Smartphone Controlled Jukebox

ECE 445 Fall 2015
Design Review
James Lang, Nikhil Sancheti, Sam Sagan
TA: Eric Clark
September 28, 2015

Contents

1. Introduction

1.1 Statement of Purpose

Jukeboxes used to be a staple of the social scene in bars. Our Smartphone-Controlled Jukebox will modernize the old-school device and create a social entertainment system for both the commercial (bar) and home application. Our team is committed to improving the experience of guests and hosts in bars and at home.

1.2 Goals

- Modernize the jukebox concept
- Provide a fun and enjoyable social entertainment system with high quality audio suitable for a bar/dorm.

1.3 Functions

- Streams and selects songs from the Spotify API
- Removes the need for the user login to the Spotify API and makes the owner the main controller for the Spotify API
- Remote Management of the Song Queue from the owner App.
- Payments types of bitcoins, venmo, credit/debit cards that adds a new revenue stream to the owner
- Seamless and enjoyable user-interface via iPhone app
- Visually appealing LED lit package responding to ambient light and the beat of the music
- Affordable jukebox with high quality audio output suitable for audiophiles

1.4 Novelty

Projects to produce high quality audio output by connecting a better DAC to the Raspberry Pi have been done before. However, these systems were built strictly for home application and require some technical knowledge to assemble. We intend to build a plug-and-play system for both home and commercial settings. Furthermore, the similar projects mentioned are no longer being sold, and used versions do not take advantage of the capabilities of the newest Raspberry Pi. We believe that with the newest Raspberry Pi and a better DAC, we can produce quantitatively better audio than has previously been available on these types of systems.

2. Design

2.1. Block Diagram

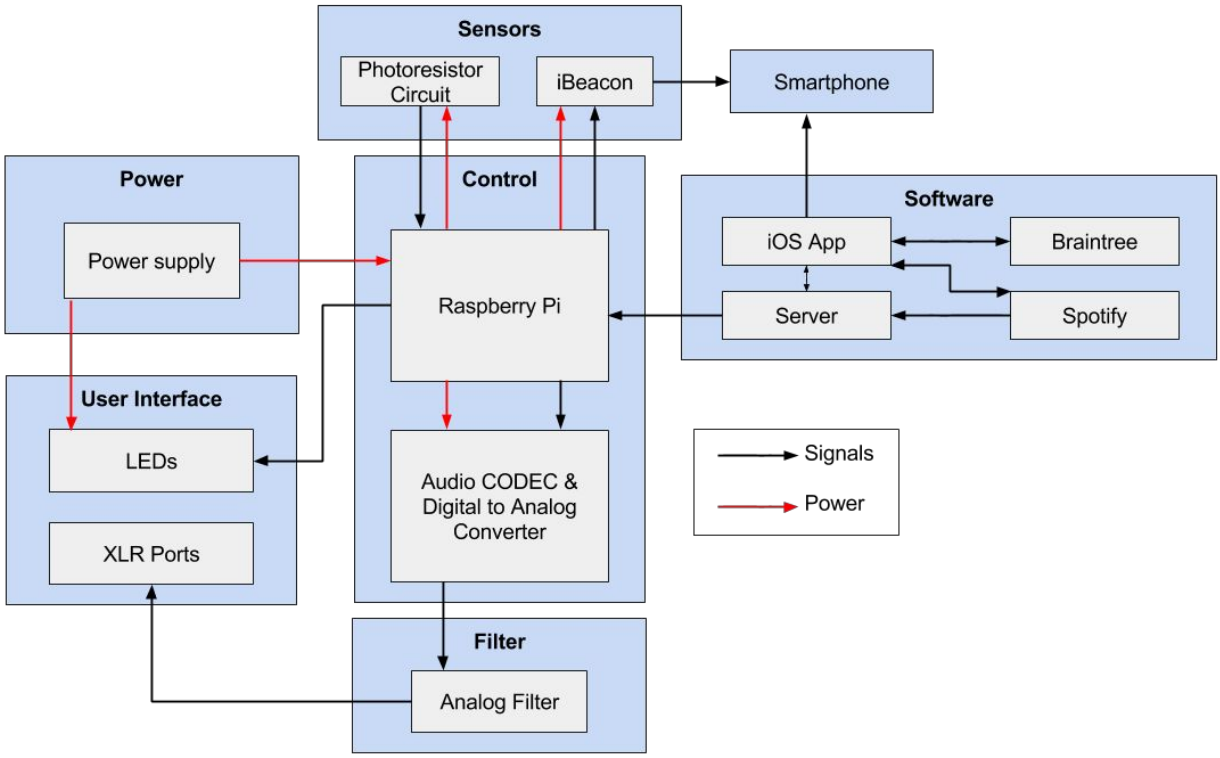


Figure 1. Top-level block diagram with signal and power flow

2.2. Block Descriptions

User Interface Module: This module takes input data such as LED controls from the Raspberry Pi and stereo audio from the CODEC. LEDs and Audio-out ports give the user access to these inputs.

LEDs: Individually addressable LEDs embedded in the Jukebox's housing are sent control signals by the Raspberry Pi. The LEDs will cycle through two modes of visual effects. The first is a (inherently pixelated) representation of the album artwork. The second is an abstract color pattern.

Audio-out Port: Stereo XLR output provides professional quality audio. We will use an unbalanced configuration; grounding the negative third pin of the XLR port.

Control Module: This module is the heart of our design. It takes streaming audio from Spotify, serializes it, and then converts from digital to analog to be heard by the user. Filtering, isolation circuitry, and AC coupling are used to decrease noise.

Raspberry Pi 2: A SoC with an ARM processor, running Raspbian OS. It takes streaming audio from Spotify over a Wi-Fi connection and serializes it to be sent to the CODEC. The RPi also processes input from the photoresistor circuit and sets the color and intensity of the LEDs. The iBeacon is used to broadcast and send identifier to nearby iPhones. Wired connections are outlined in the following table.

Name	Pin #	Function	Connection
PCM_CLK	J8-12	Bit Clock	CODEC SCLK
PCM_FS	J8-35	Frame Clock	CODEC LRCK
PCM_DOUT	J8-40	Serial Audio Data Output	CODEC SDIN
MOSI(SPI)	J8-19	LED Data Output	LED DI
SCLK(SPI)	J8-23	LED Serial Clock	LED CI

Audio CODEC: Decodes and converts the serial, digital audio signal of the songs from the RPi to analog. This high quality, stereo analog signal is sent to the User Interface Module, so it can be accessed through the audio-out ports. We use a Cirrus Logic CS4272 audio codec. We will implement isolation, low-pass filtering, AC coupling, and various control circuits around the CODEC IC. The CODEC has an external crystal to create the master clock signal. We will implement the CODEC's driver on the Raspberry Pi. This will generate necessary channel and bit clocks from the master clock. The driver will stream the serial audio in time with the bit clock. We describe the function and connection of the CODEC's pins in the following table.

