1. **Executive Summary**

“Music is love” – a song by David Crosby (of Crosby, Stills, Nash – and sometimes Young, among other groups); for some people, there is little in this world that is as capable of evoking emotions as strongly as music is able to. For some in this group, there is nothing more amazing than music; and because of that, the group has decided that our project should be focused around music.

Helium will be a music playing system capable of identifying individuals in a room, through the use of an RFID scanner. The system will encompass an intelligent song selection algorithm which will learn the musical tastes of each of its registered users. Once a user has listened to enough music using the devices for the device to “know” the user, the device will be able to generate playlists perfectly suited for that user. Ideally, the device will be able to generate playlists for each user that will be far superior than the user could create themselves. The device will associate variables with a user that most people never think of, for example, what kind of music does a user like listening to when it’s raining? When it’s cold? When it’s bright and sunny?

To further enhance the user’s music listening experience: the device will be capable of interpreting various voice commands – to keep the device hands free; it will feature a graphic equalizer – for getting the sound just right; and it will listen for conversations, and reduce its volume – so it won’t interrupt anyone.

Helium will feature a built in speaker as well as ports for additional speakers. The group will attempt to keep the device both low power and high fidelity. To do this, the group will have to balance the device’s power consumption and the thermal noise that it will create (mostly from the equalizer).

To keep the device functional for all its users, both registered and unregistered, a mobile app will be developed for controlling the device. With the addition of this feature, the user will now have three different method for accessing the true star of this device – its content, the music! The user will be able to start the device with a few taps of their phone, a couple of words, or just by walking near it while carrying an RFID tag.

The group hopes to further reduce the amount of user input necessary by adding an ability for the device to detect the user who is speaking. Although, this might just be wishful thinking. There is a lot that the group would like to do with this project, but not nearly enough time to do it all; nevertheless, the group will strive to make listening to music with Helium as easy and acoustically pleasing as possible.

1. **Project Description**

**2.1 Motivation**

The motivation for the core of this project, a music playing device, was primarily for the group to environ itself with a project that is related to a personal interest, music. In addition, the group wanted to be able to put a project on their resumes that could lead to employment at companies in the music industry; Pandora is one such company of great interest. The desire to create something that might have practical use to the group was also part of the original motivation; however, since the group has accepted sponsorship, the group is unlikely to be permitted to keep the prototype created for the project.

The original idea for the project was to create a music playing device that would monitor a room and play music when it is inhabited but quiet. The motivation for that idea being a dislike of being in a quiet room. Audio and visual methods were both considered for achieving this goal; however, the group decided to go with an RFID scanner. RF is considered an extremely important topic for electrical engineers and so the group decided it would be the best choice for resume building.

The group’s primary software developer has work experience and academic interest in machine learning, as well as a love for Pandora, in particular its song selection algorithm. Accordingly, the group decided to include a complex song selection algorithm in the project. Multiple methods of wireless communication and a database development component were also added to the project to ensure both an appropriate amount of software design was included in the project and that the group would garner experience in popular job skills.

To ensure that the project consisted of an appropriate amount of hardware design, an equalization module was added to the project. The group chose an equalizer both to gain experience with developing filters and in the hope that it would prove useful in developing their knowledge of music.

The project's name was chosen as an homage to the 90s noise rock band, Helium.

**2.2 Objectives**

The following is a list of objectives for the project divided into two groups; the first are high priority objectives, these are objectives that are considered critical to the project and will receive the majority of the group’s attention until they are achieved. The second group are low priority objectives; these will generally be objectives that are deemed as *optional*. Also, in general, these two groups will each be sorted from highest to lowest priority.

**High Priority Objectives**

1. The group will create a device capable of decoding *.mp3* audio files and sending the decoded output, with various levels of gain (volume), to external speakers.
2. The device will be capable of storing *.mp3* files, playlists, and user preferences on itself.
3. The device will be capable of identifying a nearby person, carrying a passive RFID tag (refer to section 3.7).
4. The device will be able to select songs without input being required from the user.
5. The device will be able to connect to the internet and query an external database.
6. The device will be capable of performing graphic equalization (refer to section 3.6.1).
7. The device will be able to automatically generate playlists for a user based on their listening habits – *smart playlists*.
8. The device will be able to wirelessly communicate with nearby mobile devices.
9. A mobile app will allow nearby mobile devices to control the device: change/create/edit playlists, switch users, switch songs, and add new users to the device.
10. The device will be capable of listening for audio input around it.
11. The device will be capable of recognizing the human voice.
12. The device will be able to interpret voice commands from nearby users.
13. The device will feature a power saving mode (for when no one is in the room that it is located in).
14. The device will allow users to use their own speakers with the device (without having to solder the speakers to the device).
15. The device will be able to log a user into, and stream from, Pandora.

**Low Priority Objectives**

1. The device will be capable of recognizing when conversations are occurring nearby – and will lower the volume appropriately.
2. The group will design and implement an algorithm for “scrapping” from the web various metadata about artists and/or songs; this will be used for making song recommendations.
3. The device will be able to determine the time of day.
4. The device will be able to determine its location (via GPS and/or WPS).
5. The device will be capable of identifying a person carrying a passive RFID tag from anywhere in the same room.
6. The device will be able to determine the weather conditions for its current location.
7. The device will be attempt to consider time of day and weather conditions when forming playlists and making recommendations of new songs.
8. Users will be able to upload new songs to the device via a Bluetooth connection to it with their mobile device.
9. The device will attempt to identify users by voice.
10. The device will attempt to identify users by time of day.
11. The device will attempt to consistently level the volume of all songs.

**2.3 Requirements and Specifications**

Table 2.3 A, below, is a list of the specific values and/or list of features that will be used to measure the project’s successfulness in meeting its objectives. The *Minimum* values indicate the minimally acceptable performance for an objective to be considered met; objectives that are considered optional will have a value of “None” in this field. The *Desired* values indicate the preferred level of performance for meeting an objective.

|  |  |  |
| --- | --- | --- |
| **Objectives** | **Minimum** | **Desired** |
| Audio files playable | .mp3 | .mp3, .wav, .wma |
| RFID read distance | 10cm (the typical maximum range for NFC) | 10m + (the length of a typical living room) |
| Signal to Noise Ratio (SNR) | 90 dB (the typical SNR for a CD) | 96 dB the Theoretical maximum for 16 bit audio |
| Power Consumption (typical) | <= 20 W (typical power consumption for a laptop) | <= 10 W (4 x the maximum power consumption of the beagle bone black) |
| Power Consumption (power saving mode) | 5 W | 2.5 W |
| Voice Commands Accuracy | 75% | 85% |
| Graphic EQ Band Count | 5 bands | 10+ bands |
| Total On-Device Storage | 1 GB | 10 GB |
| On Device Song Storage | 50 songs (~500 MB) | 1000 songs (~1 GB) |
| External Song Storage | 1000 songs (~1 GB) | 1000+ songs |
| Primary Device Size | 20cm\*20cm \*3cm (LWH; 1200 cm^3) | 15cm\*15cm\*3cm (765 cm^3) |
| Primary Device Weight | 2.3 kg (~5 lbs) | 1 kg (2.2 lbs) |
| Secondary Device Size | 15cm\*15cm\*3cm (765 cm^3) | 13cm\*13cm\*3cm (507 cm^3) |
| Secondary Device Weight | 2 kg (4.4 lbs) | 1 kg (2.2 lbs) |
| Maximum User Count | 4 Users (@ 1000 songs) | 8 Users (@ 1000 songs) |
| Smart Playlist Algorithm Performance | O(n), where n is the number of songs in the user’s *Master Playlist* (see section 4.2.2) | O(1) – time to execute is independent of the number of songs. |
| Smart Playlist Accuracy:  Given N songs, user skips S songs; Accuracy = (N-S)/N | 50% | 80% |
| Song Selection Statistical Model relies on… | Naïve Bayes or Binomial Logistic Regression | Multinomial Logistic Regression |
| Number of Attributes per Song | 16 (with the possibility of unknown) | 32+ (with the possibility of unknown) |
| Unknown Song Attribute Rate | 90% + (web scrapping might fail) | 20% |
| Number of Attributes per Song per User | 16 (no unknowns) | 32+ (no unknowns) |
| Song Recommendation Algorithm Performance | None | O(1) |
| Person Voice Identification Accuracy | None | 75% |
| Location Services | None | 1. WPS 2. Standalone GPS 3. Combination of WPS and GPS |
| Location Services Error Distance | None | <= 1 mile |
| Location Services Time to Calculate | None | <= 15 minutes (real time response is not needed) |
| Volume Leveling | None | The variance in volume between songs will be between -2 dB to +2 dB |
| Audio Output Ports | 1 (+ 1 hardwired speaker) | 4 (+ 1 hardwired speaker) |

*Table 2.3 A: Minimum and Desired Specifications.*

1. **Research**

**3.1 Similar Projects**

The use of researching past project is to see where they did well and where they did not do so well. Then to use this information to learn what is possible, difficult, or easy.

The motivation mp3 player is a similar project because both of our main focuses are to play music. The main parts of their design were the power supply, master microcontroller, slave microcontroller, the display, touch controls, EKG module, SD memory and the MP3 decoder. It would be useful to mention parts of the project that is helpful and similar. They used an MSP430 series microcontroller which would be a good topic of debate when we decide to choose a how to control our device. They also had a display, which is a topic we are not sure we will have on our device. The power supply was a Lithium Polymer USB charger and battery. The power went from the main power to a regulator to the main MSP 430.A SD memory card was used to store music and other software.

Music in motion is also a similar project that is from spring 2014. The power system is useful to talk about if we are going to make our device portable. They use 9 volts on input that goes to a 5 volt regulator and a 3v3 regulator. From there the regulators supply the power to each part of the device. A regulated power supply is usually an embedded circuit; it converts AC into a constant DC. This project is using batteries so the input is DC to start so the regulators are used to supply a stable voltage to a circuit that must be operated within certain power supply limits.

Guitar amplifier is a relevant project because the amplification system. We are going to need to play music and possibly build the amplifier for the music. An equalizer will be need for ours because many frequencies are being used rather than a hand full that is produced by the guitar.

This is not a senior design project but the amazon echo is also relevant to our project. The marketing team for the echo is pushing that it is a part of the family. There is a family that says

“Echo play rock and roll.” Then the device plays music in the genre rock and roll. Then another person says to play a different kind of music and the same happens. The idea of having a voice commanded device to play music that we like is very similar and we will take notes.

Siri on the iPhone and the really any voice recognition device that can play music is what we are looking at.

**3.2 MCU Selection**

The device’s MCU will have to be capable of controlling its various subsystems, communicating with other devices (via Blue Tooth and Wi-Fi), and processing complex algorithms (see sections 4.2.2 and 4.2.3). The group has elected to separate the RFID system from the main device; to do so, a second MCU will be used for controlling the RFID unit, processing the information that it scans, and communicating – via Blue Tooth – with the main device.

**Primary Device**

The primary device’s MCU must be capable of the following:

* Controlling the gain/attenuation of a 5 band (minimum) equalizer;
* Controlling the audio output voltage (volume);
* Controlling the power delivered to its own subsystems (allowing for a power saving mode);
* Connecting to the internet, via Wi-Fi;
  + Reliable data transfer (TCP);
* Communicating with another device, via Blue Tooth;
  + Quick data transfer (UDP) with the secondary device;
  + Possibly, reliable data transfer with multiple devices;
* Efficiently storing and locating song files;
* Efficiently storing and manipulating complex data structures;
* Processing complex statistical algorithms.

**Secondary Device**

The secondary device’s MCU must be capable of the following:

* Controlling the RFID scanner;
* Processing the RFID scanner’s output;
  + Identifying the user associated with the detected RFID tag;
* Communicating with the primary device.

**Considered MCUs**

The following MCUs were considered for both the primary and secondary devices:

* BeagleBone Black
  + by Texas Instruments
  + $55
  + Processor: AM335x 1 GHz ARM Cortex A8
  + RAM: 512 MB DDR3
  + Storage: 4 GB on-board Flash Storage
  + USB Ports: 1 (USB1.0) + 1 miniUSB
  + Power Usage 2.3 W
  + OS: Yes, Linux
* Raspberry Pi model B+
  + by Raspberry Pi Foundation
  + $35
  + Processor: ARM1176JZF-S 700 MHz
  + RAM: 512 MB SDRAM
  + Storage: MicroSD
  + USB Ports: 4
  + Power Usage: 3 W
  + OS: Yes, Linux
* MSP 430x41x
  + by Texas Instruments
  + Cost Varies (<$5 for most models)
  + Processor: 16 bit RISC CPU (8 MHz)
  + RAM: 256 B
  + Storage: 8 kB ROM
  + Power Usage: 7.2 mW/MHz (max)
  + OS: No

**Selected MCUs**

Since the **primary device** will have to be capable of processing complex statistical algorithms, communicating with other devices over both Wi-Fi and Blue Tooth, querying a database, playing music, and storing large amounts of user and song data – as well as controlling multiple subsystems – a powerful MCU with an OS was selected. The group chose to use the **BeagleBone Black** as the MCU for the primary device. This was selected over the Raspberry Pi for 2 reasons:

1. It has a faster processor: 1 GHz vs 700 MHz; and
2. It is made by TI; the group would like to garner experience with TI parts as much as possible.

For the **secondary device**, an **MSP 430** will be used. The RFID tag processing and wireless communications algorithms used by this device will be relatively simple, compared to the primary device, and will not require large amounts of RAM or storage; because of this, a powerful device is not needed and would consume far more power than necessary for the tasks it will be responsible for.

**3.3 Wireless Communications Research**

Helium will be required to communicate with several servers: the song server, which will store excess songs and metadata that the primary device doesn’t have the space for; a WPS location server, for determining the device’s location using Wi-Fi; and a weather server, for determining the weather conditions at the device’s location – this will be used as a factor in the song selection algorithm (see section 4.2.3). The following hardware will be considered for implementing wireless communications with the internet:

**SparkFun WiFi Breakout - CC3000**

This breakout board uses TI’s CC3000 Wi-Fi module. This module will implement many of the more challenging software elements of communicating over internet. This would be a great choice for out project which is high on aspirations for software, but low on software developers.

**Features**

* Communicates with 802.11 b/g
* On chip security accelerators for common personal networks
  + WEP;
  + WPA;
  + WPA2.
* IPv4 TCP/IP stack
* Supports 4 simultaneous TCP/UDP sockets
* Built in network protocols:
  + ARP;
  + ICMP;
  + DHCP;
  + DNS client;
* Interfaces with 4 wire SPI, clock speed up to 16 MHz
* Cost: ~$35

Another option offered by TI is the WL1837MOD; this module offers both Wi-FI at the 2.4 GHz range and Bluetooth.

**Features**

* Communicates with 802.11 a/b/g/n
* When idle, less than 800 uA for power consumption
* 80 Mbps for TCP
* 100 Mbps for UDP

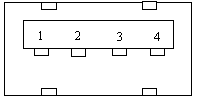
Since the group requires both a Wi-Fi and a Bluetooth module for our project, a device that implements both on a single chip would be particularly beneficial for the project. However, this device does seem to lack the on chip security accelerators that the C3000 contains. Since the networks that we will be connecting will almost definitely have security features, the group will use the C3000 for its wireless communications.

**3.4 Power Research**

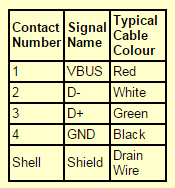
**3.4.1 Power via USB**

Power using a USB port is simple and easy. A USB runs on 5V and the maximum current that can be drawn is 500 milliamps. Therefore, the max load can be 2.5 Watts based off of Power(W)=Voltage multiplied by Amps. So this will only work if our device only needs this much voltage and power.­

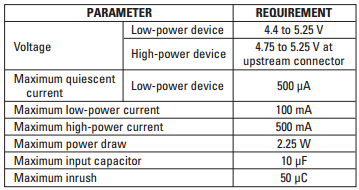
We would need to use the wires inside of the USB. After stripping the wire we would use the wires that supply power. These are usually the red and black wires while the other two are for data. Figure 3.4 A is the illustration of a typical USB port, table 3.4 B illustrates the name of the strip and the typical color of the wire, and Table 3.4 C shows the requirements for the whole USB port.



*Figure 3.4 A: USB port diagram; Courtesy of Texas Instruments.*



*Table 3.4 B: Typical wire set up; Courtesy of Texas Instruments.*



*Table 3.4 C: requirements for the USB port; Courtesy of Texas Instruments.*

There are two possible configurations for the USB products. One is single peripheral connected directly to a host or two a set of peripheral connected through hubs to the host. With single peripheral the current limiting does not cause an issue unless there is a large input capacitor placed across the power supply voltage. If a set of peripherals are used then current limiting will be required because of the unknown nature of many being plugged to the hub.

You can implement current limits in two different ways. Using a discrete power device with external control circuits would be the best option for cost for higher power devices. The other way is using switches that are integrated into the controllers. This option will be best for low power devices.

Powering low voltage electronics can be done a few ways. Low voltage meaning 3.3 V and this is limited to 0.65A. To provide this power you can use linear regulators, switching power supplies, and charge pumps. Linear regulators are low cost and the best option for lower voltages. They are a system used to maintain a steady voltage. This is very similar to a variable resistor that continuously adjusts to maintain a constant output voltage. This is possible due to the voltage divider low. A switching regulator is the same but the purpose is to keep an average value of output.

**3.4.2 Batteries**

An easy way for project to be portable would be for the device to be powered by batteries. There are many different kinds of batteries. There is AAA, AA, C, D, 9V and the list can go on but these are the most common portable batteries. AAA batteries are commonly used for devices like remotes, music players and cameras. The voltage of an AAA is 1.5V for an alkaline design. The benefit of using AAA is that it is small.

AA batteries are the most sold batteries. There are many different kinds of AA batteries due to the type of material it is made out of. There is Zinc-Carbon, Alkaline, Li-FeS2, NiCd, NiMH and NiZn. The most common is alkaline, which has a nominal voltage of 1.5V and the capacity under 50 milliamp constant drain is 1800 to 2600 milliamp hours.

C batteries are commonly used for medium drain devices like toys and musical instruments. Alkaline C batteries can hold up to an 8000 milliamp hours. D batteries are commonly used for high current drain applications. The nominal voltage is 1.5V and the capacity is 12000 to 18000 mAh. 9 Volt batteries are commonly named the transistor battery for being used for transistor radios. They are made up of six 1.5V cells enclosed by a wrapper. The typical capacity is 565 mAh for an alkaline 9V.

Another important part would be to mention the different kinds of materials batteries are made of and their uses.

**Carbon Zinc**, which also known as the “Heavy Duty” battery. These are the cheapest batteries on the market. These batteries have an open circuit voltage of 1.5 V and have high volume of productivity.

The advantage:

* lowest cost

The disadvantages:

* Short shelf life;
* Deteriorates rapidly in cold weather;
* Oxidation on the zinc wall eventually causes the contents to leak out.

**Alkaline** batteries are another type of primary battery. They have a 1.5V primary cell, and are the most popular general purpose battery. Similar to the carbon zinc, or Leclanche cell, the alkaline battery uses zinc and manganese. However, the electrolyte is Potassium hydroxide. Alkaline batteries have double the energy density of a Leclanche cell, last about 4 to 9 times longer, and are available mostly anywhere. The battery looks like a Leclanche cell, but the electrolyte is different. Noticeably, there is an anode current collector in the shape of a rod in the middle of the anode mixture, and a cathode as the outer shell, surrounded by an outer sleeve. As can be imagined, the similar makeup of the two batteries means they operate in much the same manner.

The advantages

* Cheap;
* Moderate drain rage.

The disadvantages are

* Bad performance at low temperatures;
* High impedance.

**Lithium Iron Disulfide’s** batteries differ from alkaline batteries in a few aspects. They are built in a spiral construction featuring two long, thin electrodes tolled together to form a jellyroll shape. This shape provides almost twenty time more interfacial surface area than a standard alkaline. This large interface helps to meet the power demands of many today’s devices. Lithium is the lightest, most active metal. When this metal is paired with iron disulfide, the energy available is 1.5V. Electrodes in lithium iron disulfide batteries are isolated from one another by a highly engineered microporous polymer membrane. This membrane allows ions to move easily during normal use, but restricts this movement in certain abuse situations to ensure product performance and safety.

The advantages:

* Direct drop in compatibility in applications using primary 1.5 V AA and AAA battery sizes
* Greater power than other primary batteries
* Superior leakage resistance compared to other primary conditions
* No added mercury, cadmium or lead

**Nickel Cadmium**. The nickel cadmium battery was once the popular choice for portable devices, but due to their cells only holding a mere 1.2V, other batteries have become more popular. However, these batteries have high energy and have a “high rate discharge capacity” making them useful for portable devices. The cathode in this battery is the nickel hydroxide, and the cadmium is the anode. The cells are sealed and utilize a recombinant system to prevent electrolyte loss and extend the useful life.

The advantages:

* cheap,
* long battery life
* High power
* Low internal resistance

The disadvantages:

* memory effect
* Containing toxic cadmium.

**Nickel Metal Hydride** is more used in devices that require large amount of energy and are used frequently. Some examples are digital campers, GPS units, and MP3 players. Cell voltage is 1.2 Volts.

The advantages:

* capacity;
* less memory effect;
* Less toxic.

The disadvantage:

* Complex charging.

**The Sealed Lead Acid** battery has a 1.5 Voltage, an energy density of 60Wh/kg or 100 Wh/L. The advantages:

* Easy to maintain
* Low self-discharge rate
* Cheap

The disadvantages:

* Large and Heavy

Polymer lithium ion batteries are very useful. There are few options for this kind of battery and the tables below, Table 3.4 D-E, compares each of them.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Size | Weight | Rating Voltage | Capacity |
| 400mAh | 5 x 25 x 35mm | 9g | 3.7V | 400 mAh |
| 850mAh | 5.7x29.5x48.27mm | 18.5g | 3.7V | 860 mAh |
| 1000mAh | 50.8 x 33.5 x 5.9 mm | 22g | 3.7V | 1000 mAh |
| 2000mAh | 5.8x54x60mm | 36g | 3.7V | 2000 mAh |

*Table 3.4 D: Comparison of lithium batteries.*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Impedance | Operating Consumption  Current | Cost |
| 400mAh | 150 milliohms | 10 uA (max) | $6.95 |
| 850mAh | Less than 300 milliohms | 0.2C5A | $7.95 |
| 1000mAh | Less than 300 milliohms | 0.2C5A | $8.95 |
| 2000mAh | Less than 300 milliohms | 0.2C5A | $12.95 |

*Table 3.4 E: comparison of lithium batteries.*

The datasheet indicates that a specific charger is needed to charge the battery. The following parameters are for the SparkFun USB LiPoly Charger - Single Cell. The basic charging circuit that allows you to charge 3.7V LiPo cells at a rate of 500mA or 100mA. It is designed to charge single-cell Li-Ion or Li-Polymer batteries. The cost is $14.95.

**Features**

• Linear Charge Management Controller:

- Integrated Pass Transistor

- Integrated Current Sense

- Reverse Discharge Protection

• High Accuracy Preset Voltage Regulation: + 0.75%

• Four Voltage Regulation Options:

* 4.20V, 4.35V, 4.40V, 4.50V

• Programmable Charge Current: 15 mA to 500 mA

Another Battery is called Polymer Li-Ion Pack with Fuel Gauge: 5V Regulated (27.01wh, 1A rate ) for 5V Mobile Electronic (2.19). The output voltage is useful with this battery. Table 3.4 F shows the specifications of the battery.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Size | Weight | Rating Voltage | Cost |
| Polymer Li-Ion Pack with Fuel Gauge | 5.8"(148mm) x 2.3"(58mm) x 0.8"(20mm) | 190 grams | 5V | $54.95 |

*Figure 3.4 F: Li-ion pack specifications.*

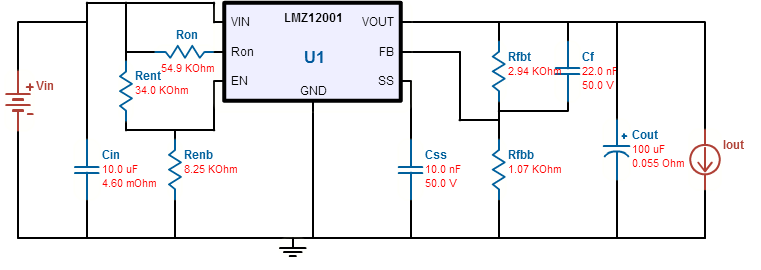
Another way to have power is to use batteries. Battery Holders will be needed if this option is chosen. A Coin Cell Battery Holder would be useful. (2xCR2032) This is a simple coin cell battery holder that can enclose two CR2032 batteries inside itself, and safely kept closed via two phillips head screws. Each battery holder will run batteries in series, output up to 6V, and is equipped with an On/Off slide switch and two 6" power wires (one positive and one negative) on the back.

**3.4.3 Regulators**

A voltage regulator is designed to automatically maintain constant voltage level. These are used in computer power supplies, alternators, and power plants. The regulation is specified by two measurements. One, load regulation is the change in output voltage for a given change in load current. Two, input regulation is the degree to which output voltage changes with input (supply) voltage changes. Other parameters that are important are temperature coefficient, initial accuracy, dropout voltage, absolute maximum ratings, output noise, quiescent current, and transient response.

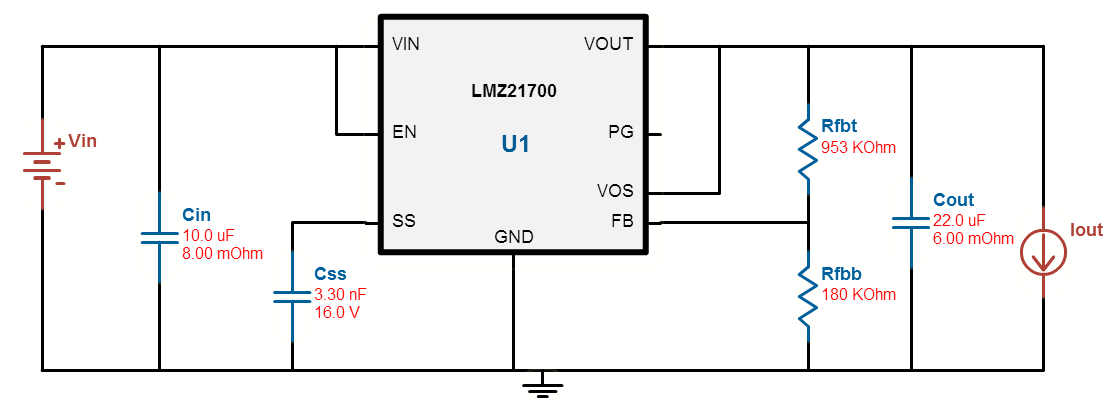
Active regulators use a least one active amplifying component like a transistor or operational amplifier. There are a few classes; such as linear regulators. These are placed between the source and the load to have a constant voltage going to the load. Linear Regulators usually contain a Zener Diode. Adding an emitter follower stage to the simple Zener regulator forms a simple series voltage regulator and substantially improves the regulation of the circuit. The most common linear regulators are the three terminal integrated circuits in the TO220.

