CELLSYNC

Final Design Report

Team 7

Engineering 340

Team Members:
Keith Conrad, Phil Overbeeke, Matt Gardner,
Jeffrey Enahoro, and Andrew Stutzman

Team Advisor:
Dr. Robert Bossemeyer

May 18, 2009

Revision 2.0

Revision History

Date	Comment	Author	Revision
04-05-2009	Title page and rough outline	Phil Overbeeke	Rev .1
04-24-2009	Started to add content written by Keith	Phil and Keith	Rev 1.0
04-26-2009	Added more content	Team	Rev 1.1
04-28-2009	Added Appendix content	Phil	Rev 1.2
04-29-2009	Submitted Draft Report to Prof. Bossemeyer	Team	Rev 1.5
05-12-2009	Reorganized the Appendices and added content	Phil Overbeeke	Rev 1.6
05-12-2009	Added content based on Prof. Bossemeyer’s comments	Team	Rev 1.7
05-17-2009	Final Report edits completed	Team	Rev 1.9
05-18-2009	Submitted Final Report to Prof. Bossemeyer	Team	Rev 2.0

Executive Summary

Since the advent of the telephone in the late 19th and early 20th centuries, the telephone has been an integral part of everyday life. While standard service, Public Telephone Switch Network (PTSN), has not changed drastically since the early 20th century, significant advancements in phone technology have been made. Voice over Internet Protocol (VoIP) systems allow calls to be made via the internet through computers. Wireless service providers (WSP) have brought the cell phone into the hands of millions. In 2005, over 210 million Americans subscribed for cell phone use [22]. That is over 2/3 of the American population of 301 million [22]. As of June 2008, 17.1% of cell phone households no longer used their landline telephone [7]. As this trend of wireless substitution continues, a gap exists between the "old technology" of landline systems and the "new technology" of the cell phone. CellSync bridges that gap by allowing the user to connect their cell phone to their landline system.

CellSync allows the user to send or receive phone calls from their "Plain Old Telephone Service" (POTS) phone through their cell phone. Existing landline phones can answer calls made to the user's cell phone as well as place calls using the user's cell phone plan. This prevents the user from needing to carry their cell phone at all times while in the home or office. As well, this allows the user to drop their landline service while retaining the conveniences of their "home" phone. This will save the user frustration from handling two phone services and money from paying two phone bills. Research shows that persons can save upwards of $33 a month using their cell phone exclusively [7].

CellSync is powered using Asterisk running on a Linux distribution. Developed by Digium, Asterisk is a telephony engine and toolkit that allows for networking of the telephone system. The Asterisk software drives the ZapMicro Telephone Network Interface card (TNIC). This card interfaces the landline system with the computer. Chan-Mobile is an Asterisk channel driver that allows a cell phone to be used as a telephone signaling interface via Bluetooth. CellSync will utilize the chan-mobile channel driver to interface cell phone call information with Asterisk which will route the call to the landline phones.

While cell phone/landline interfacing devices exist on the market, CellSync provides a method ideal for users looking for less expensive alternatives with greater functionality and flexibility. The team hopes to allow for voice mail access, text messaging, voice activation and other cell phone features. Similar products range from $100-$300, while CellSync aims to market this product to the consumer at a price around $100.

Table of Contents

Figures

Tables

Terms and Abbreviations

Term or Abbreviation	Definition
ADC	Analog to Digital Converter
AGI	Asterisk Gateway Interface
ANSI	American National Standards Institute
BOM	Bill of Materials
CD	Compact Disc
CE	European Conformity
DAC	Digital to Analog Converter
DAHDI	Digium Asterisk Hardware Device Interface
DMA	Direct Memory Access
DSP	Digital Signal Processing
FCC	Federal Communications Commission
FPGA	Field Programmable Gate Array
FXO	Foreign Exchange Office
FXS	Foreign Exchange Station
GUI	Graphical User Interface
ID	Identification
IO	Input / Output
IP	Intellectual Property
ISM	Industrial, Scientific, and Medical Band
LCD	Liquid Crystal Display
NIST	National Institute of Standards and Technology
OS	Operating System
OSS	Open Source Software
OSHA	Occupational Safety and Health Administration
PBX	Private Branch Exchange
PC	Personal Computer
PiaF	PBX in a Flash
POTS	Plain Old Telephone Service
PPFS	Project Proposal and Feasibility Study
PTSN	Public Telephone Switch Network
RAM	Random Access Memory
REN	Ringer Equivalence Number
RoHS	Restriction of Hazardous Substances Directive
SCC	Standards Council of Canada
SIG	Bluetooth Special Interest Group
SDRAM	Synchronous Dynamic Random Access Memory
TCP	Transmission Control Protocol
TNIC	Telephone Network Interface Card
UL	Underwriters Laboratories
USB	Universal Serial Bus
VoIP	Voice over Internet Protocol

1. Introduction

1.1 Senior Design Engineering

The engineering program at Calvin College culminates in courses Engineering 339 and 340. These courses aim to serve as a transition from the academic setting to the professional world through the design process of a project chosen by each team. The teams are formed of three to five students who select a project in their concentration. The first semester, Engineering 339, focuses on work in team formation, project identification, and a project feasibility study. Class lectures focus on state-of-the-art product design, teamwork, design norms, communication, management, research, ethics, and conflict resolution in the light of a reformed Christian mindset. The second semester, Engineering 340, emphasizes completion of the design work initiated during the first semester. Project work involves developing a prototype designed for primary functionality while entailing task specifications in the light of design norms for engineering analysis. In May of 2009, a final presentation is given detailing the senior design project.

1.2 Team 7: CellSync

CellSync is a team comprised of five senior electrical and computer engineering students at Calvin College. The team members are: Keith Conrad, Jeffrey Enahoro, Matt Gardner, Phil Overbeeke, and Andrew Stutzman. The whole team is pictured in Figure 1.1.

1.2.1 Keith Conrad

Keith Conrad has an interest in electrical engineering that includes analog electronic design, especially in audio applications, and power systems. He also competed with the Calvin College Knight's Swim Team of which he was captain for his senior year. He will be employed at D. C. Cook Nuclear Power Plant and plans on graduate studies in business and education.

1.2.2 Jeffrey Enahoro

Jeffrey Enahoro is from Nigeria. Other than engineering, he enjoys helping others. He has volunteered in a couple of places, one of them being Urban Promise, Wilmington, Delaware where he gained skills in organization, planning, and perseverance. He has served in leadership positions in his school, one of them being a Knollcrest East Serving Others (KESO) leader.

1.2.3 Matt Gardner

Matt Gardner is from Worthington, Ohio. He worked during the summer at SEA Inc. where he gained experience and knowledge in forensic science. He attended Thomas Worthington High School before attending Calvin College. His future plans are to enter the field of fire investigations.

1.2.4 Phil Overbeeke

Phil Overbeeke is from the Grand Rapids area. His interests in electrical engineering include electrical devices and hardware. Outside of engineering he enjoys travelling, studying genealogy, and socializing. He has a strong desire to live abroad for either an engineering career or another line of work.

1.2.5 Andrew Stutzman

Andrew Stutzman is from Kutztown, Pennsylvania. While working as an intern at Ametek Aerospace this summer, he developed skills in firmware design as he explored the challenges of airplane fuel gauge engineering. Andrew attended Kutztown Area High School before coming to Calvin. He is a member of the cross country and track and field teams at Calvin.

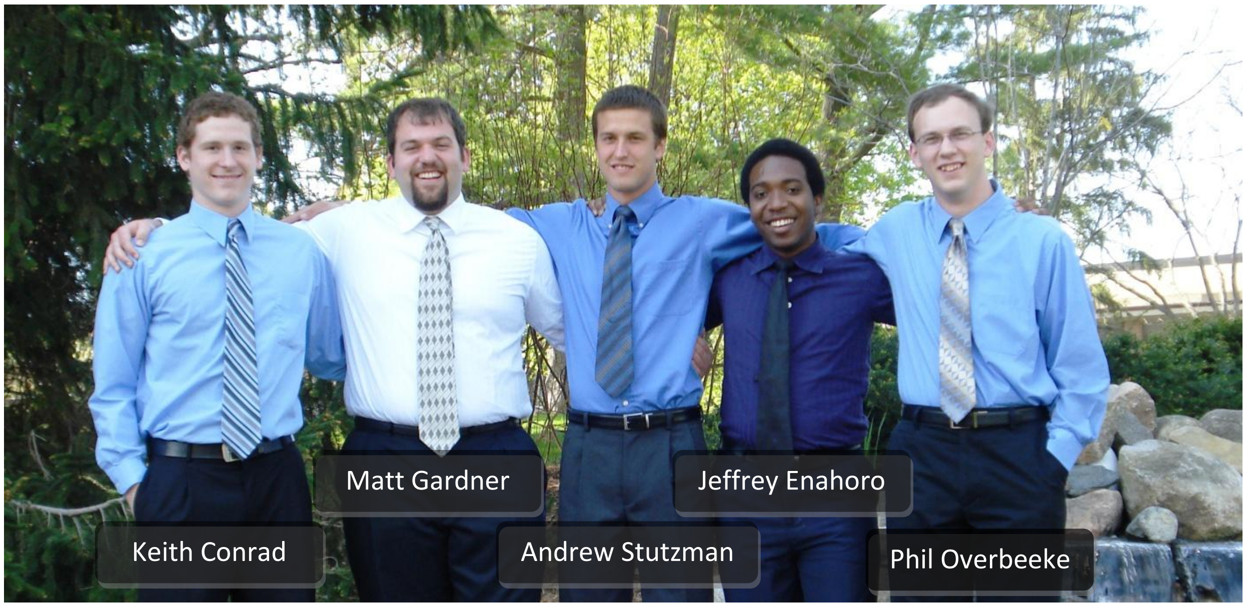


Figure 1.1: Team 7 – CellSync

1.3 Introduction to the Problem

In today's society more people are exclusively using cell phones as their means of telecommunication. This increases the importance of the cell phone and results in a decrease in use of preexisting landline systems that are in homes. For example, college students who live in houses often do not have a landline phone for cost reasons. This has led to an increasing number of unused landline systems in homes.

Furthermore, other homes are bearing the cost of paying for two phone bills, one for their landline and the other for their cell phone. Landline systems lack several features common to cell phones whereas cell phones lack the convenience of the landline phone placement throughout the household. CellSync aims to bridge these convenience gaps providing the functionality of the cell phone with the availability of the home phone.

Since most recent cell phones use Bluetooth as a means of communication with peripheral devices, this wireless technology can be utilized to communicate between the cell phone and the landline phones. The CellSync product aims to utilize this to allow the user to send and receive calls from the landline phone through their cell phone either replacing or co-existing with their landline phone service.

2. Background

There are a few key elements involved in the inspiration and development in this product. The seeming necessity of owning a cell phone in today's society has led to the growing trend toward cell phone users abandoning their landline phones. According to a study done by Nielsen Mobile, as of June 2008, 20.2 million households, or 17.1 percent of all U.S. households use their cell phone exclusively over their landline phone. These households use 45 percent more minutes on their cell phone compared to landline users, but pay only 10 percent more for their cell phone service. A single-person household nets an average saving of $33 per month by abandoning their landline service. This cost decreases by $6.69 for each additional wireless subscriber. Exclusive cell phone use has grown by 3-4 percent per year, and the trend doesn't seem to be slowing [7]. As well, the number of Bluetooth enabled cell phones has been increasing annually from 65% in 2005 to 69% in 2007 to 81% in 2008 [12][16]. As stated by one reviewer: "Bluetooth has become virtually a ubiquitous technology, and it has become pretty much a standard in most new cell phones" [12].

This has provided the opportunity for a product to capitalize on a user's desire to maintain the convenience and familiarity of their home phone while taking advantage of their cell phone plan. The predominance of Bluetooth capability in phones has provided a convenient gateway between cell phones and landlines. The availability of a wide range of Bluetooth compatible hardware and freely available, highly supported Private Branch Exchange (PBX) software has made developing a device that integrates cell phone and landline technology realizable and practical.

The Bluetooth capability standard in most phones and the widespread availability of Bluetooth compatible devices such as development kits, modules, and microcontrollers has allowed this project to be feasible. Bluetooth is a short range wireless connectivity standard defined by the Bluetooth Special Interest Group (SIG). The Bluetooth SIG is comprised of the major leaders in telecommunications, computing, and network industries, including "Promoter member" companies Ericsson, Intel, Lenovo, Microsoft, Motorola, Nokia, and Toshiba as well as thousands of "Associate" and "Adopter" member companies. Membership to the Bluetooth SIG is required to market a product with the Bluetooth logo or the Bluetooth name; such as "Bluetooth Enabled", "Bluetooth Capable", or simply just "Bluetooth". The Bluetooth protocol operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec. The 2.4 GHz ISM band is available and unlicensed in most countries [6]. Several stacks have been developed for programming for Bluetooth. Several Windows based stacks are available, while BlueZ is the official Linux Bluetooth stack.

Private Branch Exchange software is readily available as freeware or for purchase. Asterisk is heavily developed and highly supported open source software. Due to free licensing of the software, users have contributed features and functionality and have reported and corrected bugs they have come across in their own development. According to the Asterisk website:

> "Users can create new functionality by writing Dialplan scripts in several of Asterisk's own extensions languages, by adding custom loadable modules written in C, or by implementing Asterisk Gateway Interface (AGI) programs using any programming language [24]."

The flexibility and reliability of this software will facilitate the development of this project.

3. Objectives

3.1 Definition of Success

There are multiple levels of success CellSync strives to achieve. In this section, "product success" is defined by a set of operating criteria the device must adhere to. "Project success" is defined by CellSync's ability to become a fully developed product suitable for the marketplace. Within each of these definitions are levels of success ranging from minimum standards for the product and the project to the most ideal results for the product and the project. As well, beyond the product and project success, the design team sees this project as an opportunity to learn and develop skills that will aid them in future endeavors.

3.1.1 Definition of Product Success

The success of the product will be defined by the functionality of the device. Primarily, the device should be able to send and receive calls from landline phone via the user's cell phone. While any means of implementing this would be considered a successfully working product, further success would be defined by additional criteria.

For the scope of the senior design course, CellSync has determined that developing a method that can send and receive calls via Asterisk on a dedicated PC shall be sufficient "proof-of-concept". The system will be required to properly accept cell phone and landline calls and route the calls to the landline handsets. This system shall be easy for non-technical users to use in both the connecting and adding of a cell phone to the CellSync device and in the sending and receiving of calls. The device should automatically pair with previously added cell phones and have a reasonable operating range of more than a few feet. Additionally, the device must have the ability to transfer calls to at least two handsets. This method will be complimented by a barebones PC prototype containing only the needed hardware and software to realize this method. This system should closely resemble the specifications of a stand-alone product.

Ideally, the design would be a standalone device that operates independently of a personal computer. As well, this device should be a plug-and-play type system that requires very little setup or installation by the user. The device should be able to operate with or without existing landline systems. The device should have a ringer equivalence number (REN) of 5. This would keep the ringer load competitive with the ringer load a typical phone service provider would allow; allowing the user to use as many landline phones as they are accustomed to. The stand-alone product would have no input/output devices only requiring the user to type in a 4 or 16 digit pin number to pair their phone. The Bluetooth range of the device should be a reasonable room size of 10 feet or more. As well, the device should have the ability to pair with a minimum of two phones at any given time. If the previous design criteria are met, the team would like to integrate cell features such as voice dialing, voicemail, and text messaging to the system. The closer the team can get to achieve these design criteria, the more successful the product will be.

3.1.2 Definition of Project Success

The success of the project will be defined by CellSync's ability to create and develop a marketable product. Minimally, this requires that the group develop a proof of concept for the device that thoroughly displays that the CellSync concept is realizable. This includes a proof that a device can be developed that meets the criteria outlined in section 3.1.1. Additionally, the CellSync design must be a design that can be brought to market for a reasonable price. The ideal market price has been defined as $100; however, a price that keeps the device competitive with similar products will be considered a success. This market price must assume costs for manufacturing, parts, overhead, labor and others. Beyond pricing, the product must exhibit qualities that would deem it a marketable product, such as those of being useful, user-friendly, and reliable. The marketability and cost of the device must be investigated thoroughly to determine whether the CellSync product will be a successful project. Marketability and budget are explored in sections 5 and 10, respectively.

3.1.3 Definition of Team Success

The design team sees this project as an opportunity to learn. CellSync believes that the designs and ideas presented in this document are fully realizable but the learning points of this project are not contingent on a finished prototype or design. Engineering 339 and 340, Engineering Senior Design outlined in section 1.1, provides several lessons and objectives it aims to instill in senior engineering students. Team CellSync wants to take from this project the knowledge necessary to proposing, designing and developing a product as proposed by the course description. This includes learning what is necessary to develop a successful product and what can prevent a product from becoming successful as well as doing so in a Reformed Christian worldview. Continually, the group wishes to gain an understanding of what makes a project successful and how each member's individual contributions account for the team's success. The Engineering Senior Design course heavily focuses on group dynamics and team work. The team members have put a continual emphasis on team oriented success rather than individual accomplishments. The knowledge gained from this project will almost certainly allow the design team to walk away from this project feeling successful in fulfilling the objectives of the senior design course.

3.2 Main Requirements

The device needs to fulfill certain requirements before it can be deemed suitable for the marketplace. The device must fulfill certain operational criteria, must be intuitive, and must have a high level of connectivity.

3.2.1 Operational Criteria

The device must be able to send and receive calls from a landline phone through a cell phone. It must do so with reasonable call clarity, low noise or distortion, and within a reasonable delay. A device that distorts the user's voice to an unrecognizable or irritable level is unacceptable. As well, the device shall not delay a call such that the caller's or callee's speech is delayed beyond a level that interrupts normal conversation. Other functionality must be beneficial and fashioned in a way that features are not difficult to use and do not impede upon the performance of the main purpose of the device, sending and receiving calls.

3.2.2 Ease of Use

The main goal of this device is to be an asset to the user. A product that is not easy to setup, use, or maintain will not be considered. Ideally, we would like our device to be a plug-and-play type device that requires no complicated wiring or programming by the user. As well, the operation of the device should be reasonably straightforward and easy to use by all persons. This includes ease in phone pairing and ease in sending and receiving calls as highlighted in section 3.1.1. The main features of the device should be clear to a person seeing the device for the first time and should be viewed as a useful product.

3.2.3 Connectivity

The product should have the capability to connect to a broad spectrum of devices. That is, the device should work with most Bluetooth enabled cell phones and with existing landline systems. As well, system should have a reasonable communication range where the user can lay their phone within relative proximity of the device. Relative proximity can be defined as a standard room size of 10 to 15 feet. If the device requires the use of a computer, it should be able to connect to the computer easily and work with most desktop computer systems. While the finished prototype may not provide this broad level of connectivity, the design should be able to be easily implemented with other hardware. More specifically, the design should be able to define a system or approach for connecting a majority of cell phone and landline systems using most desktop computers.

3.3 Course Requirements

The objectives of this course can be divided into the fall and spring semesters. Most of the research and planning will be performed in the fall semester, while much of the design will be done in the spring semester. There are many due dates throughout the year that will keep the team on task. The Project Preliminary Feasibility Study (PPFS) is the major course document objective of the fall semester. The project must be completed by the end of the spring semester in time for the senior design presentation. The due dates for the fall semester are listed in Table 3.1. A Gantt chart of the deliverables and class requirements is given in Appendix A.

Table 3.1: Senior Design Deliverables - Fall

Date	Requirement
September 17, 2008	Team Defined
September 22, 2008	Project Defined
October 6, 2008	PPFS Outline due
October 13, 2008	Project Website posted
October 13, 2008	Project Poster due
October 20, 2008	Industrial Consultant Brief due
October 20, 2008	Introductory Oral Presentations begin
November 17, 2008	PPFS Draft due
December 3, 2008	Final PPFS due
December 12, 2008	Preliminary Design Memo due

The design and development of a prototype will be completed in the second semester. Again, there are several due dates meant to keep the group progressing throughout the semester. The work of the second semester will culminate in a presentation during Senior Design Night, May 9, 2009, and a final design report. Due dates for the second semester are listed in the table below, Table 3.2

Table 3.2: Senior Design Deliverables - Spring

Date	Requirement
February 25, 2009	Team Description for Banquet
March 2, 2009	Executive Summary
March 9, 2009	Oral Presentations #1
April 24, 2009	Website Upgrade
April 28, 2009	Draft Design Report Due
May 4, 2009	Banquet Posters Due
May 8, 2009	Oral Presentations #2
May 9, 2009	Open House/ Senior Banquet
May 18, 2009	Final Design Report Due

3.4 Product Requirements

3.4.1 Functional

The device must provide the user with a way to have their cell phone calls received and to place a phone call through their home landline handsets. Furthermore, it would be ideal to include cell phone technologies into the services that are allowable by the device. The device must allow the user to easily link their cell phone to their landline handsets and to place and receive the calls.

3.4.2 Power

The device must have its own power source. The power source must allow the device to function continuously while the device is on. The power source and circuitry must provide an adequate voltage for the electronic elements to correctly function.

3.4.3 Interface

3.4.3.1 Human-Machine Interface

The interface must be intuitive and easy to use. Pairing should be no more complicated than pairing of a any other typical Bluetooth device including Bluetooth earpieces and headsets used for cell phones; mice or keyboards used for computers; among several other recognized Bluetooth products. It is essential that the device provide visual and tactile feedback.

3.4.3.2 Visual

The visual interface must provide the user with information regarding the connectivity between the device and the cell phone. Also, it must provide the user with an interface to input passkeys for Bluetooth security purposes. The interface must therefore provide the user with timely feedback when the user provides tactile input to the device.

3.4.3.3 Tactile

The tactile interface must allow the user to input commands to the device in order to control its operation. This interface must be intuitive and simple, with enough functionality to fulfill the device operation requirements.

3.4.3.4 Data and Audio

The data and audio interface must be transparent with very little user input. The data transferred between the device and the cell phone must controlled by the device itself. The interface must provide reliable cell phone coverage. The interface must provide accurate data transfer between the cell phone and the device. This means that the device can have no unreasonable time delays or differences between using their cell phone and using their cell phone service via the device. The interface will be affected only by the tactile input from the user.

