Direct Music Synced LED Strips

By
Daniel Vargas
William Reinhard
Siyan Shaikh

Final Report for ECE 445, Fall 2020
TA: William Zhang

Team 1
December 2020

Abstract

This project centers around the design and creation of a more accurate, configurable, and entertaining LED lighting strip system that is able to reflect changes in audio by changing the LED's brightness and color. Currently existing LED strips lack functionality and accuracy due to inefficient microphones and simple circuitry. Our project addresses the downsides of these strips while adding additional user-centric features. We leveraged the frequency identification ability of a digital Fast Fourier Transform (FFT) to map the analog audio input sound to a range of frequency bins, which were then used to drive the brightness of the LED lights. These LED lights are simultaneously controllable and customizable by a connected Bluetooth enabled Android phone which connects to a Bluetooth receiver powered by a complementary power circuit sitting on our unique printed circuit board (PCB).

Table of Contents

1. Introduction

1.1 Objective

Created by renowned University of Illinois Professor Nick Holonyak, Light Emitting Diodes (LEDs) have long been known as a revolutionary technology that has shaped much of the consumer electronics industry. They provide a clean, bright, and customizable source of light that works well in combination with other electronics to generate everything from screen displays to home lighting. One particular use for LEDs is within LED light "strips," which are used as a decorative electronic item to add a colorful light display to any home. These light strips often come with a feature to "react" to music and audio, but they have shortcomings when performing audio to light conversion reliably.

We set out to directly connect the LED strips with analog music, using FFT analysis to generate broader levels of pitch differentiation. This enabled us to have the LED lights display corresponding to both sound level and pitch. To address the customizability, we developed an application that allows for pitch selection and specific color outputs displayed with aural direction on two separate "left" and "right" LED strips.

1.2 Background

Modern-day LED strips marketing to consumers with audio tracking capabilities are often driven by a microphone that detects sound and simply reflects the changes in sound level as changes in the intensity of the LEDs [1]. As such, there is often an extensive delay between when sound is audibly heard and when it is accurately reflected by the LED lights it is connected to. There is also no filter in place to remove ambient noise and no differentiation between someone simply speaking and the intended audio source to display. In addition, only the intensity of brightness is changed and not the color, leading to a less immersive experience. Users are limited in what can be displayed by the patterns pre-programmed by the manufacturer, and there is no way to differentiate between left or right-side audio output.

Our approach is innovative because it addresses the pitfalls of currently available products on the market by directly connecting to the music source, enabling us to relate changes within the audio (tone, pitch, frequency, and direction) to the LED strips. We also enhanced user customization and input by offering an interface to select color themes, patterns, and palettes.

1.3 High-Level Requirements

- Our project must be able to reflect changes in frequency and intensity of sound by changing the LED color or change in color strength (power) at about 16 *ms* faster than microphone-based LEDs.
- Our light strip should be able to reflect custom color selections based on five pitch ranges: Bass (20-60 Hz), Low Midrange (250-500 Hz), Midrange (500-2 kHz), High Midrange (2 kHz-4 kHz), and Presence (4 kHz-4.5 kHz), and strobe patterns based on user input from an application.
- The application will allow the user to select "color themes" to prevent colors from blending.

2. Design (More description and info in blocks)

Both the LEDs and Controller Module are powered by 12 V power supplies regulated by a power distribution element. Data is generated from user input via the user interface that submits requests to the back-end application. This back-end application then prepares and generates the necessary data sent over Bluetooth. The actual transmission is handled by the Bluetooth driver and is sent to the Bluetooth Receiver, which sends data to the FFT and pulse width modulation (PWM) circuits. These generate the necessary output to drive the color, intensity, and functional behavior of both the left and right LED strips.

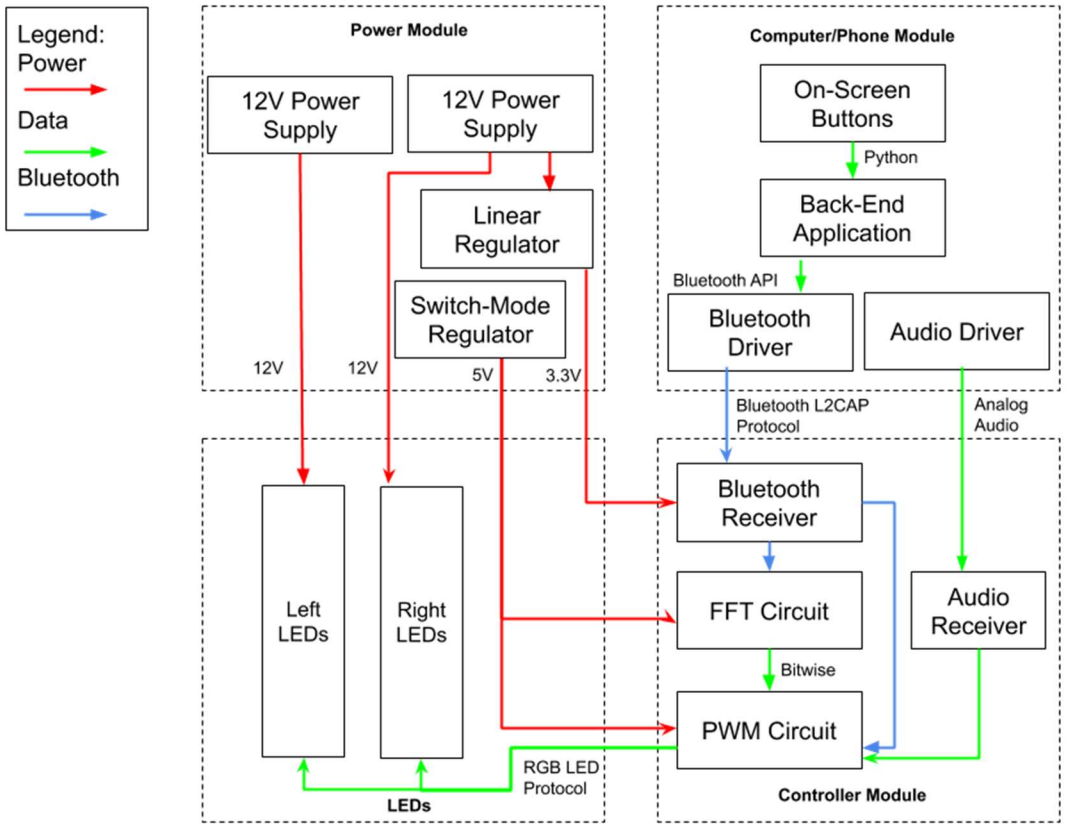

Figure 1: Block diagram of the components and modules

2.1 Power Module

2.1.1 Power Supply

The power module is a critical component of our overall design, as each of the subsystems require different voltages to operate effectively. At the core of our design, we utilize two 12 V, 4 A DC power supplies to power the entirety of the circuit, as our LED lights require 43 W to reach maximum brightness [1]. One of the power supplies directly powers an LED strip, while another distributes power to both an LED strip and the PCB.

2.1.2 Switch-Mode Regulator

We leverage a 12 V to 5 V switch-mode regulator to step the voltage from the power supplies down to the 5V required for the ATMEGA2560 to operate according to spec. This is a change from our original design, where we planned to use a linear regulator to step 12 V down to 5 V. Upon testing we identified current spikes that caused anomalies in the behavior of the ATMEGA2560, leading us to switch design to a more current-stable switch-mode regulator. To keep the ATMEGA2560 from overheating, we determined that we wanted to keep input current under 10 mA, based on lab testing.

