# IMUsic

By

John Born

Kuang Wang

Miguel Torres

Final Report for ECE 445, Senior Design, Fall 2019

TA: Kristina Miller

11 December 2019

Project No. 1

## Abstract

IMUsic is a set of wireless wearable devices meant to provide dancers and musicians with the ability to create live music through physical movement. These devices strap onto a performer's limbs, record orientation data, and generate music using this data through a virtual synthesizer environment. Various musical characteristics such as pitch, volume, panning, and background music may be controlled by these devices. This report details the engineering challenges, design processes, and technology used to develop IMUsic.

## Contents

## 1. Introduction

Choreographed movement is known to enhance the musical performance experience. However, due to conflicting physical requirements, kinetic performers—such as dancers—are unable to contribute to the music composition process. For instance, many musicians need trained dexterity to play their instrument, thereby limiting their range of motion. Their range of motion is also limited since they must maintain a close proximity to their instrument. In contrast, the dancer needs freedom of motion throughout the performance space, and therefore must remain untethered. Making use of modern motion sensors, particularly Inertial Measurement Units (IMUs), performers can use orientation data to play virtual instruments, plugging the performer into the music composition process.

### 1.1 Objective

The objective of the IMUsic project is to use a performer’s movement to play a virtual instrument through a set of wearable wrist and ankle controllers. The orientation data of the controllers is used to control the instrument’s sonic characteristics, such as note, volume, pan location, and timbre. Mapping schemes for the data will be detailed in Section 2.3.2. The data is transmitted wirelessly over a localized Wi-Fi network using the Open Sound Control (OSC) protoco —a URL style transmission—to a laptop hosting the virtual instrument. An overview of the system can be seen in Figure 1.

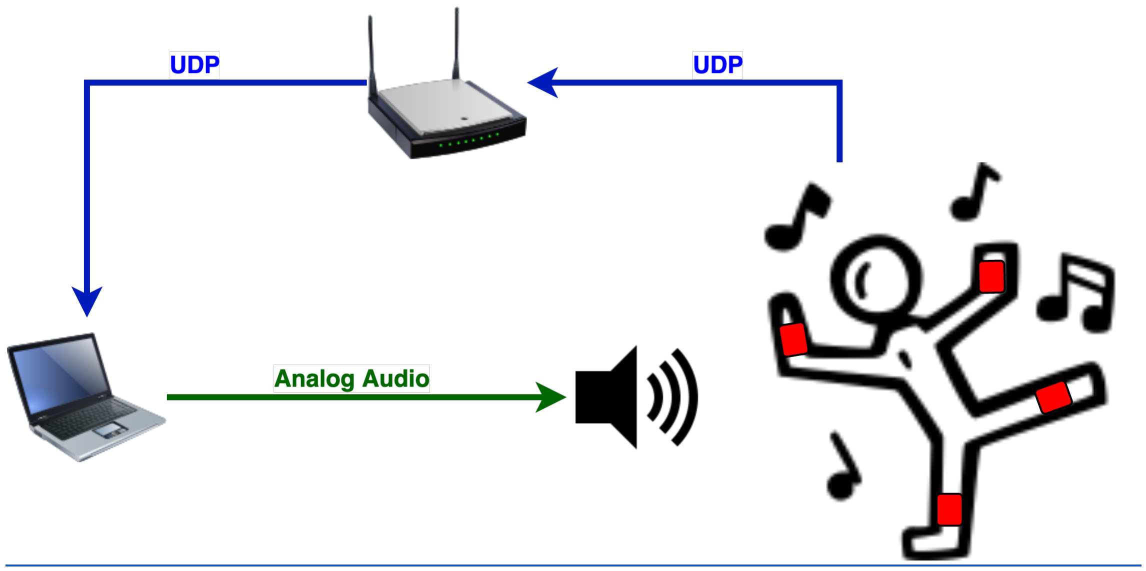


Figure 1 Diagram of the overall system functionality. The system allows for a feedback loop of creativity, where the dancer directly impacts the music being played while the music impacts the way the dancer moves.

## 1.2 Subsystem Overview

The IMUsic controller system is divided into three subsystems: Power, Control, and Wi-Fi, which can be viewed in Figure 2. The Power Subsystem is comprised of a Lithium-Polymer (Li-Po) battery supplying 3.7 V to a voltage regulator which steps the voltage down to 3.3 V and minimizes any fluctuations from the battery. The Control Subsystem consists of an IMU and a Wi-Fi enabled Microcontroller Unit (MCU). The MPU-9250 IMU utilizes 3 sensors: an accelerometer, a gyroscope, and a magnetometer [1], to track orientation information—in the form of Quaternions—and transmits the data via an Inter-Integrated Circuit ($I^2C$) bus to the MCU. The sensor data is translated to pitch, roll, and yaw Euler Angles, then packaged for transmission, according to OSC messaging requirements, by the MCU—the ESP32 [2]—over a wireless User Datagram Protocol (UDP) connection. The Wi-Fi Subsystem links the IMUsic controllers to a computer via a local Wi-Fi network facilitated by a consumer-grade Wi-Fi router. The computer receives the OSC messages from the network and are decoded by SuperCollider, the virtual instrument environment. The instrument mapping, described in Section 2.3.2, is done within SuperCollider and audio output is generated for the speakers.

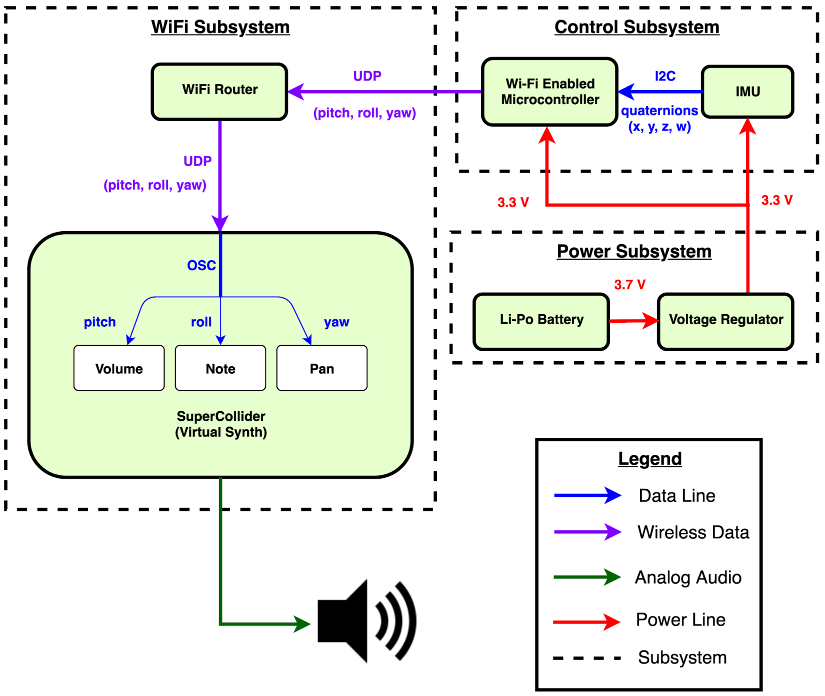


Figure 2 Block Diagram illustrating each subsystem of the IMUsic controllers.

## 2. Design

For our prototypes, seen in Figure 3, we used a breakout board, the ESP32 Thing designed by SparkFun, which included a built-in antenna for the MCU. Our first PCB layout was designed with the idea of housing the breakout board to integrate all of the components onto one board. However, when the breakout board was situated on the PCB, connectivity to the wireless network could only be established if the antenna was physically touched. With the limited knowledge of radio frequency theory, we suspected that the issue was due to a change of impedance when the breakout board was attached to the PCB. Given that connectivity to the wireless network could only be established through touch, we attempted to tune the antenna by soldering a 0.1 μF capacitor in parallel with no success. However, the accidental removal of the capacitor severed the connection to the antenna and fixed our connectivity problem. We are still unclear as to why this occurred. Alternatively, our final design, which can be seen in Figure 4, used the ESP32 WROVER-B module, developed by Espressif, did not experience this issue.

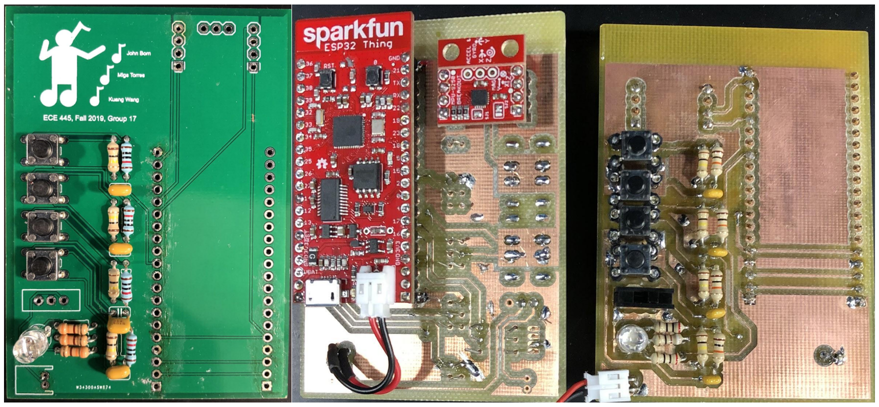


Figure 3 Pictures of the first set of prototype PCBs which house the SparkFun breakout boards for the MCU and IMU. The picture on the left was printed by PCBWay, while the picture on the right was printed by the ECE Electronics Service Shop

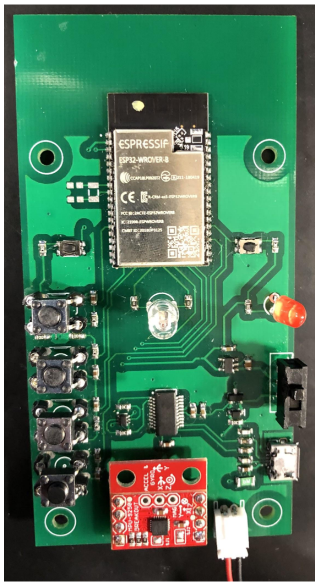


Figure 4 Picture of the final PCB design which uses the ESP32 WROVER module. The space where the antenna is situated has no metallic parts intentionally, as was suggested by the ESP 32 Hardware Design Guidelines [3].

