Bluetooth Enabled Gloves for Controlling Music

ECE 445: Final Report

Team Members:

Saicharan Bandikallu (sb35)

Mehul Aggarwal (mehula3)

Oliver Johnson (owj2)

TA: Akshatkumar Sanatbhai Sanghvi

Professor: Viktor Gruev

Due: 2/23/2023

Contents

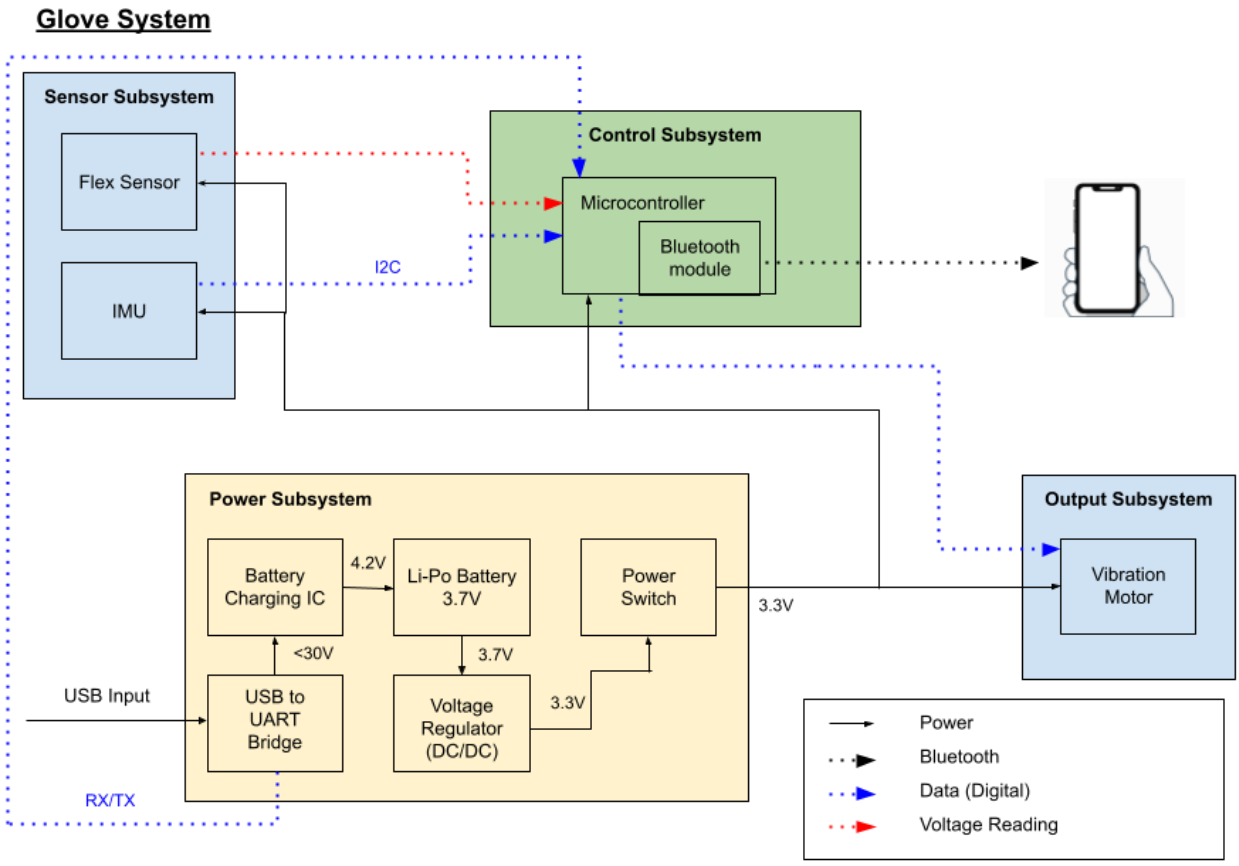
1 Introduction

1.1 Problem and Solution

Traditional gloves are often bulky and make it hard to control music playback on a phone. Even gloves manufactured to be touch screen compatible are quite inconsistent. Our solution to this problem is to build a glove that can connect to a user's phone via bluetooth and allow the user to control their music settings.

1.2 Block Diagram

Figure 1: Block Diagram



The four critical blocks are the power, sensor, control, and output subsystems. The power subsystem supplies power to the three other subsystems. The sensor subsystem takes data from the hand motions and sends that data to the control system. The control subsystem takes the hand motion data and detects when the user has made a relevant gesture and what gesture has been made. Once a gesture has been detected, the control subsystem sends the appropriate signals to the phone and the output system. The output subsystem takes commands from the control subsystem, and then vibrates the motor when the user makes a predetermined gesture.