3.4.3.5 Personal ID Security

The communication between the cell phone and the device must be secure and must not allow unauthorized access to the signals passed between devices. This can be done with the use of encryption and security features of Bluetooth included in its specification. A third party must not be able to access the transmitted data without authorization.

3.4.3.6 Environmental

The main environmental goal is to be good stewards of resources. By designing a working system we will cut down on the number of landline systems being discarded. Our product will also make use of older computers that could be used to implement the product. The materials and components selected for the device and any proposed manufacturing processes, must have minimal impact on the environment.

3.4.3.7 Economic

The main economic goal for the product is to eliminate one telecommunications bill per house hold, specifically the landline service. The secondary economic goal is to be competitive with other devices similar to the CellSync product. The first goal will be accomplished by providing a complete working prototype for the system. This will allow people to cancel their existing landline phone bill while still having the advantages of a home phone network. The second goal will be accomplished by designing a product that can retail for under $150. The retail price of the product is discussed more in depth in the economic analysis section.

3.4.3.8 Safety

The safety goal for the product is to design one that does not contain any electrical shock hazards or sharp edges.

4. Christian Perspective

As Christians we believe that everything that we do is to be done to the glory of God, including our project. For this reason we aim to create our product with characteristics that we believe would be compatible with a reformed Christian worldview.

4.1 Design Norms

4.1.1 Integrity

We believe in doing things with integrity and excellence, and would like to do the same with our product. We would like our device to be easy to use. That is, we would like the user to be able to connect any Bluetooth enabled cell phone to the device without any difficult setup. This requires a user friendly interface between the user and the Asterisk/Linux configuration files. As well, we would like the user to be able to use the landline handsets they have readily available to them. We do not want the device to require a phone with specific buttons or functions that are not standard on normal phones. In this sense, we would like our device to be a plug-and-play type device.

4.1.2 Stewardship

God gave us the earth and the resources in it that we may be good stewards over them. These resources vary from natural resources; animal, plant, mineral and rock, to more individual resources; ability time, and money. The intent of this device is ultimately to provide a service that saves the user time and money. This device will take advantage of landline systems pre-existing in homes and offices, freeing the user from needing to purchase any networking equipment. This functionality also prevents waste from these systems going either unused or being removed and discarded. While this device allows the user to utilize his or her landline system, they need only to have cell phone service. This allows the user to save money over paying multiple phone bills saving the average user $33 per month.

4.1.3 Trust

We are called by God to be faithful and loyal, so our product must be the same. In designing this device, we would like to ensure that we produce a quality product. We would like our device to be functional and reliable without excessive need for maintenance or replacement. We want the user to be confident that the device they are using will stand up to day to day use and function properly repeatedly.

4.1.4 Caring

The device needs to be designed in such a way that phones and phone calls are secure. Bluetooth is an open source protocol available to anyone, on a wide variety of devices. We would like to ensure that only the user of the device can hear or contribute to his or her conversation. As well, as most phones are Bluetooth enabled, we want to ensure that a non-user is safe from having calls made from his or her phone via our device.

5. Market Research

5.1 Market Trends and Products

5.1.1 Cell Phone and Landline Use

In the US today, the population of cell phone users is growing much faster than the population of landline users. According to a poll by Harris Interactive, 85% of US adults have cell phones compared to the 71% having landline phones [16]. With the growing population of cell phone users, many of the landline phones are becoming useless and a liability to most people in the United States. Research by the National Center of Health Statistics showed that one out of every six households in the US does not have a landline phone but does have at least one cell phone. This shows that there is a growing rate of users changing over to cell phone only households [16].

Some people are realizing that they can survive without their landline phone service and decide to cut that bill from their budget. Others are reluctant to do this because they don't want to lose all of the advantages that landline services still have to offer. They end up paying for two phone bills. Landline phones offer a more permanent household service and the ability to network many handsets throughout the house. CellSync's product idea will allow those who are reluctant to switch over to only cell phone service to cut their landline bill. CellSync will achieve this by combining the advantages of cell phone and landline phone service into one powerful tool.

CellSync has developed a method for PC users to connect their cell phone service with their landline phones. As almost 60% of United States households own a PC, CellSync's method would be very realizable in several American homes. As well, this method serves as a proof-of-concept to develop a stand-alone product accomplishing the same goals.

5.1.2 Existing Products

There are some existing products in the market that are similar to CellSync. These include Xlink's Bluetooth Cell Gateway, Dock N' Talk by Phone Labs, and the Blue Raven Mobile Mate 1380. Some existing products are pictured in Figure 5.1, from left to right, Blue Raven Mobile Mate, Dock-N-Talk, Xlink Bluetooth Cell Gateway. Each of these products allows consumers to connect their mobile phones to their home landline phones. Xlink adds additional features such as caller ID, call waiting, and echo cancelation. Phone Labs has features such as voice recognition and voicemail. The Blue Raven Mobile Mate 1380 doesn't list any additional features, but is competitively priced at $99.00. Table 5.1 below lists the price of each product as well as the main features.

Table 5.1: Existing Market Product Comparison

Company	Product	Feature Set	Price
Blue Raven	Mobile Mate 1380	Make and receive cell phone calls from regular home phones	$99.00
Phone Labs	Dock-N-Talk	Connect up to 8 Bluetooth enabled cell phones Support for over 1,800 cell phone models Use local number portability to move landline number to cell phone and keep using landlines Charges cell phone while docked Voice recognition Voicemail access	$159.99
Xlink	Bluetooth Cell Gateway	Connect up to 3 cell phones to home phone system Caller ID, Call Waiting Voice and Speed Dialing Voicemail access	$169.99

Figure 5.1: Some Existing Market Products

5.1.3 CellSync's Place in the Market

CellSync allows for great flexibility in features. The advantage of using the Asterisk software is the nearly unlimited flexibility it allows. Like the other available products, it is possible to send and receive cell phone calls from a home phone. However, with this software it is possible to send and receive SMS messages from a home phone; include functionality so a call placed to your home phone will ring your cell phone even if you are out of the house, no more missed calls; setup voicemail profiles specific to every user in your household; as well as a host of other options and features the products listed above cannot offer.

6. Intellectual Property Concerns

Some patents exist in the category of landline and cell phone integration. These patents are summarized in Table 6.1 below. As described in the Market Research section, Xlink, Phone Labs, Blue Raven Technology, and others have products on the market that are similar to CellSync. These products make it possible to connect landline phones to cell phones while combining the advantages of each. CellSync's software approach is certainly unique to each of the discovered patents. However, it would be necessary to hire a lawyer to investigate CellSync's product and similar patents before marketing it in order to make sure that no patents have been violated.

Table 6.1: Patent Research

Patent Number	Description	Product	Year
6785517	Concurrent wireless/landline interface apparatus and method	An apparatus for interfacing at least one landline telephone service and at least one cell-type wireless telephone	2004
6959172 (by Dock N Talk)	Docking station for enabling landline telephones to send/receive calls via a docked mobile telephone	A docking station, which connects a cell telephone with one or more landline telephones	2005
7020488 (by Dock N Talk)	Communications unit, system and methods for providing multiple access to a wireless transceiver	A first wireless transceiver port operable to communicate with a first wireless transceiver operable to conduct wireless communications with a wireless base station	2006
7177664	Bluetooth interface between cell and wired telephone networks	An interconnect device for connecting a Bluetooth compliant cell telephone to one or more wired telephones on an existing wired network	2007
7363045	Systems and methods for exchanging data and audio between cell telephones and landline telephones	Connects landline to cell telephones using Bluetooth technology	2008

Asterisk is an open source software program. Products containing open source software may be sold, but certain restrictions apply. A program created with open source software must include source code and must allow distribution in source code form as well as in compiled form. If the product is distributed without source code, the source code must be publicly downloadable. The program must allow modifications and redistribution by others. Restrictions cannot be placed on other software that is distributed along with the licensed software. CellSync will operate within all of these restrictions. If CellSync were to develop a stand-alone product, proprietary code would be written to develop the product from scratch. Using the PBX approach for developing the device, the source code would be CellSync specific, reducing the computing power, memory capacity, and cache availability needed by the current CellSync approach. This would benefit the user by reducing the size of the device, reducing the cost of the device, and improving the intuitivism of the device.

7. Alternative Solutions

In order to most successfully achieve CellSync's project goal, allowing users to route their cell phone calls through their existing landline phones, the team investigated multiple design alternatives. Several methods were investigated in the first semester. The two methods dismissed from the first semester are discussed briefly in sections 7.1 and 7.2. A more thorough description of these designs can be found in the Project Proposal and Feasibility Study document completed for the first semester.
Additional alternatives were pursued in the second semester. These alternatives are described in sections 7.3 through 7.5. These alternatives are more closely related and contributed to the final design described in section 8.

7.1 Bluetooth Module

In this implementation, the Bluetooth module would connect to the cell phone and provide digital signal processing (DSP) so that the landline can "understand" the signal. The Bluegiga WT32 module was selected as the most viable candidate for this approach. This module has applications in wireless mono speakers, wireless speakers, VoIP handsets, and Hands-free car kits. The hands-free option was especially promising because it allows communication with cell phones and the audio data exchange between devices. The module is pictured in Figure 7.1.

Figure 7.1: Bluegiga WT32 Module

7.1.1 CellSync's Position on Module

The team's primary objective of the project is to allow cell phone calls to be made and received on landline systems. This module would allow a connection to a cell phone to be made, but the team is unsure of how that would be passed to the landline. While this module would allow for a small form factor device, it would be difficult to work with due to the proprietary limitations of cell phones.

Finally, this module would present the team with less functionality than is desired out of the design. This option would most likely only support an audio connection to mobile phones, which could be passed to landline handsets. The team would like to implement support for other cell phone features. Since this module is primarily an audio module it would be difficult to implement text or data based features.

7.2 Altera DE2 Board

The Altera DE2 development and education board, shown in Figure 7.2, has a field programmable gate array (FPGA) processor, synchronous dynamic random access memory (SDRAM), and a liquid crystal display (LCD). The BOOST Lite Bluetooth Baseband Core works with the DE2 board to allow for wireless Bluetooth communications. The team has used the Altera DE2 Development Board in its engineering coursework and building a device to achieve the desired CellSync functionality may be feasible.

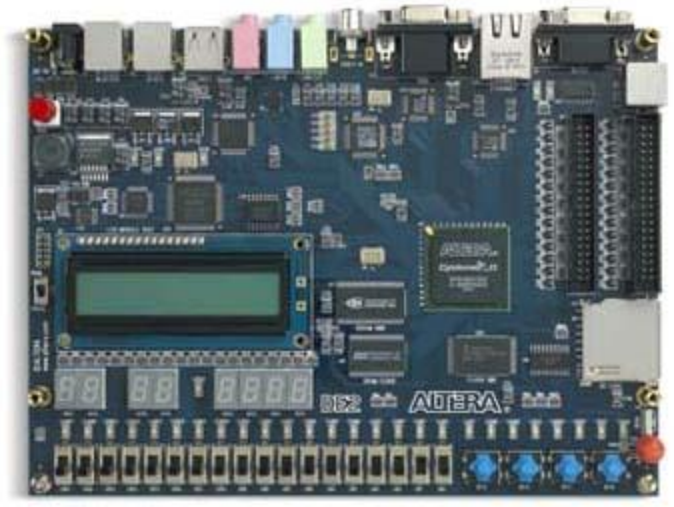


Figure 7.2: Altera DE2 Development Board

7.2.1 CellSync Position with Alternative

The coding of this design would be extremely difficult and time intensive. Significant effort would be spent attempting to integrate the needed components. Assuming that can be accomplished, more time would be needed to write the code to achieve the desired functionality. The possibility of finishing the design in the allotted time would be slim and the proposed alternatives from the first semester seemed more feasible

7.3 Development Board

The team hoped to develop the CellSync method on a PC then transfer the required data to a better equipped development board (compared to the DE2 Board). There are several development boards capable of running a Linux operating system. The Calvin College Engineering department has a more powerful Altera DE2 Development Board which is capable of running Linux. As well, there are several Linux distributions designed to run on embedded systems, such as NexusWare Core and MontaVista Linux. Additionally, there is an Asterisk based Linux distribution, AstLinux, designed for embedded system use.

7.3.1 CellSync Position with Alternative

Further research into this approach revealed one common denominator; Asterisk is not meant to be run on development boards. Several graduate and industry level research projects have gone into this alternative and while success has been found installing Asterisk on a development board, all studies concluded that it was very difficult and not an ideal implementation of Asterisk. A list of minimal requirements provided by the research projects disqualifies both DE2 boards. Boards that are better suited for this approach cost several thousand dollars and are well out of budget. As well, the time and effort involved in realizing this implementation would not be realistic for the time frame of the senior design course.

7.4 Operating System Selection

CellSync had identified two viable options for the operating system using the PC based approach. AsteriskNOW is a custom Linux distribution developed by Digium, the creators of Asterisk. AsteriskNOW contained only the applications, files, and components necessary to operate Asterisk and ran the most up to date version of Asterisk. PBX in a Flash (PiaF) runs Asterisk on an underlying CentOS Linux distribution customized for Asterisk. PBX in a Flash boasts being the most well documented Asterisk PBX distribution as well as the easiest to upgrade and configure.

7.4.1 PBX in a Flash

The team had initially decided to work with PBX in a Flash as it was indeed found to be the best documented and seemingly the easiest to configure and upgrade. The operating system was installed on the computer provided by Bob DeKraker during the January interim. After a short time of working with PiaF, and further research into the software, it became apparent that AsteriskNOW would be better suited for our project. Our plan at this time was to use chan-mobile, the Asterisk channel driver, to accomplish cell phone connectivity. PiaF was not well suited for this application and while PiaF was very well documented in several applications, this was not one of them.

7.4.2 AsteriskNOW

The teams desire to work with chan-mobile led to the selection of AsteriskNOW for the operating system. AsteriskNOW guaranteed the ability to use chan-mobile as compared to PiaF which was uncertain. As well, AsteriskNOW used the most recent distribution of Asterisk. This was ideal as there was documentation provided on the Asterisk homepage outlining the installation and configuration of chan-mobile with that version of Asterisk. Several frustrating weeks of work with AsteriskNOW led team members to seek help from Gary Draving, Calvin College Computer Science lab manager. Under Gary Draving’s advice, the team discarded AsteriskNOW for a general purpose Linux distribution. Several reasons led to this decision. Primarily, the lack of a graphical user interface made file management difficult on the AsteriskNOW machine. A general purpose distribution allowed for browsing, editing, and generally allowed for file management with greater ease and control as compared to AsteriskNOW. As well, AsteriskNOW lacked components needed to run chan-mobile. The Asterisk software on the AsteriskNOW distribution was complete and would be able to run chan-mobile but additional applications were needed. AsteriskNOW did not provide a well documented list of commands for working the underlying operating system. This is why PiaF was initially chosen. This also meant troubleshooting system errors and downloading required software repositories was unnecessarily difficult with AsteriskNOW. A general purpose Linux distribution would allow for this functionality to be achieved intuitively.

7.4.3 General Purpose Distribution Selection

The real catalyst for switching operating systems was the inability to get the Bluetooth libraries downloaded, configured, installed, and operational. Under Gary Draving's guidance and assistance, the team installed the Ubuntu 8.10 server operating system and Asterisk 1.4.24.1. The Ubuntu operating system lacked a GCC compiler required to install Asterisk. Downloading the GCC compiler was taking a considerable amount of time and for fear the download would timeout or become corrupt because of the poor download rate Gary Draving advised using OpenSuse. OpenSuse has several advantages over Ubuntu and is ideal to work with for a variety of reasons more thoroughly discussed in section 8.

8. Prototype Design – PBX with Asterisk

8.1 Prototype Design - The CellSync Method

The CellSync method is a means of connecting one's cell phone to their landline phone. This method is realized using the open source private branch exchange software Asterisk, a telephone network interface card and a Bluetooth USB dongle. Asterisk manages incoming and outgoing calls to and from both the user's landline and cell phone. A telephone network interface card interfaces call and speech data to and from the landline to the Asterisk software. The Bluetooth dongle relays call and speech to and from the cell phone to chan-mobile. Chan-mobile, an Asterisk channel driver, allows Bluetooth devices to be used as Foreign Exchange Office (FXO) or Foreign Exchange Station (FXS) channels.

8.2 The Computer

As a result of the feasibility study from fall semester, the team chose to develop the CellSync method using Asterisk software and a PC. Given the 14 week time frame and $300.00 budget, this is the method that was deemed most feasible. To more closely resemble a stand-alone device that would be brought to market, the CellSync method will be implemented on a barebones computer system. This system will better realize the overall look, feel, and performance of a CellSync device brought to market.

8.2.1 System Requirements

The CellSync method was born out of two different machines, the computer used for development and the computer used for the barebones system. Both computers had to achieve a minimum set of specifications defined by the operating system, the Asterisk software, and the ZapMicro TNIC.

8.2.1.1 Asterisk System Requirements

The CellSync method was developed as a "hobby system" as according to Table 8.1, from Asterisk: The future of Telephony. The CellSync method is designed for 3 channels; the landline FXO channel, the cell phone FXO channel, and the computer driven FXS channel. The recommended minimum system requirements for this system are a 400 MHz x86 processor, with 256 MB RAM, and using no more than 5 channels.

Table 8.1: comparison of Computer Systems

Purpose	Number of Channels	Minimum Recommendations
Hobby system	No more than 5	400 MHz x86, 256 MB RAM
Small Office system	5 to 10	1 GHz x86, 512 MB RAM
Small business system	Up to 25	3 GHz x86, 1 GB RAM
Medium to large system	More than 25	Dual CPUs, possibly also multiple servers in a distributed architecture

8.2.1.2 The ZapMicro Network Interface Card System Requirements

One of the main components of the CellSync method is the telephone network interface card. The TNIC contains the analog to digital and digital to analog converter and generates the home line signal (dial tone) and ringer signal. The team compiled Table 8.2 with the minimum and desired components for the card.

Table 8.2: Minimum vs. Desired Telephone Interface Card Requirements

Type	Number of Channels	FXO Compatible	FXS Compatible	FXO/FXS Card Included	Echo Cancellation	Price ($)
Minimum	1	NO	YES	YES-1Channel	NO	< 70
Desired	2	YES	YES	YES-1Channel	YES	50

These minimum requirements allowed for the selection of a specific card. Table 8.3 lists cards that met the requirements and seemed the most suitable for the project.

Table 8.3: Telephone Interface Card Options

Company	Product	Max Number of Channels	FXO Compatible	FXS Compatible	FXO/FXS Card Included	Echo Cancellation	Price ($)
Yeastar	TDM800	8	YES	YES	YES-2 Channels	NO	208.95
Rhino	R8FXX-EC	8	YES	YES	NO	YES	279.00
Sangoma	A20400	24	YES	YES	NO	YES	614.76
ZapMicro	ZMA400P	4	YES	YES	YES-1 Channel	NO	124.95

The ZapMicro ZMA400P11 was selected for the CellSync method. The component is reasonably priced at 124.95 and had one FXO and one FXS port. The maker of the card is also recommended for use with Asterisk.

This card had the following set of system requirements as listed on the ZMA400P datasheet:

- 500Mhz Pentium III or better processor
- 64MB RAM
- Available PCI Slot 5v

8.2.1.3 Operating System Requirements

The final limiting factors for the system requirements are those of the Operating system. The prototype model will be running OpenSUSE. OpenSUSE is a Linux distribution developed by the openSUSE Project sponsored by Novell. The minimum system requirements as according to the openSUSE website are:

- Processor: Intel: Pentium 1-4 or Xeon; AMD: Duron, Athlon, Athlon XP, Athlon MP, Athlon 64, Sempron or Opteron
- Main memory: At least 256 MB; 512 MB recommended
- Hard disk: At least 500 MB for minimal system; 2.5 GB recommended for standard system
- booting from CD/DVD drive for installation, or support for booting over network

From the development computer it was found the CellSync method requires at least 8 GB of hard drive allocation. Given this data a minimum set of system requirements can be compiled:

- 256 RAM
- 500Mhz Pentium III
- 8 GB hard drive
- Available PCI Slot 5v

8.2.2 The Development Computer

Two machines were used in the development of the CellSync method. The first machine was a Dell provided by Bob DeKraker, the Calvin College Engineering Lab manager. Development started on this machine, however, Gary Draving, the Calvin College Computer Science lab manager provided a second machine, a Chandelle. This was done for convenience reasons while switching operating systems from AsteriskNOW to openSUSE. The second machine provided that the progress made on the first machine would not be lost. Both development computers far exceeded the minimum operating criteria for the system.

8.2.2.1 Computer Configuration

The computer was installed with openSUSE 10.3. This operating system was recommended by Gary Draving as it is the Linux distribution he is most familiar with and is ideal for a development environment. The Bluetooth protocol for Linux, BlueZ-4.37, was download from the BlueZ.org website to the computer's desktop. The .tar.gz download file was moved to /usr/local for installation. The library installed on the system using the commands, comments proceed a ';':

```
# cd /usr/local ;move into the /usr/local directory
# tar xvfz bluez-4.37 ;extract file bluez-4.37
# cd bluez-4.37 ;move into the blue-4.37 directory
# ./configure ;builds a make file for the program in the directory
# make ;builds programs in the makefile
# make install ;installs programs in makefile throughout the computer
```

This was required for use with the Bluetooth USB adapter to communicate with the cell phone. Once development of the system is complete, the development computer will be mirrored onto the barebones computer for the final, operational, prototype.

8.2.3 The Barebones Computer

Difficulty came in selecting a machine for the barebones system. The desire was to create a system that closely resembled the minimum operating criteria for a final marketable product. The challenge was to find a computer that fell within the aforementioned criteria and fell within the budget. God's grace provided a computer that closely matched the desired needs and came free of cost. The machine was donated by Calvin Professor Emeritus Glen Van Andel. This connection was found through Bob DeKraker. The computer used for the barebones system is a Dell 4300S. To match the small form factor that a product brought to market would have, the computer was disassembled, stripped of unnecessary components and reassembled with a chassis composed of polycarbonate to show the inner workings of the system.

