BROWN
School of Engineering
Brown University

The Perfect Sportscast
An Adjustable Radio/TV Delayer with Bluetooth

Technical Documentation and Development Roadmap

Submitted in partial fulfillment of the requirements of the Senior Capstone Design course (ENGN 1650 - Embedded Microprocessor Design) at the School of Engineering at Brown University

May 2015

Submitted by:
Nakhare, Gaurav
Stanley, Jonah
Taub, Brandon

Table of Contents

1 - Introduction

This report outlines the design, development, and testing of an embedded system prototype for a real-time radio/TV delayer with Bluetooth output. This project was part of the *ENGN1650: Embedded Microprocessor Design* course fulfilling the capstone design course requirement for undergraduate engineers.

1.1 - Product Description

The Perfect Sportscast - Bluetooth edition (referred to henceforth as 'radio-delayer' or 'product') is an adjustable-delay AM/FM radio/TV synchronizer, intended to be a compact, pocket-sized device that delays a user-selected radio feed (AM/FM station) by a pre-specified amount between 0 and 30 seconds. This means that the user can listen to radio commentary of an event (for example, during a football game) while watching live streaming of the game on TV. The radio-delayer provides two options for audio playback: headphones (standard 3.5mm jack compatible with a wide range of wired headphones) and Bluetooth connection capability to sync with an external speaker. The LCD display provides a user interface to a variety of other features, including up to 6 radio presets, volume control, options to sync with and "remember" Bluetooth devices, and time-delay settings, and the product can be charged via USB.

The desired specs for the product included a full AM/FM radio with presets, adjustable up to 30 seconds of delay, with 3-5 hours battery life (easily rechargeable using USB). A built-in speaker option was discontinued in favor of Bluetooth compatibility. The final, technical specifications, for the product are detailed in Section 2.2 of this report.

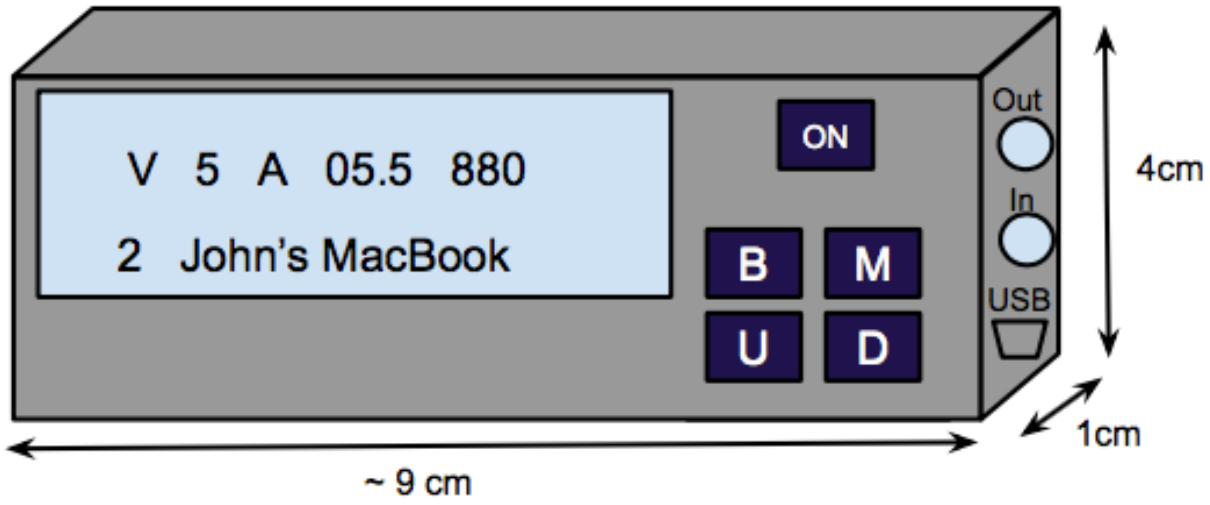

Figure 1: Prototype sketch with dimensions

1.2 - Motivation and Intended Applications

Many sports fans prefer the radio commentary of a sporting event to the television commentary. However, the TV feed is often behind the radio signal, and this delay itself can change. If the delay between the two sources (TV and radio) is more than a couple of seconds (it has been found to be up to 30 seconds[1]), runs or points scored will be announced over the radio before shown on the TV thus having a negative effect on the viewing experience. The *Perfect Sportscast - Bluetooth edition* aims to solve that problem by adding a delay to the radio signal and allowing the two devices to be completely synchronized. While originally designed with sports fans in mind, this product can also be used for the news, special announcements and addresses etc.

1.3 - Existing Competition

The design of the *Perfect Sportscast - Bluetooth edition* involved research into existing products that serve similar purposes. Two such products are described below.

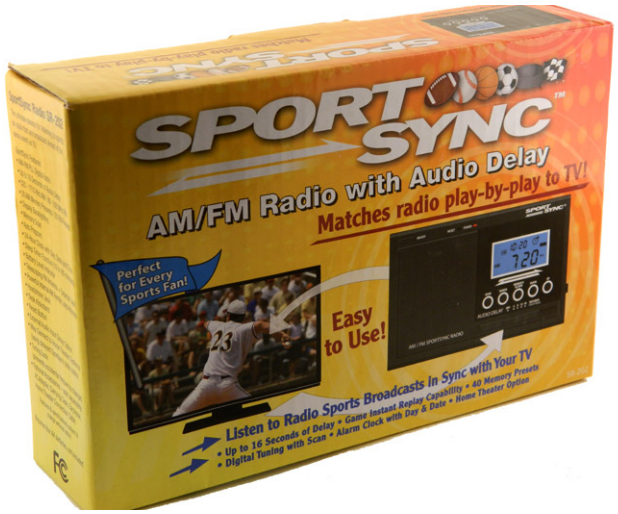

Figure 2: Sport Sync Radio device

The first product is the Sport Sync Radio (http://www.sportsyncradio.com), which is also an audio delayer. However, this product can only delay up to 16 seconds (as opposed to 30 seconds by our product), and at $69.95 still requires separate purchases for accessories such as batteries, AC Adapter and home theater cable.

Easy Radio Delayer (http://www.easyradiodelayer.com) is a free smartphone app that synchronizes radio applications with digital TV. Being a software product, however, it cannot be used for physical TV, and requires a stable connection to an internet radio station. Additionally, this product has a limited list of radio stations (AM/FM frequencies) that can be tuned into. Current limitations of this product include the fact that when used with 0 second delay, the radio audio comes after TV (i.e. the delay of the product is greater than that of the TV with respect to radio), effectively rendering the product unhelpful.

[1] http://en.wikipedia.org/wiki/Broadcast_delay

Additionally, the Bluetooth compatibility is a feature unique to our product, which we feel is important for customers who have already invested in a durable Bluetooth speaker. Bluetooth SIG, Inc. has forecasted "the Bluetooth enabled wireless audio market is forecasted to grow to more than 250 million units by 2018.[2]" and according to research from ReportsnReports.com "the global Bluetooth speaker market is set to grow at a CAGR of 38.73% during the forecast period 2014-2019.[3]"

2 - Functional and Technical Specifications

Functional specifications describe the desired user interface and product behavior at a high-level, while technical specifications provide numerical data for key features of the product's components.

2.1 – Functional Specifications

2.1.1 - Delay Mechanism

The delay mechanism is achieved by compressing and storing data in the microprocessor (BF707 from Analog Devices), and decompressing and playing out (either via 3.5mm audio jack or Bluetooth). The storage is done in the on-board memory of the microprocessor.

2.1.2 - Device Mobility

The *Perfect Sportscast - Bluetooth edition* is designed to be a pocket-sized device, both in terms of PCB footprint and weight. The other requirement is that it is battery-powered for easy portability, with a battery life that is greater than an average football game (estimated to be around 3-5 hours with an average game length of 3.12 hours[4]). Non-rechargeable batteries involve cost, complexity in the case, and frustration for the customer. The solution was thus to use a Li-Ion rechargeable battery that can be charged without explicitly removing the battery. We chose a battery charged by USB (5V) with the option of a wallwart using an external 5V AC-DC adapter.

2.1.3 - Visual Feedback

An LCD display is an integral part of the product for visual feedback to increase/decrease volume, delay, frequency, set presets, choose AM/FM, and connect to a Bluetooth speaker. There is also a set of LEDs to indicate that the battery is being charged. More details can be found in Section 4.4.

[2] http://www.bluetooth.com/Pages/Consumer-Electronics-Market.aspx
[3] http://www.prnewswire.com/news-releases/bluetooth-speakers-market-worth-7-billion-by-2019-283587821.html
[4] http://qz.com/150577/an-average-nfl-game-more-than-100-commercials-and-just-11-minutes-of-play/

2.2 – Technical Specifications

Parameter	Value	Unit	Comments
USB Micro-B Receptacle			
- Maximum Current	1.8	A	per contact
- Maximum Voltage	100	V	AC power
- Resistance	30	mΩ	Contacts
3.5 mm Headphone Jack			
- Typical Input Voltage	12	V	DC
- Maximum Input Current	1	A	
- Maximum Contact Resistance	50	mΩ	Between terminal and mating
- Voltage Withstand	500	V	plug (AC, at 50/60 Hz for 1 min)
TPA6130A2 Headphone Driver			
- Minimum Load Impedance	12.8	Ω	
- Minimum/Maximum Supply Voltage	2.5/5.5	V	Recommended conditions
- Signal-Noise Ratio	98	dB	
LM3658 Charging Chip			
- AC Adapter Voltage Min/Max	4.5/6	V	
- USB Power Voltage Min/Max	4.35/6	V	
- USB Full-Rate Charge Current	450/500	mA	Typical/maximum values
CM1214 ESD Protection			
- Maximum Voltage between channels	7	V	
- Maximum Power Rating	225	mW	
TL3315NF160Q			
- Contact Resistance	200	mΩ	
- Dielectric Strength	250	VAC	
BF707			
-$V_{DD-EXT1}$	1.8	V	
-$V_{DD-EXT2}$	3.3	V	
- F_{CCLK}	400	MHz	
- F_{OCLK}	50	MHz	Output Clock
- I_{IH}	10	uA	Input Current
- C_{IN}	5.2	pF	
- I_{DD}	90	mA	
- L1 cache	1	MB	
- GPIO	47	pins	
UBP002 Battery			
- Typical Voltage Range	3.0-4.2	V	
- Average Voltage	3.7	V	
- Capacity	900	mAh	
ADP1612			
- Typical Voltage Range	1.8-5.5	V	
- Quiescent Current Range	700-1300	uA	
- Typical Output Voltage Range	Vin-20	V	

- Oscillator Frequency	550-720	kHz	
Ferrite Loop Antenna - Inductance	200	uH	Can be tuned for Si4730
ADP5024 - Typical Buck Voltage Range - Typical LDO Voltage Range - Startup Time (BUCK1, LDO) - Startup Time (BUCK2) - Buck Output Voltage Accuracy - Buck Current Limit - LDO Current Limit	 2.4-5.5 1.7-5.5 250 300 +/-1.8 1.2 300	 V V us us % A mA	
Si4730 - Analog Input Voltage Range - Digital Input Voltage Range - Digital Output Current Range (FM) - Analog Output Current Range (FM) - Digital Output Current Range (AM) - Analog Output Current Range (AM)	 2.7-5.5 1.6-3.6 10.5-13.5 9.9-12.5 8.5-11 8-10.2	 V V mA mA mA mA	
MIC803-26VD3VC Timer IC - Operating Voltage Range - Vcc to RESET delay - Supply Current Range	 1.0-5.5 10 4.5-10	 V us uA	
BTM511 -UART Data Rate - Voltage Range - $V_{I/O}$ Range - Current during operation - Idle Current	 >300 3-3.6 1.7-3.6 >70 >1	 kbps V V mA mA	
LCM-S01602DTR/M - Voltage Range - Current Range	 4.7-5.5 2-3	 V mA	
NS2520SB-32_768KHZ-NSA3536C - Output Frequency - Voltage - Current - Rise/Fall Time - C_L	 32.768 1.8/2.5/3.3 .22/.24/.26 200 15	 kHz V mA ns pF	
ECS-2025-2033 Oscillator - Frequency Range - V_{DD} - Rise/Fall time - Current	 .75-75 3.3 10 11	 MHz V ns mA	

Table 1: Summary of Technical Specifications and Electrical Characteristics for major components of the product

3 - Development Summary

3.1 - Division of Tasks

Primary project roles and contributions were as follows:

Nakhare, Gaurav
Responsibilities: Hardware design and component selection (radio chip/voltage regulator selection, schematic entry), product design (external case), and Fall 2014 schedule management

Gaurav's major role was on the hardware side. During the initial stages of the project, he designed the block diagrams for the product details various voltage levels, and used this to select components for the power circuitry. He also selected the Silicon Labs solution for the radio IC. He was responsible for the weekly meetings in the fall semester to ensure presentations (for class) were updated with progress. Towards the end of the Fall semester, he worked on schematic entry in DxDesigner, particularly selecting the components for passives, verifying that IC supporting circuitry matched supplier reference designs, and including external oscillators as recommended by Professor Patterson. In the spring, he worked on the initial portion of the layout and debugging the hardware prototype for the radio IC. He assisted with the assembly of the BF707 and Bluetooth module, and debugging of the ADP5024 power supply.