Name	#	Function	Connection
XTO, XTI	1,2	For optional XTal Osc ckt	Crystal and capcitors as defined on CS4272 datasheet
MCLK	3	Master clock output	To a RaspPi GPIO pin
LRCK	4	Left/Right clock input	From RaspPi pin J8-35
SCLK	5	Serial Clock input	From RaspPi pin J8-12
SDOUT (M/S)	6	Serial Audio Data Output – sets slave mode if left unconnected	Unconnected
SDIN	7	Serial Audio Data Input in 2's complement format	RaspPi pin J8-40

DGND	8	Digital Ground input	Grounded
VD	9	Digital Power input	3.3 V (3.1,5.25) – ckt on CS4272 datasheet
VL	10	Positive Logic Power input	3.3 V (2.37,5.25) – ckt on CS 4272 datasheet
M0	11	Speed mode selector bits M1M0 input	00 for single speed
M1	12	“”	“”
I2S/LJ	13	Serial Audio Interface Select input	10k Pullup to VL for I2S
RST	14	Reset input - low power state when driven low	Passively high input from RaspPi GPIO pin
VCOM	15	Common mode voltage output	Ckt on CS4272 datasheet
AINA- AINA+ AINB+ AINB-	16 17 18 19	Differential Analog Input	Unconnected
VA	20	Analog power input	5 +/-.25 V – ckt on CS4272 datasheet
AGND	21	Analog ground input	Ckt on CS4272 datasheet
FILT+	22	Positive Voltage reference output	Ckt on CS4272 datasheet
AMUTEC	23	Channel A Mute control output grounds analog out A when high	Ckt on CS4272 datasheet
AOUTA+ AOUTA- AOUTB+ AOUTB-	24 25 26 27	Differential Analog Audio output	Ckt on CS4272 datasheet for filtering and mute control
BMUTEC	28	Channel B Mute control output grounds analog out B when high	Ckt on CS4272 datasheet

Filter:
The CS4272 is a high end IC with great specs, but it requires external filtering. Both channels have two differential analog outputs that need filtering. To maximize flatness of the frequency response in the audible range, we use a low-pass 3rd-order, active Butterworth filter. We eliminate DC components with AC coupling capacitors at the filter output. This filter was designed to have less than 1dB of attenuation throughout the audible frequency range 20Hz to 20kHz.

Software:
iOS App

The iOS user app will be the user's main point of interaction with the jukebox. The main functions of the app is to

- Search for the iBeacon
- Select the jukebox
- Search for songs from the Spotify API
- Select a song
- Pay to play the song
- See the player queue

Braintree

Braintree API was chosen for payments. It has the flexibility to accept payments from different platforms including PayPal, Bitcoin, Venmo, Apple Pay, Android Pay and credit cards. This will need not only an addition on the client but also on the server. The purpose of this is to authenticate payments and set the amount to pay for the service. It accepts 'nonces' which represent different types of payments such as debit cards, bitcoins,etc and then completes the transaction. It sends a token back to complete the transaction and tells the client if the payment processed or failed.

Spotify

The Spotify API provides high quality audio via its API's. It helps search for the songs and stream the songs. A premium account is needed by the owner to stream songs.

Server

The server functions to provide user sessions, handle jukebox sessions, stream songs, and finish payment loop.

Sensors:
Our sensors allow our Jukebox to respond to changes in ambient lighting and proximity of users.

iBeacon

The iBeacon has the purpose of creating a way to uniquely identify a jukebox with a unique major and minor as per Apple specifications. The UUID specified gives the family of the product(jukeboxes)

and the major and minor values help identify the exact jukebox. The iBeacon also provides RSSI values that can be captured from scanning for the iBeacon from the app which gives a better feel for distance to the Jukebox. It can add capabilities for social interactions in the beacon range.

Photosensor

This circuit produces a logical high when the ambient light is high. Conversely, it produces a low when there is little to no light. A photoresistor is the sensor that reads ambient light levels. Its resistance changes with ambient light. It is important that the circuit produces logical values within the Raspberry Pi's thresholds of .8V and 1.3V when exposed to natural light and dark indoor lighting.

Power:

The power requirements are important to keep both the Raspberry Pi and the LED's fully functional. Both have a voltage level of 5V but the LED's need 3.5A of current for the whole strip and the Raspberry Pi needs 1A of current. These are the two main components to be powered. The power circuit is built using an AC/DC converter along with four linear regulators. The LEDs will be divided and power in 3 separate strips to allow for lower amperage per line. One linear regulator is rated at 5V 1A and the other three are rated at 5V 3A. The AC/DC converter gives an output of 5V 10A which makes it a suitable source of powering the circuit.

2.3 Schematics and Flowcharts

Our audio processing circuitry consists of Cirrus Logic CS4272 audio codec with internal digital-to-analog converter (DAC). The IC pinout with connections, the analog low-pass filter, and mute circuitry are included in figures 2-4 respectively. Note the system-level blocks for the analog low-pass filter and mute circuitry in the top-level schematic.

The codec and DAC converts a serial audio input into stereo analog output. The analog output is low-pass filtered and AC coupled; essentially producing band-pass filtering on the audio output at the extents of the audible range (20Hz - 20kHz). The codec outputs a mute signal that is usually held high; when the codec outputs a low mute signal, the audio output on that respective channel is grounded through a BJT; muting the audio.

Filter design notes:

- Output impedance of CS4272 analog output pins are about 3kOhms, thus we use a 5.11k resistor at the input to the filter to avoid over-loading the the IC.
- The 3rd order Butterworth filter consists of a 1st-order Butterworth primary stage and a 2nd-order Butterworth secondary stage; both in the Sallen-Key configuration.
- We set the 3dB cutoff frequency to 24kHz to ensure that the amplitude response at 20kHz is no more than .99dB down.
- Design equations are as follows:
    - 1st stage: fc=1/(2*Pi*R*C)

- 2nd stage: fc=1/(2*Pi*Sqrt(R1*R2*C1*C2)) and C1(R1+R2)=Sqrt(Ra*R2*C1*C2)
- We set R1 to 5.11kOhms and C1 to 1nF and solve for the rest.

Simulations for the low-pass filtering and AC coupling frequency response and the mute signal's function are included in the simulations section 2.4.