After stripping down the computer, the next step to build the chassis was to measure all the components that the system needed to contain. Similar components to the motherboard, power supply, power button, and hard drive were found in Google Sketchup's 3D Warehouse and dimensioned to the correct size as shown in Appendix B**.** After each component was measured the chassis could be designed to enclose the components in the smallest area possible. The team was concerned about the case overheating so extra space was added to allow for increased air flow. The final dimensions of the polycarbonate chassis are shown in Appendix B. The case needed to allow for several input and output devices. The team needed to account for the power cord, telephone lines for the TNIC, the mouse, keyboard, and monitor to use the CellSync menu; and the power button. The team decided to use adapters for the phone lines and power supply to provide a professional look to the prototype. The prototype was designed with a small 7.25 inch by 3 inch door to allow access for the mouse, keyboard, and monitor. A small door allows the user to unhook the I/O devices when desired. After obtaining the adapters, the back side of the prototype was designed as shown in Appendix B. A power button was mounted in the center of the front panel. A mockup of all these pieces can be seen in Appendix B. After all the sides were designed the last step was to put all the parts and sides together in Google Sketchup to ensure that all measurements were correctly dimensioned. This proved to be an important step as the original design contained a .25 inch error on the left and right sides. After these parts were corrected, the design fit together properly as shown in Figure 8.1.

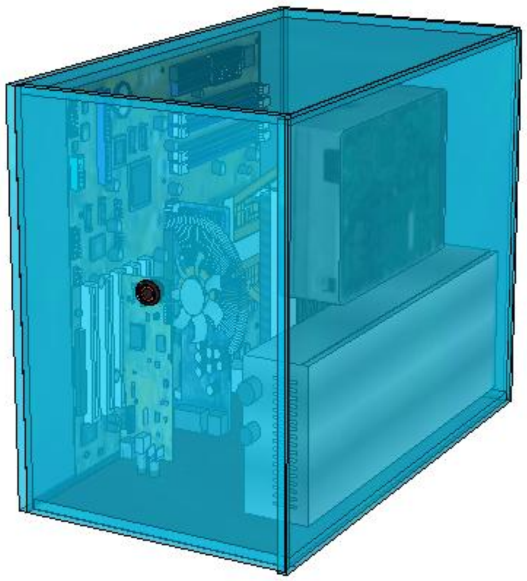


Figure 8.1: Google Sketchup of Prototype Design

After a successful design was completed in Google Sketchup the team constructed the chassis out of polycarbonate in Calvin's Machine Shop with the help of Phil Jasperse. After the construction was completed the developmental computer was mirrored over to the prototype computer. The process of mirroring one computer to another is when all files are copied from one computer's hard drive and placed in another computer's hard drive in the exact same memory locations. The results of the prototype can be seen in Figure 8.2.

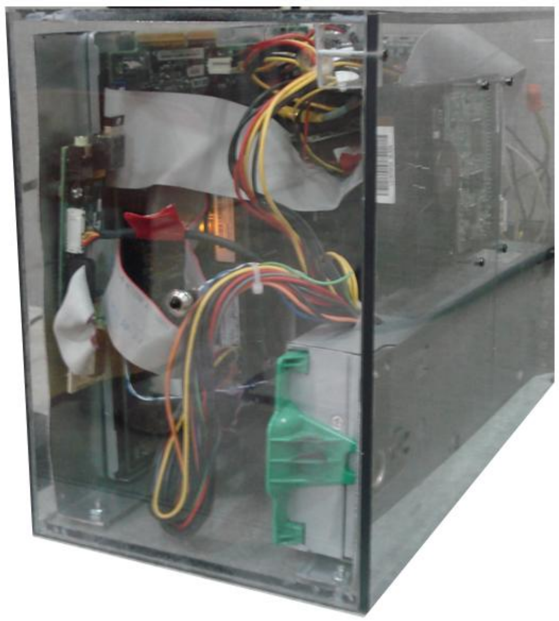


Figure 8.2: Photograph of CellSync Prototype

8.3 What is Asterisk?

As stated on the Asterisk home page:

> "Asterisk is the world's leading open source PBX, telephony engine, and telephony applications toolkit. Offering flexibility unheard of in the world of proprietary communications, Asterisk empowers developers and integrators to create advanced communication solutions...for free."

Developed by Digium, Inc, Asterisk is the leading open source PBX software. Asterisk contains the applications to use voicemail, host conferencing, call queuing and agents, music on hold, and call parking [20]. These are all standard features built into the software. Asterisk allows for flexibility unprecedented in other PBX software packages. The Asterisk website contains hundreds of available patches with positive feedback and developer support from user forums. The robust and flexible nature of Asterisk led the team to use this approach to build the CellSync product.

8.3.1 Asterisk System Configuration

Asterisk is not fully functional, or barely functional for that matter, after installation. Asterisk was downloaded using the command:

```
#svn checkout http://svn.digium.com/svn/asterisk/branches/1.6.0/ ;install
file from repository
```

The system was configured for installation and installed using the commands:

```
# ./configure
# make
# make install
# make check config ; check that 'make install' encountered no errors
# make samples ; make samples of program files
```

At this point, the Asterisk software is ready to be edited for user defined operation. These operations are controlled by user defined settings including Asterisk Add-ons and the extensions.conf file.

8.3.2 Asterisk Add-Ons

An essential part of the CellSync method is an Asterisk Add-on, chan-mobile. According to the chan-mobile home page: "chan-mobile is an Asterisk channel driver that allows you to use Bluetooth devices as FXO or FXS channels." The Asterisk Add-ons repository was downloaded, installed, and configured using the commands:

```
# cd /usr/src
# svn checkout http://svn.digium.com/svn/asterisk-addons/branches/1.6.0/
# cd asterisk-addons
# make clean ; 'make' command removing unneeded files after execution
# make install
```

These commands download the add-ons repository version 1.6.0 to the source folder on the computer and install the various software components throughout the computer as needed. The file of interest was installed to the /usr/etc/asterisk folder. The mobile.conf file allows for the addition of Bluetooth devices to the Dialplan. The Dialplan is defined and outlined in section 8.3.5. For the CellSync method, this allowed for the use of cell phones in the Dialplan.

8.3.3 Asterisk GUI

In addition to the asterisk add-ons the Asterisk Graphical User Interface GUI 2.0 was downloaded. The graphical user interface allows for editing of essential configuration files, monitoring of Asterisk, setting up of calling rules, setting up hardware, and other essential functions. The GUI was downloaded, installed, and configured using the commands:

```
# svn co http://svn.digium.com/svn/asterisk-gui/branches/2.0 asterisk-gui
# cd asterisk-gui
# ./configure && make && make install ; '&&' concatenates commands
```

After installing the GUI the http.conf and manager.conf files in the /usr/etc folder were edited to access to the computer over an internet connection. Following the link http://0.0.0.0:8088/asterisk/static/config/index.html accesses the Asterisk online GUI.

8.3.4 DADHI and Libpri

To interface the hardware with Asterisk, the Digium Asterisk Hardware Device Interface (DAHDI) Linux kernel packages needed to be installed. Additionally, the Libpri library needed to be installed for PRI support. The primary rate interface (PRI) is a telecommunications standard for carrying multiple DS0 voice and data transmissions between a network and a user. These packages were downloaded, installed, and configured using the commands:

```
# svn http://svn.digium.com/svn/dahdi/linux-complete/trunk
# cd dahdi-linux-2.0.0
# make clean
# make
# make install
# cd dahdi-tools-2.0.0
#./configure
# make
# make install
# make config ;install makefile for configuration file
```

After doing that it is necessary to redo the commands to install and configure Asterisk. After this is complete running the commands:

```
# dahdi_genconf ;generate configuration file
# dahdi_cfg ;configure DAHDI hardware
```

Generates dahdi-channels.conf and configures the FXO and FXS channels on the TMD400P TNIC. This process configures the ZapMicro Telephone Network Interface card and allows the card to work with Asterisk.

8.3.5 The Dialplan

According to Asterisk: The Future of Telephony, the Dialplan is “truly the heart of any Asterisk system, as it defines how Asterisk handles inbound and outbound calls. In a nutshell, the Dialplan consists of a list of instructions or steps that Asterisk will follow.” The Dialplan basically consists of rules and instructions that Asterisk uses to handle and direct the calls that come into and go out of the system. The Dialplan is located in the extensions.conf file located in the /etc/asterisk directory. See Appendix C for the complete extensions.conf file. It consists of contexts within the file that are used to separate and control various parts of the telephony system. The contexts only communicate between one another when specifically called to. The contexts generate the system info Asterisk uses to communicate to and from the home phones and the outgoing phone signal.

Many of the contexts were generated automatically by the asterisk system, some of which were necessary for our system to work and others were not. Other contexts we created. Some contexts, like globals, were used to specify variables which were used throughout the Dialplan.

The four main contexts of the Dialplan that the team worked with and created are incoming_landline, incoming_cell, outgoing_landline, outgoing_cell. In incoming_landline, the first two lines check the caller-id of the incoming calls from the FXO port; then send the incoming call to the FXS port and is received from the landline handset phone. The fourth line sends the caller to voicemail if no one picks up the call after a certain number of rings.

In incoming_cell, the first line sends the calls that come from chan-mobile to the FXS port. The chan-mobile identifies the caller-id from cell phones and sends the call to the incoming_cell context on the Dialplan. The second line answers the call that stops the cell phone from ringing and sends it to the third line which sends the call to the voicemail. The voicemail is defined by a context in another file called voicemail.conf.

In outgoing_landline, the line identifies that by pressing 6 before a four-digit number, the call will go out through the FXO port to the phone service. In the outgoing_mobile context, when 9 is dialed followed by 10 digits, the first line receives a call made from the landline handset. The next few lines (from “exten = _9XXXXXXXXXX,n,Dial(Mobile/135798642Phil135798642/${EXTEN:1},15)” to “exten = _9XXXXXXXXXX,n,Dial(Mobile/135798642Keith135798642/${EXTEN:1},15)”) dial the 10-digit number through the specified cell phones. It attempts to make the call through the first cell phone on the user list. If that cell phone is not paired to the system, Asterisk will immediately realize that the call was unsuccessful and will attempt to make the call through the next cell phone on the user list. Asterisk continues this procedure through the cell phone users in the order that they were added to the system. When this process reaches the cell phone that is paired to the system the call will be successfully sent through this cell phone. This is only made possible by the other feature on the Asterisk system chan-mobile. The next line hangs up the call if no one answers the call or if no cell phone was paired. The next lines are used to identify nine speed-dial numbers. These speed-dial numbers are specified in the globals context and the variables SpeedDial 1 through SpeedDial9 is used in the outgoing_cell context.

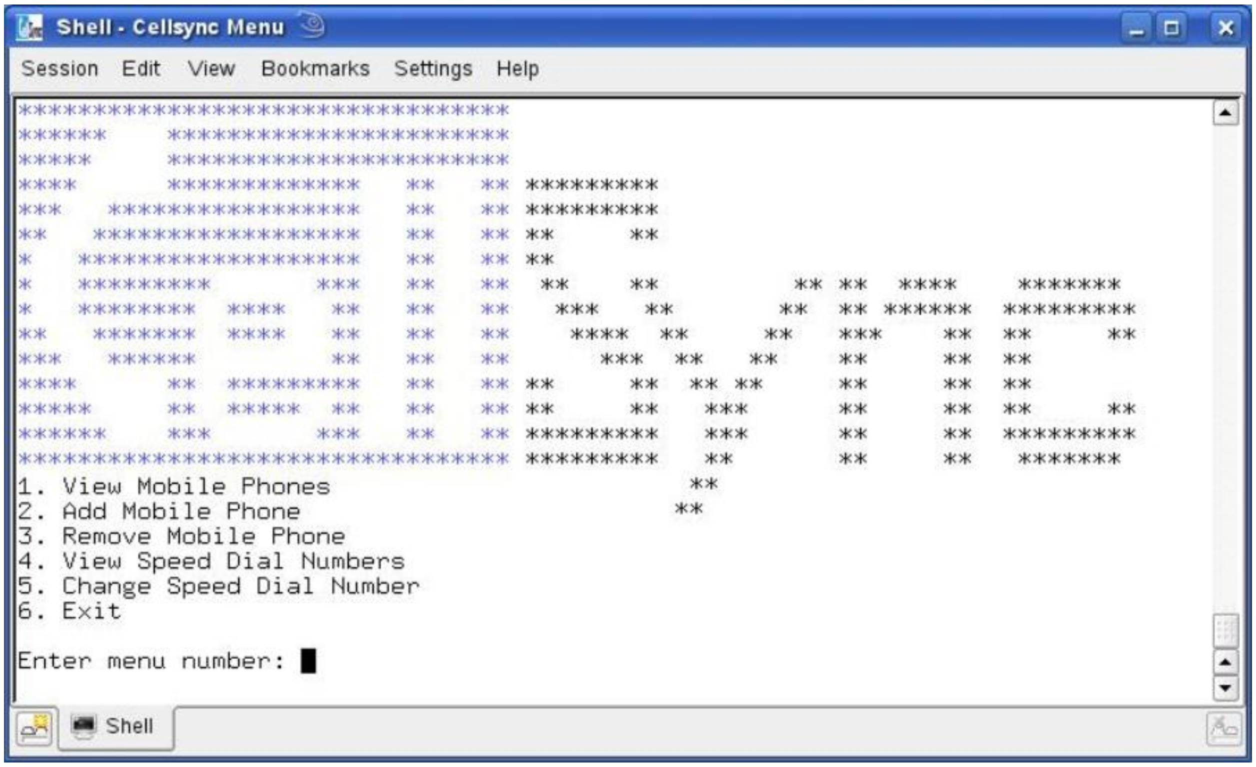
Many of the other contexts, like conferencing and directory, were not used but show that the Dialplan can incorporate other features into the system.

8.4 CellSync Menu

CellSync's prototype contains a text based menu user interface (TUI). There are many features of the CellSync device that can be modified by adding or changing programming code in the Asterisk files. However, typical end users will not understand how to write this programming code, and even if they did, it would be unnecessarily time consuming and inefficient. CellSync developed the user interface menu from scratch in order to allow the user to change the features of the system without knowing how to do the Asterisk programming. The CellSync menu uses simple instructions to prompt the user for necessary input to modify the files in order to satisfy the needs of the user. The CellSync menu program automatically generates programming code and writes it to the Asterisk files based on the user input from the keyboard.

The Linux operating system makes modification of files from the command line convenient for the programmer. The CellSync Menu program is written in the Bourne Again Shell script programming language. This is a language that understands Linux command line commands and also has many additional features such as variable storage, arithmetic, testing expressions, and looping. See Appendix D for the complete CellSync menu source code (comments are designated by # at the start of the line).

Figure 8.3 is a screen shot of the opening page of the CellSync Menu. See Appendix F for more screen shots of the CellSync Menu. The user has the option to view the mobile phones that are registered, add a mobile phone to the system, remove a mobile phone, view the speed dial numbers, change speed dial numbers, and exit the CellSync menu.



```
*********************************
******    ***********************
*****     ***********************
****      *************   **   ** *********
***   *****************   **   ** *********
**   ******************   **   ** **     **
*   *******************   **   ** **
*   *********       ***   **   **  **    **         ** **  ****    *******
*   ********  ****   **   **   **   ***   **       **  ** ******  *********
**   *******  ****   **   **   **    ****  **     **   ***    **  **     **
***   ******         **   **   **      ***  **   **    **     **  **
****      **  *********   **   ** **     **  ** **     **     **  **
*****     **  *****  **   **   ** **     **   ***      **     **  **     **
******    ***       ***   **   ** *********   ***      **     **  *********
********************************* *********   **       **     **   *******
1. View Mobile Phones                        **
2. Add Mobile Phone                         **
3. Remove Mobile Phone
4. View Speed Dial Numbers
5. Change Speed Dial Number
6. Exit

Enter menu number: █
```

Figure 8.3: CellSync Main Menu

A screen shot from many of these options can be seen in Appendix F. When the user selects View Mobile Phones, the CellSync menu program prints the MobileUsers file to the screen. This file contains all of the registered phones.

When the user selects Add Mobile Phone, the program prompts the user for a phone name and a phone address. The user can also type in “scan” instead of the address, and the program will automatically search for the Bluetooth address of all phones in range of the adapter. The names of the phones found as well as their addresses will be printed to the screen. The CellSync menu program will then generate code in the extensions.conf, mobile.conf, and MobileUsers files according to the input received from the keyboard. In Appendix E, the Bluetooth connectivity files are shown for what files were edited so phones could connect.

When Remove Mobile Phone is selected, the user is prompted to enter the name of the phone to be removed. The program then performs a search for this phone and deletes every occurrence of it in the MobileUsers, mobile.conf, and extensions.conf files.

Entering the menu number for View Speed Dial Numbers displays all speed dial entries and the phone numbers associated with them. These numbers are stored in the folder /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers.

When the Change Speed Dial Number menu option is selected, the user is prompted for the two digit speed dial number and then for a ten digit phone number to associate with it. The CellSync menu program uses a search algorithm to find that speed dial number in the extensions.conf and SpeedDialNumbers files and changes it to the new number.

Entering “6” on the main menu page exits the CellSync menu program. It should also be noted that the CellSync menu program was carefully tested and is designed to handle various forms of user typos and unusual input sequences.

8.5 System Operation

With the individual hardware and software components fully installed and configured, the system operates accordingly:

Before calling the user must:

1. Attach the incoming phone line from their phone company to the FXO port on the system (optional for sending/receiving landline calls)
2. Connect their cell phone to the system by entering the pin 1234 on their cell phone when prompted and on their computer when prompted. This process is no different than connecting a cell phone to a Bluetooth headset.
3. Connect their home phone network to the FXS port on the system. This is done by simply connecting a phone cord to the FXS port on the system to a phone jack in the user’s home. Alternatively, the system can be connected directly to a single landline phone, or to a base station for multiple wireless phones. This allows the system to be used without an existing landline network.

For an incoming call:

1. An incoming call will be acknowledged by Asterisk
2. Asterisk will route the call information to the Dialplan
3. The Dialplan will recognize the FXO port the call originated from, either cell phone or landline, and activate the appropriate context
4. For a cell phone call, the Dialplan will activate the incoming_mobile context
5. For a landline call the Dialplan will activate the incoming context
6. The incoming and incoming_mobile context will send a ringing signal to the landline phone
7. If the landline phone is answered the call will resume as any normal phone call
8. If the call goes unanswered the call will be directed to voicemail
9. For a cell phone call, the call will be routed to the cell phone voicemail
10. For a landline call, the call will be routed to the system voicemail

If the call originates from the CellSync user:

1. To initiate a cell phone call the user must dial 5, then the ten digit phone number of the person the wish to reach
2. To initiate a landline call the user must dial 9, then the ten digit phone number of the person they wish to reach

3. For cell phone calls, the digit 9 activates the CellCall context
4. The call is then routed through the cell phone via Bluetooth and the call resumes as any normal cell phone call.
5. For landline calls, the digit 6 activates the outgoing context
6. The call is then routed through the FXO port on the TNIC and the call will resume as any normal landline phone call

This system overview can be seen graphically in Figure 8.4.

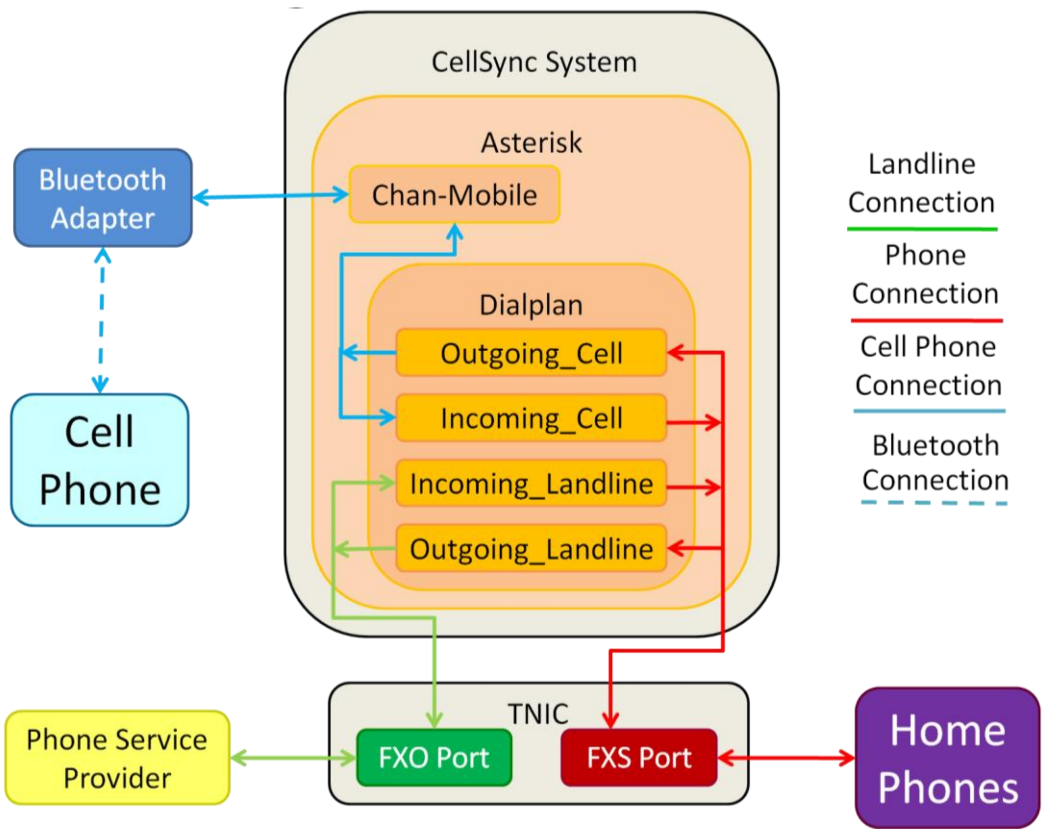


Figure 8.4: CellSync Method Prototype System Overview

9. Testing

9.1 General Testing

The device, being a telecommunications device marketed in the United States as a Bluetooth enabled device would be required to undergo multiple series of testing. The device must be tested for operability, safety, performance and security as defined by several organizations including the design criteria like the Federal Communications Commission's (FCC) compliance standards, Bluetooth SIG's system specifications, Underwriters Laboratories Inc.'s (UL) safety standards, and others. Some of these tests must be performed during product development and others when the final product is completed.