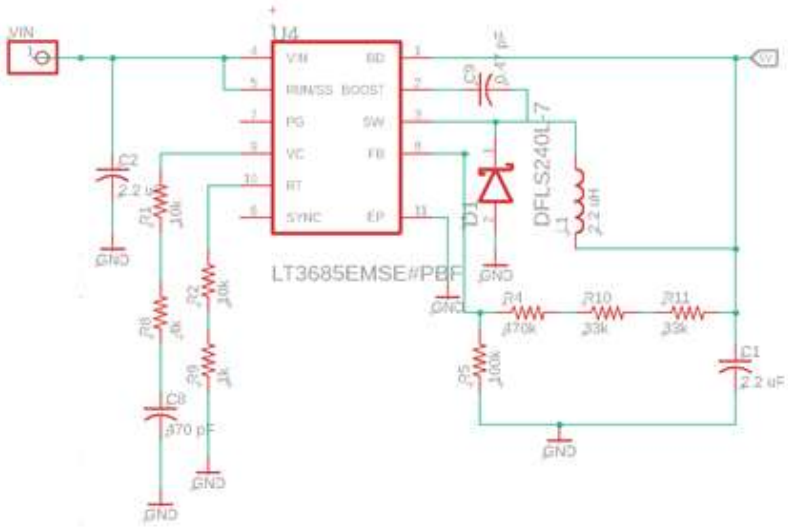

Figure 2: Schematic for LT3685 switch-mode regulator

2.1.3 Linear Regulator

We step the power supplies from 12 V down to 3.3 V for the HC-06 Bluetooth module using an ADP3309 linear regulator. This setup did not require much current regulation as the HC-06 Bluetooth module included small current regulation. It comes flanked by two capacitors to reduce noise. This device is responsible for sending power to the Bluetooth receiver.

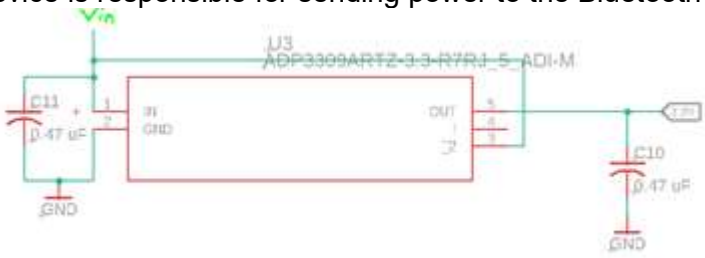

Figure 3: A schematic of a ADP3309 linear regulator

2.2 Controller Module

2.2.1 FFT Code

This FFT code is run on the ATMEGA2560, which we selected for its high processing clock speed (16 MHz) and available on-chip flash memory. ATMEGA microcontrollers are also compatible with Arduino sketches which allowed us access to the myriad of Arduino embedded libraries available for public use. Our FFT program leverages the open-source ArduinoFFT Library to handle the sampling, calculation, and presentation of audio frequency data.

The FFT code serves to analyze the different frequencies of the music inputted, generating an actionable output for the PWM in accordance to which of the five ranges the audio falls under. This component was integral to distinguishing music fragments that fall into the highlighted frequency bucket. With these frequency buckets we then determine the brightness of the LEDs and send the final values to the LEDs to drive the light.

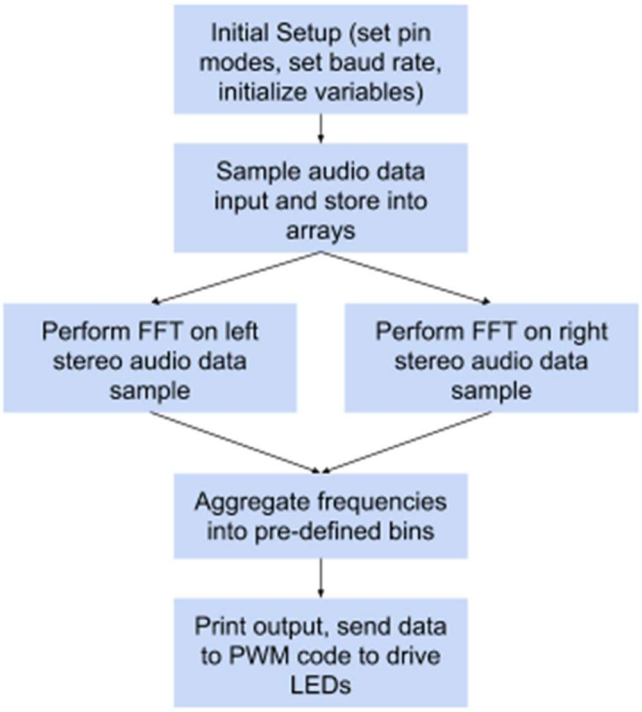

Figure 4:Pseudocode for controller FFT code [2]

The critical part of our design relies on effective real-time Fast Fourier Transform (FFT) analysis of the music being played. Our sensitivity to frequencies and frequency range will depend on our FFT size, and sample rate. We followed the Nyquist Theorem

$$X_c(j\Omega) = 0 \qquad \text{for } |\Omega| > \Omega_N.$$

Then $x_c(t)$ is uniquely determined by its samples $x[n] = x_c(nT)$, $n = 0, \pm 1, \pm 2, \ldots$, if

$$\Omega_s = \frac{2\pi}{T} > 2\Omega_N.$$

Equation 1: Nyquist Theorem for sampling frequency

where it reads that for a sample rates we must make sure it is greater than twice the highest frequency of the samples to avoid aliasing. This means our sampling rate must be double the maximum frequency we wish to measure, which was 4500 Hz. We selected this maximum range as above 4000 Hz hits the ‘presence’ range. This frequency range is mostly composed of harmonics, which are integer multiples of a fundamental tone. It is redundant to measure further beyond 4500 Hz as we would read frequencies that are already represented at lower frequency levels. Harmonics are also difficult to pick up by the human ear, and thus do not contribute to the user experience [3].

Selecting 4500 Hz means we must sample at 9000 Hz. This will directly impact our latency. Smaller bin sizing indicates that they can fill their bin sizes faster, however, FFT calculation speeds are smaller with our 16 MHz processor. With an FFT bin size of 256 we used the following equation

$$ExecutionTime = k_{FFT} N \log_2 N$$

Equation 2: FFT execution time

where N is the number of bins and k_{FFT} is the time it takes to carry out a single FFT bin, in our case ~12 nanoseconds [4]. We calculated our total execution time to be around 2.56 *ms* for a full spectrum FFT.

2.2.2 PWM/Light Code

The PWM code is also run on the ATMEGA2560 and is responsible for taking input from the FFT code as shown in figure 4 and reflecting those changes in the overall light intensity. This code segment also sets the color of the LED strips in accordance with the information that is sent via Bluetooth from the mobile phone application. The equation used to calculate the final brightness for left and right channels is

$$brightness_{left/right} = (frequencyBins_{left/right}[presence]/2300) * 255$$

Equation 3: Brightness equation for left and right channels of presence frequency bin

$FrequencyBins_{left/right}$ is the presence component of a specific audio sample. The value 2300 was selected as a normalizing factor because as we tested the maximum values of each frequency bin, the largest value we encountered was 2300.