Taub, Brandon
Responsibilities: Market research, hardware component selection, schematic entry, board layout, assembly and testing, initial BF707 code porting

Brandon's responsibilities on the team involved both software and hardware development. In the fall, he took charge of researching existing products in the market and determined how best to proceed with this project in terms of price point and target market. Brandon worked with Ken early on to get existing code for UART and I2C running on the new processor. He was in charge of schematic entry, and created most of the new schematic symbols for components. He also handled most of the layout, and worked closely with Professor Patterson to layout and route the board in the easiest way possible. Finally, he worked with Professor Patterson and Arpie Kaloustian on board assembly, bring-up, and testing.

Stanley, Jonah
Responsibilities: Bluetooth, Software design, testing and integration of hardware components

Jonah's major role on the team was to be the leader on the software side of development. In the fall, he mainly looked into the various Bluetooth modules and how to integrate them with the design. In addition, he worked on algorithms for the general program flow of the prototype. Most of that work was with Ken Silverman to work with the BF707 microprocessor to get it to communicate with the various components. In the spring, Jonah mostly worked on integrating each component's software into the microprocessor and making test programs in order to fully prove their functionality.

3.2 – Acknowledgements

Additional acknowledgments are due to the following people:

Professor William Patterson
Professor Patterson spent a lot of time teaching us about the protocols that would be used on our board, designed a radio-board prototype to help test the radio chip, reviewed our schematics, and completed the tricky portions of the BF707 layout and board assembly and bringup.

Professor Harvey Silverman
Professor Silverman served as the inspiration for the project, provided constructive criticism, suggested the BF707, provided lab space and equipment, and helped debug the radio board and final-product board.

Ken Silverman
Ken helped greatly with the testing of the BF707 chip. As it was a new product, much of the code for the BF50x series processor needed to be ported over to the new processor. He also helped greatly with debugging test code and making sure all of the communication protocols were accurately set up.

Arpie Kaloustian
Arpie patiently dealt with requests for soldering parts on the final boards and pins on various test points at different stages of debugging.

3.2 - Timeline

(B = Brandon, J = Jonah, G = Gaurav)

Week 1: (Sep 10 - Sep 16)
B: After settling on the Sports Radio idea, researched market openings.
J: Initial specifications of the product, prototype sketch, initial cost estimates
G: Initial calculations of sampling frequency, ADC precision, delay, and required memory, and initial block diagram

Week 2: (Sep 17 - Sep 23)
B, G, J: Initial box sketch and button layout
B: Initial cost estimates of buttons, LCD display, SRAM, and DRAM, microprocessor, Bluetooth module, and batteries
J: Researched Bluetooth specifications and feasibility, initial structure of preset memory
G: Updated product specification to reflect new components and cost structure

Week 3: (Sep 24 - Sep 30)
B, G, J: Decreased dimensions of the box, and simplified the button layout by using fewer buttons, researched potential microcontrollers
B: Researched Si4730 radio chip and peripheral components; updated cost estimates.

J: Researched UART Bluetooth devices and potential microprocessor requirements
G: Created detailed block diagram down to part level interactions and communication protocols

Week 4: (Oct 1 - Oct 7)
B, G, J: Updated current draw estimates and costs
B: Added USB-micro interface to charge board and audio out jack to the physical sketch, researched DRAM modules and costs
J: Researched Nordic Bluetooth module
G: Updated block diagram with charging components, researched details of charging system and power supply circuitry.

Week 5: (Oct 8 - Oct 14)
B: Researched cheap option for tactile switches - found the switches to be used in the final product. Researched battery options based on updated power estimates.
J: Decided to use RN4020 Bluetooth module instead of Nordic device because of speed and ease of interface with BF707. Began to brainstorm software and evaluation board to be used with BF707. Brainstormed compression algorithm.
G: Updated box-sketch dimensions, with interface consisting of only four buttons. Updated block diagram with the initial solutions for buck/boost chips and removed external memory chips. Looked into DAC options. Calculated product power consumption.

Week 6: (Oct 15 - Oct 21)
B: Researched other switch options, (including ones with rubber membrane cover), both blocks of switches and individual switches. Updated power supply calculations with the BTM511 power requirements.
J: Added Lithium-Ion battery to the block diagram, and specified protocols on some connections on the block diagram. Looked into alternatives for RN4020 because of shortcomings discovered. Looked into Laird and TiWi-uB2 options. Wrote initial Bluetooth connection pseudocode.
G: Updated device sketch based on new tactile switch size, and came up with initial dimensions estimates. Looked into new ADP buck/boost parts which were recommended by course instructors.

Week 7: (Oct 22 - Oct 28)
B, G, J: Determined peripheral resistors and capacitors that would need to be used with the radio chip that was selected (these requirements seemed reasonable).
B: Researched possible mechanical switch options, switch part, looked into circuit requirements, researched specifications. Discussed headphone driver options with professors, based on peripheral requirements and advanced features of some of the parts. Worked with Ken Silverman on BF707: ported old code, implemented communication protocols including UART and I2C, and began working on SPI.
J: Communicated with Laird about DAC requirement and codecs needed to make the Bluetooth module work.
G: Updated device sketch based on smaller depth but larger height of the UBP002 battery. Researched battery specifications and cost to select the UBP002. Looked into ADP2504 buck/LDO solution and computed values for supporting circuitry. Updated Bluetooth and battery parts on the block diagram.

Week 8: (Oct 29 - Nov 4)
B: Looked into AD5663 option for DAC for Bluetooth module (allowed sampling up to 48 kHz).
J: Ordered Bluetooth module, researched ways of communicating with it (UART, PCM, I2S, analog). Decided to use analog, which would require use of DAC.
G: Added DAC, "on" button, and updated GPIO connections to the block diagram to include LCD display interface. Had radio chip mounted on test board, and built the breadboard circuit (as per Silicon Labs recommended design) to start testing radio chip functionality.

Week 9: (Nov 5 - Nov 11)
B: Began schematic entry: created symbols for Si4730, TDA7266D (headphone driver), LM4910 (FM antenna driver), and obtained Analog Devices symbols from Professor Silverman. Started the circuitry around the power path.
J: Having initial problems communicating between the BF707 and the radio chip over I2C. Decided that no DAC would be needed because the BF707 can communicate over I2S. Able to make Bluetooth module send music to a Bluetooth speaker as well as connect to iPhone.
G: Worked on schematic entry to begin the circuitry for the radio IC, complete the power path, and add BF707 peripheral components. Added headphone amplifier and removed external DAC from block diagram. Updated connection interfaces. Continued breadboarding around the radio chip.

Week 10: (Nov 12 - Nov 18)
B: Schematic entry: created symbols for the rest of the parts, added 0 Ohm resistors between sections of the power path, added additional peripherals to circuit.
J: Able to power up radio chip over I2C and get revision number and some properties.
G: Removed LM4910 from block diagram, replace old boost regulator with ADP1612, and added ADP5024 to provide all the buck voltages. Continued radio-chip testing with Jonah.

Week 11: (Nov 19 - Nov 25)
B: Researched interface requirements between BF707 and display - brainstormed potential header dimensions. Finished power path and connections to radio chip on schematic.
J, G: Added pull-up resistor to I2C clock. Soldered pins to fix potential AM connection issues. Replaced AM antenna and tried to adjust tuning capacitor within the radio chip. Discovered low signal/noise ratio.
G: Added Digikey/Mouser part numbers for all the passives on the schematic, and worked on the schematic entry for the new headphone driver from Texas Instruments. Added LM3658 charging chip, detailed GPIO connections to the LCD display, and 3.5mm jacks (a second jack was added to serve as line input).

Week 12: (Nov 26 - Dec 2)
B: Updated cost estimate - final Fall 2014 cost estimate was $49.61. Looked into maximum current draw through a USB interface
B, G: Fixed schematic errors and warnings with Professor Patterson. Added JTAG header and updated some resistor and capacitor parts based on Digikey availability.

J: Finished Bluetooth module connection code. Finalized general software algorithm for the user interface. Worked on software response to each button input. Determined which characters on the LCD display will be used for which part of the display interface.
G: Added "on" button and line in to device sketch, and updated display with expected display format. Updated the block diagram with inputs, and color-coded based on section of the schematic.

Week 13: (Dec 3 - Dec 9)
B, G, J: Finalized hardware and software schedules for continued progress in Spring 2015.
B, G: Completed schematic errors and warnings after review with Professor Patterson.
J: Continued debugging the breadboarded circuit for the radio IC

Weeks 14, 15, 16, 17, 18: (Dec 10 - Jan 17)
Finals Period and Winter Break

Week 19: (Jan 18 - Jan 24)
B, G: Worked on schematic entry based on edits from Professor Patterson over winter break.
B, G, J: Met as a team to discuss an informal division of tasks and a timeline for the spring semester. G decided to focus on the remainder of schematic entry and then help J with the hardware aspects of implementation of the Si4730 radio-IC evaluation board designed by Professor Patterson over winter break. B had the most layout experience and agreed to work on the layout of the board.

Week 20: (Jan 25 - Jan 31)
G: Completed changes from first round of review of schematic entry
J: Worked on coding general design flow

Week 21: (Feb 1 - Feb 7)
B, G: Sent the most recent schematic archive to Professor Patterson for a second design review, and received feedback.
J: Worked on coding general design flow

Week 22: (Feb 8 - Feb 14)
B, G: Updated schematics based on additional feedback from Professor Patterson
G, J: LCD code testing began, turned into debugging the breakout board pins

Week 23: (Feb 15 - Feb 21)
B, G: Began layout of the board, particularly exporting parts and importing decals
J: LCD testing completed

Week 24: (Feb 22 - Feb 38)
B, G: Layout of the board continues
G, J: Radio board progress began (assembly of the board completed by Arpie)

Week 25: (Mar 1 - Mar 7)
B, G: Layout of the board continued, this time with a focus on revisiting incorrect decals and importing new decals after renaming parts on the schematic.
G, J: Continued debugging for the radio chip

Week 26: (Mar 8 - Mar 14)
B: Worked on the layout of the board including placement and initial routing.
G: Assisted with the layout and helped probe the radio board as needed by J.
J: Continued writing software for the radio chip

Week 27: (Mar 15 - Mar 21)
B, G: Worked with Professor Patterson to finalize parts of the layout.
G, J: Debugged radio board to prevent an intermittent reset problem due to a missing pull-up resistor on the Micrel IC

Week 28: (Mar 22 - Mar 28)
Spring Break (Professor Patterson completed layout and send board to *Advanced Circuits* for production.)

Week 29: (Mar 29 - Apr 4)
G: Made changes made to radio test board to ensure proper hardware functionality, specifically the Micrel timer IC.
G, J: continued to debug radio board

Week 30: (Apr 5 - Apr 11)
B, G: Began assembly of the board with Professor Patterson. B worked on A-side components and G worked on B-side (Blackfin + Bluetooth module)
G, J: continued to debug radio board

Week 31: (Apr 12 - Apr 18)
B, G: Continued assembly of the board with Professor Patterson
J: Continued to try to debug radio board and began looking at Bluetooth module integration

Week 32: (Apr 19 - Apr 25)
G, J: Successfully recorded output on the analog pins of the radio board with help from Professor Silverman. Moved on to Bluetooth UART.

Week 33: (Apr 26 - May 2)
B, G, J: Worked on individual portions of the report
B, G: Worked on the bring-up of the ADP1612 for a steady 5V power supply.
G: Finalized parts list to ensure all the parts were available and placed in the project's part box. Worked through revisions from course instructors and Arpie. Updated the missing resistive divider and soldered on power supply connections.

Week 34: (May 3 - May 9)
B, G, J: consolidated and worked on common parts of the ENGN1650 final report
B, G: worked with course instructors to debug the ADP5024 for steady 3.3V, 1.8V, and 1.1V power supplies.

