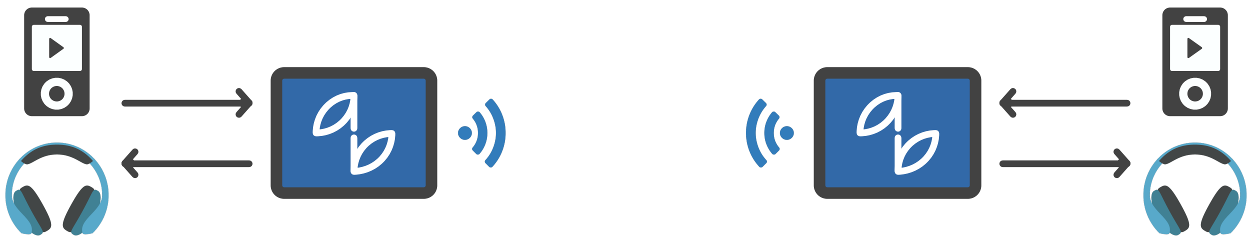
Wireless Audio Bridge

by

Daniel Hodges



Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2017

Table of Contents

List of Tables and Figures

Abstract

Bluetooth, a wireless technology standard used to exchange data using radio transmissions, has made a significant impact on the audio technology industry, specifically the way people listen to music. This technology enables a wireless listening experience, eliminating the need for wires or cables between audio devices as previously required. Audio can transmit from a cellphone or laptop (an output device) to a pair of headphones or speakers (an input device) without any physical connection. Compared to alternative wireless listening solutions (i.e. FM transmitter, radio broadcast), Bluetooth offers greater reliability, audio fidelity, portability, and ease of use. To obtain this functionality, both the output and input devices require Bluetooth compatibility. This confines non-Bluetooth compatible devices to wired connections.

There exists potential for a network of units that allows two non-Bluetooth compatible devices to communicate for a wireless listening experience. These units use Bluetooth technology to communicate wirelessly, and transfer audio signals between audio devices with minimal distortion and latency. A transmitting unit connects to an audio output device and transmits the audio signals to a receiving unit. The receiving unit outputs the audio signals to an audio input device.

Chapter 1: Introduction

The initial idea behind this project derives from an interest in Bluetooth technology in combination with audio. Originally conceived as a wireless alternative to RS-232 cables, Bluetooth intended to connect several devices together over a distance. Since its conception in 1994, Bluetooth gained popularity through many industries and continues to spread. For the audio industry, Bluetooth started as the main technology behind the application of audio streaming between portable devices [12].

Audio streaming, a constant audio transmission over a data network, allows users to listen to audio in near real-time (unnoticeable delay between source and sink). This technology behind audio streaming utilizes a buffering system and a secure data stream platform to allow the users to listen to audio without interruption [13]. The advancements of the Bluetooth standard allow for wireless audio streaming with high fidelity, reliability, and power efficiency. The number of Bluetooth devices capable of audio streaming continues to grow, specifically in the consumer market. Manufacturers increasingly integrate Bluetooth technology into both audio output devices (cellphones, laptops, portable audio players etc.) and audio input devices (headphones, speakers, etc.) for wireless audio streaming. Purchasing Bluetooth compatible devices remains the ideal solution for wireless audio streaming, however, for many users it proves impractical.

Compatibility and cost deter customers from investing in Bluetooth wireless streaming. The typical setup for streaming wireless audio includes a cell phone or laptop to transmit the audio and a pair of headphones or speakers to receive the audio. To stream audio over Bluetooth, both the transmitting and receiving device require Bluetooth technology. Cases of incompatibility between older and newer Bluetooth devices arise from the developments and upgrades to the Bluetooth standard. Bluetooth devices also remain an expensive purchase [14]. Most Bluetooth compatible cell phones and laptops exist on the higher end of the price bracket. Prices of Bluetooth compatible headphones or speakers can extend well above the price of comparable non-Bluetooth products.

This project establishes a practical method of wireless audio streaming for non-Bluetooth compatible devices, while maintaining the benefits of Bluetooth technology. The basic concept consists of a transmitting unit connected to an audio output device transmitting audio signals to a receiving unit connected to an audio input device, creating a “Wireless Audio Bridge”. Integrating Bluetooth technology into these units, rather than the audio devices themselves, eliminates the device compatibility barrier.

Chapter 2: Customer Needs, Requirements, and Specifications

Customer Needs

This system aims to provide another method for people to listen to audio wirelessly. Research on alternative solutions in the market identified unfulfilled customer needs. This system intends to cover the needs of compatibility, functionality, portability, and cost while providing an attractive wireless listening solution.

Most of the current wireless listening solutions require purchasing specific devices, limiting customer's options. This project aims to support the majority of audio output and input devices. The transmitting and receiving units can interchange amongst audio devices, allowing virtually any two devices to communicate wirelessly. The system meets the portability need, allowing the user to move freely when listening to audio. This primarily applies to wearable listening devices (i.e. headphones, headsets), but can extend to stationary devices as well (i.e. speakers). Wireless connections provide the user the ability to move freely (relative to the audio source); the system must accommodate this functionality while remaining relatively compact. Cost diverts many consumers from investing in a wireless audio solution. This system provides a cost-effective solution to wireless audio listening.

An additional function this system provides is wireless audio sharing. This feature is enabled by outputting the audio signal on the transmitting unit as well as the receiving unit. The user with the transmitting unit (and the audio source) can listen to the audio directly from the transmitting unit, while another user listens to the same audio from the receiving unit. These units can also switch transmit and receive modes, allowing users to take turns being the "host" of the audio sharing experience.

Aside from the primary needs, the system fulfills additional needs pertaining to audio listening. The system must enhance a listening experience without introducing any substantial impediments. This leads to needs of ease of use, high-quality audio, and long battery life. Ease of use entails a simple operation of the system and minimal user interaction. The system sets up easily and does not require configuration or maintenance. The transmitting and receiving units introduce minimal distortion and noise to ensure the quality of the audio signals. The battery life is sufficient for using listening to audio hours at a time, in order to minimize the replacing or recharging of the unit's batteries. The system must also remain safe for the user to use and carry around. The system must be easy to use and provide an intuitive user interface. These customer needs translate into marketing requirements as shown in Table I.

TABLE I
WIRELESS AUDIO BRIDGE MARKETING REQUIREMENTS

Marketing Requirements
1. Long wireless range.
2. Excellent audio streaming.
3. Long-lasting battery life.
4. Widely compatible.
5. Portable.
6. Low cost.
7. Safe to use.
8. Intuitive user interface.

Requirements and Specifications

The requirements and specifications for the wireless audio transmitter and receiver system appear in Table II. The engineering specifications support the marketing requirements of Table I, while providing constraints and guidelines for the design of the system.

The first few engineering specifications support the important aspects of the system's functionality and performance. The system uses Bluetooth technology to wirelessly transmit audio data. The range of a wireless system limits the physical distance between devices in the network. For portable wireless audio listening, typically both the audio output device and audio input device operate in close proximity to each other. Thus, a range of 30 feet between devices proves adequate and practical for this system. The system streams stereo audio signals between the receiving and transmitting units. Stereophonic sound appears in most audio devices and remains desirable over monophonic sound [15]. The time delay, or latency, between the transmitting and receiving unit's audio signals must remain miniscule. Audio latency could negatively affect the listening experience (for situations requiring small delay) if too large. Contributors to latency include analog-to-digital conversion, processing of signals, transmission time, and digital-to-analog conversion. The 500-millisecond constraint accounts for the time it takes the signals to transmit while ensuring the audio delay remains relatively small [6]. The time it takes the transmitting and receiving unit to establish a connection also plays a part in the audio streaming experience. The constraint of 5 seconds provides enough time for the initialization of the units. While listening to audio on the receiving end of the system, the user should adjust the audio's volume as desired. The receiving unit has a user adjustable volume control used to increase or decrease the audio output's volume. A mute and unmute feature also allows the user to quickly disable and re-enable audio output, without closing the wireless connection altogether.

Another specification ensures the wide compatibility requirement. To support the majority of portable audio devices, the system interfaces with devices using standard 3.5 mm audio ports. 3.5 mm audio ports are nearly universal for portable audio devices including cellphones, computers, portable media players, headphones, etc. [9].

The following two specifications cover the portability and ease of use of the system. Each unit should constrain to the size of a typical handheld device to allow for portability while remaining large enough to hold an adequate sized battery. Each unit should fit in a typical pocket or purse to allow easy transportation. The battery life of each unit must last long enough for a typical listening time (at least a few hours). The batteries should be rechargeable, preventing the need to swap out batteries once depleted. Another specification covers the cost of the system. The system remains competitive in its cost to compete with other wireless listening solutions.

Electrical devices, especially handheld types, typically enclose its electrical components for safety purposes. The device should not expose its electrical components to the outside environment in order to protect the components as well as humans from possible shock events. Exposed electrical components could cause a transfer of energy between the device and the user, which can harm either side. An additional specification ensures both the transmitting and receiving unit has packaging made to properly enclose its internal components.

To enable the wireless audio sharing feature, the transmitting unit must output the audio directly from the audio source. In order to switch the direction of audio sharing, the transmitting and receiving units must be able to swap transmit and receive modes. Each unit is not confined to either a transmit or receive mode, allowing either unit to act as the source or the sink. To notify the user of the current mode, colored LEDs indicate whether the unit is configured as a transmitter or receiver.

TABLE II
WIRELESS AUDIO BRIDGE REQUIREMENTS AND SPECIFICATIONS

Marketing Requirements	**Engineering Specifications**	**Justification**
1, 2, 4	The transmitting and receiving unit communicates using Bluetooth technology.	Bluetooth technology allows for high quality and reliable audio streaming.
1, 2	The transmitting and receiving unit communicates and streams audio within a range of at least 30 feet.	This range yields an adequate distance for wireless audio streaming, while providing enough distance for the units to communicate properly.
2	The transmitting unit and receiving unit stream stereo audio signals.	Most audio devices operate with stereo audio signals.
2	The audio latency between the transmitting and receiving unit does not exceed 500 ms.	It takes time to transmit signals between audio devices.
2, 8	The transmitting and receiving unit establishes audio communication in less than 5 second after power-up.	This time allows each unit to initialize and establish communication prior to streaming audio data.
2, 8	The receiving unit has an adjustable volume control between no volume and a maximum volume level, as well as mute/unmute capability.	The audio device connected to the receiving unit should have a user adjustable volume control.
4, 8	The transmitting and receiving unit uses standard 3.5 mm audio ports (male and/or female) to interface with audio devices.	3.5 mm audio ports are nearly universal for portable audio devices.
5	The dimensions of each unit do not exceed 3” x 2” x 1”.	Each unit must remain the size of a typical handheld device to ensure portability and ease of use.
2, 3, 5	Each unit operates for at least 4 hours of audio streaming on one battery life.	This correlates with the physical size constraint on each unit, which limits the potential size of the battery.
3, 6, 7, 8	The battery in each unit has recharging capability.	Rechargeable batteries prevent the need for swapping out batteries when depleted.
6	The total parts cost does not exceed $60 per unit.	The total cost of the system competes with other wireless listening alternatives.
7	The transmitting and receiving unit completely encloses its electrical components.	The system should properly enclose all electrical components to avoid the transfer of energy between the device and humans.
2, 8	The transmitting unit outputs audio directly from the audio source.	This enables a user to listen to audio directly from the audio source.
2, 8	Both units in the system are capable of switching between transmit and receive modes.	Neither unit is confined to a transmit or receive mode, allowing either unit to be the source or the sink of the audio.
8	Two different colored LEDs indicate the current transmit or receive mode of each unit.	The current mode of each unit should be easily and visibly identifiable.

Chapter 3: Functional Decomposition

Level 0

The Level 0 functionality portrays the overall functionality of the system. The block diagram in Figure I and functional requirements in Table III convey the system's primary inputs and ouputs.

The system has five inputs: a stereo audio signal, a 5 V DC power source, a volume control input, a mute/unmute input, and a transmit/receive mode swap input . The system receives these inputs and outputs a stereo audio signal at the specified volume. The 5 V DC power source supplies power to they system and charges the battery. The mute/unmute input disables and re-enables audio output. Asserting the mode swap input swaps the current modes of the transmit and receive units. Two LED indication outputs allows the user to recognize the current transmit/receive mode of each unit.

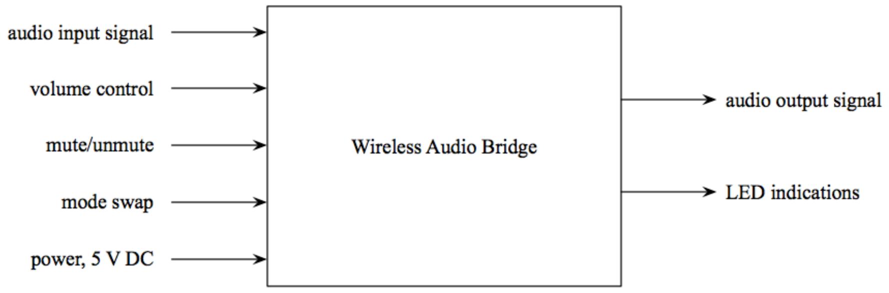


FIGURE I
WIRELESS AUDIO BRIDGE LEVEL 0 BLOCK DIAGRAM

TABLE III
WIRELESS AUDIO BRIDGE FUNCTIONAL REQUIREMENTS

Module	Wireless Audio Bridge
Inputs	- Audio input signal: stereo analog signal - Volume control: user adjustable - Mute/unmute: user configurable - Mode swap: user configurable - Power: 5 V DC
Outputs	- Audio output signal: stereo analog signal - LED indications: light source
Functionality	Wirelessly transmit an audio signal between audio devices. Via a user volume control, the audio output volume varies between no volume and a maximum volume level. A mute/unmute control disables and re-enables audio output. Mode swap input swaps transmit/receive modes of each unit. A 5 V DC input charges the system. LEDs indicate the current mode of each unit.

Level 1

The Level 1 functionality includes the subsystems or modules of the architecture. The Level 1 block diagram in Figure II contains three modules: a transmitting unit, a receiving unit, and batteries. Tables IV, V, and VI convey the functional requirements of each of these modules.

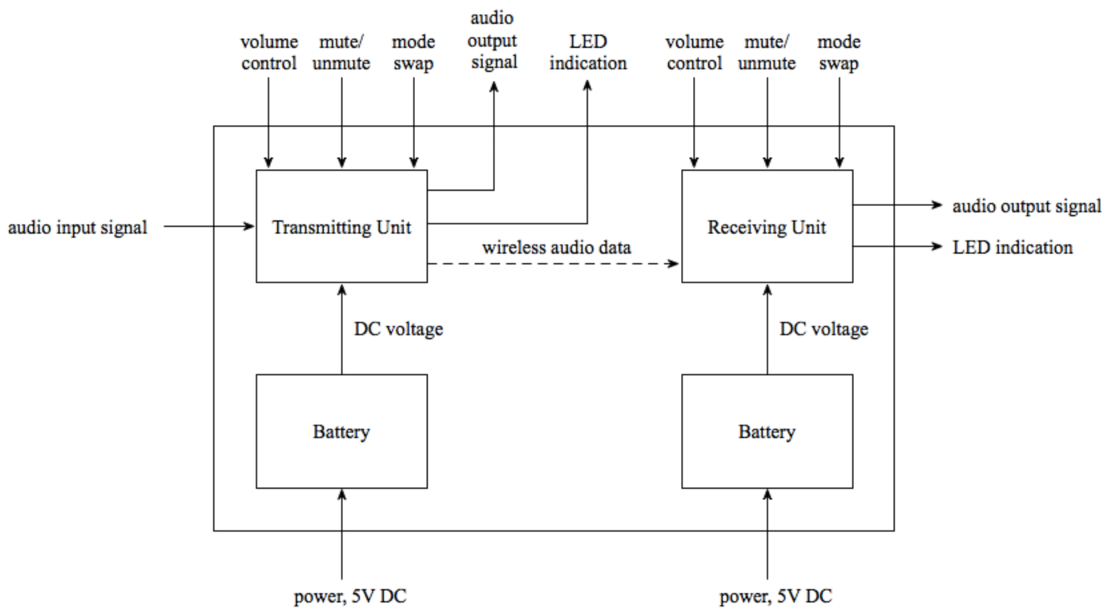


FIGURE II
WIRELESS AUDIO BRIDGE LEVEL 1 BLOCK DIAGRAM

The transmitting unit receives the audio input signal from an audio output device, and outputs the audio data wirelessly using Bluetooth protocol. Bluetooth devices transmit and receive radio waves that occupies a section of the 2.4 GHz ISM (industrial, scientific, and medical) band 83.5 MHz wide [18]. The audio signal is also outputted directly from the transmitting unit. A user volume control varies the audio output volume between no volume and a maximum volume level. A mute/unmute control disables and re-enables audio output. The mode swap input switches the unit to receive mode. A colored LED indicates that the unit is in transmit mode. A battery provides the required DC voltage and current for operation.

