Bluetooth Enabled Gloves for Controlling Music

ECE 445: Design Document

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

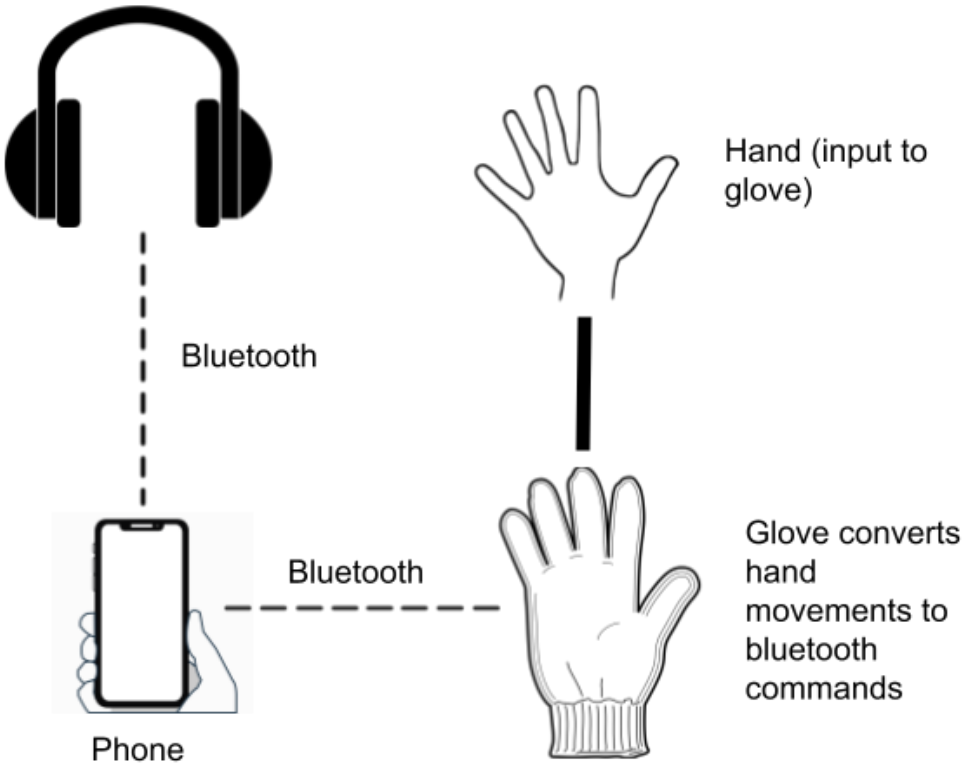
There are many situations in our lives where we may be wearing gloves. Activities such as construction, gardening, woodworking, and serving food require gloves. We also wear gloves when it's cold outside. Since gloves are often bulky, wearing them makes it difficult for people to be able to control the music playback on their headphones and accept/decline calls. It's hard to press buttons on your headphones, take your phone out of your pocket, or enter touchscreen commands with gloves on. Thus, people have to take their gloves off if they want to control their music playback and accept/decline calls. This is very inconvenient.

1.2 Solution

We will create a system of technological gloves that can be used to fix this problem. The gloves will be able to connect via bluetooth to a user's phone and allow the user to control their music settings and accept/reject calls. The music settings the gloves will control are volume up/down, play/pause, and next/previous track. This will be possible through the use of flex sensors mounted on all five fingers and an IMU sensor that will be mounted on the hand. The IMU sensor will allow us to capture movement and changes in orientation at the wrist joint. The microcontroller will transform this sensor data into useful bluetooth commands, allowing the user to control their music/phone without having to remove their gloves.

1.3 Visual Aid

Figure 1: Visual Aid



1.4 High-Level Requirements List

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

1.4.1 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 seven 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 as many false positive gestures as possible to prevent the user from providing unintended commands to the phone.

1.4.2 Correctly Convert Predetermined Gestures into Correlated Bluetooth Commands

Once one of the seven predetermined gestures is detected, the glove will need to provide the correct bluetooth command to control the phone. We will use HFP protocol to accept and reject phone calls and the AVRCP protocol to control music. The goal of this requirement is to correctly use these protocols to send the specified command to the phone based on the gestures detected by the microcontroller. We will know if the correct bluetooth command is sent based on the change we see on the phone.

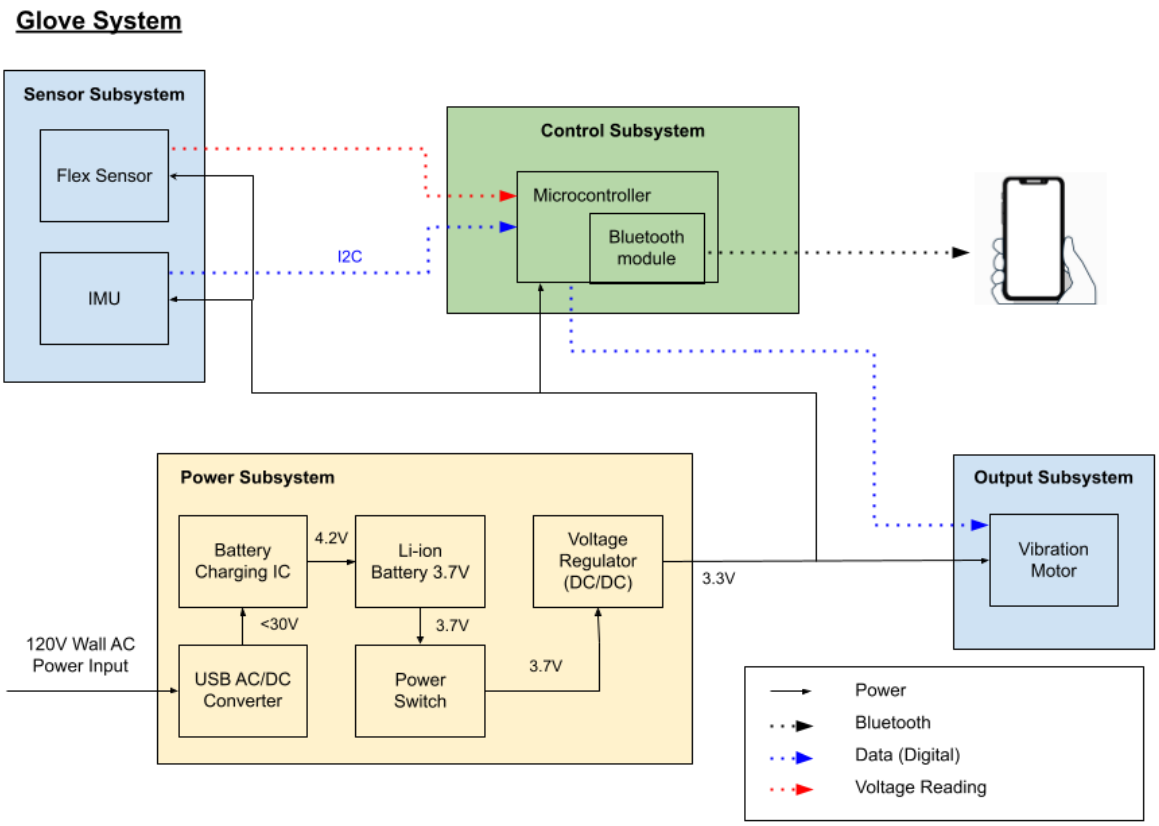
1.4.3 Glove Has Small Form Factor and Battery Life Sustains a Session of Use