A circuit design to have a 3.3V input from the power source is as follows. This will be used for the main board. Figure 3.4 G shows the schematic of the regulator.



*Figure 3.4 G: Regulator schematic.*

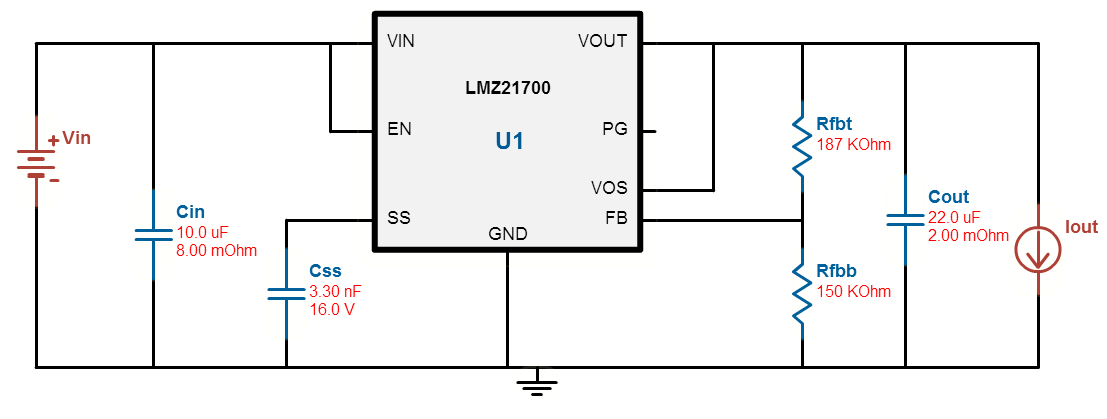
Another rectifier will be need for the amplifier. The circuit schematic is as shown in Figure 3.4 H.



*Figure 3.4 H: schematic of the Rectifier.*

The next class of regulators are the switching regulators. Theses regulators rapidly switch a series device on and off. The duty cycle of the switch sets how much charge is transferred to the load. This is controlled by a similar feedback mechanism as in a linear regulator. Because the series element is either fully conducting, or switched off, it dissipates almost no power; this is what gives the switching design its efficiency. Switching regulators are also able to generate output voltages which are higher than the input, or of opposite polarity — something not possible with a linear design. The main advantage of this is the efficiency, the small size, and lower heat. Disadvantages are complexity, high frequency energy that the low pass filter must clock and ripple voltage.

Another rectifier will be needed to have an input of 1.8V. The follow is the schematic diagram Figure 3.4 I.



*Figure 3.4 I: schematic of Rectifier*

These schematics may or may not change based off decisions made in building the device. The following is a table on what power will be needed.

**3.5 Audio Input Research**

**3.5.1 Microphone Selection**

The group would like the primary device to implement both voice commands and conversation detection. Implementing these features will require a microphone, a sound card and a stereo jack. To keep the device both small in size and light in weight, we would like to use a relatively small microphone and mount it to the device.

Depending on which card we decide to get we need to be aware of the minimum signal level being at least 10 millivolts and for larger bit cards maybe 100 millivolts. This parameter will be known on the data sheet of the microphone we decide to buy.

An electret microphone is a type of microphone that would be useful. This kind of microphone is an electrostatic induction microphone, which means we have no need of using a permanently charged material to supply the microphone. But due to the FET there will a need for power to turn on the FET. The resistor value can be from 1k to 22k. The capacitor value can be between 1uF and 22uF.

The most common type of electret microphone is the simple modification. It is used by most PC’s and this will be set up by the following. Connect the shield to connector’s sleeve, then connect the ring and tip to the central conductor. However there is a stereo microphone inputs that can be used for high performance speech recognitions and advanced noise canceling applications. These are called unidirectional electret microphones and are connected differently than the simple model. Connect the shield of both microphones to the sleeve of the plug, the left mic to the tip and the right mic to the ring.

**Microphones**

After looking at microphones there were a few that have a possibility of working. They are the SparkFun Electret Microphone Breakout, MEMS Microphone (INMP401), USB Plug Mini Microphone Desktop Studio Speech Recording F Skype MSN and Mini 3.5mm Flexible MIC Microphone for Laptop PC Skype MSN.

Tables 3.5 A – C, below, shows the important specifications for various microphones under consideration.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Size | Frequency Range | Sensitivity | Load Resistance | Supply Voltage |
| SparkFun Electret Microphone Breakout | Length/Diameter = 9.7mm  Height = 4.5mm | 100 to 10000Hz | -46 plus or minus 2 dB | 680 ohms | 1.5V |
| ADMP401 MEMS Microphone | 4.72mm x 3.76mm x 1.0mm | 100Hz to 15kHz | -42 dB | 200 ohms | 1.5V to 3.3V |
| USB Plug Mini Microphone | USB size | N/A | N/A | USB resistance | USB voltage |
| Mini 3.5mm Flexible MIC | 12 cm long | 30Hz to 16000Hz | -62 plus or minus 3 dB | 3.5 mm audio jack | 3.5mm audio jack |

*Table 3.5 A: Microphone specifications.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Supply Current | Working Voltage | Input | Additional comments |
| SparkFun Electret Microphone Breakout | Max 0.5 mA | 1V to 10 V DC | Termination Pins | Need to connect to board we decided to buy |
| ADMP401 MEMS Microphone | 250 uA | 2.7 to 5.5 V | Termination Pins | Need to connect to board we decided to buy |
| USB Plug Mini Microphone | USB current | 5V | USB | Need to buy a USB input |
| Mini 3.5mm Flexible MIC | 3.5 mm audio jack | 3.5 mm audio jack | Microphone is connected through a 3.5 mm audio jack | Need to buy a audio jack |

*Table 3.5 B: Microphone specifications, continued.*

|  |  |  |
| --- | --- | --- |
|  | Maximum Sensitivity to Noise Ratio | Cost |
| SparkFun Electret Microphone Breakout | 58 dB | $7.95 |
| ADMP401 MEMS Microphone | 62dB | $3.95 |
| Mini 3.5mm Flexible MIC | N/A | $2.79 |
| Mini 3.5mm Flexible MIC | N/A | $1.97 |

*Table 3.5 C: Microphone specifications, continued further.*

After looking at these microphones the SparkFun microphone looks like the best available option. Next, the output will need to be amplified to be useable by a soundcard or by the MCU; for this task, an operational amplifier will be used.

**Operational Amplifiers**

Options for the op amp used to be used in the project include the OPA series 344 and the TI LMV 1032. The OPA series is directly recommended for the use of the SparkFun microphone. They are used for precision, low power, and miniature applications. They are ideal for analog to digital converters. They cost $1.72 each.

The LMV 1032 is made for 3 wire analog electret microphones. This would reduce the supply voltage needed and would cost $1.67. The LMV 1032 and OPA 344 are detailed in Table 3.5 D, below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Operating Voltage Range | Gain Bandwidth | Cost | Maximum Input signal |
| LMV 1032 | 1.7V to 5V | 6.2 or 15.4 or 25.5. | $1.67 | 170mVpp |
| OPA 344 | 2.5V to 5.5 V | 1 MHz, G >1 | $1.72 | 0.5 beyond the rail voltage |

*Table 3.5 D: Op amp specifications.*

**Audio Jacks**

Options for 3.5mm jacks include the PRT-08032 ROHS and the BOB-11570 ROHS. Both of these jacks are similar but after doing some research the decision on which to use is dependent on how voice commands the voice commands will be processed. Further details on these 3.5mm jacks are contained in Table 3.5 E, below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cost | Pins | Comments | Size |
| PRT-08032 ROHS | $1.50 | Ground, Left and Right | Pins are very breakable | 12x11x5 cm |
| BOB-11570 ROHS | $3.95 | Tip, Ring1, Ring2 and sleeve | No connector | 2x1 cm |

*Table 3.5 E: 3.5mm jacks.*

If the mic with the USB option is chosen then we would need to buy a board that can process audio and have an available USB to serial breakout. There is the BOB-00718 ROHS that costs $14.95. But this seems like the least desirable option.

**3.5.2 Voice Command Processing Research**

Implementing voice commands is a moderately high priority object for the project; they will help in reducing the amount of user effort required to control the device. The group would like to implement, at minimum, the following single word commands:

|  |  |
| --- | --- |
| Command | Action |
| ON | Wake the device up from sleep mode and begin playing music. |
| OFF | Stop playing music and enter sleep mode. |
| PLAY | Begin playing music (used to exit PAUSE). |
| PAUSE | Stop playing music. |
| SKIP | Stop playing the current song and begin playing the next song. |
| LOUDER | Increase the volume. |
| SOFTER | Decrease the volume. |

*Table 3.5 F: Single word voice commands.*

The group must take precautions to keep voice commands from being activated by accident, such as when any of the above command words are used in conversation. One way to deal with this would be to begin all commands with a less common word, such as ‘Helium’.

**Speech Recognition**

Speech recognition can be classified as either speaker dependent or speaker independent. Table 3.5 G details the pros and cons of both of these methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Details | Pros | Cons | Comment |
| Speaker dependent | User trains the system with commands | High command count  High accuracy of word recognition | The system only responds to the person that trained the system. | Most used |
| Speaker independent | System is trained to respond to words | Responds to whoever speaks. | Command word count is usually lower. Lower accuracy | Used in telephone systems |

*Table 3.5 G: Speech recognition, speaker-dependent vs –independent.*

Speaker dependent system is a system where the speaker will read sections of text into the speech recognition system and then match the command with the text. This will use the speaker’s personal voice and being more accurate for each individual person. Speaker independent system is a system where there is no personal input of voice commands. Both systems have pros and cons to both. For the speaker dependent, it would be nice for us to use exact voices and exact commands to reach our objective. However, we would all need to individually input our voice commands since we all have different voices. Then this will make the device an exclusive device to the four of us. This would not be good for commercial use because a consumer would like for the device to already know what he or she is saying.

How this works is we say words and the microphone on the device will take that audio input and then send this to software. The software will use speech recognition and then we create an action that matches which command we say. As vocabulary size increases, the percent error increases as well.

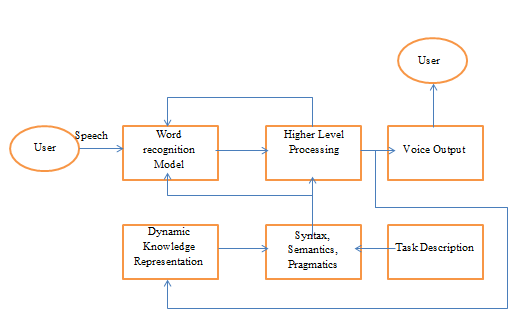
There will be some tradeoffs for the proper objective and the accuracy of the speech recognition. Using single words is by far the most accurate. As our commands get more complicated than one word then the speech recognition could become more difficult. Discontinuous speech is when the speech has sentences that are separated by a moment of silence. Then there is continuous speech. This means when someone is speaking and the separation between sentences are rather difficult this is called continuous speech. This is more difficult for the speech recognition to translate.

We would need to add in the word “Ummm” because of people’s nature to say ummm when they talk. This we can just exclude the word from having any addition to the text or have any action. Accounting for adverse conditions needs to be factor in our decision of voice commands. Condition that would lower the accuracy of the voice commands would be echoes, room acoustics, too many people talking, and outside noises.

Voice commands are taken in by each individual sound and then put together through the software. Meaning every signal is broken down into smaller more basic sub signals. The process the software goes through is as follows. First, digitizing the speech, the normal sampling rate is 8000 samples per second. The next step is to analyze the frequencies of the speed in small intervals. A normal time interval for this is ten milliseconds. A sound can be described by amplitude and frequency. So when putting the speech into many frames the software can now look at the individual frames and digitized the sound waves with numbers. This is represented in an analog wave form. The following step is classifying the now analog speech based off of statistical analysis. Then finally matching the sound with words based on the data saved in the speech database.

The usable voice frequency band range is 300Hz to 3400Hz. Average adult male voice frequencies is 85 to 180Hz. Average adult female voice frequencies is 165 to 255 Hz.

Here is a diagram of speech recognition, Figure 3.5 H.



*Figure 3.5 H: speech recognition diagram.*

There is another way to categorize voice recognition and that is through styles of speech shown in Table 3.5 I.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Details | Pros | Cons | Details |
| Isolated | Commands in short lengths. Approximately one second long | Can use easy commands | Must pause between each word | Most common |
| Connected | Command length is approximately 2 seconds | Commands can be multiple words | Word recognition number to around 20 | Half way between isolated and continuous. |
| Continuous | Is continuously taking in commands | Available to take commands whenever | Difficult for a recognizer to find a command. Low accuracy. | Under continual development |

*Table 3.5 I: speech recognition applications.*

There is a way to use voice recognition through hardware and software. Through hardware, buying a module would be wise. The following are some modules

**Voice Recognition Module – Arduino Compatible, Model: AU\_VOI\_REC\_00A**

This module can recognize up to 15 voice instructions and the package includes a microphone. To connect this to our main board a USB module will be needed. To program the voices a USB TTL module will also be needed.

**EasyVR Shield 2.0 – Voice Recognition Shield**

Can hold 28 custom speaker commands and as a UART interface. Table 3.5 J-K compare the two voice recognition modules.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Size | Saved Commands | Price | Required voltage | Current |
| Voice Recognition Module -- Arduino Compatible | 30 by 47.5 mm | 15 | 23.50 | 4.5-5.5V | Less than 40mA |
| EasyVR Shield 2.0 - Voice Recognition Shield | 7.2 by 5.5 cm | 28 | 49.95 | 3.3 to 5 V | 12mA |

*Table 3.5 J: Voice recognition modules*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Digital Interface | Analog Interface | Recognition Accuracy | Comments |
| Voice Recognition Module -- Arduino Compatible | 5V TTL level UART interface and GPIO | 3.5mm mono-channel microphone connector plus microphone pin interface | 99 percent | This module has many different options and settings. Require USB TTL module |
| EasyVR Shield 2.0 - Voice Recognition Shield | Serial interface | Microphone jack | 99 percent | Manual is very helpful and has many modes. |

*Table 3.5 K: voice recognition module.*

Both serial part settings are the same

1.Baud rate: 9600

2. Parity bit: None

3. Data bit: 8

4. Stop bit: 1

If the first option, voice recognition module – Arduino compatible, is chosen a USB TTL module would be require to save voice commands. If the latter, easyVR, has a USB cord that comes with the package that allows us to save commands.

The following are options for the USB TTL module.

USB TLL Module with 5V or 3.3V compared to CP2102 is shown on Table 3.5 L.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Voltage | Compatible | Price |
| USB TLL Module with 5V or 3.3V | 3.3 or 5 V | Windows, Linux and MAXOS | $4.90 |
| CP2102 | 3.0 to 3.6 V | Windows, MAC, Linux | $5.70 |

*Table 3.5 L: Comparison of USB TLL and CP2102.*

**Voice Recognition through Software**

Most of the software for voice recognition is used in personal computers. A compatible sound card and a computer would be required. This method is better for voice recognition but not practical for anything other than being convenient. There are plenty of software’s that are being sold from $45 to #130. Then there many free software’s that read text back and one that caught attention was the “Free voice to text.” How this would if the microphone took into audio, then translates the voice into text, and then the text can correspond to a certain command. The issue would be for the software to stop converting and to just convert the wanted commands. Table 3.5 M compares the pros and cons of hardware vs. software for voice recognition.

|  |  |  |
| --- | --- | --- |
|  | Pros | Cons |
| Hardware | Easy to buy and use. Package is convenient | Limited to hardware’s parameter |
| Software | Freedom to have many voice commands. Flexible to user’s needs. | Requires computer and compatible sound card. Not economically viable. Needs knowledge in programming |

*Table 3.5 M: voice recognition hardware vs. software.*

**3.6 Audio Output Research**

**3.6.1 Graphic Equalization**

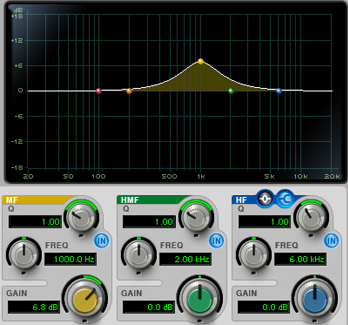
Graphic Equalization is a tool used to customize the sound qualities of the music you are listening to. The basic principle is that you can modify the way the music sounds by either cutting or boosting the gain of a band-pass filter embedded within the equalizer circuit. A circuit will contain multiple filters that allow control of the entire audible range of the human ear (20 Hz to 20 kHz). Each frequency has a different sound, and by manipulating the gain by boosting you can make a certain frequency range louder to create a personal sound on the recording. An equalizer is commonly seen on DJ equipment with multiple sliders centered on a certain frequency. An example of such a setup is shown below in *Figure 3.6.1 A*.



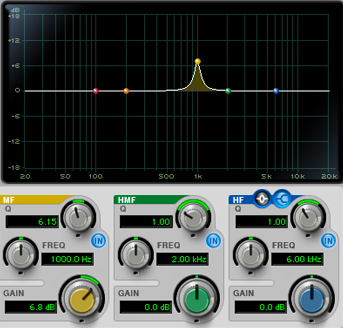
*Figure 3.6.1 A: Audio Equalizer with sliders*

This particular setup has 10 sliders, which means this is a 10 band graphic equalizer. There are two things to note when dealing with equalization: the type of equalizer; of which there are two types most generally used, graphic equalizers and parametric equalizers; and the other is the number of bands.

First, the difference between a graphic equalizer and a parametric equalizer is, primarily, the amount of customization possible. A graphic equalizer can only control one aspect of the frequency, level (cutting or boosting the gain of each band). While a parametric equalizer can control three aspects of the frequency: level; center frequency, which determines the frequency at the peak of the curve for the band-pass filter; and bandwidth, which can be made to allow more frequencies through the filter. The most important benefit of parametric equalization comes from the ability to modify the bandwidth as well as cutting and boosting. While it’s nice to be able to change the center frequency because you can isolate the most important frequency to boost, it’s also not as vital to the design for this project. The main purpose of an equalizer for this project would be to remove distorted frequencies to make a smooth sound come out of the speakers. This would be accomplished most efficiently by altering the bandwidth. If there is a frequency that causes distortion or undesired sound, the best way to address that problem would be to effectively pinch the curve so that only a small range of frequencies are boosted. The bandwidth is inversely related to the quality factor, Q. Q is measured as the center frequency divided by the bandwidth, thus the inverse relationship. The quality factor is the actual parameter that can be modified to allow the desired bandwidth through the filter. This can be best demonstrated in the two figures below.



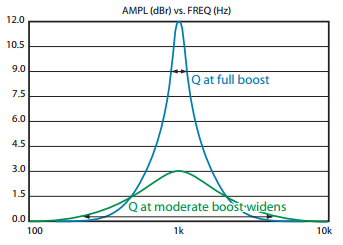
*Figure 3.6.1 B: Parametric equalizer with small Q value, large bandwidth*



*Figure 3.6.1 C: Parametric equalizer with large Q value, small bandwidth*

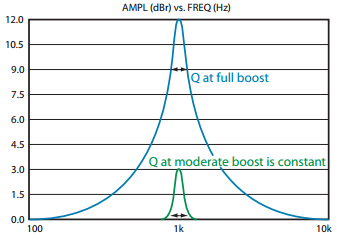
In *Figure 3.6.1 B*, note that the low Q value flattens the curve of the filter allowing a wider bandwidth to be boosted at 6.8 dB. *Figure 3.6.1 C* shows the same signal that has been modified with a higher Q value of 6.15, as opposed to 1. The bandwidth has been decreased significantly by simply increasing the Q, and most of the frequencies that were boosted in the previous setup no longer pass through the filter. In this manner, trouble frequencies can be removed from the signal creating a smoother sound. Not to say that a graphic equalizer can’t remove distortion or trouble frequencies. Graphic equalizers are just as useful, but lack the depth that the parametric equalizer can achieve.

Now, back to the second part of an equalizer that determines the effectiveness, the number of bands. A high quality graphic equalizer usually contains either 15 bands with each bandwidth set at 2/3 octave widths, or 30 bands at 1/3 octave widths. However, there lies a problem with conventional graphic equalizers using a set bandwidth. For example, when you set the bandwidth to be 1/3 octave, you create a new value for Q on each band because the equation for Q is given as Q = ; where is the center frequency and is the bandwidth. Both values are preset for each band, so Q is always the variable being solved for. This becomes a problem because ideally, if a single slider is boosted 3 dB, then only the 1/3 octave frequency band passes and is actually boosted. What happens is the filter only maintains a bandwidth of 1/3 octave at extreme boost or cuts. At lower gain boost/cut positions, the bandwidth of the modified filter is greater than 1 octave as shown below.



*Figure 3.6.1 D: Conventional 1/3 octave graphic equalizer*

What should be happening here is the modified bandwidth should only allow the frequencies within the 1/3 octave bandwidth to be boosted. Instead of setting a value for the bandwidth, a value needs to be set for Q. This is called a Constant-Q graphic equalizer, and it allows for the modified bandwidth to remain at 1/3 octave for each band and with any gain applied. The figure below shows the result when implementing a Constant-Q equalizer.



*Figure 3.6.1 E: Constant-Q equalizer*

By keeping a constant Q, each filter can be designed for the center frequency and narrow bandwidth required.

Going back to the number of bands, a parametric equalizer typically only has four or five bands, with a maximum of eight. This is due to the fact that there are three control knobs for each band, which takes up space, component resources, and adds complexity to the device.

To summarize, parametric equalizers offer very precise manipulation, but the limited number of bands, added component cost, and distortion makes the graphic equalizer a better choice. A 1/3 octave constant-Q equalizer, specifically with only 10 bands. The scope of this project does not call for any more than that. Studio quality 15 or 30 band equalizers are overkill. All the added component cost, design, and implementation should not improve sound quality enough to justify incorporation into the system.

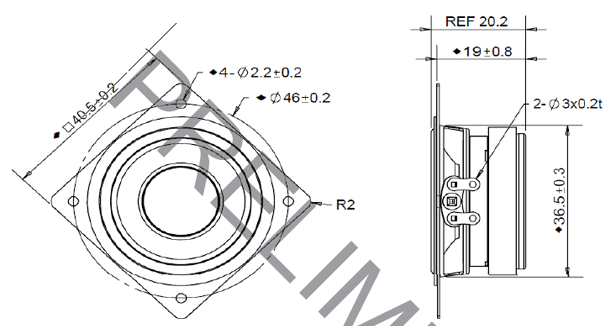
**3.6.2 Speaker**

There are a few main components to consider when selecting a speaker for this system. First and foremost is to cover as much of the human audible frequency range as possible. The average human can pick up frequencies from 20 Hz to 20 kHz. It is very difficult to find a speaker that can produce sound at 20 Hz. Most speaker that can accomplish this have a very large power consumption and high cost, which does not work well with this system. There are many speakers that can get down to around 200 Hz on the low end of the frequency range. The speaker also needs to produce these sounds at a decent volume. So, there are four major components that are taken into consideration for selection. Frequency range, power, volume, and cost. Because of the scalability of this system, three options will be considered. Essentially, a high-end, middle-tier, and low-end option. The four criteria are presented for each option in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Peerless PMT30N18AL04-04 19mm (3/4”) Full Range | Peerless TC5FB00-04 1.5" Full Range Paper Cone | Peerless TC7FD00-04 2.5” Full Range 4 ohm |
| Range | 250 Hz – 20 kHz | 200 Hz – 20 kHz | 130 Hz – 20 kHz |
| Power | 1.5 W | 5 W | 20 W |
| Volume | 78 dB | 80.1 dB | 82.4 dB |
| Cost | $9.15 | $9.75 | $9.85 |

*Table 3.6.2 A: Speaker Comparison*

Looking at the options presented, the speaker with the most desirable range is the 2.5” speaker, or high-end option. The most desirable power option is the ¾”, low-end option. High-end has the best volume, and low-end has the cheapest cost. Given the options, the 1.5” speaker, though it didn’t have the best of anything to offer, is the ideal candidate at this point in time. It has a low enough power requirement to implement easily, sufficient volume, and a pretty desirable frequency range for the price. If it turns out that 80.1 dB isn’t loud enough for the system, then higher volume speakers can be implemented at a higher cost and with more power. Calculations will be made to see how much power can be supported in the event that a more powerful speaker can in fact be used. That is the main concern at this point, making sure the system can handle it. Each category will increase in efficiency if an amplifier can supply the right amount of voltage. The 1.5” speaker is also a suitable option for the system given its small size. The 1.5” is referring to the size of the transducer. The full diameter of the speaker with housing is roughly 1.6” (40.5 mm). The measurements are shown below, in millimeters.



*Figure 3.6.2 A: Speaker measurements in millimeters*

The height is 36.5 mm (1.44”), and the width is 20.2 mm (0.80”). This makes for a very small, lightweight design. Here is a graphic comparing the size of the speaker to a quarter, for reference.



*Figure 3.6.2 B: 1.5” Speaker size reference*

**3.6.2 Amplification**

The ability to adjust volume is a necessity for any music playing device. This can easily be achieved by sending the audio output of the device through an amplifier, whose gain is controlled by the MCU, before sending it to any speakers. According to one source on volume control, what is typically done when the volume is increased by a displayed percentage is that the output voltage is adjust by the volume percentage squared (see Equation 3.6 A). Example 3.6 A, below, explains this more clearly.

*Equation 3.6 A:*

*Vo = Output Voltage, VL = Volume Level (expressed as a percentage)*

*Vo = VL^2\*MaxVoltageOut*

*Example 3.6 A:*

*If the volume of the device is set to 50%, then the output voltage, Vo, will be 25% of the maximum output voltage*

By controlling volume using the method described in Equation 3.6 A, an increase in volume should be more noticeable. The group will test both this method and a linear increase method (i.e. 50% volume level = 50% of maximum output voltage); to determine if this method is actually the best. Unless otherwise indicated, any mention of volume control will refer to the method described by Equation 3.6 A.

**3.6.3 MP3 Decoder**

In order for Helium to be able to play .mp3 files, it will need to be able to decode them. These files can either be decoded using software or hardware. Since the group’s software workload is already somewhat disproportionate to the group make up (3 EEs and 1 CpE), the group will use a hardware device for decoding the .mp3 files.