## 2.1 Power Subsystem

The power subsystem consists of a Li-Po battery and voltage regulator. The Li-Po battery delivers power to a voltage regulator which, in turn, outputs a regulated 3.3V for all circuit components.

### 2.1.1 Li-Po Battery

Our controllers are using a lightweight, 400 mAh, rechargeable Li-Po battery pack rated at 3.7 V. Additionally, the small form factor of the battery adds a negligible amount of weight. Using equation (1) we can calculate how much charge we need to power the controllers for a duration of 2 hours.

$$\text{mAh} = \text{mA} \times \text{seconds} \div 3600 \times 1000 \qquad (1)$$

### 2.1.2 Voltage Regulator

The IMUsic controllers use the AP2112K 3.3 V voltage regulator to smooth out the power supplied by the Li-Po battery as well as maintain a constant 3.3 V for the ESP32 and the MPU-9250.

## 2.2 Control Subsystem

The control subsystem consists of the IMU and Wi-Fi-enabled microcontroller. This is the device that takes in the user’s orientation data and sends it to the virtual synthesizer.

### 2.2.1 IMU

The IMU being used is the MPU-9250 which is a 9-Axis Gyroscope, Accelerometer, and Magnetometer chip. It collects inertial data (angular velocity, angular acceleration, and geomagnetic field strength) from each subsensor and compiles these into orientation data output as time-stamped quaternions. The output quaternions are computed using a proprietary on-chip sensor fusion algorithm called MotionFusion. This orientation output is used to control the virtual synthesizer.

### 2.2.2 Wi-Fi Enabled Microcontroller

The microcontroller being used is the ESP32. This is a Wi-Fi capable MCU for use with Internet of Things (IoT) devices. This microcontroller is used to take in data from the IMU sensor, package the data for transmission according to OSC protocol requirements, and transmit that data with its onboard antenna. The ESP32 complies with the IEEE 802.11b/g/n standards allowing us to theoretically transmit data at 150 Mbps [2].

## 2.3 Wi-Fi Subsystem

### 2.3.1 Wi-Fi Router

A consumer grade wireless router is used to facilitate a local wireless network for communication between the controllers and the computer. The router is capable of transmitting data at rates closest to the theoretical limit of 150 Mbps and ensure the transfer rate is greater than 100 (+/- 10) Mbps.

### 2.3.2 SuperCollider (Virtual Synth)

This is an audio synthesis IDE that uses class-based, object oriented programming to facilitate algorithmic processing of audio. We use this IDE to custom build a software synthesizer whose sonic characteristics are controlled by IMU data. The software allows us to directly map pitch, yaw, and roll information received from each device to the software synthesizer's note frequency, volume, modulation, filter cutoffs, panning, and other possible controls. An example of the mapping is depicted in Figure 5.

### 2.3.3 Instrument Mapping

The mapping schemes we decided to use for the first iteration of the virtual instrument includes: pitch data to filter cutoff, roll data to note, and yaw data to FM modulation rate. A linear mapping to these sonic characteristics is established using Equations (2), (3), (4), and (5) where the range of the IMU data (-180 degree to 180 degree) is transformed to the ranges seen in Table 1.

$$p_{min} = c_{min} m + b \tag{2}$$

$$p_{max} = c_{max} m + b \tag{3}$$

$$m = \frac{p_{max} - p_{min}}{c_{max} - c_{min}} \tag{4}$$

$$b = \left(\frac{p_{min} c_{max} - p_{max} c_{min}}{c_{max} - c_{min}}\right) \tag{5}$$

Table 1 Instrument parameters and their ranges

| Parameter | Range |
|---|---|
| Note | 2 Octaves |
| Sound Amplitude | -60 dB – -3 dB |
| Stereo Panning | Left – Right |
| Low-pass Filter Cutoff Frequency | 50 Hz – 15 kHz |
| Modulation Rate | 0.01 Hz – 220 Hz |
| Modulation Amount | 0 Hz – 200 Hz |

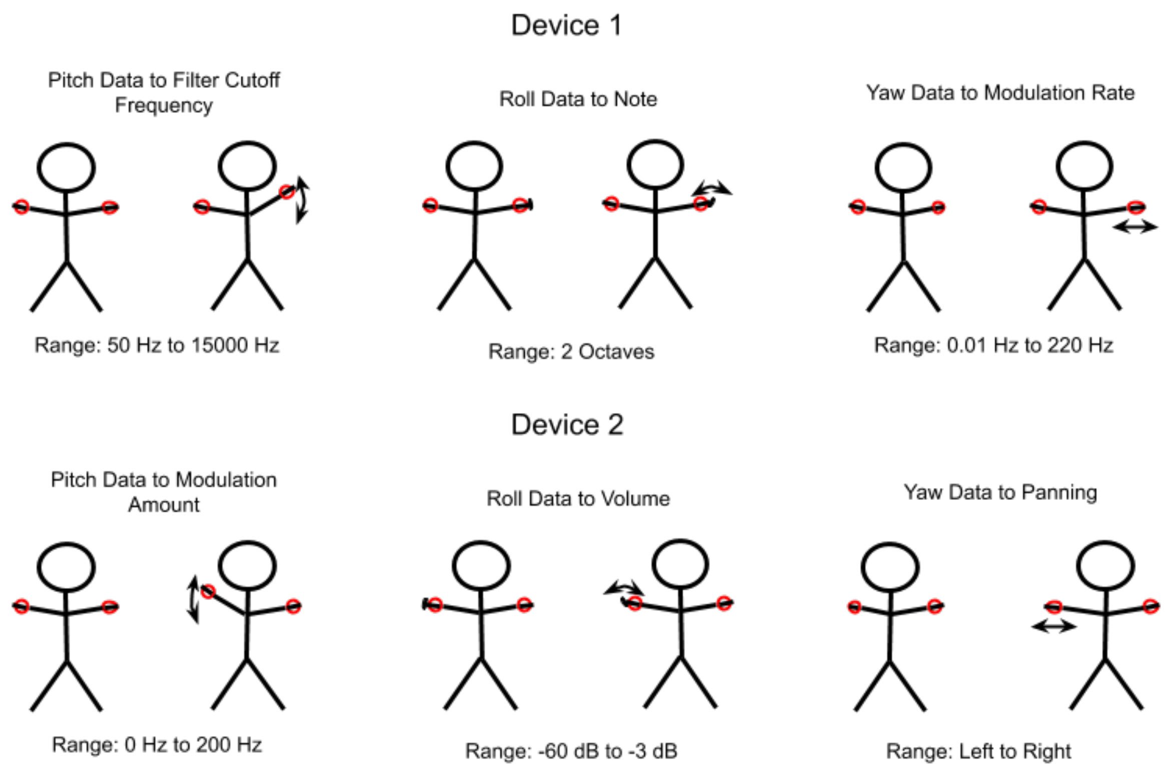


Figure 5 A graph depicting one of the possible mapping schemes and their ranges using 2 devices.

## 2.4 Orientation Representation

### 2.4.1 Euler Angles

IMUs are able to output orientation data in a variety of formats. Some of the most popular orientation representations include Euler angles, quaternions, and axis-angle representations. These all act on different geometric and mathematical principles. However, IMUsic required an orientation representation that is intuitive for the users. We decided to use Euler angles since they have intuitive physical representations. Euler angles function by having 3 axes of rotation whose rotations are commonly referred to as pitch, yaw, and roll (in the flight community). Pitch can be thought of as turning the nose of an airplane upward or downward. Yaw can be thought of as turning the nose of an airplane left or right. Roll can be thought of as tilting the wings of an airplane downward or upward. In contrast to quaternions (which represent orientation using 4-dimensional vectors) and axis-angle rotations (which represent orientation as rotations about an arbitrary axis), Euler angles are more intuitive in physical situations. The MPU 9250 library we decided to use output orientation as a quaternion. We translated this quaternion to Euler angle representation using the following equations ($roll = \phi, pitch = \theta, yaw = \psi$).[4]

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \mathrm{atan2}(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \mathrm{asin}(2(q_0 q_2 - q_3 q_1)) \\ \mathrm{atan2}(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \quad (6)$$

### 2.4.2 Mapping to Sound Characteristics

In order to map Euler angles to sonic characteristics, it is important to ensure that any discontinuities resulting from changes in orientation of the controllers do not create jarring audio jumps. Since Euler angles are range limited at +/-180 degrees (see Figure 6), this would cause a noticeable jump in note change as the lowest note is played when the orientation in the roll axis reaches -180 degrees and the highest note is played at a 180 degree orientation. We accounted for these boundaries between edges using a circular mapping implemented by taking the absolute value of the Euler angles (see Figure 7).