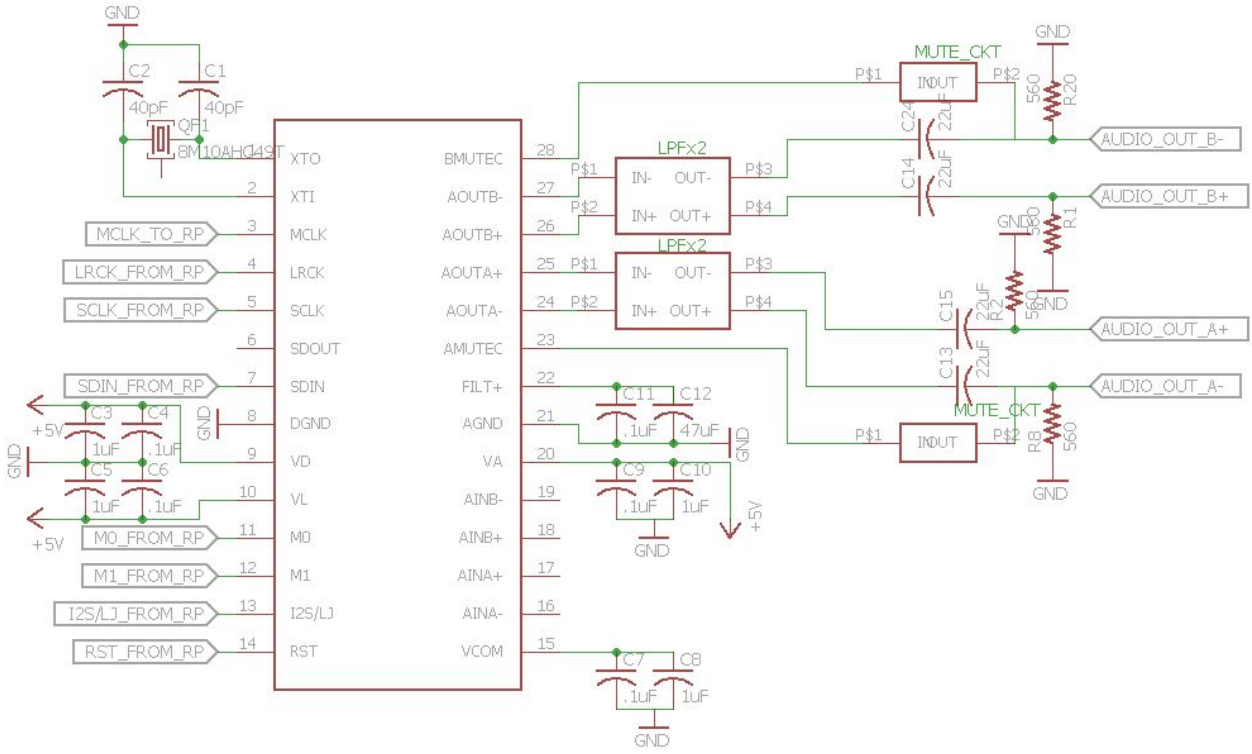


Figure 2. Top level audio processing circuitry

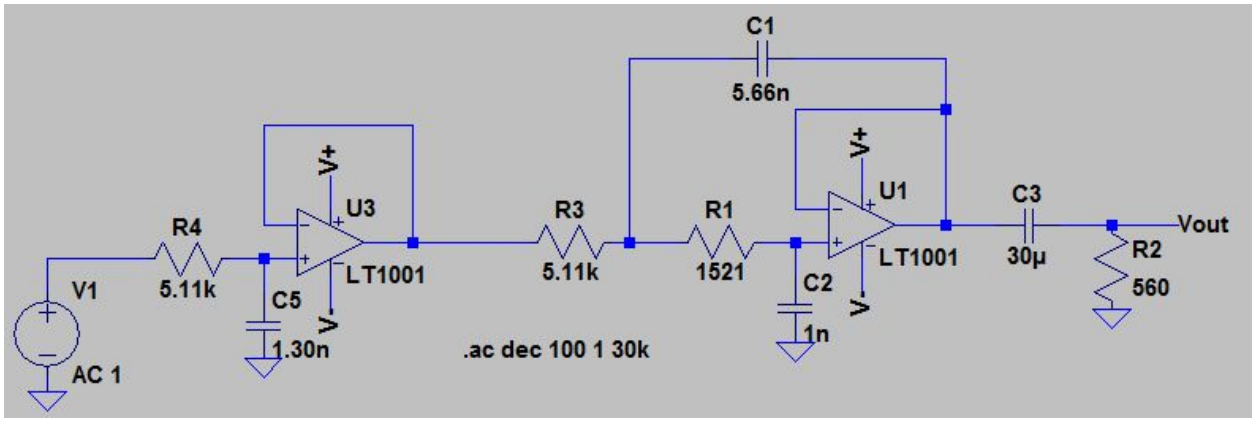


Figure 3. Analog low-pass filter and AC coupling at audio output

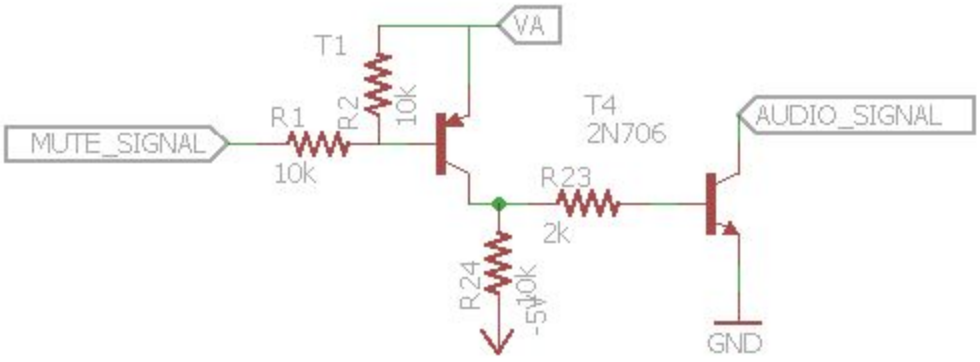


Figure 4. Mute circuitry grounds audio output when MUTE_SIGNAL is low

Our Jukebox responds to ambient light to turn on or off the LEDs. We designed a photosensor circuit with a photoresistor that produces a logic high in bright light and a low in darkness. Additionally, the LEDs and Raspberry Pi need external power. The Raspberry Pi has standard adapters, but we cannot power the LEDs through the Raspberry Pi due to the amperage requirements. Thus, we created a power supply circuit with an AC/DC converter and linear regulators to power the Raspberry Pi and 3 LED strips.

The schematics for the photosensor and power supply are included here in figures 5 and 6, respectively. The photosensor circuit is simulated in section 2.4.