2.2.3 Bluetooth Receiver

The Bluetooth receiver is responsible for receiving information from the phone application that dictates the theme color selection of the LEDs. This is the main interface between our PCB design and the user application and uses the L2CAP Bluetooth protocol.

```
void readInBluetooth() {
  while (Serial2.available())
  {
    delay(10);
    char c = Serial2.read();
    if(c != ',' && rangeThemeFlag == 0) {
      btInput1 += c;
    }
    else if(c == ','){
      rangeThemeFlag = 1;
    }
    else{
      btInput2 += c;
    }
  }
```

Figure 5: Code responsible for reading input from Bluetooth receiver

The Bluetooth receiver was linked to the ATMEGA2560 as shown in figure 6 [5].

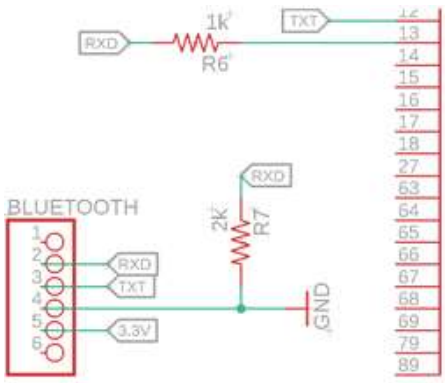

Figure 6: Schematic of the connection between the Bluetooth receiver and the ATMEGA2560 circuit

2.3 LED Module

The LED module is made up of two sections, the left and right LED blocks in parallel. They are identical in every way except in the data that they receive. These blocks are made up of LEDs in parallel that receive RGB data from the controller to change colors as the controller input changes. The lights being displayed are aligned with the analog music data and match the audio being played.

The LEDs are capable of running at 25 kHz in order to reduce color flashing, as the IEEE threshold for increased seizure risk is between 3-60 Hz [6]. The WS2815 LED strips that we're using cannot operate at a frequency less than 400 Hz/s, meaning they will never operate in the seizure risk range [7].

2.4 Computer/Phone Module

This module is made up of an Android application. This is the main interface with the user; allowing them to decide what theme they would like to see the LEDs show. The user interface first displays a seizure warning in accordance with IEEE standards, which will be discussed in a later section. Users can then choose from five different themes, each of them corresponding to a predefined 16 color RGB color palette. The application then enables the user to select a specific range from 0-4.5 kHz, in accordance with Bass, Low-Midrange, Midrange, High-Midrange, and Presence frequency bins. The application sends pitch and color preference to the Bluetooth receiver in the controller module using standard L2CAP protocol. After initially doing some development with Flutter, it became apparent that we were allocating too much time toward developing the phone app, which isn't a primary focus of the project. Upon discussion with our TA, we settled on using MITAppInventor to develop the application. The final product is depicted in figure 7.

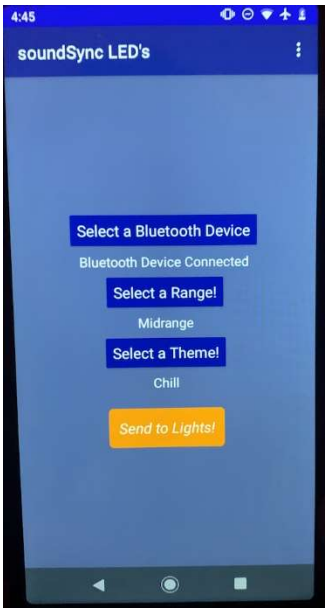

Figure 7: Final Android application for range & theme selection

We also planned and included the ability to change the settings in real-time. Pressing the ‘Send to Lights!’ button instantly updates the controller color and pitch preferences to the selected Bluetooth device.

2.5 Overall Device Schematic

Our PCB (figure 8) was designed to keep testing and errors in mind. We included pinouts for more digital and analog pins than we needed for the LEDS, as well as options to wire in the Bluetooth receiver in case our final PCB had flaws. The two voltage regulators and circuits for the controller are all integrated onto the PCB, alongside the Bluetooth receiver, audio jack receiver, and the ATMEGA2560 controller. The circuit board has more resistors than initially deemed necessary as the exact resistor values we needed were not available, so there are multiple resistors in series to sum up to the values we need. One error of the design is that it is difficult to both power the PCB and the LEDs without soldering their wires together, which leads to a cumbersome design. A future development of the PCB would include power printouts for the LEDs that are connected to the singular VIN pinhead.

When soldering the PCB, we were careful to solder the power circuits first to test if they were working properly, and for early versions of the bootloader, we would put 5V from an Arduino board at the positive terminal of the C1 capacitor to power our board. We then placed the capacitor back and were able to bootload using our power circuits.

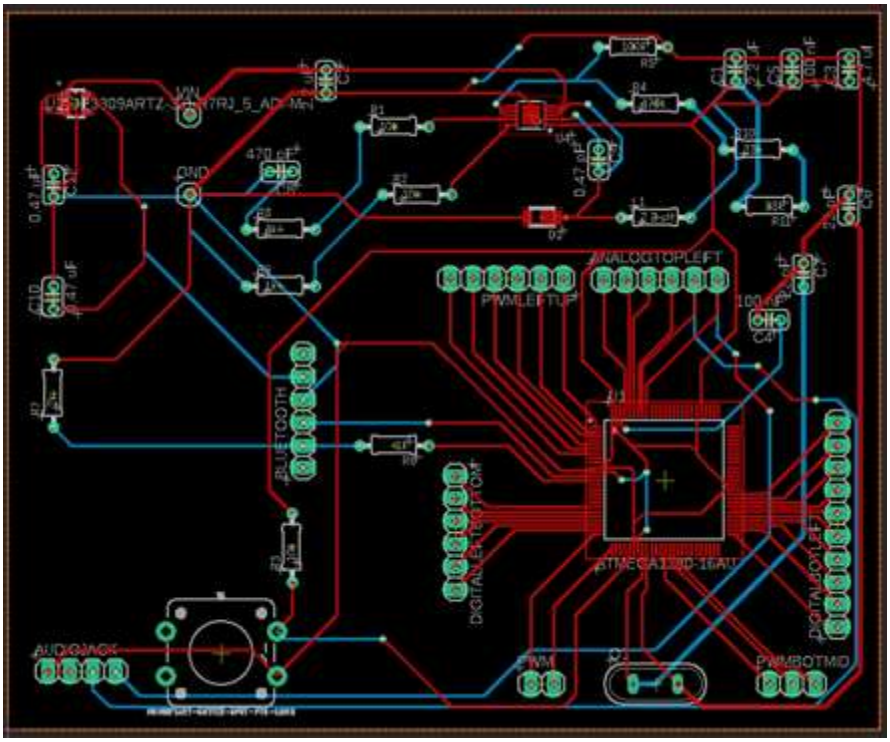

Figure 8: Complete PCB schematic

We also included to have all the pinouts close to the ATMEGA2560 and the power components further away for easier debugging. We knew that the likely sources of issues would be the power supply, so having them at the edges of the PCB allowed for easier soldering and removal of parts. We also included the audio jack pinouts towards the edge for easier access as well.