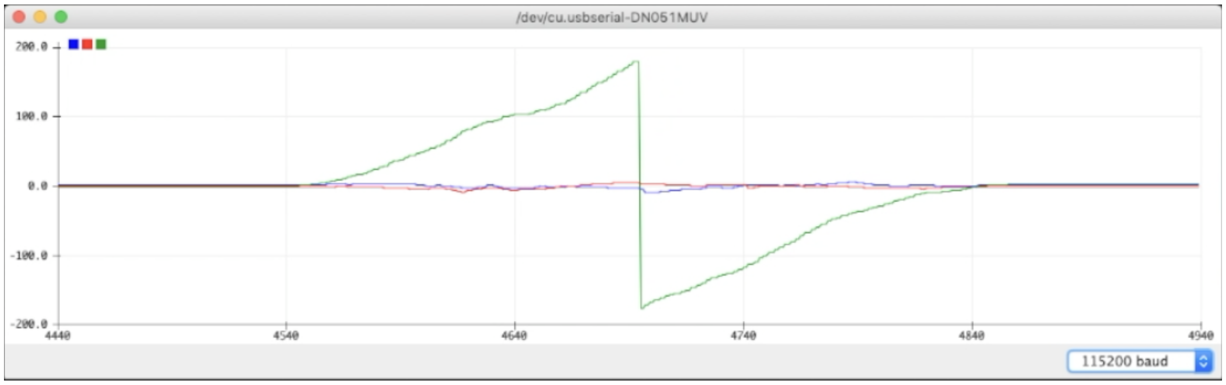


Figure 6 A screenshot of the Arduino's plotter depicting the change in orientation about the yaw axis.

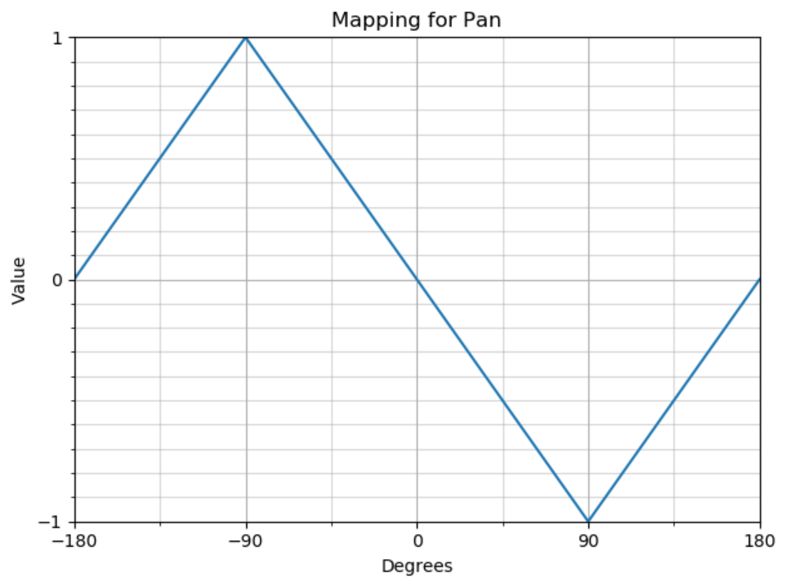


Figure 7 A graph of the circular mapping for pan, where a value of 1 is hard left and a value of -1 is hard right.

## 3. Design Verification

### 3.1 Power Subsystem

#### 3.1.1 Li-Po Battery

**Requirement:** A typical musical performance lasts about 2 hours, so it is necessary that our controllers last at least 2 hours with one battery.

**Verification:** We connected one of the controllers to a lab bench power supply and recorded the amperage draw for 6 seconds.

**Results:** The controller draws an average of 103.45 mA, as can be seen in Figure 8. Using equation (1) we calculate that 172.8 mAh are required, allowing the controllers to operate for about 2.3 hours on a full charge.

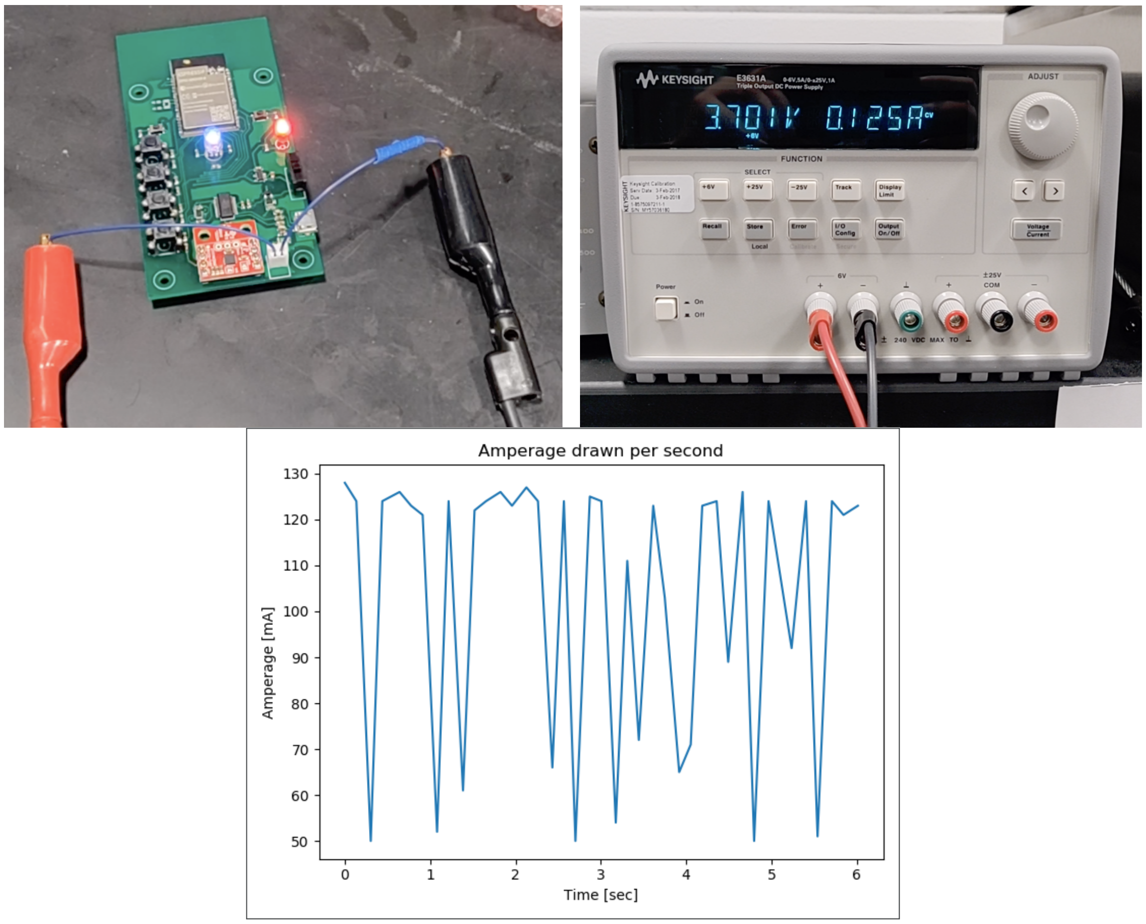


Figure 8 Pictures of the IMUsic controller (on the left) being powered by a lab bench power supply (on the right) set to 3.701 V and a graph of the amperage drawn by the controller in a 6 second period (bottom).

#### 3.1.2 Voltage Regulator

**Requirement:** We require that the AP112K delivers a steady 3.3 V to the Control Subsystem, regardless if it is connected to a battery or USB.

**Verification:** We connected one of the controllers to a battery and probed the output of the voltage regulator. We repeated this with the controller connected to USB power.

**Results:** The AP2112K 3.3 V voltage regulator steps down the Li-Po battery voltage to a steady 3.3 V, which can be seen in Figure 9, as is required by the IMU and the MCU.

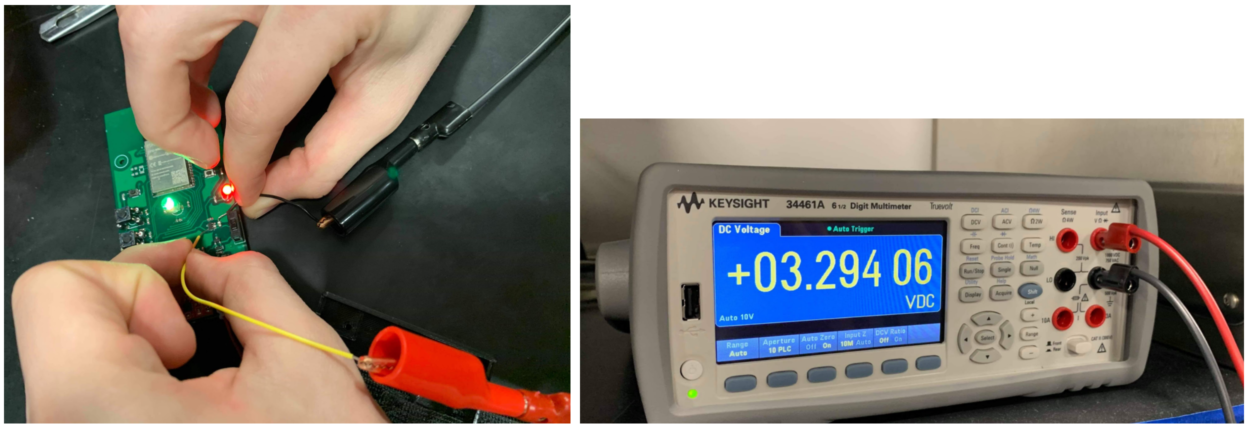


Figure 9 Measuring the voltage output of the AP2112K 3.3 V voltage regulator from the IMUsic controller (on the left) with a lab bench digital multimeter (on the right).

## 3.2 Control Subsystem

### 3.2.1 IMU

**Requirement 1:** We require that the IMU is able to maintain the correct orientation value for each axis to within +/- 10 degrees after a 180 degree rotation.

**Verification 1:** We rotated the controller by 180 degrees on a single axis and used the Arduino IDE plotter to verify that the change in orientation is maintained to within the required range. Then, we rotated the controller back to its original position, or by 180 degrees in the opposite direction, and verified that a +/- 10 degree accuracy was maintained once again. This was repeated on each axis.