1.3 High-Level Requirements List

In order to solve the problem in its entirety, we had three quantitative characteristics of an ideal system that will accomplish our goals:

1.3.1 Correctly Identify Predetermined Gestures From Sensor Readings.

We need to be able to detect when the user provides input that matches any one of the five predetermined gestures based on the sensor readings from the flex sensors and IMU sensor. In order to detect this, we will need to create a model that identifies the correct command based on a hand gesture. We also want to limit false gestures as much as possible to prevent the user from providing unintended commands to the phone.

1.3.2 Correctly Convert Predetermined Gestures into Correlated Bluetooth Commands

Once one of the five predetermined gestures is detected, the glove will need to provide the correct bluetooth command to control the phone. We will use the BLE protocol and the BLEKeyboard library to control music. We will know if the correct bluetooth command is sent based on the change we see on the phone.

1.3.3 Glove Has Small Form Factor and Battery Life Sustains a Session of Use

We also want to ensure the user has a final product that fits comfortably on their hand unobtrusively in a small form factor, while also having a decent battery life. We set a target goal of an average battery life of 3 hours for the glove. After testing our final design by actively using the glove for 33% of the time in a 3 minute time period, we extrapolated this data with a linear fit in order to calculate a battery life of 12.5 hours, which far exceeds the 3 hour battery life requirement.

2 Design

2.1 Design Procedure

The requirement of the power subsystem is to have a rechargeable battery that provides 3.3V to the other subsystems. For the rechargeable battery, we decided to use a Li-Po battery as they are the safest option with a good battery life and small form factor. It is also easier to find a battery IC for single-cell Li-Po batteries so that we could create a battery charging circuit. The other option would be to use disposable AA or AAA batteries, but this would make the design bulkier and would not allow for a closed system. Also, we decided to use a USB-C connector for charging to make use of the latest generation of technology.

For the sensor subsystem, our goal is to collect data to map out the movements of a user's fingers and hand. In order to track finger movements, our initial idea was to either use flex sensors or hall effect sensors on the knuckles with small magnets mounted on the fingers. In the end, we decided to choose the flex sensors because they made the mechanical design of the glove simpler, and it would be easier to collect and interpret the data from these sensors. We believe that with the hall effect sensors, we would have run into a lot of issues with noisy and inconsistent data readings, which would unnecessarily complicate our software data processing. For hand motion and orientation data, we decided to use an IMU. We initially planned to use the

gyroscope to detect hand orientation and the accelerometer to detect hand motion. About halfway through the semester, when we were working with our IMU data, we were using gyroscope readings to determine the orientation/position of the user's hand, and we were using accelerometer readings to determine the motion of the user's hand. However, we had a lot of issues with obtaining consistent readings from the gyroscope and found it difficult to determine the user's hand orientation. So, instead, we decided to stop using the gyroscope and only use the accelerometer. We included a calibration sequence where we set an initial zero position using the accelerometer data. This made it much easier to determine the glove's orientation. Overall, this ended up being more desirable because we were using one less sensor and reducing the amount of data we had to process.

For the microcontroller, we decided to go with the ESP32-WROOM-32E as it has an integrated bluetooth module, which will allow us to easily send bluetooth signals to the phone without the need for an external antenna. Also, the ESP32 microcontroller is very popular, so there's a lot of debugging documentation online. There are also many dedicated ADC (Analog-to-Digital Converter) pins on the ESP32 to convert our flex sensor voltage readings into digital voltage readings to analyze.

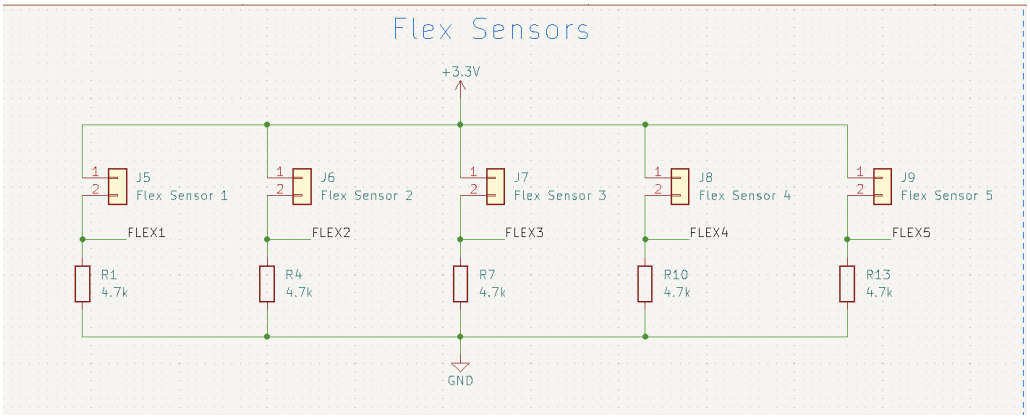
For our output system, we wanted to provide the user with haptic feedback, and so we decided to use a vibration motor as it is small, easy to mount, and easy to interface with. The other option would be to use a speaker, but this implementation would be more complicated and more obnoxious than the instant feedback we would get with a vibration motor. In addition, the audible feedback from the speaker is probably a lot less desirable than the silent, undetectable feedback from the vibration motor.