3 Verification

3.1 Power System

The power systems were individually measured using oscilloscopes before and during operation of the entire design. The system can successfully drop 12 V to 5 V and to 3.3 V with currents under 4 mA for our controller, while maintaining tight bounds on our output voltage noise.

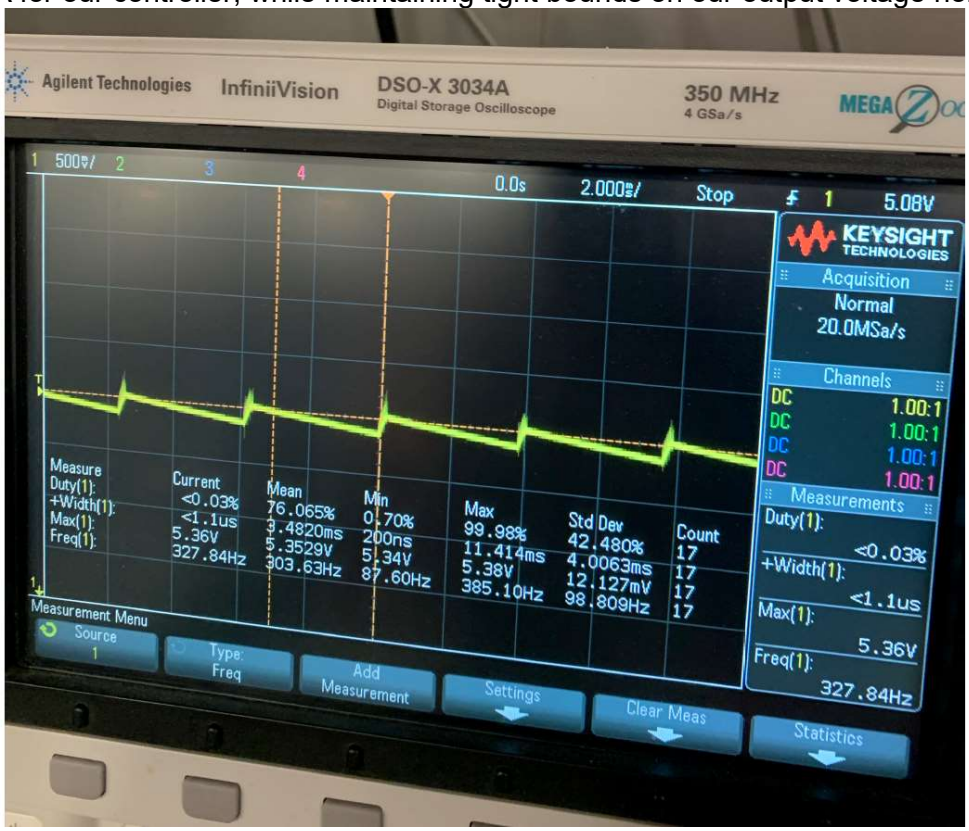

Figure 9: 12V to 5V regulator output

3.2 LED System

The LED system should be able to display multiple colors as well as handle real-time input from our controller. We confirmed LED functionality by driving a basic RGB signal to the lights, and then confirmed it could handle changing input by creating a flashing RGB signal. Figure 10 depicts the initial testing that was completed to verify the LEDs were working and could be addressed individually. We verified the remainder as we could visually see the lights changing with the music it was sampling.

Testing All Lights

```
For each light in the light strip:
     Light = ClassicRGB(255, 0, 0) //Sets each light to Red

FastLED.show() //Fast Led's Function to display the colors
```

Testing Individual Addressability

```
For each light in the light strip:
     Light 15 = ClassicRGB(0, 255, 0) //Sets the 15th light to Green

FastLED.show() //Fast Led's Function to display the colors
```

Figure 10: Pseudocode to test LED functionality

3.3 App/Bluetooth System

In order to verify proper functionality of the application system, we transmitted data from the Android device over Bluetooth, which was ultimately output on the serial monitor. Figure 11 exemplifies transmissions of each of the color themes and frequency ranges, demonstrating complete functionality.

Figure 11: Desired serial monitor input from the Bluetooth module

This proved that our smartphone could connect to the Bluetooth receiver, and that the receiver was sending the correct data to our controller.

3.4 Controller System

3.4.1 Audio Input

To test that the audio input was being received, we connected the TRRS audio jack to a secondary Arduino Mega and used the serial monitor to output the raw data and view the audio as a signal. We then used an iPhone as a playback device and played singular frequencies at a constant volume to ensure the serial plotter, and thereby the Arduino, was reading audio input data. The output generated with the plotter was consistent with the expected output for each signal that was tested.

3.4.2 FFT Code

The FFT code was first tested with a generated sinusoidal signal of frequency 1000 Hz (and a sampling rate of 5000 Hz). The lower sampling rate was to test if the ATMEGA2560 could handle the real-time FFT. This signal was sampled and then passed into the FFT calculation, which verified the correct frequency and that the FFT library was functioning properly

```
19:10:00.755 -> 703.125000Hz 15.9997
19:10:00.755 -> 738.281250Hz 16.5302
19:10:00.755 -> 773.437500Hz 16.1558
19:10:00.755 -> 808.593750Hz 22.5839
19:10:00.755 -> 843.750000Hz 23.2861
19:10:00.755 -> 878.906250Hz 15.9027
19:10:00.800 -> 914.062500Hz 15.3008
19:10:00.800 -> 949.218750Hz 675.6170
19:10:00.800 -> 984.375000Hz 3110.2780
19:10:00.800 -> 1019.531250Hz 2368.1164
19:10:00.800 -> 1054.687500Hz 196.6931
19:10:00.800 -> 1089.843750Hz 5.3674
19:10:00.800 -> 1125.000000Hz 18.9076
19:10:00.800 -> 1160.156250Hz 21.6600
19:10:00.800 -> 1195.312500Hz 19.9096
19:10:00.800 -> 1230.468750Hz 21.4969
19:10:00.800 -> 1265.625000Hz 12.3226
19:10:00.800 -> 1300.781250Hz 20.0652
19:10:00.800 -> 1335.937500Hz 9.6516
19:10:00.800 -> 1371.093750Hz 16.5551
```

Figure 12: The resulting frequency concentrations of the 1000 Hz signal

Setting the sampling rate to 9000 Hz, we tested the directional capabilities of the FFT by feeding in only left or right audio at arbitrary frequencies. Both directions worked satisfactorily and correctly determined the highest intensity in the right frequency bin.

```
19:12:41.043 ->
19:12:41.043 -> Bass (20-60 Hz): 361.60 | 0.00
19:12:41.043 -> Low Midrange (250-500 Hz): 25.10 | 0.00
19:12:41.093 -> Midrange (500-2000 Hz): 1530.33 | 0.00
19:12:41.093 -> High Midrange (2000-4000 Hz): 937.07 | 0.00
19:12:41.093 -> Presence (4000-6500 Hz): 634.73 | 0.00
19:12:41.417 ->
```

Figure 13: Input of a 1000 Hz signal from the *left* analog audio input

```
19:13:34.476 ->
19:13:34.476 -> Bass (20-60 Hz): 0.00 | 380.48
19:13:34.476 -> Low Midrange (250-500 Hz): 0.00 | 58.33
19:13:34.524 -> Midrange (500-2000 Hz): 0.00 | 2544.34
19:13:34.524 -> High Midrange (2000-4000 Hz): 0.00 | 553.07
19:13:34.524 -> Presence (4000-6500 Hz): 0.00 | 727.17
19:13:34.850 ->
```