The FCC sets the operating standards for telecommunication devices. In order to bring a product to market in the United States the product must be registered with the FCC. Several companies are authorized to certify devices. One such company named Inertek says this on their company website:

> “The FCC requires that all customer premise telecom equipment (FCC Part 15) and network attached telecom equipment (FCC Part 68) to be sold in the U.S. must meet minimum compliance standards. FCC Part 15 requires emissions testing to prevent harmful radio interference, while FCC Part 68 testing is designed to prevent basic harms-to-the-network such as hazardous voltages” [10].

Inertek is accredited by the National Institute of Standards and Technology (NIST) and designated by the FCC as a Telecommunications Certification Body (TCB). The completed device would be required to be tested by a company such as Inertek.

The Bluetooth SIG has standards the device must meet or exceed in order to be marketed with the Bluetooth name or logo. This is what the Bluetooth website has to say about product qualification standards:

Bluetooth qualification is the certification process required for any product using Bluetooth technology [2]. Bluetooth qualification requires certain testing standards for all products that use the Bluetooth wireless technology. Qualification is a necessary pre-condition of the intellectual property license for the Bluetooth technology. Qualification is also necessary in order to apply the Bluetooth trademark to a product [2].

In addition to registering as a Bluetooth Adoptee Member the team would need to send the completed device to an outside company to assure the product meets these requirements. Elliott Laboratories is one such company that does Bluetooth qualification testing. As stated on their company profile: “Elliot Laboratories provides the full range of regulatory testing services required for Bluetooth devices and our authorized Bluetooth Qualification Test Facility partners provide all of the protocols and interoperability testing services required for Bluetooth certification” [2].

As well, the device should be sent to Underwriters Laboratories (UL) during and following the development of the device for operation safety testing. The UL product certification program is accredited by the U.S. Occupational Safety & Health Administration (OSHA), the American National Standards Institute (ANSI), and the Standards Council of Canada (SCC). UL is an independent, not-for-profit, nongovernmental organization. UL has this to say about pertaining to their product testing:

> “To establish certification, samples of a product submitted by manufacturers for certification are tested and evaluated. If UL decides the product fulfills all applicable requirements it authorizes the manufacturer to apply a certification mark to production of the samples submitted, or issues a certificate or notification that the product is now certified by UL” [2].

If this device is brought to market, several other tests can be done. These can include tests like the European equivalent of FCC and UL testing such as CE, European Conformity, and Restriction of Hazardous Substances (ROHS) testing. Finally, when Team CellSync is properly satisfied that the CellSync device sufficiently meets these criteria as well as the criteria defined by the team, the product will be tested for rigidity and general wear and tear to assure the device will continue to uphold and operate under wear and tear of normal everyday use.

9.2 CellSync Testing

The team tested the device continually throughout the development of the product. Tests included both hardware and software testing. CellSync tested for the robustness of the program, absence of bugs, usage of resources (processor and memory), absence of delays, absence of echoes, and quality of call.

To ensure that the TNIC was properly powering the phones a dial tone test was performed. At first this test was not successful but after properly configuring the TNIC a dial tone was achieved. From there, the Dialplan was tested and retested to achieve the desired functionality. A “Weasels” test was performed to ensure that the TNIC was communicating properly with Asterisk. When an incoming caller heard the pre-recorded message, it was confirmed that the TNIC was communicating with Asterisk and accessing the Dialplan. The Dialplan was then modified to send and receive landline calls. Several times throughout the development of the Dialplan testing was done. Each test provided feedback that allowed for debugging of the Dialplan. Appendix G shows a printout of a log sheet from a day of testing. The log sheet not only read out errors that occurred but also showed what processes where being performed and where the system was at when the errors occurred. When this functionality was achieved similar tests were done for cell phone calls. After the Dialplan had been debugged and fully developed it was locked to allow for the development of the CellSync menu. This menu was not only tested to ensure that it worked properly but to ensure that it accounted for typos and unusual entries. This testing showed that when a phone name was entered that was embedded in another phone name, removing one would delete both. The menu was edited to correct this and other bugs. Once the CellSync method was fully developed, the system was mirrored onto the prototype. After this process the system was tested to ensure it was fully functional and nothing was lost. During the mirroring process there were a few difficulties encountered but those were easily dealt with. Fortunately, the mirroring process worked flawlessly and the system performed correctly and completely.

The team also performed tests on system performance. Tests were done to ensure that the system did not increase latency beyond a reasonable amount. The tests concluded that the system did introduce additional latency into the calls but that this latency did not impede normal phone conversation. As well, the team ensured that call clarity was not distorted. In fact, the call clarity was found to be improved in most cases. The team was very happy with the results of the tests and was confident in the systems performance as a result of its vigorous testing procedures.

10. Cost Estimates and Business Plan

10.1 Prototype Costs

The prototype was composed of several components. The building block for the system was a Dell 4300S computer donated by Glen Van Andel, a former Calvin professor. Required for the system was a telephone network interface card purchased from ZapMicro and a Bluetooth USB adapter purchased from Amazon.com. To create a small form factor, a custom enclosure was constructed of polycarbonate and excess sheet metal. The total cost of the prototype was $165.00 thanks in large part to the donation of the PC by Glen Van Andel. The prototype costs are summarized in Table 10.1.

Table 10.1: Initial prototype cost breakdown of PBX/Asterisk alternative

Item	Unit Cost
ZMA400P11 TNIC	$126.70
Bluetooth Adapter	$30.00
Asterisk Program	$0.00
Barebones Computer	$0.00
Polycarbonate Enclosure	$60.88
Total Prototype Cost	**$217.58**

10.2 Total Production Cost

Significant research and calculations were done to determine whether this product would succeed in the market place and what profit the device might accumulate. There are two feasible methods for developing this device and bringing it to market; the “From Scratch” (FS) method and the “Modules” (M) method.

10.2.1 The CellSync Method

To manufacture and distribute the existing CellSync method would incur some costs. These would include the cost of a TNIC, a Bluetooth adapter, the CellSync developed Asterisk based software, a download CD, and packaging. A list of these costs can be found in Table 10.2. The total per unit cost of production is $56.10. This cost does not include labor. It can be assumed that this cost will drop over time as the team develops ways to produce the device more cost efficiently.

Table 10.2: Production per Unit Cost of CellSync Method

Item	Unit Cost	Quantity	Total Cost
Telephone Interface Card	$50.00	1	$50.00
Bluetooth Adapter	$5.00	1	$5.00
Asterisk Program	Free	1	Free
Download CD	$0.10	1	$0.10
Packaging	$1.00	1	$1.00
Total Material Cost per Unit			**$56.10**

10.2.2 Stand-alone Device

If brought to the market, CellSync would develop a proprietary stand-alone device. This device would have all the essential hardware and software components without any unnecessary components. This would save the user money over a dedicated PC system as there would be no need to have a dedicated machine to use for the CellSync method as well as consuming less power than the PC consumes while in operation. There would be several more individual components needed to create such a device.

Several of these components come bundled in different modules. While this approach would minimize labor and construction costs it would increase the cost of the components. A price break down of these components can be found in Table 10.3. The total cost per device would be $144.30. After production and labor costs are added and the product is marked up for retail sale, the cost to consumer would be around $300.00. While this cost remains competitive with some competing products, this price exceeds the desired price range of $100.00 to $150.00.

Table 10.3: Production per Unit Cost of Modules Method for 100 Part Orders

Item	Unit Cost
FSX Analog Module	$46.90
FXO Analog Module	$43.40
Bluetooth Module	$25.00
Microcontroller/Processor	$5.00
Chassis	$5.00
Passive components	$10.00
Power Source	$8.00
Software	$0.00
Packaging	$1.00
Total Material Cost per Unit	**$144.30**

Table 10.4 outlines the cost of building the device essentially from scratch, building each module from its separate parts. This would increase labor and production costs significantly over the previous approach but would drastically reduce the cost per device of the components. The final cost per device would be $79.50. After labor and production costs, the device could be marketed around $150.00.

Table 10.4: Production per Unit Cost of From Scratch for 100 Part Orders

Item	Unit Cost
FSX Analog Interface	$20.00
FXO Analog Interface	$20.00
Bluetooth Processor	$5.50
Microcontroller/Processor	$5.00
Chassis	$5.00
Passive components	$10.00
Power Source	$8.00
Software	$0.00
Memory	$5.00
Packaging	$1.00
Total Material Cost per Unit	**$79.50**

10.3 Total Labor and Production Costs

Overhead costs for production of the CellSync product would be low but production costs would be high. Because none of the components are made on site there is no need for significant machinery. However, because there are several components involved in the manufacturing of the device, several line laborers would be needed. Working at $15.00/hour for 40 hours a week 50 weeks throughout the year, it is estimated that this will require 8 full time employees for the "From Scratch" method and 4 for the "Modules" method. The number of line laborers needed would drop as demand decreases, and therefore production, of the device drops. Production line equipment and building expenses are estimated at a $10,000/year fixed cost for the lifetime of the production of the device. According to the National Federation of Independent Business, it is common for a new company to spend 10 to 15% of its predicted yearly revenue on advertisement [21]. Advertisement costs start at $150,000 for the FS method and $90,000 for the M method for the first year and drop as expected revenue drops. Four marketing employees for the FS method and two for the M method will be employed for the first year. Marketing employees earn a salary wage of $40,000. It is also expected that a team of eight engineers will be needed for the FS method and four for the M method in the first year. Engineering salary has been set at $50,000. The number of engineers and marketers needed is expected to drop over the lifetime of the production of the device. From this data, the total annual labor and production costs can be calculated. Table 10.5 lists these costs for bench mark years 1, 4, and 10 for the device.

Table 10.5: Total Annual Labor and Production Costs

Operating Costs Analysis	Operating Costs: Year 1			Operating Costs: Year 4			Operating Costs: Year 10		
	From Scratch Method			From Scratch Method			From Scratch Method		
Item	Unit Cost	Quantity	Total Cost	Unit Cost	Quantity	Total Cost	Unit Cost	Quantity	Total Cost
Manufacturing Line Labor	$30,000	8	$240,000	$30,000	8	$240,000	$30,000	3.2	$96,000
Manufacturing Line Equipment, Building	$10,000	1	$10,000	$10,000	1	$10,000	$10,000	1	$10,000
Advertising Costs	$150,000	1	$150,000	$80,000	1	$80,000	$80,000	1	$80,000
Marketing Labor	$40,000	4	$160,000	$40,000	2.8	$112,000	$40,000	0.4	$16,000
Engineering Labor	$50,000	8	$400,000	$50,000	3.8	$190,000	$50,000	2	$100,000
Total Annual Production Cost (TAPC)			**$960,000**			**$632,000**			**$302,000**
	Modules Method			Modules Method			Modules Method		
Item	Unit Cost	Quantity	Total Cost	Unit Cost	Quantity	Total Cost	Unit Cost	Quantity	Total Cost
Manufacturing Line Labor	$30,000	4	$120,000	$30,000	4	$120,000	$30,000	1.6	$48,000
Manufacturing Line Equipment, Building	$10,000	1	$10,000	$10,000	1	$10,000	$10,000	1	$10,000
Advertising Costs	$100,000	1	$100,000	$80,000	1	$80,000	$80,000	1	$80,000
Marketing Labor	$40,000	4	$160,000	$40,000	2.8	$112,000	$40,000	0.4	$16,000
Engineering Labor	$50,000	4	$200,000	$50,000	1.9	$95,000	$50,000	0.5	$25,000
Total Annual Production Cost (TAPC)			$590,000			**$417,000**			**$179,000**

10.4 Break-Even Analysis

The selling price of the product will be set to $150.00 for the FS method and $300.00 for the M method. These prices are competitive with similar products. With annual savings of $33.00 each month, the consumer will be able to reach payback in only five and nine months respectively. The device starts generating revenue at years 2 and 3 and breaks even between years 3 and 4 for both methods. The expected breakeven number is 38,007 devices sold and occurs in the fourth year of production. The break even number can be found using the following formula:

(Cost of Device – Cost of Materials)(Break Even Number of units sold) – Production Costs = 0

The annual break even numbers and the total breakeven point can be found in Table 10.6.

Table 10.6: Break-Even Analysis

Yearly Break Even Analysis				
Method	From Scratch		Modules	
TAPC	$960,000		$590,000	
PPD	$150.00		$300.00	
Year	Cost	BE #	Cost	BE #
Year 1	$79.50	13,617	$144.30	3,725
Year 2	$75.53	9,572	$137.09	2,657
Year 3	$71.75	7,906	$130.23	2,191
Year 4	$68.16	6,913	$123.72	1,905
Year 5	$64.75	5,766	$117.53	1,587
Year 6	$61.52	4,824	$111.66	1,325
Year 7	$58.44	4,017	$106.07	1,100
Year 8	$55.52	3,309	$100.77	901
Year 9	$52.74	2,674	$95.73	724
Year 10	$50.10	2,096	$90.94	562

Complete Break Even Analysis				
Year	BE #	Units Sold	BE Difference	BE Count
Year 1	13,617	10,000	3,617	3,617
Year 2	9,572	10,000	(428)	3,189
Year 3	7,906	10,000	(2,094)	1,095
Year 4	6,913	10,000	(3,087)	(1,993)

Total Units Sold - Excess BE Count = BE Number

40,000 - 1,993 = 38007

10.5 Total Revenue

This product is marketable to all cell phone owners. According to Earth Trends, 213,212,000 Americans have at least one mobile phone. It is estimated that CellSync can sell 10,000 units for the FS method and 300 units in for the Modules method for the first four years. This generates the revenue seen in Table 10.7 for the benchmark years of year 1 and year 4.

Table 10.7: Total Annual Profit

Total Revenue: Year 1			
From Scratch			
Item	Value per Unit	# of Units	Total Value
Material Cost	($79.50)	10,000	($795,000.00)
TAPC	($960,000)	1	($960,000)
Product Revenue	$150.00	10,000	$1,500,000
Total Annual Profit			($255,000.00)
Modules			
Item	Value per Unit	# of Units	Total Value
Material Cost	($144.30)	3,000	($432,900.00)
TAPC	($590,000)	1	($590,000)
Product Revenue	$300.00	3,000	$900,000
Total Annual Profit			($122,900.00)

Total Revenue: Year 4			
From Scratch			
Item	Value per Unit	# of Units	Total Value
Material Cost	($68.16)	10,000	($681,613.13)
TAPC	($632,000)	1	($632,000)
Product Revenue	$150.00	10,000	$1,500,000
Total Annual Profit			$186,386.88
Modules			
Item	Value per Unit	# of Units	Total Value
Material Cost	($123.72)	3,000	($371,157.64)
TAPC	($417,000)	1	($417,000)
Product Revenue	$300.00	3,000	$900,000
Total Annual Profit			$111,842.36

10.6 Ten Year Outlook

The lifetime of the product is estimated at ten years. Throughout this time, production and material costs will decrease as research finds ways to make production more cost efficient and as production methods become established. The number of units sold will start to drop after four years, until it reaches 4,000 and 1200 at ten years, at which time it is no longer justifiable to continue production. The total profit for ten years of production is predicted to be just under $1.7 million for the FS method and $870,000 for the M method. Table 10.8 summarizes these results.

Table 10.8: Ten Year Outlook Annual Profit

Total Revenue: 10 Year Period						
From Scratch						
Year	Material Cost	TAPC	# Units Sold	Product Revenue	Total Annual Profit	Total Profit
1	($79.50)	($960,000.00)	10,000	$1,500,000.00	($255,000.00)	($255,000.00)
2	($75.53)	($712,865.99)	10,000	$1,500,000.00	$31,884.01	($223,115.99)
3	($71.75)	($618,627.70)	10,000	$1,500,000.00	$163,884.80	($59,231.19)
4	($68.16)	($565,731.98)	10,000	$1,500,000.00	$252,654.89	$193,423.70
5	($64.75)	($491,558.22)	9,000	$1,350,000.00	$275,662.56	$469,086.26
6	($61.52)	($426,827.02)	8,000	$1,200,000.00	$281,048.30	$750,134.56
7	($58.44)	($367,820.97)	7,000	$1,050,000.00	$273,100.40	$1,023,234.96
8	($55.52)	($312,597.98)	6,000	$900,000.00	$254,295.13	$1,277,530.09
9	($52.74)	($260,033.17)	5,000	$750,000.00	$226,257.21	$1,503,787.30
10	($50.10)	($209,424.21)	4,000	$600,000.00	$190,156.47	$1,693,943.77
Total	($5,230,568.98)	($4,925,487.25)	79,000	$11,850,000.00	**$1,693,943.77**	

Total Revenue: 10 Year Period						
Modules						
Year	Material Cost	TAPC	# Units Sold	Product Revenue	Total Annual Profit	Total Profit
1	($144.30)	($640,000.00)	3,000	$900,000.00	($172,900.00)	($172,900.00)
2	($137.09)	($492,865.99)	3,000	$900,000.00	($4,120.99)	($177,020.99)
3	($130.23)	($431,961.03)	3,000	$900,000.00	$77,346.72	($99,674.27)
4	($123.72)	($395,731.98)	3,000	$900,000.00	$133,110.38	$33,436.11
5	($117.53)	($343,558.22)	2,700	$810,000.00	$149,102.00	$182,538.10
6	($111.66)	($297,493.69)	2,400	$720,000.00	$154,530.50	$337,068.60
7	($106.07)	($255,249.54)	2,100	$630,000.00	$151,995.57	$489,064.17
8	($100.77)	($215,597.98)	1,800	$540,000.00	$143,015.90	$632,080.06
9	($95.73)	($177,810.95)	1,500	$450,000.00	$128,591.70	$760,671.76
10	($90.94)	($141,424.21)	1,200	$360,000.00	$109,441.80	$870,113.56
Total	($2,848,192.85)	($3,391,693.60)	23,700	$7,110,000.00	**$870,113.56**	

11. Work Structure

11.1 Team organization

In CellSync everybody brings in their unique views, ideas, strengths, intuition, and experiences. There is no specific leader in CellSync, but over time Matt, Keith, and Phil have become the overall leads of the team. Andrew and Jeff specialized on tasks that required a more focused part of the project, rather than broad scale aspects. Tasks were delegated in order to use each person's skills to the best of their ability.

11.2 Task Specification

Although no team leader has been appointed some tasks need to be completed by specific people to keep the order of the team.

11.2.1 Meeting Agendas

Matt is responsible for the agendas for the meetings. He makes sure that the meetings stay focused so we do not waste time in the meetings.

11.2.2 Meeting Minutes

Phil and Keith are responsible for keeping track of the meeting by recording what has been done in the meeting. Also, these brief notes get typed and put in a Google Doc to share with the group so as to keep them informed of progress.

11.2.3 Scheduling

We schedule times for meetings that work best for everyone or just a core group that needs to work together. Phil is mainly responsible for sending reminders of the meetings out via email or a Google Calendar event notification.

11.2.4 Website Design and maintenance

Phil and his brother William have designed the website from the ground up and are responsible for maintaining the website. Phil has tried to regularly update the website to reflect the most recent changes in the project and accomplishments.

11.2.5 Finances

Andrew and Keith keep track of the finances. They make sure we spend our money wisely and keep record of the money spent. They are therefore responsible for creating an accurate market strategy for product development.

11.2.6 Documentation

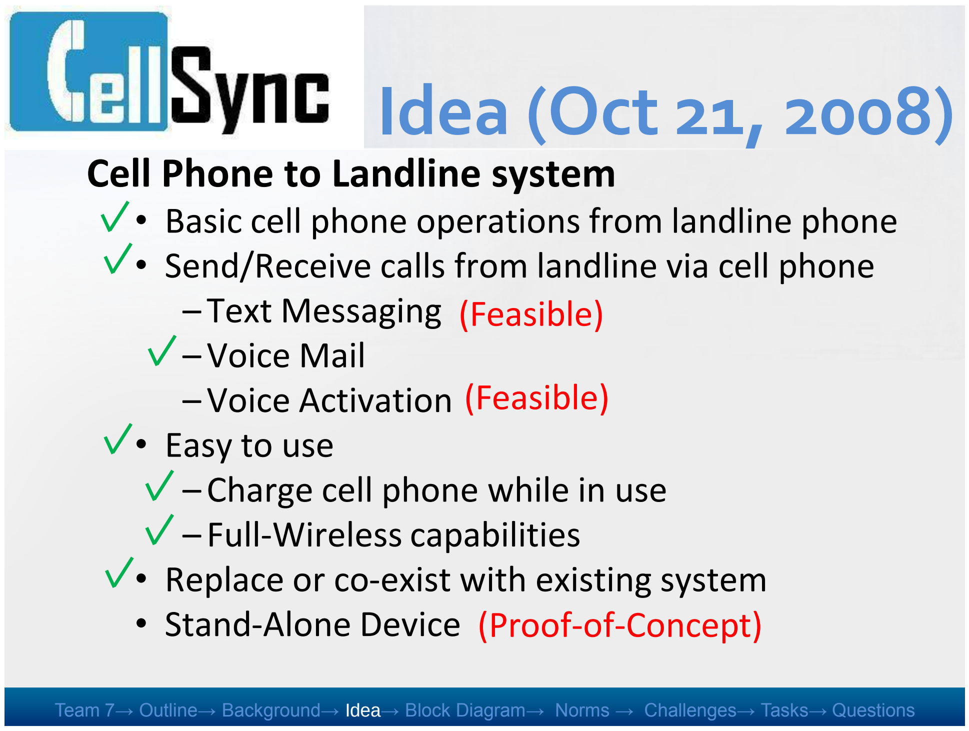
Each member is responsible for documenting research and tasks completed. Keith and Phil update the Gantt Charts to reflect the most recent progress in the project. Appendix A has the Gantt chart.

11.2.7 Presentations

Each member of the team is required to give at least one presentation during the time of the project. Keith creates an initial draft presentation that the team then decides on what else should be included or excluded.

12. Conclusion

The team has deemed this project highly successful. In investigating the level of success the team reached, the team looked back at its original objectives from the beginning of the year before any design decisions had been made. At this point in the project development an approach had not yet been decided on and in fact the final approach had not yet been discovered by the team. The slide shown in Figure 12.1 shows the "Idea" defined at the beginning of the project development.