TABLE IV
TRANSMITTING UNIT FUNCTIONAL REQUIREMENTS

Module	Transmitting Unit
Inputs	- Audio input signal: stereo analog signal - Volume control: user adjustable - Mute/unmute: user configurable - Mode swap: user configurable - DC voltage: 3.3 V DC
Outputs	- Wireless audio data: 2.4 to 2.485 GHz radio waves - Audio output signal: stereo analog signal - LED indication: light source
Functionality	Wirelessly transmit an audio signal from an audio output device to the receiving unit.

The receiving unit receives the wireless audio data over Bluetooth from the transmitting unit, and converts the data back into the original audio signal. The audio signal outputs to an audio device with an audio input. A user volume control varies the audio output volume between no volume and a maximum volume level. A mute/unmute control disables and re-enables audio output. The mode swap input switches the unit to transmit mode. A colored LED indicates that the unit is in receive mode. A battery provides the required DC voltage and current for operation.

TABLE V
RECEIVING UNIT FUNCTIONAL REQUIREMENTS

Module	Receiving Unit
Inputs	- Wireless audio data: 2.4 to 2.485 GHz radio waves - Volume control: user adjustable - Mute/unmute: user configurable - Mode swap: user configurable - DC voltage: 3.3 V DC
Outputs	- Audio output signal: stereo analog signal - LED indication: light source
Functionality	Receives wireless audio data from the transmitting unit and outputs the original audio signal.

The battery provides the required DC voltage and current for the system. The system uses two batteries; one for the transmitting unit and one for the receiving unit. Bluetooth audio modules, including the Microchip RN52 [3] and the BlueCreation BC127 [11], typically operate with a 3.3 V DC supply. The working current of these modules range from 15 to 30 mA. The balance between physical size and amp-hour capacity of the battery affects the portability and size related constraints on the system. Rechargeable type batteries require input power to recharge after depletion.

TABLE VI
BATTERY FUNCTIONAL REQUIREMENTS

Module	Battery
Inputs	- DC voltage: 5 V DC
Outputs	- DC voltage: ~2.5-3.7 V DC with at least 100 mA of current
Functionality	Provides required voltage and current to supply the transmitting and receiving units. Must last at least 4 hours (based on specification) while system operates.

Chapter 4: Project Planning

In order to address time and cost management for this project, Gantt charts and Cost Estimates table appear in Figures III, IV, and V and Table VII.

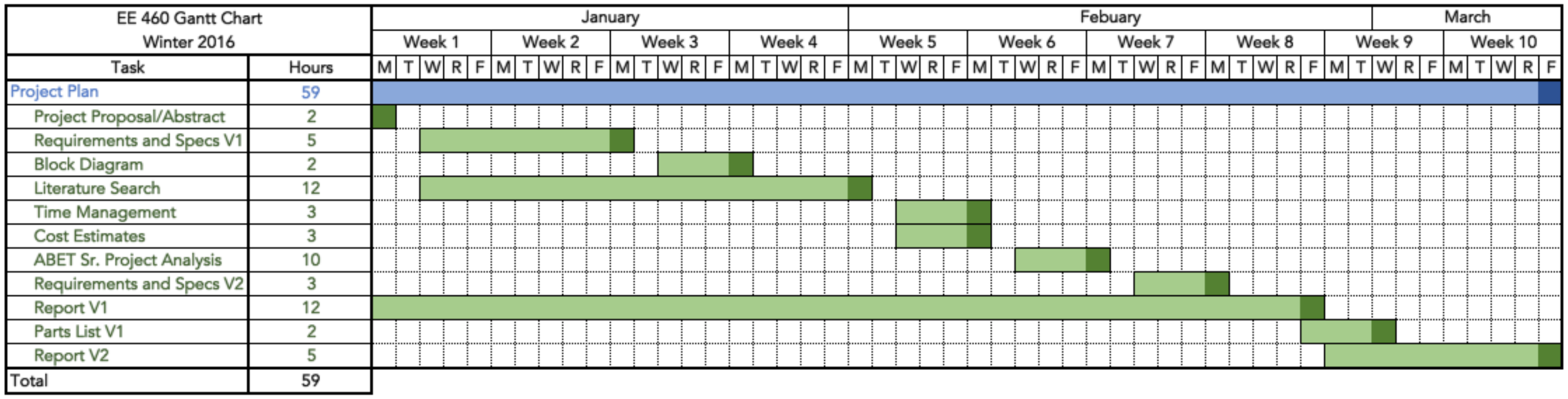
The Gantt chart depicts the projected tasks of the project, spanning from January 2016 to June of 2017 across the three senior project courses (EE 460, EE 461, and EE 462). As seen in the chart, the main tasks include project preparation, concept development, design-build-test cycles, and the final report. Sub-tasks convey the individual efforts required for completion of the corresponding main task.

The projected time in hours for each task is shown under the "Hours" column. Using the PERT (project evaluation review technique) technique [16], a method for estimating activity duration, the following equation estimates the duration of each task.

$$t_e = \frac{t_a + 4t_m + t_b}{6}$$

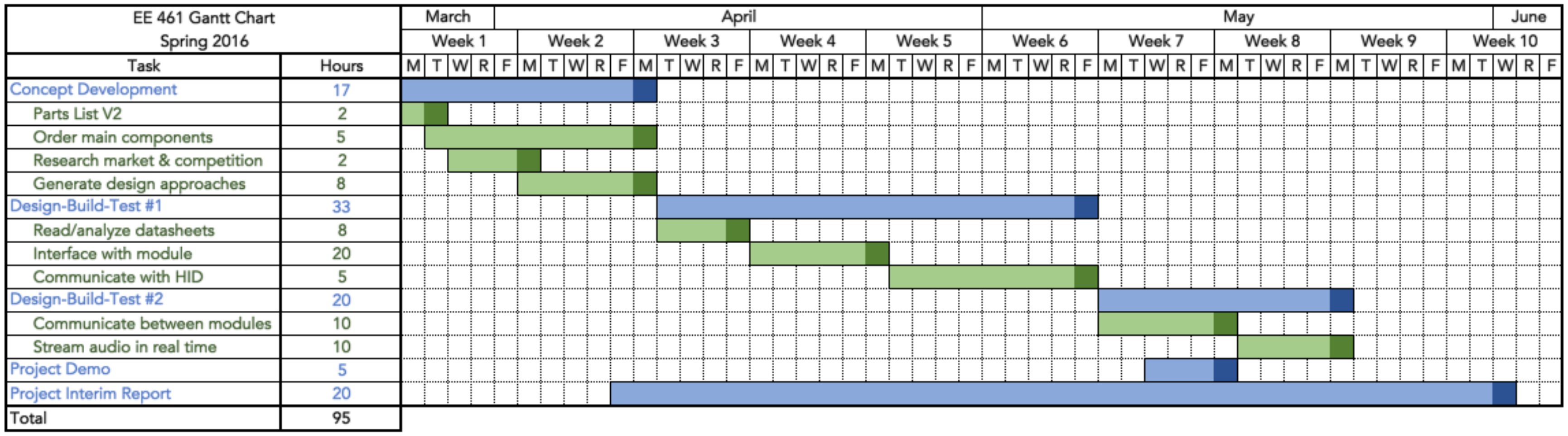
$$t_e\text{: }estimate\ result$$
$$t_a\text{: }most\ optimistic\ estimate$$
$$t_m\text{: }most\ realistic\ estimate$$
$$t_b\text{: }most\ pessimistic\ estimate$$

The "Hours" column lists the results of this equation applied to each task (the results rounded to the nearest hour). The total duration of the project tasks due to this model results in approximately 313 hours.



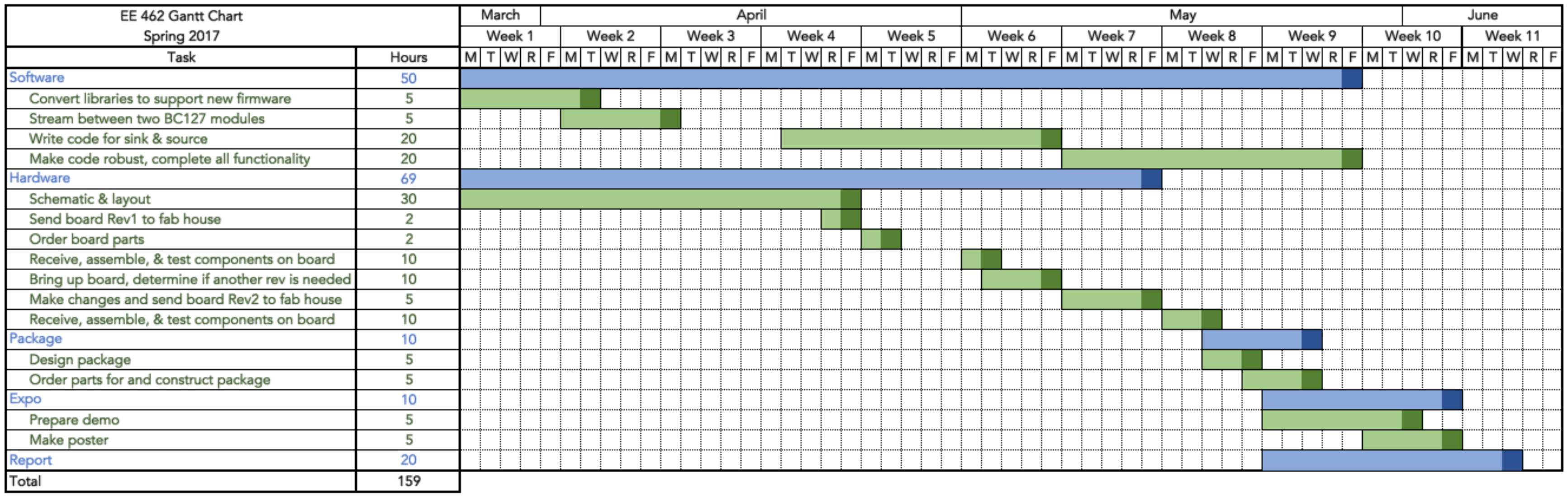
EE 460 Gantt Chart Winter 2016	
Task	Hours
Project Plan	59
Project Proposal/Abstract	2
Requirements and Specs V1	5
Block Diagram	2
Literature Search	12
Time Management	3
Cost Estimates	3
ABET Sr. Project Analysis	10
Requirements and Specs V2	3
Report V1	12
Parts List V1	2
Report V2	5
Total	59

FIGURE III
EE 460 (WINTER 2016) GANTT CHART



EE 461 Gantt Chart Spring 2016	
Task	Hours
Concept Development	17
Parts List V2	2
Order main components	5
Research market & competition	2
Generate design approaches	8
Design-Build-Test #1	33
Read/analyze datasheets	8
Interface with module	20
Communicate with HID	5
Design-Build-Test #2	20
Communicate between modules	10
Stream audio in real time	10
Project Demo	5
Project Interim Report	20
Total	95

FIGURE IV
EE 461 (SPRING 2016) GANTT CHART



EE 462 Gantt Chart Spring 2017	
Task	Hours
Software	50
Convert libraries to support new firmware	5
Stream between two BC127 modules	5
Write code for sink & source	20
Make code robust, complete all functionality	20
Hardware	69
Schematic & layout	30
Send board Rev1 to fab house	2
Order board parts	2
Receive, assemble, & test components on board	10
Bring up board, determine if another rev is needed	10
Make changes and send board Rev2 to fab house	5
Receive, assemble, & test components on board	10
Package	10
Design package	5
Order parts for and construct package	5
Expo	10
Prepare demo	5
Make poster	5
Report	20
Total	159

FIGURE V
EE 462 (SPRING 2017) GANTT CHART

The estimated costs for this project appear in Table VII. Parts and labor expenses constitutes the total cost of this project. Not included are tax, shipping, or other miscellaneous costs.

The list of parts shown anticipate the required parts (and corresponding costs) needed for the design and development of the project. The test equipment for the development of this project does not apply as the development takes place in facilities with available test equipment resources.

The labor cost derives from the total predicted hours of the project as well as the equation using the PERT technique [16].

$$cost_e = \frac{cost_a + 4cost_m + cost_b}{6}$$

$$cost_e\text{: } estimate\ result$$
$$cost_a\text{: } most\ optimistic\ estimate$$
$$cost_m\text{: } most\ realistic\ estimate$$
$$cost_b\text{: } most\ pessimistic\ estimate$$

Assuming a pay rate of \$35/hr, the estimated cost results in approximately \$11,000. The total cost, including parts and labor costs, results in \$11,440 in U.S. dollars.

TABLE VII
WIRELESS AUDIO BRIDGE COST ESTIMATION

Item	*Quantity*	*Cost*	*Total Cost*
Parts			**$495**
• BC127 module	3	$30	$90
• BC127 development board	2	$40	$80
• Sparkfun ProMicro	3	$20	$60
• Headphone driver IC	5	$1	$5
• Lithium-ion battery	2	$20	$40
• PCBs	10	$5	$50
• Packaging	2	$20	$40
• Soldering tools	N/A	$30	$30
• Test/development parts and tools	N/A	$50	$50
• Misc. parts (connectors, switches, discrete components, etc.)	N/A	$50	$50
Labor		313 hr	**$10,955**
Total			**$11,450**

Chapter 5: Design

Design Concept

This system consists of two units, a transmitter and a receiver. Each unit has the capability to act as the transmitter or receiver, thus the hardware of each unit is designed to be identical.

There are 3 main components providing the functionalities of the system: a Bluetooth module, a headphone driver, and a microcontroller. The Bluetooth module is used to send or receive (depending on its transmit/receive mode) audio signals to or from the other unit. The headphone driver is the audio amplifier used to convert and amplify the Bluetooth module's audio signals to drive a supported audio input device. The microcontroller contains the state machine used to control the system and other components: it sends appropriate commands to the Bluetooth module, handles I/O (buttons, LEDs, etc.), and ultimately ensures the transmit and receive units are connected and streaming. Other supporting components are used to alongside these aforementioned components.

The size and portability requirements are met by integrating the hardware components onto a printed circuit board. This board is designed to physically hold and electrically connect the components together into a single package (one for each unit).

The software portion of this design lies in the microcontroller. The microcontroller needs to be programmed to control all the other components in each unit. The program is designed to behave like a state machine; relying on various inputs to control its outputs. Hardware interrupts are heavily used to indicate various events, which allows the microcontroller to react in an appropriate and timely manner.

Hardware Design

Bluetooth module:

The basic capabilities the Bluetooth module must have are audio streaming, transmit or receive modes, integrated antenna, and battery charging capabilities. BlueCreation's BC127 Bluetooth module provides these features in a relatively inexpensive and small package. The block diagram of the BC127 module in Figure VI shows the major features of the module.

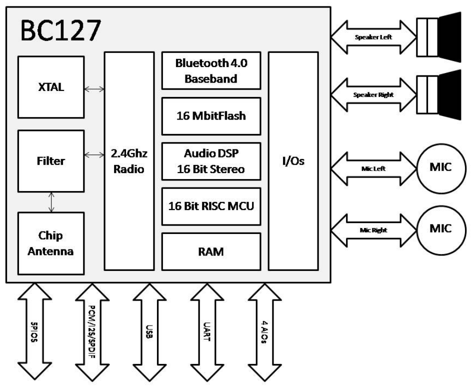


FIGURE VI
BC127 BLUETOOTH MODULE BLOCK DIAGRAM

In terms of audio streaming functionality, the BC127 module has all that is required to support and meet the audio specifications of this system. It supports various Bluetooth profiles including A2DP and AVRCP profiles, used for audio streaming. A2DP (Advanced Audio Distribuation Profile) is the profile which defines how audio can stream over Bluetooth from one device to another. AVRCP (Audio/Video Remote Control Profile) is the profile used to remote control various audio devices. The most common use for AVRCP is skipping tracks, adjusting volume, and pausing/playing audio remotely. The BC127 module contains two sampling channels, allowing the streaming of stereo audio. The resolution of the converters (ADCs and DACs) for each channel is 16 bits (BC127-HD supports 24 bit resolution), introducing minimal degradation of the streamed audio. Each channel has an analog and digital programmable gain stage, allowing for volume adjustment. The analog audio diagram of the BC127 module in Figure VII shows the high-level conversion process. Each module can act as the source (transmitter) or sink (receiver) of the audio, allowing for mode swapping capability.