Figure 14: Input of a 1000 Hz signal from the *right* analog audio input

Finally, we tested the full capability of the FFT system by playing an arbitrary song (Bohemian Rhapsody by Queen) and monitored the output as the song was playing and verified that changes in the left/right audio reflected with the output FFT

```
19:14:25.141 ->
19:14:25.141 -> Bass (20-60 Hz): 2520.47 | 2443.94
19:14:25.141 -> Low Midrange (250-500 Hz): 6167.93 | 6395.13
19:14:25.141 -> Midrange (500-2000 Hz): 20954.91 | 20779.00
19:14:25.189 -> High Midrange (2000-4000 Hz): 11897.49 | 13294.29
19:14:25.189 -> Presence (4000-6500 Hz): 12036.70 | 13533.47
19:14:25.514 ->
19:14:25.514 -> Bass (20-60 Hz): 3056.09 | 3110.13
19:14:25.514 -> Low Midrange (250-500 Hz): 7835.04 | 7842.81
19:14:25.561 -> Midrange (500-2000 Hz): 17149.99 | 16888.98
19:14:25.561 -> High Midrange (2000-4000 Hz): 6035.03 | 4887.31
19:14:25.561 -> Presence (4000-6500 Hz): 4788.78 | 4395.20
19:14:25.892 ->
19:14:25.892 -> Bass (20-60 Hz): 2973.56 | 3027.46
19:14:25.892 -> Low Midrange (250-500 Hz): 2128.91 | 1622.00
19:14:25.937 -> Midrange (500-2000 Hz): 25913.93 | 24043.28
19:14:25.937 -> High Midrange (2000-4000 Hz): 10036.37 | 12963.20
19:14:25.937 -> Presence (4000-6500 Hz): 11723.84 | 14909.70
19:14:26.266 ->
```

Figure 15: A sample of the output during the playback of Bohemian Rhapsody

3.5 FFT Timing Analysis

With our FFT software confirmed, we turn to its calculation speed. Our initial design FFT timing was estimated at 2.56 *ms* for a single full frequency range FFT to complete. We verified the small latency timing by measuring the difference in timings from input to the controller to the output of frequency bin sizes.

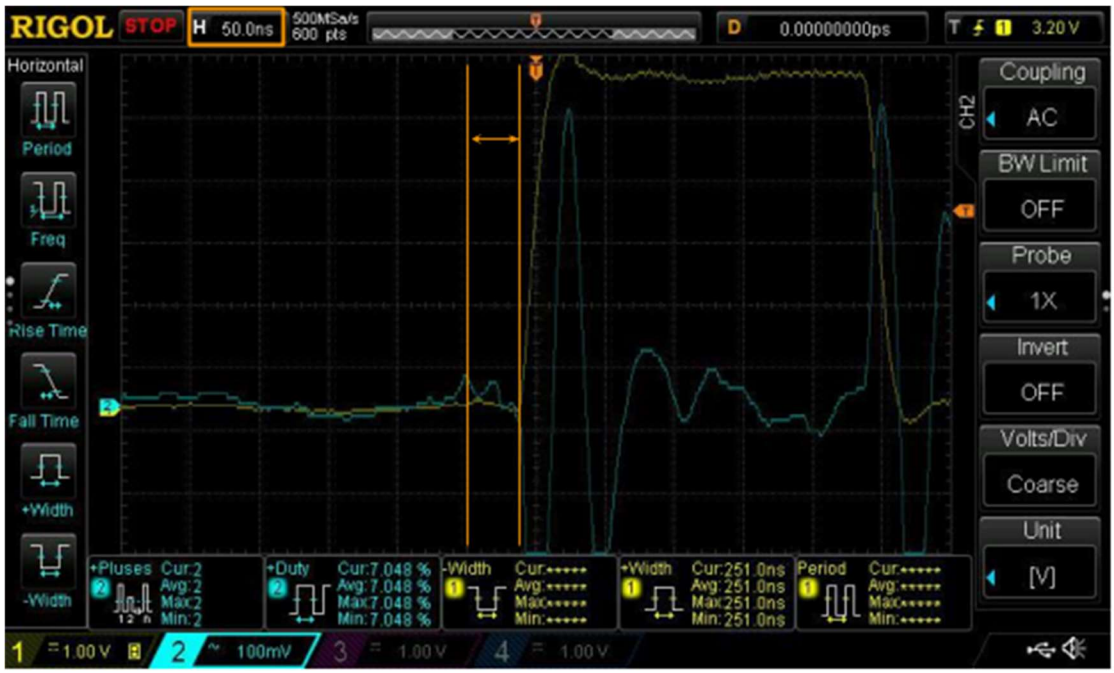

Figure 16: Audio input (yellow) vs data output (blue)

Above we can see there is a delay in the peak of data output after the audio increases, and we notice another delay in data output once the audio has sharply decreased. These delays were calculated to be less than 50 *ns*, which can be seen in figure 16. Each of the divisions in figure 16 are 50 ns, and the difference between the first audio peak and when data is within one division. This is much less than our anticipated calculation timings and served to manipulate the FFT bins for increased speed. We believe this to be an in error in our initial timing as we used the wrong constant for k_{FFT}, as it was used for much slower processors. We also significantly reduced our sample size from 256 to 64. We came to this number by testing and visually confirming which bin sizes resulted in the fastest translation from audio to light. The accuracy was kept almost identical which we verified by checking the FFT outputs as demonstrated in figure 15. We beat our microphone-based competitors by more than 16 *ms* as their calculated latency values were about 20 *ms*, which was based on the datasheet provided by competitors [1].

Additionally, we also changed the size of the five FFT bins to incorporate larger ranges of frequencies to fully capture the input audio and allow for greater intensities in brightness for the different ranges.

4 Costs and Schedule

4.1 Labor

We determined our total labor cost to be $18,750, based on the labor rate of 3 graduate-level engineers at $25/hour, working 12.5 hours per week over the course of 10 weeks. Equation 4 demonstrates how we came to our labor cost.

$$3\ Engineers * \frac{\$25}{hour} * \frac{12.5\ hours}{week} * 10\ weeks * 2.5 = \$23,437.50$$

Equation 4: Total cost analysis of labor

Table 1: Cost Analysis Table: Labor

Partner Name	Number of Hours	$ per hour	2.5x Multiplier	Total
Daniel	125 hours	$25.00	2.5	$7,812.50
Siyan	125 hours	$25.00	2.5	$7,812.50
William	125 hours	$25.00	2.5	$7,812.50
Total				**$23,437.50**

4.2 Parts

The necessary components for our project totaled $92.2. Actual costs for the project, including test & validation equipment as well as component volume are included in appendix A. It should be noted that we didn't use the machine shop for our project, but in future work we aim to develop an enclosure with assistance from the machine shop.