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 enough battery life to last a session's worth of wearing. We have set a target goal of an average battery life of 3 hours for the glove. This will be plenty of time for a gloves session. In order to test the battery life, we will hook up our system to an oscilloscope and measure voltage across a shunt resistor, triggering the oscilloscope when the current goes to 0A.

2 Design

2.1 Block and Physical Diagram

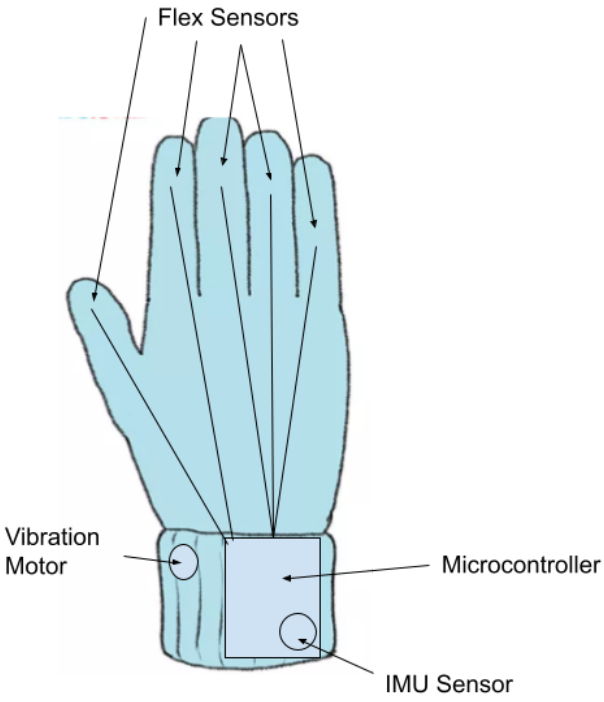
Figure 2: Block Diagram



The four critical subsystems 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. Once a gesture has been detected, it sends the appropriate signals to the phone and the output system. The output

subsystem takes data from the controller. The output subsystem is such that it vibrates when the user makes the gesture indicating he or she is about to signal.

Figure 3: Physical Diagram



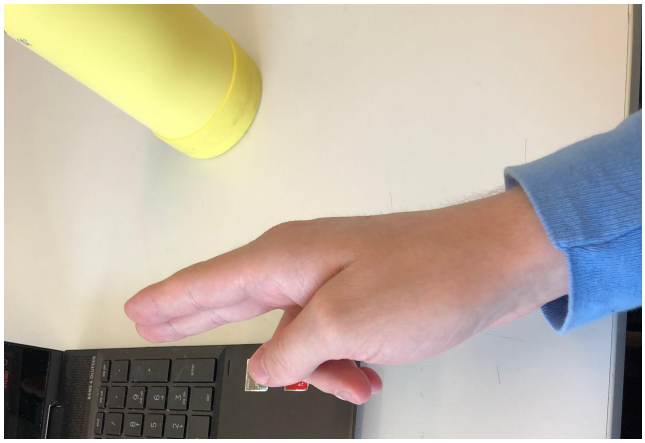
2.2 Subsystems

2.2.1 Sensor Subsystem

The sensor subsystem will consist of the IMU sensor and five flex sensors. The IMU sensor will detect orientation and movement using an internal accelerometer and gyroscope. The flex sensors will indicate which fingers are flexed at the knuckle. In conjunction, these sensors will indicate the current gesture of the user's hand. The IMU and flex sensors will connect to the data I/O pins of the ESP32.

There will be five different gestures to detect. For play/pause, the user will make a fingers-together gesture that will be detected by the flex sensors exclusively. For the rest of the signals, the user will point their index and middle finger forward and place their other fingers down as seen in the following picture.