CellSync

Idea (Oct 21, 2008)

Cell Phone to Landline system

- ✓ Basic cell phone operations from landline phone
- ✓ Send/Receive calls from landline via cell phone
  - Text Messaging (Feasible)
  - ✓ Voice Mail
  - Voice Activation (Feasible)
- ✓ Easy to use
  - ✓ Charge cell phone while in use
  - ✓ Full-Wireless capabilities
- ✓ Replace or co-exist with existing system
- Stand-Alone Device (Proof-of-Concept)

Team 7→ Outline→ Background→ Idea→ Block Diagram→ Norms → Challenges→ Tasks→ Questions

Figure 12.1: "Idea" modified from "Project Introduction" presentation Oct 21, 2008

As can be seen, a high level of success was achieved. The final design and prototype performed the basic functionality the team set out to accomplish. This would have been deemed successful; however several additional, secondary objectives were met as well. The prototype was easy to use and allowed for fully wireless home phone systems using a wireless base station. As well, the prototype allowed the user to charge the cell phone while using the device. This was important as Bluetooth drains the cell phone battery at a faster rate than normal use. Additionally, the prototype had a functional voicemail and worked with or without landline phone service. The final design lacked a few design points desired but allowed for the addition of text messaging and voice activation through reasonably feasible changes to the system software. The final design was not a stand-alone device but did stand as a good proof-of-concept that a stand-alone device matching the design criteria laid out in section 3.1.1 was possible.

Additionally, the team was successful from a financial and educational viewpoint. The final cost of the prototype was $232.65, under the $300.00 budget allotted for the course and the marketable product would be financially viable in the marketplace. The Cost Analysis section shows that this product would be a profitable project but would most likely be the most profitable as an additional product for an existing company.

The project ultimately taught the team about the design process and teamwork. The team took this project from conception, through research, design, build, test, rebuilt, to the final prototype. While going through the design process the team not only absorbed new and interesting information but learned valuable lessons. These lessons included using the resources that are available to you; relying on each other to complete their designated tasks; and scheduling. The group feels they have learned valuable information that will help them be more successful when entering the engineering industry.

The design process of this project strained both our time and knowledge. The project drew upon all our previous knowledge while providing avenues to gain new knowledge. The team believes that the chosen project was a success from a functional, financial, and education, perspective as well as providing all the intangibles that are associated with projects of this nature.

13. Acknowledgements

Team CellSync would like to thank the following people who made our project possible and provided us with help throughout the year.

Calvin Engineering faculty and staff:

- Bob DeKraker – Providing computer support for our team and the engineering department
- Chuck Holwerda – Providing technical support for our team and the engineering department
- Philip Jasperse – Providing technical support for our team and the engineering department
- Gary Draving – Providing Linux and computer support for our team and the CS department
- David Miller – Providing telephone line access for our team
- Michelle Krul – Administration of the senior design course including website activation and document publication
- Stanley Abdalla & Sharlita Field – Managing the Engineering Building facilities
- Professor Emeritus Glen Van Andel -- Providing the computer for the barebones system
- Prof. Robert Bossemeyer – Senior Design advisor and mentor for our team
- Prof. David Wunder – Heading the Senior Design course

14. Works Cited

[1] "About." BlueZ: Official Linux Bluetooth protocol stack. Bluetooth Special Interest Group. 16 Nov. 2008 <http://www.bluez.org/about/>.

[2] "Bluetooth Product Testing." Elliot: A NTS Company. 2007. ELLIOTT LABORATORIES. 1 Dec. 2008 <http://www.elliottlabs.com/wireless_testing_bluetooth.htm>.

[3] "Bluetooth SMD Module - Bluegiga." Sparkfun.com. SparkFun Electronics. <http://http://www.sparkfun.com/commerce/product_info.php?products_id=8771>.

[4] Brouwer, Randall. "Lab 2 – Memory and DMA Transfers." Engineering 325 (2008)

[5] Brouwer, Randall. "NIOS II: New Components and External I/O." Engineering 325 (2008)

[6] "Build with Bluetooth Technology." Building With Bluetooth Technology. Bluetooth. 16 Nov. 2008 <http://www.bluetooth.com/bluetooth/technology/building/>.

[7] Call My Cell: Wireless Substitution in the United States. Rep.No. Nielsen Mobile, The Nielsen Company. New York, NY: The Nielsen Company, 2008.

[8] Digi-Key Catalog. Thief River Falls, MN: Digi-Key, 2008.4.

[9] Engdahl, Tomi. "Telephone line audio interface circuits." Epanorama.net. 1996. 16 Nov. 2008 <http://www.epanorama.net/circuits/teleinterface.html>.

[10] "FCC Testing & Certification." FCC Testing & Certification. 2004. Intertek. 1 Dec. 2008 <http://www.intertek-etlsemko.com/ >.

[11] Grace Digital Inc. ITC-BT Cell Bluetooth Gateway Owner's Manual. Brochure. San Diego, CA: Author.

[12] Graham, Lee. "U.S. Consumer Mobile Phone Unit-Sales Declined 13 Percent Year-over-Year in Q2 2008." The NPD Group. 19 Aug. 2008. The NPD Group, Inc. 1 Dec. 2008 <http://www.npd.com/press/releases/press_080819.html>.

[13] Huang, Albert, and Larry Rudolph. Bluetooth Essentials for Programmers. New York, NY: Cambridge UP, 2007.

[14] Huang, Albert. "PyBlueZ." Google Code Project Feed. 22 Jan. 07. PyBlueZ. 16 Nov. 2008 <http://code.google.com/p/pybluez/>.

[15] iWrap User's Manual. Bluetooth Modules, Bluegiga Technologies. ver 2.2.2 Hingham, MA: Bluegiga Technologies, 2007.

[16] Milanesi, Carolina, Stan Bruederle, Hugues J. De La Vergne, Ann Liang, Nahoko Mitsuyama, and Tuong Huy Nguyen. Market Focus: Bluetooth in Mobile Devices, Worldwide, 2004-2009. Rep.No. Gartner, Inc. Stamford, CT: Gartner, Inc, 2005.

[17] Nios II Processor Reference Handbook. Vol. 8.1. San Jose, CA: Altera Corp, 2008.

[18] Phone Labs Technology Company Inc. Dock-N-Talk User's Guide. Brochure. 1.4st ed. Bridgeport, CT: Author.

[19] "Qualification Program." Bluetooth. Bluetooth SIG. 1 Dec. 2008 <http://www.bluetooth.com/bluetooth/technology/building/qualification/>.

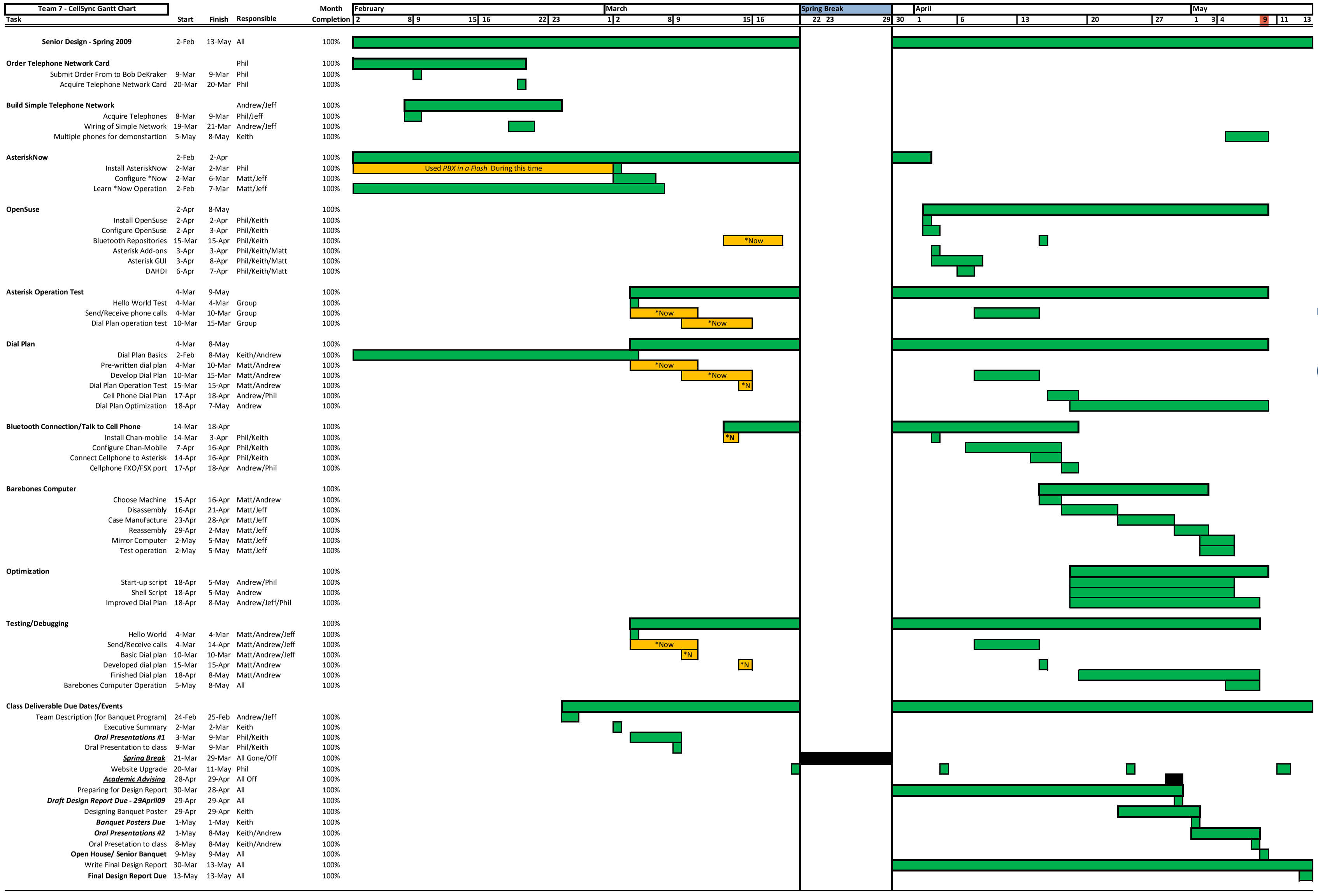
[20] Van Meggelen, Jim, Jared Smith, and Leif Madsen. Asterisk : The Future of Telephony. 2nd ed. Danbury, CA: O'Reilly Media, Incorporated, 2007.

[21] "What Percentage of Your Revenue Should You Spend on Advertising?" Tools Tips and Practical Management Advice. 20 May 2002. National Federation of Independent Business. 2 Dec. 2008 <http://www.nfib.com/object/3309832.html>.

[22] The World Telecommunication/ICT Indicators Database. Rep.No. Market Information and Statistics, Internation Telecommunications Union. 12th ed. Geneva: International Telecommunication Union, 2008. 2007. ICT Indicators Database. 16 Nov. 2008 <http://www.itu.int/itu-d/ict/statistics/>.

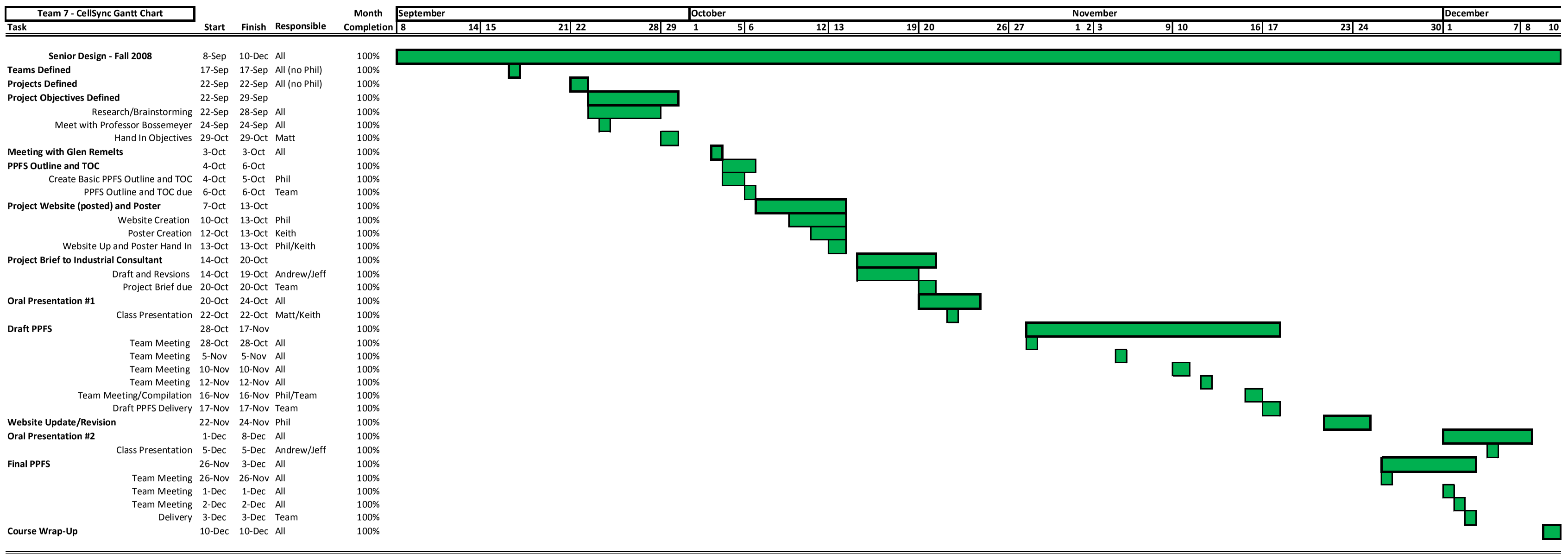
[23] WT32 Data Sheet. Bluetooth Modules, Bluegiga Technologies. ver. 1.1 Hingham, MA: Bluegiga Technologies, 2008.

Appendix A: Gantt Chart – Spring



Team 7 - CellSync Gantt Chart				Month
Task	Start	Finish	Responsible	Completion
Senior Design - Spring 2009	2-Feb	13-May	All	100%
Order Telephone Network Card			Phil	100%
Submit Order From to Bob DeKraker	9-Mar	9-Mar	Phil	100%
Acquire Telephone Network Card	20-Mar	20-Mar	Phil	100%
Build Simple Telephone Network			Andrew/Jeff	100%
Acquire Telephones	8-Mar	9-Mar	Phil/Jeff	100%
Wiring of Simple Network	19-Mar	21-Mar	Andrew/Jeff	100%
Multiple phones for demonstartion	5-May	8-May	Keith	100%
AsteriskNow	2-Feb	2-Apr		100%
Install AsteriskNow	2-Mar	2-Mar	Phil	100%
Configure *Now	2-Mar	6-Mar	Matt/Jeff	100%
Learn *Now Operation	2-Feb	7-Mar	Matt/Jeff	100%
OpenSuse	2-Apr	8-May		100%
Install OpenSuse	2-Apr	2-Apr	Phil/Keith	100%
Configure OpenSuse	2-Apr	3-Apr	Phil/Keith	100%
Bluetooth Repositories	15-Mar	15-Apr	Phil/Keith	100%
Asterisk Add-ons	3-Apr	3-Apr	Phil/Keith/Matt	100%
Asterisk GUI	3-Apr	8-Apr	Phil/Keith/Matt	100%
DAHDI	6-Apr	7-Apr	Phil/Keith/Matt	100%
Asterisk Operation Test	4-Mar	9-May		100%
Hello World Test	4-Mar	4-Mar	Group	100%
Send/Receive phone calls	4-Mar	10-Mar	Group	100%
Dial Plan operation test	10-Mar	15-Mar	Group	100%
Dial Plan	4-Mar	8-May		100%
Dial Plan Basics	2-Feb	8-May	Keith/Andrew	100%
Pre-written dial plan	4-Mar	10-Mar	Matt/Andrew	100%
Develop Dial Plan	10-Mar	15-Mar	Matt/Andrew	100%
Dial Plan Operation Test	15-Mar	15-Apr	Matt/Andrew	100%
Cell Phone Dial Plan	17-Apr	18-Apr	Andrew/Phil	100%
Dial Plan Optimization	18-Apr	7-May	Andrew	100%
Bluetooth Connection/Talk to Cell Phone	14-Mar	18-Apr		100%
Install Chan-moblie	14-Mar	3-Apr	Phil/Keith	100%
Configure Chan-Mobile	7-Apr	16-Apr	Phil/Keith	100%
Connect Cellphone to Asterisk	14-Apr	16-Apr	Phil/Keith	100%
Cellphone FXO/FSX port	17-Apr	18-Apr	Andrew/Phil	100%
Barebones Computer				100%
Choose Machine	15-Apr	16-Apr	Matt/Andrew	100%
Disassembly	16-Apr	21-Apr	Matt/Jeff	100%
Case Manufacture	23-Apr	28-Apr	Matt/Jeff	100%
Reassembly	29-Apr	2-May	Matt/Jeff	100%
Mirror Computer	2-May	5-May	Matt/Jeff	100%
Test operation	2-May	5-May	Matt/Jeff	100%
Optimization				100%
Start-up script	18-Apr	5-May	Andrew/Phil	100%
Shell Script	18-Apr	5-May	Andrew	100%
Improved Dial Plan	18-Apr	8-May	Andrew/Jeff/Phil	100%
Testing/Debugging				100%
Hello World	4-Mar	4-Mar	Matt/Andrew/Jeff	100%
Send/Receive calls	4-Mar	14-Apr	Matt/Andrew/Jeff	100%
Basic Dial plan	10-Mar	10-Mar	Matt/Andrew/Jeff	100%
Developed dial plan	15-Mar	15-Apr	Matt/Andrew	100%
Finished Dial plan	18-Apr	8-May	Matt/Andrew	100%
Barebones Computer Operation	5-May	8-May	All	100%
Class Deliverable Due Dates/Events				
Team Description (for Banquet Program)	24-Feb	25-Feb	Andrew/Jeff	100%
Executive Summary	2-Mar	2-Mar	Keith	100%
Oral Presentations #1	3-Mar	9-Mar	Phil/Keith	100%
Oral Presentation to class	9-Mar	9-Mar	Phil/Keith	100%
Spring Break	21-Mar	29-Mar	All Gone/Off	100%
Website Upgrade	20-Mar	11-May	Phil	100%
Academic Advising	28-Apr	29-Apr	All Off	100%
Preparing for Design Report	30-Mar	28-Apr	All	100%
Draft Design Report Due - 29April09	29-Apr	29-Apr	All	100%
Designing Banquet Poster	29-Apr	29-Apr	Keith	100%
Banquet Posters Due	1-May	1-May	Keith	100%
Oral Presentations #2	1-May	8-May	Keith/Andrew	100%
Oral Presetation to class	8-May	8-May	Keith/Andrew	100%
Open House/ Senior Banquet	9-May	9-May	All	100%
Write Final Design Report	30-Mar	13-May	All	100%
Final Design Report Due	13-May	13-May	All	100%

Appendix A: Gantt Chart – Fall

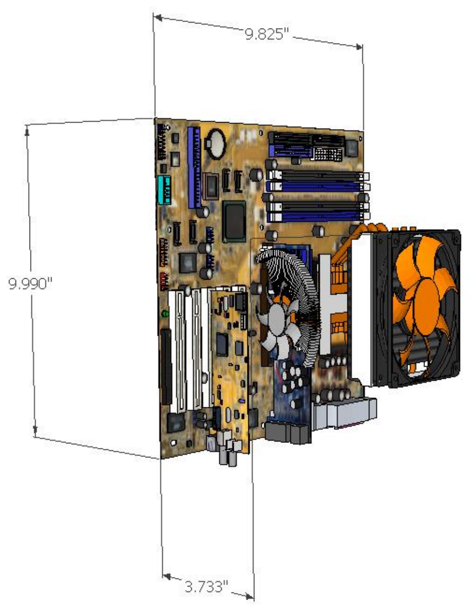


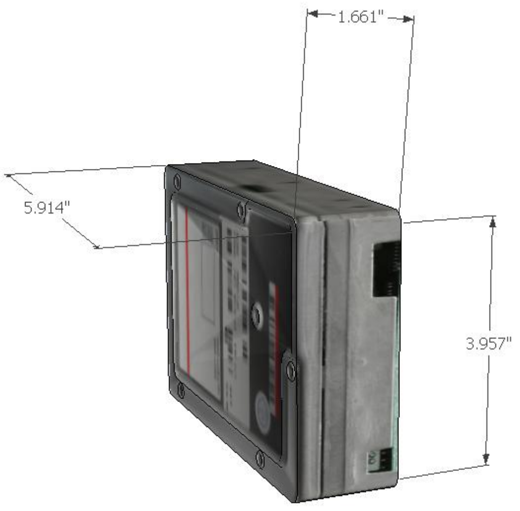
Team 7 - CellSync Gantt Chart

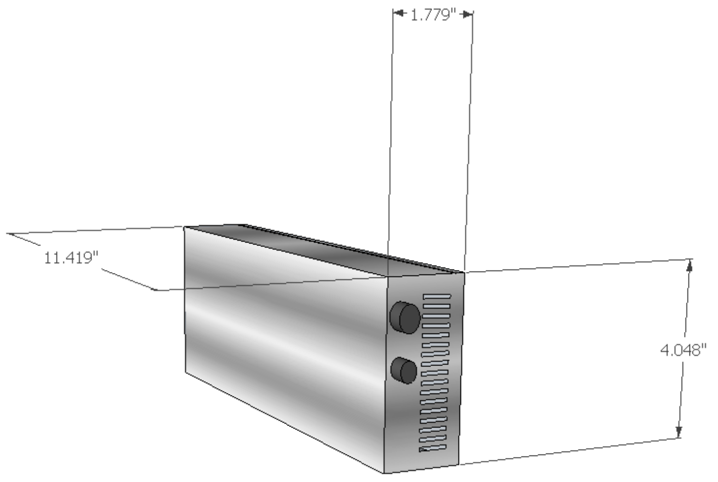
Task	Start	Finish	Responsible	Completion
Senior Design - Fall 2008	8-Sep	10-Dec	All	100%
Teams Defined	17-Sep	17-Sep	All (no Phil)	100%
Projects Defined	22-Sep	22-Sep	All (no Phil)	100%
Project Objectives Defined	22-Sep	29-Sep		100%
Research/Brainstorming	22-Sep	28-Sep	All	100%
Meet with Professor Bossemeyer	24-Sep	24-Sep	All	100%
Hand In Objectives	29-Oct	29-Oct	Matt	100%
Meeting with Glen Remelts	3-Oct	3-Oct	All	100%
PPFS Outline and TOC	4-Oct	6-Oct		100%
Create Basic PPFS Outline and TOC	4-Oct	5-Oct	Phil	100%
PPFS Outline and TOC due	6-Oct	6-Oct	Team	100%
Project Website (posted) and Poster	7-Oct	13-Oct		100%
Website Creation	10-Oct	13-Oct	Phil	100%
Poster Creation	12-Oct	13-Oct	Keith	100%
Website Up and Poster Hand In	13-Oct	13-Oct	Phil/Keith	100%
Project Brief to Industrial Consultant	14-Oct	20-Oct		100%
Draft and Revsions	14-Oct	19-Oct	Andrew/Jeff	100%
Project Brief due	20-Oct	20-Oct	Team	100%
Oral Presentation #1	20-Oct	24-Oct	All	100%
Class Presentation	22-Oct	22-Oct	Matt/Keith	100%
Draft PPFS	28-Oct	17-Nov		100%
Team Meeting	28-Oct	28-Oct	All	100%
Team Meeting	5-Nov	5-Nov	All	100%
Team Meeting	10-Nov	10-Nov	All	100%
Team Meeting	12-Nov	12-Nov	All	100%
Team Meeting/Compilation	16-Nov	16-Nov	Phil/Team	100%
Draft PPFS Delivery	17-Nov	17-Nov	Team	100%
Website Update/Revision	22-Nov	24-Nov	Phil	100%
Oral Presentation #2	1-Dec	8-Dec	All	100%
Class Presentation	5-Dec	5-Dec	Andrew/Jeff	100%
Final PPFS	26-Nov	3-Dec	All	100%
Team Meeting	26-Nov	26-Nov	All	100%
Team Meeting	1-Dec	1-Dec	All	100%
Team Meeting	2-Dec	2-Dec	All	100%
Delivery	3-Dec	3-Dec	Team	100%
Course Wrap-Up	10-Dec	10-Dec	All	100%