The company, VLSI solutions offers many ICs with the ability to both decode and encode MP3 files, as well as audio in many other formats. A few of their ICs are compared below

**VS1011**

The VS1011 is a basic MP3 and WAV decoder. It offers the following features:

* Decodes MP3 and WAV files;
* Up to 320 kbps MP3 quality;
* Streaming support for MP3 and WAV;
* Operates from 2.3V to 3.6V;
* High quality DAC converter;
* 5.5 KiB RAM on chip for user code;
* Low power operation;
* Bass and treble controls;
* Serial control and data interfaces;
* 4 GPIO pins
* Costs ~$20

**VS1063**

The VS1063 is a more advanced audio codec. It offers the following features:

* Encodes (in mono or stereo) at 8-48 kHz the following formats:
  + MP3;
  + Ogg Vorbis;
  + 16-bit PCM;
  + IMA ADPCM;
  + G.711 u-law and A-law;
  + G.722 ADPCM;
* Decodes the following formats at up to 48 kHz:
  + MP3;
  + MP2;
  + Ogg Vorbis;
  + MPEG4 / 2 AAC-LC(+PNS), HE-AAC v2 (Level 3) (SBR + PS);
  + WMA4.0/4.1/7/8/9 all profiles (5-384 kbps);
  + FLAC lossless audio (up to 24 bits);
  + WAV: 16-bit PCM, IMA ADPCM, G.711 u-law and A-law, G.722 ADPCM;
* Streaming support;
* Upto 96 KiB RAM for user code and data;
* Unique ID for code protection;
* I2S interface for external DAC;
* High-quality stereo A/D converter;
* Serial control and data interfaces;
* Can be used either as a slave co-processor or as a standalone processor;
* UART for debugging purposes;
* New functions may be added with software and 12 GPIO pins;
* Ear Speaker Spatial Processing;
* Bass and treble controls;
* 5 band equalizer;
* Low-power operation;
* High-quality on-chip stereo DAC with no phase error between channels;
* Zero-cross detection for smooth volume change;
* Quiet power-on and power-off
* Separate voltages for analog, digital, I/O
* SPI flash boot for special applications
* Costs ~$31

Although the VS1063 offers many more features and is capable of decoding in more formats than the VS1011 for an insignificant increase in price, the IC offers many features that the group won’t need it to perform. The group would like to build our own 5 band equalizer, so there is no need for the one included with the VS1063. The group is not interested in decoding the device being able to decode the Ogg Vorbis or WMA formats. And the group does not need an IC capable of encoding. Because of these several reasons, the group will use the simpler, and cheaper, VS1011. It offers all the features that we would like at a lower cost.

**3.6.4 High Fidelity Audio**

The group would like to design a device capable of delivering high quality audio output; high quality audio is typically referred to as “high fidelity.” High fidelity audio can be described by the following characteristics:

* Low noise
  + All digital audio will have noise due to the quantization process;
  + Thermal noise will always be produced in the circuitry for temperatures above 0 K
  + The group would like to aim for a signal-to-noise ratio (SNR) of at least 90 dB
* Little distortion
  + Phase distortion can be caused by filters (used for equalization) with a non-linear relationship between phase and frequency
    - The audibility of phase distortion can depend on various conditions
    - Bessel filters have a linear relationship between phase and frequency
  + Harmonic distortion can be caused by the non-linear components
    - High fidelity devices tend to have < 5% Total Harmonic Distortion
* Accurate frequency response
  + Excluding any desired equalization effects
  + Humans can hear at around a range of 20 – 20k Hz

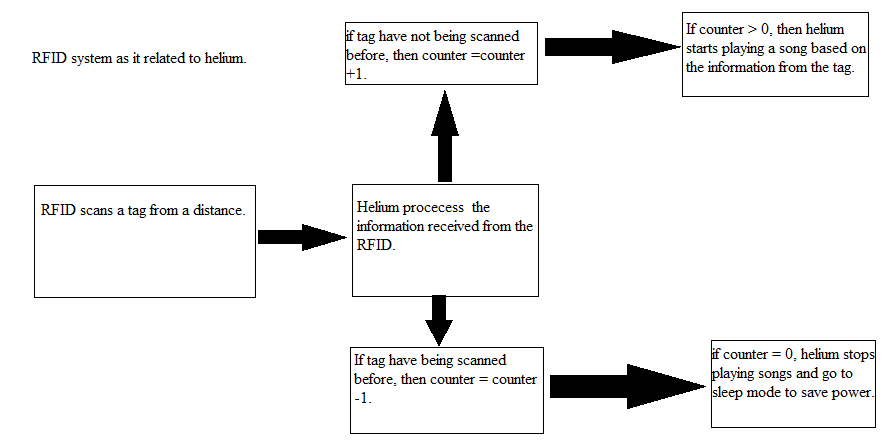
**3.6.5 Volume Leveling**

The group would like to implement some sort of volume leveling module; due to the “Loudness War,” newer music can be significantly louder than older music. The group would like to prevent any harsh changes of volume between songs. There are a few ways to do this, both through hardware and software. The most difficult option of those would be to use hardware. This would require “gain control” and using a board that has this function. However, software seems so much easier. There are two programs that are fee to download online. One is MP3 Gain and the uniqueness of this program is that MP3Gain does not just do peak normalization but instead the program uses statistical analysis. At the end of using this program all of the songs will have the same volume. This program will be needed to go through the range of songs that would be on a memory card.

Platinumnotes is another program that is recommended. This program told to be very easy to use and user friendly. The program uses studio filters to process files, corrects pitch, and improves volume. This program was created by the developers behind Mixed In Key.

**3.7 RFID Research**

One of the group’s high priority objectives for the project is to have Helium be capable of determining the number of people in the room that it is located; in addition, the device should be able to play songs without user input. The device must be able to determine the person that came in to the room and to track when that person leaves the room. One such device is an RFID system, in which the reader must be able to read at a distance the transponder. Another reason is that the tags that have the information, does not matter if it’s covered or not, the RFID reader can see it. The Figure 3.7 A shows an illustration how the RFID should work with Helium.



*Figure 3.7 A: Information flow chart of Helium*

**3.7.1 RFID Transceiver**

The Transceiver, is the base station of the RFID reader. Its main job is to demodulate the signal received from a tag, and to be able to modulate signals to tags, to turn on or to reprogram the tags. The range of the reader depends on the type of frequency, the range is shown on Table 3.7 B. By creating a mobile RFID reader, the group decided that the range needs to be at most 1m which is enough range to cover the entrance of a room and be able to read the tags.

|  |  |
| --- | --- |
| 120-150 kHz | 10 cm |
| 13.56 MHz | 10 cm-1m |
| 433 MHz | 1-100m |
| 902-928 MHz | 1 to 12m |
| 2450-5800 MHz | 1 to 2 m |
| 3.1-10 GHz | 200m |

*Table 3.7 B: The range of the reader depending on the frequency it operates at.*

Since the range that Helium will need to reach is at the least 50 cm, the group decided to get a 13.56 MHz frequency base station chip to reach the required range that is stated above. The table below, Table 3.7 C, illustrates the different transceivers that can be used for the RFID and some of its specifications.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **EM4094** | **TRF7970A** | **ST95HF** |
| Operational voltage (Vin) | 3.3V – 3.6V  4.5V – 5.5V | 2.7V – 5.5 V | -0.7V – 5.5V |
| Frequency | 13.56 MHz | 13.56 MHz | 13.56 MHz |
| Cost | $10 | $19.10 | $5.32 |
| Operation mode | Sleep mode/  Active mode | Sleep mode/  Active mode | Sleep mode/  Active mode |
| Transmission rate | 10 kbps |  | 26 kbps |
| Support NFC? | Yes | yes | yes |
| Multiple sub-carrier? | Yes | No | no |
| Operational junction temperature | 85°C | 125°C | 85°C |

*Table 3.7 C: RFID transceiver specifications*

**3.7.2 RFID transponder**

The transponder, or tags, is the part of the RFID that holds the information that will transmitted to the reader. Depending on which tag is bought, the tags will come in either a passive tag, an active tag, or a semi-active passive. An active tag comes with its own power supply and always transmitting its signal. The semi-active tag uses a battery to help send the signal to the RFID reader and will be asleep while no RFID is present. The passive tag does not have a battery, and it only transmit its signal by the power of the readers. Another part to consider is that some tags come with the ability to be read-only or read/write. Table 3.7 D compares and contrast the different type of tags.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Power | transmission | cost | Signal strength  required | Range |
| Active tags | Battery assisted | Always transmitting | $15 to $100 | low | 100+m |
| Semi-active tags | Battery assisted in sending the signal. | Transmits when the RFID reader is near by | $15 to $100 | low | 3m |
| Passive tags | No battery uses. Uses the power from the reader. | Transmit when RFID reader is nearby. | $0.15 to $5.00 | Strong | 3m |

*Table 3.7 D: The different characteristics of tags.*

Below, Table 3.7 E, will list some of the transponders that provide the 13.56 MHz used to communicate with the transceiver.

|  |  |  |
| --- | --- | --- |
|  | EM4135 | MFRC522 |
| Data storage | 2.4 Kbits of EEPROM memory |  |
| Function | Read/write compatible | Read/write compatible |
| Can data be permanently saved? | Yes | yes |
| Range | 50 mm | 50mm |
| Type of tag | Passive tag | Active tag |
| Identifier | 64-bit unique identifier |  |
| Cost | $6.00 | $6.44 |

*Table: 3.7 E: Selection of transponder*

**3.7.3 Digital Processor**

To be able to use the demodulated data, the RFID must be connected to a microcontroller. Depending on the code, the MCU will authentic the tag, and read the information within. Since the group is planning on having the Reader be mobile the MCU must be able to transfer the data from the tag to the main location of Helium, where the playlist is located to be searched. Since we are going for low cost and low power usage some of the MCU to choose are below, Tabl3 3.7 F.

|  |  |  |
| --- | --- | --- |
|  | **MSP430F5529lp** | **STM32L0xx** |
| Max supply voltage | 3.6V | 1.65V – 3.6V |
| Power consumption | Low power | Low power |
| System clock | Up to 25 MHz | Up to 32 MHz |
| Cost | $5.39 |  |
| Operation | 16-bit RISC | 32-bit ARM cortex |
| Memory | 128kb flash memory | 64kb flash memory |
| Max junction temperature | 125°C | 125°C |

*Table 3.7 F: Digital processor for the RFID reader.*

**3.7.4 Bluetooth**

As stated above, to have greater flexibility with the RFID reader, the reader must be mobile. One of the ways to have the reader be mobile is to add Bluetooth capabilities to the MCU. The reason why Bluetooth works is that since Helium will be located inside the room where the RFID will be used the longer range provided from Wi-Fi hence power can be saved by using Bluetooth instead of Wi-Fi. Bluetooth also has a symmetrical connection between Bluetooth devices allowing for Helium and the reader to connect to each other without a problem. Another reason to use Bluetooth is that because it works only for short ranges, it can be set up to avoid interference with other frequency bands when sending the signal to the main body of Helium. Some of the Bluetooth modules that can be used are listed on the table below, Table 3.7 G.

|  |  |  |
| --- | --- | --- |
|  | PAN1315ETU | HC-05 module |
| Power supply | 1.7V-4.8V | 3.3V |
| Power scan | 135uA | Low power |
| Bluetooth version | V2 | V2 |
| Operating temperature | -20°C to 70°C | -40°C to 85°C |
| Cost | $71.61 | $7.80 |

*Table 3.7 G: Bluetooth selection table.*

Whichever device is purchased for Helium, the organization will be that one of the Bluetooth device, the one from the RFID, will act as the master for setting up the connection. Once the connection has being established, the RFID system will be turned to a slave module to send the required data back to the main body of the Helium. While setting up the Bluetooth device to function for communication with the RFID and Helium, the Bluetooth device will be set up to allow other Bluetooth devices such as cellphones to control Helium as an alternative to using the RFID device or the NFC function if the user wish to add new songs or control other aspects of Helium.

**3.7.5 LCD MODULE**

An LCD display would help to in figuring out that state the RFID reader it is in. By using an LCD module, different messages can be displayed on screen as a case of troubleshooting if there is a problem or just to know if the RFID is ready to accept a signal from a transponder. This would work great to see if the connection between the RFID works or not as the information inside the transponder can be displayed on-screen and then compare the result to the results given to Helium. Some LCD that could be used to work with the RFID is given below at Table 3.7 H.

|  |  |  |
| --- | --- | --- |
|  | **RioRand™ LCD Module** | **Color LCD Booster Pack for MSP430.** |
| Display | White on blue display | 2.2”, 320x240 pixel color LCD |
| Display size | 20x4 character display | See above. |
| Backlight | Can use potentiometer to control the backlight | Optional backlight transistor |
| Cost | $12.00 | $16.00 |
| Pins | Have to be soldered | Can be attached to MSP430 |
| microSD card | No slot | Open slot |
| Programming | Can use energia to convert from arduino library. | Can use ccs, energia and has support from ti. |

*Table 3.7 H: selections for LCD module.*

**3.8 Statistical Models for Song Selection**

**3.8.1 Naïve Bayes**

Naïve Bayes is a relatively simple method of creating a statistical model for a classifier. It is based off of Bayes theorem, which states, *P(C|s) = P(s|C)\*P(C) / P(s)*. A Bayes classifier would work as follows:

*Given an item, s, with one of n possible classifications, C0 - Cn, the most likely classification for s is the classification that maximizes:*

*P(Ck|s) = P(s|Ck)P(Ck) / P(s)*

*Since, for all n possible classifications, the item, s, would be the same, the above can be reduced to the form:*

*P(Ck|s) = P(s|Ck)P(Ck)*

In the case of song selection, there are two possible classifications: *C0, user wouldn’t skip*, and *C1, user would skip.* While calculating the probability that a user would skip or not skip, *P(Ck)*, is a simple matter of keeping track of how often a user skips songs, calculating the probability that, for a particular song, the user would skip that song, *P(s|Ck),* is a little more difficult. A user would have to listen to the same song numerous times before an accurate measure of *P(s|Ck)* could be calculated. For thousands, or even just hundreds, of songs, this is impractical.

To facilitate the calculation of *P(s|Ck)* for what could, theoretically, be an infinite number of songs, each song will be represented by a finite collection of **features**. For each feature, x, the probability *P(xi|Ck)* can be calculated much more easily. Assuming that the probability of each feature is independent of one another, *P(s|Ck)* could be expressed as follows:

*For a song with features x0 through xm,*

*P(s|Ck) = P(x0|Ck)\*P(x1|Ck)\*…\*P(xm|Ck)*

However, the assumption that each feature is independent of one another is NOT true, and therefore, the above equation is NOT mathematically correct. In practice, the above equation does work reasonably well despite the incorrect assumption of independence. This *naïve* assumption of feature independence is where the classifiers name, *Naïve Bayes*, comes from. Typically, Naïve Bayes classifiers see an accuracy rate of around 80-85%.

The group has some experience in creating Naïve Bayes text classifiers. The group already has access to code for Naïve Bayes text classifiers in both Python and Java. This code can easily be repurposed for use in the song selection algorithm. Using this method would give the group a fairly significant head start in coding the song selection algorithm.

**3.8.2 Logistic Regression**

Another potential option for generating the statistical model of the song selection algorithm is to use logistic regression; in particular, binomial logistic regression (since there are only two possible classifications). Typically, classifiers based off of some form of regression tend to perform better than classifiers based off of Naïve Bayes. The tradeoff for this better performance is that creating a model using logistic regression is more difficult.

Logistic regression is based off of the logistic function (see Equation 3.8 A), whose output will always be between 0 and 1, meaning that it can be interpreted as a probability. The independent variable, t, in the Logistic Function, represents a function of a linear combination of independent variables; this is referred to as the logit (see Equation 3.8 B).

*Equation 3.8 A: The Logistic Function*

*F(t) = 1/[1+e^(-t)]*

*Equation 3.8 B: The Logit Function*

*t = g(x1, x2,… xn) = B0 + B1\*x1 + B2\*x2 + … + Bn\*xn*

In binomial logistic regression, each x is a predictor with a value of either 0 or 1 and each coefficient, B, is a weight applied to that predictor (with B0 being the intercept). For each song, if a predictor is present, it will be represented with a 1, and a 0 otherwise. When a coefficient takes on a positive value, it indicates a *case*; and when a coefficient takes on a negative value, it indicates *not a case.* In the case of song selection, a *case* would mean that the user wouldn’t skip the song. If the value of the logit is greater than 0, the logistic function will return a probability greater than 50%; at 0, it will return 50%; and at less than 0, it will return a probability less than 50%.

The process of finding each predictor for the logit is equivalent to the process of finding features in Naïve Bayes. However, determining the coefficients, B, is not as simple as counting occurrences for Naïve Bayes. There are various methods of determining these coefficients. The group already has developed a prototype for a method called **iterative optimization**. This relatively simple prototype is able to determine coefficients which lead to a model with around an 80% accuracy rate. The iterative optimization technique used in the prototype works by taking in two inputs: the first, a file containing hand-marked test cases (see example 3.8 A); and the second, an initial logit (see example 3.8 B). The output will be an improved logit and the accuracy of that logit on the given test cases.

*Example 3.8 A:*

*For a file with N test cases and M predictors each, the file should be in the following format:*

*x0 x1 x2 … xM-1 result0*

*x0 x1 x2 … xM-1 result1*

*…*

*x0 x1 x2 … xM-1 resultN-1*

*Where each predictor will either be present, 1, or not present, 0; and each result will either be case, 1, or not case, 0.*

*Example 3.8 B:*

*An initial logit with M predictors would be in the following format:*

*B1 B2 B3 … BM*

Each coefficient in Example 3.8 B should be either: a 1, if that predictor tends to indicate a *case*; or a -1 if that predictor tends to indicate *not a case*. The intercept, B0, should be omitted; it will be calculated using Equation 3.8 C. One method of determining the initial coefficients is to calculate two weights for the feature using Naïve Bayes and to

*Equation 3.8 C: Optimal Intercept Calculation*

*B0 = Log(1 / (-1 + (1 / CASES)))*

*Where CASES is the percentage of the test cases which are cases (result = 1). This formula was found by setting t in the logistic function equal to B0 and then solving for B0.*

The group’s **iterative** **optimization** method works, essentially, by repeatedly increasing or decreasing each of the coefficients and measuring the accuracy of the resulting logit on the test cases. After adjusting the logit several hundred times, the programs output will be the best performing logit. The downside to this method is that it takes a large amount of processing time to determine an acceptable logit, one whose performance is only about the same as a Naïve Bayes classifier. If this method is to be used on the BeagleBone Black, optimizations must be made reduce the processing time and its accuracy would have to be improved to justify using this method over Naïve Bayes.

**3.9 GPS Research**

Global Positioning System, GPS for short, is a very useful and sophisticated technology. How it works is the device need to be able to connect to multiple GPS satellites. The satellites are all synchronized with time and the locations are measure precisely. The location of the devise is recorded and the satellites use a numbers to give a precise location. These locations are described by using latitude and longitude. For, example the coordinates for UCF are 28.6016° N, 81.2005° W. GPS technology is now in many different devise and has many different uses. They are in phones, watches, apps, social media and so much more. The most practical use of GPS during a person day is weather and directions.

In our project we plan to use GPS. This is an idea that would allow the user to ask what the weather is outside. Then the device would either say or display the weather through a speaker or display. There are some options to do achieve this function. Either buy a GPS module or connect to wifi to find our location. Below are some modules for GPS.

GPS Receiver - EM-506 (48 Channel)

This complete module is built upon the same technology as the ET-301, but includes on-board voltage regulation, LED status indicator, battery backed RAM, and a built-in patch antenna! 6-pin interface cable included.

**Features:**

* 48-Channel Receiver
* Extremely high sensitivity : -163dBm
* 2.5m Positional Accuracy
* Hot Start : 1s
* Warm Start : 35s / 15s with CGEE
* Cold Start : 35s / 15s with CGEE
* 45-55mA at 4.5-6.5V

SparkFun Venus GPS with SMA Connector

This board includes a SMA connector to attach an external antenna, headers for 3.3V serial data, NAV (lock) indication, Pulse-Per-Second output, and external Flash support. We’ve also provided solder jumpers to easily configure the power consumption, boot memory, and backup supply. This board requires a regulated 3.3V supply to operate; at full power the board uses up to 90mA, at reduced power it requires up to 60mA.

**Features:**

* Up to 20Hz update rate
* -148dBm cold start sensitivity
* -165dBm tracking sensitivity
* 29 second cold start TTFF
* 3.5 second TTFF with AGPS
* 1 second hot start
* 2.5m accuracy

GPS Receiver - GP-635T (50 Channel)

The slim design makes it ideal for applications where you don’t have a lot of space to work in. This receiver works with 6-pin 1mm pitch JST type cables and connectors.

**Features:**

* -161dBm tracking sensitivity
* 27 second cold start time
* Slim design
* TTL serial

GPS Receiver - GP-2106 SiRF IV (48 Channel)

The GP-2106 is a small form factor GPS receiver with built-in antenna. Powered by the SiRF Star IV. It also has a hibernate mode which can go as low as 30uA while maintaining a hot start. This module runs at 1.8V VCC, however, you can use 3.3V TTL on the TX and RX pins. Also, the ON/OFF pin needs to be toggled high (to 1.8V) or low in order to turn on the module. The module can acquire satellites as low as -163dBm

**Features:**

* SiRF IV chipset (3db gain over SiRF III)
* 48 channel
* Cold start - 35 second
* Warm start - 35 second­
* Hot start - 1 second
* 1Hz update rate
* 1.8VCC input
* 4800bps (default)

The update rate of a GPS module is basically how often it recalculates and reports its position. The standard for most devices is 1Hz (Only once per second). Keep in mind, though, that a fast update rate means that there are more NMEA sentences flying out of the module, some microprocessors will be quickly overwhelmed trying to parse that much data. On the plus side, if you have a module that runs at 5 or 10Hz, it can usually be configured to run at an easier pace.

The Number of Channels is an important factor. Even though there are only so many GPS satellites in view at any given time, the number of channels that your module runs will affect your time to first fix. Since the module doesn’t know which satellites are in view, the more frequencies that you can check at once, the faster you’ll find a fix. After you get a lock, some modules will shut down the extra blocks of channels to save power. If you don’t mind waiting a little longer for a lock, 12 or 14 channels will work just fine for tracking.

The antenna type means what is the antenna external or the chip. The external type means there will be a need to buy an antenna and connect it to the module. The Chip type means the antenna is part of the chip and no extra antenna is needed to buy. Both antennas would need to face the sky and GPS signals indoors can be hit or miss sometimes. The accuracy of the GPS depends on a few factors. They are the modules, time of day, clarity of reception, weather and location.

The hot start is when the GPS device remembers its last calculated position and the satellites in view, the almanac used (information about all the satellites in the constellation), the UTC Time and makes an attempt to lock onto the same satellites and calculate a new position based upon the previous information. This is the quickest GPS lock but it only works if you are generally in the same location as you were when the GPS was last turned off. The cold start is when the GPS device removes all the information, attempts to locate satellites and then calculates a GPS lock. This takes the longest because there is no known information.

High Sensitivity GPS receiver use hard ware and digital signal processing to search for GPS signals very quickly. This results in very fast times to first fix when the signals are at their normal levels, for example outdoors. When GPS signals are weak, for example indoors, the extra processing power can be used to integrate weak signals to the point where they can be used to provide a position or timing solution. High Sensitivity GPS can provide positioning in many but not all indoor locations. Signals are either heavily attenuated by the building materials or reflected as in multipath. Table 3.9 A illustrated the different types of antenna for GPS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Antenna Type | Dimensions | Start Time | Number of channels |
| GPS Receiver - EM-506 (48 Channel) | Chip | 30x30x10.7mm | Cold: 35 seconds  Hot: 1 second | 48 |
| SparkFun Venus GPS with SMA Connector | External | 32x19x14mm | Cold: 29seconds  Hot: 1 second | 14 |
| GPS Receiver - GP-635T (50 Channel) | Chip | 35x8x6.5mm | Cold: 27 seconds  Hot: 1 second | 50 |
| GPS Receiver - GP-2106 SiRF IV (48 Channel) | Chip | 21x6x6.2mm | Cold: 35 seconds  Hot: 1 second | 48 |

*Table 3.9 A: antenna comparison.*

All of these modules will give an output that can display the longitude and latitude on any simple script.

An alternative to GPS would be to use WPS, Wi-Fi Positioning Service. This method looks up the position of a given Wi-Fi network from a database. This will only work if the location of the network that the device is in has its location listed in a WPS database. It is likely that UCF’s network is available using this service. Skyhook offers an API for querying their WPS server. This method will probably be used over a GPS device to help keep the device low power and light weight; however, if the WPS turns out to be horribly inaccurate, GPS is still an option.

**3.9 Printed Circuit Board**

A printed circuit board (PCB) will be used to connect all of the hardware components of the primary device together; using PCB will help prevent failures at the wire junctions, it will also help prevent short circuits when any wire insulation begins to age, although this will likely not be an issue during our project. On a PCB board solder is the metal that makes the electrical connections between the surface of the PCB and the electronic components.

There are many layers in a PCB. The order goes silkscreen, solder mask, copper and then substrate (FR4). Some structure names are annular ring, which is the ring of copper around a plated through hole in a PCB. DRC is the design rule check. Drill hits are the places on a design where a hole should be drilled. A finger is an exposed metal pad along the edge of the board. Mouse bits are spots that create weak spots on board so that they can be broken. The pad is the portion of exposed metal on the surface of the board. The panel is when many smaller boards compose a larger circuit. Slots are any holes in a board which is not round.

Soldering is a skill will be necessary in order to properly attach the hardware components to the board; however the group does not have any experience with soldering. Options for doing this to either learn to do soldering or get a club at UCF to do soldering for nominal fee.

Determining the layer count is an important part of a PCB design. There are many potential layers in a PCB but the more layers the more expensive. The PCB.Multi-layer printed circuit boards have trace layers inside the board. Deciding layer count depends on design complexity. Going with a single or double layer would be the best decision. The next step is to figure out how big the PCB is going to be. There is a link on the senior design website that leads students on PCB design. One of the websites called OSH Park, an electric ecosystem, sells 2 layer boards for $5 per square inch (with 3 copies of your board included in that price) and ship in under 12 calendar days from ordering. Also 4 layer boards for $10 per square inch. This is the best and most efficient deal.

The group will use Eagle to design the PCB. This would be a helpful program to catch errors before the board is fabricated. Aaron Ford will be in charge of designing, and if need be, soldering components to, the group’s PCB board.

1. **Design**

Helium will be split into two devices; the primary device, which plays the music, among other tasks; and the secondary device, which scans for RFID tags. The block diagram in Figure 4.0 A, below, shows what we think a modular view of our project would look like. The group decided to separate project into two separate devices for two reasons.