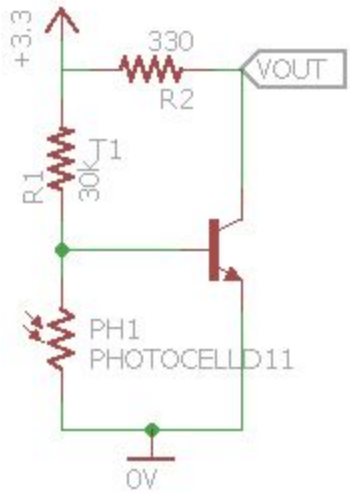


Figure 5. Photosensor circuit outputs a high in full light and low in darkness

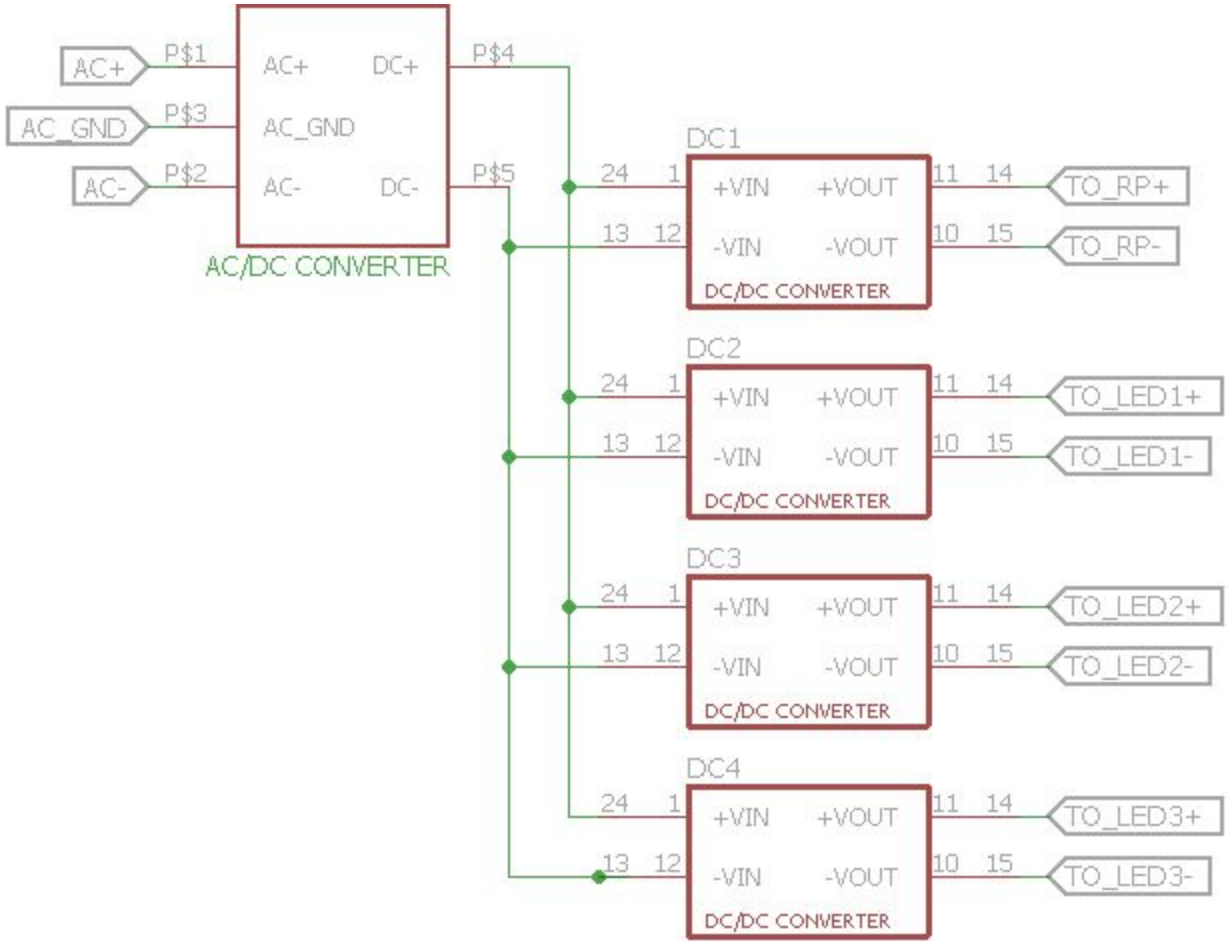


Figure 6. Power supply for Raspberry Pi and LEDs

The power supply has a AC/DC converter that takes a wall plug input and gives out a DC 5V output at 10Amps. The Raspberry Pi needs 1A at 5V. The LED strips need 3Amps at 5V each. The are placed in parallel. The DC/DC converter is a linear regulator that gives out the outputs needed for the raspberry pi and the LED's.

2.4 Simulations and Models

Our Jukebox produces high-quality audio output. This means that it is low-noise and has a highly consistent (flat) frequency response over the entirety of the audible range (20Hz - 20kHz). Decoupling circuitry limits the noise in digital-to-analog conversion. At the analog output, a low-pass filter and AC coupling capacitor eliminate frequency components from outside of the audible range and maintain a consistent (+/- .5 dB) frequency response within it. Following the simulation results, we model the behavior at DC and at high frequency.

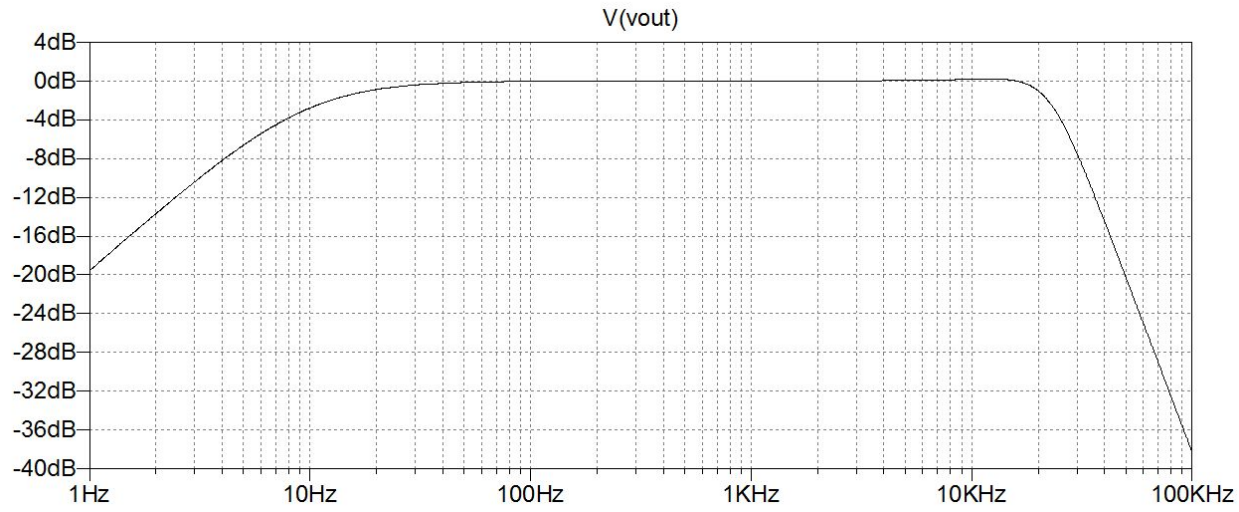


Figure 7. Output analog filter frequency response

Our photosensor circuit must output a logical high when ambient lights are on and a logical low when lights are off. The photoresistor changes its resistance given the ambient light. In full light, its resistance is between 5k-10k Ohms. In darkness it goes up to 200 kOhms. Thus, these regions must have logical high and low output consistent with the Raspberry Pi's thresholds of .8V and 1.3V. This simulation demonstrates that our photosensor circuit meets these requirements in theory.

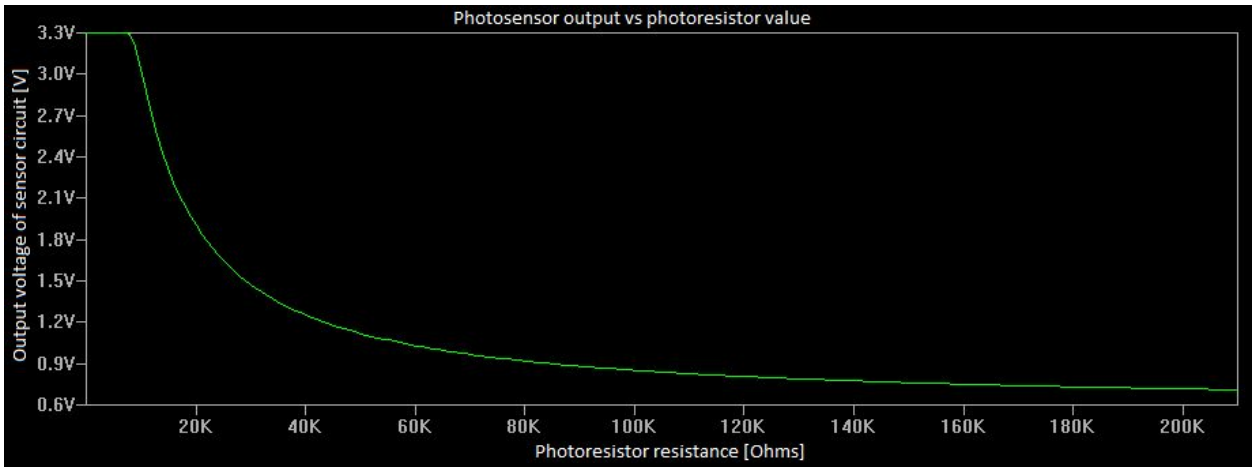


Figure 10. Photosensor output given photoresistor values which change with ambient light level

The mute circuitry recommended by Cirrus Logic grounds the analog output whenever unwanted (non-audio) signals are propagating through the codec. This eliminates pops and crackles that occur during state transitions within the codec. In this simulation, we give the audio output the arbitrary DC value of 2.5 V meant to represent the oscillating audio output. When the mute signal (active low) is high, the DC signal is present on the output channel, but when the mute signal is low, the analog output is grounded.