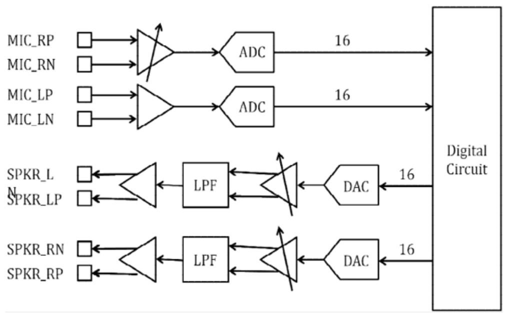


FIGURE VII

BC127 ANALOG AUDIO DIAGRAM

The BC127 module comes preloaded with BlueCreation Melody software, allowing the module to be integrated and controlled specifically for any system. It uses UART as its command interface, allowing simple serial communication between the module and a controller. Melody software has two operating modes which define how data coming from UART is processed: command and data mode. Data mode is used for BLE, IAP, or SPP Bluetooth profiles (not used in this system) and is not required for this system. In command mode, the module accepts commands from the host (microcontroller) via UART. There are over sixty commands the module supports used to control the various features as well as request pertinent data from the module. Certain parameters of the module are configurable and can be stored in flash memory to maintain certain settings after a power cycle. By using these control and configuration commands, the BC127 module can be controlled to fit into many applications.

There are two main power rails required to power the BC127's internal circuitry. The BC127 datasheet calls VBAT the main supply to the internal switch mode and LDO (low-dropout) regulators, requiring an input voltage ranging from 2.8 – 4.25V. This supply is intended to be connected directly to a battery (lithium-ion). The other power rail, VDD_PADS, supplies all the I/O domains including UART, LEDs, and other digital circuitry. A 3.3V external supply is recommended to power this rail. The BC127 module is capable of charging a lithium-ion battery properly, as long as supported external components are connected appropriately. Charging current ranges from 0 to 200mA, and the BC127 supports "trickle charge" as well as "fast charge" modes dependent on the battery voltage state.

Regarding the physical package of the module, the BC127 is a surface mount 51-pin package with 0.6mm land pads. The dimensions of the package are 11.8 x 18 x 3.2 mm. An integrated antenna lies in the upper right corner of the package, requiring a ground clearance to prevent detuning.

The BC127 contains a multitude of other functions and features not previously mentioned, but are either not applicable or not required in the scope of this system.

Headphone driver:

The headphone driver must appropriately convert and amplify the BC127 outgoing audio signal in order to drive a supporting audio input device (headphones, speakers). Most headphones and speakers are driven with single-ended signals, or signals referenced to ground. For stereo audio devices, both left and right channel ground-referenced signals are required. The connection scheme for a standard 3.5mm stereo audio jack is shown in Figure VIII.

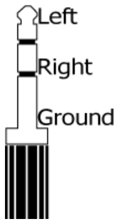


FIGURE VIII
STANDARD 3.5MM AUDIO JACK CONNECTION

The BC127 module provides a differential balanced audio output, with each channel being represented as a differential pair of signals. The headphone driver must convert the BC127 differential signals to single ended in order to drive the audio devices. Texas Instrument's TPA6138A2 provides this functionality and is commonly used for this particular application. The TPA6138A2 is capable of driving 25 mW into a 32 Ω load, supporting the majority of audio devices. It is powered with a single 3.3 V rail, making it suitable for this system's power architecture. Figure IX shows the block diagram of the TPA6138A2 as well as the supporting external circuitry for a typical application.

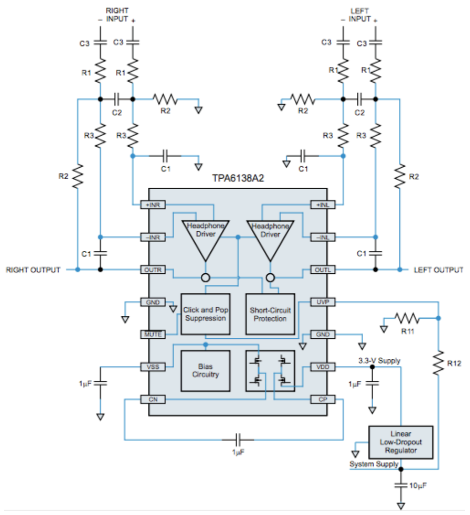


FIGURE IX
TPA6138A2 TYPICAL APPLICATION SCHEMATIC

The TPA6138A2 contains a headphone driver for both left and right channels, converting the BC127's differential signals to single ended signals for audio devices. As seen in Figure 1X, input circuitry into these headphone drivers are used as a combination of filters and gain stages. Appropriate values for these components must be chosen to deliver the desired frequency response and default gain settings.

Other features the TPA6138A2 includes are mute circuitry, an internal charge pump (to create the negative voltage rail), click and pop suppression, and UVP (under voltage protection).

Microcontroller:

The microcontroller is used as the “master” of the system, which will be programmed to control the rest of the system’s components. This microcontroller for this system must consume low-power, have UART capability, contain sufficient digital I/O, support hardware interrupts and timers, and operate off of 3.3 V. These criteria match a multitude of microcontrollers currently on the market, however, Atmel’s ATMega32U4 microchip is used due to its low cost, general purpose use, and Arduino compatibility. The benefit of an Arduino compatible microcontroller is multifaceted: high-level programming (all that’s required for this system), many included libraries, large support community, and simple user interface.

Rather than integrating the ATMega32U4 chip as a standalone component to the system (requiring the need to select and purchase needed supporting components), a development board with the ATMega32U4 along with required and additional components proves a better option. This method’s main advantage is less design and integration work (as to not reinvent the wheel). A couple disadvantages include increased cost (purchasing individual components results in lower total BOM cost) and physical space constraints (the system’s design must conform to the physical size and layout of the development board). The single advantage proves to outweigh the disadvantages for the scope of this project. The development board with the ATMega32U4 used is SparkFun’s ProMicro 3.3V/8MHz board shown in Figure X.

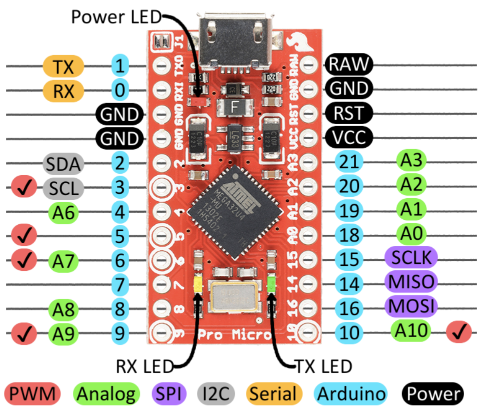


FIGURE X

SPARKFUN PROMICRO DIAGRAM AND PINOUT

The ProMicro gives access to all the pertinent I/O of the ATMega32U4, while providing additional circuitry and components commonly used in conjunction with the ATMega32U4. The ProMicro is programmed and can be powered through an onboard micro-USB connector, which can be used as the charging port of this system. The board has an external 8MHz oscillator providing the clock for the ATMega32U4. It includes a 3.3V voltage regulator, accepting unregulated power commonly coming from a battery. It also has LEDs for power and RX/TX indication and a fuse/diodes for any over-current/back-current events.

Board Design

In order to meet the physical dimension and portability specifications of the system, the hardware components are integrated and assembled on a printed circuit board. Just like the hardware, the board design of the transmitter and receiver units are identical (as the units are able to switch between transmit and receive modes).

Schematic capture:

The first step in board design, after hardware selection, is schematic capture. Autodesk EAGLE provides powerful yet easy to use PCB design software, proving adequate for this system's board design. The final schematic of the Wireless Audio Bridge is shown in Figure XI. The schematic is divided into six functional sections: ProMicro, Audio, BTNs/LEDs, Serial, Power, and GPIO.

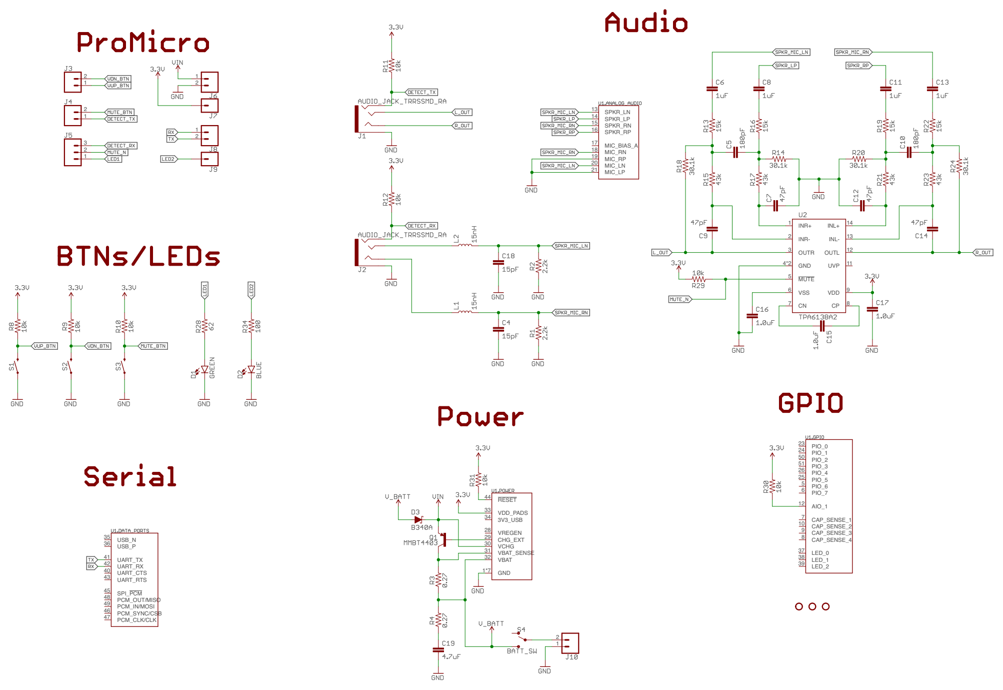


FIGURE XI

WIRELESS AUDIO BRIDGE SCHEMATIC

The three "main" components of the system (BC127, TPA6138A2, and ProMicro) are shown as U1, U2, and connectors J3-J9 respectively. As the ProMicro is its own separate board, it will be "stacked" on top of the Wireless Audio Bridge main board. The ProMicro's schematic is shown in Figure XII. The other components (along with component values) in the schematic are selected to support these key components.

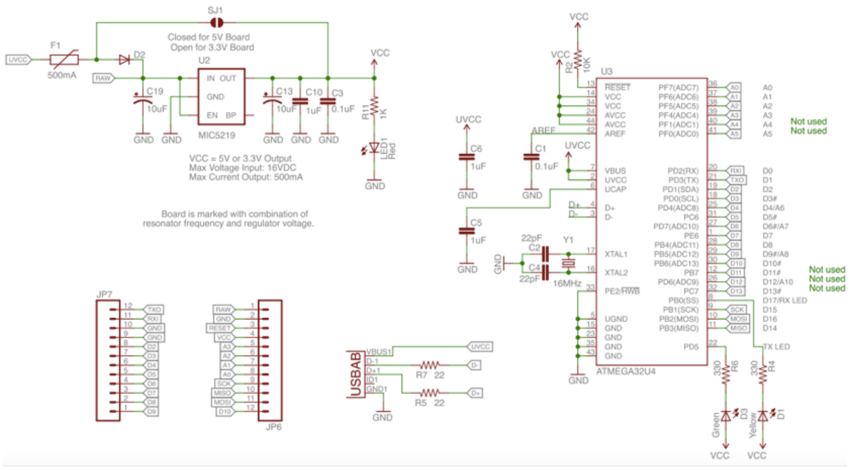


FIGURE XII

PROMICRO SCHEMATIC

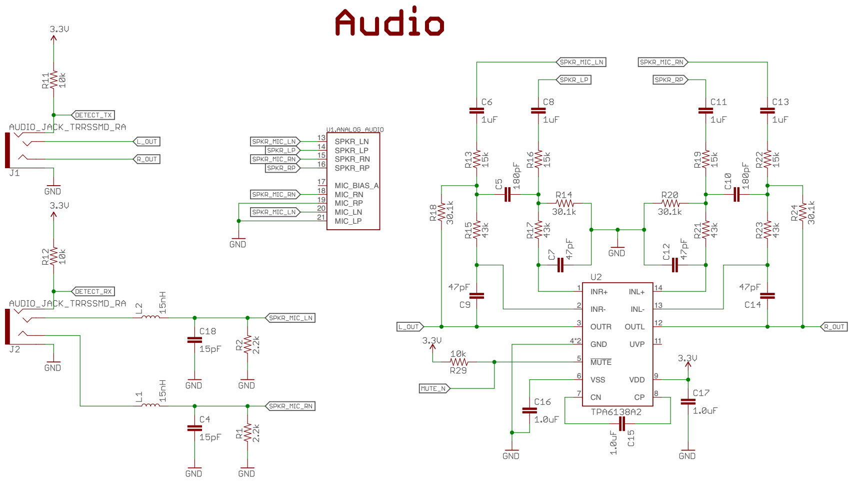


FIGURE XIII

WIRELESS AUDIO BRIDGE SCHEMATIC - AUDIO

The audio portion of the schematic is shown in Figure XIII. The headphone driver, TPA6138A2, is represented as U2 along with supporting circuitry. As per the TPA6138A2 datasheet, the gain setting resistors are selected to provide a gain of 2 for each channel. Input resistors and capacitors selected result in a (high-pass) cutoff frequency of 10.6 Hz defined by the following equation:

$$f_{cIN} = \frac{1}{2\pi R_{IN} C_{IN}}$$
$$R_{IN} = 15\ k\Omega$$
$$C_{IN} = 1\ uF$$
$$f_{cIN} = \frac{1}{2\pi(15\ k\Omega)(1\ uF)} = 10.6\ Hz$$

Values of supply capacitors are chosen as the datasheet suggests. An active-low MUTE input allows the TPA6138A2 to be muted and unmuted. With a pull-up resistor to supply (3.3 V), the signal can be toggled by the ProMicro.

The headphone audio jacks selected is a female 3.5mm TRS audio connector, allowing any 3.5mm male connector to be inserted. These connectors have an additional pin (aside from left, right, and ground pins) which can be used for insert detection. This pin is pulled up high via a pull-up resistor, allowing the ProMicro to detect an insert event. One connector is used for input, and the other for output. The path from the input connector to the audio inputs of the BC127 module contains circuitry to appropriately filter the input signal. The BC127 audio output routes to the input of TPA6138A2 and its output directly connects to the output connector. To enable the "audio sharing feature" the audio input signals are also routed to the input of the TPA6138A2. This allows audio to output when the unit is in transmitting mode.

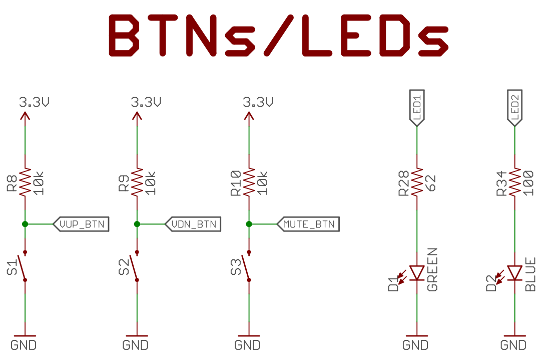


FIGURE XIV
WIRELESS AUDIO BRIDGE SCHEMATIC - BTNS/LEDS

The buttons and LEDs used in each unit is shown in Figure XIV. Two buttons control volume (volume up and down) and a third button controls the mute/unmute function. All three buttons are connected as ground-asserting switches, resulting in active-low signals. These signals route as digital inputs to the ProMicro. Two LEDs indicate the current mode of the particular unit; a green LED indicates receive mode and a blue LED indicates transmit mode. The series resistor for each LED is chosen based on the LED diode's forward voltage and current using the following equation:

$$R_{series} \approx \frac{3.3V - V_F}{I_F}$$

Both LEDs are controlled by the ProMicro as digital outputs.

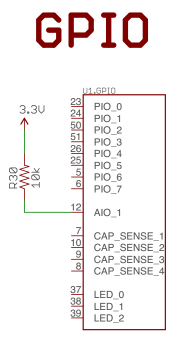


FIGURE XV

WIRELESS AUDIO BRIDGE SCHEMATIC - GPIO

Figure XV shows the GPIO portion of the BC127 module. None of the BC127 GPIO capabilities (PIO, CAP_SENSE, and LEDs) are used in this system, thus are left as no connects. The only I/O connected is AIO_1, which is pulled up to 3.3V. This allows the BC127 module to charge a battery without the need of a thermistor in the architecture.