Figure 4: Point for Signaling Gestures

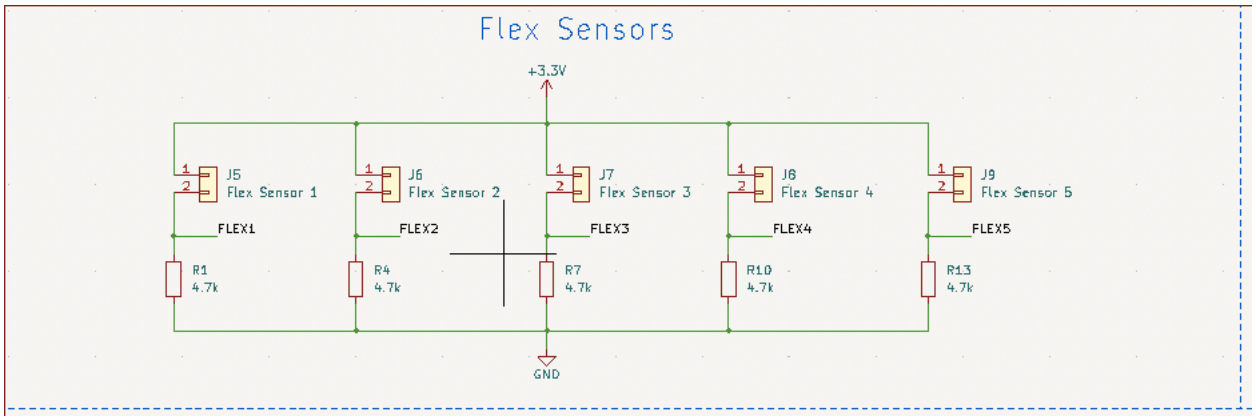


This gesture will be detected by the flex sensors exclusively. This will trigger a vibration from the output subsystem. At this point, the user will move their hand up for volume up, down for volume down, to the right for the next track, and to the left for previous track. This motion will be detected by the IMU exclusively. The reason for the two-part signal is so users don't do an accidental signal. The first part uses exclusively flex sensors and the second part uses the IMU exclusively to simplify the logic performed by the microcontroller.

The IMU will be biased to 3.3V and the output data will directly connect to the I/O pins of the ESP32 microcontroller with I^2C interface. The flex sensor can take a range of input voltages up to 5V, so we will bias the flex sensors to 3.3V as well. The output will also connect to the microcontroller. Since the flex sensor is just a variable resistor, we will construct a simple voltage divider circuit with a fixed resistance and read the voltage between the two resistors. This voltage value will change depending on the resistance value of the flex sensor. The more

"flexed" the sensor is, the greater the resistance value. The sensor subsystem sends gesture data to the control subsystem and gets power from the power subsystem.

Figure 5: Sensor Subsystem Circuit Schematic



Flex sensors work as variable resistors and their resistance depends on the amount they're flexed. For the flex sensors we're using, the resistance value is 10kΩ when unflexed and 2kΩ when flexed 90 degrees. To measure this resistance change, the microcontroller will need to see a voltage change since it cannot read resistance. In order to do this, we can measure the voltage drop across a fixed shunt resistor. We approximate a 4.75kΩ shunt resistor will provide us with a voltage range of 1.0627V to 2.322V at the microcontroller GPIO pin. This is within the acceptable range of the microcontroller pins. The calculations are as follows:

Figure 6: Calculations

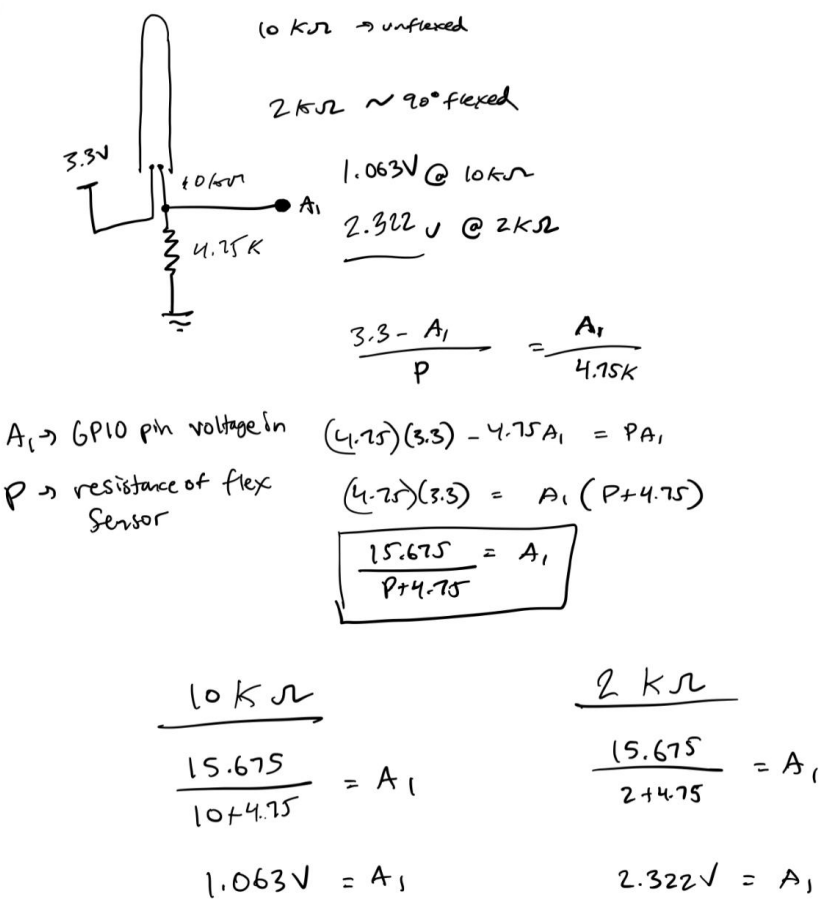
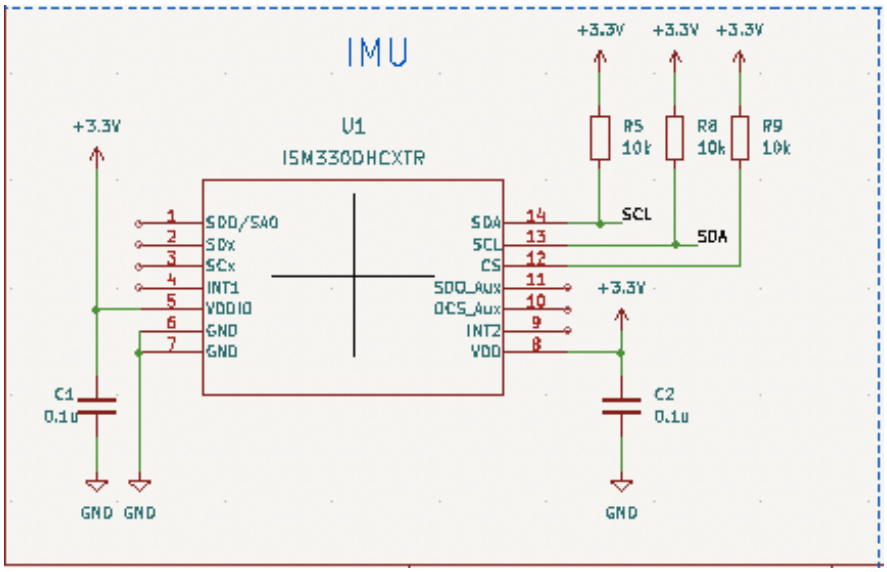