2.2 Design Details

2.2.1 Sensor Subsystem

The sensor subsystem consists of the flex sensors and the IMU. The flex sensors detect whether or not a finger is flexed and the IMU detects acceleration data of the hand movements.

Figure 2: Sensor Subsystem Circuit Schematic



Five flex sensors are sandwiched between two gloves and run along the five fingers, so they flex with the fingers. These are variable resistors and their resistance decreases when flexed, increasing the current through them. The flex sensors are placed in a voltage divider circuit with a shunt resistor. The microcontroller then reads the voltage drop across the shunt resistor due to the change in resistance of the flex sensors. As such, the microcontroller is able to read a finger flexing as a change in voltage on an input GPIO pin. To determine the proper resistance value for the shunt resistors to get a good reading, the calculations in Figure 3 were performed.

Figure 3: Flex Sensor Shunt Resistor Calculations

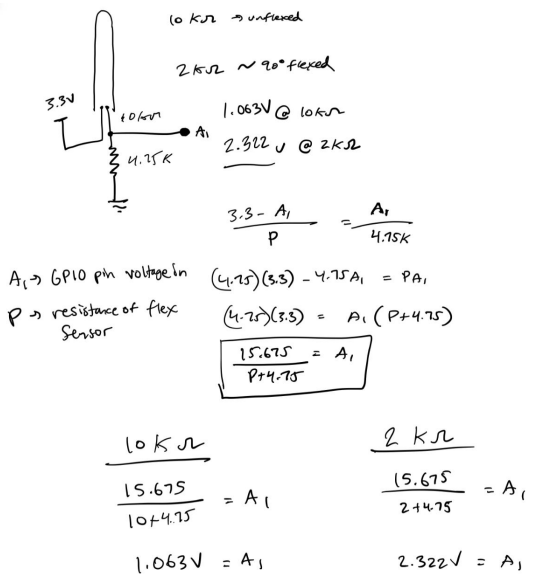
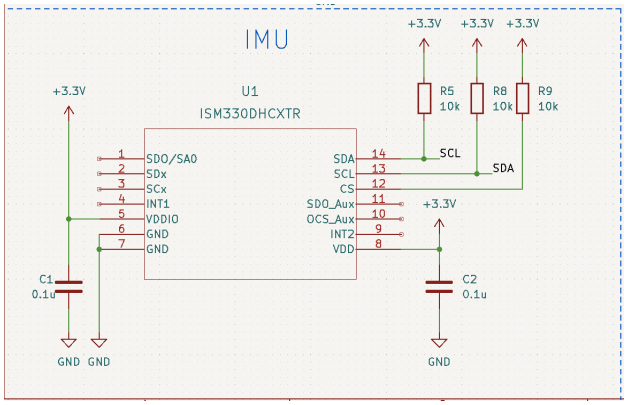