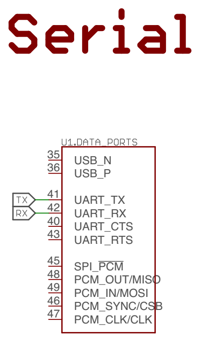


FIGURE XVI

WIRELESS AUDIO BRIDGE SCHEMATIC - SERIAL

The only serial interface used in this architecture is UART as shown in Figure XVI. The TX and RX signals are routed directly to the RX and TX pins (respectively) of the ProMicro.

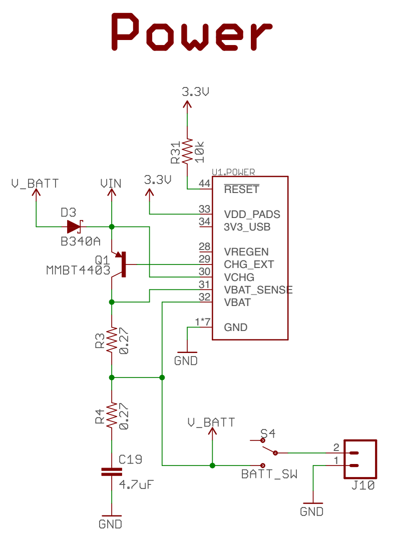


FIGURE XVII

WIRELESS AUDIO BRIDGE SCHEMATIC - POWER

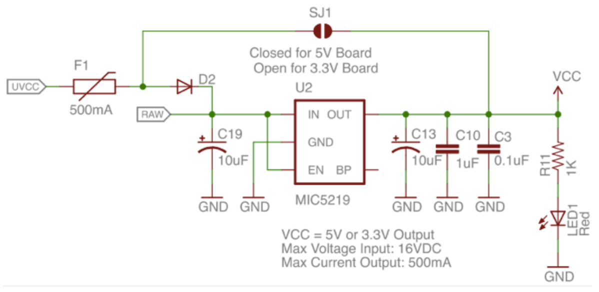


FIGURE XVIII

PROMICRO SCHEMATIC - POWER

The power related circuitry of the Wireless Audio Bridge (WAB) and ProMicro are shown in Figure XVII and Figure XVIII respectively. Power rails VIN of the WAB and RAW of ProMicro are tied together and serve as input to the MIC5219 3.3 V regulator. VCC, the output of the regulator, of ProMicro is connected to 3.3V of WAB supplying the 3.3 V rail for the main supply of the system. UVCC of ProMicro is the 5 V input coming from the micro-USB connector. V_BATT of WAB is the battery input and can act as the input to the regulator (and supply the entire system) if the 5 V UVCC is not applied (no micro-USB inserted). A high-rated diode between V_BATT and VIN prevents any back current into the battery from the 5 V input (if micro-USB inserted)

Additional components are required for the BC127 to properly charge a connected battery as seen in Figure XV. The BC127's CHG_EXT controls the external PNP transistor to control the battery charge current. A 270 mΩ sense resistor allows the BC127 module to monitor both battery voltage and charge current used to properly charge the battery. A series RC circuit is connected at the positive terminal of the battery to slowly charge up the capacitor once a battery is connected. Switch S4 is a SPDT (single pole dual throw) switch, enabling quick user connect and disconnect of the battery. Connector J1 is a two-terminal standard lithium-ion battery connector. The battery chosen is a 3.7 V 400 mAh rated lithium-ion battery. The 400 mAh capacity allows for a long audio streaming time while not being to physically large.

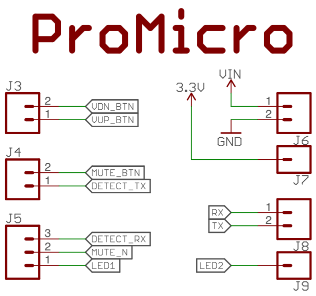


FIGURE XIX
WIRELESS AUDIO BRIDGE SCHEMATIC - PROMICRO CONNECTION

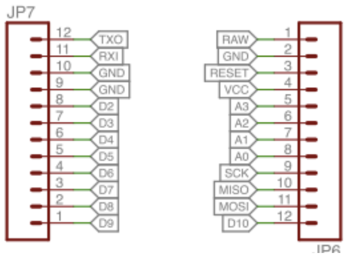


FIGURE XX
PROMICRO SCHEMATIC - CONNECTOR

The connection between the WAB and ProMicro boards is shown in Figure XIX. Not all pins of the ProMicro are used in this system; unused pins are left as no connect (and don't require a connector). As the ProMicro board is intended to be "stacked" on top of the WAB board, the WAB pinout of the ProMicro must be carefully and purposely selected based on the ProMicro's pinout, shown in Figure XX. Power related nets (VIN, GND, 3.3V) are tied directly to the ProMicro's corresponding nets (RAW, GND, VCC). All other I/O are chosen based on the function requirement of each signal and the function of each ProMicro I/O. To allow serial communication to a host computer (for debugging) as well as UART communication to the BC127 module, the BC127 UART interface does not use the TXO and RXI pins of the ProMicro. Rather, a "SoftwareSerial" library will be used to allow serial communication on another set of pins of the ProMicro (as seen in connector J8 of Figure XIX).

Layout:

The design of the layout starts with a placement study of the components in order to define the physical dimensions of the board. With the intention of using a 2-layer board (two electrical signal layers), components must be placed purposefully to enable good routing.

The packages and dimensions of the main components (BC127 module, headphone driver, ProMicro) are fixed, however, most of the other components have multiple package options. An 0603 package size for the passives (resistors, capacitors, and inductors) prove adequate as they are not too large (for space constraints) and not too small (for easy hand soldering). LEDs are also chosen with an 0603 package size. Right-angle buttons are selected for the volume and mute buttons. The pnp transistor and battery diode use standard SMD package sizes. A through-hole type button is chosen for rigidity against multiple presses. The power switch used is a common surface mount type switch (with through-hole inserts for rigidity). The audio connectors are a standard package for 3.5mm female audio connectors. The battery connector is a standard two-pin connector for a lithium ion battery. Three drill holes are added to allow the use of standoffs.

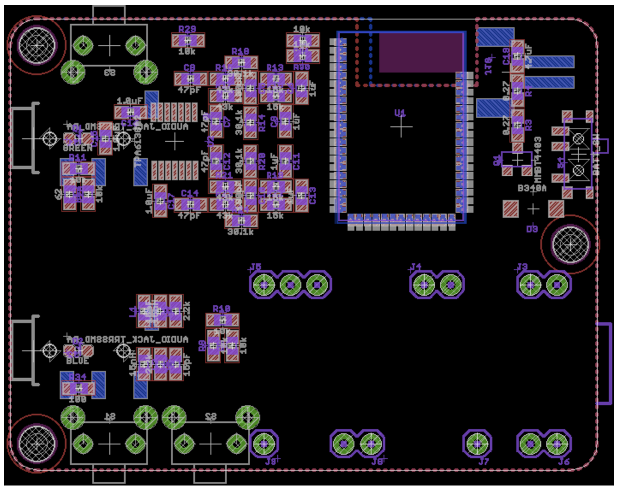


FIGURE XXI

WIRELESS AUDIO BRIDGE – FINAL COMPONENT PLACEMENT

Figure XXI shows the component placement of the final board design. The dimensions of the board come out to be 2.2 x 1.7 in. Aside from the power switch, the components intended for user access (connectors and buttons) are placed on the bottom side of the board. All other components are placed on the top.

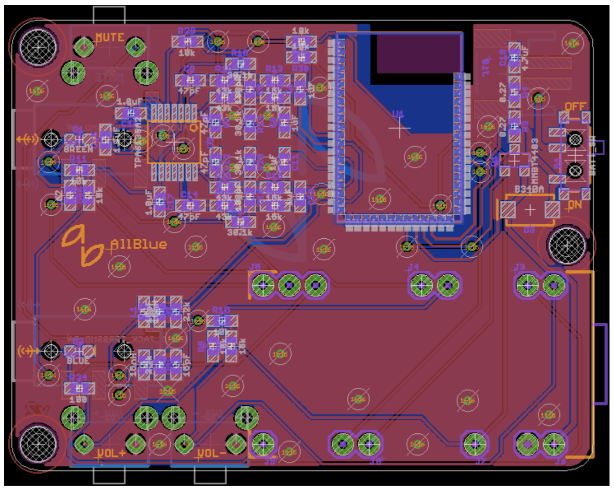


FIGURE XXII

WIRELESS AUDIO BRIDGE – FINAL BOARD LAYOUT

The final layout, including routing, is shown in Figure XXII. Digital signals are routed with 8 mil traces, power signals with 20 mil. Top to bottom layer vias have a drill size of 20 mil, being used only where needed. A ground pour is applied on both top and bottom layers. Ground vias are spread around the board to avoid large ground loops. The integrated antenna of the BC127 lies in the top right corner of the module; a ground keepout layer is used in that area to prevent antenna detuning. Appropriate silkscreen is used nearby connectors and buttons for easy indication.

Software Design

The software design lies in the embedded programming of the ProMicro's ATMega32U4 microcontroller. The Arduino IDE is used to write and upload the program onto the ProMicro via its micro-USB. There are two main tasks or workflows in developing the code for the Wireless Audio Bridge: establishing functions to properly communicate with the BC127 module, and creating the "state machine" program to implement all the required functionality of this system's transmitter and receiver.

The BC127 module is the component in the system mainly intended to send (or receive) an audio signal to (or from) another BC127 module. To enable this behavior, the BC127 module must be sent appropriate commands at the appropriate times. This is the job of the microcontroller; it must properly initialize the BC127 module, send appropriate commands dependent on the current state, and receive responses from the module and react accordingly.

First, a set of basic functions must be established in order to send commands and receive responses using the UART interface. Sparkfun provides a library to do just this, however, the code requires some modification to comply with the BC127 module's firmware. The two basic functions required are command issuing and parameter setting functions. With these two functions formed and well written, proper control over the BC127 module can be established. All other functions (for pairing, music commands, volume control, battery charging, etc.) are built off of these two basic functions.

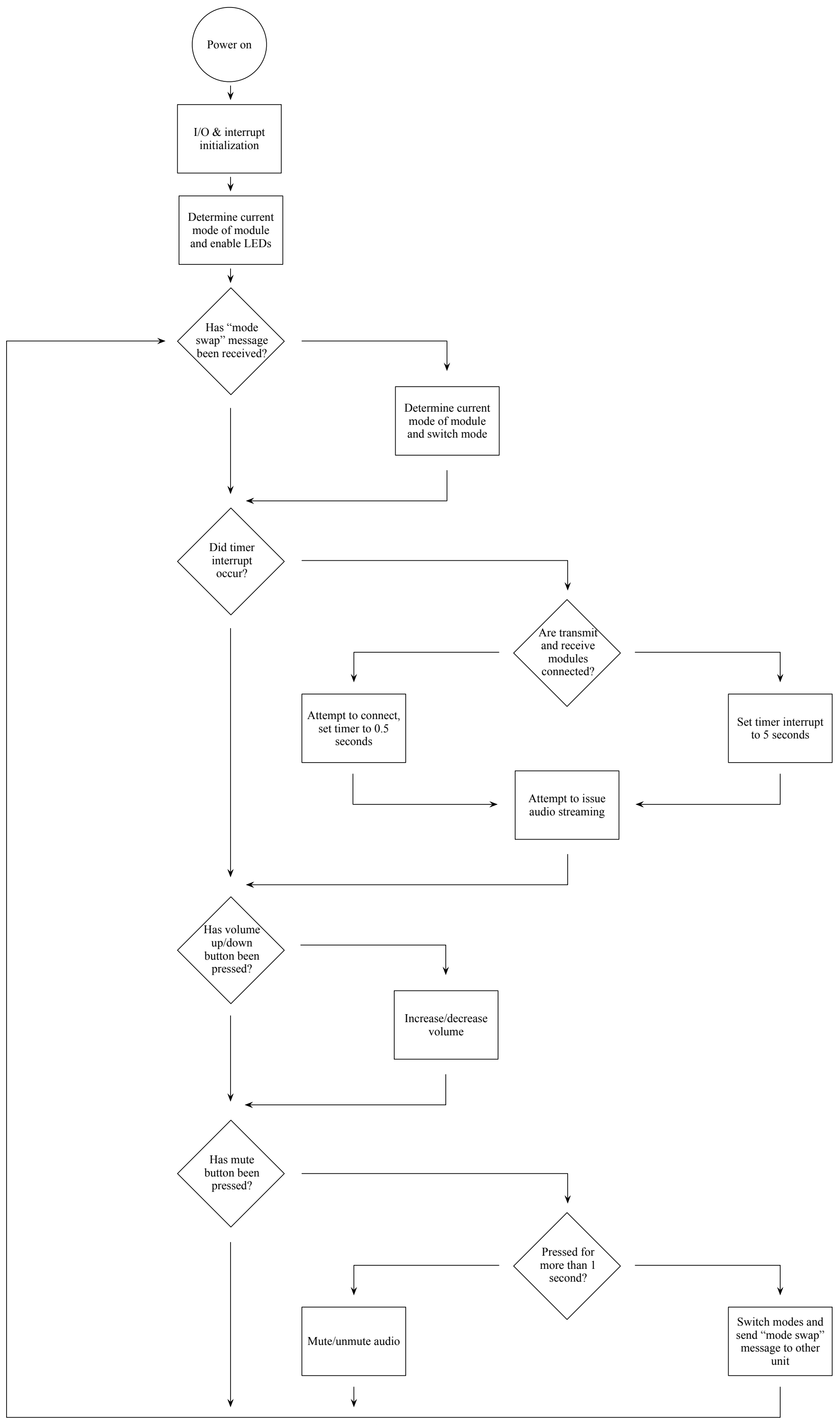

Power on
I/O & interrupt initialization
Determine current mode of module and enable LEDs
Has "mode swap" message been received?
Determine current mode of module and switch mode
Did timer interrupt occur?
Are transmit and receive modules connected?
Attempt to connect, set timer to 0.5 seconds
Set timer interrupt to 5 seconds
Attempt to issue audio streaming
Has volume up/down button been pressed?
Increase/decrease volume
Has mute button been pressed?
Pressed for more than 1 second?
Mute/unmute audio
Switch modes and send "mode swap" message to other unit

FIGURE XXIII

HIGH-LEVEL SOFTWARE FLOWCHART

A high-level flowchart of the final program of the system is shown in Figure XXIII. After power is applied to the system, I/O and interrupts are initialized. A serial port is configured for debugging, inputs and outputs (along with their interrupts) are set, and the timer interrupt is configured. The mode of the BC127 module is determined (either transmit or receive) and the corresponding LED is enabled. After this initialization process, interrupts are enabled and the program enters the main loop.

The main loop is written as series of conditional statements, each performing some action if the particular condition is met. The order of the conditional blocks is not particularly important, as the program will continue to loop. All conditional blocks are triggered off a particular hardware interrupt (button or timer) except for the "mode swap" block. The ISRs (interrupt service routine) for each interrupt simply sets a flag variable. The corresponding conditional block performs some action, then resets the flag variable.

The mute conditional block performs an action dependent on the duration of the button press. For a mute button press less than 1 second, the headphone driver is muted or unmuted (dependent on previous state). If the button is held down longer than 1 second, a "mode swap" is initiated. The "mode swap" swaps the unit's mode (from transmitter to receiver or vice versa), and sends a message to the other connected unit to initiate its own mode swap. This allows either unit, transmitter or receiver, to issue the mode swap. The mode swap message the unit sends is through the AVRCP profile. The AVRCP profile allows for remote control between connected devices, specifically for volume, track, and play/pause control. For these BC127 modules, when volume is changed on either unit, a notification of this change is sent to the other unit. The "mode swap" functionality takes advantage of this notification behavior. When the mute button is held for longer than 1 second, a command to change the volume to 0 (no volume) is sent to the BC127 module. This command is chosen as it is not required for the system's functionality (for providing no volume, the headphone driver is instead muted). After the other unit's BC127 module receives this 0 volume notification, the ProMicro is able to detect this notification (explained later on), and the ProMicro then issues the mode swap of its unit.

The "mode swap" conditional block is the only conditional block not triggered by an interrupt. This purpose of this conditional block is to receive the mode swap message (volume set to 0) from the other unit (if initiated). To do this, the UART serial data from the BC127 module is continuously monitored. If the serial data equals the string "ABS_VOL 11 0\r" (the message received when the volume is set to 0), the ProMicro then transitions into changing the mode of the BC127.

The volume up and down conditional blocks (triggered of the corresponding button press) sends a volume up or down command to the BC127. Due to the mode swap "takeover" of the volume level 0 command, the headphone driver is instead muted once the volume reaches below level 1.

A timer based conditional block is used to attempt connection to the other unit (if not connected) and start audio streaming. Once the timer interrupt occurs, a function returns the status of the connection state. If the units are connected, the timer is set to a slow frequency (5 seconds) and the LED corresponding to the unit's mode is enabled. If the units aren't connected, a connect function is initiated and the unit attempts to pair with the other unit. The timer is set to a faster frequency (0.5 seconds) to continually and quickly attempt to connect to the other unit. The LED corresponding to the unit's mode also blinks at this 2 Hz rate, indicating the units are currently not connected.