1. An early idea for the person detection task using RFID was to put the scanner near the door so that it wouldn’t have to scan too far to detect when someone enters. Although we don’t believe that we will be able to create a scanner capable of scanning the area of an entire doorway, the group decided that the RFID scanner should be placed in a convenient place for users to swipe their tags near. Since the place that is convenient for the scanner might not be convenient for playing music, the group decided to split the systems for the greatest flexibility.
2. One of the group’s goals for this project is to create a “high fidelity” music player. Thermal noise from electronic equipment will reduce the signal-to-noise ratio of the audio output from Helium. We believe, that by separating the two devices, that we will be able to achieve a higher quality audio output.

In Figure 4.0 A, Bluetooth communications are only shown between the primary and the secondary devices; however, there might be as many as 5 additional mobile devices communicating with the primary device, which will act as the master device for the Bluetooth communications.

The group has not yet determined the max voltage necessary for the second AMP to produce. Ideally, this AMP will only be able to produce a max voltage that is well below the level of loudness associated with hearing damage. However, this will depend on the speakers attached. If time permits, the group may add a module after the splitter that senses the impedance of any speakers and limits the output from the AMP to a safe level.

The group believes that this project might not have the correct proportion of hardware to software for the make-up of our group, 3 EEs and 1 CpE. Additional hardware modules will probably have to be added to this device.

AMP

MIC

SS

Power

System

Voice Command

Module

mp3

decoder

Equalizer

MCU

To Internet

AMP

Splitter

Bluetooth

Wi-Fi

To Speakers

Primary Device

Secondary Device

Bluetooth

RFID

SCANNER

Power

System

MCU

Digital

Analog

Power

*Figure 4.0 A: Block diagram of Helium*

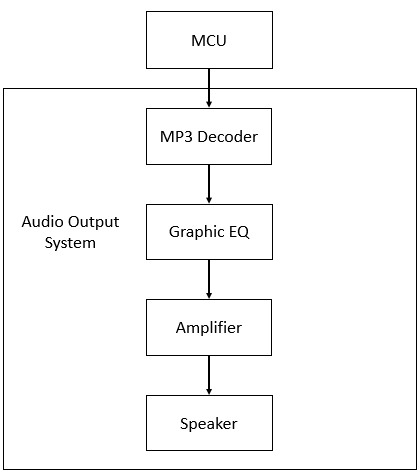
**4.1 Hardware Design**

**4.1.1 Communications System**

Communications must occur between: the primary and secondary devices; the primary device and multiple mobile devices; and the primary device and the media server. Since the first two types of communication will involve nearby devices, Bluetooth will be used. Since the primary device and the media server might be far from each other, Wi-Fi will be used. Most of the communications system will involve software, this is detailed further in section 4.2.6.

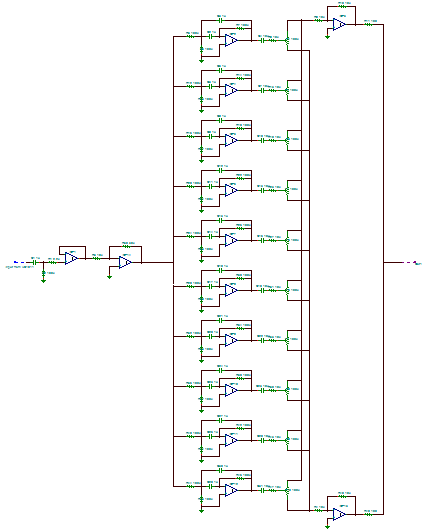
**4.1.2 Audio Output System**

The audio output system consists of the MP3 decoder, graphic equalizer, amplifier, and speaker. The layout of the block diagram with the flow of the signal is shown below.



*Figure 4.1.7 A: Audio Output System Block Diagram*

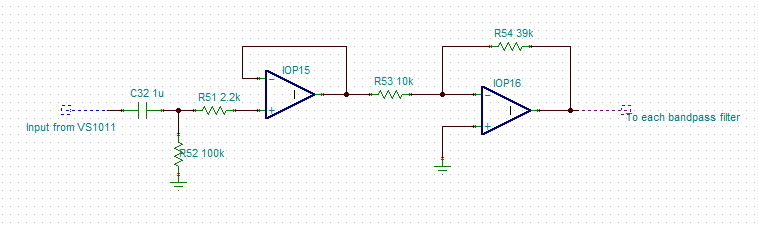
An MP3 file is selected to play from the MCU. The signal is then sent from the Beagle board into the audio output system. The MP3 file is sent through the decoder to create a digital signal. The decoder contains a digital to analog converter that will create an analog signal that is able to be transmitted through the speaker and heard. Once the analog signal is sent from the decoder, it will enter the graphic equalizer circuit. The graphic equalizer circuit will allow modification of the signal to change the sound heard from the speakers. After the signal is manipulated, it will be amplified to provide enough power for the speaker. The design of each component of the audio output system will be outlined in the following sections. The final design is shown in the figure below.



*Figure 4.1.7 B: Final Graphic Equalizer Design*

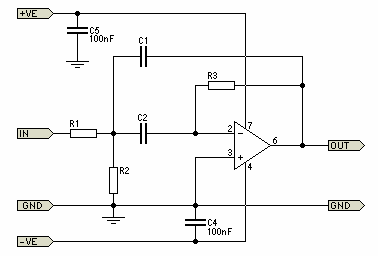
It’s impossible to see individual components in this view, so the design is broken up into components for the purpose of outlining the design. First, the input from the decoder feeding into the 10 filters; second, the 10 filters, which feed into the third and fourth parts; the cut or boost stages, which then feeds into the output. Each of the filters has a separate set of components, so each filter will need to be shown separate.

Beginning with the input stage, the output from the VS1011 Decoder feeds into the equalizer circuit, is amplified, and inverted to fix the phase shift created by using the Multiple Feedback Bandpass filters.



*Figure 4.1.7 C: Input Stage of the Equalizer*

Next, the input stage feeds into all 10 bandpass filters. Each filter is a reiteration of a basic Multiple Feedback Bandpass Filter, but tailored to each center frequency. With this type of filter, there are a handful of important values that determine the functionality of the band. Each filter has an input resistance, attenuator resistance, feedback resistance, center frequency, bandpass gain, Q-factor, and capacitance. A reference design of a basic Multiple Feedback Bandpass filter is shown below.



*Figure 4.1.7 D: Basic Multiple Feedback Bandpass Filter*

The filter design uses two capacitors, each of which will be the same value, and selected based on the center frequency of the given filter. The gain is ideally unity gain, so that value will be set at 1. For each filter, the frequency, gain, Q-factor, and capacitance will be set, and the resistor values are then calculated. The equations for each resistor value are as follows. The input resistance, referred to as R1 in the reference design, is given by the equation:

R1 =

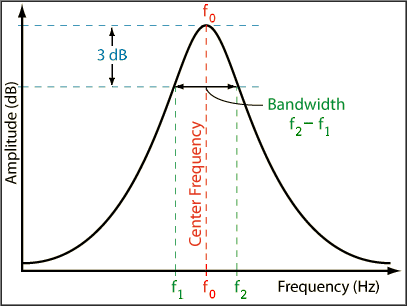
The attenuator resistance, referred to as R2 in the reference design, is given by the equation:

R2 =

The feedback resistance, referred to as R3 in the reference design, is given by the equation:

R3 =

For each of the equations, G is the bandpass gain, which has a value of 1. C is the capacitance represented by C1 and C2 in the reference design, which are set to the same value. Q is the Q-factor, and f is the center frequency. Each of these three equations is applied to every band of the equalizer to determine the components used to form each filter. The values for the center frequencies are industry standard for an equalizer with 10 bands. When determining which center frequencies to use, the idea is to cover the range from 20 Hz to 20 kHz without having any gaps in between the range of each band. With 30 band equalizers, the bandwidth of each filter can be smaller, which allows for 1/3 octave bandwidths. For this design, 10 band equalizers need to space each band evenly to be able to cover the range. The bandwidth of each filter is supposed to allow only a certain range of frequencies through while at full boost centered at a frequency. The figure below shows that bandwidth as the range that pass after a 3dB drop in amplitude.



*Figure 4.1.7 E: Bandwidth At Full Boost*

The relationship between center frequency, bandwidth, and Q is shown in the equation:

= ,

where is the center frequency, as shown in the figure above. By keeping a constant Q factor, and selecting the center frequency, the bandwidth can be solved for. However, the actual values of and are unknown. The values can be derived by observing the fact that the bandwidth is determined by the range formed after a 3 dB drop from maximum boost at the center frequency. The equations for and are as follows: the lower cutoff frequency, , is

and the upper cutoff frequency, , is

There is also a relationship between Q and bandwidth in octaves for each band on the equalizer. To evenly space each band on a 10 band equalizer, the bandwidths will be one octave apart, which means each center frequency doubles. This is ideal for the audible range because it is essentially 0-20 kHz, which means if you have 10 bands doubles each time, then that will cover all frequencies. Knowing that each band will be spaced one octave apart, the relationship between Q and octaves determines the Q-factor that is held constant to form the Constant-Q equalizer. The equation relating the two is:

,

where N is the bandwidth in octaves. Since N has been determined to be 1, the equation yields Q = 1.41421. The industry standard values are 31, 63, 125, 250, 500, 1k, 2k, 4k, 8k, and 16 kHz. Each center frequency is doubled because of the octave and Q relationship. Using the equations presented in the paragraph above, these values can be proven to cover every frequency possible in the human audible range. The lower and upper cutoff frequency equations when used with each center frequency provide the results shown in the table below, which displays the overlap between each band to ensure no frequency gaps.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 31 Hz | 22 Hz | 44 Hz |
| 63 Hz | 44 Hz | 89 Hz |
| 125 Hz | 88 Hz | 177 Hz |
| 250 Hz | 177 Hz | 354 Hz |
| 500 Hz | 354 Hz | 707 Hz |
| 1 kHz | 707 Hz | 1.414 kHz |
| 2 kHz | 1.414 kHz | 2.828 kHz |
| 4 kHz | 2.828 kHz | 5.657 kHz |
| 8 kHz | 5.657 kHz | 11.314 kHz |
| 16 kHz | 11.314 kHz | 22.627 kHz |

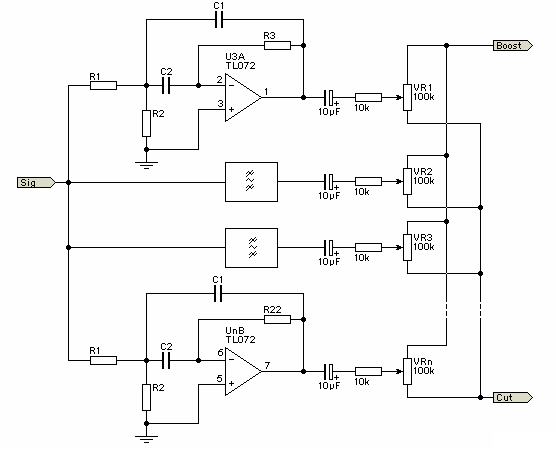
*Table 4.1 A: Calculated Bandwidth Values*

As seen in the table, the 10 center frequencies chosen cover from 22 Hz to 22 kHz, which is perfectly suitable. The 20-22 Hz range is barely audible, so isn’t as important to cover, and the max frequency of 22 kHz covers the 20 kHz band, with extra coverage not even necessary. At this point, the bandwidth in octaves, center frequencies, bandpass gain, and Q-factor have been determined. This leaves the capacitance values, which can be estimated using the following table that provides the ideal capacitor values based on certain frequency ranges.

|  |  |  |
| --- | --- | --- |
| f(min) | f(max) | Capacitance |
| 20 Hz | 80 Hz | 330 nF |
| 80 Hz | 300 Hz | 82 nF |
| 300 Hz | 1.2 kHz | 22 nF |
| 1.2 kHz | 4.8 kHz | 5.6 nF |
| 4.8 kHz | 20 kHz | 1.5 nF |

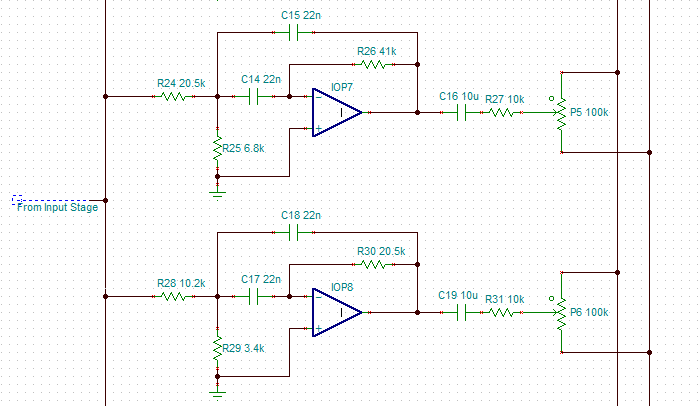
*Table 4.1 B: Ideal Capacitor Values Based on Frequency*

Now, the resistor values can be calculated for each filter in the equalizer circuit. After the resistance values for each filter are calculated, the filters are designed in parallel and the same basic design is reiterated as shown in the reference design below.



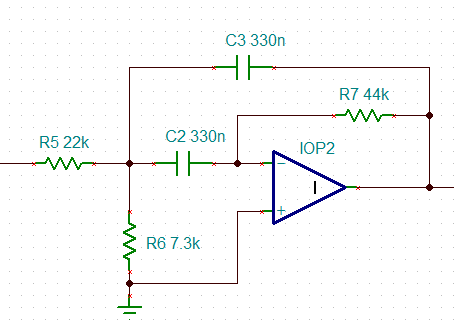
*Figure 4.1 F: Multiple Band Filter Reference Design*

Using this reference, the final design was made, and a section of the circuit design showing two of the filters is shown below.



*Figure 4.1 G: 2 of the 10 Band-pass Filters*

For the sake of being able to actually see the setup of each filter, only two are displayed to show that each band is reiterated, but with specific values for C, R1, R2, and R3. Each filter, along with the resistor calculations are as displayed as follows:

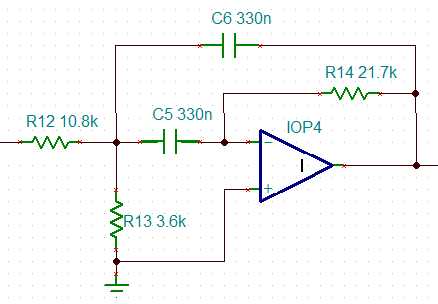


*Figure 4.1 H: 31 Hz Center Frequency Filter*

R1 = = = 22.097 kΩ

R2 = = = 7.334 kΩ

R3 = = 44.004 kΩ

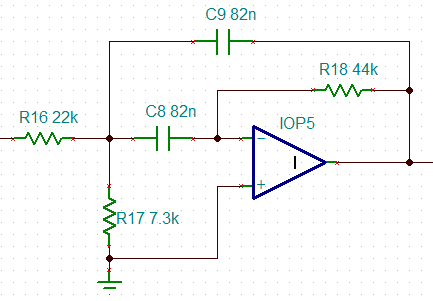


*Figure 4.1 I: 63 Hz Center Frequency Filter*

R1 = = = 10.826 kΩ

R2 = = = 3.608 kΩ

R3 = = 21.652 kΩ

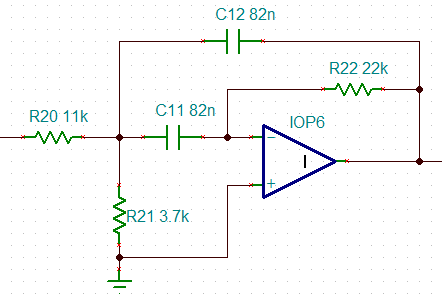


*Figure 4.1 J: 125 Hz Center Frequency Filter*

R1 = = = 21.958 kΩ

R2 = = = 7.319 kΩ

R3 = = 43.917 kΩ

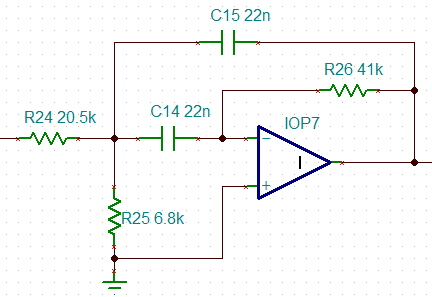


*Figure 4.1.7 K: 250 Hz Center Frequency Filter*

R1 = = = 10.979 kΩ

R2 = = = 3.659 kΩ

R3 = = 21.958 kΩ

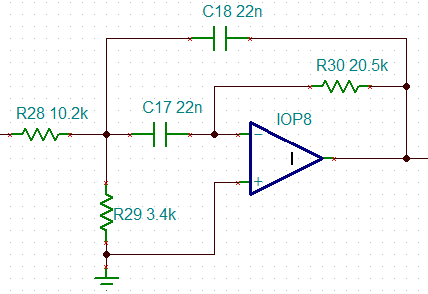


*Figure 4.1.7 L: 500 Hz Center Frequency Filter*

R1 = = = 20.461 kΩ

R2 = = = 6.82 kΩ

R3 = = 40.923 kΩ

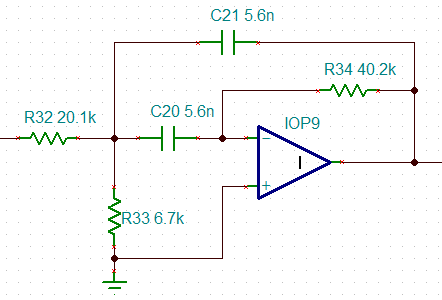


*Figure 4.1.7 M: 1 kHz Center Frequency Filter*

R1 = = = 10.230 kΩ

R2 = = = 3.410 kΩ

R3 = = 20.461 kΩ

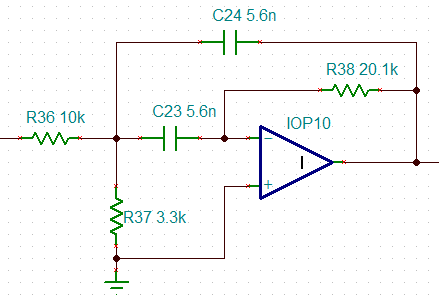


*Figure 4.1.7 N: 2 kHz Center Frequency Filter*

R1 = = = 20.096 kΩ

R2 = = = 6.698 kΩ

R3 = = 40.192 kΩ



*Figure 4.1 O: 4 kHz Center Frequency Filter*

R1 = = = 10.048 kΩ

R2 = = = 3.349 kΩ

R3 = = 20.096 kΩ

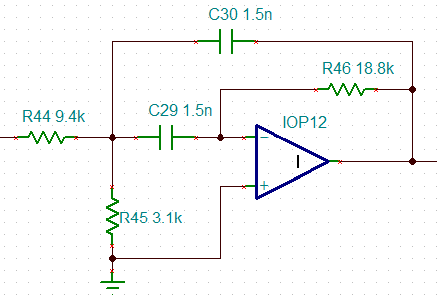


*Figure 4.1 P: 8 kHz Center Frequency Filter*

R1 = = = 18.756 kΩ

R2 = = = 6.252 kΩ

R3 = = 37.513 kΩ



*Figure 4.1 Q: 16 kHz Center Frequency Filter*

R1 = = = 9.378 kΩ

R2 = = = 3.126 kΩ

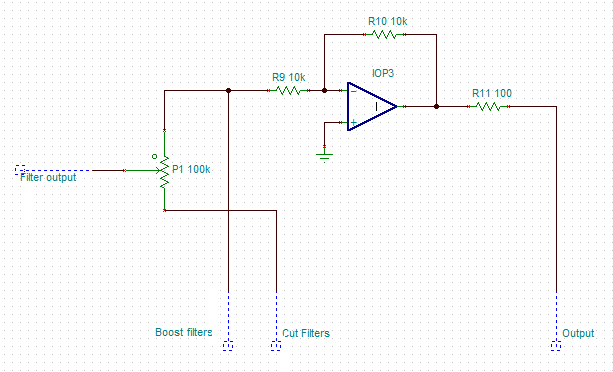
R3 = = 18.756 kΩ

An overall summary of the components used in each filter can be seen in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | C | R1 | R2 | R3 |
| 31 Hz | 330 nF | 22 kΩ | 7.3 kΩ | 44 kΩ |
| 63 Hz | 330 nF | 10.8 kΩ | 3.6 kΩ | 21.7 kΩ |
| 125 Hz | 82 nF | 22 kΩ | 7.3 kΩ | 44 kΩ |
| 250 Hz | 82 nF | 11 kΩ | 3.7 kΩ | 22 kΩ |
| 500 Hz | 22 nF | 20.5 kΩ | 6.8 kΩ | 41 kΩ |
| 1 kHz | 22 nF | 10.2 kΩ | 3.4 kΩ | 20.5 kΩ |
| 2 kHz | 5.6 nF | 20.1 kΩ | 6.7 kΩ | 40.2 kΩ |
| 4 kHz | 5.6 nF | 10 kΩ | 3.3 kΩ | 20.1 kΩ |
| 8 kHz | 1.5 nF | 18.8 kΩ | 6.3 kΩ | 37.5 kΩ |
| 16 kHz | 1.5 nF | 9.4 kΩ | 3.1 kΩ | 18.8 kΩ |

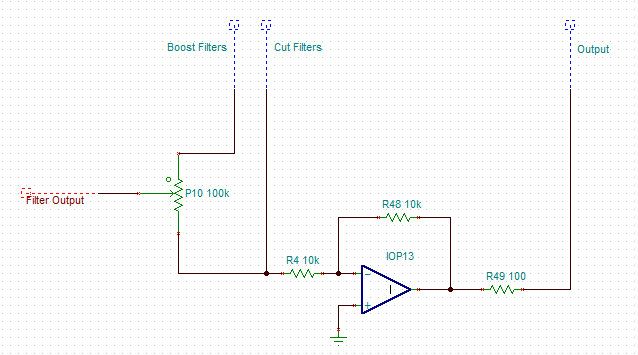
*Table 4.1 C: Summary of Filter Circuit Elements*

Now that the 10 bands have been designed, the last stages are the boost and cut designs. The Boost Stage is shown below.



*Figure 4.1 R: Frequency Response Boost Stage*

The output of each filter ultimately feeds into a 100 kΩ potentiometer. If the potentiometer is shifted towards the boost stage, then the boost circuit is activated, and positive gain is applied to the center frequency. The same applies, but vice versa for the cut stage, and a negative gain is applied to the center frequency. The cut stage is pictured below.



*Figure 4.1.7 S: Frequency Response Cut Stage*

Both the boost and cut stage feed into the output and the circuit is completed. The signal is now fed into an amplifier and turned into the music we hear through the speaker.

**4.1.3 RFID System**

According to our research, and the different choices between RFID base station and transponders, it does not looks like there is a way to can create an RFID reader that would be able to cover the door of a room. The best that can be done is have the different users at the least move their tags, or cell phones, in front of the RFID reader so Helium can take their information and starts playing songs. The reason is that to implement a longer range scanner would require UHF reader and tags which would be cost prohibited when there other parts that can be improved for more efficiency. Figure 4.11 illustrates the block diagram of the RFID system.

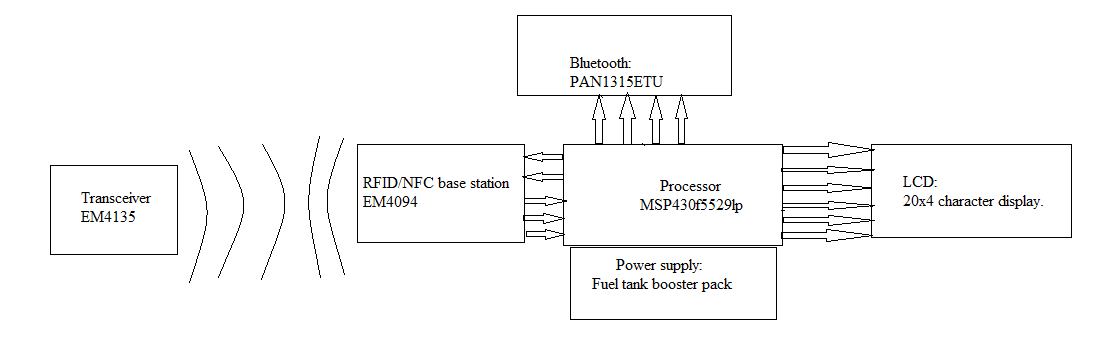


Figure 4.11: RFID system block diagram

**Transceiver**

Looking at the previous list choices for the base of the RFID Reader, the group decided to go with the EM4094. The EM4094 is an analog integrated transceiver produced by EM electronics to read/write information to compatible transponders. By adding a few extra components, the EM4094 can take the frequency of 13.56 MHz which is required to read/write information for the tags. When reading the transponder, the EM4094 requires a digital processor to take the information and convert it to readable data for Helium to be able to understand. When not in used the RFID can be put to a power down state to save power. Figure 4.12 shows the pin layout of the EM4094.

Permission from EM microelectronic pending

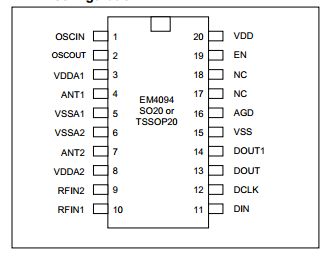


Figure 4.12: pin layout of the EM4094

The EM4094 will be used as a way for Helium to be able to read the information from the tags that will turn it on and start playing a song. The EM4094 will demodulate the data from the transponder for the digital processor so Helium can play the right genre of the song. Since preferences can change, the EM4094 can later write new information to the transponder for later use for the user. Another important aspect of the EM4094 is its ability to use NFC to communicate with cellphones that have the chips installed, saving the cost in bothering of getting the tags. When the same tag is used again it will tell Helium that the person has left the room and if no one else is inside the room, Helium will go to sleep mode. Another reason the EM4094 analog transponder will be used is that it can be set to go to sleep mode to save power if there is no tags present. Using Figure 4.13, as reference the group will be able to design the base station to connect to the digital processor with no problem, looks like it needs at the least five pins for I/O. Figure 4.14 is the circuit that will be implemented with the five open pins being the one belonging to the MCU.

Permission from EM microelectronic pending

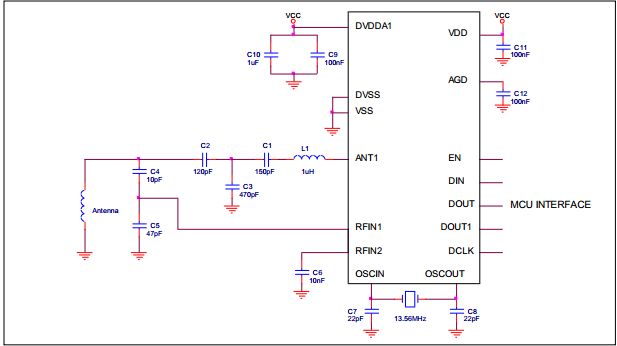


Figure 4.13: reference for the design of the RFID reader.



Figure 4.14: The circuit that will be built, VDD (left most) is just there to signify antenna.