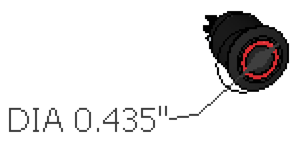
Appendix B: Prototype Design Sketches

1. Motherboard, Harddrive, Power Supply, and Power Button

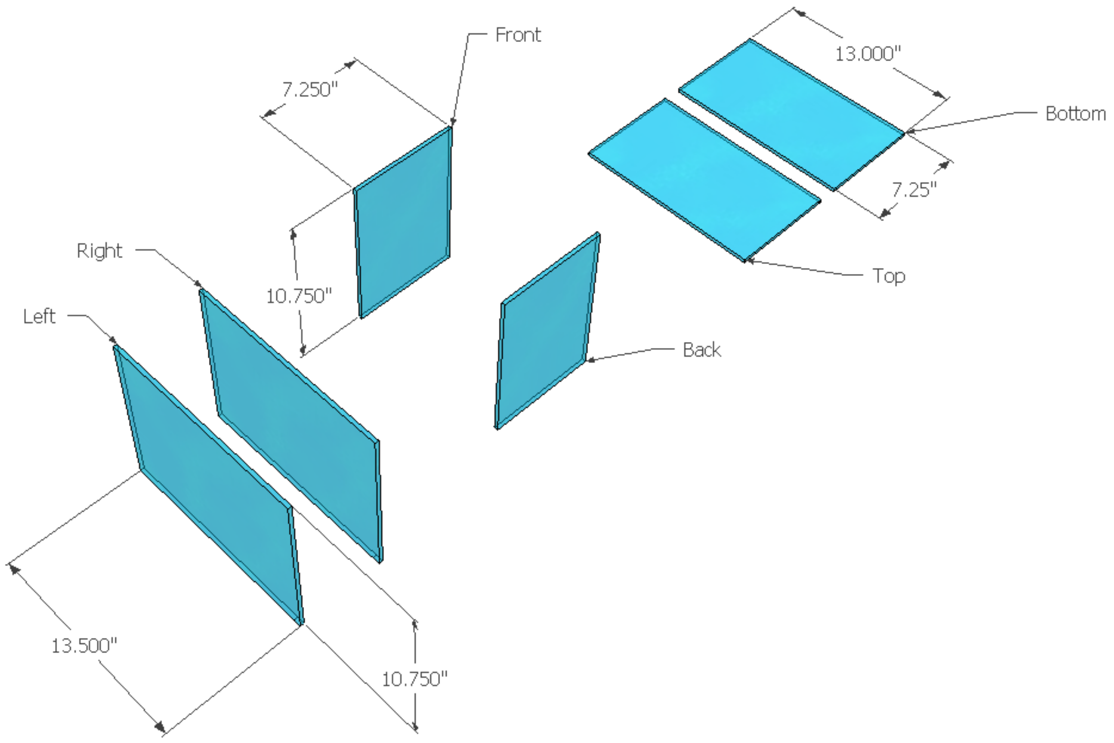


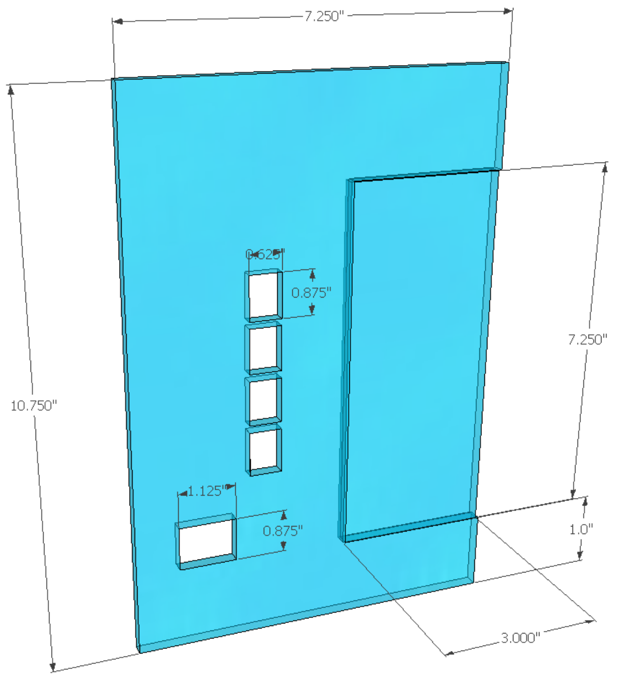


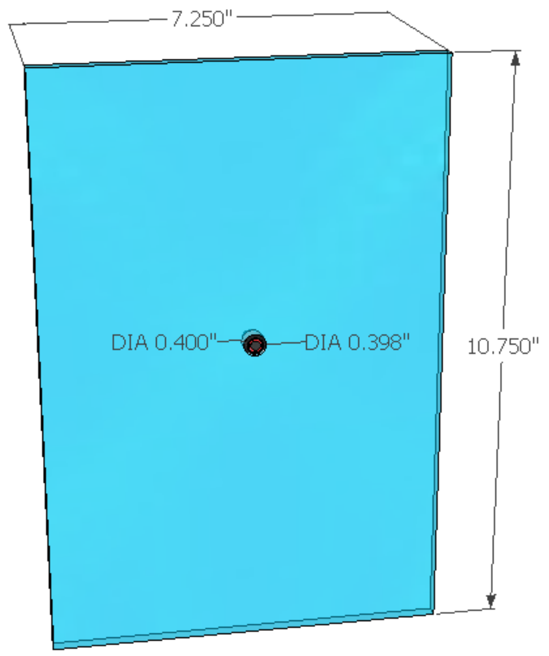




2. Polycarbonate Case Design







Appendix C: Asterisk System

1. extensions.conf – AsteriskNOW version

```
;! Filename: extensions.conf (/etc/asterisk/extensions.conf)
;! Generator: Manager
;! Creation Date: Sun Mar 15 13:06:39 2009
;!
[asterisk_guitools]
exten = executecommand,1,System(${command})
exten = executecommand,n,Hangup()
exten = record_vmenu,1,Answer
exten = record_vmenu,n,Playback(vm-intro)
exten = record_vmenu,n,Record(${var1})
exten = record_vmenu,n,Playback(vm-saved)
exten = record_vmenu,n,Playback(vm-goodbye)
exten = record_vmenu,n,Hangup
exten = play_file,1,Answer
exten = play_file,n,Playback(${var1})
exten = play_file,n,Hangup

[general]
autofallthrough = no
writeprotect = no
static = yes
clearglobalvars = no
priorityjumping = no

[globals]
cellsync = Zap/2
OUTBOUNDTRUNK = Zap/1

[numberplan-custom-1]
plancomment = DialPlan1
include = default
include = parkedcalls
exten = _9XXXX!,1,Macro(trunkdial,${trunk_1}/6${EXTEN:1},${trunk_1_cid})
comment = _9XXXX!,1,Campus,standard

[macro-trunkdial]
exten = s,1,set(CALLERID(all)=${IF($["${LEN(${CALLERID(num)})}" > "6"]?${CALLERI
D(all)}:${ARG2})})
exten = s,n,Dial(${ARG1})
exten = s,n,Goto(s-${DIALSTATUS},1)
exten = s-NOANSWER,1,Hangup
exten = s-BUSY,1,Hangup
exten = _s-.,1,NoOp

[default]
exten = 6050,1,VoiceMailMain

[DID_trunk_1]
exten = s,1,Answer()
exten = s,n,Dial(${cellsync},10)
exten = s,n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy)
exten = s,n(unavail),VoiceMail(6050@default,u)
exten = s,n,Hangup()
exten = s,n(busy),VoiceMail(6050@default,b)
exten = s,n,Hangup()
```

2. extensions.conf - openSUSE version

```
;! Filename: extensions.conf (/etc/asterisk/extensions.conf)
;!
; extensions.conf - the Asterisk Dialplan

; Static extension configuration file, used by
; the pbx_config module. This is where you configure all your
; inbound and outbound calls in Asterisk.
;
; The "General" category is for certain variables.

[general]              ;general context (all contexts are bracketed by [])
static = yes
writeprotect = no
clearglobalvars = no
autofallthrough = no
priorityjumping = no

; The "Globals" category contains global variables that can be referenced
; in the Dialplan with the GLOBAL Dialplan function:
; ${GLOBAL(VARIABLE)}
; ${${GLOBAL(VARIABLE)}} or ${text${GLOBAL(VARIABLE)}} or any hybrid

;set global variables
[globals]
CONSOLE = Console/dsp
IAXINFO = guest
TRUNK = DAHDI/G2
TRUNKMSD = 1
FEATURES =
DIALOPTIONS =
RINGTIME = 20
FOLLOWMEOPTIONS =
PAGING_HEADER = Intercom
trunk_1 = DAHDI/G1
CID_trunk_1 =
SpeedDial1 = 111-111-1111            ;define SpeedDial1
comment = MarkerSpeedDial1           ;label used during search to change speed dial number from
CellSync Menu
SpeedDial2 = 222-222-2222
comment = MarkerSpeedDial2
SpeedDial3 = 333-333-3333
comment = MarkerSpeedDial3
SpeedDial4 = 444-444-4444
comment = MarkerSpeedDial4
SpeedDial5 = 555-555-5555
comment = MarkerSpeedDial5
SpeedDial6 = 666-666-6666
comment = MarkerSpeedDial6
SpeedDial7 = 777-777-7777
comment = MarkerSpeedDial7
SpeedDial8 = 888-888-8888
comment = MarkerSpeedDial8
SpeedDial9 = 999-999-9999
comment = MarkerSpeedDial9

[macro-trunkdial]             ;macro for making outgoing calls
exten = s,1,Dial(${ARG1})            ;dial the number
exten = s,n,Goto(s-${DIALSTATUS},1)
exten = s-NOANSWER,1,Hangup          ;hangup if no one answers
exten = s-BUSY,1,Hangup                      ;hangup if busy
exten = _s-.,1,NoOp

[stdexten]
exten => _X.,50000(stdexten),NoOp(Start stdexten)
exten => _X.,n,Set(LOCAL(ext)=${ARG1})
exten => _X.,n,Set(LOCAL(dev)=${ARG2})
exten => _X.,n,Set(LOCAL(cntx)=${ARG3})

exten => _X.,n,Set(LOCAL(mbx)="${ext}"$["${cntx}" ? "@${cntx}" :: ""])
exten => _X.,n,Dial(${dev},20)  ; Ring the interface, 20 seconds maximum
exten => _X.,n,Goto(stdexten-${DIALSTATUS},1)  ; Jump based on status
(NOANSWER,BUSY,CHANUNAVAIL,CONGESTION,ANSWER)

exten => stdexten-NOANSWER,1,Voicemail(${mbx},u)  ; If unavailable, send to voicemail w/ unavail
announce
exten => stdexten-NOANSWER,n,NoOp(Finish stdexten NOANSWER)
exten => stdexten-NOANSWER,n,Return()  ; If they press #, return to start
exten => stdexten-BUSY,1,Voicemail(${mbx},b)
; If busy, send to voicemail w/ busy announce
exten => stdexten-BUSY,n,NoOp(Finish stdexten BUSY)
exten => stdexten-BUSY,n,Return()  ; If they press #, return to start

exten => _stdexten[\-].,1,Goto(s-NOANSWER,1)  ; Treat anything else as no answer

exten => a,1,VoicemailMain(${mbx})  ; If they press *, send the user into VoicemailMain
exten => a,n,Return()

[default]
exten = _1,1,VoiceMailMain(${CALLERID(num)}@default)

[macro-stdexten]              ;macro for taking incoming calls
exten = s,1,Set(__DYNAMIC_FEATURES=${FEATURES})
exten = s,2,GotoIf($["${FOLLOWME_${ARG1}}" = "1"]?5:3)
exten = s,3,Dial(${ARG2},${RINGTIME},${DIALOPTIONS})        ;dial FXS handset
exten = s,4,Goto(s-${DIALSTATUS},1)
exten = s,5,Macro(stdexten-followme,${ARG1},${ARG2})
exten = s-NOANSWER,1,Voicemail(${ARG1},u)           ;go to voicemail if no answer
exten = s-NOANSWER,2,Goto(default,s,1)
exten = s-BUSY,1,Voicemail(${ARG1},b)               ;go to voicemail if busy
exten = s-BUSY,2,Goto(default,s,1)
exten = _s-.,1,Goto(s-NOANSWER,1)
exten = a,1,VoicemailMain(${ARG1})

[asterisk_guitools]           ;this context creates the functionality of the Asterisk GUI
exten = executecommand,1,System(${command})
exten = executecommand,n,Hangup()
exten = record_vmenu,1,Answer
exten = record_vmenu,n,Playback(vm-intro)
exten = record_vmenu,n,Record(${var1})
exten = record_vmenu,n,Playback(vm-saved)
exten = record_vmenu,n,Playback(vm-goodbye)
exten = record_vmenu,n,Hangup
exten = play_file,1,Answer
exten = play_file,n,Playback(${var1})
exten = play_file,n,Hangup

[macro-trunkdial-failover-0.3]               ;macro for making outgoing calls
exten = s,1,GotoIf($[${LEN(${FMCIDNUM})} > 6]?1-fmsetcid,1)
exten = s,2,GotoIf($[${LEN(${GLOBAL_OUTBOUNDCIDNAME})} > 1]?1-setgbobname,1)
exten = s,3,Set(CALLERID(num)=${IF($[${LEN(${CID_${CALLERID(num)}})} >
2]?${CID_${CALLERID(num)}}:)})
exten = s,n,GotoIf($[${LEN(${CALLERID(num)})} > 6]?1-dial,1)
exten = s,n,Set(CALLERID(all)=${IF($[${LEN(${CID_${ARG3}})} >
6]?${CID_${ARG3}}:${GLOBAL_OUTBOUNDCID})})
exten = s,n,Goto(1-dial,1)
exten = 1-setgbobname,1,Set(CALLERID(name)=${GLOBAL_OUTBOUNDCIDNAME})
exten = 1-setgbobname,n,Goto(s,3)
exten = 1-fmsetcid,1,Set(CALLERID(num)=${FMCIDNUM})
exten = 1-fmsetcid,n,Set(CALLERID(name)=${FMCIDNAME})
exten = 1-fmsetcid,n,Goto(1-dial,1)
exten = 1-dial,1,Dial(${ARG1})
exten = 1-dial,n,Gotoif(${LEN(${ARG2})} > 0 ?1-${DIALSTATUS},1:1-out,1)
exten = 1-CHANUNAVAIL,1,Dial(${ARG2})
exten = 1-CHANUNAVAIL,n,Hangup()
exten = 1-CONGESTION,1,Dial(${ARG2})
exten = 1-CONGESTION,n,Hangup()
exten = 1-out,1,Hangup()

[Incoming_Landline]           ;incoming landline calls routed here
;set Caller ID
exten = s,1,ExecIf($[ "${CALLERID(num)}"="" ],SetCallerPres,unavailable)
exten = s,2,ExecIf($[ "${CALLERID(num)}"="" ],Set,CALLERID(all)=unknown <0000000>)

exten = s,3,Goto(default,1234,1)      ;dial FXS handset
exten = s,n,VoiceMailMain(${CALLERID(num)})          ;go to voicemail if no one answers
```

```
[Outgoing_Landline]           ;outgoing landline calls routed here
;if a four digit number is dialed preceded by a 6, dial this number over the landline campus
network
exten = _6XXXX,1,Macro(trunkdial-failover-0.3,${trunk_1}/6${EXTEN:1},,trunk_1,)

;set Dialplan features
[DLPN_DialPlan1]
include = CellCall
include = CallingRule_Calvin
include = default
include = parkedcalls
include = conferences
include = ringgroups
include = voicemenus
include = queues
include = voicemailgroups
include = directory
include = pagegroups
include = page_an_extension

[Incoming_Cell]               ;incoming calls to the cell phone are routed here
exten = s,1,Goto(default,1234,1)      ;dial the FXS handset
exten = s,n,Answer()                          ;answer the call
exten = s,n,VoiceMailMain(${CALLERID(num)})          ;go to voicemail if no one answers the phone

[Outgoing_Cell]               ; outgoing calls through the cell phone are routed here
;if the number 9 is dialed followed by a 10 digit number, strip the 9 from the number...
;and dial the 10 digit number through the cell phone
;the call is attempted to be made through each registered mobile phone in succession
;the call fails each time until it reaches the phone that is paired, at which time...
;the call is successful (this takes an imperceptibly small amount of time)
exten = _9XXXXXXXXXX,1,Answer()
exten = _9XXXXXXXXXX,n,Dial(Mobile/135798642Phil135798642/${EXTEN:1},15)
exten = _9XXXXXXXXXX,n,Dial(Mobile/135798642Andrew135798642/${EXTEN:1},15)
exten = _9XXXXXXXXXX,n,Dial(Mobile/135798642Keith135798642/${EXTEN:1},15)
comment = LabelOutgoingCell
exten = _9XXXXXXXXXX,n,hangup

;speed dial numbers
;when the number 5 is dialed followed by one other number, dial the speed dial number...
;stored in the globals section for that number through the first paired cell phone
exten = _51,1,Answer()
exten = _51,n,Dial(Mobile/135798642Phil135798642/${SpeedDial1})
exten = _51,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial1})
exten = _51,n,Dial(Mobile/135798642Keith135798642/${SpeedDial1})
comment = LabelSpeedDial1
exten = _51,n,hangup
exten = _52,1,Answer()
exten = _52,n,Dial(Mobile/135798642Phil135798642/${SpeedDial2})
exten = _52,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial2})
exten = _52,n,Dial(Mobile/135798642Keith135798642/${SpeedDial2})
comment = LabelSpeedDial2
exten = _52,n,hangup
exten = _53,1,Answer()
exten = _53,n,Dial(Mobile/135798642Phil135798642/${SpeedDial3})
exten = _53,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial3})
exten = _53,n,Dial(Mobile/135798642Keith135798642/${SpeedDial3})
comment = LabelSpeedDial3
exten = _53,n,hangup
exten = _54,1,Answer()
exten = _54,n,Dial(Mobile/135798642Phil135798642/${SpeedDial4})
exten = _54,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial4})
exten = _54,n,Dial(Mobile/135798642Keith135798642/${SpeedDial4})
comment = LabelSpeedDial4
exten = _54,n,hangup
exten = _55,1,Answer()
exten = _55,n,Dial(Mobile/135798642Phil135798642/${SpeedDial5})
exten = _55,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial5})
exten = _55,n,Dial(Mobile/135798642Keith135798642/${SpeedDial5})
comment = LabelSpeedDial5
exten = _55,n,hangup
exten = _56,1,Answer()
exten = _56,n,Dial(Mobile/135798642Phil135798642/${SpeedDial6})
exten = _56,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial6})
exten = _56,n,Dial(Mobile/135798642Keith135798642/${SpeedDial6})
comment = LabelSpeedDial6
exten = _56,n,hangup
exten = _57,1,Answer()
exten = _57,n,Dial(Mobile/135798642Phil135798642/${SpeedDial7})
exten = _57,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial7})
exten = _57,n,Dial(Mobile/135798642Keith135798642/${SpeedDial7})
comment = LabelSpeedDial7
exten = _57,n,hangup
exten = _58,1,Answer()
exten = _58,n,Dial(Mobile/135798642Phil135798642/${SpeedDial8})
exten = _58,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial8})
exten = _58,n,Dial(Mobile/135798642Keith135798642/${SpeedDial8})
comment = LabelSpeedDial8
exten = _58,n,hangup
exten = _59,1,Answer()
exten = _59,n,Dial(Mobile/135798642Phil135798642/${SpeedDial9})
exten = _59,n,Dial(Mobile/135798642Andrew135798642/${SpeedDial9})
exten = _59,n,Dial(Mobile/135798642Keith135798642/${SpeedDial9})
comment = LabelSpeedDial9
exten = _59,n,hangup
```