All these conditional blocks result in a program ensuring the transmitting and receiving units are connected, streaming audio, and respond to user inputs as intended.

Package Design

A case is used to enclose the board and its components for both functional and aesthetic purposes. As each unit is inherently an electrical system, energy in the form of electricity is accessible at the board level. A package around the board ensures no energy is transferred from outside objects inward or vice versa. The case is also used to provide an appealing device, while at the same time providing proper access to certain components (buttons, connectors, switches).

The overall shape of the case is a rectangular box, with cutouts for the connectors, buttons, and switch. The material chosen is acrylic with a frosted clear finish. Acrylic plastic can be easily machined and sanded. A frosted clear color allows for the LEDs to visibly shine through the case. The final case with a unit enclosed is shown in Figure XXIV. Powered on, connected, and enclosed transmitting and receiving units are shown in Figure XXV.

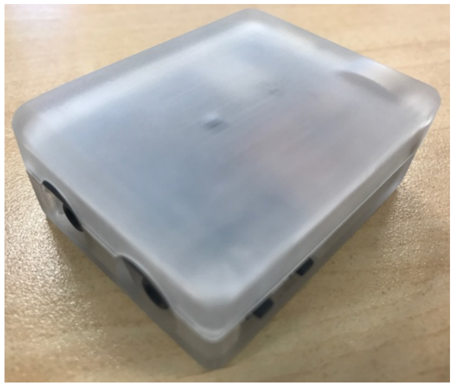


FIGURE XXIV
WIRELESS AUDIO BRIDGE CASE

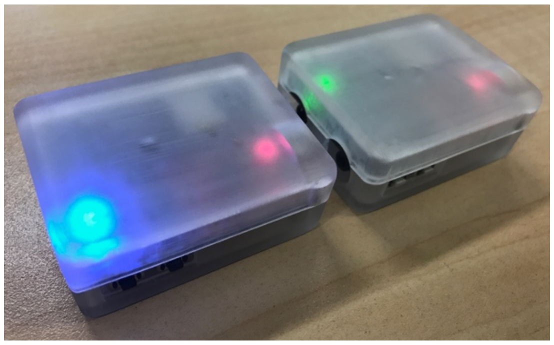


FIGURE XXV
PACKAGED TRANSMITTING AND RECEIVING UNIT

Chapter 6: Testing, Development, and Integration

An iterative approach was taken in the development of this system, resulting in overlap between project phases. After the initial design concept was established, results and findings from testing and developing on prototypes were used to improve on the design. Integrating components together brought up unforeseen issues, also resulting in design adjustments. These design changes required further testing and verification, and so on and so forth. This process continued until the system specifications were sufficiently met.

Prototyping

The initial testing of this project focused on the BC127 module. The intention was to explore its features, verify its functionality, and understand its usage. Multiple development boards exist on the market for these certain reasons; the Sparkfun Purpletooth Jamboree provides a full-function board with the BC127 module shown in Figure XXVI.

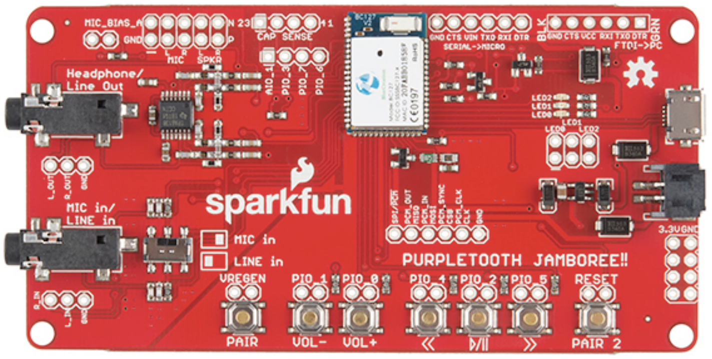


FIGURE XXVI

SPARKFUN PURPLETOOTH JAMBOREE BC127 DEVELOPMENT BOARD

The Purpletooth Jamboree includes access to all the BC127 modules pins as well as supporting components to use this as a standalone board. Usage of a serial terminal and a serial-to-USB converter, communication between the BC127 module and a host computer can be established. After evaluating the BC127 and verifying its functionality, a prototype of the Wireless Audio Bridge began to take form. The ProMicro was connected and brought up to interact with the Purpletooth Jamboree board, buttons and switches were added for triggering inputs, and transmitting and receiving units were conceived. During this development phase, the implementation of many of the system's specifications and features had to be rethought and reengineered.

In the initial design concept, the mode swapping feature was intended to be one-sided, meaning that each unit would require a manual switching of modes. It was found possible to issue the mode swap on just one of the units (which then notifies the other unit) only after exploring the behavior of the BC127 module.

Figuring out a method to allow audio output on the transmitting unit was also dependent on deliberate testing. Initially, this functionality was thought to be only possible using a low resistance mux (multiplexor) to switch between the audio source input and BC127 module's audio output paths into the headphone driver's input. However, because the BC127 module's audio output signals are left as open (high-Z) when not being driven, the mux was proven unnecessary.

PCB Revisions

The layout of the board went through a couple revisions before becoming the final design. Two revisions of the design were sent to a PCB manufacturer as shown in Figures XXVII and XXVIII.

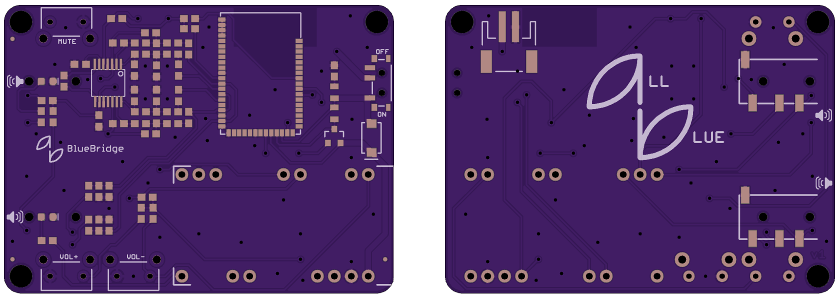


FIGURE XXVII

WIRELESS AUDIO BRIDGE PCB REVISION 1

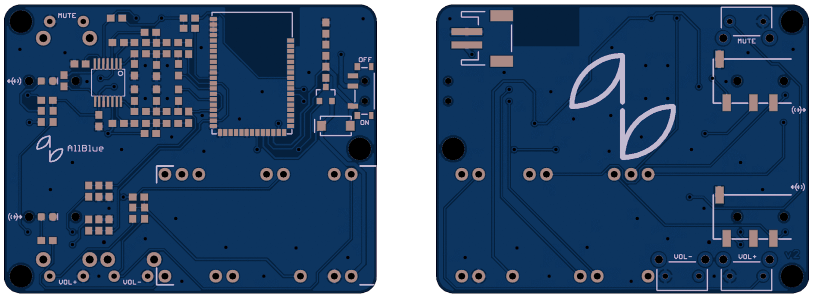


FIGURE XXVIII

WIRELESS AUDIO BRIDGE PCB REVISION 2

After receiving, assembling, and testing version 1 of the design, the board functioned as intended (verifying the schematic design). Evaluating the board from a user interface perspective revealed potential improvements of the placement of components. This initiated the design of a second version of the board. Moving from version 1 to version 2, no major schematic changes were needed, only mechanical related adjustments. The width of the board was slightly decreased for a smaller unit footprint. Buttons were moved from the top of the board to the bottom. An unused drill hole for the ProMicro board was removed. The battery connector was rotated to a more suitable angle. The third standoff drill hole was moved toward the center of the board for increased stability. The color of the overlay switched from purple to blue, and the silkscreen underwent a few changes. Figure XXIX shows one unit fully assembled on the version 2 board.

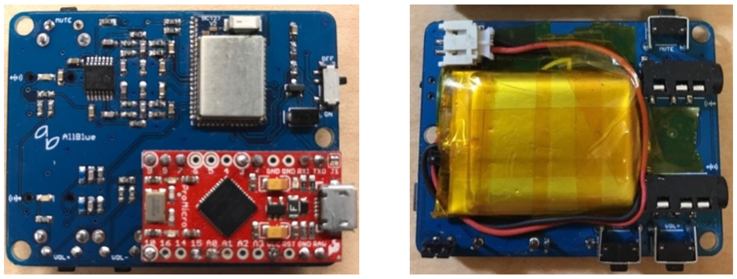


FIGURE XXIX

WIRELESS AUDIO BRIDGE ASSEMBLED UNIT

Software Development

On the software side of things, the code was continually refined as the hardware developed. Discoveries of the capabilities and limitations of the ProMicro (from a programming standpoint) had effects on the hardware as well, specifically the ProMicro pinout. Defining and testing I/O of the ProMicro was the first task on the software side. Finalizing the ProMicro pinout early on was important in preventing any potential schematic changes.

After finalizing the pinout, the main software flow is established. The program's high-level purpose is to perform some action based on a certain event. If no events occur, the program remains idle. This behavior is made possible by the use of hardware interrupts. A timer interrupt is used to check if the transmitting and receiving unit are connected and streaming audio. All other interrupts are used for button detections. These are pin change interrupts, which trigger an event if a particular pin changes state. After any of these interrupts are triggered, the program will run through a certain sequence of code. The development of this code involved a combination of planning, function creation, and mostly trial and error.

To activate the mode swapping functionality, a couple methods were explored. The first method was to take advantage of the insert/removal detection capability of the 3.5 mm audio connector used. With this capability, the ProMicro could detect when an audio jack is inserted into either connector (input or output). When an audio jack is inserted into the input connector, the mode of the unit could be switched to or remain the transmitter. If inserted in the output connector, the mode could be switched to or remain the receiver. While this method could work, it could become somewhat confusing and problematic if audio jacks are inserted into both connectors of one unit. Thus, the method for initiating mode swapping was chosen to be via button activation. Instead of adding a fourth button to the design (requiring a hardware change), a long-press on the mute button is chosen to activate mode swapping. The implementation of this feature entailed a minor software change.

Package Construction

In constructing a package to enclose the board and its components, various methods and means to do so were explored. The method chosen must provide a balance of cost, design/construction time, and type of material. The most commonly used method for this type of application is 3D printing. 3D printing allows for a cheap, fast, yet rigid solution for packaging. The designed 3D CAD model of the package is shown in Figure XXX.

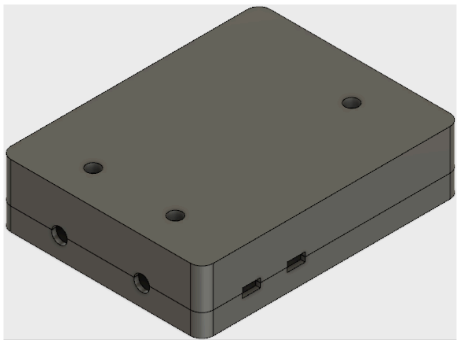


FIGURE XXX
PACKAGE 3D CAD MODEL

Unfortunately, due to the lack of both time and 3D printing resources, the 3D print was not able to be made. The runner-up method was machining a package out of plastic, which was the final method used to construct the package. The 3D CAD model was used as a rough guideline for establishing the shape of the package. For each unit, two 1" acrylic blocks were milled, drilled, and sanded to create two enclosing halves shown in Figure XXXI.

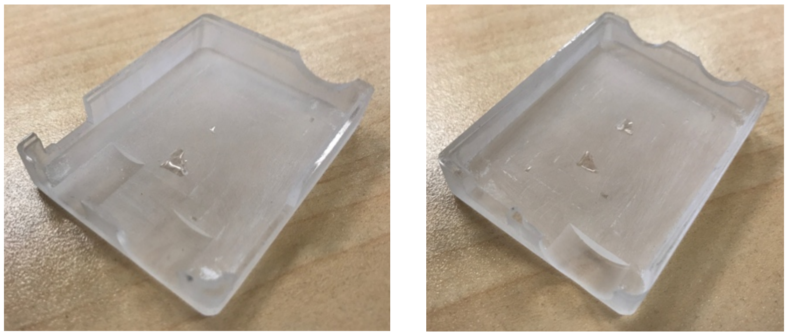


FIGURE XXXI
PACKAGE TOP AND BOTTOM HALF

Test Cases

To verify the system's functionality, a series of test cases were developed and executed to ensure fulfillment of the specifications of Table II. Table VII shows the tests and results of each.

TABLE VII
WIRELESS AUDIO BRIDGE TEST CASES

Specifications	Test Description	Result
The transmitting and receiving unit communicates using Bluetooth technology.	Both transmitting and receiving unit uses a BC127 module, which uses Bluetooth for both audio streaming and communication.	Pass
The transmitting and receiving unit communicates and streams audio within a range of at least 30 feet.	During audio streaming, the transmitting and receiving unit were incrementally separated from 0 feet to 50 feet. Audio streaming performed as expected at 30 feet.	Pass*
The transmitting unit and receiving unit stream stereo audio signals.	An audio device with stereo output was connected to the transmitter. An audio device with a stereo input was connected to the receiver. Both left and right channels were heard.	Pass
The audio latency between the transmitting and receiving unit does not exceed 500 ms.	An audio signal with a 1 Hz beat was used as the audio source into the transmitter. Using a stopwatch, the time delay between beats on the transmitter and receiver side was recorded (multiple times). The average time was 220 ms.	Pass
The transmitting and receiving unit establishes audio communication in less than 5 second after power-up.	After applying power to the transmitting and receiving units simultaneously, a stopwatch was used to record the connection time (once audio started playing). The average time was less than 5 seconds.	Pass
The receiving unit has an adjustable volume control between no volume and a maximum volume level, as well as mute/unmute capability.	The volume up, volume down, and mute buttons were asserted many times, in and out of order. Each button function performed as expected.	Pass
The transmitting and receiving unit uses standard 3.5 mm audio ports (male and/or female) to interface with audio devices.	3.5 mm female audio connectors are used for both audio inputs and outputs. Various 3.5 mm audio devices (input and output) were tested with the transmitting and receiving units.	Pass
The dimensions of each unit do not exceed 3" x 2" x 1".	With the case, the final dimensions of the unit are 2.5" x 2" x 1". Without the case, the dimensions of the unit are 2.2" x 1.7" x 0.5".	Pass
Each unit operates for at least 4 hours of audio streaming on one battery life.	After fully charging both transmitting and receiving units (with audio streaming at 75% volume), the receiving unit battery depleted after 6.5 hours.	Pass
The battery in each unit has recharging capability.	After depleting the battery of a unit, a 5 V (up to) 500 mA source was applied to the charging port. The battery eventually reached max capacity and the unit could be powered.	Pass

Specifications	Test Description	Result
The total parts cost does not exceed $60 per unit.	The final BOM cost of each unit is $56.	Pass
The transmitting and receiving unit completely encloses its electrical components.	Except for the connectors and buttons, the case encloses all the circuitry from the user.	Pass
The transmitting unit outputs audio directly from the audio source.	An audio input device (headphones) was inserted into the transmitting unit. While streaming audio, the audio was outputted through the audio input device.	Pass
Both units in the system are capable of switching between transmit and receive modes.	On either the transmitting or receiving unit, the mute button was held down for more than 1 second (initiating mode swap) while streaming audio. The audio streaming direction swapped.	Pass
Two different colored LEDs indicate the current transmit or receive mode of each unit.	With a unit in transmitting mode, the blue LED is lit. With a unit in receiving mode, the green LED is lit.	Pass

* No loss of connection was observed in an open area with line of sight between units. Around corners and behind obstacles decreased the range.

Chapter 7: Conclusions and Future Work

The Wireless Audio Bridge fulfills the needs and provides the functionalities as intended in Chapters 1 and 2. With the creation of this system, the transmitting and receiving unit together enables a wireless listening solution for many audio devices that previously didn't have wireless functionality. An infinite amount of approaches can be used to implement this type of system; the design decisions of the Wireless Audio Bridge were based off of cost, time, available resources, implementation ability, and desired features. Despite the current state of the system meeting the defined specifications and requirements, improvements to the Wireless Audio Bridge in many areas can make the system a better overall product.

Cost is a big area of potential improvement of the Wireless Audio Bridge. The current BOM cost for each unit comes out to be around \$60. This results in a total cost of \$120 for both a transmitting and receiving unit. Many approaches can be used to reduce the cost of the system. Purchasing components in bulk can significantly reduce the price per unit of many components. This would only be sensible if the system were to be mass produced. The Sparkfun ProMicro board, costing a third of the total cost of the system at \$20, can be replaced with components that provide the same function. The ProMicro board was used in the system due to time and development reasons, however, purchasing and designing individual components (ATMega32U4, 3.3V regulator, micro-USB connector, etc.) onto the Wireless Audio Bridge board directly would greatly reduce cost.