4 - Component Selection and Design

4.1 - Bluetooth

In order to help value to the device, it was decided to add Bluetooth capabilities to the prototype. Previously, the only way for a user to listen to the radio delayer would be to plug in headphones or speaker cables into the device. Adding Bluetooth would allow for users to connect their radio delayer with their home entertainment system wirelessly. As Bluetooth has become a standard in many home entertainment systems, this would be a hopefully easy way to allow multiple users to enjoy the product at once.

Bluetooth describes the variety of protocols that provide methods for exchanging data over a short range. Bluetooth allows for multiple devices to be connected to each other at the same time but under certain profiles, it is more common for there to be a master/slave (source and sink) model.

The Bluetooth protocol stack is a system of layers that define how two devices communicate with each other. At the base level is the Link Management Protocol (LMP), which sets up and controls the radio link between the two devices. At the next level is the Logical Link Control and Adaption Protocol (L2CAP). This is used to multiplex between multiple logical connections when the devices are using multiple higher-level protocols. The L2CAP channel can either be in Streaming Mode (no retransmission) or in Enhanced Retransmission Mode depending on the amount of reliability needed. The third level in the system is the Service Discovery Protocol (SDP) which allows a device to discover other devices as well as what Bluetooth profiles the secondary device uses along with any configuration settings needed to communicate with that device. Beyond those three levels, the other layers of the stack are not mandatory but most also use the Radio Frequency Communications (RFCOMM) protocol which defines how a serial port is emulated.

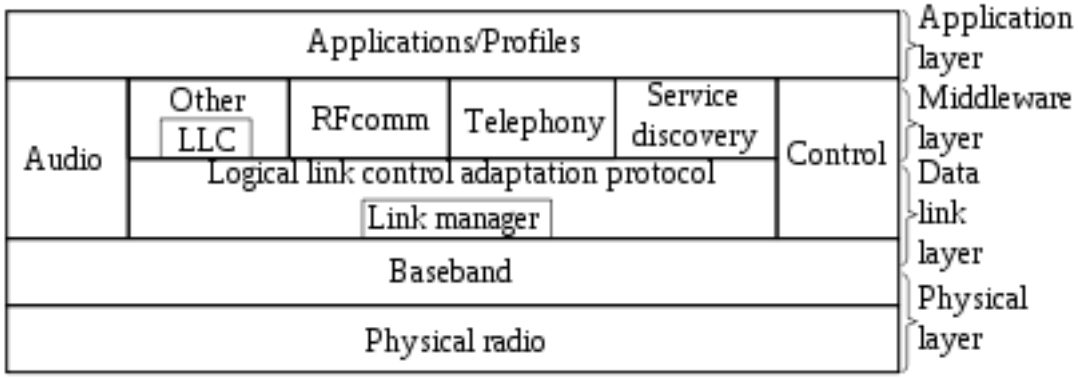

Figure 3: Bluetooth protocol stack

When two devices want to connect, one of the devices enters discoverable mode. In this mode, the device transmits its name, class, list of services and some technical information. This way, multiple devices can be differentiated from each other. Bluetooth devices also have unique 48

bit addresses but in most cases, a friendly name is set to allow for users to more accurately see what they want to connect to. Upon wanting to connect to a device, the devices begin to enter pairing mode. Under this mode, a shared secret is created to allow for the two to cryptographically authenticate the other. Most Bluetooth protocols require some form of encryption but it is not strictly necessary. There are two main types of pairing processes. The first is legacy pairing in which both devices require a user to input a 4 digit passcode. The other is Secure Simple Pairing (SSP), which can use a more advanced algorithm or even just allow direct pairing without any other user interaction. Next, there is a list of Bluetooth versions numbered 1-4. Each one offers additional features or energy specifications.

In addition, there are a variety of Bluetooth profiles. Each one defines a different method of sharing data. These range from ways of specifying how headsets, audio, image, health or even fax data is shared between devices. For our prototype, we needed to use the Advanced Audio Distribution Profile (A2DP). This profile defines how multimedia audio can be streamed from one device to another. They are designed to be a unidirectional transfer of audio data in either stereo or mono channels. In addition, the profile supports MP3 data.

For our prototype, it was decided that a full Bluetooth module would be used instead of making it from scratch. The idea was that the ease of use would vastly outweigh the small difference in price. However, there were a few complications that made searching for the right Bluetooth module difficult. For our device, we required that the radio delayer act as the source of the audio data. In addition, the profile to send audio data is A2DP. Many of the modules found either only were able to operate as the sink of data or did not support the A2DP profile. A2DP also only operates in the Bluetooth 2.0 specification so it was not possible to take advantage of Bluetooth 4.0 low energy capabilities.

After searching, it was found that the best module to use was the BTM511 chip from Laird. This device can operate as either a source or a sink and supported the A2DP profile needed. The module allows for commands to be sent over UART. In addition, data can be sent over a PCM, UART or analog input. For the purposes of the device, it was decided that the audio data would be communicated over an analog signal, as the radio delayer does not change the individual bytes but only changes the time they occur. This would also allow for the data to be sent to both the headphone driver and Bluetooth device easily with not much additional code. In addition, the BTM511 offers a variety of GPIO pins that could be used for test purposes.

4.2 - Radio Chip

The decision to have an on-board radio added an immense degree of complexity to the product, but was a decision made with the ultimate goal of providing a better user experience-instead of having to connect an external radio and worry about a jumble of wires, users could simply tune into, and preset, AM/FM stations using the LCD display.

Since radio-frequency circuits have very specific parameters for noise, capacitance and tuning, and ESD protection, a radio-on-IC approach was the only solution. While Broadcom, Icom, Onsemi (LA1837M) and ST were all possible candidates, Silicon Labs' Si473x series provided an antenna-input to audio-output radio IC with AM/FM functionality and analog and digital

output modes. The part is low-footprint, designed specifically for portable electronic equipment, low-noise (on-board LNA) and fit well in the cost constraints[5].

As seen in Figure 4[6], the Si4730 hosts on-board ADCs and DACs to provide digital and analog output modes. The FM antenna can be achieved using a headphone jack. The chip communicates with the microprocessor (Blackfin) over I2C/TWI. In terms of the analog output mode, the LOUT and ROUT pins are the left and right outputs of the stereo mode respectively, driven by high-fidelity stereo DACs. This is the mode used in the actual product. However, the chip also provides communication over TDM/I2S at either 32, 44.1 or 48kHz speeds in DSP mode, left justified, or I2S audio data format. 8, 16, 20, or 24 bit modes can be used. The reference clock and crystal oscillator both have a typical frequency of 32.768kHz, and a soft mute is also available.

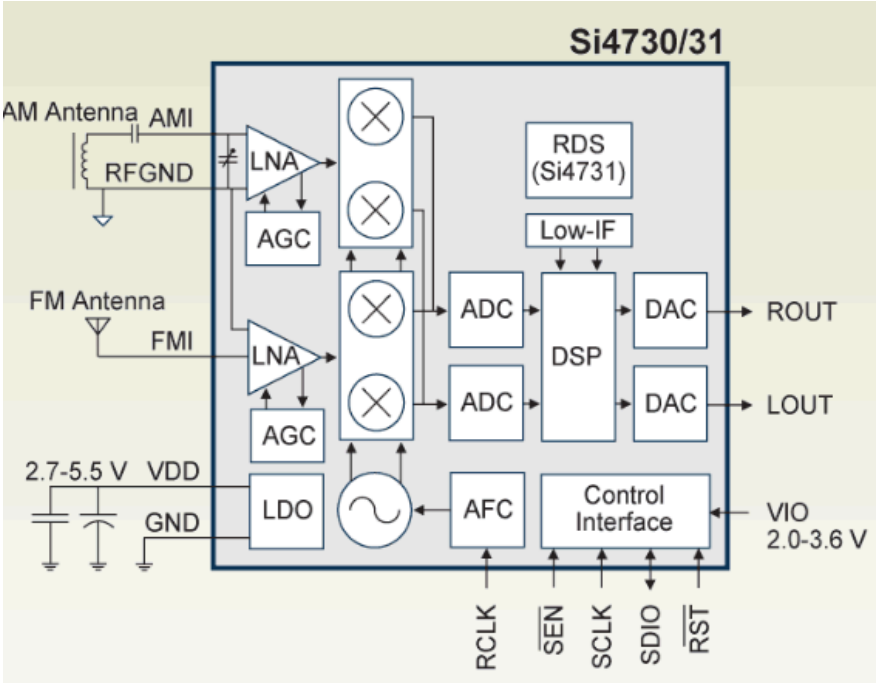

Figure 4: A block diagram of the Si4730

The figure on the following page shows the typical application schematic (as per datasheet)

[5] http://www.silabs.com/products/audio/digital-radio/Pages/default.aspx
[6] http://www.silabs.com/products/audio/fm-am-receiver/Pages/si473031.aspx

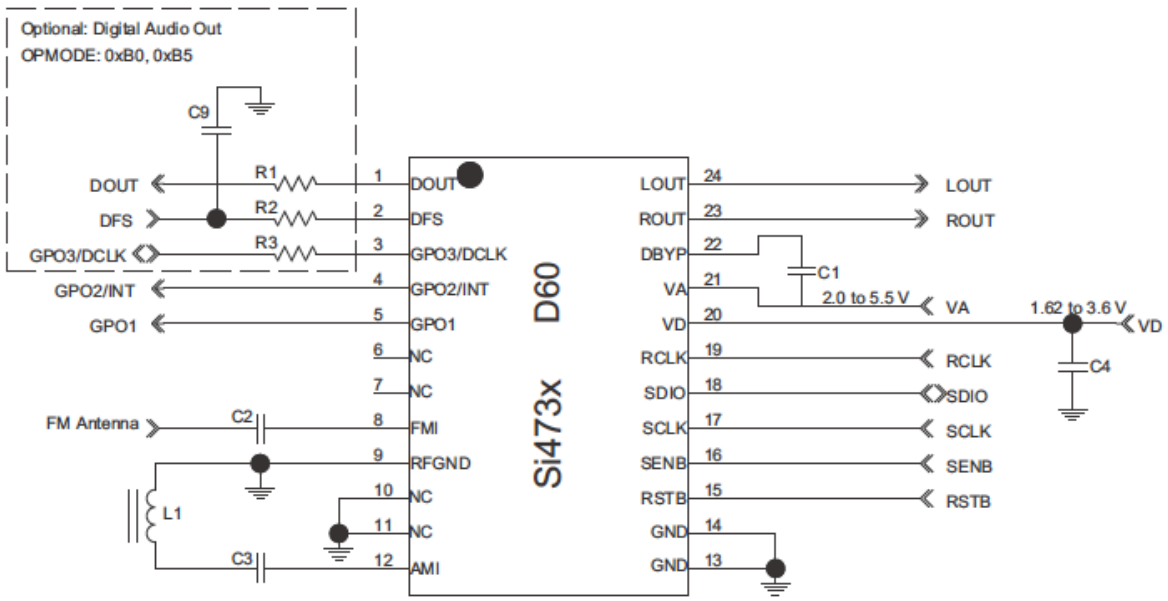

Figure 5: Typical application schematic for the Si4730

4.3 - Microprocessor

There were many factors that went into the processor design for the radio delayer. One of the major limitations in the type of processor that would be used is the amount of memory required to delay radio data. Without taking into account compression algorithms, the following is an equation relating the amount of storage needed for a given maximum delay time.

$$Memory = \frac{samples}{second}\frac{bits}{samples}MaxDelay \qquad \textbf{[Equation 1]}$$

Luckily, it is not necessary to go all the way 44.1kHz sampling speed and a much slower rate can be used. However, even with minimizing sampling speed and limiting the amount of delay that users can use, over 1MB of space is needed. In addition, though negligible in comparison, it would also be nice for users to be able to store their presets in between uses. Not many inexpensive processors have that amount of internal memory. While it would be possible to use an external SRAM or DRAM chip, this can potentially add a lot of complications to the prototype. It would then become necessary to coordinate between the two chips with very accurate timing in order to relay the information back to the user. In addition, this would require that the processor has some type of built-in external memory controller or else the code would become much more complicated.

The next bit of complication is the wide variety of devices that the processor must be able to communicate. With all of these devices, there are also a large number of pins that need to be used. After considering all of these requirements, the processor that was chosen was the BF707

chip from Analog Devices. With internal SRAM, it would not longer be required for a DRAM controller to be coded.

The BF707 can operate at up to 400MHz so it can easily deal with interpreting information from the radio chip quickly. In addition, it has a large number of pins and it can support I2C, 2 UART connections, 2 SPORT (with I2S) connections and a total of 184 pins. This means that it can easily interact with all of the devices that we need. In addition, it has an internal 4-channel, 12 bit ADC to deal with analog input from the radio chip if needed.

What is the most impressive about the chip is the amount of internal memory it has. Along with some non-volatile memory, it has three levels of cache. This includes 1MB of on-chip L2 SRAM. This removes the need for any external memory. However, this does somewhat limit the amount that the user can delay.