To really understand how the EM4094 works, is best to look at the block diagram. Figure 4.15 shows the block diagram of the EM4094. The block diagram shows an insight on how the reader works, the antenna connects to one of the ant pins and a RFIN pin, the signal from the antenna is then demodulated and goes to a comparator before going to the I/O multiplex before going to the microcontroller. Now, if a signal is to be sent to the transponder, The MCU sends out the signal to the Din section of the EM4094 in which the multiplex sends it to the modulator, a quartz of 13.56 MHz provides the oscillator with the carrier frequency to modulate the signal in which the it goes to the transmitter antenna to be sent to the transponder.

Pending permission from EM microelectronic.

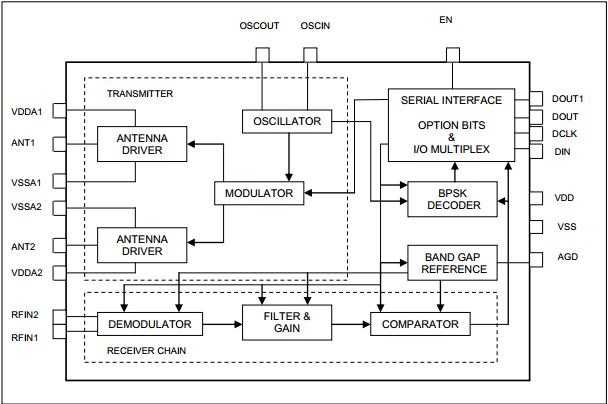


Figure 4.15: Block diagram of EM4094

**Transponder**

The EM4135 CMOS IC manufactured by EM electronics will be used to work this project. The EM4135 have 2.4k bit of memory organized in 38 words of 64 bits each, plenty of memory for what the group wishes to do. It’s compatible with the EM4095 base station for the reader, but because it is a passive tag the range of the RFID will be limited to the centimeters at the best case. Figure 4.16 shows the block diagram of the EM4135, this will help in seeing how it works before and after a signal appears.

Pending permission from EM microelectronic.

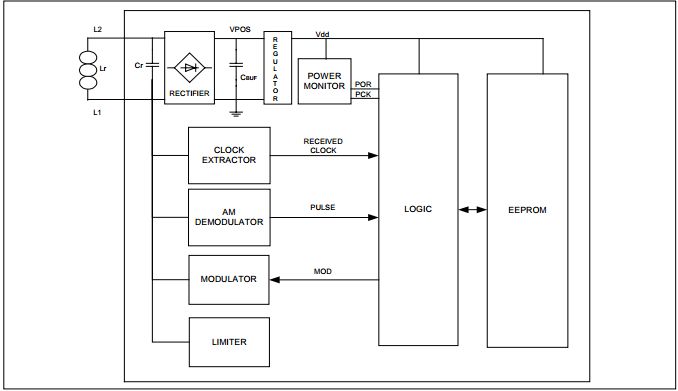


Figure 4.16: EM4135 Block diagram.

As shown on the block diagram, the transponder receives a signal through the antenna before flowing to a rectifier to be converted to dc voltage for the logic gate and EEPROM. The signal is demodulated to a pulse then based on the demodulated signal, the logic section will send the memory after modulating it to the antenna with the output signal caught by the RFID reader.

The group chose this transponder for our project because it is compatible with the EM4094 Transceiver that was chosen to be the base of the RFID reader. With the amount of memory it can store, the group will be able to program the transponder to be password protected along with adding some information of the user to identify which song is most popular so that Helium can predict which song to play when someone enters the room and starts playing songs.

**Digital signal processor**

The MSP430 is a low power consumption MCU that can be easily programed through code composer, or energia for simpler programming. By using the I/O the MSP430 can connect the RFID base station, and adding another piece of hardware, the MSP430 can transfer data wirelessly to another source, allowing the RFID reader to be mobile to a certain range.

The reason for the MSP430f5529 will be used for Helium, is a low power and the low cost of $5.39. The MSP430 can be easily programmed through said compilers above, and then be left at its own device without being connected to a laptop or computer for power. The power supply to power the MSP430 and all the other components can be supply through a battery booster pack to do all its calculation. This allows for greater mobility as the RFID reader can be moved around with ease.

**Bluetooth**

The PAN1315ETU-ND manufactured by Panasonic will be used to add Bluetooth capabilities to the MSP430. The PAN1325ETU is a low power consumption (1.7V-4.8V) module for the MSP430 that adds pins to the module for easier connection between the two devices. Figure 4.17 illustrates the PAN1315ETU block diagram.

Reason for this piece of equipment to be used instead of the cheaper HC-05 module, is that it was created to be compatible with an MSP430. This will save time in configuring the programming side between the MSP430 and the main body of Helium instead of trying to work. This module also come with its own pins so it can be connected to the MSP430 using an F/F wire for simple interface.

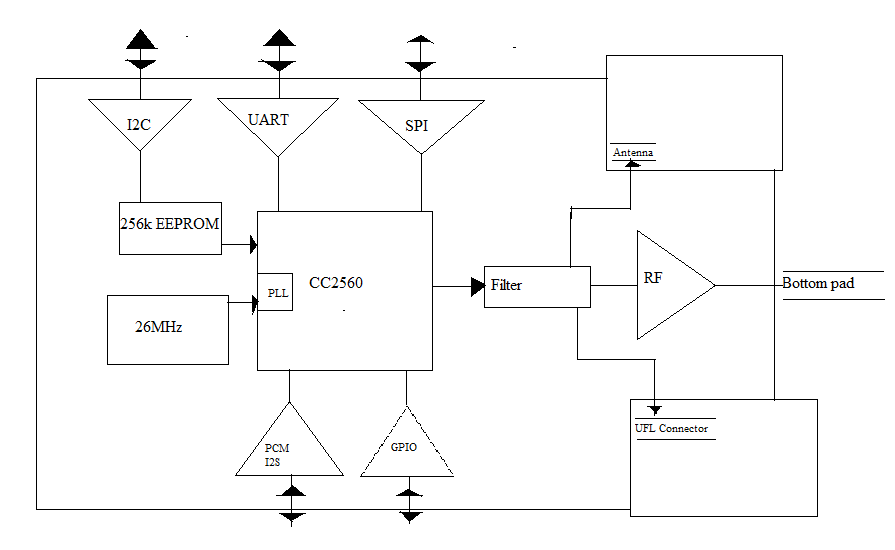


Figure 4.17: PAN1315ETU block diagram

**Fuel Tank Booster Pack**

As stated above, the MSP430 will required its own separate power supply if the RFID reader would become mobile. Texas instruments sell a rechargeable fuel tank for the MSP430 for this case. The battery itself comes with some extra perks of its own that can help keep maintenance of the RFID system. This includes: a screen that tells the state of the battery (such as voltage, temperature, and so on).

**LCD MODULE**

The RioRand™ LCD Module is a LCD display for the Arduino board. With the 20x4 line it should be enough space to write out the state of the RFID reader. This board was chosen because the display is done with white letter on a blue background make things easier to read what is on display. Another reason is that the energia compiler that will be used can take in Arduino libraries to translate the results to work with the MSP430. By using the libraries, and modifying some of the syntax, the LCD display can be set up within days instead of taking the time to create a new set of libraries to make sure the LCD works with the MSP430. Figure 4.18 illustrates the design for the LCD to connect to the MSP430, the MSP430p325 is used to illustrate the I/O to connect to the LCD while the power source and ground are there since the Multism have the source and ground pins hidden.

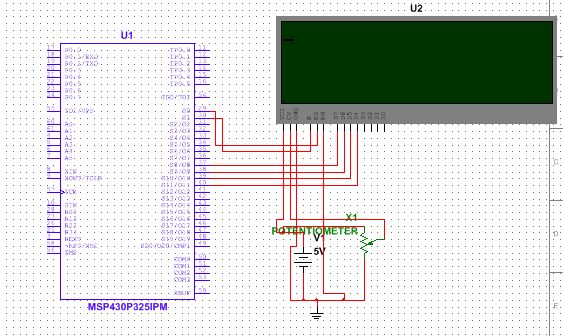


Figure 4.18: Design to connect the MSP430 to the LCD module.

**4.2 Software Design**

**4.2.1 Overview of Software**

Helium’s **hardware** is broken into two sub-systems: the first, controlled by the BeagleBone Black, will be responsible for the majority of the system’s processing; the second, controlled by the MSP 430, will only be responsible for processing the RFID tags and sending the results to the primary device. Accordingly, Helium’s **software** will also be broken into two sub-systems.

**Primary Device**

The BeagleBone Black used in the primary device will be running Ubuntu 14.04.1; in addition, the device will also run JDK 7 for ARM. The majority of the group’s programming experience is in both Java and Python. To take advantage of the group’s strengths, the majority of the software on the primary device will be written in Java – with some modules written in C. Java was chosen over Python because of its superior speed (the median results for one set of benchmarking tests indicates that Java is 27 times faster than Python 3).

The class diagram, in Figure 4.2 A, shows an overview of the classes which will be running on the device. The program’s main loop will be ran from the Init class. Init will be responsible for coordinating data flow between the device’s three main control modules: DeviceController, MediaController, and Communication; each of which will be ran in their own thread. Init will also be responsible for carrying out the various tasks of the UserManagement module.

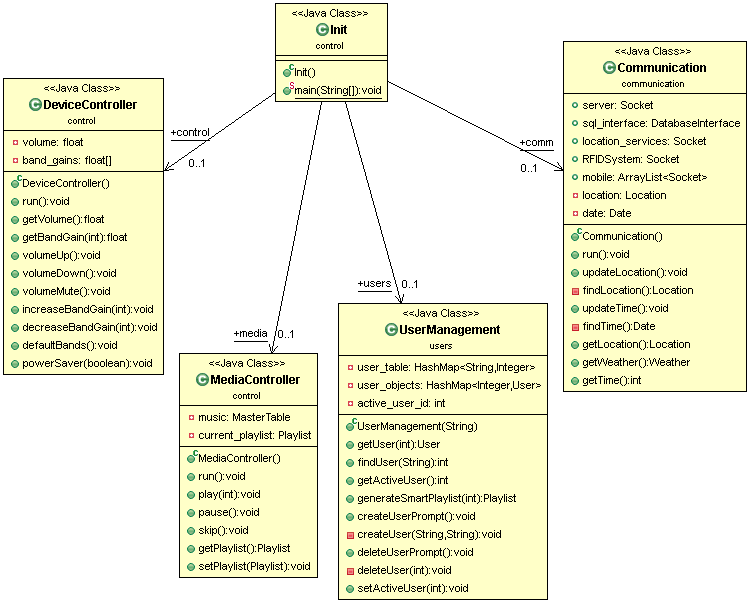


Figure 4.2 : A UML class diagram showing an overview of the primary device's software.

The **DeviceController** module will be responsible for controlling the various hardware components of the primary device. This will include controlling the overall volume, the gain of each band in the equalizer, and cutting off and routing the power to the various hardware components when turning on and off the power saving mode and/or device. The volume and gain will be adjusted by controlling digital potentiometers; power routing will be controlled using digital switches. The volume and gain for each of the bands will be stored as floats between 0.00 and 1.00, representing the percentage of maximum gain.

The **MediaController** module will be responsible for controlling the playback of songs as well as song storage (see section 4.2.2) and song selection (see section 4.2.3). Playback methods will include:

* play(int song\_id): This method will begin decoding the song, with the given song\_id, from the beginning.
* pause(): This method will toggle the decoding process on and off.
* skip(): This method will end the decoding process of the current song and execute play(current\_playlist.nextSong()).

The **Communication** module will be responsible for communication with the secondary device, servers on the internet, and possibly numerous mobile devices. Figure 4.2 B shows the Communication module in more detail. The Communication module will also have Location and Date objects for keeping track of weather and time of day. Brief details are provided in the Table 4.2 A, below; for more detailed information on the messages sent and packet formats, refer to section 4.2.6.

|  |  |  |
| --- | --- | --- |
| **Communication with** | **Details** | **Technology** |
| Secondary Device | The secondary device will send a user ID, corresponding to the user associated with the detected RFID tag. The primary device will send a small packet for updating user ID’s when they are changed by the user. | Bluetooth |
| WPS Location Server | Using an API provided by Skyhook, the device will attempt to determine its location. | Wi-Fi |
| Weather Server | Using an API provided by World Weather Online, the device will attempt to determine the weather conditions for its current location. | Wi-Fi |
| Mobile Device(s) | Mobile devices will send control commands to the primary device (see section 4.2.9). | Bluetooth |

*Table 4.2 A: Overview of communications.*

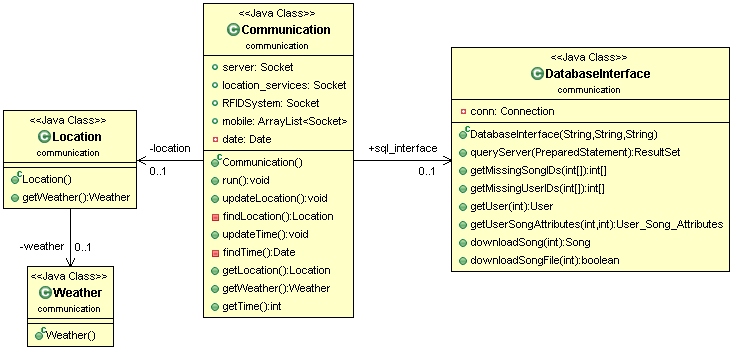


Figure 4.2 : A UML class diagram showing the Communication module in more detail.

The **UserManagement** module will be responsible for storing user data, as well as creating new users and deleting users. The UserManagement module will store **User** objects in a Hash Map, with the user’s ID as the key. There is another Hash Map, user\_table, which maps user names to user IDs. The User module is shown in more detail in figure 4.2 C. The playlists for each user will also be stored, as objects, in each User object. The methods for song selection (see section 4.2.3) will be contained within the **Playlist** module.

The bulk of the user data stored in each User will be in the form of **User\_Song\_Attributes** objects (see section 4.2.2: Efficient Storage of Metadata). There will be one User\_Song\_Attributes object for each song on the device (and possibly, for each song on the server); these objects will be stored in a Hash Map, with the ID of the song, that each object corresponds to, as the key.

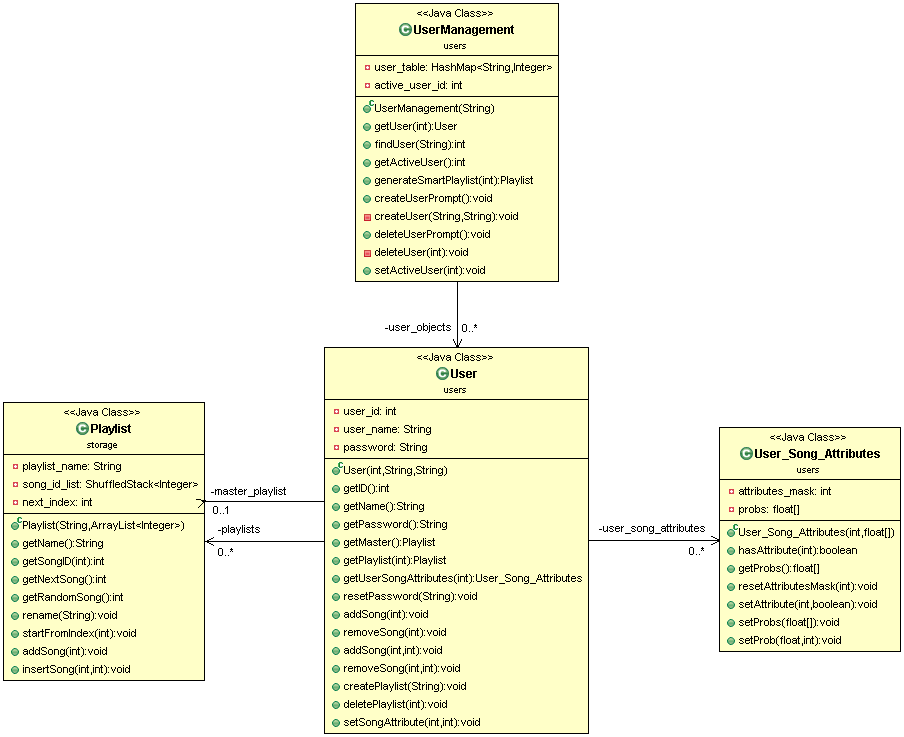


Figure 4.2 : A UML class diagram showing the UserManagement module in more detail.

For a detailed look at all of the classes and method stubs that make up the software on the primary device, refer to section 5.4.

**Secondary Device**

The secondary device lacks the processing power and RAM necessary to run an operating system or an embedded JDK; accordingly, all software on this device will be written in C. The device’s software will consist of the following structures and methods:

*struct user\_table{*

*int num\_users;*

*char\* user\_ids; // Each user is identified by a 8 bit character*

*int\*\* user\_map; // Maps an array of RFID tags to a particular user ID*

*};*

When communicating with the primary device, the two devices will send packets to one another in the following format. Each packet will be 6 bytes long; 1 byte for the command and 5 bytes for the payload. For the payload, the first byte will be a user ID and the last four will be for an RFID tag number. The payload may be empty. Each device may only have one outstanding packet at a time. For a list of commands, refer to section 4.2.6.

*struct packet{*

*char cmd;*

*char payload[5];*

*};*

*char determineUserID(int RFIDtag); // Converts an RFID tag number to a user ID*

*void addUser(int user\_id, int RFIDtag);*

*void addRFIDtag(int user\_id, int RFIDtag);*

*void deleteUser(int user\_id);*

*void deleteRFIDtag(int user\_id, int RFIDtag);*

*struct packet recvFromPrimary(); // Checks for any packets sent from the*

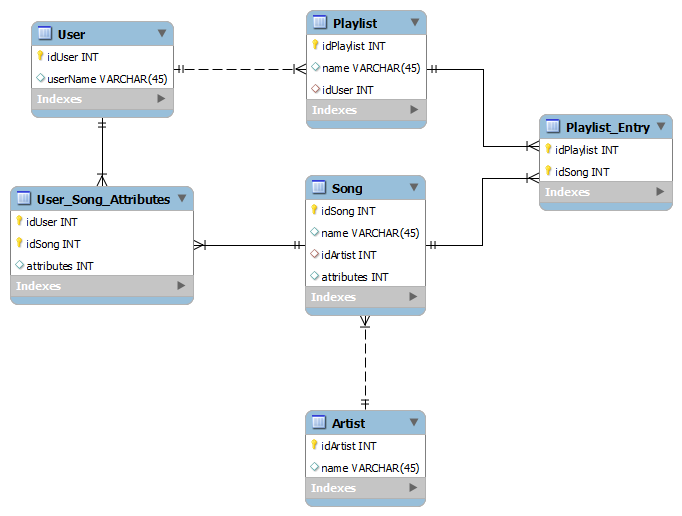
*// primary device*

*void toMainDevice(struct packet pkt); // Sends, via Bluetooth, the given packet to the // main device*

**4.2.2 Song Storage**

Song storage will be divided into 2 distinct modules. The 1st module will be ran on the device and use the device’s available flash memory to store songs and the metadata associated with them. It will only store a subset of the audio data to be stored, as well as the entire set of user data. The 2nd module will be a MySQL database (see figure 4.2 D) ran on a server external to the device (most likely, a laptop). It will store the entire set of song data (mp3 files and metadata), as well as the entire set of user data.

The device should be able to operate without a connection to the external server and/or a mobile device, which may or may not have a connection to the server. The device will attempt to reestablish connection with the server whenever Wi-Fi is available, and when not – and as an optional feature – will attempt to forward updates through a connected mobile device to the server. The minimum, and most simplistic, method for solving any consistency issues that may arise will be to treat the data on the device as authoritative when resolving conflicts. Optional solutions to consistency issues might involve time stamping, as an example.



*Figure 4.2 D: User and Song/Artist metadata (mp3 files not included in the EER Diagram)*

**Efficient Storage of Metadata**

The device (and/or server) will have to store a set of attributes for each user for each song (see figure 4.2 D: User\_Song\_Attributes and Song). This data is required for creating playlists (see section 4.2.3) and recommending new songs (see section 4.2.4). These attributes can be divided into 2 sets. The 1st set of attributes (see Figure 4.2 D: Song.attributes) are the properties intrinsic to the song, such as genre, key, instruments used, etc., these attributes only need to be stored once per song (they will not vary with user); the 2nd set of attributes (see figure 4.2 D: User\_Song\_Attributes.attributes) are the properties of each song specific to each user. These will indicate when the user tends to prefer listening to this song; properties might include time of day, weather conditions, and/or how frequently the user likes to listen to this song.

To keep the storage space needed to store these attributes to a minimum, attributes encoded using the fewest possible bits and will be stored in 32 bit integers with other attributes (see Example 4.2 A). In Figure 4.2 D, only 1 INT is displayed per Song and per User\_Song\_Attributes; more integers may be added if needed. For most attributes, a single bit will indicate a single attribute, but in some cases this might not be possible.

*Example 4.2 A:*

*There are 2 possible attributes (denoted by underscore):*

*Feature Acoustic Guitars \_ x x x x*

*Genre, with 16 possible values x\_ \_ \_ \_*

*A song has the following attributes:*

*Feature’s acoustic guitars and its genre is “rock” (denoted by x 0 0 1 1)*

*Its 5 bit value would be 1 0 0 1 1 == 19*

Alternatively, a single bit can be used for each of the possible 16 genres in Example 4.2 A, although this would require more space, it would make storing multiple genres possible; also, in order to use binomial logistic regression or Naïve Bayes for song selection, each attribute would have to be encoded as a binary value, since there are only two possible classes.

**Playlists**

Users will be able to create and edit their playlists which will be stored on both the device and on the server. These playlists will contain references to songs and their metadata. Since the device will contain songs from multiple users, a *Master Playlist* will be kept for each user. This list will be a quick reference of the songs that any particular user likes.

The *Smart Playlists* generated by the selection algorithm (see section 4.2.3) will NOT be able to be edited by the user nor viewed. The reason for this is that the songs contained within them are predictions of the next several songs that the user would probably most enjoy listening to; if a user dislikes one of the predictions, then the entire list might be thrown out and recalculated.

Since the same song may be stored in many different playlists, only the song IDs will be stored in a playlist; this will save space. In particular, the song IDs will be stored in a custom container, *ShuffledStack* (see figure 4.2 E). When a playlist is played in order, the *ShuffledStack* will behave just like a queue (an ArrayList). However, when *shuffle* mode is turned on, the *ShuffledStack* will be able to, ideally, return random elements within itself without repeating any objects twice in a row. For N songs, each song will have a 1/N chance of being selected. Complicating this process, a *Playlist* will be able to contain multiple reference to the same song; meaning that a song that appears twice in a playlist should have a 1/2N chance of being selected. In addition to not repeating the same object twice in a row, for an object that only appears **once** in the *ShuffledStack*, that object should NOT be repeated until all other N-1 objects have been returned; for an object that appears twice, that object should NOT be repeated more than twice until all other N-2 objects have been returned; and so on and so forth. Ideally, the *ShuffledStack* object will be able to achieve this behavior within a constant retrieval time for objects; this however, might not be practical.

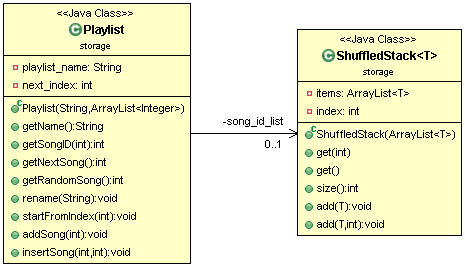
****

Figure 4.2 : A UML class diagram showing the Playlist and the ShuffledStack classes.

**4.2.3 Song Selection Algorithm**

The ideal song selection algorithm (for a single listener) could be stated simply as follows: *given a set of N possible songs to play, play the song that the user would most enjoy listening to at that time –* making the *optimal* selection. Implementing this algorithm is not simple, however. The *optimal* selection can vary from user to user and from song to song – most people would not like to hear the same song twice in a row. Some users like a lot of variety in the songs they listen to in a single listening session, while others might like more consistency in their individual listening session. There are many possible attributes that might affect a user’s listening habits, such as:

* Genre
* Time since last play
* Plays per hour/day/week
* Weather
* Time of day
* Play density for genre (how often this genre has been played in a period of time)
* Play density for ANY song attribute
  + Artist
  + Features acoustic guitars
  + Features acoustic piano
  + Female vocals
  + Male vocals
  + Boy-girl vocals
  + No vocals (instrumental)
  + Fast tempo
  + Length of song
  + etc…
* Activity
  + Sitting alone
  + Talking with a friend
  + Having a party
  + etc…
* And many more…

The importance of these attributes can vary from user to user; and even for a single user, they might vary with time. Assuming these *hypothesis* to be true, the group’s song selection will work as follows:

*Given a list of M attributes – which might be significant in the selection of the optimal song at any given time – for each user, create a 1xM matrix indicating the significance of each of these attributes. When selecting a song to play from a list of N possible songs, for each of the N songs – each with an M x1 matrix indicating the presence of an attribute (1 for true, 0 for false) – multiply these two matrices. The result will be, for each of the N songs, a 1x1 matrix containing the likelihood that the user would like to hear that song. From the N songs, the optimal song will be the song with the maximum likelihood.*

In order for the above algorithm to work, the significance of each attribute must be determined for each user. How well the song selection algorithm performs will be based directly on how well the well the *significance* values are chosen for each user. These *significance* values will be probabilities and will be determined using statistical methods such as Naïve Bayes or Logistic Regression (see sections 3.8.1 and 3.8.2, respectively). Tentatively, the group will plan to use Logistic Regression to calculate these significance values.

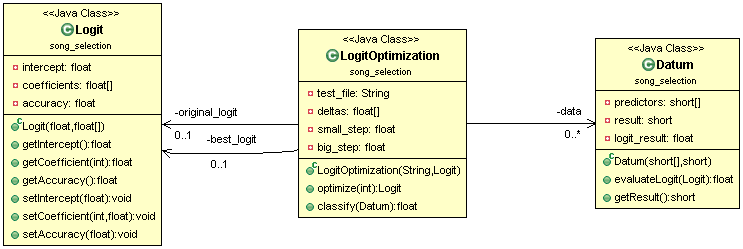


Figure 4.2 : A UML class diagram for the Logit Optimization process.

Figure 4.2 F shows the classes that would be involved in the *logit* *optimization* process. Although only a single bit is needed to store the predictors and result for each Datum, a short integer will be used to simplify the multiplication. A LogitOptimization object would be initialized with a String, *test\_file*, in the form mentioned in Example 3.8 A, and a Logit, *original\_logit,* in the form mentioned in Example 3.8 B. The following tasks will be performed at initialization:

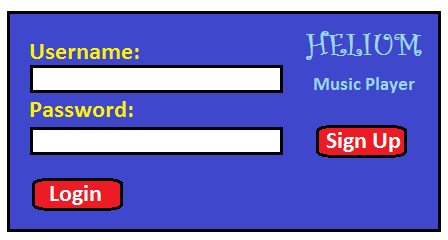
* A File, *aFile,* object will be created using the String *test\_file;*
* An ArrayList<Datum>, *data*, will be initialized;
* Each line in *aFile* will be stored as a Datum object in *data*;
* For *original\_logit*, the intercept will be calculated using Equation 3.8 C;
* For each Datum, d, in *data*:
  + *result* = d.evaluateLogit(original\_logit);
  + If *result* and *d*.*result* both indicate either *case* or *not case*, *correct++;*
* original\_logit.accuracy = *correct* / data.size();
* The *best\_logit* will be set equal to the *original*\_*logit*.