Figure 4: IMU Circuit Schematic

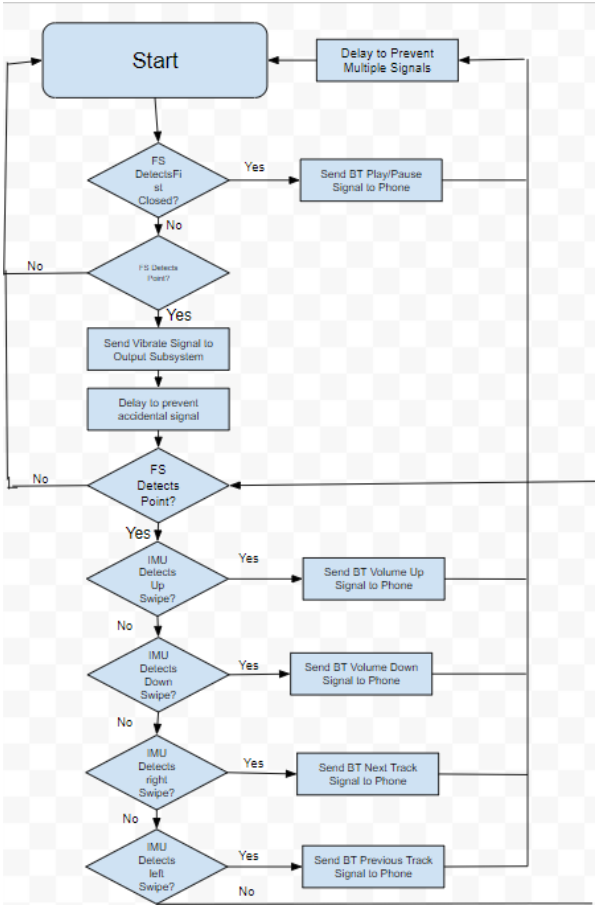


The IMU detects acceleration and orientation data and sends it to the microcontroller through I^2C protocol. The microcontroller sees this data as float values for x, y, z acceleration and orientation. Through our building process, we eventually decided that the gyroscope was no longer needed and so we only used the accelerometer in order to detect both orientation and movement.

2.2.2 Control Subsystem

The control subsystem takes data from the sensor subsystem and outputs bluetooth signals to the connected device as well as feedback signals to the output subsystem.

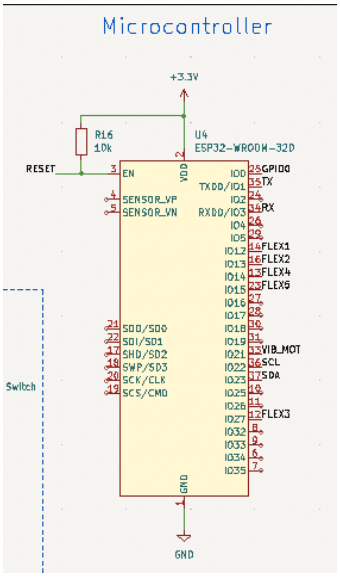
Figure 5: Control System Flow Chart



The flow chart in Figure 8 is the model we based our microcontroller code on. The user needs to put their gloved hand at 90º facing forward when they turn the device on for calibration. A vibration signal will be sent to the output subsystem when the device is connected and calibrated. At this point, the user can relax their hand. The user can now close their fist to send a play/pause bluetooth signal or do a two-finger point in the calibration orientation to initiate any of the other four signals. Both actions will send a vibrate signal to the output subsystem. This works because the accelerometer is zeroed in the calibration orientation. This is the only position

in which the accelerometer will read approximately zero, so the microcontroller will read correct orientation when a two-finger point is detected and the accelerometer reads near-zero acceleration. Once the correct orientation is recognized by the microcontroller, it waits for a right, left, up, or down swipe-induced acceleration reading and then sends the appropriate bluetooth signal based on the swipe direction. A signal is also sent to the output subsystem in order to provide haptic feedback to indicate that a command has been performed.

Figure 6: Control System Circuit Schematic



The output from the reset circuit is connected to the enable pin, so the code will re-run every time the reset button is pressed. The GPIO0 circuit output is connected to the GPIO0 button so that when the button is pressed, the circuit will enter bootloader mode and the code can be flashed.

Figure 7: Button Circuit Schematic

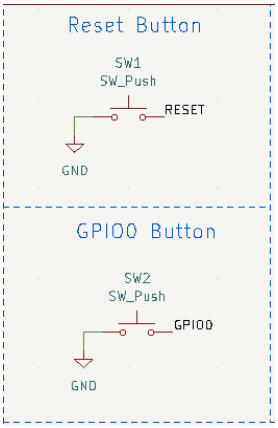
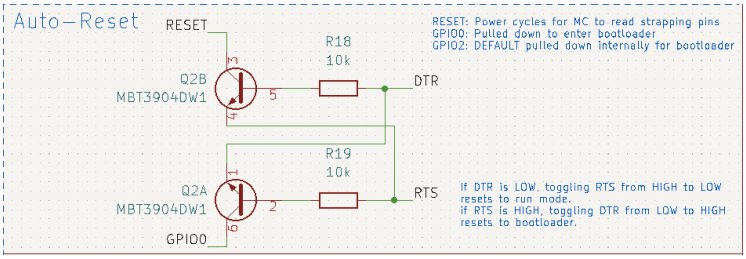