Figure 7: IMU Circuit Schematic



For the IMU sensor, we will be using the I^2C digital communication protocol to communicate with the microcontroller. As per the data sheet, this requires the SCL and SDA lines to be pulled up using 10kΩ resistors. We added a decoupling capacitor to the VDDIO and VDD lines to get a more stable DC voltage. The CS is also pulled high in order to indicate that we will be operating the IMU in I^2C mode.

Table 1: Sensor Subsystem Requirements and Verifications Table

Requirements	**Verifications**
When the flex sensors are unflexed (flat at 180 degrees), the resistance value of the sensors should be 10k ohms ± 100 ohms and when flexed to a 90 ± 1 degree angle in the center, the resistance value should be 2k ohms ± 100 ohms. We need to ensure that the flex sensor acts as a potentiometer with a linear relationship between the resistance and the bend angle.	1. Ensure the flex sensor is flat at 180 degrees using a protractor. Connect the 2 probes of an ohmmeter to the 2 ends of the flex sensor and measure the resistance. 2. Bend the flex sensor to 90 ± 1 degrees in the middle using a protractor. Connect the 2 probes of an ohmmeter to the 2 ends of the flex sensor and measure the resistance. 3. Calculate the linear relationship between the resistance and the bend angle of the flex sensor using these 2 points. 4. Use this linear relationship to determine the theoretical resistance of the flex sensor at a bend angle of 45 degrees. 5. Bend the flex sensor to 45 ± 1 degrees in the middle using a protractor. Connect the 2 probes of an ohmmeter to the 2 ends of the flex sensor and measure the resistance. 6. Compare the theoretical value of the resistance at 45 degree bend angle to the experimental value of the resistance at 45 degree bend angle and determine the error (difference) between these 2 values. This error should be less than 100 ohms.
The accelerometer readings from the IMU sensor should be accurate to within ±5% when measured under the same motion.	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 facing up. Note down the accelerometer value for the z-axis (this reading should be -9.8 ± 0.1 ms^{-2}). 3. Place the IMU flat on a desk with the y-axis facing

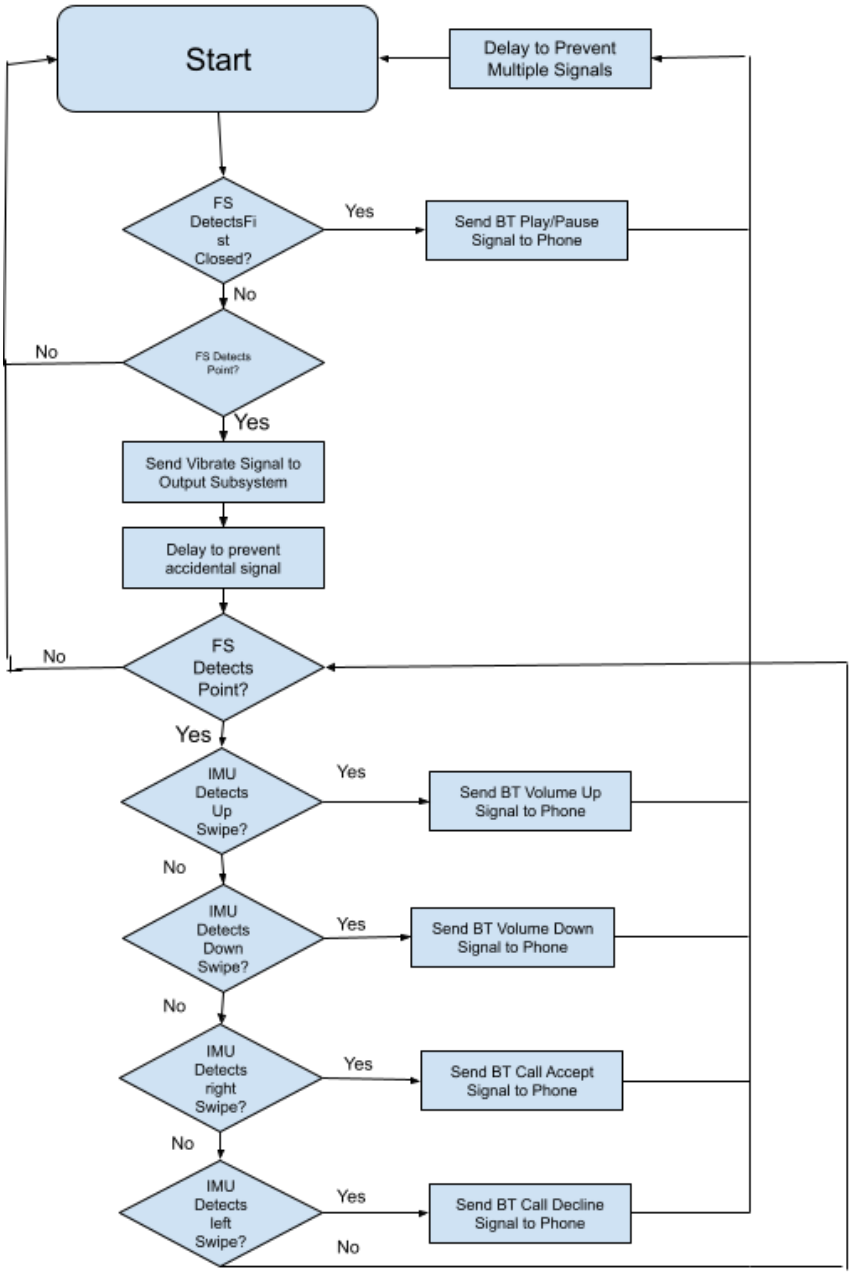
	up. Note down the accelerometer value for the y-axis (this reading should be -9.8 ± 0.1 ms^{-2}). 4. Place the IMU flat on a desk with the x-axis facing up. Note down the accelerometer value for the x-axis (this reading should be -9.8 ± 0.1 ms^{-2}). 5. Repeat steps 2-4 for another 3 trials. Calculate 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%.
The gyroscope readings from the IMU sensor should be accurate to within ±5% when measured under the same orientation.	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 facing up and x-axis facing right. Note down the 3 measured values from the gyroscope readings. 3. Randomize the IMU to a different orientation and then return it back to the previous position and measure the gyroscope readings. Calculate the error (percent difference) between each axis for the readings. Repeat this again another 2 times and measure the error compared to the first trial. Calculate the mean error from all of these trials for each axis. This error should be less than 5%. 4. Repeat this entire process for the following different orientations: a. Z-axis facing down and x-axis facing left b. Z-axis facing right and x-axis facing down