3. http.conf

```
;! Filename: http.conf (/etc/asterisk/http.conf)
;! Generator: Manager
;! Creation Date: Thu May  7 11:20:09 2009
;!
;
; Asterisk Builtin mini-HTTP server
;
;
; Note about Asterisk documentation:
;   If Asterisk was installed from a tarball, then the HTML documentation should
;   be installed in the static-http/docs directory which is
;   (/var/lib/asterisk/static-http/docs) on linux by default.  If the Asterisk
;   HTTP server is enabled in this file by setting the "enabled", "bindaddr",
;   and "bindport" options, then you should be able to view the documentation
;   remotely by browsing to:
;       http://<server_ip>:<bindport>/static/docs/index.html
;
[general]
;
; Whether HTTP/HTTPS interface is enabled or not.  Default is no.
; This also affects manager/rawman/mxml access (see manager.conf)
;
enabled = yes
;
; Address to bind to, both for HTTP and HTTPS.  Default is 0.0.0.0
;
bindaddr = 127.0.0.1
;
; Port to bind to for HTTP sessions (default is 8088)
;
bindport = 8088
;
; Prefix allows you to specify a prefix for all requests
; to the server.  The default is blank.  If uncommented
; all requests must begin with /asterisk
;
prefix = asterisk
;
```

```
; Whether Asterisk should serve static content from http-static
; Default is no.
;
enablestatic = yes
;
; Redirect one URI to another.  This is how you would set a
; default page.
;   Syntax: redirect=<from here> <to there>
; For example, if you are using the Asterisk-gui,
; it is convenient to enable the following redirect:
;
redirect = / /static/config/cfgbasic.html
[post_mappings]
backups = /var/lib/asterisk/gui_backups
moh = /var/lib/asterisk/moh
```

4. manager.conf

```
; AMI - The Asterisk Manager Interface
;
; Third party application call management support and PBX event supervision
;
; This configuration file is read every time someone logs in
;
; Use the "manager show commands" at the CLI to list available manager commands
; and their authorization levels.
;
; "manager show command <command>" will show a help text.
;
; ---------------------------- SECURITY NOTE -------------------------------
; Note that you should not enable the AMI on a public IP address. If needed,
; block this TCP port with iptables (or another FW software) and reach it
; with IPsec, SSH, or SSL vpn tunnel.  You can also make the manager
; interface available over http/https if Asterisk's http server is enabled in
; http.conf and if both "enabled" and "webenabled" are set to yes in
; this file.  Both default to no.  httptimeout provides the maximum
; timeout in seconds before a web based session is discarded.  The
; default is 60 seconds.
;
[general]
enabled = yes
webenabled = yes
port = 5038

;httptimeout = 60
; a) httptimeout sets the Max-Age of the http cookie
; b) httptimeout is the amount of time the webserver waits
;    on a action=waitevent request (actually its httptimeout-10)
; c) httptimeout is also the amount of time the webserver keeps
;    a http session alive after completing a successful action

bindaddr = 0.0.0.0

; Parameters that control AMI over TLS. ("enabled" must be set too).
; You can open a connection to this socket with e.g.
;
;       openssl s_client -connect my_host:5039
;
;   sslenable=no              ; set to YES to enable it
;   sslbindport=5039          ; the port to bind to
;   sslbindaddr=0.0.0.0               ; address to bind to, default to bindaddr
;   sslcert=/tmp/asterisk.pem ; path to the certificate.
;   sslcipher=<cipher string>   ; string specifying which SSL ciphers to use or not use
;
;allowmultiplelogin = yes             ; IF set to no, rejects manager logins that are already in
use.
;                               ; The default is yes.
;
;displayconnects = yes
;
; Add a Unix epoch timestamp to events (not action responses)
;
;timestampevents = yes

; debug = on   ; enable some debugging info in AMI messages (default off).
               ; Also accessible through the "manager debug" CLI command.
[cellsync]
secret = cellsync
;deny=0.0.0.0/0.0.0.0
;permit=209.16.236.73/255.255.255.0
;
; If the device connected via this user accepts input slowly,
; the timeout for writes to it can be increased to keep it
; from being disconnected (value is in milliseconds)
;
; writetimeout = 100
;
;displayconnects = yes ; Display on CLI user login/logoff
;
; Authorization for various classes
;
; Read authorization permits you to receive asynchronous events, in general.
; Write authorization permits you to send commands and get back responses.  The
; following classes exist:
;
; system    - General information about the system and ability to run system
;             management commands, such as Shutdown, Restart, and Reload.
; call      - Information about channels and ability to set information in a
;             running channel.
; log       - Logging information.  Read-only.
; verbose   - Verbose information.  Read-only.
; agent     - Information about queues and agents and ability to add queue
;             members to a queue.
; user      - Permission to send and receive UserEvent.
; config    - Ability to read and write configuration files.
; command   - Permission to run CLI commands.  Write-only.
; dtmf      - Receive DTMF events.  Read-only.
; reporting - Ability to get information about the system.
; cdr       - Output of cdr_manager, if loaded.  Read-only.
; Dialplan  - Receive NewExten and VarSet events.  Read-only.
; originate - Permission to originate new calls.  Write-only.
;
read = system,call,log,verbose,agent,user,config,dtmf,reporting,cdr,Dialplan
write = system,call,agent,user,config,command,reporting,originate
```

5. mobile.conf

```
;! Filename: mobile.conf (/etc/asterisk/mobile.conf)
;! Generator: Manager
;! Creation Date: Tue Apr 28 19:05:54 2009
;!
;
; mobile.conf
; configuration file for chan_mobile
```

```
;
[general]
interval = 30

; The following is a list of adapters we use.
; id must be unique and address is the bdaddr of the adapter from hciconfig.
; Each adapter may only have one device (headset or phone) connected at a time.
; Add an [adapter] entry for each adapter you have.

[adapter]
id = BlueZ
address = 00:1B:DC:00:07:1A
;forcemaster=yes      ; attempt to force adapter into master mode. default is no.
;alignmentdetection=yes ; enable this if you sometimes get 'white noise' on asterisk side of the
call
; its a bug in the bluetooth adapter firmware, enabling this will compensate for it.
; default is no.

[adapter]
[135798642Phil135798642]
address = 08:00:7B:F2:65:33   ;135798642Phil135798642
port = 4       ;135798642Phil135798642
context = incoming_mobile     ;135798642Phil135798642
adapter = BlueZ       ;135798642Phil135798642
group = 1      ;135798642Phil135798642

[135798642Andrew135798642]
address = 00:1C:62:06:99:D4   ;135798642Andrew135798642
port = 4       ;135798642Andrew135798642
context = incoming_mobile     ;135798642Andrew135798642
adapter = BlueZ       ;135798642Andrew135798642
group = 1      ;135798642Andrew135798642

[135798642Keith135798642]
address = 08:00:7B:E5:9E:82   ;135798642Keith135798642
port = 4       ;135798642Keith135798642
context = incoming_mobile     ;135798642Keith135798642
adapter = BlueZ       ;135798642Keith135798642
group = 1      ;135798642Keith135798642
```

6. modules.conf

```
;! Filename: modules.conf (/etc/asterisk/modules.conf)
;! Generator: Manager
;! Creation Date: Fri May  1 17:57:08 2009
;!
;
; Asterisk configuration file
;
; Module Loader configuration file
;

[modules]
autoload = yes
preload = func_strings.so
noload = pbx_gtkconsole.so
load = res_musiconhold.so
noload = chan_alsa.so
load = chan_mobile.so ;load needed so that it would pair with cell phone on Asterisk restart
```

Appendix D: Scripts

1. Startup Script

```
#!/bin/sh ;This file was added in the directory /etc/init.d/boot.local to have the script run on boot
asterisk ;starts Asterisk software
modprobe wctdm ;Finds the ZapMicro TNIC
dahdi_genconf ;Generates the configuration file for the TNIC
dahdi_cfg ;Configures the TNIC for use with Asterisk
asterisk -r -x "core restart now" ;Tells the Asterisk CLI to restart Asterisk after all changes have been made
```

2. CellSync Menu Script

```
#!/bin/sh
#CellSync Menu program
# /root/Desktop/CellSyncMenuFiles/CellSyncMenu   #location of CellSync Menu program
clear    #clear the homescreen

menu=5
while [ $menu != 6 ]   #loop through the menu until the user decides to exit
do

clear
#print the CellSync logo to the main menu screen
echo -e "\033[34m\033[1m*********************************                               "
echo -e "\033[34m\033[1m******    ***********************                               "
echo -e "\033[34m\033[1m*****     ***********************                               "
echo -e "\033[34m\033[1m****      *************   **   ** \033[30m\033[0m*********                     "
echo -e "\033[34m\033[1m***   *****************   **   ** \033[30m\033[0m*********                     "
echo -e "\033[34m\033[1m**   ******************   **   ** \033[30m\033[0m**     **                     "
echo -e "\033[34m\033[1m*   *******************   **   ** \033[30m\033[0m**                            "
echo -e "\033[34m\033[1m*   *********       ***   **   **  \033[30m\033[0m**    **         ** **  ****    *******"
echo -e "\033[34m\033[1m*   ********  ****   **   **   **  \033[30m\033[0m ***   **       **  ** ******  *********"
echo -e "\033[34m\033[1m**   *******  ****   **   **   **   \033[30m\033[0m ****  **     **   ***    **  **     **"
echo -e "\033[34m\033[1m***   ******         **   **   **    \033[30m\033[0m  ***  **   **    **     **  **       "
echo -e "\033[34m\033[1m****      **  *********   **   ** \033[30m\033[0m**     **  ** **     **     **  **       "
echo -e "\033[34m\033[1m*****     **  *****  **   **   ** \033[30m\033[0m**     **   ***      **     **  **     **"
echo -e "\033[34m\033[1m******    ***       ***   **   ** \033[30m\033[0m*********   ***      **     **  *********"
echo -e "\033[34m\033[1m********************************* \033[30m\033[0m*********   **       **     **   ******* "
#print the main menu to the screen
echo -e "1. View Mobile Phones                        **"
echo -e "2. Add Mobile Phone                         **"
echo -e "3. Remove Mobile Phone"
echo -e "4. View Speed Dial Numbers"
echo -e "5. Change Speed Dial Number"
echo -e "6. Exit\n"
echo -n "Enter menu number: "     #prompt the user for menu selection
read menu     #read the user's menu selection
clear

case $menu in      #go to the instructions for the menu number the user has selected

       "1")   #View Mobile Phones
       echo -e "\t\033[1mMobile Phones\n\033[0m"
       echo -e "\033[1mName\t\tAddress \033[0m"
       cat /root/Desktop/CellSyncMenuFiles/MobileUsers |    #print the current mobile users to the screen
       while IFS="         " read word1 word2 word3 other
       do
       echo -e "$word1\t\t$word2"
       done

       echo -e "\n"
       read -p "Press enter to go back to main menu" ;;      #pause until the user presses enter

       "2")    #Add Mobile Phone
       echo -n "Phone name: "     #prompt the user for phone name
        read phone_name        #read phone name from the keyboard
       phone_length=${#phone_name}      #find the length of the phone name

       while [ $phone_name != "" ]      #if the user presses enter without typing anything first,
       do                               #the main menu screen will come back up without adding a new phone

       while [ $phone_length -lt 4 ]    #check to see that the length of the phone name is at least 4 characters
        do
       echo "Phone name length must be at least 4 characters long"
       echo "Enter new phone name: "
       read phone_name
       phone_length=${#phone_name}
       done

       clear
              #prompt for Bluetooth address
        echo -n "Enter phone address (include colons) or type "scan" to scan for address: "
        read phone_address


       #if the user entered "scan" instead of the bluetooth address, perform hcitool scan
       while [ $phone_address == "scan" ]
       do
```

```
    echo -e "\n"
    hcitool scan     #search for phones with Bluetooth on in range of adapter and print to screen
    echo -e "scan complete\n"
    echo -e "Enter phone address (include colons) or type "scan" to retry address scan: "
    read phone_address
    done



    while [ ${#phone_address} != 17 ]     #test to see that the entered Bluetooth address is the valid length
     do
     echo "Phone address format is XX:XX:XX:XX:XX:XX"
     echo "Enter phone address (include colons): "
     read phone_address
     done

    clear
    #Code the phone name. This is necessary because of the searching algorithm used when removing a phone...
    #This ensures that only one phone will be removed even when the name of one phone is fully contained in...
    #the name of another phone,
    phone_name_code="135798642${phone_name}135798642"

    #append the necessary lines of code to the end of the MobileUsers file as well as to the...
    #Asterisk extensions.conf file
echo -e "$phone_name\t\t$phone_address\t\t$phone_name_code" >> /root/Desktop/CellSyncMenuFiles/MobileUsers
echo "exten = _9XXXXXXXXXX,n,Dial(Mobile/$phone_name_code/\${EXTEN:1},15)" >> /etc/asterisk/extensions.conf
    echo "exten = _51,n,Dial(Mobile/$phone_name_code/\${SpeedDial1})" >> /etc/asterisk/extensions.conf
    echo "exten = _52,n,Dial(Mobile/$phone_name_code/\${SpeedDial2})" >> /etc/asterisk/extensions.conf
    echo "exten = _53,n,Dial(Mobile/$phone_name_code/\${SpeedDial3})" >> /etc/asterisk/extensions.conf
    echo "exten = _54,n,Dial(Mobile/$phone_name_code/\${SpeedDial4})" >> /etc/asterisk/extensions.conf
    echo "exten = _55,n,Dial(Mobile/$phone_name_code/\${SpeedDial5})" >> /etc/asterisk/extensions.conf
    echo "exten = _56,n,Dial(Mobile/$phone_name_code/\${SpeedDial6})" >> /etc/asterisk/extensions.conf
    echo "exten = _57,n,Dial(Mobile/$phone_name_code/\${SpeedDial7})" >> /etc/asterisk/extensions.conf
    echo "exten = _58,n,Dial(Mobile/$phone_name_code/\${SpeedDial8})" >> /etc/asterisk/extensions.conf
    echo "exten = _59,n,Dial(Mobile/$phone_name_code/\${SpeedDial9})" >> /etc/asterisk/extensions.conf

    #insert the appended lines of code to the proper lines in extensions.conf
    ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/AddNewUserExtensions.exrc

    #append the necessary lines of code to the end of the Asterisk mobile.conf file
    echo "[$phone_name_code]" >> /etc/asterisk/mobile.conf
    echo -e "address = $phone_address\t;$phone_name_code" >> /etc/asterisk/mobile.conf
    echo -e "port = 4\t;$phone_name_code" >> /etc/asterisk/mobile.conf
    echo -e "context = incoming_mobile\t;$phone_name_code" >> /etc/asterisk/mobile.conf
    echo -e "adapter = BlueZ\t;$phone_name_code" >> /etc/asterisk/mobile.conf
    echo -e "group = 1\t;$phone_name_code" >> /etc/asterisk/mobile.conf
     asterisk -r -x "core restart now"    #restart asterisk in order to load changes
    phone_name=""     #set phone_name to "" in order to leave loop and return to main menu
    read -p "Press enter to go back to main menu"
    done ;;

    "3")   #Remove Mobile Phone
     echo -e "\t\033[1mMobile Phones\n\033[1m"
     echo -e "\033[1mName\t\tAddress \033[0m"

            #print registered mobile phones to screen
     cat /root/Desktop/CellSyncMenuFiles/MobileUsers |
     while IFS="           " read word1 word2 word3 other
     do
     echo -e "$word1\t\t$word2"
     done

     echo -e "\nType name to remove and press enter"
    echo -e "Or just press enter to exit without removing name: "
    read remove_name
    remove_name_length=${#remove_name}

    #return to main menu if user pressed enter without typing phone name
    while [ $remove_name != "" ]
    do

     while [ $remove_name_length -lt 4 ]       #ensure that phone name is at least 4 characters long
     do
     echo "All phone names are at least 4 characters long"
     echo "Type name to remove and press enter: "
     read remove_name
     remove_name_length=${#remove_name}
     done

    remove_name="135798642${remove_name}135798642"      #encode phone name to remove

    #edit file stream from MobileUsers and write to temporary file
    sed "/$remove_name/d" /root/Desktop/CellSyncMenuFiles/MobileUsers > /root/Desktop/CellSyncMenuFiles/MobileUsersRemoveCopy
    rm /root/Desktop/CellSyncMenuFiles/MobileUsers     #delete the old MobileUsers file

    #move the temporary MobileUsers file to the old location
    mv /root/Desktop/CellSyncMenuFiles/MobileUsersRemoveCopy /root/Desktop/CellSyncMenuFiles/MobileUsers

    #edit file stream from extensions.conf and write to temporary file
    sed "/$remove_name/d" /etc/asterisk/extensions.conf > /root/Desktop/CellSyncMenuFiles/extensionsRemoveCopy
```

```
    rm /etc/asterisk/extensions.conf      #delete the old extensions.conf file

    #move the temporary extensions.conf file to the old location
    mv /root/Desktop/CellSyncMenuFiles/extensionsRemoveCopy /etc/asterisk/extensions.conf

    #edit file stream from mobile.conf and write to temporary file
    sed "/$remove_name/d" /etc/asterisk/mobile.conf > /root/Desktop/CellSyncMenuFiles/mobileRemoveCopy
rm /etc/asterisk/mobile.conf      #delete the old mobile.conf file

    #move the temporary mobile.conf file to the old location
mv /root/Desktop/CellSyncMenuFiles/mobileRemoveCopy /etc/asterisk/mobile.conf
    asterisk -r -x "core restart now"     #Restart Asterisk to load changes
    remove_name=""
    done
    ;;

    "4")    #View Speed Dial Numbers
     echo -e "\t\033[1mSpeed Dial Numbers\n\033[1m"
     echo -e "\033[1mSpeed Dial\tPhone Number \033[0m"

            #print speed dial numbers to screen
     cat /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers |
     while IFS="           " read speed_dial phone_number label word1 other
     do
     echo -e "$speed_dial\t\t$phone_number"
     done

     echo -e "\n"
     read -p "Press enter to go back to main menu" ;;        #pause until user presses enter



    "5")    #Change Speed Dial Number
     echo -e "\t\033[1mSpeed Dial Numbers\n\033[1m"
     echo -e "\033[1mSpeed Dial\tPhone Number \033[0m"

            #print speed dial numbers
     cat /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers |
     while IFS="           " read speed_dial phone_number label other
     do
     echo -e "$speed_dial\t\t$phone_number"
     done

     echo -e "\nEnter two digit speed dial to change: "  #prompt user for speed dial number to change
     read speed_dial              #read speed dial number to change
```

```
 speed_dial_length=${#speed_dial}     #find speed dial length

while [ $speed_dial != "" ]           #return to main menu if user presses enter before typing number
do

while [ $speed_dial -lt 51 ]    #ensure that speed dial number is greater than or equal to 51
do

 echo "Speed dial is two digits of form 5X "
 echo "Enter new speed dial: "
 read speed_dial



while [ $speed_dial -gt 59 ]  #ensure that speed dial number is less than or equal to 59
do

echo "Speed dial is two digits of form 5X "
 echo "Enter new speed dial: "
 read speed_dial

done
done



       #this command sequence is repeated to account for all possible input sequences
 while [ $speed_dial -gt 59 ]
 do

 echo "Speed dial is two digits of form 5X "
 echo "Enter new speed dial: "
 read speed_dial



 while [ $speed_dial -lt 51 ]
 do

 echo "Speed dial is two digits of form 5X "
 echo "Enter new speed dial: "
 read speed_dial

 done
 done
```

```
        speed_base=`expr $speed_dial - 50`
        speed_base_less_one=`expr $speed_base - 1`

        echo -e "\nEnter phone number in format XXX-XXX-XXXX: "            #prompt the user for phone number
        read speed_phone_number              #read phone number
        speed_phone_number_length=${#speed_phone_number}

        while [ $speed_phone_number_length != 12 ]          #ensure that phone number length is valid
        do
        echo "Speed dial phone number is of form XXX-XXX-XXXX"
        echo "Enter new speed dial phone number: "
        read speed_phone_number
        speed_phone_number_length=${#speed_phone_number}
        done

        #append speed dial progam code to extensions.conf file and SpeedDialNumbers file
        echo "SpeedDial$speed_base = $speed_phone_number" >> /etc/asterisk/extensions.conf
        echo -e "$speed_dial\t\t$speed_phone_number\t\tMarkerSpeedDial$speed_base_less_one" >>
/root/Desktop/CellSyncMenuFiles/SpeedDialNumbers

        #use a specific script file to edit the extensions.conf and SpeedDialNumbers...
        #files depending on the speed dial number
        case $speed_base in
               "1") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial1
                    ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial1 ;;
                "2") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial2
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial2 ;;

                "3") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial3
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial3 ;;

                "4") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial4
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial4 ;;

                "5") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial5
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial5 ;;

                "6") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial6
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial6 ;;

                "7") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial7
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial7 ;;

                "8") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial8
```

```
                        ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial8 ;;

                "9") ex /etc/asterisk/extensions.conf < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial9
                     ex /root/Desktop/CellSyncMenuFiles/SpeedDialNumbers < /root/Desktop/CellSyncMenuFiles/ChangeSpeedDial9 ;;

               *) ;;
        esac

        speed_dial=""
        done
        asterisk -r -x "core restart now"            #restart Asterisk to load changes
        clear
        read -p "Press enter to go back to main menu"
        ;;


        "6")    #Exit
        echo "Good-bye";;

        *)             #handle invalid menu entries
        echo "Invalid menu number"
        read -p "Press enter to go back to main menu" ;;

esac    #end menu case statement

done

exit 0
```

3. CellSync Menu Files

The following files, located in the /root/Desktop/CellsyncMenuFiles folder, are called in the CellsyncMenu shell script program. These files are program function scripts, lists of commands that will be used by the CellsyncMenu shell script.