Given a number, N, *LogitOptimization.optimize(N)* will adjust all the coefficients of *best\_logit* up to N times each and measure the accuracy of the new Logit each time; setting a new *best\_logit* when necessary. Ideally, after N iterations, the *best\_logit* will have an accuracy of at least 80% on the test set. To make a song selection, the following would occur:

* With S possible songs to choose from, select P songs:
  + P may equal S, but this may cause long processing times for large values of S;
  + If P != S, select the P songs randomly from S;
* Represent each of the P selected songs as a Datum, d, minus the result:
  + We couldn’t possibly know what the result would be yet;
* For each of the P Datum, calculate the probability that user would want to hear that song using *LogitOptimization.classify(d)*;
  + If the probability is less than 50%, the user would probably skip the song;
  + Select M songs with the M highest probabilities **over** 50%:
    - A large value for M means that the algorithm might have to be ran less often, but could result in more skips;
    - M = 1 should reduce the likelihood of skips, BUT it will mean that the algorithm must be ran more often;
    - The group will experiment with different values of M to achieve the highest level of accuracy without causing noticeable processing delays;
  + If there are no songs with a probability greater than 50%:
    - If the device has adequate time to process the algorithm again, process the song selection algorithm again
      * The *ShuffledStack* object will be used to ensure that the same songs aren’t randomly selected again.
    - Else, select the song with the highest probability;
* Form a Playlist with the selected songs and begin playing the songs;
  + The playlist should be sorted from highest to lowest probability.

**4.2.4 Mobile App**

A mobile app will be developed to allow users to control Helium from their phones. The user will be able to connect their phones to the device using Bluetooth. Since the group has a severely limited number of man hours to develop software, the App will be kept fairly simple. Figure 4.2 G shows what the login screen of the app might look like.



*Figure 4.2 G: Login screen for Helium mobile app.*

The app will have the following screens:

* Login (Figure 4.2 G):
  + Will double as the sign up screen;
  + Errors logging in or signing up will be displayed to the right of the “login” button;
  + An alternative version of this screen might be included for when the device is held vertically.
* Sign up confirmation:
  + Will indicate that the sign up was successful;
  + One button, “continue”;
  + An alternative vertical screen might be included.
* Root selection screen (Figure 4.2 H):
  + The (horizontal) root selection screen will be divided into two vertical panes;
  + Initially, the left pane will contain the five selections pictured in Figure 4.2 G;
  + A yellow star will indicate which item is selected in the left pane;
  + When an item is selected from the left pane, the right pane will fill with all the items in that group (scrolling up and down will be possible if all items can’t fit);
  + Selecting an item in the right pane will move the right pane to the left pane and fill the right pane with all the items that belong to the selected item (see Figure 4.2 I);
    - For Playlists and Artists, this would be songs;
    - Selecting a song will start playing the song;
  + Swiping left will quickly take the user back to the previous screen
    - For example, swiping left on the screen shown in Figure 4.2 I will take the user back to the screen shown in Figure 4.2 H;

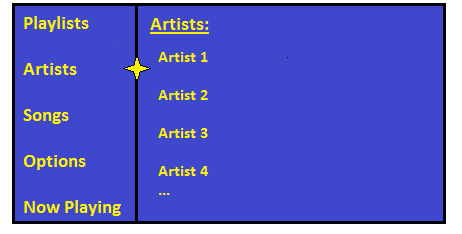
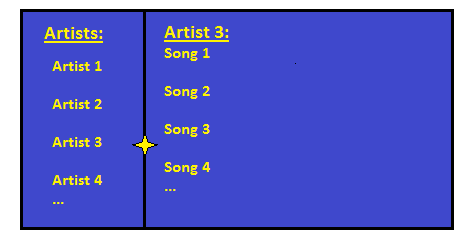


Figure 4.2 : The root selection screen with "Artists" selected.

* + Selecting “Options” will bring up the Options screen;
  + Selecting “Now Playing” will go to the playback control screen;
  + An alternative vertical screen might be included;
    - This would probably operate completely different from how the horizontal version is described above.



*Figure 4.2 I: The root selection screen after selecting "Artist 3".*

* Options screen:
  + Will include sign out button;
  + A shortcut to the equalization screen;
  + Possibly options for uploading new songs.
* Equalization Screen:
  + This screen will allow the user to adjust the gain up and down for each of the bands on the equalizer.
  + Will include some preset options, such as “flat” OR a link to a preset equalization screen;
  + Might include an option to save the current curve as a preset equalization curve.
* Preset equalization screen:
  + If time permits, this screen would allow the user to select from preset equalization curves.
* Queue screen:
  + Displays a list of the next few songs to be played in the current playlist:
    - For a normal playlist with *shuffle* turned off, this will display the entire playlist
    - For a *smart playlist* or a *shuffled* playlist, only a few of the songs will be in the queue at a time.
  + Swiping left or right will return to the playback control screen;
  + Clicking and dragging a song will allow the user to reorder the queue;
* Playback control screen:
  + Will display the name of the song currently playing as well as its length;
  + Will include basic playback control options:
    - Play
    - Pause
    - Skip Song
    - Jump to time in song
    - Shuffle
  + A link to the Equalization screen;
  + A link to the Queue screen;
  + Possibly, the album cover art for the currently playing song.

The app will be developed as a web app so that users with different types of phones will be able to use it without the group having to develop multiple apps. For the purpose of both uploading songs and quickly obtaining the list of song IDs already on the device, the mobile app will be able to connect to the same song server as the primary device. To the primary device, the following commands will be able to be sent by the mobile device:

* play(song\_id)
* pause()
* skip()
* seek(time)
  + where time is an integer representing seconds
* startPlaylist(playlist)
* toggleShuffle()
* login(user\_id, password)
* createUser(user\_id, password)
* removeFromPlaylist(user\_id, playlist, index)
* addToPlaylist(user\_id, index, song\_id)
* deletePlaylist(user\_id, playlist)
* createPlaylist(user\_id, playlist)
* createSmartPlaylist(user\_id)

For further information on how these commands will be formatted and sent, refer to section 4.2.6 Wireless Communications.

**4.2.5 RFID processing software**

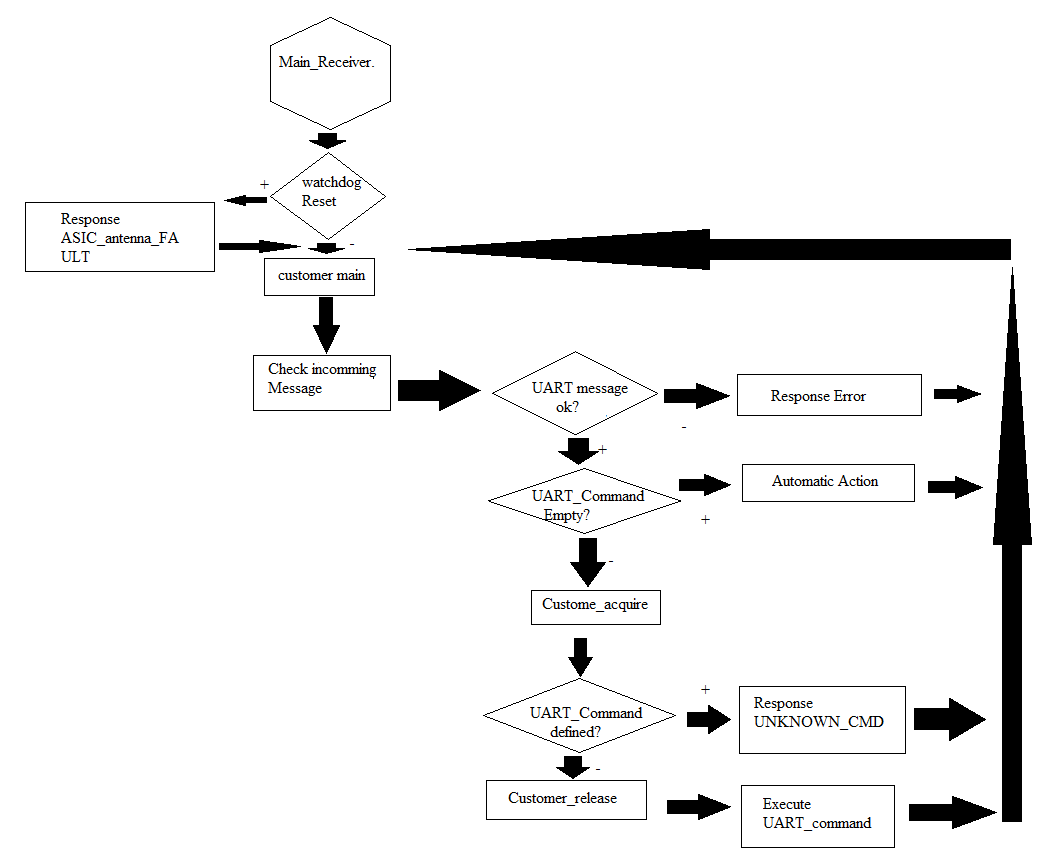
The programming of the RFID will include a mixture of C programming and assembly language for the MSP430 using code composer compiler. The RFID system will include the program of the three materials required for this device, the LCD screen, the RFID base station and the Bluetooth device.

The EM4094 RFID base station comes with a disk that the C program code within the disk. The code only works for the Atmega64, but the code can modified to be able to work for the MSP430. The code comes in four levels that will be described in detail in Table 4.2 J.

|  |  |
| --- | --- |
| Level 1 | Define decoding routings |
| Level 2 | Define low level data send and data extraction routines |
| Level 3 | Define high level data transformations and, main loop body, boot loader and simcard. |
| Level 4 | Define UART communication routines |

*Table 4.2 J: the levels for the RFID*

The coding of the RFID must be able to take the information from the transponder and be able to store the data to a register within the MSP430 to later process the information. Part of the stage is to identify the person, then the RFID must send the information of the song to the Beagle bone to search for the song in question. Figure 4.2 K illustrates the flow chart of the RFID reader.



*Figure 4.2 J: Flow chart of RFID reader*

The LCD module will include the state of the RFID and will state the name of the individual within the transponder. Table 4.2 L illustrates what the LCD must display for easier troubleshoot.

|  |  |
| --- | --- |
| Location | Response |
| Main loop | RFID reader on standby  Welcome! |
| ASIC\_antenna\_Fault | There was an error on the antenna. |
| Response error | Error failure to connect to active transmitter |
| Automatic Action | There is no information within the transmitter |
| Response Unknown\_CMD | Error  Input wrong command |
| Customer acquire | Hello “-----“  Your song will play shortly |
| Customer release | Have a nice day! |
| Execute UART\_command | Sending song request |

*Table 4.2 L: LCD display programming*

**4.2.6 Wireless Communications**

Helium must be capable of both communicating with a server on the internet, and communicating wirelessly between its two distinct devices; the relatively powerful BeagleBone Black, and the more simple and low-power MSP430. The devices will use Wi-Fi to connect to the internet and Bluetooth to communicate with each other.

**Wi-Fi Communications**

The primary device will be using Wi-Fi to connect to the internet and query the song server. It will have to be able to both manipulate and query from this server; these queries might include relatively large music files, in the form of binary objects. Reliable data transfer (rdt) will be essential for communicating with the song server. To reduce the amount of software development required for wireless communications, the group will use a Wi-Fi module in the primary device that already implements TCP. The only communications over the internet will be SQL queries, so this communication process will be simplified considerably for the group.

**Bluetooth Communications**

The primary and secondary devices will communicate with one another via Bluetooth. Since the MSP430 is a relatively simple device, the communications will have to be kept simple. In addition, the primary device must be capable of communicating with mobile devices, for control purposes.

The secondary device will only be transmitting user IDs, RFID tag numbers, and one of only a few simple messages. The secondary and primary devices will be able to send the following messages between one another:

* user entered
* user left
* associate user\_id with RFID
* delete association between user\_id and RFID
* ACK (for rdt purposes)

The user\_id will be stored as a single 8-bit number and the RFID will be stored as a 32-bit number.

The data sent between the primary and secondary devices will be formatted as shown in table …; to keep things simple, the commands will be stored using a single byte (instead of just the necessary 3 bits); this will also make it simpler to add additional commands to the device in the future and keep the size of all packets some number of bytes. Since only certain commands will need the RFID, the presence of the optional 4 Bytes can be determined by the Command byte. Table … shows how the commands will be encoded.

|  |  |  |
| --- | --- | --- |
| 1 Byte | 1 Byte | 4 Bytes (optional) |
| Command | User\_id | RFID |

|  |  |
| --- | --- |
| Command | Code |
| User entered | 0000 0001 |
| User exited | 0000 0010 |
| Associate user\_id with RFID | 1000 0000 |
| Delete association between user\_id and RFID | 0001 0000 |
| ACK | 0000 1000 |

Mobile devices will be able to send the following commands to the primary device.

* play(song\_id)
* pause()
* skip()
* seek(time)
  + where time is an integer representing seconds
* startPlaylist(playlist)
* toggleShuffle()
* login(user\_id, password)
* createUser(user\_id, password)
* removeFromPlaylist(user\_id, playlist, index)
* addToPlaylist(user\_id, index, song\_id)
* deletePlaylist(user\_id, playlist)
* createPlaylist(user\_id, playlist)
* createSmartPlaylist(user\_id)

In addition, the group would like for a user to be able to upload new songs to the primary device using Bluetooth; however, this is only a low priority objective. All of these commands will also be able to be encoded using a single byte and the user\_id’s sent will still be only a single byte; however, integers will be required for both the playlist and the index. Rather than sending the plain text which would necessitate adding a length field (for pathwords of arbitrary length), the passwords will be hashed into a 32-bit integer. The format used for sending commands to the primary device is shown in table…

|  |  |  |  |
| --- | --- | --- | --- |
| 1 Byte | 4 Bytes | 4 Bytes | 4 Bytes |
| Command | User\_id | Password OR playlist index | Song\_id OR song index (for a playlist) |

The only data that the primary device will send to the second device is a single byte at a time as an ACK.

In figure 4.2 K, below, a typical sequence of communications between the primary, secondary, and mobile devices, as well as the media server, is shown. Two users log in via their mobile device and make requests to play songs. The second user’s request is denied because the first user has a higher priority than the second user. The group will probably use a “first come, first served” method for assigning priority, or we may allow for certain users to control the priority of others. The “check for message” action from the primary device doesn’t represent an actual message sent over Bluetooth, but rather indicates that the device is checking the thread dealing with communications with the other device.

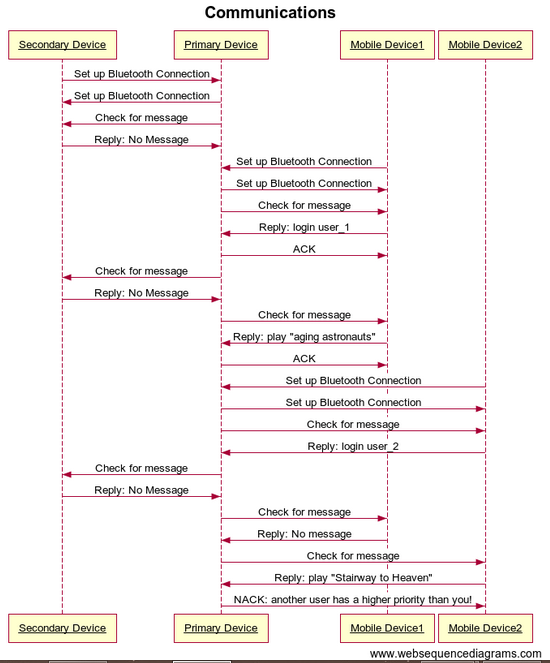


Figure 4.2 K: A Sequence diagram of two users logging in via mobile devices.

**5.0 Administration**

Before beginning the construction on the prototype for Helium, it is best to first set up a budget, get to an agreement on who is going to be in charge of each module; and to decide on deadlines for each module. Making these decisions now will help to avoid future complications with the project and help to ensure that the project will be completed on time.

**5.1 Budget and parts acquisition**

RFID system costs:

EM 4094 $5.00

MSP430f5529 $7.00

EM4135 $6.00

PAN1315ETU-ND (blue tooth) $71.61 \* 2

Fuel tank $27.00

Quartz crystal (13.56MHz) $0.27

RioRand™ LCD Module $10.99

Total $200.39

Amplifier and audio output system costs:

10k Potentiometers 5pcs $3.97 \* 2

AD5206BN10 DIGITAL POTENTIOMETER $16.14 \* 3

Set of 10 Pieces IC TL084CN TL084, 084 14 pins DIP $7.43

Sunkee 10pcs LM339 LM339n- Quad Comparator ICs $11.48

VS1011 - MP3 Decoder Circuit $20.64

Total $79.77

MCU system and audio input:

SanDisk Cruzer CZ36 32GB USB 2.0 Flash Drive $14.99

BeagleBone Black $55.00

BeagleBone Black Cape - LCD (4.3") $59.95

SparkFun Electret Microphone Breakout $7.95

Total $137.89

Miscellaneous:

Resistor pack (10ohms – 10Mohms $9.64

Capacitor pack (4.7 pf - 470 uf) $20.00

CD74HC4016 $1.40 \* 5

Printed Circuit Board $4.00

Total $40.64

Cost for the whole project $459.69

The time estimate and the location of purchased is displayed in Table 5.1 A-B.

|  |  |  |
| --- | --- | --- |
| Material | Location | Time estimate of shipping |
| EM 4094 | Asia Components Co | Ten business days |
| MSP430f5529 | TI website | Already arrived to destination |
| EM4135 | Asia Components Co | Ten business days |
| PAN1315ETU-ND (blue tooth) | digi-key corporation | Five business days |
| Fuel tank | TI website | Six business days |
| Quartz crystal (13.56MHz) | Amazon.com | Five business days |
| RioRand™ LCD Module | Amazon.com | December 9th |
| 10k Potentiometers 5pcs | Amazon.com | Five business days |
| AD5206BN10 DIGITAL POTENTIOMETER | Amazon.com | Ten business days |
| Set of 10 Pieces IC TL084CN TL084, 084 14 pins DIP | Amazon.com | Already arrived to destination |
| Sunkee 10pcs LM339 LM339n- Quad Comparator ICs | Amazon.com | Five business days |
| VS1011 - MP3 Decoder Circuit | Sparkfun.com | Seven business days |
| SanDisk Cruzer CZ36 32GB USB 2.0 Flash Drive | Amazon.com | Ten business days |
| Beagle bone black | Sparkfun.com | Seven business days |

*Table 5.1 A: Delivery time estimate and location of purchase*

|  |  |  |
| --- | --- | --- |
| Beaglebone Black Cape - LCD (4.3") | Sparkfun.com | Seven business days |
| Audio-Technica ATR-1100 | Amazon.com | Eight business days |
| Resistor pack (10ohms – 10Mohms | Amazon.com | five business days |
| Capacitor pack (4.7 pf - 470 uf) | Amazon.com | Five business days |
| Beaglebone Black Cape - LCD (4.3") | Sparkfun.com | Seven business days |
| Audio-Technica ATR-1100 | Amazon.com | Eight business days |
| CD74HC4016 | Ti.com | Seven business days |
| Printed circuit board | Amazon.com | Seven business days |

*Table 5.1 B: Delivery time estimate and location of purchase.*

**5.2 Individual Responsibilities and time estimates**

Now that it is known how much it is going to cost to build the project, and how long it will take to acquire the materials for each section, it is time to establish some deadlines for each section. The figure below, Figure 5.2 A, has the name, section, and the estimated time that should take to set things up for each member.

|  |  |  |
| --- | --- | --- |
| Name | Section | Time |
| Giovanni Goris | RFID System + miscellaneous | Four weeks |
| Brandon Jones | Project Management, Programming Helium | Five weeks |
| Aaron Ford | Power system, audio input, PCB | Four weeks |
| Nick Guillotte | Equalizer, digital to analog converter, amplifier, audio output, wireless communication. | Four weeks |

*Figure 5.2 A: The responsibilities for each member of the group.*

**Giovanni Goris**

Giovanni will be responsible for the RFID system, the secondary device, and any programming necessary for it to communicate with the primary device. The secondary device involves putting together a working reader that will be able to wireless send the information to Helium. One of the most important aspect of the RFID system is that it must connect to the beagle black MCU that will be used to connect the other sections together. The assembly of the RFID should take, at most, one week to put together and should take another week to write the code to communication to a computer terminal to send the information. The third week is to improve the code to be able to use the Bluetooth to connect to another Bluetooth source. During the fourth week the RFID can be tested by itself to see if the transponder can send the information to the computer that the reader is connected, this can be done using UART to backdoor the transmission to the computer after the MSP430 receives the information.

**Brandon Jones**

Brandon will be programming Helium, as well as the MySQL database, which will act as the media server for the primary device. In addition, Brandon will be the project manager for the group. It will be his responsibility to monitor the progress of the project and to keep everyone on task. The media server for Helium will take, at most, a week to build, this time includes the time it will take to write the SQL queries that will be necessary for retrieving data from it. The programming necessary for device control and media playback, as well as communications, should take around 2-3 weeks. The amount of time necessary to develop the song selection algorithm can be a bit difficult to predict, as there is no easy way to measure its performance; for now, we will predict about 2 weeks.

**Aaron Ford**

Aaron will be responsible for the power system, audio input (voice commands), and anything related to PCB. The power system should be able to supply enough energy to turn on Helium and its components. One of the roles is to make sure that the one in charge of the power system takes into account the heat that each component will dissipate and to make sure to get some heat sinks to avoid overheating. The time that will take to analyze the design and come up with solutions to any power problem or overheating problem should take around two weeks at the least.

The audio input system will be used to voice commands to Helium to pick different songs at the discretion of the user. The designer of the input system must be able to create a way for Helium to convert sound waves to usable data, and to be able to deny the sound waves produced by Helium itself while playing the songs for the users. This should take about one week to create, and another week to test to see if the component can only pass the frequency ranges of human speech.

**Nick Guillotte**

Nick will be responsible for all aspects of the Helium’s audio output, this includes the equalizer, as well as any DACs or amplifiers that will be necessary. A digital to analog system needs to be created to be able to convert the outputs to a workable source for the audio output source. It should be able to convert the song stored inside Helium to a workable voltage output that can then be used by the equalizer to be boosted at certain frequencies. The digital to analog converter should take at the least one week to be implemented to the circuit.

To be able to play songs from Helium, an amplifier will be needed to convert the digital signal to be heard as sound. The amplifier should be able to play the songs at a range of at the least two meters to be able to be heard by most users in the room. The next part of the amplifier, it must come with a way to turn the volume up or down so using some potential meters to control the output of the sound will be required.

As stated above the output sound should not be able to interfere with the input system of Helium. Using an equalizer, the output signal can be programmed to sound more natural for the users to listen, and boost the sound from certain frequencies out while canceling the sound from certain frequencies. The shelving equalizer can be used to boost the frequencies between certain ranges as a way to avoid the low frequencies that will be used for the input audio system. This should take at the least two weeks to design and implement.

**Weekly Progress Chart**

Tables 5.2 B and C, below, details the group’s activities for each week until the project is finished. Several **Priority Deadline**s have been established in the chart below. These deadlines are not meant just to indicate a deadline for some task, but rather, to indicate the date when either the task must be completed by OR the responsible member MUST address the issue with the group. Addressing the issue with the group might include asking for an extension, asking for additional help, or requesting replacement parts. By the end of the first week of classes in the spring semester, each member will be required to acknowledge each of their priority deadlines. This way, no member can claim that they didn’t know about a deadline.

The group has had issues meeting the deadlines for this first semester, and because of that, the weekly progress chart has been completely reworked. This includes the addition of priority deadlines. Several catch up weeks have been added to the schedule, this will help keep the group more evenly paced. In addition, the group will be scheduling “all hands on deck” days, several times throughout the upcoming semester. These will be days where the group will be expected to make significant progress on the project in a single night. These days are meant to simulate the rushed, all night, last minute working schedules that the group attempted to work at before missing the deadline for this report. Our hope is that by forcing these long, difficult nights at convenient times during the semester, rather than the last few weeks of the project, that we will save ourselves from a lot of stress.

|  |  |
| --- | --- |
| December 15 | Order all parts required for the project |
| January 06-12 | Receive all parts from amazon.com  Get to know the BeagleBone Black.  Begin building RFID scanner.  Look into sources of metadata for songs and artists. |
| Jan 13-19 | **Priority Deadline (All, Jan 16): Individual acknowledgements of priority deadlines due.**  Begin coding the *MediaController* module.  Implement media server.  Begin coding the secondary device.  Get to know the voice command module.  Get to know the microphone. |
| Jan 20-26 | Build prototype for equalizer.  Begin coding the *DeviceController* module.  Implement the LCD screen for the RFID system.  Test the LCD screen to display the state of the RFID.  Implement temporary battery power for the primary device (should provide enough power for just the BeagleBone Black).  Begin integration and testing of microphone/voice command module and the *MediaController* module with the BeagleBone Black.  **Priority deadline (Aaron, Jan 26): Find a viable source for metadata collection.** |
| Jan 27-Feb 2 | Measure audio output quality.  Finish testing voice commands.  Begin song selection algorithm.  Implement the power system for the BeagleBone Black.  Test MCU to connect to remote server using Ethernet connection.  Test equalizer prototype.  Test RFID tag writing.  **Priority Deadline (All, Feb 2): The primary device will play music.**  **Priority Deadline (Nick, Feb 2): Finalize design for equalizer.** |

*Table 5.2 B: Implementing and testing schedule*

|  |  |
| --- | --- |
| Feb 3-9 | Begin implementing Bluetooth communications between primary and secondary devices.  Begin coding *Communication* module.  Look into conversation detection.  Volume control prototype (mechanically controlled).  **Priority Deadline (Giovanny, Feb 9): Finnish programming and building the secondary device.** |
| Feb 10-16 | Catch up week! Work on any uncompleted tasks.  **Late Deadline (Feb 16): All priority deadlines before Feb 3 should be finished.** |
| Feb 17-23 | Implement volume control on primary device.  Implement wireless communications between primary device and media server.  Test equalizer on audio output from mp3 decoder. |
| Feb 24-March 2 | **Priority Deadline (All, Feb 27): Last call for additional or replacement parts.**  Finish designing power supply and implement cooling system if required.  Test the power supply required and integrate with the primary device.  Final tests of volume control and gain control of equalizer.  Final tests of wireless communications with media server. |
| March 3-9 | Catch up week! Work on any uncompleted tasks. |
| March 10-16 | Begin populating media server.  Evaluate song selection algorithm’s performance |
| March 17-23 | Begin development of mobile app.  If on time, begin development of conversation detection module; else, drop feature.  Investigate methods of increasing RFID scan distance. |
| March 24-30 | If on time, look into improvements of song selection algorithm.  Test mobile app.  If on time, look into uploading data to media server via mobile device.  All unit tests should be finished or in progress. |
| March 31-April 6 | **Priority Deadline (All, Mar 31): Start full system testing (all unit tests should be complete and successful).**  Complete full system testing, or create full system bug report. |
| April 7-13 | Resolve full system bugs (if any).  Unit testing for bug fixes (if any).  Begin second round of full system tests. |

*Table 5.2 C: Implementing and testing schedule.*

Placeholder

**5.4 Software Development Plan**

The group will be coding in Java for the primary device and C for the secondary device, in addition, a MySql server will also be designed. There is relatively small amount of coding needed for the secondary device, whenever possible, resources will be found online for implementing the most complicated tasks, such as TCP/UDP, on this device. Giovanny Goris will be responsible for developing the software for the secondary device.