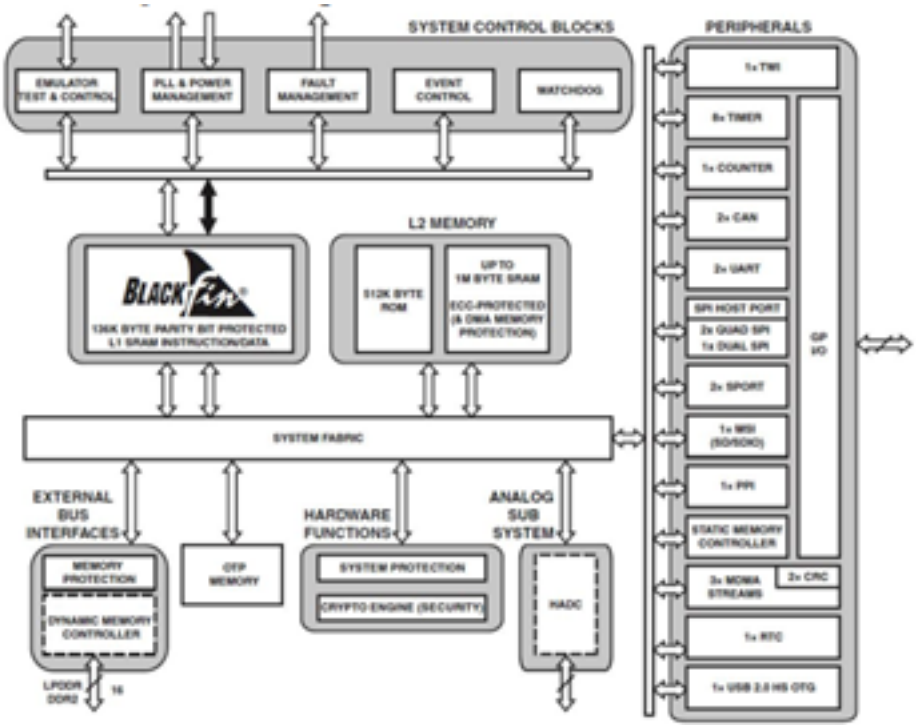

Figure 6: A block diagram of the BF707

Lastly, the biggest downside to using the BF707 is that it was only released fall 2014. This means that there is a very limited amount of support code for the device and that most of the code was the first foray for the team. However, since it is in the BF family, there were a lot of similarities in its capabilities that allowed older model code to be used.

4.4 - LCD Display

For the LCD screen, it was only important that all of the necessary information be displayed to the user. Some helpful information for the user includes: if the radio is on AM or FM, the

frequency it is tuned to, the volume setting, the mode, the delay and the preset information. In addition, if Bluetooth is on, some portion of the friendly name should also be displayed to the user. It was decided that 2 lines of 16 characters would be sufficient for this goal. For example, volume can be reduced to 1 character, frequency can be 4 characters, AM/FM is 2, delay is 3, preset is 1 and mode is 1. Whatever space is left over can be used to display part of the friendly Bluetooth name when it is connected. As most inexpensive LCD displays communicate over common GPIOs, no specialized software or peripherals are needed from the processor. The LCD that was eventually chosen was the LCM-S01602DTR-M module from Lumex Inc.

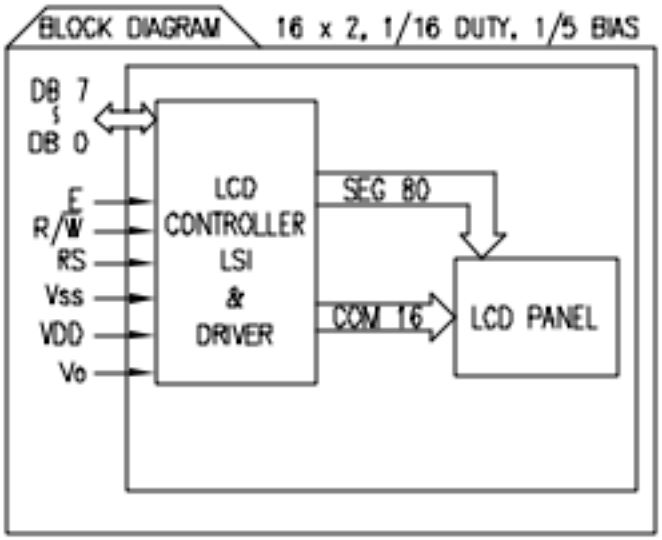

Figure 7: A block diagram of the LCD display

PIN CONFIGURATION

PIN NO.	SYMBOL	LEVEL	FUNCTION	
1	Vss	–	POWER SUPPLY	GND (0V)
2	V_{DD}	–		5V
3	Vo	–		FOR LCD DRIVE
4	RS	H/L	REGISTER SELECT SIGNAL H: DATA INPUT L: INSTRUCTION INPUT	
5	R/$\overline{W}$	H/L	H: DATA READ (MODULE-->MPU) L: DATA WRITE (MODULE<--MPU)	
6	E	H,H->L	ENABLE	
7~14	DB0~DB7	H/L	DATA BUS-SOFTWARE SELECTABLE 4 OR 8 BIT MODE.	
15	NC	–	NO CONNECTION	
16	NC	–	NO CONNECTION	

Figure 8: Pin configuration for the LCD display

The LCD communicates over 11 wires: 8 are for the data bus so that 1 byte can be sent in parallel. The other 3 wires are for enabling the chip, sending instructions and a RW bit. The module itself costs only $4.11 at 1000 pc quantity and was one of the cheapest available for the size of the screen.

4.5 - Rechargeable Battery and Charging Chip

Given that the product was designed to be pocket-sized and easily portable, it had to be battery run. It was decided to use rechargeable lithium ion batteries to achieve two purposes: first, this prevents additional cost and frustration during use, and second, it eliminates the user from having to remove the battery and thus eliminates complexities in the design of the case. Based on prior years' projects, the UBP002 from Ultralife Corporation was chosen. For the charging chip, the LM3658 was chosen since it provided both USB (5V) and wallwort capabilities.

Figure 9: The UBP002 battery

Based on our power budget seen in Section 4.8, 800mAh was deemed sufficient. The UBP002 is a compact battery providing average voltage of 3.7V and 900mAh designed specifically for portable electronics. It can be used for hundreds of cycles before an appreciable loss in capacity (Ultralife claims "> 500 cycles before the cycle life drops to 80% of initial capacity"[7]), and provides two 24AWG wires for the positive and negative terminals.

The figure below shows the performance graphs of the UBP002, as described in the datasheet.

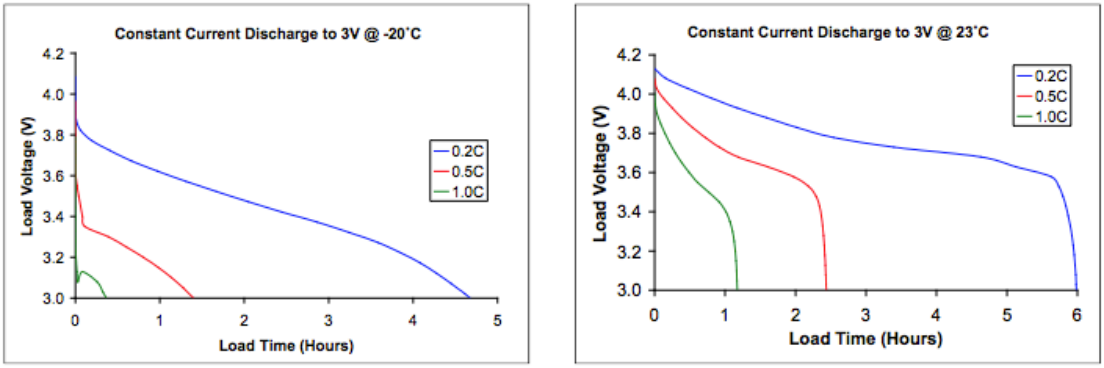

Figure 10: Load voltage versus load time for the UBP002

[7] http://m.ultralifecorporation.com/be-commercial/products/rechargeable/ubp002/

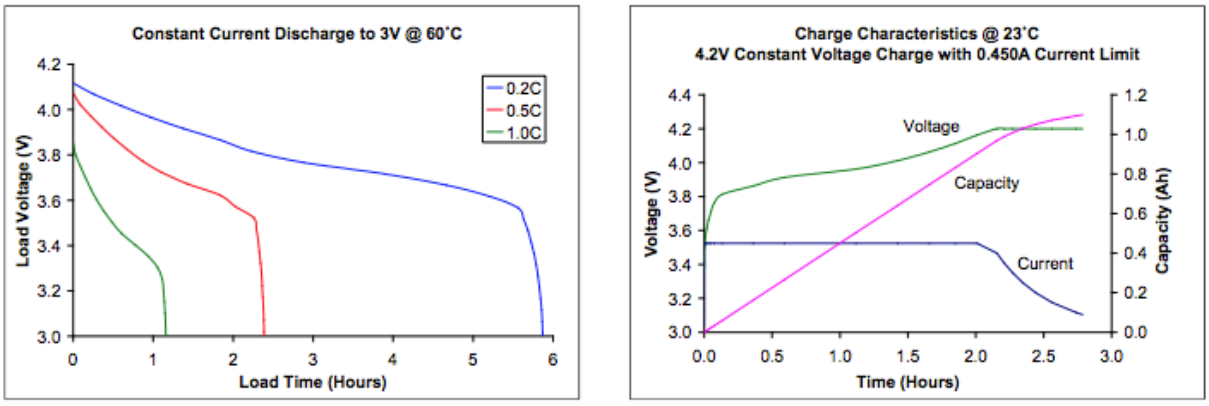

Figure 11: Voltage, time, and current plots for the UBP002

An additional IC was also needed to charge the lithium-ion battery from the external USB power. To this end, the LM3658 chip was chosen because of its small footprint and ease of use. Its pins were used in the following way:

- CHG_IN pin was connected to a test point in the case where a wallwort connector would be used.
- USBPWR pin was connected to the VCC pin of the USB receptacle.
- USB_SEL was tied to VCC, so that when USB power is connected, it is allowed to draw up to 500 mA, instead of the 100 mA lower limit.
- EN_B was tied LOW to enable the charger.
- STAT1 & STAT2 were connected to LEDs to indicate charging status (described below)
- Ts was connected to the negative terminal of the battery connector.
- BATT was connected to the positive terminal of the battery connector.
- LEDs are used in our design to indicate four different charging statuses: STAT1 LOW (LED on) indicates “charging”, STAT2 LOW (LED on) indicates “done charging”, both HIGH (both LEDs off) indicates power-down, and both LOW (both LEDs on) indicates a battery issue.

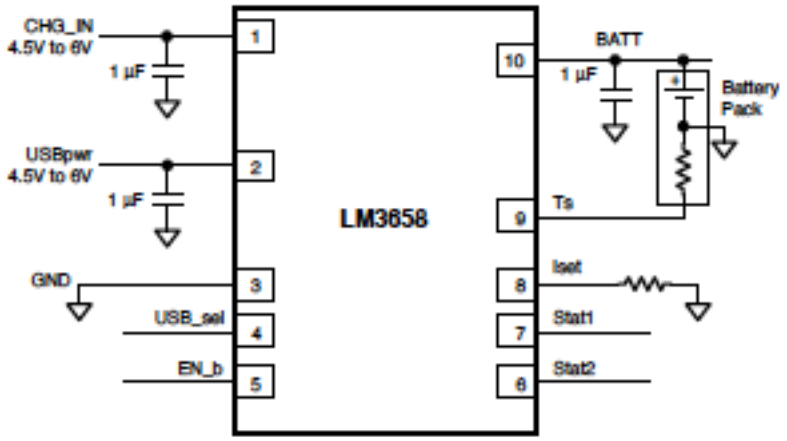

Figure 12: Typical application circuit for the LM3658

It allows USB charge currents of 500 mA (standard limit) when in USB power mode, but also allows a wall wart connection through an AC power adapter. Other features include continuous battery temperature monitoring and power FET regulation, and 2 open-drain outputs for LEDs to indicate status of charging. It offer 5 modes of operation: pre-qualification, near-depleted Battery preconditioning mode, constant-current mode, constant-voltage sleep Mode with Ultra Low Quiescent Current mode, top-off mode and maintenance mode.
The LM3658 uses thermally-regulated FETs to specify optimal charging of the battery. In all cases, the charge current is limited by the value of R_{ISET} given by $\frac{K_{ISET}}{I_{CHG}}$, the USB selection of 100mA or 500mA, or the junction temperature. The charge current is thus a function of the thermal conductivity of the package and charge settings as seen below. T_A is the ambient temperature.

$$I_{CHG} = min\left[\frac{K_{ISET}}{R_{ISET}}\ OR\ USB_{SEL}\ OR\ \frac{120°C - T_A}{\theta_{JA}\,(V_{CC} - V_{BATT})}\right] \qquad \text{[Equation 2]}$$

4.6 - USB-Micro B

In order to charge the battery, we decided to include a USB-Micro B interface. Through USB, 1 Amp can be routinely delivered from a wall connection or a computer, and often up to 1.5 Amps (although the LM3658 limits the USB current draw to 500 mA). This interface also has the benefit of being very commonplace, as all android phones and many other devices are charged through USB-Micro. In addition, the receptacle for this connection is physically much smaller than the USB-Mini alternative.