3.1. MobileUsers

```
#located at /root/Desktop/CellsyncMenuFiles/MobileUsers
#This file stores the mobile users that are registered in the system
Phil       08:00:7B:F2:65:33         135798642Phil135798642
Andrew     00:1C:62:06:99:D4         135798642Andrew135798642
Keith      08:00:7B:E5:9E:82         135798642Keith135798642
```

3.2. AddNewUserExtensions.exrc

```
#located at /root/Desktop/CellsyncMenuFiles/AddNewUserExtensions.exrc
#This script is called in CellsyncMenu with the ex command to edit the extensions.conf file
$-9 co /LabelOutgoingCell/-
$-8 co /LabelSpeedDial1/-
$-7 co /LabelSpeedDial2/-
$-6 co /LabelSpeedDial3/-
$-5 co /LabelSpeedDial4/-
$-4 co /LabelSpeedDial5/-
$-3 co /LabelSpeedDial6/-
$-2 co /LabelSpeedDial7/-
$-1 co /LabelSpeedDial8/-
$ co /LabelSpeedDial9/-
$-9,$ d
wq
```

3.3. SpeedDialNumbers

```
#located at /root/Desktop/CellsyncMenuFiles/SpeedDialNumbers
#This file stores the speed dial numbers and their settings
                                    MarkerSpeedDialNeg
51          111-111-1111            MarkerSpeedDial0
52          222-222-2222            MarkerSpeedDial1
53          333-333-3333            MarkerSpeedDial2
54          444-444-4444            MarkerSpeedDial3
55          555-555-5555            MarkerSpeedDial4
56          666-666-6666            MarkerSpeedDial5
57          777-777-7777            MarkerSpeedDial6
58          888-888-8888            MarkerSpeedDial7
59          999-999-9999            MarkerSpeedDial8
                                    MarkerSpeedDial9
```

3.4. ChangeSpeedDial1 through 9

```
#ChangeSpeedDial1
#located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
#This file, as well as the similar ones following, it is called in the CellsyncMenu program and…
#is used to edit the Speed DialNumbers file
/MarkerSpeedDial1/- d
$ co /MarkerSpeedDial1/-
$ d
wq
#ChangeSpeedDial2      located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
/MarkerSpeedDial2/- d
$ co /MarkerSpeedDial2/-
$ d
wq

#ChangeSpeedDial3     located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
/MarkerSpeedDial3/- d
$ co /MarkerSpeedDial3/-
$ d
wq

#ChangeSpeedDial4      located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
/MarkerSpeedDial4/- d
$ co /MarkerSpeedDial4/-
$ d
wq

#ChangeSpeedDial5      located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
/MarkerSpeedDial5/- d
$ co /MarkerSpeedDial5/-
$ d
wq

#ChangeSpeedDial6      located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1
/MarkerSpeedDial6/- d
$ co /MarkerSpeedDial6/-
$ d
wq
```

#ChangeSpeedDial7 located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1

```
/MarkerSpeedDial7/- d
$ co /MarkerSpeedDial7/-
$ d
wq
```

#ChangeSpeedDial8 located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1

```
/MarkerSpeedDial8/- d
$ co /MarkerSpeedDial8/-
$ d
wq
```

#ChangeSpeedDial9 located at /root/Desktop/CellsyncMenuFiles/ChangeSpeedDial1

```
/MarkerSpeedDial9/- d
$ co /MarkerSpeedDial9/-
$ d
wq
```

Appendix E: Bluetooth Configuration Files

1. hcid.conf

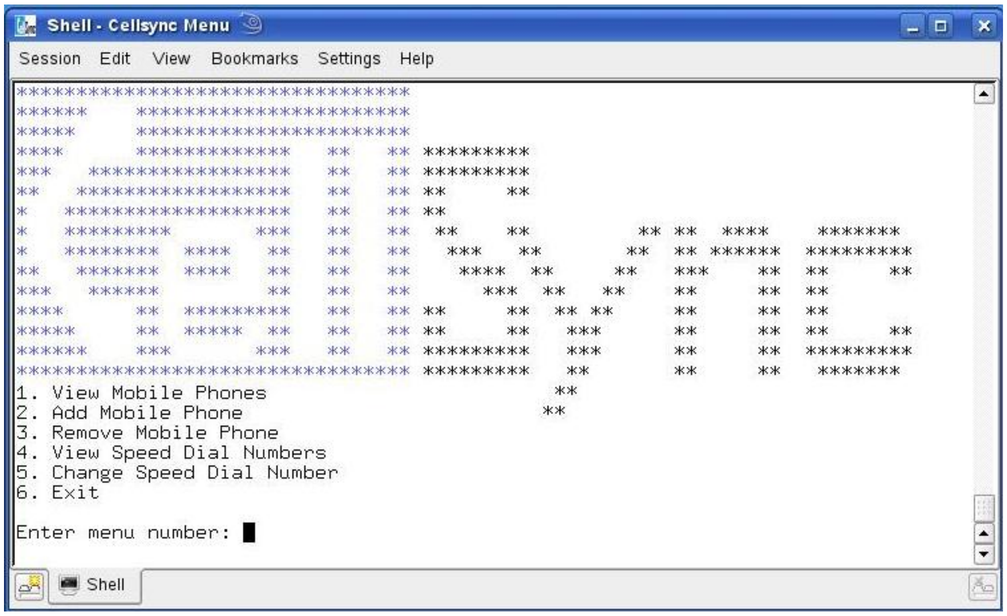
```
#
# HCI daemon configuration file.
#

# HCId options
options {
        # Automatically initialize new devices
        autoinit yes;

        # Security Manager mode
        #   none - Security manager disabled
        #   auto - Use local PIN for incoming connections
        #   user - Always ask user for a PIN
        #
        security auto;

        # Pairing mode
        #   none  - Pairing disabled
        #   multi - Allow pairing with already paired devices
        #   once  - Pair once and deny successive attempts
        pairing multi;

        # PIN Helper
        pin_helper /etc/bluetooth/pin;

        # Default PIN code for incoming connections
        passkey "1234";
}

# Default settings for HCI devices
device {
        # Local device name
        #   %d - device id
        #   %h - host name
        name "BlueZ";


        # Local device class
        # e.g.
        #  0xsss100 = Computer
         #  0xsss104 = Computer Desktop
         #  0xsss108 = Computer Server
         #  0xsss10c = Computer Laptop
         # The 'sss' above defines the service-class (not quite, only the
         # first 11 bits, the next 11 define the device-class, than 2 format bits.)
         # See https://www.bluetooth.org/foundry/assignnumb/document/baseband
         # for more information.
         # 0x100bbb stands for "Object Transfer  (v-Inbox, v-Folder, ...)"
         # 0x020bbb stands for "Networking (LAN, Ad hoc, ...)"
        class 0x100104;

        # Default packet type
        #pkt_type DH1,DM1,HV1;

        # Inquiry and Page scan
        # valid parameters: enable | disable
        iscan enable;
        pscan enable;
        discovto 0;

        # Default link mode
        #   none   - no specific policy
        #   accept - always accept incoming connections
        #   master - become master on incoming connections,
        #            deny role switch on outgoing connections
        lm accept;

        # Default link policy
        #   none    - no specific policy
        #   rswitch - allow role switch
        #   hold    - allow hold mode
        #   sniff   - allow sniff mode
        #   park    - allow park mode
        lp rswitch,hold,sniff,park;
}
```

2. PIN Files

```
/etc/bluetooth/pin ;location of pin file and its contents

PIN:1234 ;We set this value to be an easy 4 digits, but can be longer

/etc/bluetooth/pin_helper ;location of pin_helper file and its contents

echo -n "1234:" ;Tells Bluetooth to use 1234 as the PIN
```

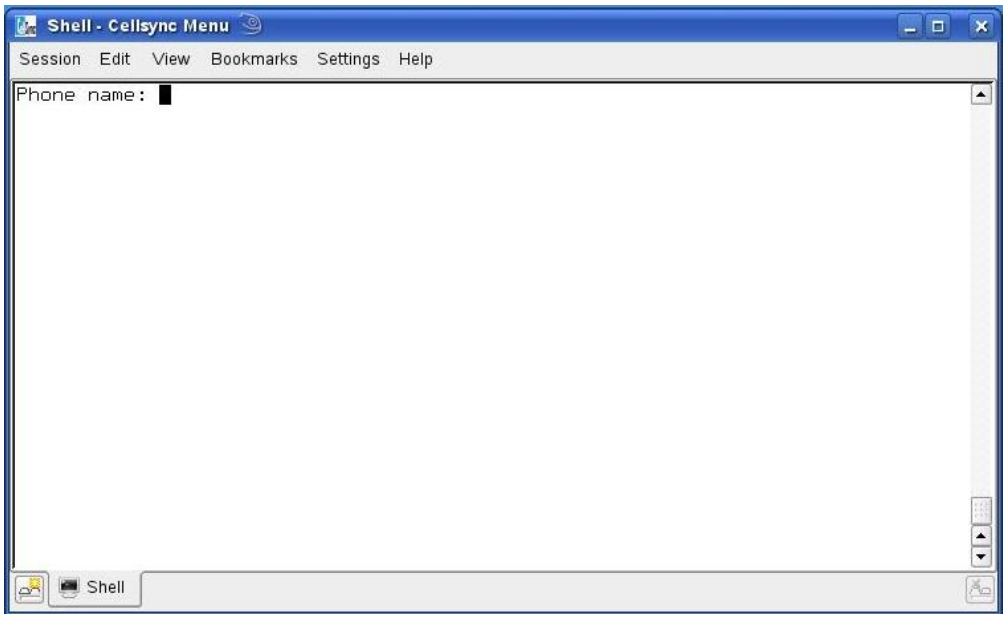
Appendix F: CellSync Menu Screenshots

1. CellSync Main Menu



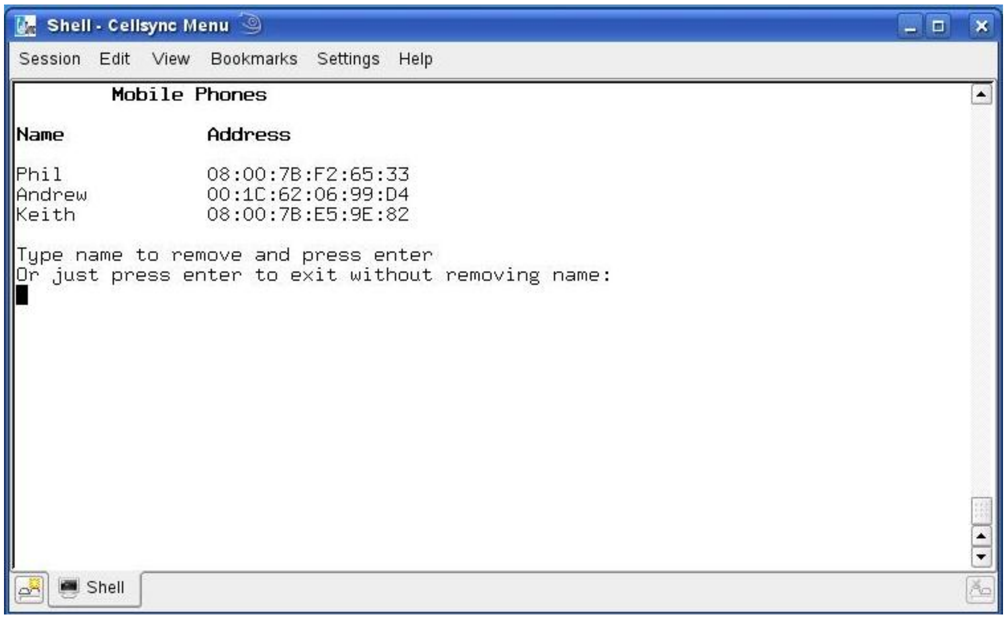
```
*********************************
******    ***********************
*****     ***********************
****      *************   **   ** *********
***   *****************   **   ** *********
**   ******************   **   ** **     **
*   *******************   **   ** **
*   *********       ***   **   **  **    **         ** **  ****    *******
*   ********  ****   **   **   **   ***   **       **  ** ******  *********
**   *******  ****   **   **   **    ****  **     **   ***    **  **     **
***   ******         **   **   **      ***  **   **    **     **  **
****      **  *********   **   ** **     **  ** **     **     **  **
*****     **  *****  **   **   ** **     **   ***      **     **  **     **
******    ***       ***   **   ** *********   ***      **     **  *********
********************************* *********   **       **     **   *******
1. View Mobile Phones                        **
2. Add Mobile Phone                         **
3. Remove Mobile Phone
4. View Speed Dial Numbers
5. Change Speed Dial Number
6. Exit

Enter menu number:
```

2. Add Mobile Phone



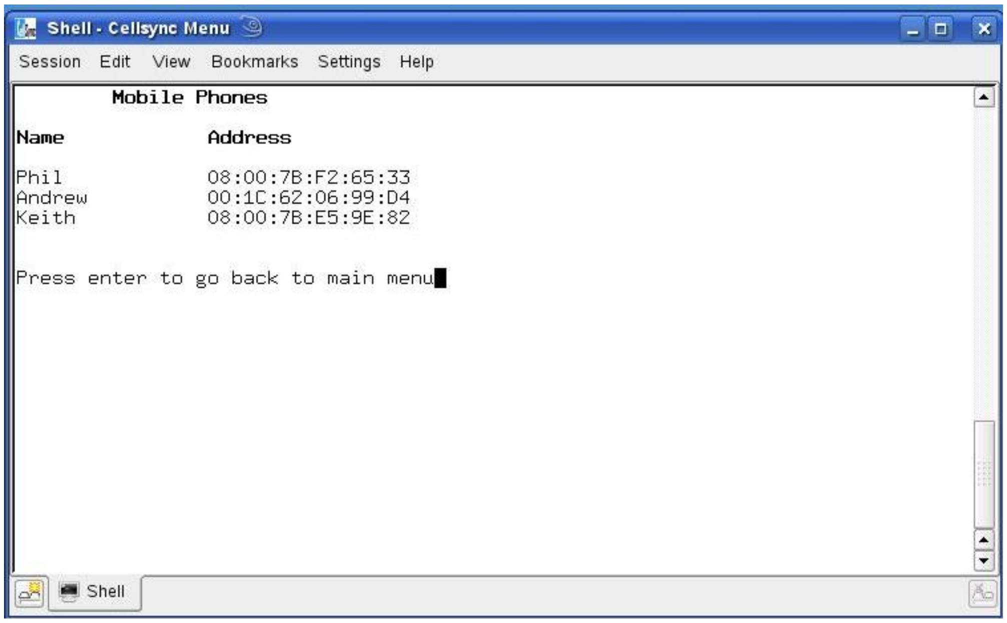
```
Phone name:
```

3. Remove Mobile Phone



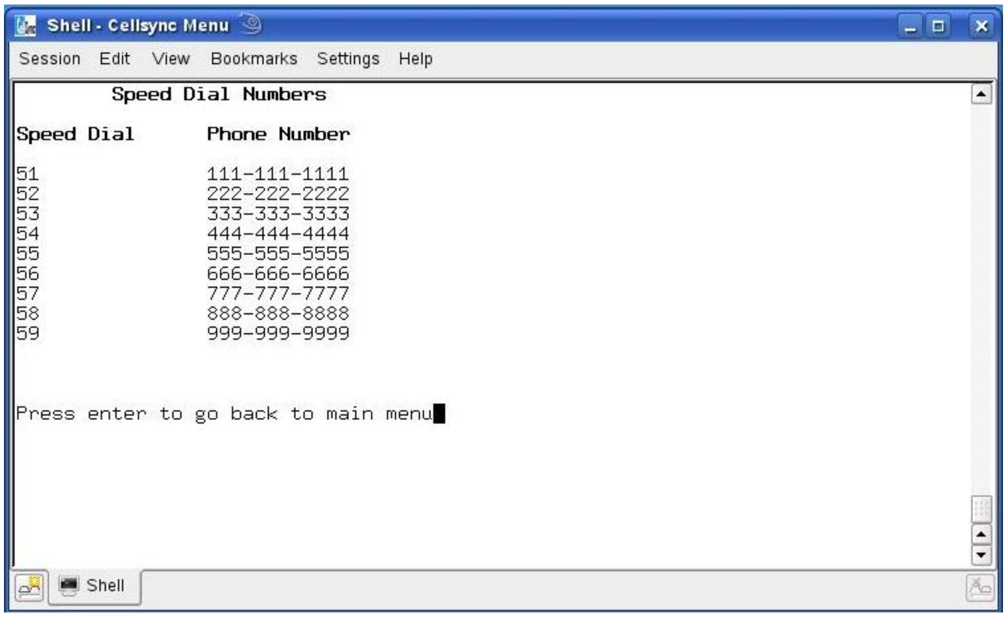
```
        Mobile Phones

Name            Address

Phil            08:00:7B:F2:65:33
Andrew          00:1C:62:06:99:D4
Keith           08:00:7B:E5:9E:82

Type name to remove and press enter
Or just press enter to exit without removing name:
```

4. View Mobile Phones



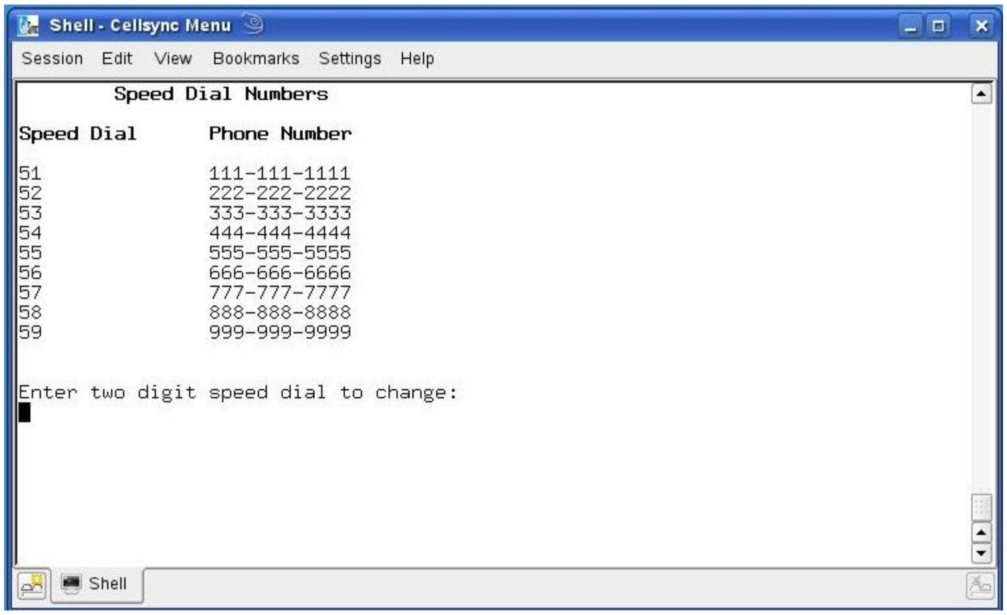
```
        Mobile Phones

Name            Address

Phil            08:00:7B:F2:65:33
Andrew          00:1C:62:06:99:D4
Keith           08:00:7B:E5:9E:82


Press enter to go back to main menu
```

5. View Speed Dial



```
         Speed Dial Numbers

Speed Dial        Phone Number

51                111-111-1111
52                222-222-2222
53                333-333-3333
54                444-444-4444
55                555-555-5555
56                666-666-6666
57                777-777-7777
58                888-888-8888
59                999-999-9999


Press enter to go back to main menu
```

6. Change Speed Dial



```
         Speed Dial Numbers

Speed Dial        Phone Number

51                111-111-1111
52                222-222-2222
53                333-333-3333
54                444-444-4444
55                555-555-5555
56                666-666-6666
57                777-777-7777
58                888-888-8888
59                999-999-9999

Enter two digit speed dial to change:
```

Appendix G: Asterisk Log Errors

```
[…]
[Apr 14 11:44:01] ERROR[5023] codec_dahdi.c: Failed to open
/dev/dahdi/transcode: No such file or directory
[Apr 14 11:44:01] WARNING[5023] app_voicemail.c: Setting 'maxmessage' has
been deprecated in favor of 'maxsecs'.

[Apr 14 11:44:01] WARNING[5023] app_voicemail.c: Setting 'minmessage' has
been deprecated in favor of 'minsecs'.
[Apr 14 11:44:01] WARNING[5023] translate.c: plc_samples 160 format f
[Apr 14 11:44:01] ERROR[5023] chan_mobile.c: Unable to open adapter blue. It
won't be enabled.

[Apr 14 11:44:01] ERROR[5023] chan_mobile.c: Unable to open adapter dlink. It
won't be enabled.
[Apr 14 11:44:01] WARNING[5023] chan_mobile.c:
***********************************************************************

[Apr 14 11:44:01] WARNING[5023] chan_mobile.c: No Adapters defined. Please
review mobile.conf. See sample for details.
[Apr 14 11:44:01] WARNING[5023] chan_mobile.c:
***********************************************************************

[Apr 14 11:44:01] ERROR[5023] chan_mobile.c: Device LGTU550 configured to use
unknown adapter dlink. It won't be enabled.
[Apr 14 11:44:01] WARNING[5023] chan_usbradio.c: File usbradio_tune.conf not
found, using default parameters.

[Apr 14 11:44:01] WARNING[5023] chan_usbradio.c: File usbradio_tune.conf not
found, using default parameters.
[Apr 14 11:44:01] ERROR[5023] chan_usbradio.c: No txvoice output configured.
[Apr 14 11:44:01] NOTICE[5023] chan_ooh323.c: -------------------------------
--------------------------------------------------
[…]
[Apr 14 11:44:22] WARNING[5130] config.c: Unknown directive '#' at line 1 of
/etc/asterisk/../dahdi/system.conf
[Apr 14 11:44:22] WARNING[5130] config.c: Unknown directive '#' at line 2 of
/etc/asterisk/../dahdi/system.conf

[Apr 14 11:45:02] WARNING[5389] app_system.c: Unable to execute 'undefined'
[Apr 14 11:45:16] ERROR[5458] callerid.c: No start bit found in fsk data.
[Apr 14 11:45:16] WARNING[5458] chan_dahdi.c: CallerID feed failed: Success
[…]
[Apr 14 12:14:40] WARNING[6422] app_dial.c: Unable to create channel of type
'DAHDI' (cause 0 - Unknown)
[Apr 14 12:14:49] WARNING[6426] chan_dahdi.c: Unable to determine channel for
data trunk_1/66017
[Apr 14 12:14:49] WARNING[6426] app_dial.c: Unable to create channel of type
'DAHDI' (cause 0 - Unknown)
[…]
[Apr 14 12:19:28] ERROR[6539] res_config_ldap.c: No directory URL or host
found.
[Apr 14 12:19:28] NOTICE[6539] res_config_ldap.c: Cannot reload LDAP RealTime
driver.
[Apr 14 12:19:28] NOTICE[6539] pbx_ael.c: Starting AEL load process.
```