There is opportunity to add more functionality and features to the Wireless Audio Bridge through software. The current system is taking advantage of only a portion of what the BC127 Bluetooth module has to offer. The BC127 is capable of streaming audio using the TWS profile. TWS (True Wireless Stereo) allows two BC127 modules to act as the receiving unit. One receiving unit outputs the left audio channel, and the other receiving unit outputs the right channel. This enables a "true" wireless stereo listening experience. Another feature the Wireless Audio Bridge does not currently support is the option to connect a unit directly to a Bluetooth enabled audio device. This feature would be used in a scenario where a user has either a Bluetooth compatible audio source (laptop, cellphone, etc.) or an audio sink (headphone, speaker, etc.). The user would be able to pair their Bluetooth enabled device to a unit, and connect a non-Bluetooth enabled device directly to the unit.

The size of each unit could be reduced to improve the overall portability of the system. The current dimensions of each unit, without the casing, is roughly 2.2" x 1.7" x 0.5". This size is comparable to the size of a small mint box. To make the unit even smaller, a few approaches can be taken. First, the package sizes of the components could be of smaller size. The discrete components used on the board have a package size of 0603. This package size was chosen to allow for easier hand soldering and rework. Choosing a smaller package size would allow the footprint of the board to decrease. Rearranging some of the components could also create some room to reduce the board's size. Most components were placed on the top of the board, but can fit on the bottom side to make use of as much surface area as possible.

The audio streaming quality has room for improvement. The BC127 module's converters (ADCs and DACs) have a resolution of 16 bits. Some audio files are naturally of 16-bit resolution, but more prevalent are high-resolution audio files with resolutions of 24 or even 32 bits. Streaming high-resolution audio with the current Wireless Audio Bridge will result in a noticeable decrease in quality. Instead of using the BC127's internal converters, it is possible to implement higher quality external converters for the analog audio signal's conversion to digital and back to analog. This would require major hardware and software revisions of the design.

From concept creation to final design, the development of the Wireless Audio Bridge required many obstacles to be encountered, risks made, and problems solved. Developing a system that is as marketable as it is functional necessitated a combination of high-level and low-level critical thinking. Merging hardware and software worlds naturally resulted in an iterative approach to the development of the system. All these experiences provided lessons that could only be learned from a project of this nature and scale.

References

[1] J. Watkinson, *The art of digital audio*, 3rd ed. Oxford: Taylor & Francis, 2011.

[2] Floros, A.; Tatlas, N.-A.; Mourjopoulos, J., "A high-quality digital audio delivery bluetooth platform," in *IEEE Transactions on Consumer Electronics,* vol.52, no.3, pp.909-916, Aug. 2006

[3] Microchip Technology, “RN52 Bluetooth Audio Module”, RN52 datasheet, Sept. 2015

[4] J. Linde and B. Tucker, "Audio transfer using the Bluetooth Low Energy Standard", U.S. Patent 8849202, Sept. 2014.

[5] He Jian; Huang Zhang-qin; Hou Yi-bin; Dong Yu-min, "Point-to-Multipoint Stereo Audio Transmitting System Based on Bluetooth," in *2010 International Conference on Communications and Mobile Computing (CMC)*, vol.3, no., pp.323-328, 12-14 April 2010

[6] Hangil Moon; Namsuk Lee; Hyunwook Kim; Sanghoon Lee, "Low latency audio coder design for bluetooth and bluetooth low energy," in *Consumer Electronics (ICCE), 2015 IEEE International Conference on Consumer Electronics (ICCE)*, vol., no., pp.138-141, 9-12 Jan. 2015

[7] Yang Yue; Xie Xiang; Wei Yaodu, "A novel objective method for evaluating the quality of streaming audio," *2009. IC-BNMT '09. 2nd IEEE International Conference on Broadband Network & Multimedia Technology*, vol., no., pp.555-559, 18-20 Oct. 2009

[8] R. Heydon, *Bluetooth low energy*. Upper Saddle River, N.J.: Prentice Hall, 2013.

[9] Wikipedia, "Phone connector (audio)", 2016. [Online]. Available: https://en.wikipedia.org/wiki/Phone_connector_(audio). [Accessed: 31- Jan- 2016].

[10] M. Mallinson, "Digital vs. Analog Volume Control", *esstech.com*, 2011. [Online]. Available: http://www.esstech.com/files/3014/4095/4308/digital-vs-analog-volume-control.pdf. [Accessed: 31- Jan- 2016].

[11]"BC127 Radio Dual Mode - BlueCreation", *Bluecreation.com*, 2017. [Online]. Available: https://www.bluecreation.com/product_info.php?products_id=38. [Accessed: 31- Jan- 2016].

[12] Bluetooth.com, "Understand More About Bluetooth Technology | Bluetooth Technology Website", 2016. [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology. [Accessed: 22- Feb- 2016].

[13] Techopedia.com, "What is Audio Streaming? - Definition from Techopedia", 2016. [Online]. Available: https://www.techopedia.com/definition/6067/audio-streaming. [Accessed: 22- Feb- 2016].

[14]D. Sellers, "Columnist: Bluetooth still 'expensive, limited'", *Macworld*, 2016. [Online]. Available: http://www.macworld.com/article/1004545/bluetooth.html. [Accessed: 22- Feb- 2016].

[15] Mcsquared.com, "Sound Systems: Mono vs. Stereo", 2016. [Online]. Available: http://www.mcsquared.com/mono-stereo.htm. [Accessed: 23- Feb- 2016].

[16] R. Ford and C. Coulston, *Design for Electrical and Computer Engineers*, McGraw-Hill, 2007, p. 37

[17] *IEEE Std 1233, 1998 Edition*, p. 4 (10/36), DOI: 10.1109/IEEESTD.1998.88826

[18] M. Shoemake, "Wi-Fi (IEEE 802.11b) and Bluetooth Coexistence Issues and Solutions for the 2.4 GHz ISM Band", *Texas Instruments*, 2001. [Online]. Available: http://focus.ti.com/pdfs/vf/bband/coexistence.pdf. [Accessed: 26- Feb- 2016].

[19] UPPSALA UNIVERSITET, "Smart, ecofriendly new battery to solve problems - Uppsala University, Sweden", 2014. [Online]. Available: http://www.uu.se/en/media/news/article/?id=3707. [Accessed: 26- Feb- 2016].

[20] J. Faludi, "Environmental Impacts of 3D Printing | Sustainability Workshop", *Sustainabilityworkshop.autodesk.com*, 2013. [Online]. Available: http://sustainabilityworkshop.autodesk.com/blog/environmental-impacts-3d-printing. [Accessed: 26- Feb- 2016].

[21] Ieee.org, "IEEE IEEE Code of Ethics", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 26- Feb- 2016].

[22] J. DeMarco, "Principlism and moral dilemmas: a new principle", *Journal of Medical Ethics*, vol. 31, no. 2, pp. 101-105, 2005.

[23]"TPA6138A2 DIRECTPATH™ Headphone Driver with Adjustable Gain | TI.com", *Ti.com*, 2015. [Online]. Available: http://www.ti.com/product/TPA6138A2. [Accessed: 13- Mar- 2017].

[24]"Understanding the BC127 Bluetooth Module - learn.sparkfun.com", *Learn.sparkfun.com*, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/understanding-the-bc127-bluetooth-module. [Accessed: 13- Mar- 2017].

[25]"Pro Micro & Fio V3 Hookup Guide - learn.sparkfun.com", *Learn.sparkfun.com*, 2017. [Online]. Available: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide. [Accessed: 13- Mar- 2017].

[26]"PCB Design & Schematic Software | EAGLE | Autodesk", *Autodesk.com*, 2017. [Online]. Available: https://www.autodesk.com/products/eagle/overview. [Accessed: 22- Mar- 2017].

Appendix A: Senior Project Analysis

Project Title: Wireless Audio Bridge

Student's Name: Daniel Hodges **Student's Signature:**

Advisor's Name: Andrew Danowitz **Advisor's Initials:** **Date:**

1. Summary of Functional Requirements

This system provides wireless capability for audio devices with wired connections. The system consists of two units, a transmitter and a receiver. The transmitter connects to an audio device with an output (computer, portable audio player, cellphone, etc.) and transmits the audio signal wirelessly to the receiver. The receiver outputs the transmitted audio signal to an audio device with an input (headphones, speaker, etc.). Volume and mute controls adjust the amplitude of the audio output, and mode swap control swaps the mode of each unit.

2. Primary Constraints

One challenge associated with the implementation of this system involves the method used to transmit audio signals between units. There exist many modules currently in the market that support wireless data transfer using various technologies. Important factors to take into account include data rate, operational range, power consumption, and physical size of components. The selection of the module greatly affects the component selection for the rest of the system.

Limited financial and time investments impacted the approaches of the project implementation. The exploration of different methods and components used in the design of the system was constrained due to lack of budget and time. With more financial resources and time, additional iterations of the system could have occurred, resulting in further refinement of the product.

An important aspect in the implementation of the system was the understanding of selected components. The three main components used in this system (BC127 module, headphone driver, ProMicro) needed to be deeply explored to learn and understand the functionalities, requirements, and limitations of each. This consisted of reading datasheets, exploring forums, and thorough prototyping.

Chapter 2 translates these and additional constraints into system requirements and specifications.

3. Economic

As this system intends to market in the consumer products area, many economic impacts need consideration. Prior to profiting from the outputs of the product, the project requires financial investments. This project needs money to advance development, purchase parts, rent equipment, etc. The project uses many tools in the development of the product including test equipment, computers, 3-D printers, etc.

The initial costs in the system's lifecycle attribute to the conceptualization and development of the design. Many work hours go into defining the concept and exploring various means of implementation. After establishing the design, testing and building the system needs a first set of parts for development. After an achieved working system, prototyping and materials costs accrue to create a potentially marketable product.

At the start of the project, the costs come from development parts and labor. The cost for the required parts estimates to $495. The anticipated labor costs, based on number of work hours, resulted in $10,955 based on a $35/hr pay rate. The project's cost does not include test equipment as the development intends to take place in facilities with available test equipment resources.

If this project goes to market, the number of sales and price per unit determines the revenue. To properly determine how much this project can earn, the design and final bill of materials first needs establishment.

The final project emerged at the end of June 2017. Assuming a well-designed product, the system should not require maintenance or upkeep aside from replacing and/or recharging its battery. In order to address time management for this project, a Gantt chart shows the development of the project from January 2016 to June 2017 as seen in Figures III, IV, and V of Chapter 4. The next step in the project is the evaluation of the product from a marketability standpoint.

4. If manufactured on a commercial basis

The number of devices sold, manufacturing cost, purchase price, profit, and operating costs of the system determine potential manufacturability on a commercial basis . Dependent on product quality, marketability, and advertising, the number of devices sold per year can range anywhere from single to triple digits. An acceptable estimation, at least for the couple years on the market, can estimate to around 50 devices sold per year. To have market potential, the internal components of the devices uses printed circuit board technology which stays relatively inexpensive given the small number of components and the physical size and of the devices. The manufacturing cost estimates to around $10 per unit. To collect a profict, the purchase price of each device surpasses around $10 to $30 above the parts and manufacturing cost of each device. With the total cost of each unit around $60, the purchase price estimates to about $80. These estimations determine the profit per year estimation. Assuming a cost of $60 per unit and a purchase price of $80, at 50 devices sold per year the profit results in $1,000. The operating cost of the system also contributes to cost. The only operating cost has to do with recharging the unit's battery. Assuming the battery in the device lasts total 5 hours (while operating) and the cost per charge estimates to $1, the operating cost results in $0.20 per hour.

5. Environmental

For manufacturability, addressing impacts on the environment and usage of the earth's resources is necessary. This system uses a printed circuit board to mechanically support and electrically connect its components. The printed circuit board manufacturing process involves many chemicals and valuable materials used to develop the boards. Many of these materials discharge into the environment in forms of solid and liquid waste. However, the recycling process of these materials have and continue to improve over the years. This system can take advantage of 3D printing for manufacturing a package. Usage of 3D printing have beneficial impacts on the environment including reduction in fossil fuel (used for transportation of goods), less waste, and recyclablity [20].

If this project ever acheives mass production, it may require obtaining real estate or land for a manufacturing facility. This brings about various environmental impacts needing of consideration including impacts on various species, plants, and ecosystems.

6. Manufacturability

This system uses a printed circuit board to mechanically support and electrically connect its components. The printed circuit board manufacturing process continues to develop and improve, making it easier for anyone to submit designs and receive boards quickly and inexpensively. This plays a major role in the manufacturing of this system for prototyping as well as for the final design. As the dimensions of the device specify under 3" x 2" x 1" (including the battery), the circuit board's footprint must be minimized. The final dimensions of the layout came out to be 2.2" x 1.7" x 0.07".

The system must also have proper packaging to enclose its internal components while appearing aesthetically pleasing. The shape and material of the packaging directly affects the manufacturing process. Certain shapes and materials may require different machining processes, which have their own benefits and drawbacks in terms of both time and cost. Forming multiple design concepts and prototypes prior to settling on a final design allows for simpler manufacturability and an overall better product.

7. Sustainability

Assuming a well-developed design, maintenance of the completed system should not arise aside from recharging and/or replacing batteries. After the initial materials used to manufacture the device, the system only uses resources to replenish the device's batteries. From a sustainability perspective, there exist many options to use for energy storage in this product. Rechargeable batteries can store electrical energy generated by renewable energy sources such as wind, solar, and hydropower sources. New technologies have developed many "eco-friendly" non-rechargeable batteries; they have less of an impact on non-renewable sources, pollution, and global warming [19].

Minimizing the physical size of the system correlates to a more efficient use of resources. Less volume taken up by the internal components correlates to less material required to enclose the components. Less material results in reduced waste and damage to the environment when the product reaches the end of its life cycle.

Upgrades of the system can improve the power consumption of the device. Extending the battery life correlates to less impact on renewable or non-renewable resources. Further research can evaluate the best method for energy storage as well as how to design a product to perfom as energy efficient as possible.

8. Ethical

To ensure this project remains ethical through its duration, analysis of two ethical frameworks explore potential ethical issues.

The IEEE Code of Ethics aims to foster awareness on ethical issues while promoting ethical behavior within IEEE fields of interest [21]. It can act as a guideline in the design and development process of this project, ensuring the project doesn't cross any ethical boundaries. As this project intends to use old and new technology, a fundamental understanding of technologies proves necessary to explore potential consequences that could arise. For example, as this system intends to physically and electrically interface with other devices, the system must ensure neither device harms the other. This entails proper understanding of interfacing various mechinal and electrical components. Also, this system uses components, ideas, and technologies developed by other parties and these contributions to the system needs acknowledgement. The IEEE Code of Ethics directly states to reject bribery in all its forms. Bribery comes in various forms including gifts, monetary payments, or privileges used in exchange for favorable treatment. Bribes could potentially give this product unfair advantages over other products in the market. Throughout the project's life cycle, the identification and avoidance of bribery ensures the project does not sustain unfair and unethical advantages.

Another ethical framework, namely ethical principlism, applies to this project. Ethical principlism uses four basic moral principles: autonomy, beneficence, nonmaleficence, and justice [22]. These four principles together applied to this project judge its ethicality. Autonomy refers to the basic human right of freedom of choice. If this system goes to market, advertising must provide honest and legitmate details on the system's functionality so potential buyers make sincere choices whether to purchase the system or not. Beneficience refers to contributing to the welfare of others. This project intends to provide another means for people to listen to audio wirelessly. It aims to enhance a listening experience by allowing the user to detach from the audio source. Nonmaleficence entails refraining from harmful actions, distinguishable from contributing helpful actions. This principle applies to the manufacturing of the system. Ensuring the final product can't harm the user in any way translates to a lot of effort. This involves the types of materials used, the encompassment of electrical components, and the levels of electromagnetic radiation. The last principle, justice, refers to administering fairness in processes and actions. To ultimately make a profit, the system must properly enter the marketing process. As this system would have competition, it remains important not to oversell or exaggerate its functionalities in order to gain market share. Also, the product should sell at a reasonable price given the cost of parts and manufacturing.

9. Health and Safety

As this system intends to use radio-frequency electric fields for wireless transfer of data, the system can not expose unhealthy levels of electromagnetic radiation to ensure the safety of humans. The system follows standards which ensure safe levels of exposure. Also, the manufacturing quality of the system must maintain itself to ensure the product safe for human use and physical contact. The system uses electrical components to operate; the system should properly enclose these components to avoid the transfer of energy between the device and humans.