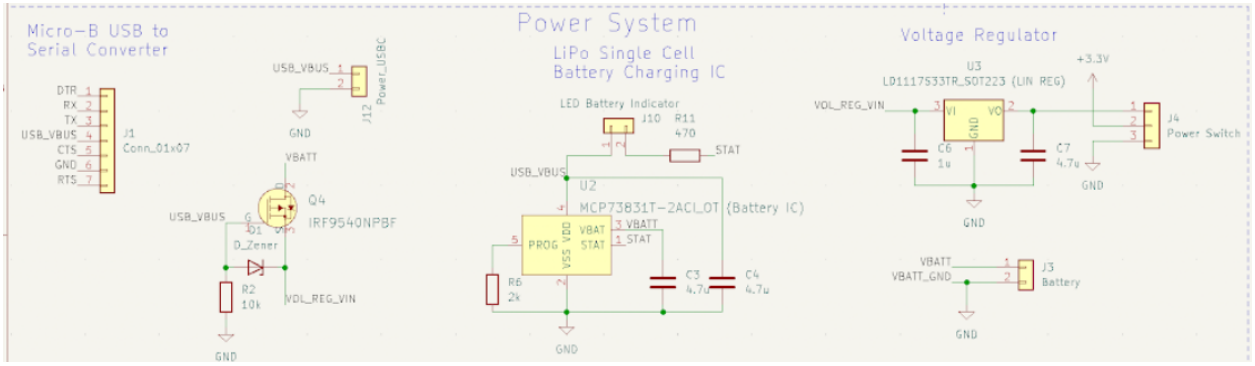
Figure 8: Auto-Reset Circuit Schematic



2.2.3 Power Subsystem

The power subsystem consists of the Micro-B USB to Serial Converter, battery charging circuit, battery, and voltage regulator.

Figure 9: Power Subsystem Circuit Schematic



The micro-B USB to serial converter breaks out the USB data lines into Rx and Tx lines. This is what enables the microcontroller to be programmed.

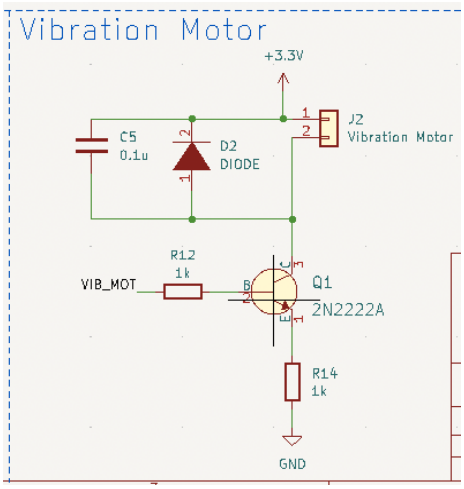
The battery charging circuit has an IC that supplies a 500mA constant current to the battery allowing it to safely be charged. This is a safe charge current as it is significantly less than the max fast charge current of 1500mA as specified in the data sheet. The battery we are using has a nominal voltage of 3.7V with a capacity of 3000mAh.

The voltage regulator circuit takes its input from the switching circuit. The switching circuit is designed such that when the USB is connected, the USB-VBUS is the input to the voltage regulator, and when the USB is disconnected, the battery is the input to the voltage regulator. This is done so that the battery isn't powering the circuit (discharging) while charging. The voltage regulator steps down the battery voltage to 3.3 V to provide the necessary voltage to the rest of the subsystems.

2.2.4 Output Subsystem

The output subsystem consists of the vibration motor. The motor is connected to an output from the control subsystem. This output signals the motor when to vibrate.

Figure 10: Output Subsystem Circuit Schematic



The vibration motor circuit is a simple switching circuit with a BJT. It is biased in a way such that when the microcontroller output is high, the vibration motor vibrates, and when the microcontroller output is low, the vibration motor will not vibrate. The VIB_MOT signal will only be high when the microcontroller determines haptic feedback is needed. In addition, to avoid any voltage spikes from the motor, we included a capacitor in parallel to the vibration motor, and to stop any reverse current, we included a diode in parallel to the vibration motor.