2.2.2 Control Subsystem

The control subsystem contains the microcontroller as well as the bluetooth module, which is already embedded in the microcontroller we're using: ESP32-WROOM. The microcontroller is responsible for taking the sensor input, detecting the user's hand signal, and sending the proper bluetooth signal to the phone. The microcontroller will be programmed such that when turned on, the bluetooth component enters pairing mode. This way, the user can pair the glove to their phone just by turning it on. If a gesture is detected, the bluetooth module

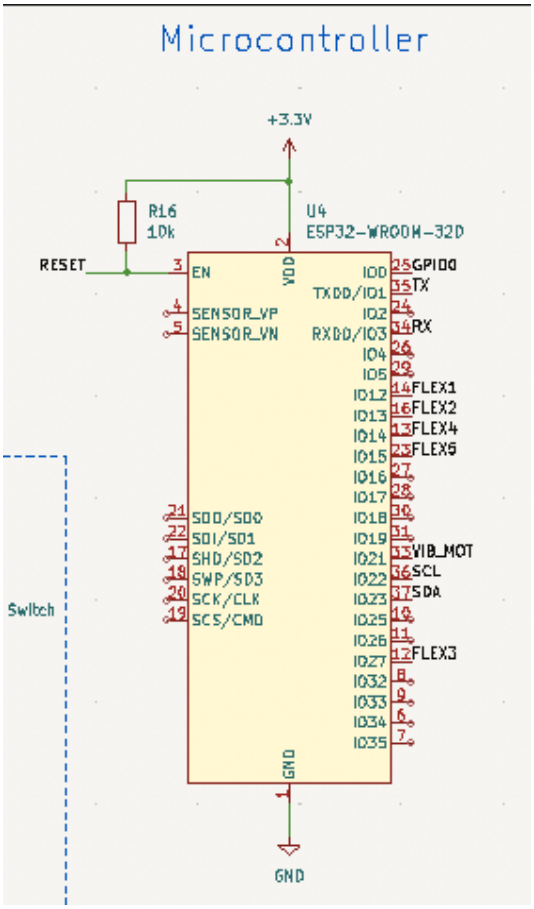
embedded in the ESP32 will translate this gesture into a bluetooth command that is sent to the user's phone. It will also connect to the vibration motor, so it can send the signal to buzz when a buzz-inducing gesture is recognized. The microcontroller will be connected to the IMU sensor, flex sensor, and vibration motor. The microcontroller will get its power from the power subsystem.

Figure 8: Control System Flow Chart



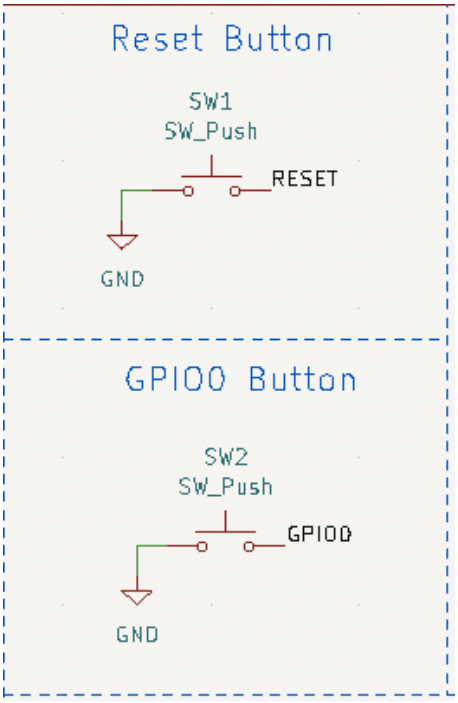
The controller checks for the play/pause gesture first because it doesn't require the point. Once the point is detected, the vibration motor vibrates to provide feedback to the user. There is then some delay to give the user some time to correct themselves if the point was an accident. At this point, the user can swipe in any of the four directions to accept/decline calls, skip a track, or go back to a previous track. The controller will cycle through the questions of "is the user pointing" or "is the user in a closed fist" until one of the two is detected, so the user will be able to perform either of those two actions at all times. Once a point is detected and verified, the controller will cycle through checking for a swipe in any of the four directions until it either detects a swipe or no longer detects a point. This simplifies the design because the first cycle only depends on the flex sensor data and the second cycle only depends on the IMU data.