Table 2: Parts Cost

Part	Manufacturer	Retail Cost ($)	Actual Cost ($)
ATMEGA2560	Microchip	$11.99	$11.99
SparkFun TRRS 3.5 mm Jack Breakout	SparkFun	$3.95	$3.95
3.5 mm Headphone Jack Splitter	Koopao	$5.99	$5.99
BTF Lighting WS2815	BTF Lighting	$29.88	$29.88

HC-06 Bluetooth Module	DSD	$8.49	$8.49
12 V Power Supply	Shenzhen Yinghui Yuan Electric Co.	$20.00	$20.00
12 V to 3.3 V Low-Dropout Linear Regulator	Texas Instruments	$1.11	$1.11
Crystal Oscillator	TXC Corporation	$0.30	$0.30
Reset Button	SparkFun	$0.50	$0.50
Catch Diode	Diodes Inc.	$0.42	$0.42
5 V Switch-Mode Converter	Linear Technology	$9.57	$9.57
Total			**$92.20**

4.3 Schedule

Over the course of the past 10 weeks, we've properly adhered to the schedule that we initially laid out in our design document. In retrospect, we would have liked to spend more time verifying and testing the PCB to ensure that we're able to get a fully functioning PCB by the second round of PCB orders. This would provide more time for us to troubleshoot our PCB design, which we found slightly compressed in our original schedule.

Table 3: Schedule Breakdown

Week	Daniel	Will	Siyan
9/27/20	Refine PCB	Begin developing color selection software	Select parts and order
10/4/20	Finish PCB and order	Finalize UI and connect to Bluetooth API	Verify spectrogram frequencies for ideal FFT specifications
10/11/20	Connect audio shield to controller and verify clean signals	Bluetooth verification	Assemble linear regulator the and verify integrity
10/18/20	Start controller/FFT logic	Establish Bluetooth connection to phone and controller	Breadboard power systems to components to verify
10/25/20	Continue FFT logic and verify with LEDs	Continue Bluetooth related software	Continue FFT logic
11/1/20	Verify music amplitude to LED power relationship	Finalize color software and verify transfer via Bluetooth	Verify music amplitude to LED power relationship
11/8/20	Prepare mock demo	Prepare mock demo	Prepare mock demo
11/15/20	Verify solder connections	Solder PCB components together and verify	Test response with different music mixes
11/22/20	Test LED response speeds	Clean UI on app	Clean UI on app
11/29/20	Prepare final report	Prepare final report	Prepare final report

5 Conclusion

5.1 Accomplishments

Overall, we successfully accomplished all our high-level requirements as well as created a visually appealing device. We were able to test with any music ranging from rap to classical music and accurately reflect changes in pitch volume in changes in lighting. The smartphone application was intuitive, easy to use, and able to communicate at any time with the controller. The device could also change settings while in use, for example changing from bass detection to presence detection. We also found a unique success as almost every person who has seen a demo has asked to purchase such a device and enjoyed using our project.

5.2 Uncertainties

Our largest uncertainty concerns the audio itself. To get analog data from a smartphone to both play and go into our device, we needed to use an audio jack splitter. This requires the speaker to have a 3.5 mm audio jack connector, which in the modern wireless age is becoming less frequent.

Another uncertainty is the duration of operation for the device, as we did not use it for hours on end and have not fully tested its longevity. There is a possibility that the LEDs overheat or deteriorate in condition with long sessions of operation, yet to this point we have not seen any indication that this might be the case.

5.3 Ethical Considerations

When developing this product, it is crucial to keep in mind the end users, and how our design might have an overall impact on their safety when using the product. By incorporating electrical components into our overall design, we must adhere to ethical standards and disclose any components that could potentially cause harm to the users [8].

5.3.1 Overheating Hazards

LEDs can reach junction temperatures up to 80° C when operating at manufacturer-recommended currents, where junction temperature is a function of ambient environment temperature, current through the LED, and amount of heat sinking material around the LED [9]. With this last parameter in mind, we aim to encase the LEDs in a translucent heat sinking material to prevent contact with skin or other sensitive objects while still providing desired exposure to light.

5.3.2 Electric Shock Hazards

The maximum amount of current is drawn when the three LED colors (RGB) are active, powered by a 12 V supply. Each of these LEDs draws ~20 mA of current, meaning that for the color white, our LED strip draws 60 mA of current per segment. For a proposed LED with 20 segments per meter, the maximum current draw is 1.2 A/meter [10]. This amount of current is more than enough to end a human life, so it is imperative that we ground the LED strips at multiple points to ensure multiple layers of redundancy and therefore minimize risk of our end users. In accordance with OSHA 1910.304 standards, we will ensure all wiring is properly grounded and circuits are closed loop as to prevent any shocks or other electrical hazards [11].

5.3.3 Seizure Warning

Users with photosensitive epilepsy may have seizures triggered by flashing lights and bold and over intensive light patterns, both of which can be produced by the LED strips featured in our design. Flashes between 3-60 Hz are known to trigger seizures for this with underlying conditions [12]. To provide proper warning, we will include an epilepsy warning within our user interface to ensure users are aware of the potential light exposure. In addition, we will design our LED strips to operate at no less than 25000 Hz to prevent visible flashing when in use.

5.3.4 Sharp Edges

In accordance with OSHA standards 1917.112, we will take necessary precautions of protecting consumers from sharp edges and pointed corners that may lead to unintended puncture wounds [13]. Within our project, we will ensure all exposed pins and connector edges are properly soldered or bent in place to prevent scratching or puncturing.

5.3.5 User Liability

We are not responsible for any misuse or purposely harmful implementations of our project's product. This project is intended for use as a leisure device and is not intended for any other purpose. We take no responsibility for unintentional injury or harm caused by the mishandling or abuse of our device. Users should take care to avoid the above listed hazards and take necessary safety precautions to handle any component of our project.

5.4 Future work

One of the biggest pieces of future work is evaluating how to reduce the cost of our device. Our primary consideration is to evaluate replacing the LED light strips with 5V alternatives, which is cheaper and allows us to eliminate the 12V to 5V power management on the PCB. Doing so also allows us to reduce the overall PCB footprint, making the device more optimal for the end user. Investigating cheaper processors is another alternative to reduce costs, but with it brings new circuit design and software.

The application we developed was specific to Android due to the time restrictions that we were bound to, which creates a minimum viable product (MVP). We hope to recreate the application for Apple devices and publish an iOS application. While creating this new application, we want to add additional customization for the end user such as adding an RGB wheel to allow users to select their desired color or including savable user profiles to allow for multiple saved settings.

As mentioned previously, we found that our current PCB design proved somewhat cumbersome to attach LEDs to, which we would like to in future designs. The problem would be largely addressed by adding additional pinouts for the 12 V input source, allowing us to attach the LEDs' power directly to the PCB.

6 References

[1] Minger, “Minger LED Strip Lights, 16.4ft RGB LED Light Strip 5050 LED Tape Lights, Color Changing LED Strip Lights with Remote for Home Lighting Kitchen Bed Flexible Strip Lights for Bar Home Decoration,” *Amazon.com*, 24-Mar-2018. [Online]. Available: https://www.amazon.com/MINGER-Changing-Lighting-Flexible-Decoration/dp/B07JP5375R/ref=sr_1_5?dchild=1&keywords=led+light+strips&qid=1601516240&sr=8-5. [Accessed: 01-Oct-2020].

[2] E. Condes, “arduinoFFT,” *arduinoFFT - Arduino Reference*. [Online]. Available: https://www.arduino.cc/reference/en/libraries/arduinofft/. [Accessed: 03-Nov-2020].