2.3 Verifications

The following tables includes the design requirements, verification tests, and test results:

Table 1: Sensor Subsystem Requirements and Verifications Table

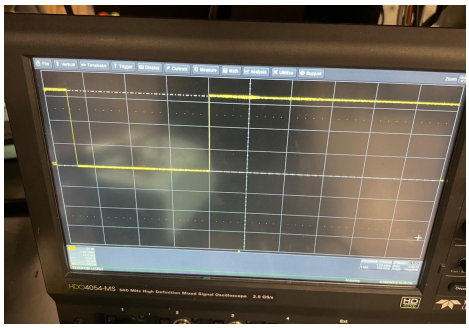
<table>
<tr><th><u>Requirements</u></th><th><u>Verifications</u></th><th><u>Test Results</u></th></tr>
<tr>
<td>When the flex sensors are in an unflexed position, they should give a raw ADC reading of 0, and when the flex sensors are flexed, they should produce a raw ADC reading of greater than 150. We are just looking for a large binary change between unflexed and flexed positions.</td>
<td>1. Connect the flex sensors to the microcontroller using a voltage divider circuit as specified in our circuit schematic
2. Ensure the flex sensors are attached to the fingers and are unflexed (fingers are straight)
3. Measure the raw ADC reading on the microcontroller using the serial monitor. Ensure this measurement is 0±5.
4. Flex all fingers into a fist.
5. Measure the raw ADC reading on the microcontroller using the serial monitor. Ensure this measurement is greater than 150.
6. Repeat steps 2-4 for 5 trials, and ensure measurements are within specified limits.</td>
<td> 
5 Trials of flexing and unflexing fingers. Peaks refer to flexed ADC value, and unflexed refers to 0 ADC value.</td>
</tr>
<tr>
<td>The accelerometer readings from the IMU sensor should be accurate to within ±5% when measured under the same motion.</td>
<td>1. Connect the IMU to the microcontroller and establish the I2C communication between the 2 devices
2. Place the IMU flat on a desk with the z-axis</td>
<td>
<table>
<tr><th>Trial</th><th>x</th><th>y</th><th>z</th></tr>
<tr><td>1</td><td>1.00</td><td>.98</td><td>1.01</td></tr>
<tr><td>2</td><td>1.00</td><td>.98</td><td>1.01</td></tr>
<tr><td>3</td><td>1.00</td><td>.98</td><td>1.01</td></tr>
</table>
</td>
</tr>
</table>

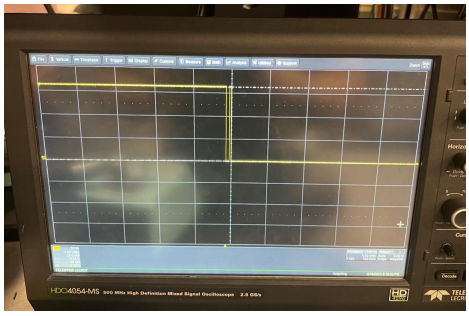
	facing up. Note down the accelerometer value for the z-axis (this reading should be around 1g). 3. Place the IMU flat on a desk with the y-axis facing up. Note down the accelerometer value for the y-axis (this reading should be around 1g). 4. Place the IMU flat on a desk with the x-axis facing up. Note down the accelerometer value for the x-axis (this reading should be around 1g). 5. Repeat steps 2-4 for another 4 trials. Calculate the error (percent difference) for each axis, compared to the first trial. 6. Then calculate the mean error for each axis from each of the trials. This mean error should be less than 5%.	

4	1.00	.98	1.01
5	1.00	.98	1.01
Mean Error	0%	0%	0%

Mean Error of IMU Acceleration Data Across 5 trials for X,Y, and Z axis

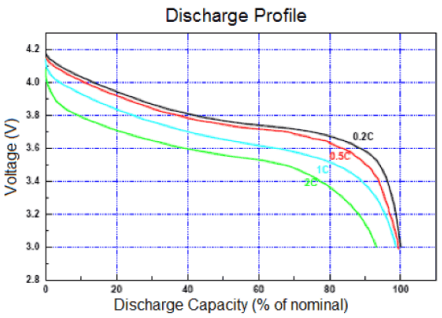
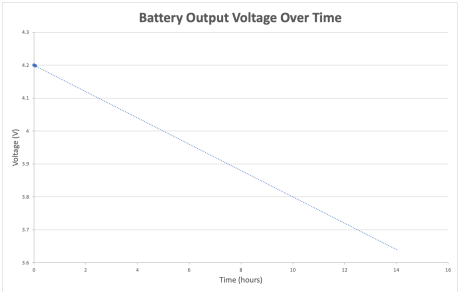
Table 2: Control Subsystem Requirements and Verifications Table