Figure 9: Control System Circuit Schematic



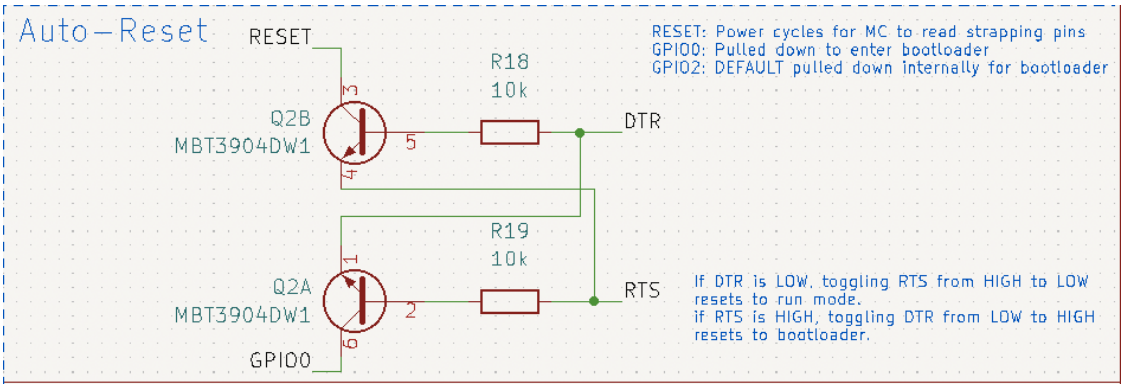
The microcontroller GPIO pins are connected to 5 flex sensors. For the flex sensors, we used the GPIO pins that have an ADC converter. The vibration motor GPIO pin will be set as an output pin in order to control the motor. The IMU sensor will use 2 GPIO ports as the SCL and SDA lines for I^2C communication. Since we're already using I^2C, these pins do not need to be connected to an ADC. We'll use the TX/RX pins for UART communication in order to flash the microcontroller. We will also connect the EN and IO0 pins to push buttons in order to indicate to the microcontroller when we want to flash from the computer or read from the microcontroller memory.

Figure 10: Button Circuit Schematic



When a button, Reset, or GPIO is pressed, the corresponding pin connects to ground.

Figure 11: Auto-Reset Circuit Schematic



When the reset pin is pressed, the enable pin of the microcontroller will be connected to ground, and when released, it will be pulled to high. This button allows us to power cycle the microcontroller. When the microcontroller is power cycled the microcontroller will read from a set of strapping pins. For our purpose, we only care about the bootloader configurations, which will be configured when the GPIO0 pin is set to low. Since we attached a button to the GPIO0 pin, when the button is pressed after the reset button has been pressed, the microcontroller will enter bootloader mode.

Table 2: Control Subsystem Requirements and Verifications Table

Requirements	**Verifications**
When the RESET button is pressed, the microcontroller is power cycled and if the RESET button and GPIO0 button are pressed together, 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 probes of an oscilloscope to the enable pin of the microcontroller and ground of the microcontroller. Connect the channel 2 probes of an oscilloscope to the GPIO0 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. Hold down the GPIO0 button and then release the RESET button. This should power cycle the microcontroller (the oscilloscope should read 3.3V ± 0.1V on channel 1). 6. Measure the voltage on channel 2. It should read 0V ± 0.1V. This ensures that we are in bootloader mode. The GPIO0 button can now be released.
Provide at least 15 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 (1k ohm resistor in figure 13). This measured voltage divided by 1k ohms should produce a value greater than 15 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. 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.
Translate identified gestures into bluetooth commands using HFP (Hands Free Protocol) and AVRCP (Audio/Video Remote Control Profile) 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

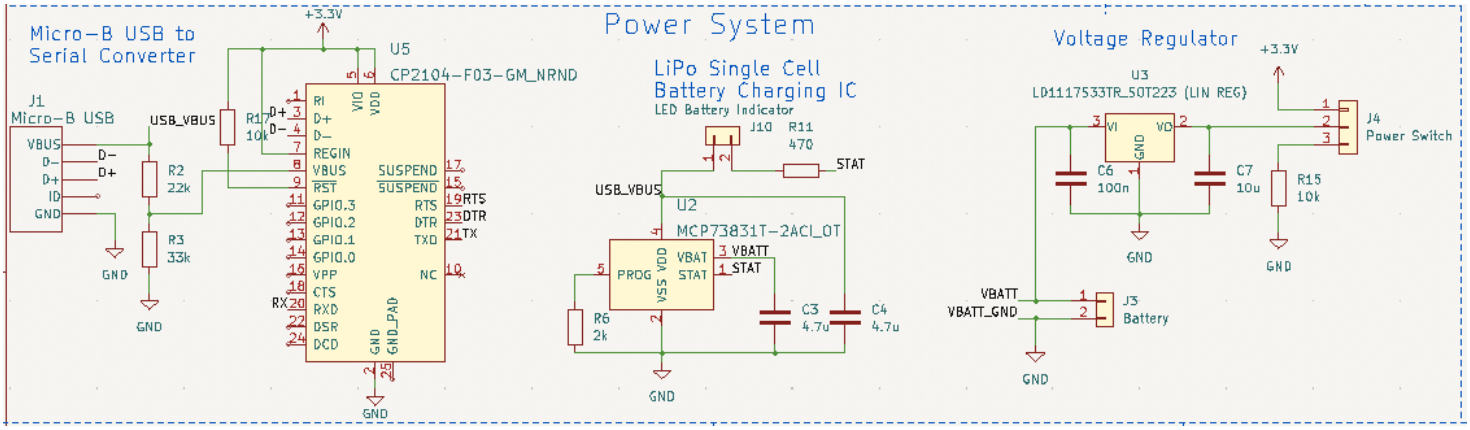
	following the HFP protocol for answering and rejecting calls and the AVRCP protocol for music playback controls (volume up/down, play/pause, next/previous track). 3. Ensure the commands are correctly executed on the iPhone when trying to answer/reject a call or when controlling music playback. 4. If there are any problems you encounter, you can debug the bluetooth protocols using the platform: WireShark.