**Results:** The IMU successfully reported accurate measurements to within the desired range as can be seen in Figure 10.

**Requirement 2:** We require that the IMU limits the amount of noise on each axis to within +/- 1 degree.

**Verification 2:** We set the controller down for 10 seconds and observed the readings through the Arduino IDE plotter.

**Results:** There were no significant changes in the readings of each axis as can be seen in Figure 11.

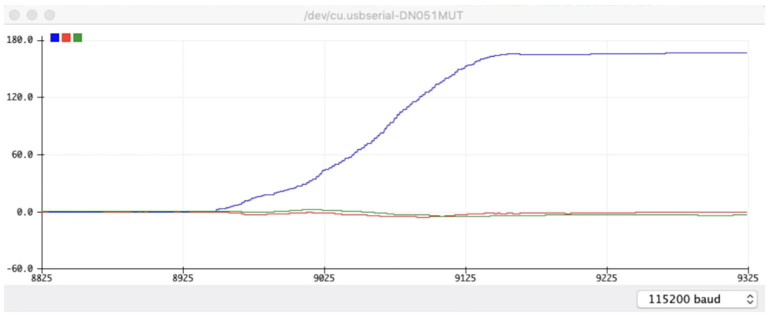


Figure 10 The Arduino IDE plotter demonstrates a 180 degree rotation about the roll axis.

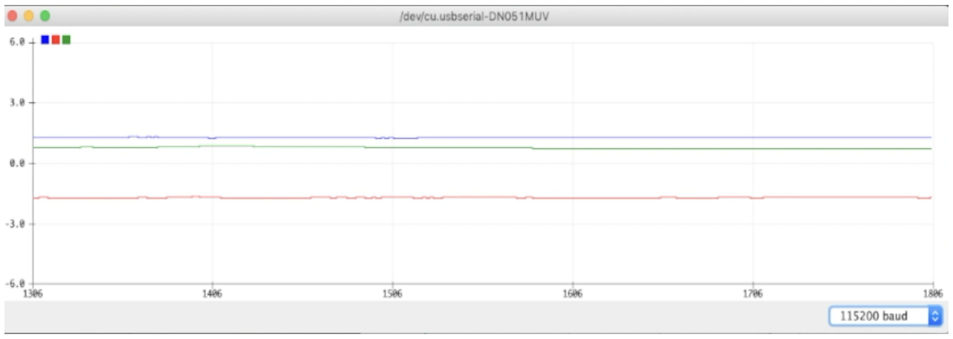


Figure 11 The Arduino IDE plotter displaying the readings of the IMU as the controller is sitting still.

### 3.2.2 Wi-Fi Enabled Microcontroller

**Requirement:** We require that the MCU is able to apply smoothing algorithms while maintaining any latency to a minimum so as not to disrupt the user experience. According to the MIDI Manufacturers Association, any latency below 10 ms is practically unnoticeable [5].

**Verification:** Load code on ESP32 that activate the DMP fusion function and display the raw IMU data as well as the smoothly fused orientation data with timestamp on therial monitor. Check the average latency between the IMU data and orientation data is lower than 10ms.

**Result:** The numerical results are not consistent, therefore unreliable. However, when using the instrument by moving it around and composing with the additional buttons, there is no perceivable delay. Rhythmic elements are easily triggered in time with SuperCollider's internal metronome.

**Requirement:** Can package pitch, yaw, and roll data according to OSC protocol standards.

**Verification:** Load code on ESP32 that sends orientation data from IMU to a serial monitor according to the OSC protocol. Read the output data from the monitor and ensure all three orientations of yaw, pitch, and roll are output as 32-bit float values.

**Result:** all three orientations of yaw, pitch, and roll are output as 32-bit float values on the monitor

## 3.3 Wi-Fi Subsystem

### 3.3.1 Wi-Fi Router

**Requirement:** Since this is a consumer grade wireless router, the only requirement needed is that it facilitates wireless communication between the controllers and the computer.

**Verification:** We plugged in the router, performed any firmware updates needed and setup a Wi-Fi network.

**Results:** The router performed as expected.

### 3.3.2 SuperCollider (Virtual Synth)

**Requirement 1:** We require that the virtual instrument is able to respond to external signals.

**Verification 1:** We setup the instrument to respond to the laptop's trackpad by controlling the playable note with the x-axis and volume with the y-axis. Therefore, moving the mouse pointer right will increase the pitch, moving left will decrease the pitch, moving up will increase the volume and moving down will decrease the volume.

**Results:** The results were successful, the instrument reacted as expected. This allowed us to swap the mapping to test other sonic characteristics, 2 at a time.

**Requirement 2:** We require that the parameter mapping ranges make perceivable sonic adjustments as dictated by movement. For instance, a rotation about the Roll axis will change the note that is played by the instrument, or a rotation about the Yaw axis will pan the instrument from left to right.

**Verification 2:** To verify the mappings we used SuperCollider's oscilloscope, frequency scope, and meters as well as our ears visually and audibly confirm that instrument note changes, modulations, filter processing, and panning all occurred.

**Results:** Audible changes to the instrument's parameters were detected and visually confirmed, see Figure 11. This allowed us to fine tune the ranges of each characteristic see Table 1 in Section 2.3.3.

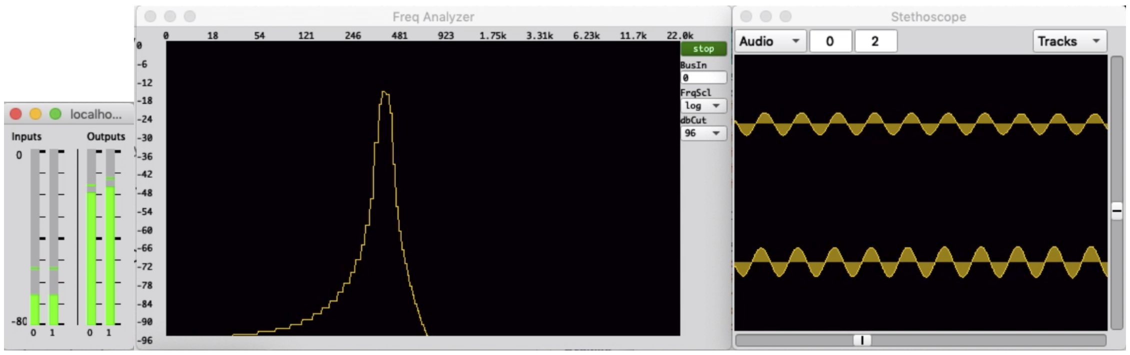
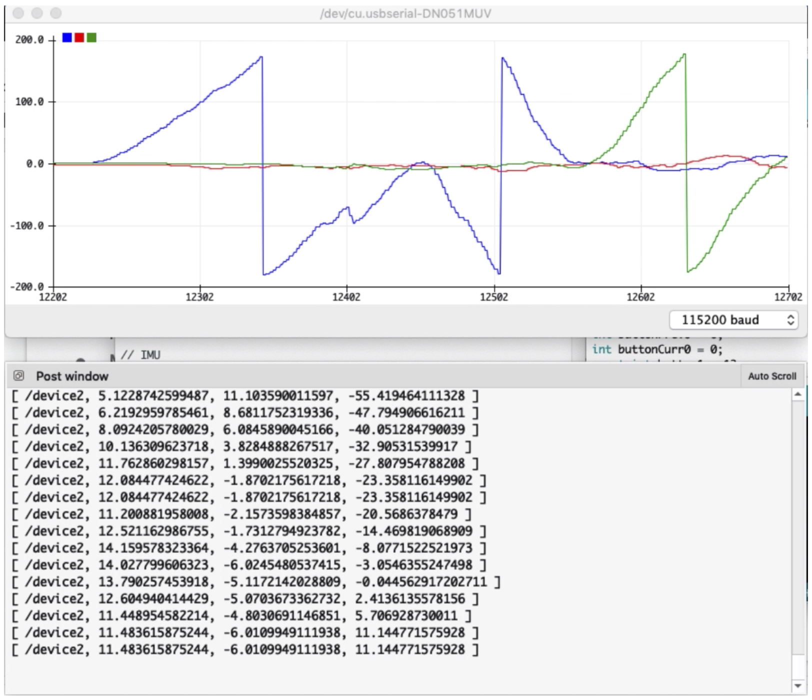


Figure 11 SuperCollider's meter (left), frequency scope (center), and oscilloscope (right).

**Requirement 3:** We require that the synthesizer receives and responds to incoming OSC messages.

**Verification:** We setup SuperCollider to post the incoming OSC messages to the monitor, see Figure 12.

**Results:** Messages were posted with no noticeable delay.



```
[ /device2, 5.1228742599487, 11.103590011597, -55.419464111328 ]
[ /device2, 6.2192959785461, 8.6811752319336, -47.794906616211 ]
[ /device2, 8.0924205780029, 6.0845890045166, -40.051284790039 ]
[ /device2, 10.136309623718, 3.8284888267517, -32.90531539917 ]
[ /device2, 11.762860298157, 1.3990025520325, -27.807954788208 ]
[ /device2, 12.084477424622, -1.8702175617218, -23.358116149902 ]
[ /device2, 12.084477424622, -1.8702175617218, -23.358116149902 ]
[ /device2, 11.200881958008, -2.1573598384857, -20.5686378479 ]
[ /device2, 12.521162986755, -1.7312794923782, -14.469819068909 ]
[ /device2, 14.159578323364, -4.2763705253601, -8.0771522521973 ]
[ /device2, 14.027799606323, -6.0245480537415, -3.0546355247498 ]
[ /device2, 13.790257453918, -5.1172142028809, -0.044562917202711 ]
[ /device2, 12.604940414429, -5.0703673362732, 2.4136135578156 ]
[ /device2, 11.448954582214, -4.8030691146851, 5.706928730011 ]
[ /device2, 11.483615875244, -6.0109949111938, 11.144771575928 ]
[ /device2, 11.483615875244, -6.0109949111938, 11.144771575928 ]
```

Figure 12 Arduino IDE’s plotter (top) showing rotations about the Roll axis (blue) and Yaw axis (green) in conjunction with SuperCollider’s Post Window (bottom) showing incoming IMU data as OSC messages.