[3] W. Smith, “The Scientist and Engineer's Guide to Digital Signal Processing” *Speed and Precision Comparisons*. [Online]. Available: http://www.dspguide.com/ch12/4.htm. [Accessed: 01-Oct-2020].

[4] W. Smith, “The Scientist and Engineer's Guide toDigital Signal ProcessingBy ,” *Speed and Precision Comparisons*. [Online]. Available: http://www.dspguide.com/ch12/4.htm. [Accessed: 01-Oct-2020].

[5] Instructables. 2014. *ATMEGA2560 Standalone Using Arduino UNO*. [online] Available at: <https://www.instructables.com/ATMEGA2560-Standalone-Using-Arduino-UNO/> [Accessed 3 November 2020].

[6] “Photosensitive Epilepsy,” *Epilepsy Society*, 21-Aug-2020. [Online]. Available: https://www.epilepsysociety.org.uk/photosensitive-epilepsy#.XkYCtWhKiUk. [Accessed: 17-Sep-2020].

[7] *WS2812B Datasheet*. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf. [Accessed: 25-Nov-2020].

[8] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 17-Sep-2020].

[9] “How are LEDs affected by heat?: LED Lighting Systems: Lighting Answers: NLPIP,” *How are LEDs affected by heat? | LED Lighting Systems | Lighting Answers | NLPIP*, May-2003. [Online]. Available: https://www.lrc.rpi.edu/programs/NLPIP/lightingAnswers/LED/heat.asp#:~:text=An LED operating in an ambient environment at,to 80°C. Junction temperature is a function of: [Accessed: 17-Sep-2020].

[10] L. Ada, T. Cooper, and T. DiCola, “RGB LED Strips,” *Adafruit Learning System*, 26-Nov-2012. [Online]. Available: https://learn.adafruit.com/rgb-led-strips/current-draw. [Accessed: 17-Sep-2020].

[11] “Department of Labor,” *1910.304 - Wiring design and protection. | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.304. [Accessed: 01-Oct-2020].

[12] “Photosensitive Epilepsy: Causes, Symptoms, and Treatment,” *WebMD*, 11-Nov-2018. [Online]. Available: https://www.webmd.com/epilepsy/guide/photosensitive-epilepsy-symptoms-causes-treatment. [Accessed: 17-Sep-2020].

[13] “United States Department of Labor,” *Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/pls/oshaweb/owadisp.show_document?p_id=10398. [Accessed: 01-Oct-2020].

Appendix A
Requirement and Verification Tables

Table 1: Requirements and Verification for Power Supply

Requirements	Verification
1. Provide a voltage output of 12V +/- 5%. 2. Support load of 0.01A to 7A of current with regulation between -3% to 5%. 3. Efficiency should be higher than 85%.	1. Connect terminals to an oscilloscope and verify that the measured voltage is within 5% of 12V. 2. Connect VDD of the constant-current test circuit depicted in Figure 4 to the DC terminal labeled in Figure 3. 1. Adjust R_s to deliver a maximum of 7A to the load, verified by a multimeter. 2. Measure the voltage across the DC terminal to ensure it never exceeds 12V +/- 5%.

Table 2: Requirements and Verification for Linear Regulator

Requirements	Verification
1. Power the ATMEGA chip by providing 3.3V +/- 2% from a 12V source. 2. Can operate between 0 - 800mA. 3. Can maintain thermal stability between up to 125℃. 4. Efficiency should be higher than 80%.	1. Connect terminals to an oscilloscope and verify that the measured voltage is within 2% of 3.3V to avoid frying the chip. 2. Connect VDD of the constant-current test circuit depicted in Figure 4 to the terminal labeled V_{out} in Figure 5. 1. Adjust R_s to deliver a maximum of 800mA to the load, verified by a multimeter. 2. Measure the voltage across the DC terminal to ensure it never exceeds 3.3V +/- 2%. 3. During steps 1 & 2, use an IR thermometer to ensure that the chip stays below 125℃.

Table 3: Requirements and Verification for Controller Module

Requirements	Verification

1. Sample at a rate of 9 kHz to obtain frequencies across the 5 different ranges (Bass, Low-Midrange, Midrange, High-Midrange, Presence). 2. Measure decibel level within +/- 5% of real decibel levels.	1. Generate a sample audio file with sounds within each of the frequency thresholds, spanning from 0 kHz to 4.5 kHz. Store the resulting output file and verify that the files are identical. 2. Use the same audio sample on a Python FFT simulation to examine decibel levels and validate that they are within 5% of the original sample.

Table 4: Requirements and Verification for PWM Circuit

Requirements	Verification
1. Reflect changes in sound intensity by scaling the amount of light intensity from black to white spectrum. 2. Generate separate color outputs corresponding to whether audio is from the left speaker or right speaker. 3. Send data at speeds of up to 800Kbps.	1. Generate sound input with increasing levels of sound intensity. Verify that light intensity increases directly in accordance with sound intensity. 2. Pick songs like Queen's Bohemian rhapsody that alternate audio from left to right. Generate an input signal into the PWM controller to the LEDs at 800kbps/(2 channels * 16 bits) = 25000 Hz or 25 kHz [17]. 3. Attach a multimeter to the line between the PWM circuit and the LEDs and set the multimeter to Ṽ. verify the corresponding frequency <25 kHz.

Table 5: Requirements and Verification for Bluetooth Receiver

Requirements	Verification
1. Communicate with UART. 2. Transmission is received up to 80m from the user.	1. Develop a prototype app that sends a signal to the receiver, and probe the V_{in} of the PWM circuit using an oscilloscope to determine if a pulse was registered. 2. Using the same application, send a signal to the receiver, located 80m away, and probe the V_{in} of the PWM circuit to see if it reflects a pulse, using an oscilloscope.

Table 6: Requirements and Verification for LED Strips

Requirements	Verification
1. Receive data at speeds of up to 800Kbps to maintain color.	1. Generate a signal from the PWM controller to the LEDs at 800kbps/(2 channels * 16 bits) = 25000 Hz or 25 kHz. We should see transitions between light colors without flickering [17].

2. Can maintain thermal stability between -25℃ and 80℃. 3. LEDs should not visibly flicker and be able to be photographed.	2. During step 1, use an IR thermometer to ensure that the LEDs operating juncture never exceeds 80℃. 3. Generate a signal from the PWM controller to the LEDs at 1400 kbps/(2 channels * 16 bits) = 25000 Hz or 25 kHz. We should be able to record video of transitions between light colors without flickering [17].

Table 7: Requirements and Verification for Application

Requirements	Verification
1. Allow users selection of 5 unique themes for their LED lights. 2. The user interface should be intuitive and encourage use. 3. The interface sends out Bluetooth signal at 9600 baud. 4. Communicate with UART.	1A. Select the bass theme for the light colors. 1B. Prepare a song sample that has a significant bass component and play it. 1C. Visually inspect the LEDs to verify that they match the color of the theme selected. 1D. Repeat for Low Midrange, Midrange, High Midrange, and Presence. 2A. Develop a prototype app that sends a signal to the receiver. 2B. Attach a multimeter to the line between the Bluetooth receiver and the V_{in} terminal of the PWM circuit. 2C. Set the multimeter to $\tilde{V}$ and verify corresponding the frequency = 9600 Hz. 3. Develop a prototype app that sends a signal to the receiver and probe the V_{in} of the PWM circuit using an oscilloscope to determine if a pulse was registered.