2.2.3 Power Subsystem

The power subsystem will be responsible for providing power to all the components in our design: the vibration motor, microcontroller, IMU sensor, and flex sensors. The power will come from a portable Li-ion battery the user can charge via a USB charging port. .

The appropriate circuitry to create a rechargeable battery system that delivers 3.3V to the other subsystems will be in the power subsystem. A 5 V USB charger will be used to charge the battery. This will input 5 V to the charging IC. The charging IC then takes that DC voltage input and outputs a steady current to charge the battery. We have a Li-ion battery (3.7 V and 3 Ah) as our rechargeable battery. We also have a power switch that will be the on/off switch for the device. We can connect the battery to a voltage regulator to step down the 3.7 V battery output to 3.3 V. This 3.3 V will be delivered to the other subsystems.

Figure 12: Power Subsystem Circuit Schematic



To power the circuitry we will be using a Micro-B USB interface to charge the battery. We will use the 5V line from the USB port and connect that to the VDD of the battery IC. We will use the charging IC to provide a constant voltage, constant current source to charge the LiPo battery. The IC we're using can take the 5V line from the USB as input and output a steady 4.2V. The current output is also controlled using a resistor. According to the data sheet, a 2kΩ resistor from the prog pin to ground should output a steady current of 500mA, which should be safe to charge a LiPo battery which has a capacity of 3Ah. We also added decoupling capacitors to the input and output of the IC for more stability as well as an LED and resistor to the STAT pin in order to indicate when the charge is complete. The STAT pin is low when the battery is finished charging.

Since we will need a 3.3V line from our LiPO battery and the LiPO battery is 3.7 V, we will use a linear voltage regulator to provide a constant 3.3V output. The 3.3V output is then connected to a switch the user can activate in order to turn on the device. Also, in order to get serial communication from the usb port to flash the microcontroller, we will need to use a USB to UART bridge, which will take the data lines of the USB as input and output the TX and RX signals that will be sent to the microcontroller.

Table 3: Power Subsystem Requirements and Verifications Table

Requirements	**Verifications**
The battery must have a capacity of at least 2.39 Ah (in order to have a battery life of at least 3 hours) and a nominal output voltage of 3.7V ± 0.1V. The battery should also not overheat (does not exceed 85 degrees celsius).	1. Connect the terminals of the battery to a voltmeter and measure the output voltage (to determine nominal output voltage). 2. Connect the battery to the final circuit (as shown in figure 14) and connect an oscilloscope to the voltage output of the battery. Set a trigger at 3.5V. 3. Measure the time it takes to hit the trigger, and ensure that this is at least 3 hours. 4. After these 3 hours, measure the temperature of the battery using a non-contact IR thermometer and ensure that the temperature does not exceed 85 degrees celsius.
Battery charging IC does not overcharge the battery (does not provide it with more energy than it can contain - maximum of 3000mAh). This is achieved if the output current is less than 25mA (5% of charging current of 500mA) when the battery is fully charged.	1. Create the LiPo Single Cell Battery Charging IC from figure 12 with the battery connected. 2. Connect a power supply with an output voltage of DC 5V to the VDD pin and GND pin of the battery charging IC. 3. Let the battery charge until the LED Battery indicator turns on. 4. Then use a current probe to connect an oscilloscope to the VBAT pin of the Battery Charging IC and VBAT pin of the battery and measure the current flowing from the Battery Charging IC to the battery. This current should measure less than 25mA
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 circuit for the Voltage Regulator from figure 12. 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

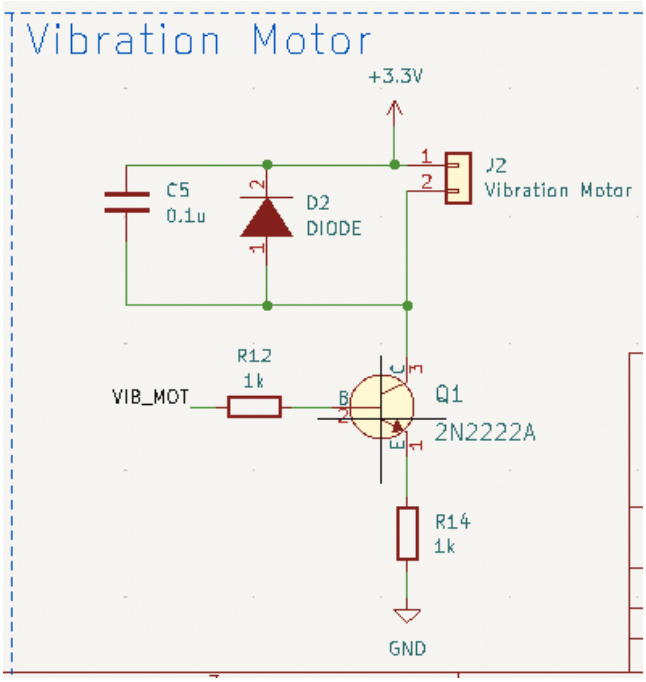
	voltage reading should be 3.3V ± 0.1V. 4. In the OFF state (switch is connecting terminals 2 and 3), the voltage reading should be less than 1mV.
Total maximum current draw for all components is 796.5 mA. Thus, the output of our voltage regulator must supply a current of 800mA ± 3mA to the rest of the circuit.	1. Create the final circuit from figure 14. 2. Program the microcontroller to constantly read input from the flex sensor and IMU and constantly output to the vibration motor. 3. Use a current probe to connect an oscilloscope to the output of the voltage regulator. Measure the current and ensure that it meets the required current value.