## 4. Costs

### 4.1 Parts

The following part costs are for the prototype devices created. The parts cost is comprised of one-time purchases totaling $51.64 and per-device purchases of $55.25. Thus, for 4 devices (one for each limb) the total system cost would be $272.64.

Table 2: Project Cost (One-Time Purchase)

| **Part** | **Part #** | **Manufacturer** | **Actual Cost($)** |
|---|---|---|---|
| Li-Po Charger (Micro-USB) | PRT-10217 | SparkFun Electronics | $8.95 |
| Wi-Fi Router | Linksys - AC1000 | Linksys | $39.99 |
| Micro-USB Cable | Belkin 4-Foot MIXIT | Belkin | $2.70 |
| **TOTAL** | | | $51.64 |

Table 3: Project Cost (Per Device)

| **Part** | **Part #** | **Manufacturer** | **Actual Cost($)** |
|---|---|---|---|
| Li-Po Battery - 850mAh | PRT-13854 | SparkFun Electronics | $4.95 |
| Voltage Regulator - 3.3V | AP2112K-3.3V | Texas Instruments | $1.14 |
| MPU-9250 Breakout | SEN-13762 | SparkFun Electronics | $14.95 |
| ESP32 WROVER-B | 1904-1007-1-ND | Espressif | $4.20 |
| PCB | N/A | PCBWay | Approx. $2.00 |
| Jumper Wire Kit (140 Pieces) | PRT-00124 | SparkFun Electronics | $5.95 |
| Enclosure | N/A | 3D Printed (UIUC Makerlab) | $8.75 |
| Elastic Adjustable Sport Armband Strap | N/A | Glo-shine | $6.19 |
| Power LED | 160-1967-ND | Lite-On Inc. | $0.05 |
| Power Switch | CWI345-ND | CW Industries | $0.66 |

| | | | |
|---|---|---|---|
| RGB LED | 1830-1015-ND | Inolux | $1.01 |
| Buttons (x4) | 450-1650-ND | TE Connectivity | $0.10 x 4 = $0.40 |
| Other Components | N/A | N/A | Approx. $5.00 |
| **TOTAL** | | | $55.25 |

### 4.2 Labor

The labor cost is based on a $40 per hour rate. The time dedicated to each project phase is estimated below. Thus, the total labor cost can be approximated at 230hrs * $40/hr = $9200.

Table 4: Labor Cost

| **Project Phase** | **Estimated Time (Hours)** |
|---|---|
| Design and Preparation | 60 |
| Ordering and Initial Prototyping | 20 |
| PCB Design | 20 |
| Enclosure CAD Design | 10 |
| Finalize PCB and CAD Designs | 15 |
| Software Synth Coding | 40 |
| Testing and Debugging | 40 |
| Poster and Presentation Preparation | 15 |
| Final Design Documents | 10 |
| **TOTAL** | 230 |

# 5. Conclusion

## 5.1 Accomplishments

Our project accomplishes all of the high-level requirements including outputting smoothing sound without spikes and discontinuities, providing more than four controllable musical characteristics of the virtual music instruments and transmitting orientation data wirelessly with latency smaller than 10 ms. beyond those requirements, we also add LED components responding to the orientation data and a set of buttons that can turn on/off the music synthesizer or add different background music. We present our devices during the demo, presentation and an instrument fair for second-grade children. We also invite a dancer (friend of Migs) to perform traditional Scottish dancing. Most of the audience and reviewers accept our concept and enjoy creating music with our devices. Users are able to quickly notice how the orientation of the devices controlling the music components and try using their movement to change the output audio. Even children can notice certain body movements create certain patterns of music and repeat the movement to recreate the sound.

## 5.2 Uncertainties

At the same time, we received many suggestions for improvement from the users and observe several issues about the current version of the devices. The one of the suggestions we received from course TA Amr is that adding some quantized control inputs. Currently our devices only use raw orientation data and the data are linearly mapped to the music components. Because of the continuity of the orientation data, we have no quantized or discrete data to control the music components. Most of the music instruments have discrete inputs such as keys and strings. By adding quantized control inputs such as number of rotation, certain gestures may bring a better experience of music composition to our device. Another observation we found during our experiment with the dancer performing Scottish dancing is that the sound produced from our device is not well fit to the traditional Scottish music. The biggest reason is that Scottish dance involving a lot of rapid movement of feet, which add too much rapid change of orientation as well as the output sound. Again quantized inputs may solve part of this problem and more research on dancing movement is required for design a better control inputs. The final uncertainty is the antenna issue. We solve this problem by accidentally sniping antenna connection on PCB, but we still have not figured out the actual reason for that.

## 5.3 Ethical considerations

We think there are minimal chances to have safety issues for this project. However, we still have two potential safety issues to keep in mind for both the design and development process of the project. First, since we use a LiPo battery as a power supply, we need to make sure the battery is connected correctly with each component requiring a power supply. This is crucial in order to avoid conditions such as a short circuit or where too much current is being drawn where the potential of causing severe burns becomes inevitable. According to the International Consumer Electronics Show (CES), the damage threshold for skin damage is defined as 43°C, so our device should keep its operating temperature much more below the threshold [6]. Second, we need to make sure the enclosure of the devices can protect users' limbs from scratching or cutting as well as ensuring comfort for extended wear.

Our design is such that the project complies with all of the Ethics requirements mentioned in the IEEE Code of Ethics [7]. There are several aspects we are primarily focusing on for the project. The IEEE Code of Ethics #5 states, "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems" [7]. We believe our project satisfied this requirement because our project goal is to develop wearable devices for software instruments, which let not only dancers but also people without music composition experience to enjoy and compose music easier. We hope that with a combination of hardware and software technology, we can provide a better way for individuals to interact with music composition. The IEEE Code of Ethics #7 and #10 mentioned about how we accept criticism from others and assist co-workers in their professional development [7]. Just like how the ECE455 course designed, we worked as a group with assistance from the course's staff. We accept any honest criticism from the course's staff, to acknowledgment and to credit each person's contribution to the project. Moreover, we assist each other to make sure we all get progress in professional development and get the project accomplished.

## 5.4 Future work

As we mentioned previously in section 5.2, it requires more researches and experiments about the quantized control input and the design of the mapping function. Besides that, we learned that the enclosure is cranky and we need to further reduce the size of the devices to achieve a better user experience. In the software aspect, we can improve the graphical user interface by replacing current base code into a compiled program or application allowing users without coding knowledge to modify the mapping functions as well as other control options of the synthesizers. Finally, the devices require more improvement and expansion of different synth sound options. To achieve this more research and experiments are required. Dancing movement, instrumental music and quantization of control input are possible topics of research we may conduct in future. Also, we can work with musicians and dancers to improve the overall quality and design of our devices.

## References

[1] InvenSense, “MPU-9250 Product Specification Revision 1.1,” MPU9250 datasheet, Jun. 2016

[2] Espressif Systems, “ESP32 Datasheet,” ESP32 Datasheet, Oct. 2016

[3] Espressif Systems, “ESP32 Hardware Design Guidelines,” ESP32 Datasheet Version 2.9, 2019

[4] “Conversion between quaternions and Euler angles”, Wikipedia.org. (2019) Online https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles [Accessed: 11 - Dec. -2019]

[5] *The complete MIDI 1.0 detailed specification: incorporating all recommended practices*. MIDI Manufacturers Association, 2006.

[6] “Design Safe Wearable Technology with Heat Transfer Modeling” C. Brianne . [Online]. Available: https://www.comsol.com/blogs/design-safe-wearable-technology-with-heat-transfer-modeling/ [Accessed: 11 - Dec. - 2019]

[7] “IEEE Code of Ethics”, Ieee.org. (2019) Online https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 11 - Dec. - 2019] [Orientation] “Orientation (Geometry), Wikipedia.org, (2019) Online https://en.wikipedia.org/wiki/Orientation_(geometry) [Accessed: 11 - Dec. - 2019]