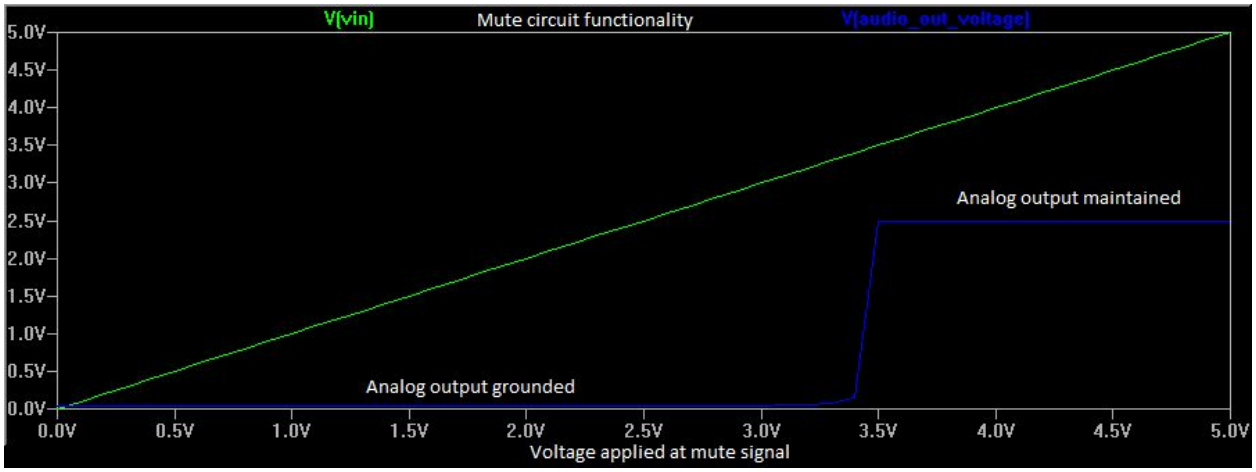


Figure 11. Mute circuit functionality

2.5 Smartphone App UI and Software Flows

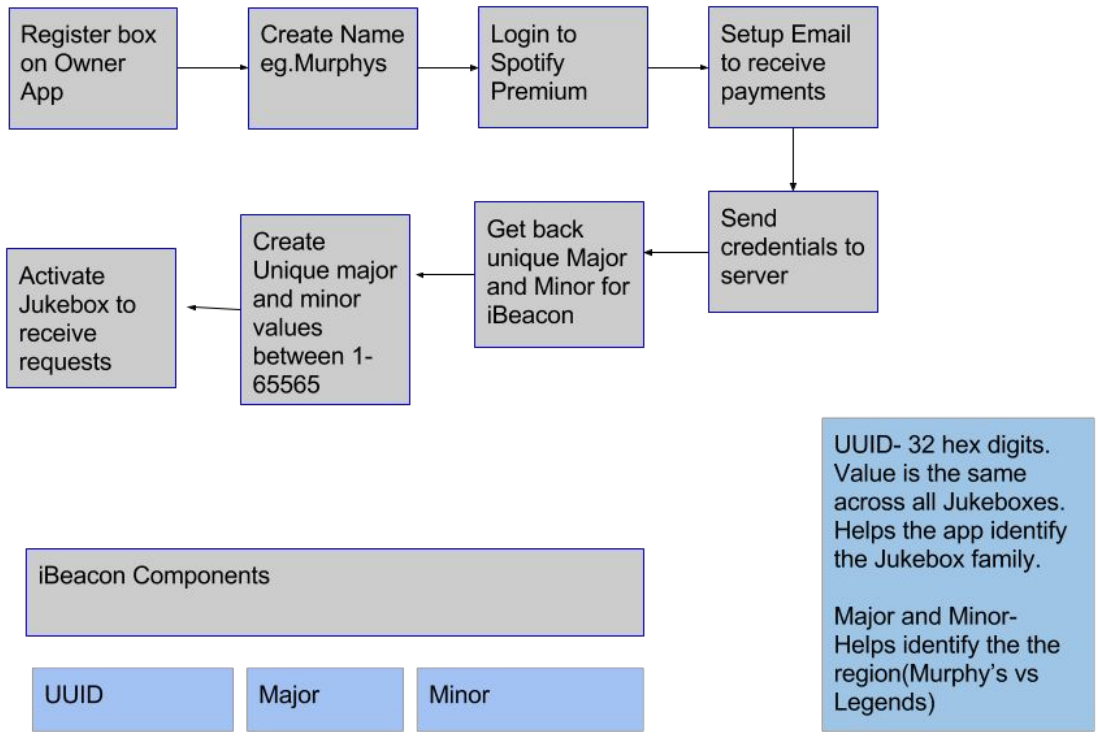


Figure 12. Authentication flow for the Jukebox

The Jukebox authentication is essential to making sure that the Jukebox is unique and has a Spotify premium account attached to it. It is a single screen for the owner but a lot more goes on in the backend to enable the jukebox. Once we have a uniquely identifiable jukebox, it is simple to start sending song request and make the jukebox customizable to the owner preference and even add the possibility to have a dashboard to see users at the bar and visualize the player queue.

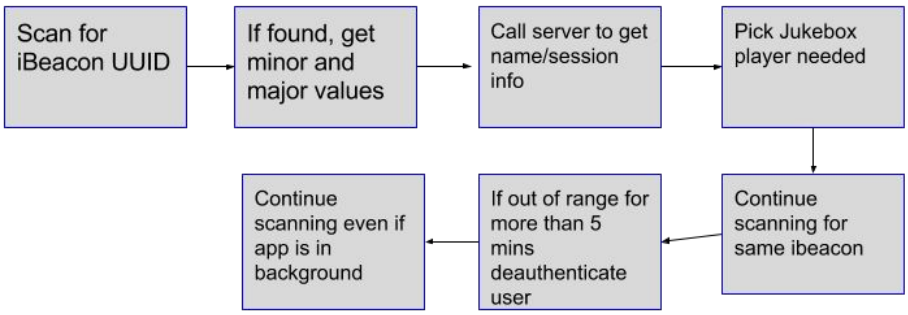


Figure 13. User Authentication Flow

The user authentication is on the iOS user app. The purpose of this component is to prevent users from playing songs after they go out of range and also make sure that the user can see the various iBeacons in that region.

