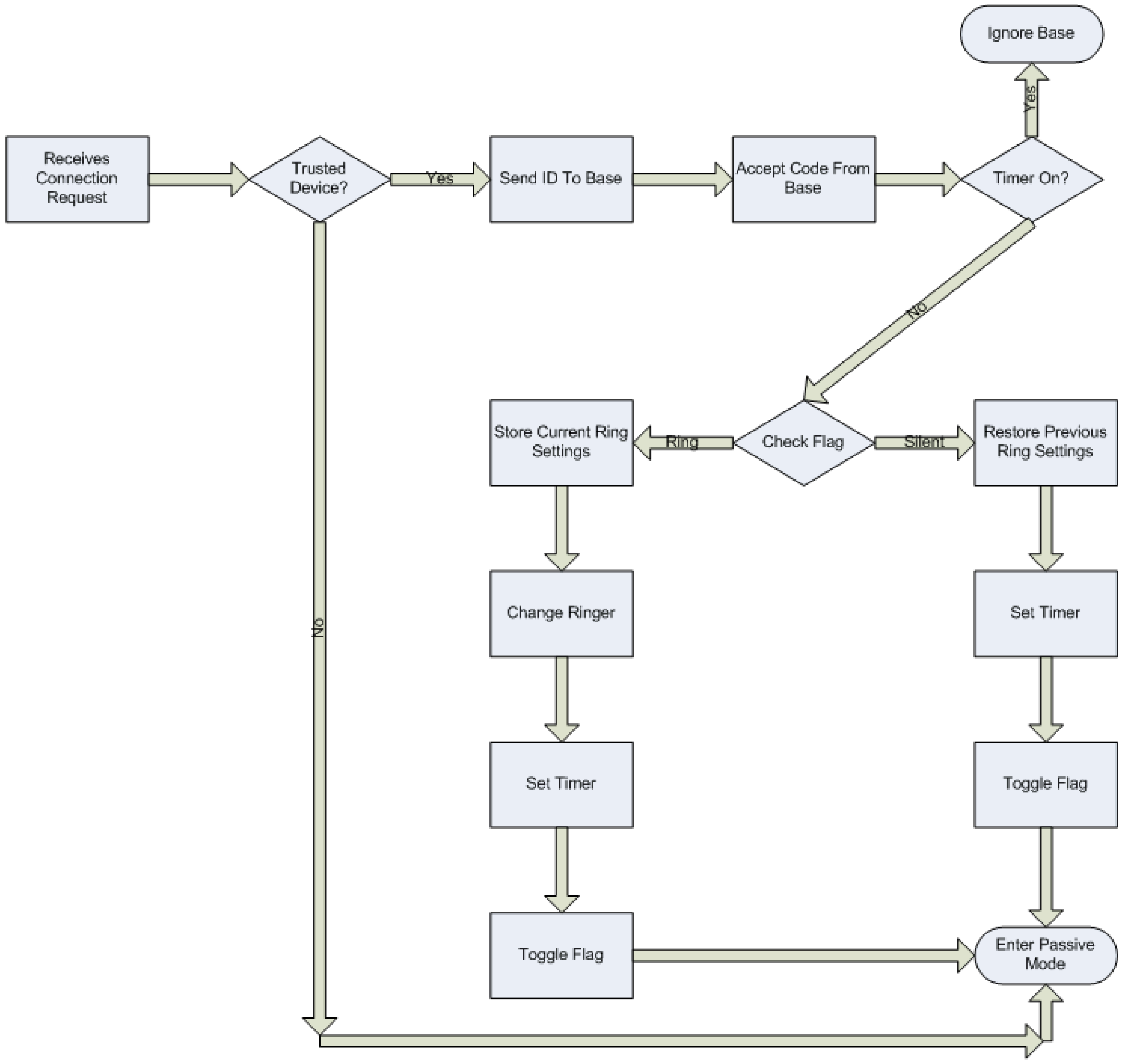
C. Final Prototype Schematic



D. Production Design



E. Client Program Flow



F. Software Documentation

Available at X:\Team6\Code\Documentation

G. Full Client and Base Station Code

Available at X:\Team6\Code

H. Client User Manual

BlúGate Application User's Manual

BlúGate is an application for the SymbianOS that allows a BlúGate device to automatically turn your cellular phone to vibrate or silent profile. Upon detecting a device again, the user's settings will be resorted. If a device is not detected for a second time, the user's settings will be restored after two hours. These devices are purchased by companies that wish to have minimum cell phone disturbances within a specified area. As a user of this application, you have the right to determine if you want the application running or not. This document explains how to use the applications, and what each command does.

Opening the Graphical User Interface (GUI)

In order to start the GUI, first you must open the applications menu on your cell phone. Then, scroll through the list of applications until you see the symbol. Click on the options button, and then select Open.

Status Setting

The status is the setting that allows you to choose if you want to allow a BlúGate device to automatically change your phone to silent or vibrate mode. There are two options for this setting, Running and Stopped. If it is set to Running, then a BlúGate device will be able to automatically change your settings if you pass by. If it is stopped, your settings will not be changed.

To change this setting, select the options button. Then, scroll through the choices until you have "Toggle Status" selected. Next, press the Ok button. This should change the setting, and you should immediately see the change on the screen. If the status was Running, and a BlúGate station had changed the phone settings, then the previous settings will be restored upon toggling status.

Mode Setting

The mode is the setting that allows you to choose which profile the BlúGate device will switch your phone to. There are two possible settings for mode: Silent or Vibrate. If the mode is set to silent, your phone will switch to Silent profile when activated by a BlúGate device, but only if the Status is set to Running. If the mode is vibrate, your phone will be set to Vibrate profile.

To change this setting, select the options button. Then, scroll through the choices until you have "Toggle Mode" selected. Next, press the Ok button. This should change the setting, and you should immediately see the change on the screen.

Exiting the Graphical User Interface (GUI)

In order to exit the application, you can either select the "Back" button, or open the options menu and select the "Exit" option. If the status is set to Running, a process will continue in the background that will open the GUI again if a BlúGate device is detected. The GUI will stay open until the user closes it. This allows the user to know when settings have been changed.