## Appendix A Requirement and Verification Table

Table 5 System Requirements and Verifications

| Requirement | Verification | Verification status (Y or N) |
|---|---|---|
| AP2112K 3.3 Voltage Regulator | | |
| 1. Supply regulated 3.3 V output from 3.7 V battery. | 1. Measured the output voltage with a DMM | Y |
| MPU-9250 | | |
| 1. IMU should detects correct orientation within +/- 10 degrees after a 180 degree flip. | 1. Setup the IMU to display orientation on a computer monitor. Record the orientation with the unit laying flat on a table. Flip the unit and ensure the orientation is flipped by 180 +/- 10 degrees. | Y |
| 2. IMU limits noise and drift to +/- 1 degree while not moving. | 2. Setup the IMU to display orientation on a computer monitor. Hang the IMU from a string in a still environment.Ensure the orientation stays within a +/- 1 degree range. | Y |
| ESP-32 | | |
| 1. Is able to apply smoothing algorithms and maintain total latency under 10ms. | 1. Load microcontroller code to output both smoothed and unsmoothed data. Plot 1 second of smoothed data and unsmoothed data to ensure it smooths properly. Load microcontroller code to output smoothed and unsmoothed data separately. Calculate orientation packet frequency for both cases. Ensure smoothed packet is no less than 50% slower than unsmoothed packet frequency. | Y |
| 2. Can package pitch, yaw, and roll data according to OSC protocol standards. | 2. Load code on ESP32 that sends orientation data from IMU to a serial monitor according to the OSC protocol. Read the output data from | Y |

| | | |
|---|---|---|
| | the monitor and ensure all three orientations of yaw, pitch, and roll are output as 32-bit float values. | |
| SuperCollider | | |
| 1. Synthesizer object responds to external inputs to manipulate parameters. | 1.<br>Setup synthesizer to respond to laptop's trackpad to adjust note with x-axis and volume with y-axis, such that moving the mouse pointer right will increase the pitch, moving left will decrease the pitch, moving up will increase the volume and moving down will decrease the volume. | Y |
| 2. Parameter mapping ranges make perceivable sonic adjustments as dictated by movement. Assign each parameter to its required range as follows:<br>a. Note range spanning at least 2 octaves<br>b. Low-Pass Filter cutoff frequencies between 50 Hz and 15 kHz<br>c. Amplitude ranges from value of 0 to 0.6<br>d. Modulator frequencies from 0.01 Hz to 220 Hz<br>e. Panning from left to right | 2.<br>a. Open level meters, scope and frequency scope.<br>b. Create a synth object to respond to laptop's trackpad. For each parameter, map the value to the x-axis and y-axis of the mouse pointer as follows:<br>○ Set x-axis to respond to note range.<br>○ Set y-axis to respond to cutoff frequency<br>c. When the mouse pointer has reached the far left side of the screen, the note pitch should be 2 octaves higher.<br>d. When the mouse pointer reaches the top of the screen, all low frequencies should no longer be heard.<br>e. Repeat steps a. through d. for the remainder of the parameters.<br>○ For amplitude, watch the level meter to see the maximum amplitude reached.<br>○ For modulator frequencies, use the frequency scope to see that multiple side-bands appear<br>○ For the panning, listen for the sound to pan from left to right. | Y |
| 3. Synthesizer receives and responds to incoming OSC data. | 3.<br>Print input messages to the SuperCollider built in terminal monitor called the Post Window. | Y |

# Appendix B Circuit Schematics (Final Design)

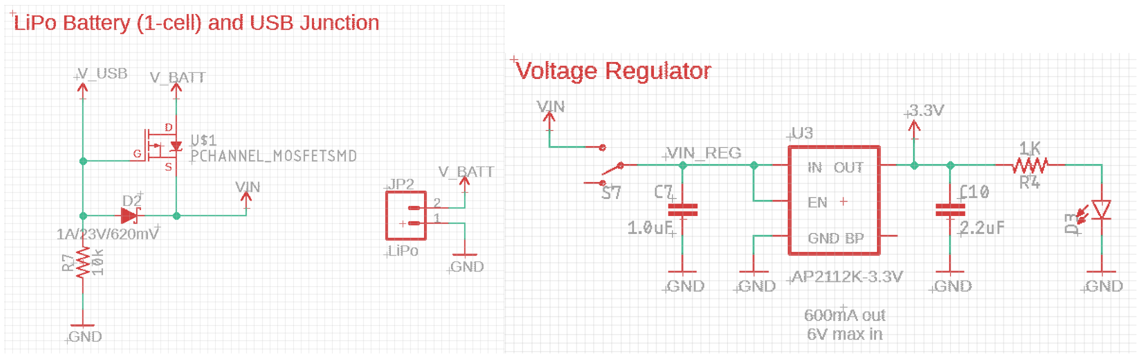


Figure 13 The Power Subsystem, Li-Po and USB power junction (left) and voltage regulator (right).

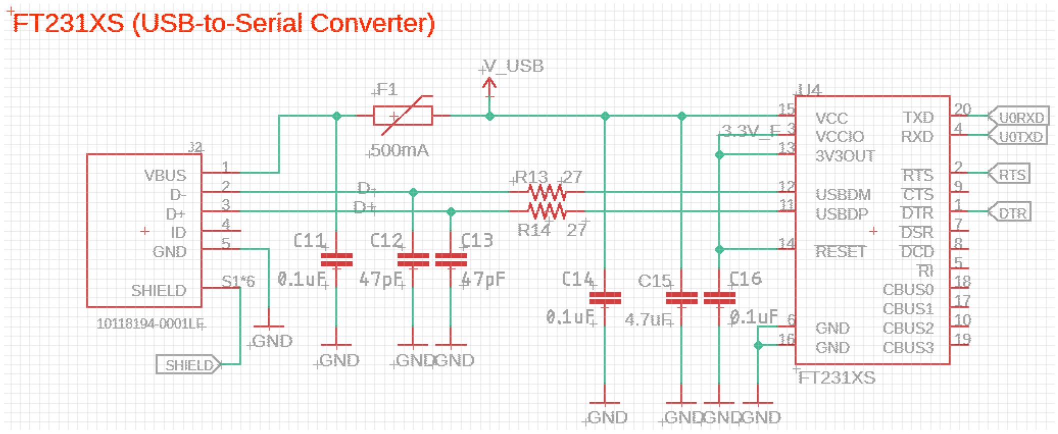
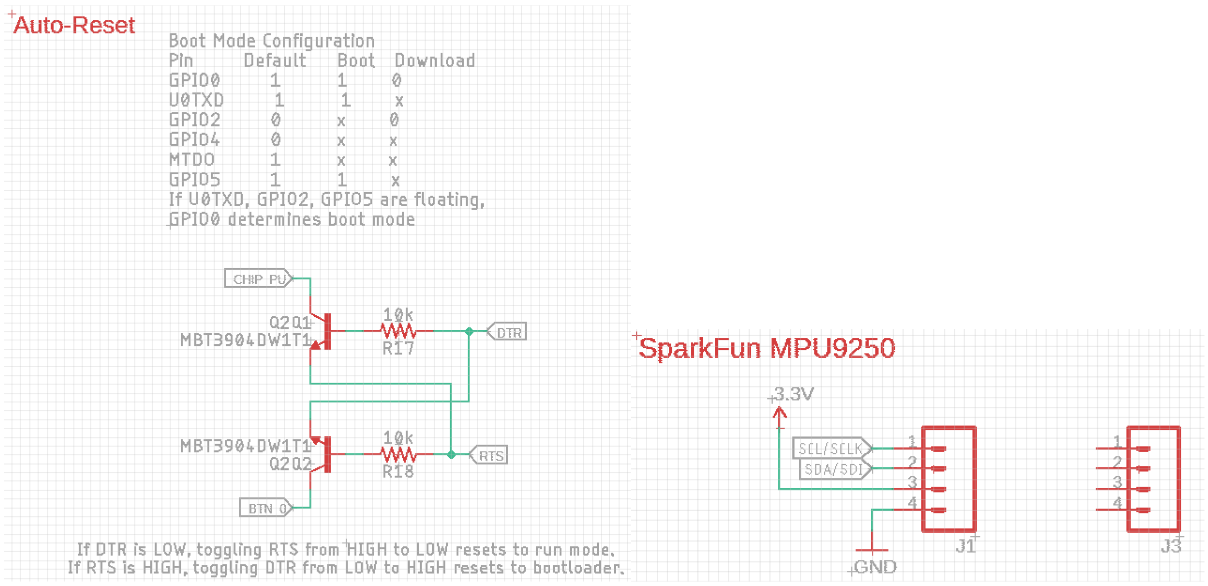


Figure 14 USB to Serial converter used for programming and debugging.



Boot Mode Configuration

| Pin | Default | Boot | Download |
|---|---|---|---|
| GPIO0 | 1 | 1 | 0 |
| U0TXD | 1 | 1 | x |
| GPIO2 | 0 | x | 0 |
| GPIO4 | 0 | x | x |
| MTDO | 1 | x | x |
| GPIO5 | 1 | 1 | x |

If U0TXD, GPIO2, GPIO5 are floating, GPIO0 determines boot mode

If DTR is LOW, toggling RTS from HIGH to LOW resets to run mode.
If RTS is HIGH, toggling DTR from LOW to HIGH resets to bootloader.

Figure 15 MCU reset and boot loading circuit (left) and IMU connection (right).

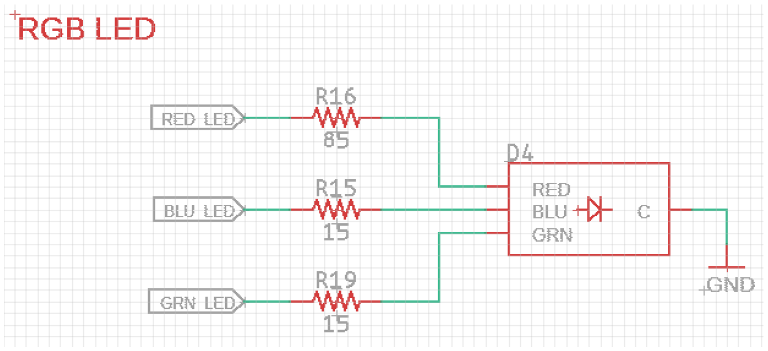


Figure 16 RGB LED.