Requirements	**Verifications**	**Test Results**
When the RESET button is pressed, the microcontroller is power cycled.	1. Connect a DC 3.3V power supply to the VDD pin of the microcontroller. 2. Connect the channel 1 probes of an oscilloscope to the enable pin of the microcontroller and ground of the microcontroller. 3. Measure the voltage of the	

	enable pin. This should measure 3.3V ± 0.1V. 4. Hold down the RESET button. Ensure that the oscilloscope reads 0V ± 0.1V on channel 1. 5. Release RESET button. This should power cycle the microcontroller (the oscilloscope should now read 3.3V ± 0.1V on channel 1).	*Microcontroller enable pin is held high initially, goes low when reset is hit, and goes back to high when reset is released*
When the GPIO0 button is pressed, the bootloader in the microcontroller is enabled.	1. Connect a DC 3.3V power supply to the VDD pin of the microcontroller. 2. Connect the channel 1 probe of an oscilloscope to the GPIO0 pin of the microcontroller and ground of the microcontroller. 3. Hold down the GPIO0 button. Measure the voltage on channel 1. It should read 0V ± 0.1V. This ensures that we are in bootloader mode. The GPIO0 button can now be released.	 *Microcontroller GPIO pin is held high initially and goes low when the GPIO0 button is pressed, enabling the microcontroller bootloader*
Provide at least 2 mA current from a GPIO pin in order to switch on BJT transistor to drive the vibration motor.	1. Connect a DC 3.3V power supply to the VDD pin of the microcontroller. 2. Connect pin GPIO21 on the microcontroller to the base node of the BJT transistor in the vibration motor circuit (specified in the figure 13). Create the Vibration motor circuit from figure 13. 3. Program the pin GPIO21 to be an output pin sending out a digital '1'. 4. Using a voltmeter, measure the voltage drop across the resistor R12 (470 ohm resistor in figure 13). This	 Voltage at Output of Vibration Motor GPIO pin 2.292V / 470Ω = 4.8766 mA 4.8666 mA > 2 mA as specified in verifications

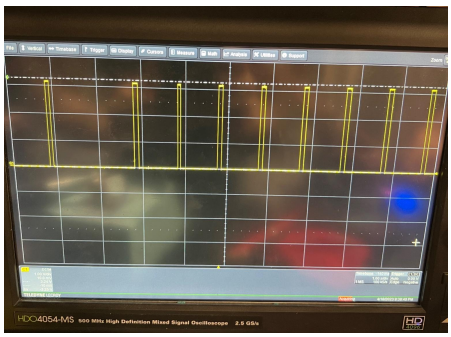
	measured voltage divided by 470 ohms should produce a value greater than 2 mA.	
When a predetermined gesture input is provided to the microcontroller through the flex sensors and IMU, the microcontroller should be able to correctly identify the gesture 90% of the time.	1. The gesture inputs that our system will take are: up swipe, down swipe, right swipe, left swipe, and close fist (refer to figure 8 for more information). 2. Perform the first gesture 10 times and each time determine if the gesture was correctly identified in the code (by using a print statement). Ensure we get a success rate of 9/10. 3. Repeat step 2 for the other 4 gestures. This will verify that our thresholds set for each gesture are correctly tuned.	This is directly observed in normal device operation. Refer to the appendix for the demo video link.
Translate identified gestures into bluetooth commands using BLE protocol and ensure these commands can be sent to a user's iPhone.	1. Establish bluetooth connection between microcontroller and an iPhone using the bluetooth protocol as specified in the ESP32-WROOM documentation. 2. Send each of our predetermined gestures as bluetooth commands following BLE protocol for music playback controls (volume up/down, play/pause, next/previous track). 3. Ensure the commands are correctly executed on the iPhone when trying to control music playback.	This is directly observed in normal device operation. Refer to the appendix for the demo video link.

Table 3: Power Subsystem Requirements and Verifications Table

Requirements	**Verifications**	**Test Results**
The battery must have a capacity of at least 2.39 Ah (in order to have a battery life of at least 3 hours).	1. Build the entire circuit with the battery attached 2. Connect the terminals of the battery to a voltmeter and measure the output voltage and ensure it is above 3.7V. 3. Turn on the power switch. 4. Run the glove for 3 minutes, providing active gestures for 20% of the time. 5. Turn off the power switch. 6. Measure the change in the output voltage of the battery. 7. Extrapolate this data to see how long it will take for battery voltage to drain to 3.7V. Ensure this number is above 3 hours.	Initial Battery Voltage: 3.8226 V After 3 minutes with 33% active use: 3.8206 V 4.2V - 3.7V = 0.5V 0.002V drop in 3 minutes Thus, 0.5V drop will occur in approximately 12.5 hours  Discharge: 3.0V cutoff at room temperature. 
The power switch must safely disconnect and reconnect the voltage regulator to the rest of the circuit. When the power switch is in the ON state, the voltage regulator should output a voltage of 3.3V ± 0.1V.	1. Create the entire PCB circuit. 2. Connect an oscilloscope to terminal 1 of the power switch (from figure 12) and to ground. 3. In the ON state (switch is connecting terminals 1 and 2 together), the voltage reading should be 3.3V ± 0.1V. 4. In the OFF state (switch is connecting terminals 2 and 3), the voltage reading	This is directly observed in normal device operation. Refer to the appendix for the demo video link.