The software required for the primary device will be much more complicated. All the Java coding will be done using the Eclipse Juno IDE. Many of the classes and methods needed for the primary device have already been designed (see section 4.2); an overview of the class structure is provided again, below, in table 5.4 A. Brandon Jones will be responsible for all of the Java code in the project.

Although the group will likely not have multiple members writing the code for the primary device; however, since the device will be coded over a long period of time, the software of the primary device will be kept strictly organized. The details of the organization of the primary device’s software are discussed in the following sections.

**5.4.1 Package Structure**

The software for the primary device will be divided into 5 packages: Control, communication, storage, users, and song selection. The classes belonging to each package are detailed below.

**Control**

The *control* package will contain the following classes:

* Init
* DeviceController
* MediaController

The *Init* class will initiate the threads for both device control (*DeviceController*) and song playback (*MediaController*). These two modules will need to run simultaneously. Any other tasks that need to be completed will be performed by the main thread in *Init*.

**Communication**

The *communication* package will contain the following classes:

* Communication
* DatabaseInterface
* Location
* Weather

All of the SQL queries necessary for communication with the song server will be generated by the methods in the *DatabaseInterface* module. Each of the public methods will generate a SQL statement in the form of a *PreparedStatement*; each of these methods will then use the private method *queryServer()* to get the result set that they will need. For the constructor, the URL, USER, and PASS fields will probably be hardcoded in the Communication module; however, they are made variables in this method so that this module won’t have to be changed if the server is.

The *Location* module will contain the code from Skyhook’s API for WPS lookup; a *Weather* object will also be stored here. The *Weather* module will run the API code to search the weather server.

**Storage**

The *storage* package will contain the following classes:

* Playlist
* MasterTable
* ShuffledStack
* Song

**Users**

The *users* package will contain the following classes:

* UserManagement
* User
* User\_song\_attributes

**Song Selection**

The *song*\_*selection* package will contain the following classes:

* SongSelecter
* Logit
* Datum
* LogitOptimization

**5.4.2 Coding Conventions**

The group’s code will focus on readability above all else. The names of all variables and methods should be kept descriptive, regardless of length. All variables will be lowercase, with ‘\_’ separating words; methods, will use camel case, suchAsThis; and will not contain any punctuation. Methods should be grouped into three areas, “Getter Methods”, “Setter Methods”, and other methods (such as constructors). For the primary device, which is programmed in java, most class variables will be kept as private and any methods that are meant to be used only by the calling class, will also be made private.

**6.0 Testing**

The next stage of project is to test the prototype before starting to build the final product of Helium to know if everything is going as planned. The test will include different sections to see if all the separate components works by themselves perfectly before putting everything together to avoid confusing and extensive troubleshooting. The first part of the testing will be first the hardware, this is to make sure there is no faults in the circuit as it gets a current through to avoid mishaps and hazards. The next set of the testing will be the software, if the component needs to be programmed, to make sure that the device can take into account every possible situation to correct any bugs that might unintentionally appear.

**6.1 Hardware Testing**

The hardware that will be the physical backbone of Helium needs to be tested to avoid situations that can physical break down Helium. Another reason that needs to be tested is to make sure the components can do its assigned role, such as send the information to Helium from a transponder, or cellphone. Helium must be able to play sound from the source to be clear enough and loud enough to be heard so testing the amplifier to see if it can play songs is a must, and testing the equalizer to make the words clear enough to be understood. Helium must be able to take into account voice command so a microphone must be able to avoid the frequency of the sound emitted by Helium and other sources to just take in the voice of a person.

**6.1.1 Test environment**

The testing region that will be used to test the components must include measuring devices to measure the voltage, current, and other measurements that needs to be taking into account. Another important aspect of the environment, is that it must include a power supply to turn on the components, and also implementation equipment such as a breadboard to prototype the device. Although an oscilloscope can be used to test the output frequency and the demodulation of the signal, the lab will also require a power source that can change frequencies to test the RFID and the equalizer to make sure that the 13.56 MHz can be demodulated and the equalizer can create a band pass filter to avoid complications with other devices.

The lab will include the following resources to allow the group to test Helium components before setting the whole project together:

* Oscilloscope: Tektronix DPO 4034B Mixed Signal: The oscilloscope will test the output of any signal that going through the different components to see if it’s following the expected calculations.
* Function Generator: AFG3022B: The Function generator will be used to test out the equalizer to see if the frequency range of the equalizer can be changed through user input as a way to clear out any unwanted disturbance.
* Triple Power supply: is needed to power the different components during the testing. The op-amps and converters will require the power supply to be operational during the testing and a Triple power supply can allow a possible and negative voltage for this purpose.
* Voltage meter: the voltage meter will be used to calculate the current and voltage of the different components, this is to make sure that the signal has enough power to function properly such as the converters or the RFID base station has enough power to provide the logic function for the MSP430 to register.
* Computer with Multisim: Using Multisim will be used to simulate the components to at least get some measurements to be used to compare the experimental results to that of the simulated results.

To understand if there is any problem with Helium a series of test must be done to see some results, Table 6.1 A shows the requirements of the experiment.

|  |  |
| --- | --- |
| Testing consideration | Description |
| C/N ratio | For the RFID to be able to work, it must be able to have a high enough carrier to noise ratio so the reader can be able to read the transponder. The quality of the ratio must be about 7.95 db. |
| Sound quality | The point of Helium is to use an RFID to play the song without the input from a user. If the user just hears static then no one would want to get a Helium when an MP3 can work just fine. |
| Interference testing | One of the test that must be done is to make sure Helium does not interfere with other devices surrounding it. The FCC has a rule that devices must not interfere with any other device meaning signal sent to and from Helium must not disrupt the signals from other devices. |
| Voice command | Since Helium will use voice command functions for the user to command Helium to act, there needs to be testing on the microphone to see if the voice of the user reaches Helium without interference. The test should include the command to be output to a screen to see if that was the original command and to test the microphone and audio output are not conflicting with each other. |
| Sound intensity testing | This type of testing is different from the quality testing in that this is a test to calculate the decibel to make sure it does not go over 85 decibels for long periods of time. A person can hear sound at 85 for at most 2 hours before it starts permanently damaging the ear so the design must take into account the time 85 decibels is reached or try to avoid this problem all together. |

*Table 6.1 A: testing consideration for Helium*

The testing of Helium must be able to done on a location that can be used to test the limit. See Table 6.1 B for the testing requirements for the RFID system. The other type of testing required for Helium are illustrated on tables 6.1 C-F.

|  |  |
| --- | --- |
| Test sequence | Description |
| Objective | To measure the carrier to noise ration of the RFID system to be able to read the transponder. The ratio must be above 7.5 dB to be able to function properly. |
| Equipment | Voltage meter and oscilloscope to find the power of the antenna for the RFID base station and transponder. Thermo sensor to measure the temperature of the antenna and transponder. |
| Test summary | The temperature of the transponder antenna and the transceiver is measured to find the system noise temperature to determine the noise power. The power is calculated to find the antenna gain of the transceiver in which the carrier power is calculated from the power received, the antenna gain and path loss. The temperature of Helium must be changed to get different results. |
| Procedure step 1 | Measure the temperature of the antenna. |
| Step 2 | Calculate the power of the antenna. |
| Step 3 | Calculate the efficiency of the antenna. |
| Step 4 | Calculate the antenna gain. |
| Step 5 | Calculate the path loss of the antenna. |
| Step 6 | Compute the noise power and carrier power.  C = 10\* log (P \* Gt \* Gr \* Lp \* Lsys).    N = 10 \* log(k \* B \* Tsys) |
| Step 7 | Divide the carrier by the noise to get the carrier to noise ratio. |
| Calculation | Calculate the noise and carrier power for the different temperatures and at different distances. This should determine how far the transponder must be from the RFID base station antenna for the device to determine if it’s there. |
| Report | This should include the description of the device, the calculations required for the carrier power and noise power, the results of the testing on different temperatures and the distance the RFID reader can work away from the transponder. |

*Table 6.1 B: Carrier to noise ratio test*

|  |  |
| --- | --- |
| Test sequence | Description |
| Objective | To determine the quality of the sound that Helium outputs. This is to make sure the signal to noise rate is high enough for the listener to hear the song with near perfect quality. |
| Equipment | Oscilloscope to measure the noise and frequency of the component. A voltage meter to determine the gain of the amplifier. |
| Test summary | The signal power and noise power is calculated to determine the quality of the sound. This include calculating the output power of the speaker, and determining the frequency response of the sound. The signal to noise ratio must be above 80 dB to function well and the distortion of the sound must be 1-5%. |
| Procedure Step 1 | Measure power output. |
| Step 2 | Measure Frequency response. |
| Step 3 | Measure noise power |
| Step 4 | Calculate the SNR of the signal.  SNR = 20 log (Signal power/Noise power). |
| Calculations | Using the frequency response and using the equalizer, find the bandwidth to measure the noise power. Using the noise power and power output find the SNR, if it’s above 80 dB then is good to go. |
| Report | This should include the description of the device, the calculations required to find the signal power, and noise power. Test the device at different bandwidth to find which region provide the best sound quality. |

*Table 6.1 C: Sound quality test*

|  |  |
| --- | --- |
| Test sequence | Description |
| Objective | To test the voice command function to receive perfect sound quality to implement command. |
| Equipment | Oscilloscope to determine the Noise power of the device. Voltage meter to determine if the A/D converter is working to connect the signal to the Beagle bone black. |
| Test summary | The microphone will be tested to see if it can send the command of the user to the beagle bone to command Helium to follow the command. |
| Procedure Step 1 | Measure the power received through the microphone. |
| Step 2 | Measure the voltage of the analog signal. |
| Step 3 | Measure the voltage of the digital signals. |
| Step 4 | Determine the output command that was said. |
| Calculations | The result should have put the signal command to a register on the beagle bone black that can then be converted to ASCII characters. |
| Report | This should include the description of the device. The calculations of the power and noise frequency should go here too. The measurements of the voltage for both the analog and digital side should be include here along with the output result stored in the register. |

*Table 6.1 D: Voice command test*

|  |  |
| --- | --- |
| Test sequence | Description |
| Objective | To measure the interference that will be caused by the Bluetooth device. |
| Equipment | A spectrum analyzer is required to determine the amount of interference with in the room with Helium. |
| Test summary | The spectrum analyzer will calculate the energy spectrum of the signal of all devices nearby and then look through the result to determine if there is interference with any other device. |
| Procedure step 1 | Use the spectrum analyzer to record the interference of the signal. |
| Step 2 | Calculate the dB of the signal if its -70dB to -80 dB the device will cause interference. |
| Step 3 | Repeat steps 1 and 2 for each device within fifty feet of Helium. |
| Calculations | Using the spectrum analyzer, determine which signal will interfere with Helium. |
| Report | The description of the device must be stated. The spectrum analyzer for each device, include Helium, must be included. Include ways to reduce wireless interference within the report to avoid any interference if there is any within the area of Helium. |

*Table 6.1 E: Interference testing.*

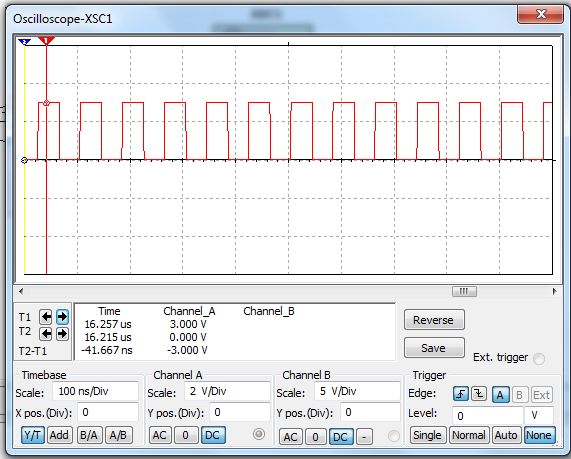
|  |  |
| --- | --- |
| Test sequence | Description |
| Objective | To calculation the total sound level of Helium to avoid long term hearing damage to the user. |
| Equipment | Use a device that can measure the sound intensity of the device as sound is playing such as a sound pressure level meter. |
| Test summary | The test will consist of calculating the total sound level and intensity of Helium to determine the maximum amount of power Helium should be able to use to power the speaker or the minimum distance the user must be away from the speaker to avoid hearing damage. |
| Procedure step 1 | Determine the intensity of the sound of the sound that is coming out of the speaker. |
| Step 2 | Determine the range that the measurement from step 1 was done away from the speaker. |
| Step 3 | Convert the intensity measurement to decibels if the device does not measure in decibels. |
| Step 4 | Using the equation L2 = L1 – 20 \* log (r1/r2), determine r2 for which the intensity level l2 is at 85 db. This is the minimum distance away from the speaker to avoid hearing damage. |
| Step 5 | Repeat step 1-4 using a different power levels for the speaker. |
| Calculations | The distance of away from the speaker should increase as the intensity level, from the speaker increases. |
| Report | The description of the device must be states. The intensity level and the distance from the speaker must be states. The calculations and minimum range from speaker must be stated and expected optimum power uses must be states to avoid hearing loss from the users. |

*Table 6.1 F: sound level testing.*

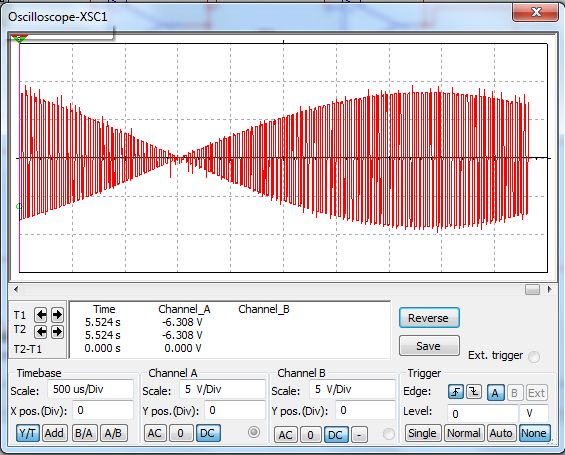
**6.1.2 Unit Testing**

**RFID system**

The testing of the RFID system will included the voltage of the inputs and outputs as the RFID connects to the MSP430, to make sure there is enough voltage to be read by the msp430. The clock of the RFID must be within the 12.53 MHz range to be able to synchronize with the inputs. Figure 6.1 G illustrates a representation of the clock that must go to one of the inputs. The next part to test is the transponder. The EM4135 uses Am modulation to send the signal to the RFID base station, in which is then demodulated and sent to the MSP430. Since the AM modulation is used then using an oscilloscope to should determine the modulated signal as is leaving the transponder, which should look close enough to Figure 6.1 H.



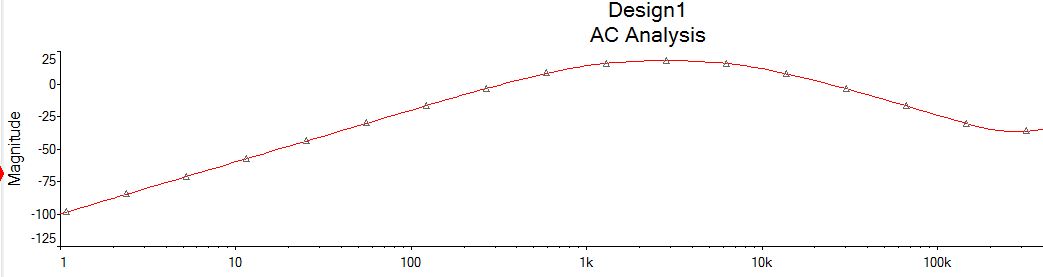
*Figure 6.1 G: The clock rate of the demodulated signal.*



*Figure 6.1 H: Modulated signal of transponder*

**Audio output**

The next test will be to make sure the equalizer works properly. Using the oscilloscope, the equalizer should be able to band-pass a certain frequency of the input signal to the amplifier, while ignoring the rest of the frequencies. Figure 6.1 I shows a representation of what the equalizer should be able to do during the test. To test the equalizer, a signal will pass through the component and the oscilloscope will check the output signal to measure the result, if the result follows what is expected of a band-pass filter then the next stage is to test the amplifier, if not then the component will be checked to see what is wrong. The amplifier should be able to take a signal and make sure to increase the output sound of the source. The test will include changing the gain to calculate the maximum distance that the sound can reach within the room. Using an oscilloscope, the gain of the circuit should be tested.



*Figure 6.1 I: Ac analysis of a band-pass filter.*

**Beagle bone black and digital to analog converter**

Although the beagle bone black would fall to a software test section, there needs to be tested if there is no fault within the microcontroller. To test this, the outputs will be set high and the output voltage of each pin will be measured to compare the result to that of the datasheet. The inputs will receive the turn on voltage and a physical inspection will reveal if there is a problem with the pin and the beagle bone black will shut down due to overheating. Once the beagle bone black is fully tested, the next part is to test the digital to analog converter to make sure the proper voltage is set to go to the equalizer. To test the converter, the best solutions to use a dc voltage source to act as the outputs of the microcontroller. The output voltage will be calculated to compare the resulting analog voltage to the expected results based on which bit is the most significant to least significant bit. Figure 6.1 K is an example of a digital to analog converter, while Table 6.1 J is the expected results of circuit in Figure 6.1 K.



*Figure 6.1 K: Simple digital to analog converter.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D4 | D3 | D2 | D1 | Output |
| 0 | 0 | 0 | 0 | 0V |
| 0 | 0 | 0 | 1 | 3.33V |
| 0 | 0 | 1 | 0 | 3.84V |
| 0 | 0 | 1 | 1 | 7.17V |
| 0 | 1 | 0 | 0 | 4.17V |
| 0 | 1 | 0 | 1 | 7.5V |
| 0 | 1 | 1 | 0 | 8.01V |
| 0 | 1 | 1 | 1 | 11.34 |
| 1 | 0 | 0 | 0 | 5V |
| 1 | 0 | 0 | 1 | 8.33V |
| 1 | 0 | 1 | 0 | 8.84V |
| 1 | 0 | 1 | 1 | 12.17V |
| 1 | 1 | 0 | 0 | 9.17V |
| 1 | 1 | 0 | 1 | 12.5V |
| 1 | 1 | 1 | 0 | 13.01V |
| 1 | 1 | 1 | 1 | 16.34V |

*Table 6.1 J: The expected results of the digital to analog circuit.*

**Graphic Equalizer**

The important measurement for the test procedure of the graphic equalizer is voltage gain. The whole premise of a graphic equalizer is the application of gain centered at certain frequencies to manipulate the tone of the music playing. The design also calls for a constant value of Q, so the test procedure will involve testing of the Q factor. In order to test the efficiency of the equalizer, the input and output voltage will be tested for each of the bandpass filters. It is crucial that each filter have the capability to apply enough gain to properly change the tone of each frequency band to cover the entire audible range. Voltage readings will be taken with each potentiometer set in the middle, at full boost, full cut, and used to generate a gain versus frequency plot. From this plot, the Q-factor can be measured to verify that the value is being held at a constant value to apply proper filtering of unnecessary frequencies.

Current readings are also important for testing the power and thermal limitations of the equalizer. Each filter needs to be able to support the current flow without overheating. Current probes will be applied to each circuit element to provide the data required to apply the power and thermal calculations to be compared with the limitations of the datasheet for each vital element.

**Speaker**

One of the values necessary to test for a speaker is the impedance. To test the efficiency of a speaker, the impedance is measured over the entire frequency range. For maximum efficiency, the impedance of a speaker should not drop below 80% of the nominal value. The nominal impedance of the speaker chosen for this design is 4 Ω. Therefore, the impedance should remain over 3.2 Ω throughout the entire frequency range.

**6.1.3 Hardware integration test**

Once all the individual components has being tested, is time to put the whole project together. The first part of the test is to determine the heat dissipated for each component. The operational component must be below a certain temperature to work properly, usually 125°C, if it is tested to go above that temperature then a heat sink must be implemented on the circuit to avoid complications. To find the temperature, the power dissipated must be calculated, once the power is calculated, the thermal resistance must be determined. The temperature of the components then has to be determined using equations such as Tja = P \* Rtb, where Tjc is the difference in temperature of the junction and the temperature of the room.

**Voice command**

Once the power supply has being connected to the completed project, is time to test if Helium can work as desired. Using a voltage meter, determine if the required number of volts is passing through the pins for the Beagle bone black as expected based on the A/D converter, using LEDs to determine the analog to bits conversion can help in the task. Once the next is determined a program should be able to store the response and then output the command on the display, if this works out then the hardware interface of the voice command is good to go.

**RFID system conversion to Helium**

Once the Bluetooth device is set up to connect, try to send some commands to Helium using a computer and the MSP430 by itself. Helium must be able to store the characters typed through the keyboard and output the results to the screen, to determine if it connects or not. If it connects the RFID system can be tested with Helium to see if a command can be sent to Helium the RFID must sent to the Beagle bone the name within the transponder and display the information on screen, along with the other information required to search for a song.

**Audio output and Helium**

The best way to test the connect of the audio output and Helium is to first pick a song manually, and then test the output voltage of the different pins to see if the voltage is the same. The next part is to see if the song is playing as well as is designed to do. Use the equalizer and amplifier to change the volume and frequency of the output signal to make sure the song can be understood.

**6.2 Software Testing**

**6.2.1 Testing Environment (TE)**

Helium’s software spans over at least three different devices: the primary device, which plays the music, among many other tasks; the secondary device, responsible for processing the RFID tags; and a server, responsible for storing any music files and meta data that might not be able to fit on the primary device. Accordingly, there will be three testing environments for software.

**Personal Computer TE**

The first testing environment will be a PC; the bulk of the software will be tested in this environment; in particular, most of the unit testing will be performed using this device. Table 6.2 A, below, shows the specs of the PC which will be used.

|  |  |
| --- | --- |
| Primary OS | Windows 8.1 (64 bit) |
| Secondary OS | Ubuntu 13.10 |
| Manufacturer | Toshiba |
| Model | Satellite C855D |
| Processor | AMD E1-1200 APU with Radeon™ HD Graphics 1..40 GHz (x64 – based) |
| RAM | 4.00 GB |

*Table 6.2 A: Specs for PC testing environment.*

**BeagleBone Black TE**

The second testing environment will be the BeagleBone Black used as the MCU for the primary device. An LCD display will be attached to this device for testing purposes (see section 3.7.5). In addition, the BeagleBone Black has 4 LEDs (2 of which can be programmed); these will be programmed as needed for debugging bugs in the software. For some phases of testing, this device will be connected to the personal computer TE.

**MSP430 TE**

The third testing environment will be the MSP430 used as the MCU of the secondary device. For some phases of testing, this device will be connected to the personal computer TE.

**6.2.2 Unit Testing**

For all tests that must be performed on the BeagleBoard Black or the MSP430, a pin will be designated as the “test pin;” the MCU will be programmed to execute the block of code that is to be tested when an input is applied to the pin (~1.5 V, DC). For convenience, several “test pins” may be programmed to allow rapid testing of multiple blocks of code; unless multiple pins are needed for a test, these pins will all be referred to as the “test pin.” The test should be independent of the length of time that the input is applied to the test pin. An additional pin, “test reset pin” programmed to reset any tests when an input is applied to it.