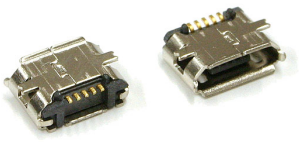

Figure 13: A view of the USB-Micro B receptacle

This USB interface has the following pins: VCC, D+, D-, SENSE, and GND. VCC provided positive voltage, and SENSE and GND were connected to the board's ground. D+ and D- could be used to transfer data, but for the purposes of this project, these pins were left unconnected. A 0.1uF capacitor was used to bypass the voltage provided from USB power.

Because the battery's recommended charge rate is 450 mA (while allowing up to 900 mA), delivering 500 mA through USB is the easiest and most effective way to charge the battery. Through USB power, the battery would charge within two hours by delivering up to 500 mA at the recommended charge voltage of 4.2 V (from the LM3658).

4.7 - Headphone Driver

In order to hear analog audio data directly from a headphone connection to a 3.5mm jack, a headphone driver was needed, and the TPA6130A2 part was chosen for this design. This part allows for stereo audio output, and has volume control capability over I2C. This part was particularly attractive because of the small peripheral component requirement. Acccording to the specification, this part will deliver 300 mW of power into 16 Ω headphones (and can also support higher impedance headphones).

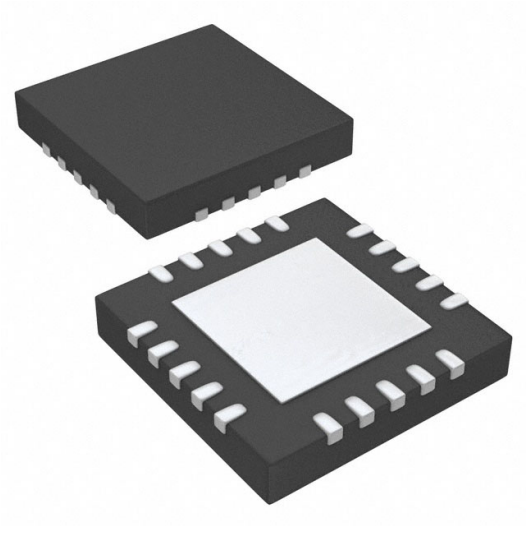

Figure 12: The TPA6130A2 headphone driver

4.8 – Power Circuitry

		I draw	Total	Power = VI	Notes
		mA	mA	mW	
Parts powered at 5V					
LCD display	LCM-S01602DTR/M	3	3	15	
Charging chip	LM3658	100		-	Doesn't draw from battery
Buck + Buck + LDO	ADP5024			69.00	Assumes n = 85%
Boost	ADP1612			81.17	Assumes n = 85%
Parts powered at 3.3V					
Bluetooth Module	BTM511	60			
RF radio chip	Si4730 (FM & AM)	17	106.27	350.69	
Headphone driver	TPA6130A2	6			
Oscillator chip	NZ2520SB-32.768KHZ	0.26			(10uA standby)
Oscillator chip	ECS-2025-2033	13			
Micro-controller	BF707	10			TBD, but estimated 10mA
ESD charging chip	CM1214-01ST	0.01			
Parts powered at 1.8V					
Micro-controller	BF707	1	1	1.8	Value is TBD, but low because
					DRAM controller is not used
Parts powered at 1.1V					
Micro-controller	BF707	35	35	38.5	90mA worst case
				total power (mw)	**556.16**

Table 2: Calculations for the total power drawn by the product

There are 4 supply voltage levels required for this product: 5V, 3.3V, 1.8V, and 1.1V. The 5V is used for the LCD display. The 3.3V powers most of the peripheral components including radio, Bluetooth module, and switches. The 1.8V and 1.1V are required for the BF707 microprocessor (1.8V is required to be supplied even though the external-DRAM controller is not being used). All 4 voltages need to be supplied from a single lithium-ion battery, which provides an output voltage around 3.7V (range is 3.0V and 4.2V). It is thus necessary to both boost and buck the battery voltage at different stages of the power flow.

The general approach was to use a boost regulator to achieve a 5V output from the battery, and then buck down from the stable 5V to 3.3V, 1.8V, and 1.1V. Instead of using 3 separate buck controllers or LDOs, however, a single integrated solution from Analog Devices was used.

The ADP1612 from Analog Devices is a DC-DC step up converter with a 1.2A limit. With a minimum input voltage of 1.8V (suitable for the lithium battery output), it can boost up to 20V. It is available in an 8-pin MSOP at $1.07 (in 1000s)

The figure below shows an application circuit for the AD1612 as used in a boost converter with V_{out}= 5V. The load current for this product is less (in mA), and so much smaller diodes/inductors were required.

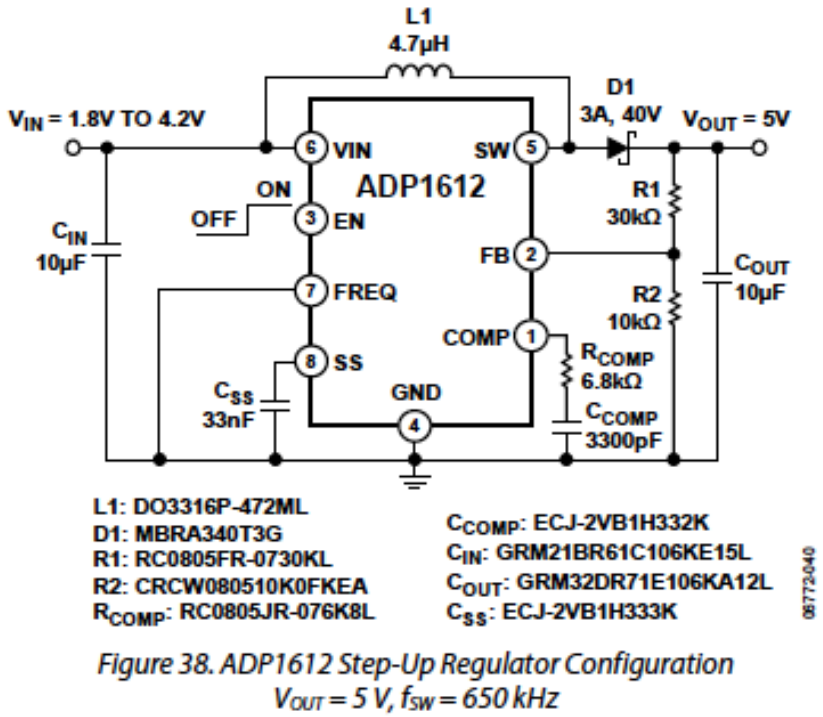

Figure 13: Typical Application Circuit for the ADP1612

The diode, inductor, and resistor divider was computed as per guidelines in the datasheet[8]

The ADP2504 is a 24-lead QFN hosting 2 (1200mA each) buck regulators and 1 LDO. This part was used to generate stable supplies of 3.3V, 1.8V, and 1.1V. At $2.91 (in 1000s) it is more cost effective, and efficient in terms of space, than separate regulator ICs. It is important to note that it is the 1.8V output, and not the 1.1V output, that is tapped from the LDO due to the minimal current draw required (the DRAM-controller requires 1.8V but is held in reset in this product).

[8] http://www.analog.com/media/en/technical-documentation/data-sheets/ADP1612_1613.pdf

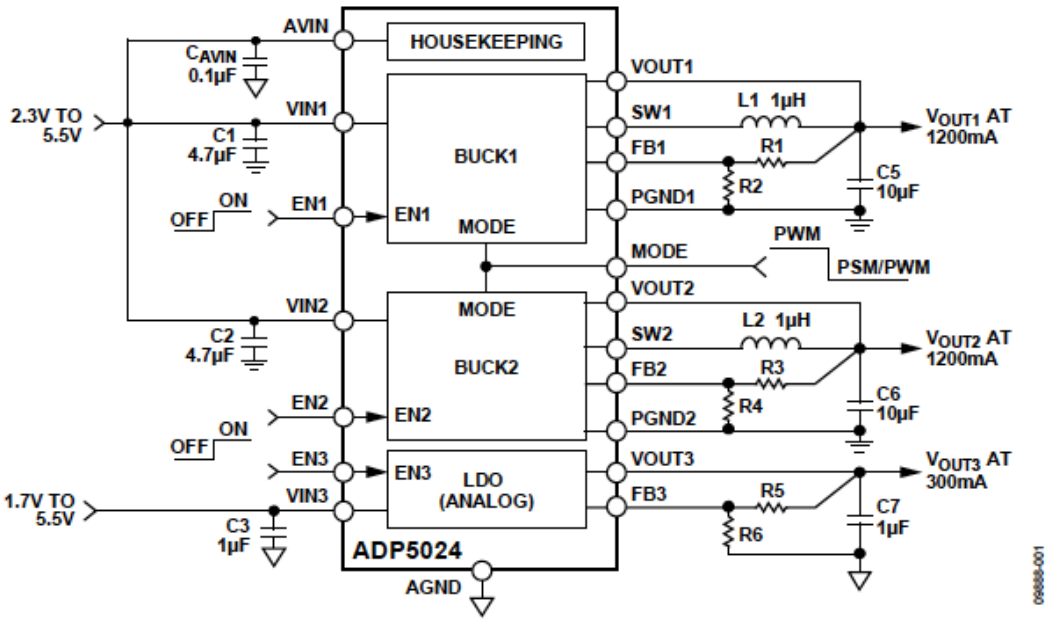

Figure 14: Typical Application Circuit for the ADP5024

The flow is thus as follows:

- 5V output from the AD1612 is fed into the Vin of the first buck, producing 3.3V
- The 3.3V output from the first buck is fed into the Vin of the second buck controller, producing 1.1V
- The same 3.3V output is fed into the input of the LDO, producing 1.8V

The design portion of the IC involved computing the values for the resistive dividers (since V_{out} is adjustable). The values for inductor and capacitor are as suggested in the datasheet[9]

[9] http://www.analog.com/media/en/technical-documentation/data-sheets/ADP5024.pdf

BUCK 1			Vout	3.3	V
			Vin	5	V
10uF Cout			VFB	0.5	V
1uH inductor on SW1					
R1	12	kohm(guess value)			
R2	2.14	kohm			
BUCK 2			Vout	1.1	V
			Vin	5	V
10uF Cout			VFB	0.5	V
1uH inductor on SW1					
R1	2.2	kohm(guess value)			
R2	1.83	kohm			
LDO			Vout	1.8	V
			Vin	3.3	V
1uF Cout			VFB	0.5	V
R1	2.2	kohm(guess value)			
R2	0.85	kohm			

Table 3: Peripheral Component Value Selection for ADP5024

4.9 - 3.5mm Jack

The audio jack is used to play out the analog output of the BF707 (delayed according to the user's input via LCD display) after a headphone driver. A 3 conductor, 3 contact solution from CUI Inc. was used ($.81 at 500pc. quantity). Future spins of the board could allow users to connect an external audio source using another 3.5mm audio jack.

5 - Hardware Implementation and Testing

5.1 - First Radio Breadboard

There were two iterations of testing. In the first iteration, the chip was simply soldered to pins on a breadboard and the circuit was built by hand.

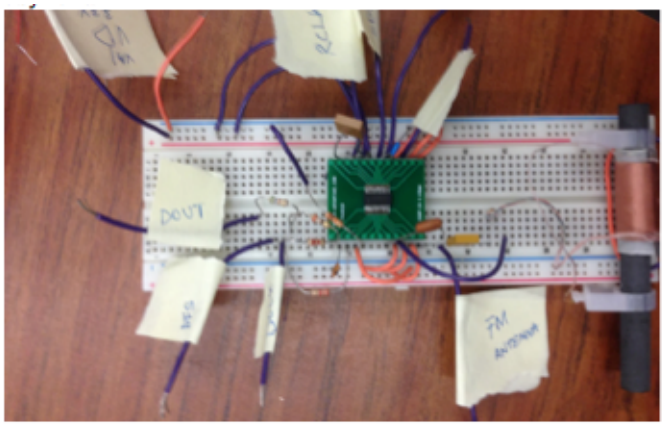

Figure 15: View of the First Radio Breadboard Circuit

In order to reduce the amount of testing needed, it was decided that analog output would be used. In order to use digital output, it would become necessary to also test a data clock from the processor as well as a frame sync line. In addition, since there wasn't an FM antenna available, instead only the AM receiver mode would be tested with the ferrite loop stick.