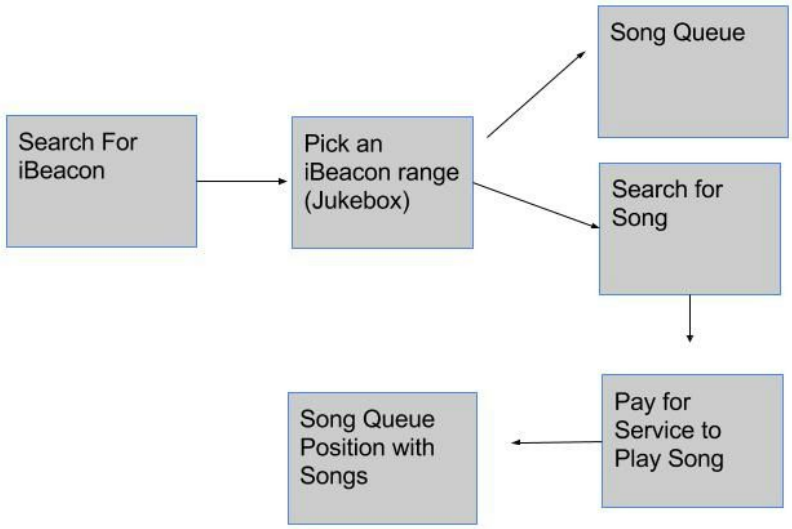


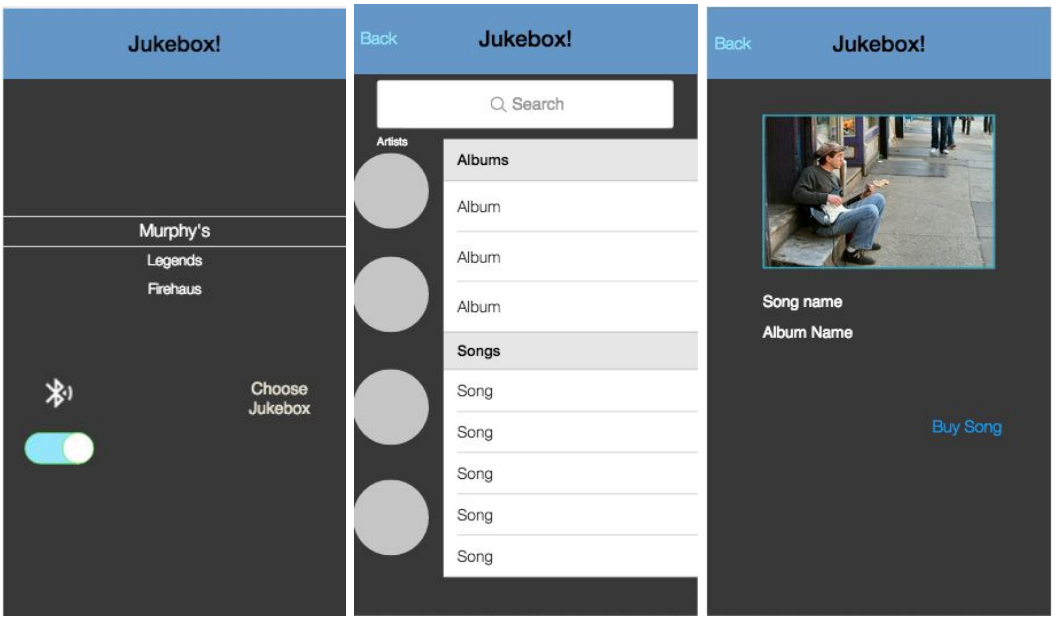
Figure 14. User Screens and Navigation Flow

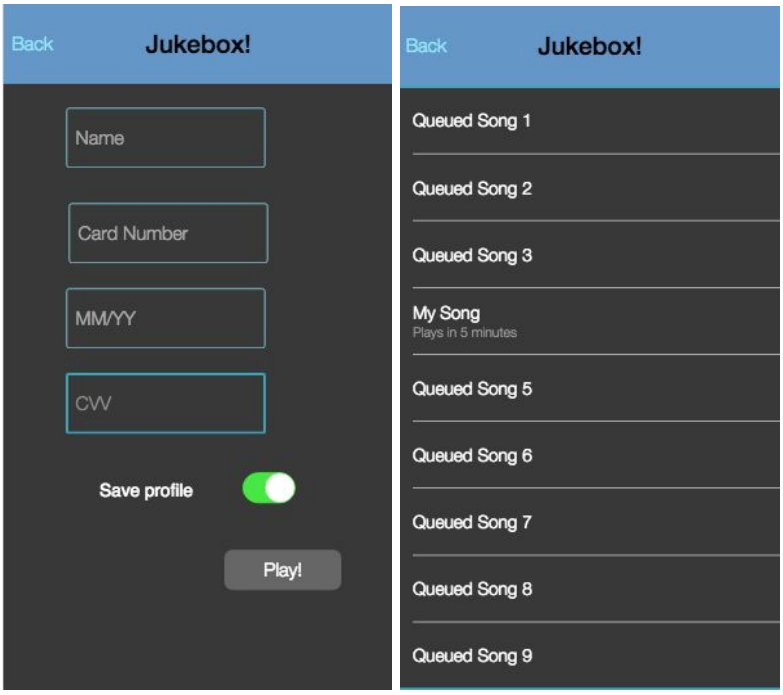
The user iOS app has the above flow. The UI design section below makes this a lot more clear and gives a better idea of how the iOS app functions. The iOS app is designed to be simple for a user and makes it

really easy to select songs and complete payments. This is the face of the project and simplicity in design was the major focus.

iBeacon Discovery The iBeacon discovery screen must have to capability to scan for iBeacons, call the server to get its name and display it as an option to get selected.
Search Songs Screen The search songs screen must be capable of searching for Spotify songs and when the song is selected, the pay screen shows up.
Pay for Song The Braintree API takes care of payments here. It can be paid via venmo, card, paypal or bitcoins. Successful payment sends the song to the server.
Song Queue The Song queue shows the order of songs and the position of your song, if any.

iOS UI Design





3. Requirements and Verification

Control Module		
Raspberry Pi 2 1. Power is being supplied to the Raspberry Pi at 5V ± 0.25V. 2. Raspberry Pi can connect to the internet. 3. GPIO Pins 12, 35, 40, 19, 23 on J8 Header can output 3.3V ± 0.1V.	1. Connect a voltmeter across TP1 and TP2. The reading should be 5V ± 0.25V. 2. $ ping google.com should show packets being received. Ctrl-C to return. 3. Connect a voltmeter across GND and applicable pin. Drive pins high and check if the reading falls in the appropriate range.	5
CODEC and DAC Convert I2S bit-serial digital audio to analog signal. Master clock must have a frequency of 12.288 MHz +/- 20ppm. Frequency response must be within a range of .01dB over the audible range (20Hz - 20kHz). 2nd harmonic must be at least 67dB down from first harmonic. The mute circuitry grounds the	Connect oscilloscope to MCLK pin and ground to read the frequency of oscillation. Use chirp tests once per octave to test the response at frequencies throughout the audible spectrum. Compare the frequency response to the response at 1kHz. Use the same chirp test to measure the second harmonic amplitude. Set the mute signal to low on each channel individually and observe the output of the channel on a multimeter.	25

analog output to under 50mV whenever the mute signal is low.		
Power Module		
Power Supply The output voltage for each parallel device must be at 5V ± 0.1V. The raspberry pi needs 1A-1.2A to function comfortably based on the data sheet. The 1 meter LED strips need 3A to function. The LED's do not need the max rating of 3.5A since it functions at 50% of the max described. The need for bright white light is limited and using a lesser power rating will give the sufficient brightness. This has been verified on the Adafruit website.	Connect a oscilloscope and check the output at the Raspberry Pi to be 5V +/- 0.1V and 1A +/- 0.1A and check if the LED's are at 5V +/- 0.1V and 3A +/- 0.5A. To measure the current a small resistance can be placed and the output voltage is measured. Use V=IR to check the current value.	10
Filter Module		
Filter Our filter must vary at most .99dB from the magnitude at 1kHz from 20Hz to 20kHz. The attenuation at 30kHz must be at least 6dB (½ of fundamental volume). Attenuation at DC must be at least 12dB (¼ of fundamental volume)	Send sinusoidal signal into filter once per octave and measure the amplitude response. Make sure a reading is taken at 20Hz, 20kHz, and 30kHz additionally. Compare sinusoidal input and output in MATLAB. Input DC signal of 5V and read output on multimeter.	
User Interface Module		
Audio Out Ports Analog signals on the leads are reproduced on pins of male XLR connector.	Apply a test voltage ramp waveform to the input pins individually and read the output with an oscilloscope.	5
LEDs Visible color corresponds to the software-defined color. Current draw at 5V when a 60-LED strip is completely turned on should be 3.5 amps or less.	Plug the LEDs into a lab power supply with 5V and observe the current drawn.	10