**Communications Testing**

Tables 6.2 B – H describe some of the various tests that will be performed to verify the functionality of the Communication module.

|  |  |
| --- | --- |
| **Communications Test 1** |  |
| Objective | To query the MySql server from the primary device; to test the Communication and DatabaseInterface classes; to test the BeagleBone Black’s Wi-Fi unit. |
| Environment | BeagleBone Black |
| Summary | The BeagleBone Black will be programmed to execute a SQL query when a designated pin is active. An LED will be programmed to turn on if the response matches a predetermined value. The LED will be programmed to turn off if an input is applied to test reset pin. |
| Setup 1 | User programmable LED 1 will be programmed to turn on if the SQL query: *SELECT idUser FROM User WHERE userName = “test\_user”;* returns the integer ‘12’. |
| Setup 2 | The value for idUser for the User with userName == “test\_user” will be set to ‘12’. |
| Step 1 | A DC input of 1.5 V will be applied to test pin 1 for approximately 1 second |
| Step 2 (Observation) | Wait 2 seconds after removing the input to the test pin. |
| Step 3 | A DC input of 1.5 V will be applied to test reset pin. |
| Expected Results | Prior to step 1, LED 1 will not be lit; for the duration of the observation step, LED 1 will be lit; after step 3 is completed, the LED will no longer be lit. Behavior is undefined during steps 1 and 3. |

*Table 6.2 B: Song server SQL query test.*

|  |  |
| --- | --- |
| **Communications Test 2** |  |
| Objective | Same as “Communications Test 1.” |
| Environment | Same … |
| Summary | Same … |
| Setup 1 | Same … |
| Setup 2 | The value for idUser for the User with userName == “test\_user” will be set to ‘13’. |
| Step 1 – Step 3 | See … |
| Expected Results | LED 1 will remain unlit throughout the entire test. |

*Table 6.2 C: Song server SQL query test 2.*

|  |  |
| --- | --- |
| **Communications Test 3** | *Will be ran once using a home network, and a second time using UCF’s network.* |
| Objective | To query the WPS location server and test the accuracy of the GPS coordinates returned. |
| Environment | PC |
| Summary | The WPS location server will be queried while connected to a Wi-Fi network. The returned GPS coordinates will be looked up and verified for accuracy. |
| Setup | The location services module will be programmed to display the GPS coordinates on the screen of the PC. |
| Step 1 | The location services module will be executed. |
| Expected Results | GPS coordinates will be displayed on the screen; the displayed GPS coordinates will be within 1 mile of the PC’s location. |

*Table 6.2 D: Location server query test (PC).*

|  |  |
| --- | --- |
| **Communications Test 4** | *Will be ran once using a home network, and a second time using UCF’s network.* |
| Objective | To query the WPS location server; to test the accuracy of the GPS coordinates returned; and to test the BeagleBone Black’s Wi-Fi unit. |
| Environment | BeagleBone Black |
| Summary | The WPS location server will be queried while connected to a Wi-Fi network; an LED will be programmed to turn on if the response matches a predetermined value. The LED will be programmed to turn off if an input is applied to test reset pin. |
| Setup 1 | User programmable LED 1 will be programmed to turn on if the GPS coordinates match those that were found during “Communications Test 4.” |
| Step 1 | A DC input of 1.5 V will be applied to test pin 1 for approximately 1 second |
| Step 2 (Observation) | Wait 2 seconds after removing the input to the test pin. |
| Step 3 | A DC input of 1.5 V will be applied to test reset pin. |
| Expected Results | Prior to step 1, LED 1 will not be lit; for the duration of the observation step, LED 1 will be lit; after step 3 is completed, the LED will no longer be lit. Behavior is undefined during steps 1 and 3. |

*Table 6.2 E: Location server query test (BeagleBoard).*

|  |  |
| --- | --- |
| **Communications Test 5** | *Will be ran several times under various weather conditions (hopefully).* |
| Objective | To query the weather server. |
| Environment | PC |
| Summary | The weather server will be queried while connected to a Wi-Fi network using the GPS coordinates found in “Communications Test 3.” |
| Setup | The weather module will be programmed to display the most recent weather conditions on the screen of the PC. |
| Step 1 | Location.getWeather() will be executed. |
| Expected Results | The weather conditions printed to the screen will match any of the weather conditions that have been present up to 15 minutes prior to running the test. |

*Table 6.2 F: Weather server query test (PC).*

|  |  |
| --- | --- |
| **Communications Test 6** | *Will be ran several times under various weather conditions (hopefully).* |
| Objective | To query the weather server and to test the BeagleBone Black’s Wi-Fi unit. |
| Environment | BeagleBone Black |
| Summary | The weather server will be queried while connected to a Wi-Fi network; an LED will be programmed to turn on if the response matches a predetermined value. The LED will be programmed to turn off if an input is applied to test reset pin. |
| Setup | User programmable LED 1 will be programmed to turn on if the weather returned is “rainy”. |
| Step 1 | A DC input of 1.5 V will be applied to test pin 1 for approximately 1 second |
| Step 2 (Observation) | Wait 2 seconds after removing the input to the test pin. |
| Step 3 | A DC input of 1.5 V will be applied to test reset pin. |
| Expected Results | Prior to step 1, LED 1 will not be lit; for the duration of the observation step – and if it is raining – LED 1 will be lit, else the LED will remain unlit; after step 3 is completed, the LED will no longer be lit. Behavior is undefined during steps 1 and 3. |

*Table 6.2 G: Weather server query test (BeagleBoard).*

|  |  |
| --- | --- |
| **Communications Test 7** | *Will be ran several times for various users* |
| Objective | To test the user login system; the song selection algorithm; the mp3 decoder; the speakers; etc… |
| Environment | BeagleBoard Black |
| Summary | An RFID tag will be briefly held near the RFID scanner and the user to whom the tag belongs to will be logged in. |
| Setup | An RFID tag will be given an ID, that ID will be associated to a user ID; a User object for that user will be on the device and contain a non-empty *master\_playlist*. |
| Step 1 | An RFID tag is swiped near the RFID scanner. |
| Expected Results | Music from the master\_playlist of the user to which the tag is associated with will begin playing. |

*Table 6.2 H: User login test.*

**Song Selection Algorithm Testing**

Tables 6.2 I – K describe some of the various tests that will be performed to verify the functionality of the song selection algorithm.

|  |  |
| --- | --- |
| **Song Selection Algorithm Test 1** | *Will be ran for various users.* |
| Objective | To collect user data for further testing; to determine if the *ShuffledStack* module is performing as expected. |
| Environment | PC |
| Summary | A new user will be suggested songs for one hour and will decide for each song whether or not to skip that song. |
| Setup | A *master\_playlist* will be created for the user with at least 1 hour worth of songs (at least 2 different songs). |
| Step 1 | The algorithm will be ran for 1 hour. |
| Expected Results | The user will not hear any songs repeated twice in a row; a base skip rate, S0, will be determined for several users. |

*Table 6.2 I: Skip rates for new users.*

|  |  |
| --- | --- |
| **Song Selection Algorithm Test 2** | *Will be ran for various users.* |
| Objective | To determine the performance of the song selection algorithm after 1 hour worth of listening for each user. |
| Environment | PC |
| Summary | A user will be suggested songs for one hour and will decide for each song whether or not to skip that song. |
| Setup | Same as “Song Selection Algorithm Test 1.” |
| Step 1 | Same … |
| Expected Results | A new skip rate, S1, will be determined for several users. For each user, S1 will be less than S0. |

*Table 6.2 J: Skip rates for hour 2.*

|  |  |
| --- | --- |
| **Song Selection Algorithm Test 3** | *Will be ran for various users.* |
| Objective | To determine the performance of the song selection algorithm after 2 hours’ worth of listening for each user. |
| Environment | PC |
| Summary | Same as “Song Selection Algorithm Test 2.” |
| Setup | Same … |
| Step 1 | The algorithm will be ran for a total of at least 5 hours (across multiple sessions). |
| Expected Results | A new skip rate, S2, will be determined for several users. For each user, S2 will be less than S0 and S1. The skip rate for each user will trend downwards with time. |

*Table 6.2 K: Skip rates for hours 3+.*

**DeviceController Testing**

Tables 6.2 K – X describe some of the various tests that will be performed to verify the functionality of the *DeviceController*.

|  |  |
| --- | --- |
| **DeviceController Test 1** |  |
| Objective | To verify the functionality of the volumeUp() method. |
| Environment | BeagleBoard Black |
| Summary | The volume of the audio output should be noticeably increased when an input is applied to a test pin. This will also result in an increase in the output voltage. |
| Setup | The volumeUp() and volumeDown() methods will be programmed to execute when an input is applied to test pins 1 and 2, respectively. The device’s volume will be set to 0 initially. The reset pin will be programmed to reset the volume to 0. |
| Step 1 | A 1.5 V input will be applied to test pin 1. |
| Expected Results | The audio output from the speaker(s) will be noticeably louder. The output voltage will be 1/16th of the maximum output voltage. |

*Table 6.2 L: Volume increase from 0/8th volume to 1/8th volume.*

|  |  |
| --- | --- |
| **DeviceController Test 2** |  |
| Objective | To verify the functionality of the volumeUp() method. |
| Environment | BeagleBoard Black |
| Summary | Same as “DeviceController Test 1.” |
| Setup | Same as “DeviceController Test 1,” except that the initial volume will be at the 1/8th level (1/64th of the maximum output voltage). |
| Step 1 | A 1.5 V input will be applied to test pin 1. |
| Expected Results | The audio output from the speaker(s) will be noticeably louder. The output voltage will be 1/8th of the maximum output voltage. |

*Table 6.2 M: Volume increase from 1/8th volume to 2/8th volume.*

|  |  |
| --- | --- |
| **DeviceController Tests 3 – 8** |  |
| Objective | To verify the functionality of the volumeUp() method. |
| Environment | BeagleBoard Black |
| Summary | The volume of the audio output should be noticeably increased when an input is applied to a test pin. This will also result in an increase in the output voltage. The max volume of the device will not exceed 90 dB |
| Setup | Same as “DeviceController Test 1,” except that the initial volume will be at the M/8th level ((M/8)^2 of the maximum output voltage). Where M is the test number. |
| Step 1 | A 1.5 V input will be applied to test pin 1. |
| Expected Results | The audio output from the speaker(s) will be noticeably louder. The output voltage will be (M/8)^2 of the maximum output voltage. The volume of the device will not exceed 90 dB at any volume level. |

*Table 6.2 N: Volume increase from 2/8th volume to max volume, at 1/8th increments.*

|  |  |
| --- | --- |
| **DeviceController Test 9** |  |
| Objective | To verify the functionality of the volumeMute() method. |
| Environment | BeagleBoard Black |
| Summary | The volume of the audio output will be set from the 4/8th level to the 0/8th level; the previous volume level should be saved. After executing the volumeMute() method a second time, the volume will return to the 4/8th lvel. |
| Setup | The volumeMute() method will be programmed to execute when an input is applied to a test pin. The volume will be set to the 4/8th level. |
| Step 1 | A 1.5 V input will be applied to the test pin. |
| Step 2 (Observe) | Wait 5 seconds. |
| Step 3 | A 1.5 V input will be applied to the test pin. |
| Expected Results | During step 2, the volume of the audio output will be set to 0 and the music will continue to be decoded. After step 3, the volume will return to the 4/8th level; the music being played will have progressed 5 seconds. |

*Table 6.2 O: Toggle mute test.*

|  |  |
| --- | --- |
| **DeviceController Test 10** |  |
| Objective | To verify the functionality of the volumeDown() method. |
| Environment | BeagleBoard Black |
| Summary | The volume of the audio output should be noticeably decreased when an input is applied to a test pin. This will also result in a decrease in the output voltage. |
| Setup | The volumeUp() and volumeDown() methods will be programmed to execute when an input is applied to test pins 1 and 2, respectively. The device’s volume will be set to 100% initially. The reset pin will be programmed to reset the volume to 0. |
| Step 1 | A 1.5 V input will be applied to test pin 2. |
| Expected Results | The audio output from the speaker(s) will be noticeably quieter. The output voltage will be 15/16th of the maximum output voltage. |

*Table 6.2 P: Volume decrease from 8/8th volume to 7/8th volume.*

|  |  |
| --- | --- |
| **DeviceController Test 11** |  |
| Objective | To verify the functionality of the volumeDown() method. |
| Environment | BeagleBoard Black |
| Summary | Same as “DeviceController Test 10.” |
| Setup | Same as “DeviceController Test 10,” except that the initial volume will be at the 7/8th level |
| Step 1 | A 1.5 V input will be applied to test pin 2. |
| Expected Results | The audio output from the speaker(s) will be noticeably quieter. The output voltage will be 7/8th of the maximum output voltage. |

*Table 6.2 Q: Volume decrease from 7/8th volume to 6/8th volume.*

|  |  |
| --- | --- |
| **DeviceController Tests 12 – 17** |  |
| Objective | To verify the functionality of the volumeDown() method. |
| Environment | BeagleBoard Black |
| Summary | The volume of the audio output should be noticeably decreased when an input is applied to a test pin. This will also result in a decrease in the output voltage. |
| Setup | Same as “DeviceController Test 10,” except that the initial volume will be at the M/8th level ((M/8)^2 of the maximum output voltage). Where M is the test number minus 9 |
| Step 1 | A 1.5 V input will be applied to test pin 2. |
| Expected Results | The audio output from the speaker(s) will be noticeably quieter. |

*Table 6.2 R: Volume decrease from 5/8th volume to min volume, at 1/8th increments.*

|  |  |
| --- | --- |
| **DeviceController Tests 18 – 32** |  |
| Objective | To verify the functionality of the increaseBandGain(band\_index) and decreaseBandGain(band\_index) method. |
| Environment | BeagleBoard Black |
| Summary | The band gain of the audio output should be noticeably increased and decreased when an input is applied to the test pins. This will also result in an increase and decrease in the output voltage. |
| Setup | The increaseBandGain(band\_index) and decreaseBandGain(band\_index) methods will be programmed to execute when an input is applied and held to test pin 1 and then input is applied to pin 2 the band gain would increase. The opposite for band gain decrease. When input is applied to Pin 2 and held then Pin1 has an input that is applied the band gain would decrease. The device’s volume will be set to 8 initially. The device’s band gain will be set to 0 initially. Increase the band gain by an increment of 1/8 and continuing to 8/8. After reaching max band gain decrease the gain until reaching 0/8. |
| Step 1 | A 1.5 V input will be applied to test pin 1 and 2. |
| Expected Results | The audio output from the speaker(s) will be noticeably louder and quieter for the appropriate tests. The volume of the device will not exceed 90 dB at any volume level. |

*Table 6.2 S:Band gain increase and decrease in 1/8 increments.*

|  |  |
| --- | --- |
| **DeviceController Tests 33 – 40** |  |
| Objective | To verify the functionality of the defaultBands() method. |
| Environment | BeagleBoard Black |
| Summary | The gain of the audio output should be set to 1. This could result in an increase or decrease in audio output depending on the initial band gain. The output voltage would also increase or decrease depending on the initial band gain. |
| Setup | Set the band gain to 0/8. The defaultBands() method will be implemented by pressing and holding down Pin 1 and Pin 2 for more than 5 seconds. After doing this once at 0/8, implement increaseBandGain(band\_index) so that the band gain is 1/8. Now again implement defaultBands(). Increase the gain until max gain level 8/8 and after each increase implement defaultBands(). |
| Step 1 | A 1.5 V input will be applied to test pin 1 and 2. |
| Expected Results | At band gain of 0/8 an increase in volume should occur. At band gain of 1/8 there should be no change. However, for band gains of higher than 1/8 there will be a noticeable decrease in audio output |

*Table 6.2 T:defaultBands sets band gain to 1.*

**6.2.3 Software Integration Testing**

Once all the individual modules have been tested, the Init module, the main module, will be tested. Since most of the software involves controlling hardware, those methods will be temporarily replaced with methods that will instead print to the screen. This way, the group can test the entire software system before performing a full system test. Tables 6.2 U – V describe some of the various tests that will be performed to verify that the software system works as a whole, minus the hardware.

|  |  |
| --- | --- |
| **Software Integration Test 1** |  |
| Objective | To test the process of a user logging in. |
| Environment | PC |
| Summary | This test will simulate a user logging in, causing the system to wake up and music to start playing |
| Setup | To simulate *user\_*1 logging in, the *Communication* module will be programmed to interpret a key press as a message being set from the RFIDSystem. Print statements will be added to each thread to indicate when they are running some task. |
| Step 1 | Begin simulation. |
| Step 2 (Observe) | Wait 5 seconds. |
| Step 3 | Press any key. |
| Expected Results | The console’s output should look like this in steps 1 and 2“  Init, check Communication: RFIDSystem  No messages  Init, check Communication: mobile  No messages”  This should be repeated several times.  After a key is pressed, the console’s output should look like this “  Init, check Communication: RFIDSystem  User\_1 has logged in!  Init, start DeviceController thread  Init, start MediaController thread  Init, users.active\_user = user\_1  Init, playlist = users.getUser(user\_1).getDefaultPlaylist()  Init, media.setPlaylist(playlist)  Init, media.play(playlist[0])” |

*Table 6.2 U: Simulation of a user logging in.*

|  |  |
| --- | --- |
| **Software Integration Test 2** |  |
| Objective | To test the process of a user starting a smart playlist and to verify that the user’s listening habits are being uploaded to a SQL server. |
| Environment | PC |
| Summary | This test will simulate a user creating a smart playlist, and making several selections to skip songs. |
| Setup | User\_1 will be logged in. To simulate a user creating a smart playlist, the Communications module will be programmed to interpret the space bar as a command to create a smart playlist. It will also be programmed to interpret the ‘s’ key as a command to skip. All threads will already be active. |
| Step 1 | Begin simulation. |
| Step 2 | Press the space bar. |
| Step 3 | Listen to a song. |
| Step 4 | Press the ‘s’ key. |
| Expected Results | The console’s output should look like this in steps 1,2 and 3  “Init, check Communication: mobile  Create smart playlist (user\_1)  Init, playlist = users.getUser(user\_1).createSmartPlaylist()  Init, media.setPlaylist(playlist)  Init, media.play(playlist[0])”  MediaController, now playing: “Aging Astronauts”  After the song “Aging Astronauts” has finished, and during step 4, the consoles output will look like this:  “MediaController, now playing: “Ladder to Heaven”  Init, check Communications: mobile  Skip song  Init, media.skip()  MediaController, now playing: “Age of Backwards”” |

*Table 6.2 V: Simulation of a user interacting with a smart playlist.*

**6.3 Full System Testing**

For final testing there will be two ways of testing each goal. One is from voice commands and the other is from an app that will be on a device. Tables 6.3 A – X, below, list some of the various tests that will be performed to verify the functionality of the full system.

**System voice command testing**

|  |  |
| --- | --- |
| **System Test 1** |  |
| Objective | To play music when user says “Play” |
| Environment | Full System |
| Summary | Once user says “Play” and is in the range of the microphone. The device will play music |
| Setup | A person will already be scanned in, the device will be on and the device will be listening to speech |
| Step 1 | Say “Play” in the range of the device |
| Expected Results | Music will play through the speakers a few seconds after the user says “Play.” |

*Table 6.3 A: Voice command “Play.”*

|  |  |
| --- | --- |
| **System Test 2** |  |
| Objective | To pause music when user says “Pause” |
| Environment | Full System |
| Summary | Once user says “Pause” and is in the range of the microphone. The device will stop playing music |
| Setup | A person will already be scanned in, the device will be on and the device will be listening to speech |
| Step 1 | Say “Pause” in the range of the device |
| Expected Results | Music will stop playing after processing the word “Pause” |

*Table 6.3 B: Voice command “Pause.”*

|  |  |
| --- | --- |
| **System Test 3-5** |  |
| Objective | Music will play based off the genre said by the user |
| Environment | Full System |
| Summary | Once user says “Play (Genre)” and is in the range of the microphone. The device will play music from the appropriate playlist |
| Setup | A person will already be scanned in, the device will be on and the device will be listening to speech |
| Step 1 | Say “Play (Genre)” in the range of the device  Genres being  Rock, Hip Hop, Classical |
| Expected Results | Music will play the appropriate music based off the genre said by the user |

*Table 6.3 C: Voice command “Play (Genre).”*

|  |  |
| --- | --- |
| **System Test 6** |  |
| Objective | To make the music louder played by the device |
| Environment | Full System |
| Summary | Once user says “Louder” and is in the range of the microphone. The device will execute the “volumeUp()” method. |
| Setup | A person will already be scanned in, the device will be on and the device will be listening to speech |
| Step 1 | Say “Louder” in the range of the device |
| Expected Results | From the perspective of the user(s) the music will be louder after saying “Louder” |

*Table 6.3 D: Voice command “Louder.”*

|  |  |
| --- | --- |
| **System Test 7** |  |
| Objective | To make the music volume decrease played by the device |
| Environment | Full System |
| Summary | Once user says “Quieter” and is in the range of the microphone. The device will execute the “volumeDown()” method. |
| Setup | A person will already be scanned in, the device will be on and the device will be listening to speech |
| Step 1 | Say “Quieter” in the range of the device |
| Expected Results | From the perspective of the user(s) the music will be quieter after saying “Quieter” |

*Table 6.3 E: Voice command “Quieter.”*

|  |  |
| --- | --- |
| **System Test 8** |  |
| Objective | To test the weather module |
| Environment | Full System |
| Summary | The device will be tested while it is raining. An LED will light up to indicate that the current weather is “rainy.” |
| Setup | A person will already be scanned in, the device will be on and it will have been raining for at least 15 minutes. |
| Step 1 | Observe |
| Expected Results | An LED will turn on. This will mean that the weather is rainy; determining the effects that this will have on the song selection algorithm will require looking at the user’s data on the SQL database. |

*Table 6.3 F: Weather check.*

|  |  |
| --- | --- |
| **System Test 9** |  |
| Objective | To test the weather module |
| Environment | Full System |
| Summary | The device will be tested while it is not raining. The weather indication LED will not light up to indicate that the current weather is “rainy.” |
| Setup | A person will already be scanned in, the device will be on and it will not have been raining for at least 15 minutes. |
| Step 1 | Observe |
| Expected Results | The LED will remain unlit. This will mean that the weather is not rainy; determining the effects that this will have on the song selection algorithm will require looking at the user’s data on the SQL database. |

*Table 6.3 G: Weather check.*

**Scanner Testing**

|  |  |
| --- | --- |
| **System Test 10** |  |
| Objective | To make sure the scanner works |
| Environment | Full System |
| Summary | The user will scan in and music will play |
| Setup | The device will be on and waiting for a person to scan in. An RFID tag will be associated with “user\_1.” |
| Step 1 | User\_1’s tag will be scanned using scanner |
| Expected Results | Music will play from user\_1’s default playlist. |

*Table 6.3 H: User login test.*

**App Testing**

|  |  |
| --- | --- |
| **System Test 11** |  |
| Objective | To play music when user gives a play input from the app |
| Environment | Full System |
| Summary | Once user pushes play. The device will play music |
| Setup | A person will already be scanned in, the device will be on and the device will be receiving from the app. |
| Step 1 | Push play on the app in range of the device |
| Expected Results | Music will play through the speakers a few seconds after the user pushing play |

*Table 6.3 I: Mobile control test; “play.”*

|  |  |
| --- | --- |
| **System Test 12** |  |
| Objective | To pause music when user gives a pause input from the app |
| Environment | Full System |
| Summary | Once user pushes pause. The device will pause music |
| Setup | A person will already be scanned in, the device will be on and the device will be receiving from the app. |
| Step 1 | Push pause on the app in range of the device |
| Expected Results | Music will play through the speakers a few seconds after the user pushes pause |

*Table 6.3 J: Mobile control test; “pause.”*

|  |  |
| --- | --- |
| **System Test 13** |  |
| Objective | To play music when user gives a mobile command to generate a smart playlist. |
| Environment | Full System |
| Summary | Once user a pushes button to create a new Smart Playlist, the device will begin selecting songs from that user’s Master Playlist based on time of day, weather, play density of various musical attributes, and various other attributes. |
| Setup | A person will already be scanned in, the device will be on and the device will have established a Bluetooth connection with a mobile device. |
| Step 1 | User pushes a button to create a new Smart Playlist. |
| Expected Results | Music will play through the speakers a few seconds after the user pushes issues the command. The user will enjoy the song selections made by the algorithm. |

*Table 6.3 K: Mobile control test; “Smart Playlist.”*

|  |  |
| --- | --- |
| **System Test 14** |  |
| Objective | To make the music louder played by the device |
| Environment | Full System |
| Summary | Once user pushes Volume up. The device will increase the volume |
| Setup | A person will already be scanned in, the device will be on and the device will be receiving from the app. |
| Step 1 | Push volume up on the app in range of the device |
| Expected Results | From the perspective of the user(s) the music will be louder |

*Table 6.3 L: Mobile control test; “volume up.”*

|  |  |
| --- | --- |
| **System Test 15** |  |
| Objective | To make the music quieter played by the device |
| Environment | Full System |
| Summary | Once user pushes Volume down. The device will decrease the volume |
| Setup | A person will already be scanned in, the device will be on and the device will be receiving from the app. |
| Step 1 | Push volume down on the app in range of the device |
| Expected Results | From the perspective of the user(s) the music will be queiter |

*Table 6.3 M: Mobile control test; “volume down.”*

|  |  |
| --- | --- |
| **System Test 16** |  |
| Objective | To measure the heat given off by the device after many hours of operation. |
| Environment | Full System |
| Summary | The device will remain in continuous, full power operation, for 4 hours. The temperature of the device will be measured every hour. |
| Setup | A user will scan into the device and monitor it for 4 hours. |
| Step 1 | After 1 hour, measure the temperature near the device. |
| Step 2 | After an additional hour, measure the temperature near the device. |
| Step 3 | After an additional hour, measure the temperature near the device. |
| Step 4 | After an additional hour, measure the temperature near the device. |
| Expected Results | At all times, the device will be at least 10% cooler than the BeagleBone Black’s maximum operating temperature. No part of the device will be painful to touch. |

*Table 6.3 M: Mobile control test; “volume down.”*

**Appendices**

**Permission for use of EM diagrams/figues**

---------- Forwarded message ----------  
From: **Jakusovszky, Mark** <[Mark.Jakusovszky@emmicro-us.com](mailto:Mark.Jakusovszky@emmicro-us.com)>  
Date: Mon, Dec 1, 2014 at 1:09 PM  
Subject: Approval for use of EM diagrams/figures  
To: "[goris3123@gmail.com](mailto:goris3123@gmail.com)" <[goris3123@gmail.com](mailto:goris3123@gmail.com)>  
Cc: "Schmick, Donna" <[Donna.Schmick@emmicro-us.com](mailto:Donna.Schmick@emmicro-us.com)>  
  
  
Giovanni -  
Please let me know what you would like to use, and I will help you get approval.  
  
Thanks,  
Mark (J) Jakusovszky  
US Sales & Marketing Manager  
EM Microelectronic-US, Inc.  
5475 Mark Dabling Blvd. Suite 200  
Colorado Springs, CO 80918  
Office:  [719-598-9224 x229](tel:719-598-9224%20x229)  
Mobile: [719-337-3179](tel:719-337-3179)  
[markj@emmicro-us.com](mailto:markj@emmicro-us.com)  
[www.emmicro-us.com](http://www.emmicro-us.com/)  
Makers of EM BLE Beacons and SENtral Sensor Fusion Co-Processors  
  
PLUS: Get a FREE Exhibits-Plus Pass for CES 2015 HERE!  
  
  
Thanks,  
Mark J.  
-----Original Message-----  
From: [website@emmicroelectronic.com](mailto:website@emmicroelectronic.com) [mailto:[website@emmicroelectronic.com](mailto:website@emmicroelectronic.com)]  
Sent: jeudi 27 novembre 2014 22:12  
To: EMM, Info  
Subject: Giovanni  
  
Submitted on Thursday, 27. November 2014 - 22:11 Submitted by anonymous user: [193.222.87.134] Submitted values are:  
  
First Name: Giovanni  
Last Name: Goris  
Address: 13079 Ruidosa loop  
ZIP: 32837  
City: orlando  
Country: United States  
Company: University of central florida  
Phone: 4074929485  
E-Mail: [goris3123@gmail.com](mailto:goris3123@gmail.com)  
Message:  
Hello,  
  
I am a student UCF during my senior design I semester.  
I would like to know who i could contact to get permission if i want to use a figure for my paper.  
  
  
The results of this submission may be viewed at:  
<http://www.emmicroelectronic.com/node/2/submission/368>  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
This e-mail message is intended only for the addressee(s) and contains  
information which may be confidential. If you are not the intended  
recipient please do not read, save, forward, disclose or copy the contents  
of this e-mail. If this e-mail has been sent to you in error, please delete this  
e-mail and any copies or links to this e-mail completely and immediately  
from your system. We also like to inform you that communication via e-mail  
over the Internet is insecure because third parties may have the possibility  
to access and manipulate e-mails.  
  
Any views expressed in this message are those of the individual sender,  
except where the sender specifically states them to be the views of  
The Swatch Group Ltd.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Permission for TI diagrams/figures**

TI grants permission for personal and non-commercial use here.

http://www.ti.com/corp/docs/legal/copyright.shtml