The hardware circuitry was fairly straightforward and it followed the circuit diagram in the manual exactly. Thus, the first thing that was tested was the ability to power on the chip. According to the software manual, there would be a power-up command that needed to be sent over I2C in order to move the chip from the power down state. When the chip receives a command, in addition to the response it also sends a status byte. However, the chip is not guaranteed to respond exactly when it gets a command. Instead, it was found that the chip must be continuously polled until it responded with the status byte. A status byte of 0x80 meant that the next command was clear to send. The third bit could also be set in order to indicate to the user that there was an error (0xC0).

Once the chip was responding to the power up command, the next step was to check the revision information. This would be to make sure that the firmware on the chip was loaded with the expected values.

Property	Returned Hex	Returned Decimal
Part Number	0x1E	30
Firmware Revision	0x3630	6.0
Patch ID	0x0000	0x0000
Component Revision	0x3730	7.0
Chip Revision	0x44	68

Table 4: Return Values of Get_Revision on Radio Chip

Now that the chip seemed to be responding correctly, the next step was to attempt to get an audio output from the chip. In order to accomplish this, silicon labs offers both the ability to allow the chip to have default values as well as the ability to get and set each property individually. In addition, the programming reference offers a set of example code to program the chip as an AM receiver as well as the state table for the chip in this state.

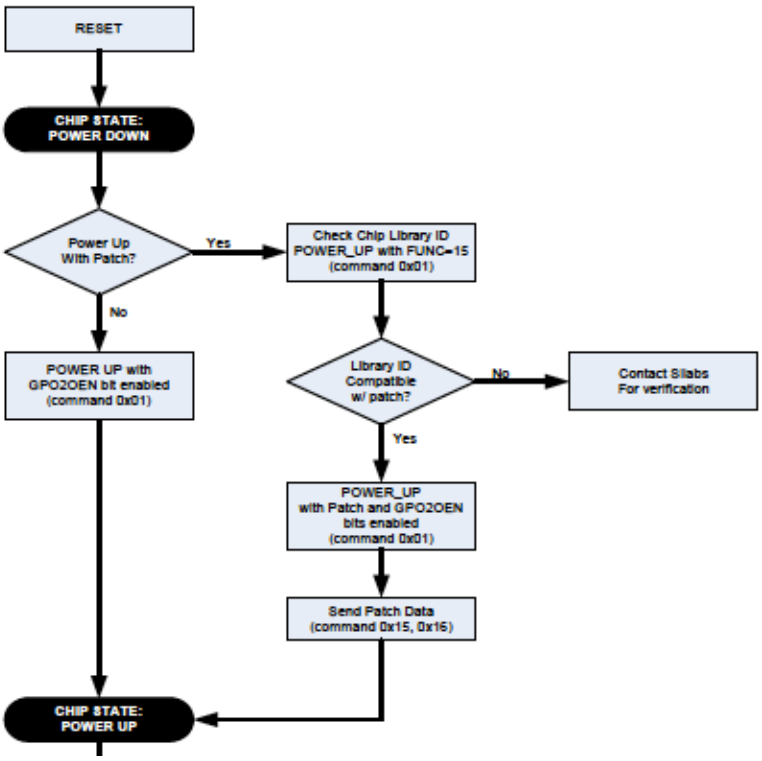

Figure 16: Flowchart of AM Radio Code

It was decided that in order to more accurately control the chip, each property would be set individually. For each property, the only response is 0x80 so the property must be read with the get property command in order to confirm the change. Errors started to arise though when radio

stations were attempted to be set. The first bit of error is that in addition to the delay in waiting for the radio chip to be ready for the next command, there is also a bit of delay when the radio sets its station. However, during this period of time, the radio chip is still able to receive more commands. Instead, an additional bit is set when the station is set. Thus, a polling command needs to be used until the radio chip has set the station correctly.

The next piece of error occurred when the data lines were accessed. In this iteration of the board, no data could be read from the radio chip. When the AM radio station information is queries, the radio chip responds with some basic diagnostic information. With this, the issue became clear. The radio chip responds with the frequency the chip it is set to, if the channel is valid, the RSSI value, the SNR value and the value that the tuning capacitor is set to. For every AM radio station that was tried, a SNR of 0 was returned. This caused the chip to go into a soft mute state and report that the AFC was railed. Even by turning off the soft mute, there was still a value of 0 for the SNR.

At this point, it was unclear what the source of the error was. A handheld radio was used to determine what AM radio signals would be stronger in the testing room but this wasn't helpful. In addition, a variety of AM tuning ferrite loops were tested but nothing changed the result. After trying out a variety of tests, it was determined that time would be better spent moving on from that basic breadboard design and assume the issue to be in either the soldering of the chip (thus causing a short in the circuit) or in the ferrite loop stick. This caused a second design to be looked into that had the ability to plug in headphones and thus allow for an FM antenna in order for a wide range of testing. In addition, the second circuit was to be designed to fit on an evaluation board instead of a simple breadboard in order to get a more reliable design.

5.2 - Schematic Entry

The figure below shows a block diagram of the product (the schematic can be found in the Appendix)

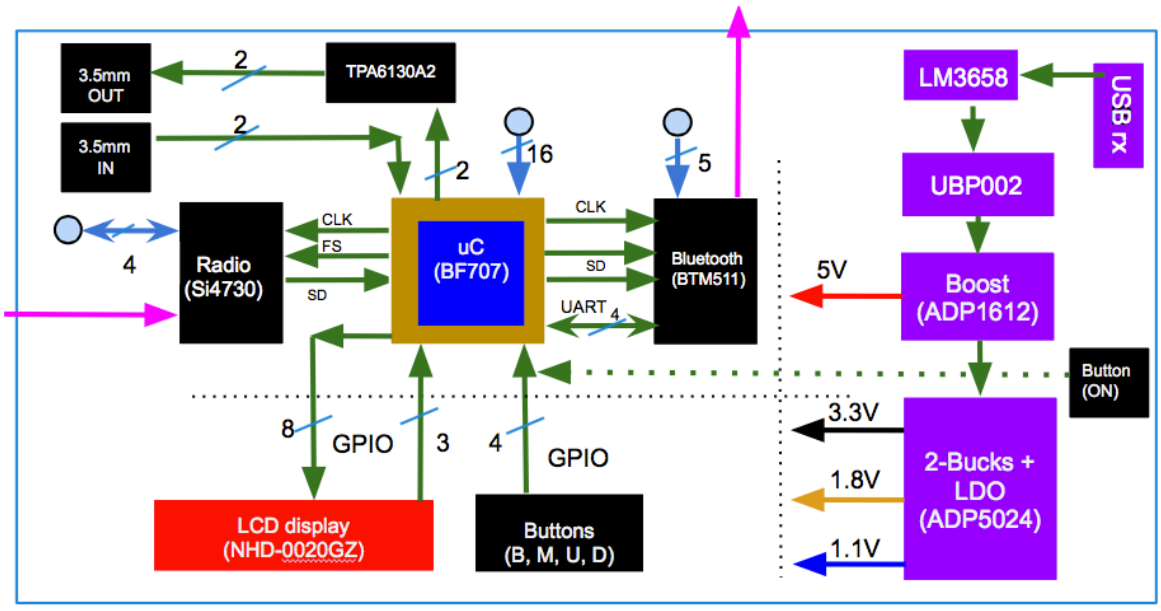

Figure 17: Block Diagram of the product showing the 4 different voltage supplies used

5.2.1 - Debug Capabilities

An important part of the schematic entry stage was the addition of debugging tools to provide access to important signals, and also to isolate the different power supply blocks from each another for protection during initial testing.

5.2.1.1 - Zero Ω Resistors

Zero Ω resistors were an important part of schematic entry for the power path. Given that the power flows from battery to boost converter to buck/LDO IC, it was crucial to place Zero Ω resistors in series with the various V_{out} pins – specifically after the boost controller, and after each output of the combined buck/LDO controller. By not populating these resistors, there was no risk of damage to the entire board during bring-up. Instead, each voltage supply could be brought up and tested for correct voltage and stability before the Zero Ω provided a 'path' to the next IC.

5.2.1.2 - Test Points

Test points were used frequently as a debugging tool. Test points were placed on important pins of the microcontroller (including instances, such as external oscillator pins, that might be required if on-board components failed), Bluetooth module, radio chip, outputs of each voltage regulator, and ground. Using test points to test unused pins on the ICs was strongly recommended by the instructors, and was important for the 184-pin BGA which is impossible to probe with logic analyzer or oscilloscope.

Large test-points were used in case solder connections might be required during debugging, particularly because ample space was available on the PCB.

5.2.1.3 - Bluetooth Module

GPIOs and UART *TX, RX, and CTS/RTS* from the BTM511 were routed to large test points as a safeguard against instances where programming the Bluetooth module proved to be difficult or error-prone.

5.2.1.4 - Jumper for External Radio

Test points were used (instead of zero Ω resistors) on the data lines connecting the radio chip to the BF707. In this way, an external radio can be attached in the case that the Si4730 integrated radio failed to provide reliable output.

5.3 - BlackFin Evaluation Board

The BF707 evaluation board came with a variety of peripherals in order to facilitate testing. The board included test pins, push buttons, and a breakout board for the many processor pins. In order to make sure that the basic capabilities of the board worked, all of these features were

tested. Lastly, the development board also included a direct USB-micro port back to the computer in order to easily program the chip.

The first bit of testing was to ensure that the UART was working. Luckily, most of this code was easily ported from previous iterations of the Blackfin processor. This bit of testing code was essential for all subsequent testing of hardware and software. With the addition of UART code, it was then possible for the Blackfin to talk directly back to the computer. With the aide of a *combasic* program, this output could then be displayed. This allowed for the emulation of a *printf* command and thus debugging information could then be printed.

In order to test that it was actually the processor communicating with the computer, an extra bit of test code was added. When a user types a letter on the keyboard, the program sends it to the processor, which then increments the character and sends it back. There were also test strings that were added for when a user pressed a specific character. At this time, each of the buttons as also tested. Code was added so that a simple test message would be outputted to the *combasic* terminal upon pressing and releasing a button.

The next part of testing was to test the breakout board. As it was later seen during the LCD testing, there seemed to be a few issues with the breakout board. This first thing that was observed was that not every GPIO pin was directed to the breakout board (this was seen directly from the data sheet). By not having every pin brought out to a test point, this would severely hinder the amount of concurrent testing that could be done during the debugging stages. In addition, it was shown that even with the limited subset of pins that were brought out, some of the pins on the breakout board did not work. This proved to be very frustrating when trying to test devices with a large number of pins that needed to be connected as the code needed to be segmented across the output ports.

Another huge problem with this was that in the software for the processor, there were 3 sets of 16 pins labeled ports A, B and C. Not only were the pins on the breakout board that were not necessarily in any port order, but the pins that were broken were also not consecutive. This meant that the code needed to work around this, something that became very difficult to write to as it had to stretch weirdly across the many ports. The last issue with the breakout board was that not all of the GPIOs were broken out and many of the pins on the breakout board were reserved for TWI or internal registers on the Blackfin processor. In order to see what pins were functional, a test program was required to probe each pin for a signal.

The last part of initial testing was for the internal memory. In order to demonstrate this, a simple large array was made. The idea was that this array would be too large to fit on the stack and be forced into internal SRAM memory. Initial values were placed in the table and mathematical operations were performed on those values. This proved that there wasn't some preprocessing that was going on that cut out the internal memory.

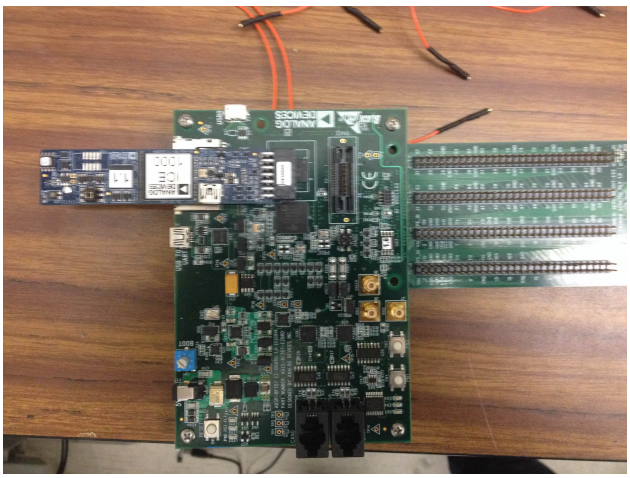

Figure 18: A View of the Blackfin Development Board

5.4 - LCD Display

The testing of the LCD device was straightforward. To operate the LCD, its enable bit must be set. There is also a small amount of set up support code to set internal properties. Lastly, the LCD can be programmed by first sending a command indicating which of the two lines to be changed and then sending a stream of bytes to the LCD with the write enabled line set.