Appendix B
Part Prices

Table 1: Part Price List

Part Name	Description	Manufacturer	Part Number	Quantity	Part Cost
ATMEGA2560	8-bit AVR RISC-based microcontroller	Microchip	ATMEGA2560-16AUR	2	$11.99
Sparkfun TRRS 3.5 mm Jack Breakout	3.5 mm audio jack	SparkFun	TRSS_Breakout_v1	1	$3.95
3.5 mm Headphone Jack Splitter	1 female to 2 male 3.5 mm headphone splitter	Koopao	M4S-BR	1	$5.99
BTF Lighting WS2815	144/m individually addressable LED strip	BTF Lighting	WS2815B	2	$29.88
Adafruit Bluefruit LE UART Friend - Bluetooth Low Energy (BLE)	A Bluetooth slave for use communicating between two microcontroller systems	Adafruit	nRF51822	2	$17.50
12 V Power Supply	120 W, 12 V switching power supply.	Shenzhen Yinghui Yuan Electric Co.	YHY-12005000	2	$5.00

12V to 3.3 V Low-Dropout Linear Regulator	A board that accepts up to 15 V and outputs a fixed voltage of 3.3 V	Texas Instruments	LM1117	2	$1.11
Resistor Set	Resistor set for circuit building	ECE Department	Various	1	Already Acquired
Capacitor Set	Capacitor set for circuit building	ECE Department	Various	1	Already Acquired
Crystal Oscillator	Oscillator for ATMega	TXC CORPORATION	9B-16.000MBBK-B	4	$0.30
Reset Button	Square 12 mm button for debugging	SparkFun	COM-09190	1	$0.50
Catch Diode	A catch diode is used to eliminate flyback	Diodes Inc.	DFLS240L	4	$0.42
5V Switch-Mode Converter	An IC that efficiently steps voltage down from 12 V to 5 V.	Linear Technology	LT3685	4	$9.57
TOTAL (pre-tax)					**$182.56**

Appendix C
Power Insights

Figure 1: Noise on the Switch-Mode regulator

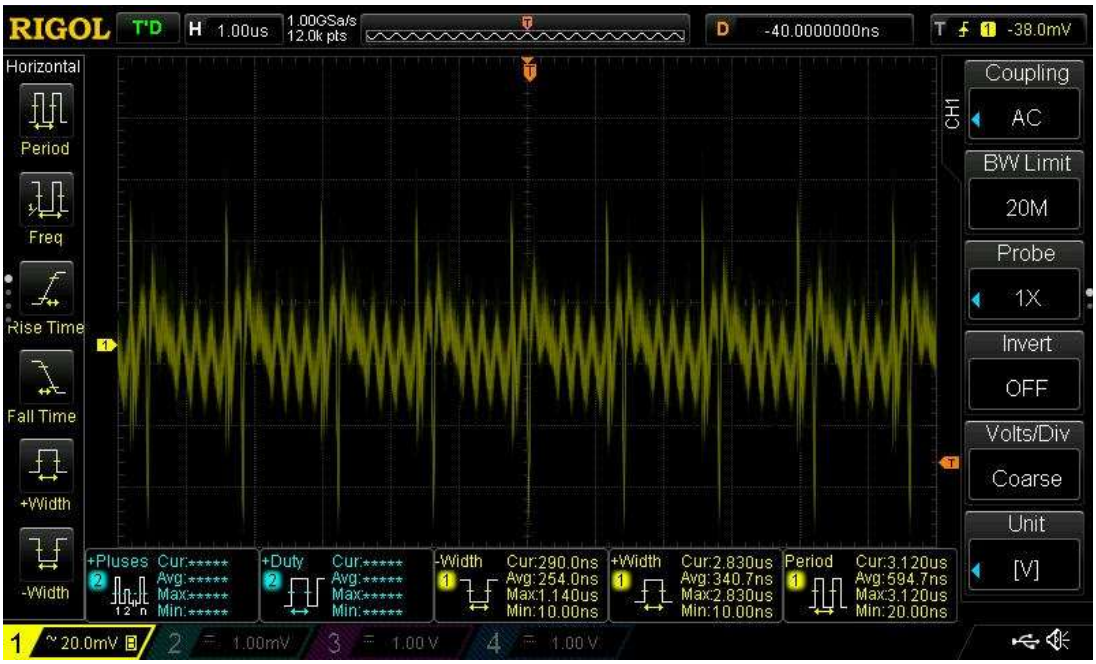

Figure 2: Noise on the 3.3V Linear Regulator

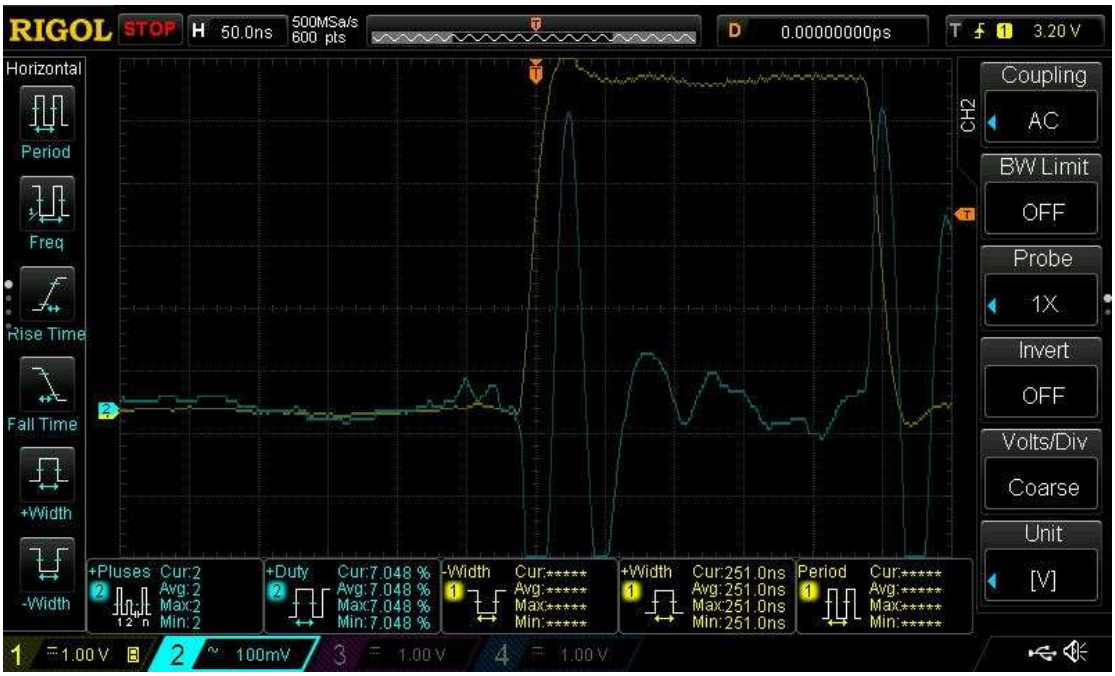

Figure 3: Audio to data (audio blue, data yellow) - data is aligned

Figure 4: Bigger picture - data going along with audio