Sensors		
Photoresistor circuit In a well lit room, the circuit should output a logical high, as defined by the Raspberry Pi's processor (1.3V or more). Conversely, in a dim or dark room, the circuit should output a logical low (.8V or less). The photoresistor itself must adhere to the following in order to accomplish this: 5-10kOhms in a well-lit room, and 200kOhms +/- 50kOhms in a dim or dark room.	Read the resistance of the photoresistor in a well-lit room with a ohmeter and then read it with the sensor covered. Finally, read the resistance in a dimly lit room. Given that the photoresistor meets the requirements, test the circuit's output with a voltmeter in the same ambient-light conditions.	15
Software		
Owner App Creates the Jukebox Name, Logs into Spotify premium and sets up payment email address endpoint. Checks if the credentials are valid.	Do the login flow and check if the Jukebox has the right credentials to call the Spotify API and play songs.	10
Server Does the authentication for the username and password. Also links the owner to the payment account. It generates the unique major and minor values to identify the jukebox. Sends a session id back to the Smartphone with an expiry of a day. Also sends metadata of the jukebox to the smartphone	Check if the generated major and minor values are unique by doing a test to generate a large subset of the values. Check if payments are routed to the owner. Once the device is in session and in range it verify that it can send song ids	5
iOS App Scans for the iBeacon UUID. If found, gets the major and minor values and requests the server to authenticate a session. Also provides the smartphone with the Jukebox details such as the name. Deauthenticate if the user is out of range for more than 5	Finds the iBeacon in the range provided. Can call the server to get the name of the beacon and picks the right jukebox. Once the user is out of range for more than 5 minutes it no longer lists the user as in range.	10

minutes. The scanning frequency for the Raspberry Pi in the foreground is 5 seconds.		
iBeacon Registers the right UUID, major and minor values so that it can be detected. It maintains reliable scanning for 40ft.	Scan for the iBeacon and confirm detection of the iBeacon zone based on the major and minor values	5

Tolerance Analysis:

High quality audio means two things. First, accurate reproduction of sound, and second, low total harmonic distortion + noise. Quantitatively, we are looking to provide a frequency response from 20Hz to 20kHz that varies by no more than 1dB (measured against the amplitude at 1kHz), and we wish to provide a THD+N of .02% (-74dB). There are three components that contribute to these measures of fidelity: the CODEC/DAC, analog output filter, and the op amps in the filter.

First of all, we are going to ignore noise levels in our calculations following because our components are already so low in noise, that the levels produced are negligible compared to the amplitude of the harmonics. The op amp has output noise of 3.3nV/Hz^½. Over our bandwidth of 19980Hz, this amounts to 470nV of noise, which is 140 dB down from our full-scale output voltage of 5V. The codec itself has an incredible THD+N of -100dB, and although it does not provide noise specs separately, we will assume that the noise produced is negligible as well. Noise will also come from the four resistors in the filter, but using the approximation that resistor noise is .13*Sqrt(R) nV/Hz^½, the total noise from the four will be no more than 3.78uV (-122dB).

Now, we will focus on the frequency response flatness, which produces the accurate reproduction of music. The CODEC specs out to +/- .01dB over the range 10Hz to 20kHz. Thus, our filter must have a passband consistency of +/- .99dB over 20Hz to 20kHz to ensure the +/- 1dB flatness we are designing our device at. If we do not achieve this flatness, the result is less strong audio at the extremes of the audible range. This is not a huge issue as the human ear does not pick up these frequencies well anyway, but it is a consideration. Note that -6dB corresponds to half the volume.

Finally, we must keep the THD down to avoid strange unintended tones in the audio at harmonics to the fundamental frequency. Harmonics come primarily from the DAC. The CODEC/DAC specs out at -100dB THD+N, meaning the THD is even a few dB better than that. We do not have a THD plot for the CODEC, but we expect the second harmonic will be at least 67dB down from the fundamental and the third and up will be even lower. It is traditional in audio measurement to look at the attenuation of analog output filters at 1.5 times the maximum passband frequency; this corresponds to 30kHz for us. The attenuation is designed to be at least 7dB. This will attenuate the second harmonic to at least -74dB and we will ignore the higher order harmonics in this approximation. We consider -74dB to be our target THD

of .02% for our design, thus we are designing to just hit this requirement, but the CODEC/DAC output is probably better than we are designing for, so we expect to actually do better than -74dB THD+N. If, however, we do not hit this mark for THD+N, the sound will become noticeably inaccurate as harmonics become audible.

4. Logistics

4.1. Cost Analysis

4.1.1. Parts

Part	Source	Description	Qnt.	Unit cost	Total cost
1138	Adafruit	RGB 60-LED strip	2	24.95	49.90
2385	Adafruit	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	1	39.95	39.95
	Spotify	Spotify Premium Account	3 mo.	10.00	30.00
603-BLED112	Mouser	BLED112 Bluetooth Dongle	1	16.00	16.00
161	Adafruit	Photo cell (CdS photoresistor)	1	3.95	3.95
777-CS4272-CZZ	Mouser	CODECs Stereo Audio CODEC 114 dB 192 kHz	2	8.65	17.3
SDC4/8GBET	Kingston	8 GB Micro SDHC	1	3.92	3.92
863-MC7805CTG	Mouser	5V 1A linear regulator	1	0.95	0.95
658	Adafruit	5V 10A Power supply	1	25.00	25.00
511-LM323T	Mouser	5V 3A linear regulator	3	0.5	1.5
BOB-00500 ROHS	Sparkfun	SparkFun SSOP to DIP Adapter - 28-Pin	2	3.95	7.9

568-NC3MBH	Mouser	Male XLR connector	2	2.53	5.06
815-ABM3B-24.576-B2	Mouser	24.576MHz crystal oscillator	2	1.14	2.28
	ECE Supply	BJT - npn and pnp	3	0.00	0.00
		Total			203.71

4.1.2. Labor

Name	**Hourly Rate**	**Hours**	**Total (Rate * Hours * 2.5)**
James	$30.00	225	$16875
Nikhil	$30.00	225	$16875
Sam	$30.00	225	$16875
Total		675	**$50,625**

4.1.3. Total Cost

Section	**Cost**
Parts	$203.71
Labor	$50,625
Total	**$50,828.71**

4.2. Schedule

Week	**Tasks**	**Responsibility**
9/14	**<u>Project Proposal Due</u>** 1. Finalize Parts List 2. Decide on APIs 3. Project Proposal	 Sam Nikhil James