While this was all easy in principle, there turned out to be a few roadblocks. The first was the issue with the broken pins as explained in the previous section. Since the LCD had 11 wires that all needed to be connected to the Blackfin processor (i.e. not running on either power or ground), it took a great deal of effort in order to properly arrange the wires once the issue of broken pins was identified. The next roadblock was to figure out how the tuning resistor that set the contrast of the display worked. When using a simple resistor pot, it was found that slight perturbations to the wires would throw the contrast and render the characters invisible.

Figure 19: A Working LCD Display

5.5 - Second Radio Board

The mostly likely failure of the first radio chip was some error in either soldering the small chip or in the wiring of the breadboard. In addition, in the breadboard design, there was no way to start the radio chip in FM mode. In order to combat these issues and hopefully get the radio chip working, it was decided that a developmental board be made. Professor Patterson agreed to design this board and send it for production over winter break to ensure no delays in the Spring semester timeline. On this development board, there were many test pins in order to easily probe signals, the addition of a headphone jack and a set of jumper pins for easy connection to the Blackfin processor. The only additional peripheral that needed to be soldered on was the ferrite loop from the original breadboard, which was done by adding pins to the test points on the board.

Once the board was connected the same code as before was run. However, no audio signal could be probed on the device. After much of the same debugging as with the first radio chip, it was decided that it was possible that the error was in the AM radio part of the chip. If the ferrite loop were not reliable, it would make the most sense to try and bypass it with the FM antenna. On the software side, this was easy to do, as there are FM commands that are analog to each AM command. It was also possible that the issue was in the powering down of the chip in between trials. Thus, the test code was changed so that by pressing one of the development board buttons, the radio chip would be powered down.

When the chip was booted in FM mode, there was finally a successful response from the radio chip. Now, the RSSI and SNR reported for a radio station were well above 0. By having a larger SNR, the chip was able to recognize the station as valid and be brought out of soft mute mode. However, there was still no audio data being output from the LOUT or ROUT chips.

After much debugging, the only thing left to try was to replace the radio chip and see if it was a soldering issue or a blown output pin. This soldering was done very carefully with proper grounding. This appeared to fix the issue and audio data could finally be seen at the pins. Since there was no headphone driver on the board, it was not possible to listen to the data. However, work could now be done so as to read the data into the Blackfin and start processing it for output.

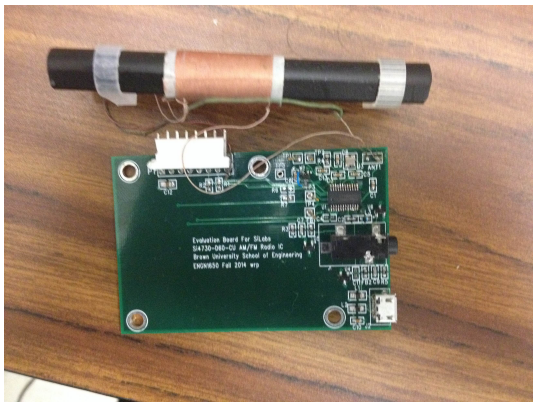

Figure 20: View of the Second Radio Board

5.6 - Board Layout

After schematic entry was finished, the process of board layout was started. The dimensions of our board were decided to be 14cm x 5cm, which was based on the size of the initial prototype that had been designed, saving room on all sides to fit into a casing. Further, it was decided that the board would have four layers: one on each side for routing and component placement (layers 1 & 4), a ground plane (layer 2), and a mixed power plane (layer 3).

Once the physical attributes of the board were decided, component placement and routing was done. The schematic was imported into the PADS Layout environment, which allowed for easy component placement and routing. Each major component (typically an IC) was separated out with all the peripherals that accompany it. Then, each one of these "modules" was placed in a different part of the board. Some initial considerations were the components that need to be on the edge of the board, such as the mounting holes, switches, 3.5mm jacks, USB receptacle, battery connector, and JTAG header. Additionally, the header for the display was placed on the top of the board and in a location such that it would be covering as few components as possible, and would line up with the hole cut into the box/casing for the product.

The BF707, BTM511, ADP5024, and the mechanical switches were all placed on layer 1 of the board, in addition to the peripheral components for each element. The JTAG header, battery connector, USB receptacle, 3.5mm jacks, Si4730, TPA6130A2 (headphone amplifier), and LM3658 were all placed on layer 4 of the board (with peripherals). Additionally, the potentiometer to control the display's contrast was placed on layer 4 so it could be physically close to the display connector without risk of being covered by the display.

Once all the components were placed on the board, they were adjusted to make sure specific peripherals were as close to major components as possible. The biggest challenge of this step was to lay out the numerous bypass capacitors for the BF707 in such a way that there was room around the BGA to ensure successful routing. Once this process was done, the power plane was drawn up distribute all the voltages (1.1 V, 1.8 V, 3.3 V, and 5 V) around the board in the most efficient way possible. Finally, the auto-router was used to rout all the connections around the board.

5.7 - Board Assembly

There were a number of challenges in the assembly of a board of this complexity level, but the most notable was the 184-pin BGA package of the BF707. To start, solder paste was applied to the bottom of the board through a mask. The first components to be placed were the Si4730, LM3658, and the TPA6130A2; once placed, these components were put through the oven and soldered down to the board successfully. Next, the other side of the board (the one containing the BGA) was done. The first time solder paste was applied, the mask wasn't lined up well enough to cover all pads of the BGA successfully. The second time, however, it was more successful, and the BGA was soldered in the oven in addition to the BTM511. Due to the lack of an x-ray or suitable equipment to observe the solder joints on each of the balls of the BGA, it is hoped that the oven allowed for proper contact between the BGA and the board.

After the major components were soldered in the oven, Arpie Kaloustian finished assembling the board by hand. A few components (such as 0Ω resistors) were not populated initially, in order to aid the debugging process for the power path.

5.8 - Board Bring-up

After assembly was finished, the board was brought up starting with the power supply. The zero zero Ω resistors were not populated, so that individual stages of the power supply could be tested separately. It was first determined that the 5 V supply on the board was working as expected, so the zero Ω resistor separating this supply from the buck/LDO regulator was populated.

Initially, there seemed to be no voltage output from the ADP5024. However, it was determined that input bypass capacitors were missing, and when these components were soldered, the power supply worked as expected. Once this was done, the zero Ω resistors between the supply and the downstream components were populated. At the time of this writing, the 3.3 V supply was being dragged down by components downstream, and work is continuing to get proper voltages out of the supply when a load is applied.

Additionally, a 16-pin ribbon cable was soldered to the LCD display, so the display could be tested without being soldered to the board (where it will cover up the main power supply).

5.9 - Board Testing

Board functionality will be tested in a holistic approach, most notably by testing interfaces with the BTM511 and Si4730. The BF707 will be programmed through JTAG with the software code.

6 - Software Implementation and Testing

6.1 - Initial Code

Beyond the various parts of the code to interface with the peripherals (i.e. radio chip and Bluetooth module), the BF707 is also responsible for the main code loop with which the user will interact. For this prototype, there is much to complicate the main program. The code can be thought of as a simple state machine for each mode. In order to reduce the number of buttons one set of up and down buttons would be used to control various settings such as volume, delay, frequency, etc. However, this means that there also needs to be a way for the user to switch what setting the buttons would be modifying. Therefore, an additional mode button was used to switch between these states. Lastly, there would also be a separate Bluetooth button in order to activate the Bluetooth module.

The user can cycle between the following modes by pressing the mode button: AM/FM, Station Frequency, Volume, Delay, and Preset. In addition, pressing the Bluetooth button will cause the radio delayer to enter a Bluetooth select mode upon scanning for nearby devices. The mode is stored by a simple integer that will be increased modulo the number of modes upon each mode button press. In volume mode, the up and down buttons are used to increase or decrease the volume of the data output between a certain range. AM/FM mode will make the up button correspond to changing to an AM receiver and the down button changes the radio delayer to a FM receiver. The station frequency mode will cause the up and down buttons to seek to the next valid radio station by sending the appropriate command to the radio chip. Next, in delay mode, the up and down buttons will increment or decrement the delay that is imposed on the device. This is accomplished by changing around the head pointer on the internal radio data buffer. The tail will remain unchanging as that is where new radio data is added.

The last main mode is preset mode. All of the variables controlling the above modes are stored in a preset *struct*. By changing the preset, users are able to load and store configurations in order to quickly switch between their favorite stations. When a preset is changed, the new configuration must be loaded by changing all of the internal variables at once. In addition, multiple commands will be sent to the radio chip in order to change the frequency and potentially change from AM to FM. Saving presets is done seamlessly as the user is always directly modifying a preset upon changing any of the above variables. Upon turning on the device, the radio delayer will load information from preset 1. It is also important that all preset information be saved to non-volatile memory, else it defeats the purpose of saving station information.

When the Bluetooth button is pressed, the commands to activate and scan for nearby devices is sent to the Bluetooth module. When the module is done scanning, all of the pairs of friendly names and addresses are stored and the first friendly device appears to the user. The user can then scroll through the list with the up and down buttons and then eventually select which to pair with by pressing the Bluetooth button again. If the radio delayer is currently paired with an audio sink, the link can be disconnected by pressing the Bluetooth button once again.
All of the above corresponds to a simple while loop that will, upon receiving a button press, first enter a switch statement depending on the type of button press (up, down, Bluetooth or mode) and then another switch statement on the current mode of the radio delayer. Since all

information needs to be displayed to the user, at the top of the loop the appropriate information will be sent to the LCD display.

The last part of the code describes how the radio data is read, stored and outputted to the user. Since many samples can be read from a SPORT at once, the easiest approach is to have a timer interrupt every so often to read a bunch of data. This will signify that a new set must be stored and output to either the headphones or Bluetooth module. The overall design of the radio delayer very closely approximates a shift register with variable shift. In order to implement this, an array of data is made with a head and tail pointer. The tail signifies where new data goes and the head is the next bit of data to output. To change the output delay, all that is needed is to change the offset between the head and the tail. In addition, data can easily be compressed or decompressed right before or after it is taken from the array.

6.2 - Bluetooth Pseudocode

For this prototype, the Bluetooth module communicates over UART. The development board for the module included both a USB-Micro to USB port to allow for easy programming from a computer as well as a *combasic* program to communicate effectively between the user and the chip. In addition, the programming reference manual for the chip included a program both to make the chip an A2DP source and sink. Due to ease of using an iPhone as a music source, the first demo was to make the development board output the streaming music. In order to do this, all that is needed was to power up the chip and set the role to an A2DP sink. Since the source initiates the contact and pairing process, this bit of code purely puts the chip into discovery mode so that upon connection from a source, all data sent to it is forwarded to the speaker output.

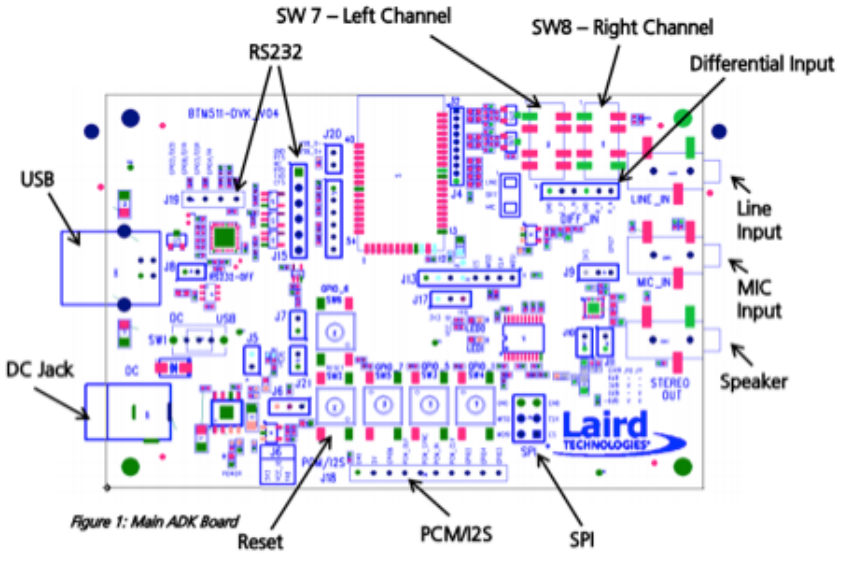

Figure 21: Overview of Bluetooth Development Board

The second demo was to make the module work as an A2DP source. This demo was slightly more involved as it means that the module needs to scan around for all discoverable devices, report back to the user all found devices and then wait for the user to connect to one of them. As when the module is in A2DP sink mode, once paired, all data given to the module is automatically sent to the sink. The code given by the programming reference did work on the first try. This meant that once UART is set up on the microprocessor, it becomes easy to use the module.