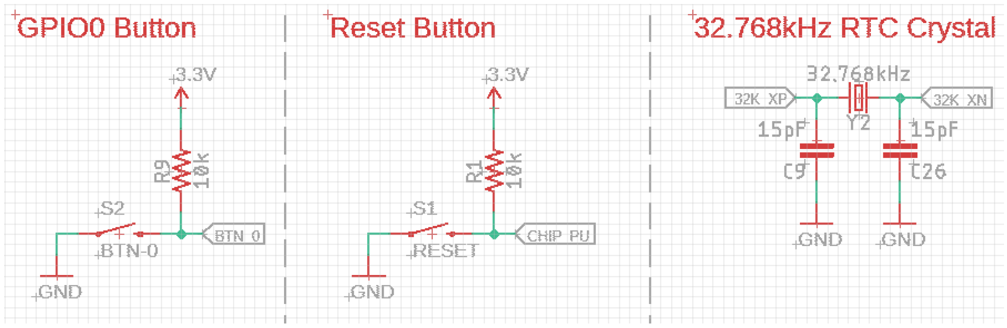


Figure 17 MCU boot loading (left) and reset (center) buttons, and low power timing clock (right).

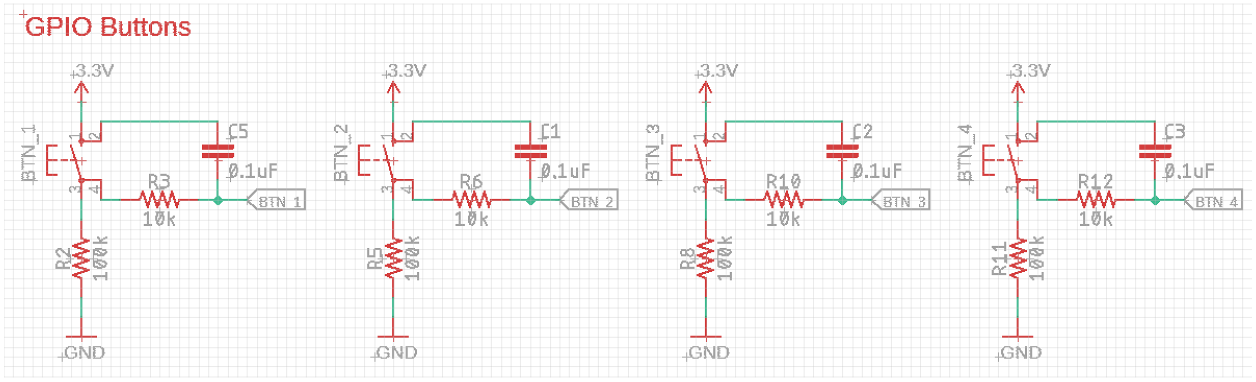
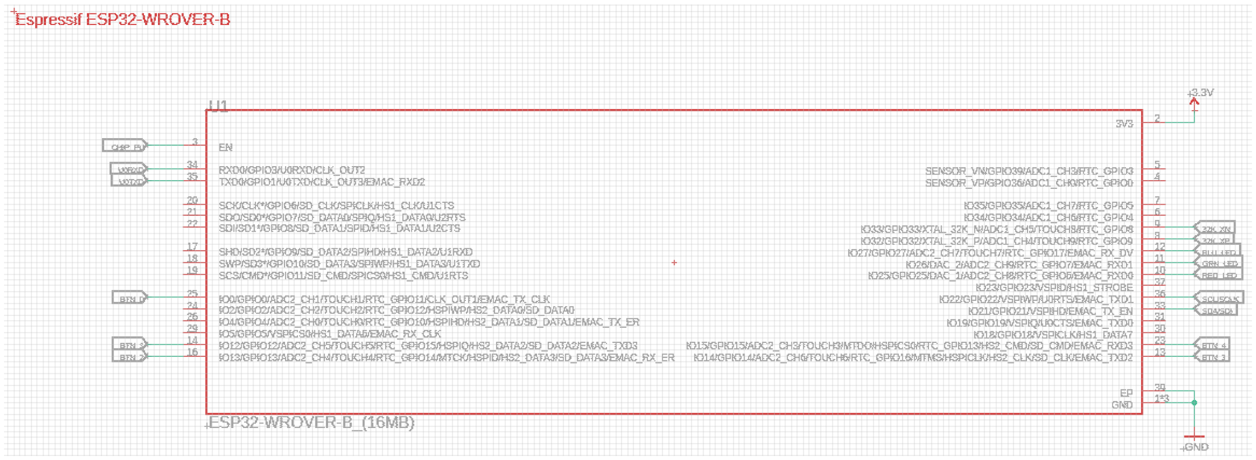


Figure 18 Programmable GPIO buttons and debouncing circuits.


Espressif ESP32-WROVER-B
U1
ESP32-WROVER-B_(16MB)
CHIP_PU
U0RXD
U0TXD
BTN_0
BTN_3
BTN_2
EN
RXD0/GPIO3/U0RXD/CLK_OUT2
TXD0/GPIO1/U0TXD/CLK_OUT3/EMAC_RXD2
SCK/CLK*/GPIO6/SD_CLK/SPICLK/HS1_CLK/U1CTS
SDO/SD0*/GPIO7/SD_DATA0/SPIQ/HS1_DATA0/U2RTS
SDI/SD1*/GPIO8/SD_DATA1/SPID/HS1_DATA1/U2CTS
SHD/SD2*/GPIO9/SD_DATA2/SPIHD/HS1_DATA2/U1RXD
SWP/SD3*/GPIO10/SD_DATA3/SPIWP/HS1_DATA3/U1TXD
SCS/CMD*/GPIO11/SD_CMD/SPICS0/HS1_CMD/U1RTS
IO0/GPIO0/ADC2_CH1/TOUCH1/RTC_GPIO11/CLK_OUT1/EMAC_TX_CLK
IO2/GPIO2/ADC2_CH2/TOUCH2/RTC_GPIO12/HSPIWP/HS2_DATA0/SD_DATA0
IO4/GPIO4/ADC2_CH0/TOUCH0/RTC_GPIO10/HSPIHD/HS2_DATA1/SD_DATA1/EMAC_TX_ER
IO5/GPIO5/VSPICS0/HS1_DATA6/EMAC_RX_CLK
IO12/GPIO12/ADC2_CH5/TOUCH5/RTC_GPIO15/HSPIQ/HS2_DATA2/SD_DATA2/EMAC_TXD3
IO13/GPIO13/ADC2_CH4/TOUCH4/RTC_GPIO14/MTCK/HSPID/HS2_DATA3/SD_DATA3/EMAC_RX_ER
+3.3V
3V3
SENSOR_VN/GPIO39/ADC1_CH3/RTC_GPIO3
SENSOR_VP/GPIO36/ADC1_CH0/RTC_GPIO0
IO35/GPIO35/ADC1_CH7/RTC_GPIO5
IO34/GPIO34/ADC1_CH6/RTC_GPIO4
IO33/GPIO33/XTAL_32K_N/ADC1_CH5/TOUCH8/RTC_GPIO8
IO32/GPIO32/XTAL_32K_P/ADC1_CH4/TOUCH9/RTC_GPIO9
IO27/GPIO27/ADC2_CH7/TOUCH7/RTC_GPIO17/EMAC_RX_DV
IO26/DAC_2/ADC2_CH9/RTC_GPIO7/EMAC_RXD1
IO25/GPIO25/DAC_1/ADC2_CH8/RTC_GPIO6/EMAC_RXD0
IO23/GPIO23/VSPID/HS1_STROBE
IO22/GPIO22/VSPIWP/U0RTS/EMAC_TXD1
IO21/GPIO21/VSPIHD/EMAC_TX_EN
IO19/GPIO19/VSPIQ/U0CTS/EMAC_TXD0
IO18/GPIO18/VSPICLK/HS1_DATA7
IO15/GPIO15/ADC2_CH3/TOUCH3/MTDO/HSPICS0/RTC_GPIO13/HS2_CMD/SD_CMD/EMAC_RXD3
IO14/GPIO14/ADC2_CH6/TOUCH6/RTC_GPIO16/MTMS/HSPICLK/HS2_CLK/SD_CLK/EMAC_TXD2
EP
GND
32K_XN
32K_XP
BLU_LED
GRN_LED
RED_LED
SCL/SCLK
SDA/SDI
BTN_4
BTN_1
GND
Figure 19 ESP 32 WROVER-B