10. Social and Political

This system intends to benefit society by providing another means to listen to audio wirelessly. A major reason why the wireless audio industry expands has to do with its acceptance by society. Devices with wired connections (i.e. telephones, computer peripherals, etc.) continue to die out as both companies and consumers switch to wireless alternatives. Being liberated from wired connections still remains as a relatively new concept that many industries continue to adopt and develop. The familiarity and acceptance of Bluetooth technology (and its applications) by society also continues to increase. Bluetooth yields many applications including audio streaming, and it remains a entity that society knows and trusts.

Direct stakeholders for the development of this project include myself and my advisor. We both have concern with the progress of the project's activities throughout its development. Indirect customers include potential customers, suppliers, and competitors. Potential customers may benefit from the functionality of this project if it goes to market. This project composes of parts needed from various suppliers thus positively affecting their business. This project may affect competitors in the market depending on how much market share it acquires.

11. Development

This project encompasses various areas in the field of engineering. The biggest aspect in the development of this system involves the integration of hardware and software to meet the system's specifications and requirements. Researching and evaluating different components and methods to enable the system's intended functionalities was a vital first step, and was also required throughout the duration of the project. An iterative approach was used in the development of the system, creating overlap between design phases. Changes to hardware affected the way the software program was written, and limitations and findings in software capabilities affected the hardware decisions. Prototyping and early testing forced reconsiderations to the implementation and integration methods of the system.

Appendix B: Code

```
#include <SparkFunbc127ver6.h>
#include <SoftwareSerial.h>

#define SINK 0
#define SOURCE 1

// digital inputs w/ interupts (all active low)
const char BTN_VUP = 0;
const char INT_VUP = 2;
const char BTN_VDN = 1;
const char INT_VDN = 3;
const char BTN_MUTE = 2;
const char INT_MUTE = 1;
const char DETECT_OUT = 3;
const char INT_OUT = 0;
const char DETECT_IN = 7;
const char INT_IN = 4;

// outputs
const char MUTE = 8; // digital, active low
const char LED_SINK = 9; // analog, active high
const char LED_SOURCE = 10; // analog, active high

// variables to hold different states
volatile char VUPstate = 0;
volatile char VDNstate = 0;
volatile char MUTEstate = 0;
volatile char DETECT_INstate = 0;
volatile char DETECT_OUTstate = 0;
volatile char CONNstate = 0; // variable tracking connection status
volatile char TIMERstate = 0;
volatile char ROLE = SINK; // variable tracking sink (0) or source (1)

// other variables
int button_hold = 0;
char determined_mode = SINK;
String buffer;

// Create a software serial port.
SoftwareSerial swPort(15, 18); // RX, TX

// create a BC127 and attach the software serial port to it.
BC127 BTModu(&swPort);

void setup()
{
  Serial.begin(9600);

  // Serial port configuration. The software port should be at 9600 baud, as that
  //  is the default speed for the BC127.
  swPort.begin(9600);

  // input setup
  pinMode(BTN_VUP, INPUT);
  attachInterrupt(INT_VUP, ISR_VUP, FALLING);
  pinMode(BTN_VDN, INPUT);
  attachInterrupt(INT_VDN, ISR_VDN, FALLING);
  pinMode(BTN_MUTE, INPUT);
  attachInterrupt(INT_MUTE, ISR_MUTE, FALLING);
  pinMode(DETECT_IN, INPUT);
  attachInterrupt(INT_IN, ISR_IN, CHANGE);
  pinMode(DETECT_OUT, INPUT);
  attachInterrupt(INT_OUT, ISR_OUT, CHANGE);

  // output setup
  pinMode(MUTE, OUTPUT);
  pinMode(LED_SINK, OUTPUT);
  pinMode(LED_SOURCE, OUTPUT);

  // unmute
  digitalWrite(MUTE, 1);

  delay(10);

  Serial.println(BTModu.battConfig());
  Serial.println(BTModu.autoConnect());
  Serial.println(BTModu.battStatus());

  // determine module mode and set corresponding LEDs
  determined_mode = BTModu.determineMode();
  if (determined_mode == SINK) {
    Serial.println("Determined mode: sink");
    ROLE = SINK;
    digitalWrite(LED_SOURCE, 0);
    digitalWrite(LED_SINK, 1);
  } else if (determined_mode == SOURCE) {
    Serial.println("Determined mode: source");
    ROLE = SOURCE;
    digitalWrite(LED_SOURCE , 1);
    digitalWrite(LED_SINK, 0);
  } else {
    sinkSetup();
  }

  //set timer1 interrupt at 1Hz
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 6000;// = (8*10^6) / (freq*1024) - 1 (must be <65536)
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

  sei(); // enable interrupts

}

void loop()
{

  // catch all UART traffic when no hardware interrupts are issued
  if (swPort.available()>0){
    char temp = "";
    while (swPort.available()>0){
      temp = swPort.read();
      buffer.concat(temp);
      delay(1);
    }
  }

  // print buffer if not empty
  if (buffer != ""){
    Serial.print("buffer:");
    Serial.println(buffer);
    //Serial.println(buffer.length());
  }

  // if UART message is ABS_VOL 11 0, switch mode to sink
  if (buffer == "ABS_VOL 11 0\r"){
    Serial.println("Command from other BC127 to switch to sink");
    switchToSink();
  }

  // if UART message is ABS_VOL 11 1, switch mode to source
  if (buffer == "ABS_VOL 11 8\r"){
    Serial.println("Command from other BC127 to switch to source");
    switchToSource();
  }

  // if UART message starts with AVRCP_STOP... set TIMERSTATE == 1 right away (instead of waiting for
interrupt)
  if (buffer.startsWith("AVRCP_STOP")){
    Serial.println("Link loss");
    TIMERstate = 1;

    // reset timer
    TCCR1A = 0;// set entire TCCR1A register to 0
    TCCR1B = 0;// same for TCCR1B
    TCNT1  = 0;//initialize counter value to 0
    OCR1A = 4000;
      // turn on CTC mode
    TCCR1B |= (1 << WGM12);
    // Set CS10 and CS12 bits for 1024 prescaler
    TCCR1B |= (1 << CS12) | (1 << CS10);
    // enable timer compare interrupt
    TIMSK1 |= (1 << OCIE1A);
  }

  buffer = ""; // clear buffer

  // every timer
  if (TIMERstate == 1)
  {
    Serial.println("timer looped");
    //Serial.println(BTModu.battStatus());
    // if connected, interrupt every 5 seconds
    if (BTModu.connectionState() == 1){
      Serial.println("Connected!");
      CONNstate = 1;
      OCR1A = 40000;


      // ensure LED is on
      if (ROLE == SINK) digitalWrite(LED_SINK, 1);
      if (ROLE == SOURCE) {
        digitalWrite(LED_SOURCE, 1);
        Serial.println(BTModu.musicCommands(BC127::PLAY));
      }

    // if not connected, interrupt every ~0.5s
    } else {
      Serial.println("Not connected!");
      CONNstate = 0;
      OCR1A = 4000;

      // blink LED and attempt to open connection if configured as source
      if (ROLE == SINK){
        Serial.println("timer sink");
        if (digitalRead(LED_SINK) == HIGH) digitalWrite(LED_SINK, 0);
        else digitalWrite(LED_SINK, 1);
      }
      if (ROLE == SOURCE){
        Serial.println("timer source");

        Serial.println(BTModu.connect("20FABB020188", BC127::A2DP));
        //Serial.println(BTModu.connect("20FABB020189", BC127::A2DP));
        Serial.println(BTModu.musicCommands(BC127::PLAY));


        if (digitalRead(LED_SOURCE) == HIGH) {
          digitalWrite(LED_SOURCE, 0);
        } else {
          digitalWrite(LED_SOURCE, 1);
        }

      }
    }

    TIMERstate = 0;
  }

  // if VUP switch is pressed, increase volume
  if (VUPstate == 1) {
    delay(200);
    Serial.println("VUP pressed");
    Serial.println(BTModu.getVolume());
    digitalWrite(MUTE, 1);
    Serial.println(BTModu.musicCommands(BC127::UP));
    VUPstate = 0;
  }

  // if VDN switch is pressed, decrease volume
  if (VDNstate == 1) {
    delay(200);
    Serial.println("VDN pressed");
    Serial.println(BTModu.getVolume());
    if (BTModu.getVolume() == 2) digitalWrite(MUTE, 0);
    else {
      Serial.println(BTModu.musicCommands(BC127::DOWN));
      digitalWrite(MUTE, 1);
    }
    VDNstate = 0;
  }
```

```
// if MUTE switch is pressed, mute audio driver
// if MUTE is held down for 1s, switch audio mode
if (MUTEstate == 1) {
  //delay(200);
  while (digitalRead(BTN_MUTE) == 0 && button_hold < 1000){
    delay(1);
    button_hold++;
  }
  if (button_hold == 1000) {
    Serial.println("Mode change");
    determined_mode = BTModu.determineMode();
    if (determined_mode == SINK) {
      digitalWrite(LED_SINK, 0);
      digitalWrite(LED_SOURCE, 1);
      Serial.println(BTModu.switchToSinkMessage());
      Serial.println(BTModu.switchToSinkMessage());
      Serial.println(BTModu.switchToSinkMessage());
      sourceSetup();
    }
    else if (determined_mode == SOURCE) {
      digitalWrite(LED_SINK, 1);
      digitalWrite(LED_SOURCE, 0);
      Serial.println(BTModu.switchToSourceMessage());
      Serial.println(BTModu.switchToSourceMessage());
      Serial.println(BTModu.switchToSourceMessage());
      sinkSetup();
    }
    else {
      Serial.println(BTModu.switchToSourceMessage());
      Serial.println(BTModu.switchToSourceMessage());
      Serial.println(BTModu.switchToSourceMessage());
      sinkSetup();
    }
    //delay(500);
  }
  else if (button_hold >= 2) {
    if (digitalRead(MUTE) == 1) {
      Serial.println("MUTE");
      digitalWrite(MUTE, 0);
    } else {
      Serial.println("UNMUTE");
      digitalWrite(MUTE, 1);
    }
  }

  button_hold = 0;
  MUTEstate = 0;
}

if (DETECT_OUTstate == 1) {
  delay(500);
  if (digitalRead(DETECT_OUT) == 1) {
    Serial.println("DETECT_OUT removed");
    digitalWrite(MUTE, 0);
  } else {
    Serial.println("DETECT_OUT inserted");
    digitalWrite(MUTE, 1);
  }
  DETECT_OUTstate = 0;
  }

  if (DETECT_INstate == 1) {
    delay(500);
    if (digitalRead(DETECT_IN) == 1) {
      Serial.println("DETECT_IN removed");
    } else {
      Serial.println("DETECT_IN inserted");
    }
    DETECT_INstate = 0;
  }

}

void getBTAddress()
{
  char index = 0;
  //String address;
  Serial.println("Getting address:");
  Serial.println(BTModu.getAddress(index, address));
  Serial.println(address);
}

void system_restore()
{
  Serial.println(BTModu.restore());
}

void switchToSink(){
  digitalWrite(MUTE, 0);
  if (ROLE == SOURCE) sinkSetup();
  digitalWrite(MUTE, 1);
}

void switchToSource(){
  digitalWrite(MUTE, 0);
  if (ROLE == SINK) sourceSetup();
  digitalWrite(MUTE, 1);
}

// setup for sink
void sinkSetup()
{
  Serial.println("Setting up sink...");
  Serial.println(BTModu.setClassicSink());
  Serial.println(BTModu.writeConfig());
  Serial.println(BTModu.reset());
  digitalWrite(LED_SOURCE, 0);
  digitalWrite(LED_SINK, 1);
  //Serial.println(BTModu.musicCommands(BC127::PLAY));
  ROLE = SINK;
  // reset timer
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  OCR1A = 4000;
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);
}

void sourceSetup()
{
  Serial.println("Setting up source...");
  Serial.println(BTModu.setClassicSource());
  Serial.println(BTModu.writeConfig());
  Serial.println(BTModu.reset());
  digitalWrite(LED_SOURCE, 1);
  digitalWrite(LED_SINK, 0);
  //Serial.println(BTModu.musicCommands(BC127::PLAY));
  ROLE = SOURCE;
  // reset timer
  TCCR1A = 0;// set entire TCCR1A register to 0
  TCCR1B = 0;// same for TCCR1B
  TCNT1  = 0;//initialize counter value to 0
  OCR1A = 4000;
  // turn on CTC mode
  TCCR1B |= (1 << WGM12);
  // Set CS10 and CS12 bits for 1024 prescaler
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);
}

//ISRs
void ISR_VUP() {
  VUPstate = 1;
}

void ISR_VDN() {
  VDNstate = 1;
}

void ISR_MUTE() {
  MUTEstate = 1;
}

void ISR_IN() {
  DETECT_INstate = 1;
}

void ISR_OUT() {
  DETECT_OUTstate = 1;
}

ISR(TIMER1_COMPA_vect) {
  TIMERstate = 1;
}
```