	should be less than 1mV.	
Total maximum current draw for all components must be less than 1 A to fit specifications. Thus, the output of our voltage regulator must supply a current of less than 1 A to the rest of the circuit.	1. Run device at max power consumption (full flex) 2. Probe current output of voltage regulator 3. Verify the current output is well below 1A	22/6 ohms = 3.67 ohms 3.27V  *When running at full power, the current is maxed out at 893 mA. This fits within the component specifications.*

Table 4: Output Subsystem Requirements and Verifications Table

Requirements	**Verifications**	**Test Results**
Vibration motor vibrates when a digital high is provided from the microcontroller and the vibration motor turns off when a digital low is provided from the microcontroller.	1. Connect a DC 3.3V power supply to the VDD pin of the microcontroller. 2. Connect pin GPIO21 on the microcontroller to the base node of the BJT transistor in the vibration motor circuit (specified in the figure 13). Create the Vibration motor circuit from figure 13. 3. Connect an oscilloscope to pin GPIO21 of ESP32 4. Program the microcontroller to send a digital high to pin GPIO21. Ensure the voltage reading is 3.3V ± 0.1V. Ensure motor is vibrating 5. Program the microcontroller to send a digital low to pin GPIO21. Ensure that the voltage reading is 0V ± 0.1V.	 *Vibration Motor Circuit Output*

	Ensure motor isn't vibrating.	

3 Costs

The average ECE graduate makes about $80,000 per year, which translates to roughly $40 per hour. We worked about 12 hours per week. Not including spring break, that's 15 weeks. Since there are three of us, the total labor costs add up to around $21,600.

A table of all parts we ordered in the project along with their prices is in the following table:

Table 5: Overall Costs Table

Quantity	Part	Price/part	Price ($)
x1	ESP32 Dev Board	22.50	22.50
x4	ESP32 Microcontroller	2.84	11.36
x1	ESP32 Microcontroller	3.30	3.30
x1	Flex Sensor	6.50	6.50
x4	Flex Sensor	6.08	24.32
x1	IMU Dev Board	24.95	24.95
x1	Vibration Motor	2.25	2.25
x1	IMU	12.23	12.23
x11	Battery IC	.73	8.03
x1	Battery	14.95	14.95
x1	USB-C Port	.86	.86
x1	Linear Voltage Regulator	1.05	1.05
x3	Linear Voltage Regulator	1.16	3.48
x4	Power Switch	.52	2.08
x1	micro-B USB Receptacle	.46	.46
x1	USB-UART Bridge Evaluation Board	7.95	7.95
x1	USB-UART Bridge	5.70	5.70
x1	Gloves	8.99	8.99
x1	Velcro	6.99	6.99

x1	PLA Filament	13.85	13.85
x1	Solderable Resistors and Capacitors	9.02	9.02
x1	Paracord	4.99	4.99
		Total	**$145.81**

4 Conclusions

By doing this project, we learned how to go through the end to end design of a project from the brainstorming phase to a prototype. We learned how to test, do root-cause analysis, and how to do verifications of a product using data.

Given more time and resources, there are a couple things we would change about our design. First, we would add a battery-life indicator on the outside of the PCB encasing so that the user knows when the battery is in need of a charge. Second, we would make the PCB smaller and build a less bulky PCB encasing. Given the size of the PCB, it's hard to get a small form factor for the glove. If we were able to make the PCB and encasing smaller, the glove would look much cleaner.

5 Citations

[5.1] US Electronics, “Li-Ion Polymer Battery Specification,” USE-104060-3K-PCBJST, 10/08/2021

[5.2] Spectra Symbol, “Flex Sensor,” 2014

[5.3] Pololu, Shaftless Vibration Motor 8x3.4mm, 1637

[5.4] Espressif Systems, “ESP-WROOM-32 Datasheet,” 3/9/2017

[5.5] Epson, “IMU (Inertial Measurement Unit) ,” M-V340PD, 3/2015

6 Appendix

Demo Video Link:

https://drive.google.com/file/d/1SqetAqnOtcYmrEkJZ76r0BpYW9VIJGyb/view?usp=sharing