9/21	**Mock Design Review** 1. Audio Processing Circuit Schematic 2. Mock Design Review 3. iPhone App Mock UI	Sam James Nikhil
9/28	**Design Review** 1. Order Parts 2. Raspberry Pi Setup 3. Design Review	Sam James Nikhil
10/5	1. Breadboard and debug audio processing circuit and output ports 2. Play music from Spotify on Raspberry Pi 3. Finish iBeacon Scanning. Build UI and integrate Braintree payments.	Sam James Nikhil
10/12	1. Breadboard and debug photosensor. Begin writing CODEC driver 2. Output I²S from Raspberry Pi 3. Write Server to accept song requests. Make sure iBeacons have unique Major/Minor values. Write payments extension for the server.	Sam James Nikhil
10/19	1. Complete CODEC driver - test audio throughput 2. Debug Audio Fidelity & Clocking 3. Test major functionality of phone-server interactions. Unit test major interactions.	Sam James Nikhil
10/26	**PCB First Revision** 1. Submit PCB design of audio circuit, power supply, and photosensor. 2. Write LED controller scripts 3. Clean up UI and complete testing the phone-server interaction	Sam James Nikhil
11/2	**Mock Demo & PCB Final Revision** 1. Test PCBs and submit changes 2. Public User Testing & Feedback 3. Add extra functionality with Social interactions for the app	Sam James Nikhil
11/9	**Mock Presentation** 1. Continue to test all hardware and driver functionality- Submit package design to machine shop 2. Test & Debug 3. Prepare for presentations and debug/test components app/server components for the demo	Sam James Nikhil

11/16	**Demo** 1. Assemble and test newly packaged product 2. Model and 3D-print housing 3. Finish Demo and prepare for the final presentation	Sam James Nikhil
11/23	**Thanksgiving Break** 1. Start writing final report. Test if necessary. 2. Software Testing 3. Software Testing	Sam James Nikhil
11/30	**Final Presentation** 1. Produce finished product 2. Finalize Presentation 3. Start writing final report	Sam James Nikhil
12/7	**Report and Lab Notebooks** 1. Finish report 2. Finish Final report and submit lab notebook 3. Finish Final report and submit lab notebook	Sam James Nikhil

4.3. Contingency Plan

APIs: If the Spotify API is not very responsive or there are issues with the open source library to stream the songs, the Soundcloud API will be picked.

CODEC/ DAC: If there are issues with writing the low level drivers/getting the DAC to work, a HIFIBerry DAC with similar specifications will be taken. It has direct integration with the Raspberry Pi and the libraries were written for a simple integration with the board. This will make the project more expensive though.

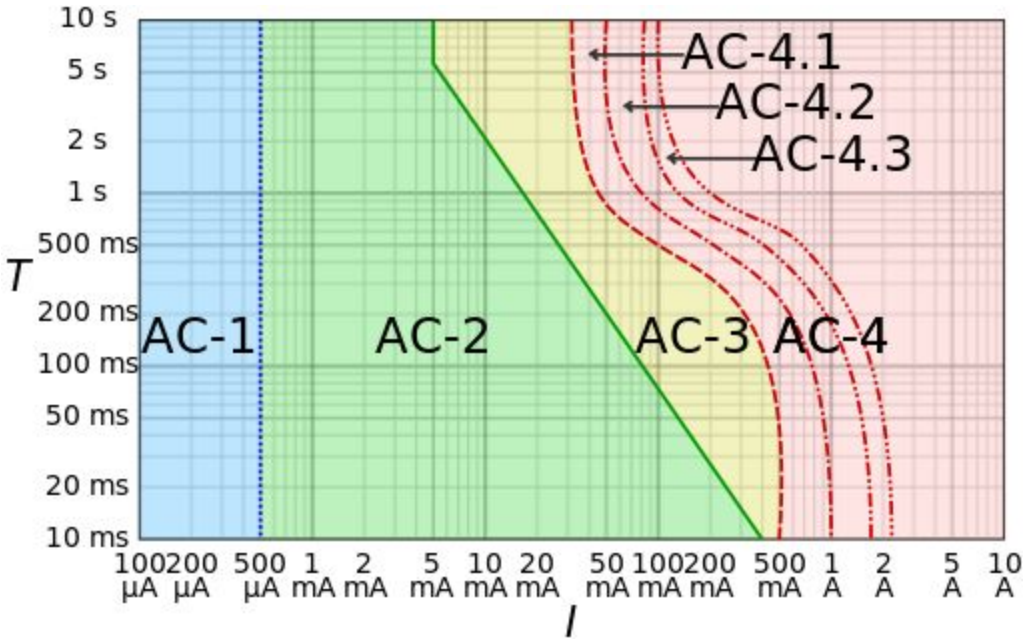
5. Integrity

5.1 Ethics

The main points of possible contention in the product is the use of the Spotify API. Trivia games are not allowed on the same app as per the API. A conflict of interest is possible with Spotify's usage of the API in terms of building a jukebox. The API could change any day and could lead to possibly making the product defunct since it is heavily dependent on the Spotify API to get songs and search for songs. A disclaimer must be provided because of the dependence on the Spotify API and the possibility that it could make the product unusable if there is a change in the terms of how the API is used or rate limited based on the usage of the app. Written permission will be requested to use the API for commercial purposes.

5.2 Safety

The power supply will plug into a wall outlet, which carries 120V AC and up to 15 A. Normal care should be taken to ensure that people cannot come into contact with the power line. Make sure that the power supply is completely plugged into the wall when in use. Do not expose liquid, raw wires or any other conductive material to the interface between the wall and the power supply.



Log-log graph of the effect of alternating current *I* of duration *T* passing from left hand to feet as defined in IEC publication 60479-1.[18]
AC-1: imperceptible
AC-2: perceptible but no muscle reaction
AC-3: muscle contraction with reversible effects
AC-4: possible irreversible effects
AC-4.1: up to 5% probability of ventricular fibrillation
AC-4.2: 5-50% probability of fibrillation
AC-4.3: over 50% probability of fibrillation

After conversion to 5V DC at 10A, the on-board power lines do not offer a threat of electrocution. However, the LEDs still require a high current. Measures to prevent short circuits will be kept in consideration to prevent components from taking damage or overheating. Power to the Raspberry Pi will be current-limited with a fuse, so that if someone accidentally shorts the Pi, it cannot draw more than 2A. Before tinkering with electrical components, make sure that the device has been unplugged, capacitors

have been given a sufficient time to discharge, and that your body is static-free. Upon completion of the project, electrical components will be contained in a housing, so user exposure to them is minimized.

6. References

Electric Shock:
https://en.wikipedia.org/wiki/Electric_shock#/media/File:IEC_TS_60479-1_electric_shock_graph.svg
CS4272 datasheet: http://www.cirrus.com/en/pubs/proDatasheet/CS4272_F1.pdf
Raspberry Pi logic levels: http://www.scribd.com/doc/101830961/GPIO-Pads-Control2
High Quality Audio with a Raspberry Pi:
http://www.ti5.tu-harburg.de/staff/meier/master/meier_audio_over_ip_embedded.pdf
Raspberry Pi 2 Model B - Pin Numbering:
http://pi4j.com/pins/model-2b-rev1.html