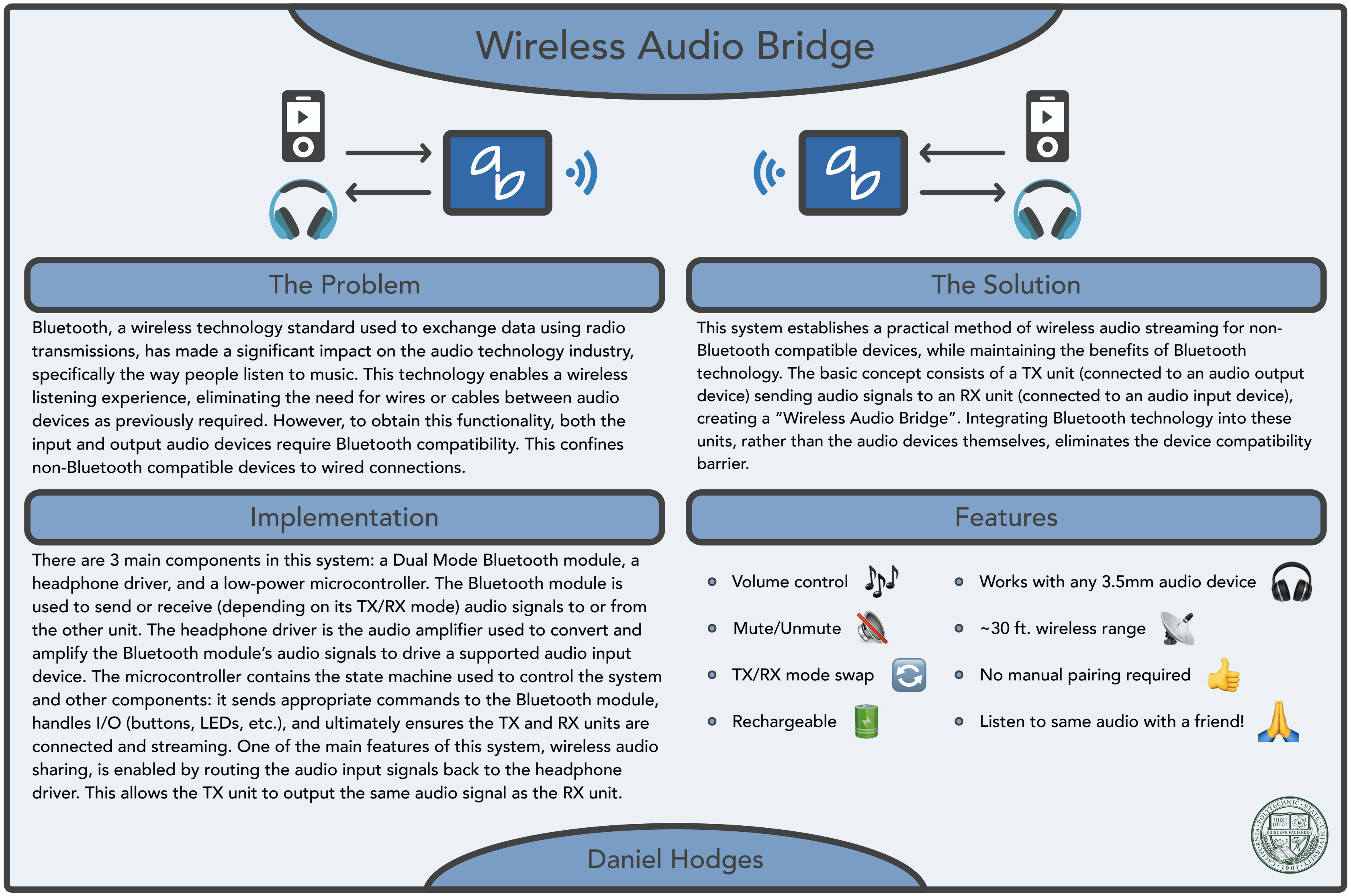
```
#ifndef BC127_h
#define BC127_h

#include <Arduino.h>
#include <SoftwareSerial.h>

class BC127
{
  public:
    // Let's make an enum for different types of connections. The
    // BC127 module can support a lot of different types of connections
    // but we'll only actually use a few of them.
    enum connType {SPP, BLE, A2DP, HFP, AVRCP, PBAP, ANY};

    // Now, make a data type for function results.
    enum opResult {REMOTE_ERROR = -5, CONNECT_ERROR, INVALID_PARAM,
                 TIMEOUT_ERROR, MODULE_ERROR, DEFAULT_ERR, SUCCESS};

    // enum for the various audio commands we can use on the module.
    enum audioCmds {PLAY, PAUSE, FORWARD, BACK, UP, DOWN, STOP};

    enum modeSetting {MODE_SINK, MODE_SOURCE, MODE_NEITHER};


    BC127(Stream* sp);
    opResult reset();
    opResult restore();
    opResult switchToSinkMessage();
    opResult switchToSourceMessage();
    opResult writeConfig();
    opResult autoConnect();
    opResult getConfig();
    opResult battConfig();
    opResult battStatus();
    opResult connect(String address, connType connection);
    modeSetting determineMode();
    opResult musicCommands(audioCmds command);
    opResult setClassicSink();
    opResult setClassicSource();
    opResult stdSetParam(String command, String param);
    opResult stdCmd(String command);
    opResult connectionState();
    int getVolume();
  private:
    BC127();
    int _baudRate;
    String _addresses[1];
    char _numAddresses;
    Stream *_serialPort;
    opResult knownStart();
};

// There are several commands that look for either OK or ERROR; let's abstract
//  support for those commands to one single private function, to save memory.
BC127::opResult BC127::stdCmd(String command)
{
  Serial.println();
  Serial.println(command);
  String buffer;
  String EOL = String("\r");

  knownStart(); // Clear the serial buffer in the module and the Arduino.

  _serialPort->print(command);
  _serialPort->print("\r");
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the command. Bog-standard Arduino stuff.
  unsigned long startTime = millis();

  // This is our timeout loop. We'll give the module 3 seconds.
  while ((startTime + 3000) > millis())
  {
    // Grow the current buffered data, until we receive the EOL string.
    if (_serialPort->available() > 0) buffer.concat(char(_serialPort->read()));

    if (buffer.endsWith(EOL))
    {
      if (buffer.startsWith("ER")) return MODULE_ERROR;
      if (buffer.startsWith("OK")) return SUCCESS;
      buffer = "";
    }
  }
  return TIMEOUT_ERROR;
}

// Similar to the command function, let's do a set parameter genrealization.
BC127::opResult BC127::stdSetParam(String command, String param)
{
  Serial.println();
  Serial.print(command);
  Serial.print(" ");
  Serial.println(param);
  String buffer;
  String EOL = String("\r");

  knownStart();  // Clear Arduino and module serial buffers.

  _serialPort->print("SET ");
  _serialPort->print(command);
  _serialPort->print(" ");
  _serialPort->print(param);
  _serialPort->print("\r");
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the reset. Bog-standard Arduino stuff.
  unsigned long startTime = millis();

  // This is our timeout loop. We'll give the module 2 seconds to reset.
  while ((startTime + 2000) > millis())
  {
    // Grow the current buffered data, until we receive the EOL string.
    if (_serialPort->available() >0) buffer.concat(char(_serialPort->read()));

    if (buffer.endsWith(EOL))
    {
      if (buffer.startsWith("ER")) return MODULE_ERROR;
      else if (buffer.startsWith("OK")) return SUCCESS;
      buffer = "";
    }
  }
  return TIMEOUT_ERROR;
}

// One of the neat features of the BC127 is the ability to control an audio
//  player remotely. This function will activate those features, programmatically.
```

```
BC127::opResult BC127::musicCommands(audioCmds command)
{
  switch(command)
  {
    case PAUSE:
      return stdCmd("MUSIC PAUSE");
    case PLAY:
      return stdCmdShort("MUSIC 10 PLAY");
    case FORWARD:
      return stdCmd("MUSIC FORWARD");
    case BACK:
      return stdCmd("MUSIC BACKWARD");
    case STOP:
      return stdCmd("MUSIC STOP");
    case UP:
      return stdCmd("VOLUME 10 UP");
    case DOWN:
      return stdCmd("VOLUME 10 DOWN");
    default:
      return INVALID_PARAM;
  }
}

BC127::opResult BC127::switchToSinkMessage()
{
  return stdCmd("VOLUME 10 0");
}

BC127::opResult BC127::switchToSourceMessage()
{
  return stdCmd("VOLUME 10 1");
}

BC127::opResult BC127::getConfig()
{
  return stdCmd("CONFIG");
 }

BC127::opResult BC127::battConfig()
{
  return stdSetParam("BATT_CONFIG", "ON 145 4220 1500 150");
}

BC127::opResult BC127::battStatus()
{
  return stdCmd("BATTERY_STATUS");
}

BC127::opResult BC127::autoConnect()
{
  return stdSetParam("AUTOCONN", "0");
}

// In order to set the module as a source for streaming audio out to another
//  device, you must set the "CLASSIC_ROLE" parameter to 1, then write/reset to
//  make that setting active. This function handles this parameter setting.
BC127::opResult BC127::setClassicSource()
{
  //return stdSetParam("CLASSIC_ROLE", "1");
  return stdSetParam("PROFILES", "0 0 0 1 1 0 0 0 0 0 0 0");
```

```
}

// Of course, we also need some way to return the module to sink mode, if we
//  want to do that.
BC127::opResult BC127::setClassicSink()
{
  //return stdSetParam("CLASSIC_ROLE", "0");
  return stdSetParam("PROFILES", "0 0 1 0 1 0 0 0 0 0 0 0");
}

int BC127::getVolume()
{
  int retVal = -1;
  String buffer = "";
  String addressTemp;
  String EOL = String("\r");

  knownStart(); // Purge serial buffers on Arduino and module.

  // Now issue the inquiry command.
  _serialPort->print("VOLUME 10");
  _serialPort->print("\r");
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the inquiry. Bog-standard Arduino stuff.
  unsigned long loopStart = millis();

  // Calculate a reset timeout value that's a tish longer than the module will
  //  use. This is our catch-all, so we don't sit in this loop forever waiting
  //  for input that will never come from the module.
  unsigned long loopTimeout = 100;

  // Oooookaaaayyy...now the fun part. A state machine that parses the input
  //  from the Bluetooth module!
  while (loopStart + loopTimeout > millis())
  {

    // Grow the current buffered data, until we receive the EOL string.
    while (_serialPort->available() > 0)
    {
      char temp = _serialPort->read();
      buffer.concat(temp);
      if (temp = '\r') break;
    }

    // Parse the current line of text from the module and see what we find out.
    if (buffer.endsWith(EOL))
    {
      // If the current line starts with "STATE", we need more parsing. This is
      //  also the only guaranteed result.
      if (buffer.startsWith("10 A2DP"))
      {
        // If "CONNECTED" is in the received string, we know we're connected,
        //  but not if we're connected with the particular profile we're
        //  interested in. So, we need to consider a bit further.
        if (buffer.substring(8,9) == "A") {
          retVal = 0xA;
        } else if (buffer.substring(8,9) == "B") {
          retVal = 0xB;
        } else if (buffer.substring(8,9) == "C") {
          retVal = 0xC;
        } else if (buffer.substring(8,9) == "D") {
          retVal = 0xD;
        } else if (buffer.substring(8,9) == "E") {
          retVal = 0xE;
        }else if (buffer.substring(8,9) == "F") {
          retVal = 0xF;
        } else {
          retVal = buffer.substring(8,9).toInt();
        }
      }
      if (buffer.startsWith("OK")) return retVal;
      buffer = "";
    }
  }
  return retVal;
}

// inquiry2
BC127::modeSetting BC127::determineMode()
{

  modeSetting retVal = MODE_NEITHER;
  String buffer = "";
  String addressTemp;
  String EOL = String("\r");

  knownStart(); // Purge serial buffers on Arduino and module.

  // Now issue the inquiry command.
  _serialPort->print("GET PROFILES");
  _serialPort->print("\r");
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the inquiry. Bog-standard Arduino stuff.
  unsigned long loopStart = millis();

  // Calculate a reset timeout value that's a tish longer than the module will
  //  use. This is our catch-all, so we don't sit in this loop forever waiting
  //  for input that will never come from the module.
  unsigned long loopTimeout = 200;

  // Oooookaaaayyy...now the fun part. A state machine that parses the input
  //  from the Bluetooth module!
  while (loopStart + loopTimeout > millis())
  {

    // Grow the current buffered data, until we receive the EOL string.
    while (_serialPort->available() > 0)
    {
      char temp = _serialPort->read();
      buffer.concat(temp);
      if (temp = '\r') break;
    }

    // Parse the current line of text from the module and see what we find out.
    if (buffer.endsWith(EOL))
    {
      // If the current line starts with "STATE", we need more parsing. This is
      //  also the only guaranteed result.
      if (buffer.startsWith("PROFILES"))
      {
        // If "CONNECTED" is in the received string, we know we're connected,
        //  but not if we're connected with the particular profile we're
        //  interested in. So, we need to consider a bit further.

        Serial.println(buffer.substring(9,32));
        if (buffer.substring(9, 32) == "0 0 0 1 1 0 0 0 0 0 0 0")
        {
          retVal = MODE_SOURCE;
        }
        if (buffer.substring(9, 32) == "0 0 1 0 1 0 0 0 0 0 0 0")
        {
          retVal = MODE_SINK;
        }
      }
      if (buffer.startsWith("OK")) return retVal;
      buffer = "";
    }
  }
  return retVal;
}

// connect by address
//  Attempts to connect to one of the Bluetooth devices which has an address
//  stored in the _addresses array.
BC127::opResult BC127::connect(String address, connType connection)
{
  // Before we go any further, we'll do a simple error check on the incoming
  //  address. We know that it should be 12 hex digits, all uppercase; to
  //  minimize execution time and code size, we'll only check that it's 12
  //  characters in length.
  if (address.length() != 12) return INVALID_PARAM;

  // We need a buffer to store incoming data.
  String buffer;
  String EOL = String("\r"); // This is just handy.

  // Convert our connType enum into the actual string we need to send to the
  //  BC127 module.
  switch(connection)
  {
    case SPP:
      buffer = " SPP";
      break;
    case BLE:
      buffer = " BLE";
      break;
    case A2DP:
      buffer = " A2DP";
      break;
    case AVRCP:
      buffer = " AVRCP";
      break;
    case HFP:
      buffer = " HFP";
      break;
    case PBAP:
      buffer = " PBAP";
      break;
    default:
      buffer = " SPP";
      break;
  }

  knownStart(); // Purge serial buffers on both the module and the Arduino.

  // Now issue the inquiry command.
  _serialPort->print("OPEN ");
  _serialPort->print(address);
  _serialPort->print(buffer);
  _serialPort->print("\r");
  // We need to wait until the command finishes before we start looking for a
  //  response; that's what flush() does.
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the connect command. Bog-standard Arduino stuff.
  unsigned long connectStart = millis();

  buffer = "";

  // The timeout on this is 5 seconds; that may be a bit long.
  while (connectStart + 10 > millis())
  {
    // Grow the current buffered data, until we receive the EOL string.
    if (_serialPort->available() >0) buffer.concat(char(_serialPort->read()));

    // At some point, the BC127 response string will contain the EOL string.
    //  Once that happens, we can figure out what the response looks like:
    //  "ERROR" - there's a syntax error in your message to the module; this is
    //    kind of unlikely, although it could happen if you call this function
    //    with an invalid address (something not entirely uppercase hex digits)
    //  "OPEN_ERROR" - most likely, the module can't find any devices with that
    //    address.
    //  "PAIR_ERROR" - the connection was refused by the remote module.
    //  "PAIR_OK" - the connection has been made, but the SPP channel is not
    //    yet open. We should probably just ignore this.
    //  "OPEN_OK" - ready to rock! This is when we should return success.
    if (buffer.endsWith(EOL))
    {
      if (buffer.startsWith("ERROR")) return MODULE_ERROR;
      if (buffer.startsWith("OPEN_ERROR")) return CONNECT_ERROR;
      if (buffer.startsWith("PAIR_ERROR")) return REMOTE_ERROR;
      if (buffer.startsWith("OPEN_OK")) return SUCCESS;
      buffer = "";
    }
  }
  return TIMEOUT_ERROR;
}

// There are times when it is useful to be able to know whether or not the
//  module is connected; this function will tell you whether or not the module
//  is connected with a certain protocol, or if it is connected at all. This is
//  particularly challenging; the strings coming from the module are so fast,
//  even at 9600 baud, that you don't really have time to parse them. The only
//  solution I've been able to come up with is to just let the buffer overflow
//  and give up on identifying connections by type.
```

```
BC127::opResult BC127::connectionState()
{
  String buffer;
  String EOL = String("\r");
  opResult retVal = TIMEOUT_ERROR;

  knownStart();

  _serialPort->print("STATUS");
  _serialPort->print("\r");
  _serialPort->flush();

  // We're going to use the internal timer to track the elapsed time since we
  //  issued the command. Bog-standard Arduino stuff.
  unsigned long startTime = millis();

  // This is our timeout loop. We'll give the module 500 milliseconds.
  //  Note: if you have more than one active connection this WILL overflow a
  //  software serial buffer. Working under this assumption, we're going to
  //  try and deal with both the overflow and no overflow case gracefully. I'm
  //  also removing the ability to check on a specific connection type, since
  //  that's what causes the overflow.
  while ((startTime + 200) > millis())
  {
    // Grow the current buffered data, until we receive the EOL string.
    while (_serialPort->available() > 0)
    {
      char temp = _serialPort->read();
      buffer.concat(temp);
      if (temp = '\r') break;
    }

    // Parse the current line of text from the module and see what we find out.
    if (buffer.endsWith(EOL))
    {
      // If the current line starts with "STATE", we need more parsing. This is
      //  also the only guaranteed result.
      if (buffer.startsWith("LINK 10"))
      {
        // If "CONNECTED" is in the received string, we know we're connected,
        //  but not if we're connected with the particular profile we're
        //  interested in. So, we need to consider a bit further.
        Serial.println(buffer.substring(36, 42));
        if (buffer.substring(36, 42) == "STREAM") retVal = SUCCESS;
        // If "CONNECTED" *isn't* there, we want to return an appropriate error.
        else retVal = CONNECT_ERROR;
      }
      // If by some miracle we *do* get to this point without a buffer overflow,
      //  we're safe to return without a buffer purge.
      if (buffer.startsWith("OK")) return retVal;
      buffer = "";
    }
  }
  // Okay, now we need to clean up our input buffer on the serial port. After
  //  all, we can be pretty sure that an overflow happened, and there's crap in
  //  the buffer.
  while (_serialPort->available() > 0) _serialPort->read();
  return retVal;
}
```

Appendix C: Project Poster



Wireless Audio Bridge

The Problem

Bluetooth, a wireless technology standard used to exchange data using radio transmissions, has made a significant impact on the audio technology industry, specifically the way people listen to music. This technology enables a wireless listening experience, eliminating the need for wires or cables between audio devices as previously required. However, to obtain this functionality, both the input and output audio devices require Bluetooth compatibility. This confines non-Bluetooth compatible devices to wired connections.

The Solution

This system establishes a practical method of wireless audio streaming for non-Bluetooth compatible devices, while maintaining the benefits of Bluetooth technology. The basic concept consists of a TX unit (connected to an audio output device) sending audio signals to an RX unit (connected to an audio input device), creating a "Wireless Audio Bridge". Integrating Bluetooth technology into these units, rather than the audio devices themselves, eliminates the device compatibility barrier.

Implementation

There are 3 main components in this system: a Dual Mode Bluetooth module, a headphone driver, and a low-power microcontroller. The Bluetooth module is used to send or receive (depending on its TX/RX mode) audio signals to or from the other unit. The headphone driver is the audio amplifier used to convert and amplify the Bluetooth module's audio signals to drive a supported audio input device. The microcontroller contains the state machine used to control the system and other components: it sends appropriate commands to the Bluetooth module, handles I/O (buttons, LEDs, etc.), and ultimately ensures the TX and RX units are connected and streaming. One of the main features of this system, wireless audio sharing, is enabled by routing the audio input signals back to the headphone driver. This allows the TX unit to output the same audio signal as the RX unit.

Features

- Volume control 🎶
- Mute/Unmute 🔇
- TX/RX mode swap 🔄
- Rechargeable 🔋
- Works with any 3.5mm audio device 🎧
- ~30 ft. wireless range 📡
- No manual pairing required 👍
- Listen to same audio with a friend! 🙏

Daniel Hodges