Bluetooth Code for A2DP Source

AT
AT&F*
ATS300=2
ATS330=5
AT+BTI
AT+APD<ADDRESS>
AT+GIU
AT+GID
AT+APH

The above is the code in order to operate the Bluetooth module as an A2DP source. For demo purposes on the development board, the first step was to plug in the USB line and plug the line in between the development board and the actual audio source. Once the *combasic* terminal is opened, the Bluetooth chip can be talked to. Whenever a command is sent to the module, a basic response of "OK" is returned if there is no error. All commands start with the prefix AT and the first command of "AT" can be seen as a pure polling command to make sure the Bluetooth module is operational.

The second command "AT&F*" is used in order to reset all internal registers of the module to their factory settings. Next, the module is able to be set into the A2DP source with the command "ATS300=2". This sets the register numbered 300 that controls if the module is in source or sink mode. The last setting on the Bluetooth module is to set what the Bluetooth module reports upon scanning. While only the physical address is needed to pair to a device, it is also helpful for the user to see the friendly name of the device so that they know they are connecting to the right thing. This setting is controlled by register 330 in the module and by setting its value to 5, pairs of addresses and friendly names are returned. In the microprocessor, these three commands will be sent when the radio delayer is turned on.

The next step is to initiate the scanning process. While there are additional registers to set the scanning frequency, the defaults will get the job done. Upon receiving the "AT+BTI" command, the module enters scanning mode and will return pairs of addresses and friendly names to the user. This process can take an undetermined amount of time and will continue until all strong enough signals are reported to the user. Luckily, the processor will return "OK" upon finishing so the microprocessor will know how long to poll the module for. In addition, since only the first 14 characters of the friendly device are stored, it is ok that the amount of information read at once from the Bluetooth module is undefined.

Once the list of sinks is returned to the user, the user can connect to any of them with the command “AT+APD” with the address of the device appended on. This means that the list of pairs of information needs to be stored on the microprocessor and while only the friendly name will be shown to the user, the corresponding address must be easily accessible.

After connecting to the sink, the following information is relayed back to the user:

PAIR 0 <ADDRESS>
CONNECT <ADDRESS>,110D,>
FS44100,INT
APSTR,>

After displaying that information, the Bluetooth module is able to automatically stream information to the sink until the “AT+APH” disconnect command is given. In addition, the Bluetooth module allows for the user to increase or decrease the audio gain with the “AT+GIU” and “AT+GID” commands respectively.

The small amount of code needed to operate the Bluetooth module signifies the ease at which it can interface with the radio delayer. On the microprocessor side, the general algorithm is as follows. As the microprocessor initializes the program, it will also reset and set the internal settings of the microprocessor. When the user wants to begin to enter Bluetooth mode, the microprocessor will initiate the scanning process and store the devices it finds. The user will then be able to cycle through the devices and then select which device to pair with. The microprocessor will then pair with the device, make sure the connection went through and then begin streaming data to the module. When the user wants to exit from Bluetooth mode, the disconnect command will be issued.

6.3 - Cross Core Environment

The main code development all took place on the Cross Core platform. One of the advantages of using this development software was that Analog Devices allows for support software to be easily added. All of the internal registers and external pins of the processor can be easily accessible under easy to understand names instead of by register address. In addition, Cross Core makes it very easy to program external devices and insert break points into the software to poll the device and provide debugging feedback. Beyond that, all of the code was done in C.

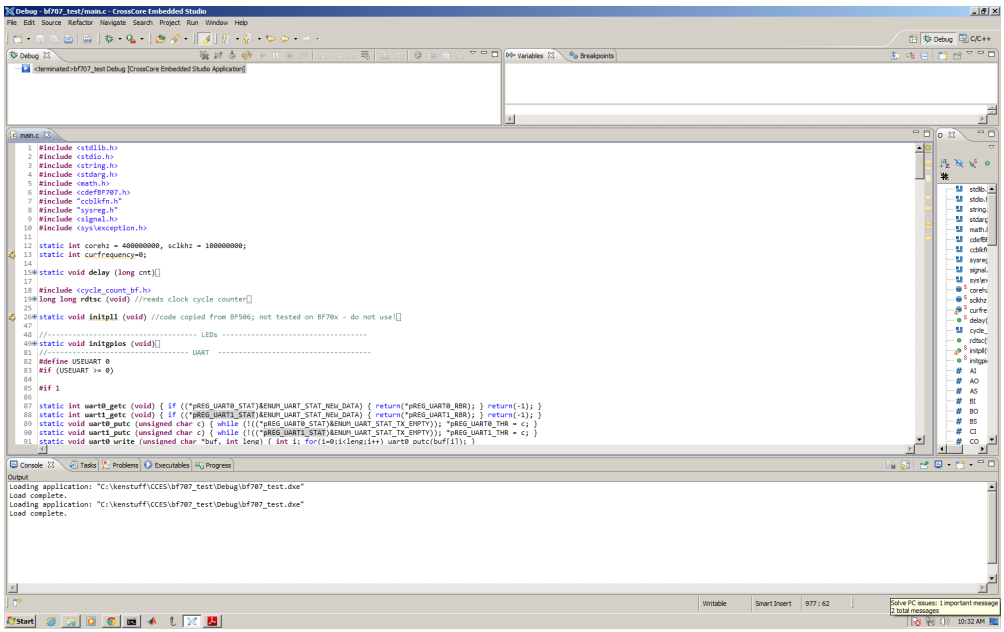

Figure 22: View of the Cross Core Development Application

7 - Economic Analysis

7.1 - Non-recurring Engineering Costs

The major contributors to the NRE costs are labor, for the engineers and consultants, and costs associated with supporting these engineers including office space, lab space and equipment, workstations, software licenses etc.

The average salary for an entry-level Electrical Engineer is taken to be $62,998 per year[10]. Estimating 110% overhead (for a small company) and that 70% of each engineer's time was allocated to the project, we have

$$\text{NRE per engineer} = (0.7 + 1.1) \times \$62{,}998 = \$113{,}396.4$$

For 3 engineers and 1 part-time consultant (Ken Silverman), this yields NRE = $396,887.4

Since the product is still in prototype stage, the NRE is doubled to $793,774.8.

Manufacturing transfer costs are also a big contributor, since an early-stage startup requires external companies to improve various aspects of the manufacturing process. This is assumed to be roughly the original NRE, resulting in a total NRE of $1,190,662.2.

Since the product is aimed at sports fans, an adventurous estimate of 350,000 units per year is made (an estimate 500,000 radios are sold per year, not excluding radio apps in smartphones[11]). Over two years, the per unit cost is thus 1190662.2/700000 = $1.7

7.2 - Manufacturing and Assembly Costs

The main price for the prototype comes from the individual components, which alone comes out to be a price of around $51.

Quantity	Part Name	Part Number	Cost (1,000 pt)
1	RF radio chip	Si4730	$2.23
1	Microcontroller	BF707	$5.00
5	Tactile Switch	TL3315NF160Q	$0.55
1	Microchip Bluetooth Module	BTM511	$19.00
1	LCD display	LCM-S01602DTR/M	$4.11
1	Boost to 5V	ADP1612	$1.07

[10] https://www.naceweb.org/s01072015/engineering-majors-top-salary-class-of-2015.aspx
[11] https://transition.fcc.gov/osp/inc-report/INoC-2-Radio.pdf

1	Buck + Buck + LDO	ADP5024	$2.91
1	USB Micro B receptacle	10118193-0001LF (Digikey)	$0.26
1	Rechargeable battery	UBP002	$10.30
2	3.5mm jack	CP-3502SJCT-ND	$1.66
1	Female barrel plug (5.5 mm)	PJ-006B-SMT-TR	$0.57
1	Headphone Driver	TPA6130A2	$0.90
1	Charging Chip	LM3658	$0.51
3	ESD protection IC	CM1214-01ST	$0.54
1	Oscillator Chip	NZ2520SB-32.768KHZ-NSA3536C	$0.85
1	Oscillator Chip	MIC803-29D3VC3 TR	$0.19
1	Potentiometer	TC33X-2-103E	$0.17
	TOTAL		**$50.82**

Table 5: Cost Summary of the Product

In addition, a board size of 14cm x 5cm at $0.04 per sq. inch adds another $0.434 to the cost of the prototype. This brings the total for the manufacturing cost to be about $51.04.

In order to calculate the assembly cost of the prototype, the following metric was used. It was assumed that ICs could be assembled at 2 cents per chip and that passives could be assembled at 1 cent for each. For the board size, there would be an additional 1 cent per square inch. With 80 passives, 14 ICs and the above board size, this adds another $1.18.

The final addition to the price is the total mechanical frame assembly price. In order to get a good estimate for this cost, there were two possible metrics. The first was that there could be an hourly cost of about $5 per hour. The second is that the job could be outsourced and have a cost of about 1-5% of the final sale price depending on the volume. If we take the sale price to be about 3 times the manufacturing cost, this would add another dollar to the final price.

With all of these factors in play, this brings the final price of the board to be about $53.22. This turned out to be much more expensive than was originally calculated. Most of this cost comes from using a ready-made Bluetooth module and for future iterations of the device it would be in the best interest to look into how to make a Bluetooth module from scratch.

7.3 - Analysis of Salability

Because of the relatively high cost of producing such a board ($53.22), this product will become expensive at the consumer level: in order to achieve a profit margin large enough to justify the production of this device, it would have to be sold to consumers for at least $150. This would

preclude the casual sports fan from buying this device. Additionally, in order to put the product into production for end users, there are likely some components, which would have to be changed. Specifically, larger buttons would probably be selected, and charging cables would likely have to also be included, further raising the cost.

However, because of the external Bluetooth capability, we feel this product does serve a specific market and offers features that aren't achieved elsewhere. For committed sports fans who feel strongly about listening to their team's radio announcers, this product gives the flexibility of having the device in your pocket and using headphones (so as not to disturb other viewers), or connecting to a Bluetooth speaker wirelessly. For such a sports fan, this device can completely revolutionize the way a game is watched, and the cost is certainly not prohibitive. Thus, we feel this device has potential in this very specific market and would be salable despite the cost.

8 – Appendix

8.1 – Links to Component Datasheets

Silicon Labs Radio Chip
https://www.silabs.com/Support%20Documents/TechnicalDocs/Si4730-31-34-35-D60.pdf

BF707 microprocessor
http://www.analog.com/static/imported-files/data_sheets/ADSP-BF700_BF701_BF702_BF703_BF704_BF705_BF706_BF707.pdf

Laird Bluetooth Module
http://www.lairdtech.com/WorkArea/linkit.aspx?LinkIdentifier=id&ItemID=4118

Lumex LCD Display
http://www.lumex.com/specs/LCM-S01602DTR%20M.pdf

USB-Micro B receptacle
http://media.digikey.com/pdf/Data%20Sheets/FCI%20PDFs/10118193-0001LF_Webpage.PDF

8.2 – Links to Hardware and Software Documentation

Bluetooth Development Board:
http://www.lairdtech.com/brandworld/library/User%20Manual%20-%20Laird%20Audio%20Development%20Kit%20%28ADK%29.pdf

Bluetooth Coding Reference:
http://www.lairdtech.com/brandworld/library/User%20Guide-%20BTM51x.pdf

Processor Breakout Board:
http://www.analog.com/media/en/technical-documentation/user-guides/SDP_BRKBD_UG_rev1.1.pdf

Processor Programming Reference:
http://www.analog.com/media/en/dsp-documentation/processor-manuals/ADSP-BF70x_Blackfin_Programming_Reference.pdf

Processor Hardware Reference:
http://www.analog.com/media/en/dsp-documentation/processor-manuals/BF70x_BlackfinProcessorHardwareReference.pdf

Radio Chip Programming Reference:
http://www.silabs.com/Support%20Documents/TechnicalDocs/AN332.pdf

8.3 – Schematics
See attached.

8.4 – Layout
See attached.

8.5 – Code

Code can be downloaded at:
https://www.dropbox.com/s/vya7oop5wvv24sr/ENGN165Code.c?dl=0