## Appendix C PCB Layout (Final Design)

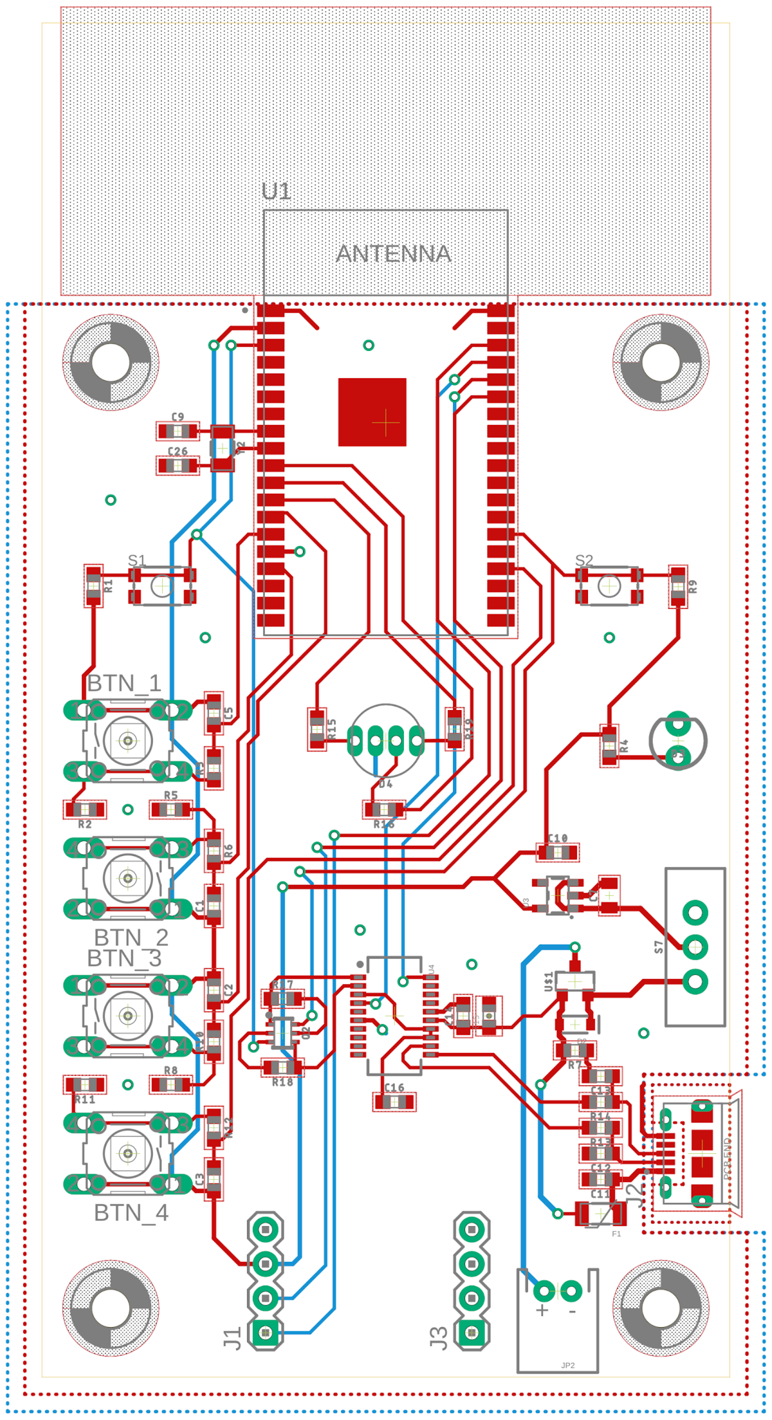


Figure 20 PCB Layout of final design

## Appendix D Circuit Schematics (Prototype Design)

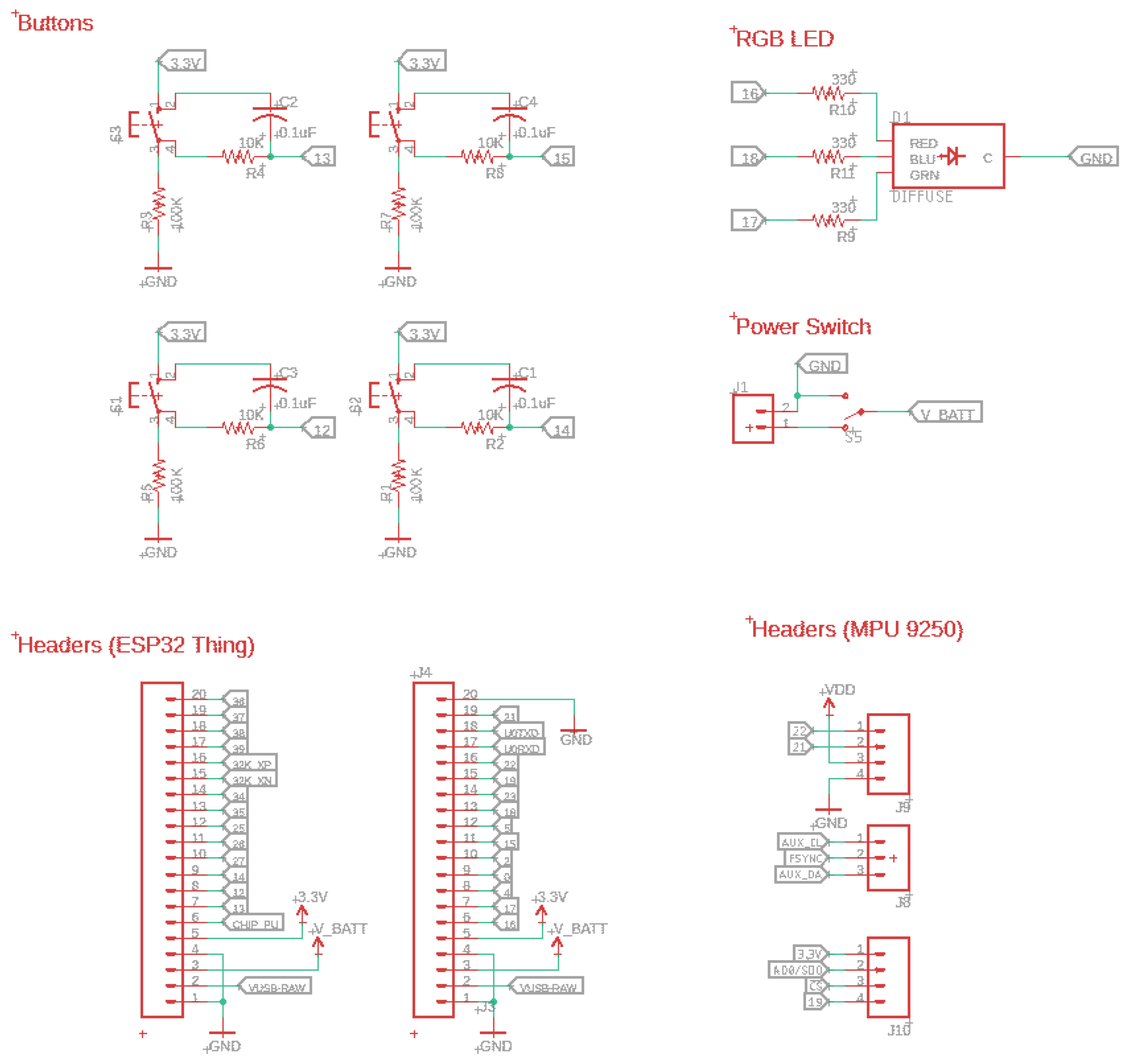


Figure 21 Prototype design circuit schematics (buttons, rgb led, ESP32 Thing, MPU 9250)

## Appendix E PCB Layout (Prototype Design)

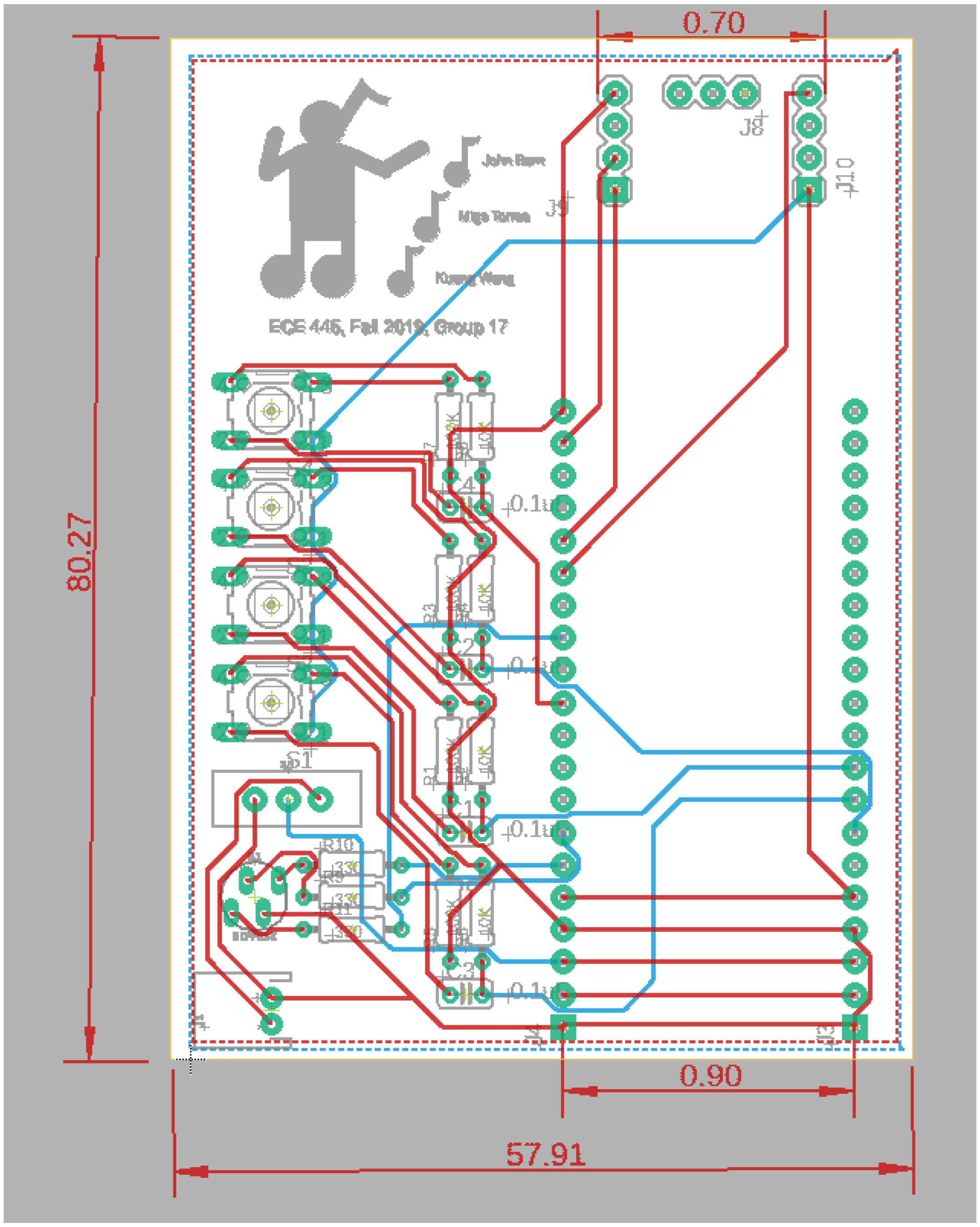


Figure 22 PCB Layout of prototype design