2.2.4 Output Subsystem

The output system will consist of a vibration motor. The vibration motor will get input from the microcontroller to control its functionality and will be biased with power from the power subsystem. The purpose of the vibration motor is to provide haptic feedback to users to alert them that their gestures have been recognised and sent as a command to their phone.

This subsystem obtains power from the power subsystem at 3.3V, which runs the vibration motor. There is also a data line connection from the control subsystem to the vibration motor in order to control when the motor is turned on and off.

Figure 13: Output Subsystem Circuit Schematic



Since the output of the microcontroller will not be able to provide enough current to drive the motor, we will use a BJT transistor as a switch. When the GPIO output is set to high, the collector and emitter pins will be connected and amplify the current provided at the base pin. This current will be provided by the 3.3V source connected to the motor. We also added a diode and capacitor in parallel to the motor in order to prevent any voltage spikes that may occur from the motor.

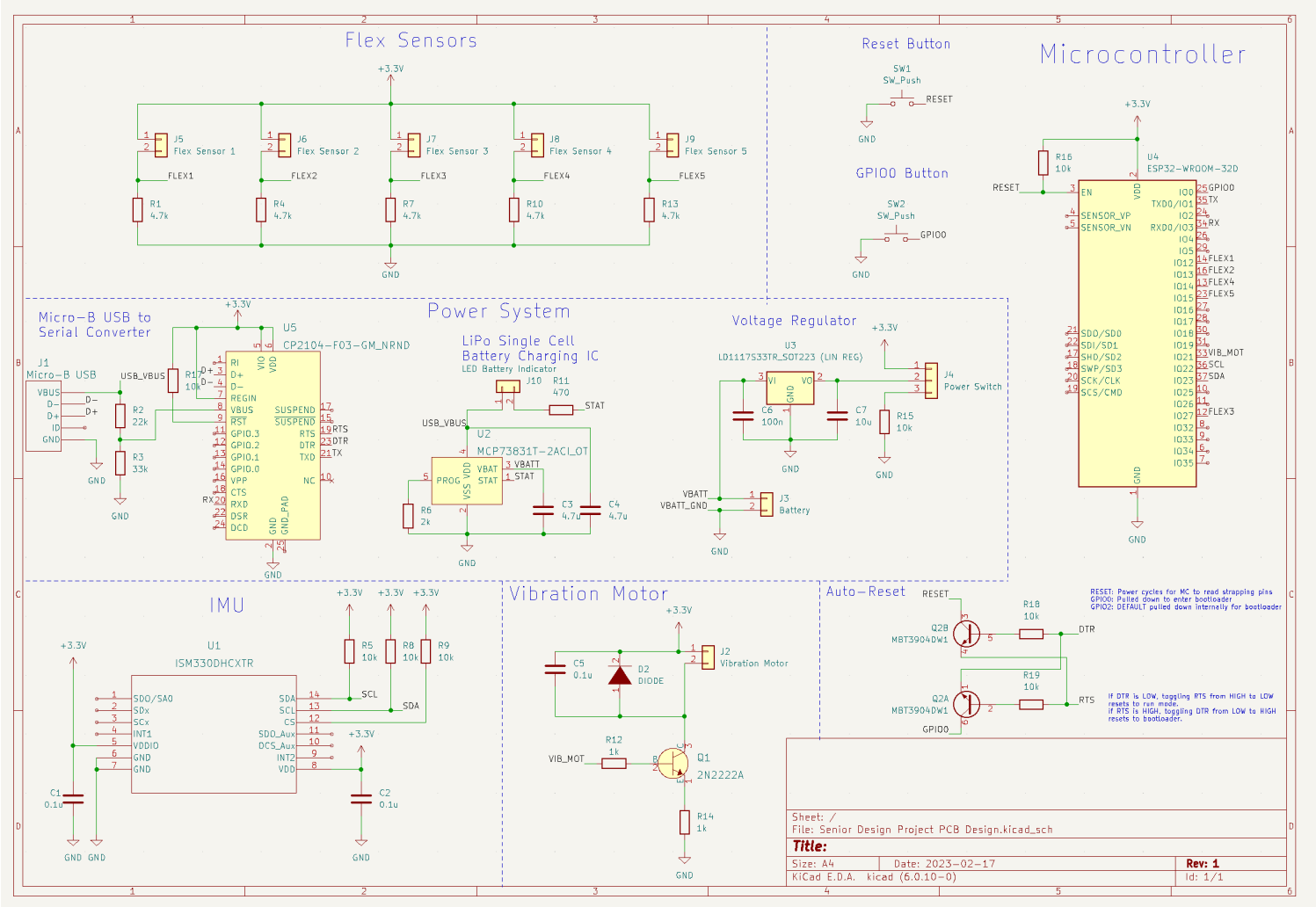
Table 4: Output Subsystem Requirements and Verifications Table

Requirements	**Verifications**
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 the microcontroller and to ground. 4. Program the microcontroller to send a digital high to pin GPIO21. Ensure that the voltage reading is 3.3V ± 0.1V. Inspect the vibration motor to check if it is vibrating. 5. Program the microcontroller to send a digital low to pin GPIO21. Ensure that the voltage reading is 0V ± 0.1V. Inspect the vibration motor and ensure it is not vibrating.

2.2.5 Full Circuit Schematic

Figure 14: Full Circuit Schematic



2.3 Tolerance Analysis

One risk to successful completion of the project is battery life. The goal is for the gloves to be able to run for 3 hours. Using some approximations with real components on the market, we can determine if this is doable. The components we're using that require power are the 5 flex sensors, 1 IMU, 1 vibration motor, and 1 esp microcontroller. Looking at datasheets of real components on the market, we can look at the max current draw from each component, add it up, and use this as the average current draw of the glove from the battery (1). This way, we avoid underestimating the power consumption of the glove. Then, this approximate current average, I_{avg}, and the battery capacity, C_{avg}, can be used to find the battery life, T_{bat} (2).

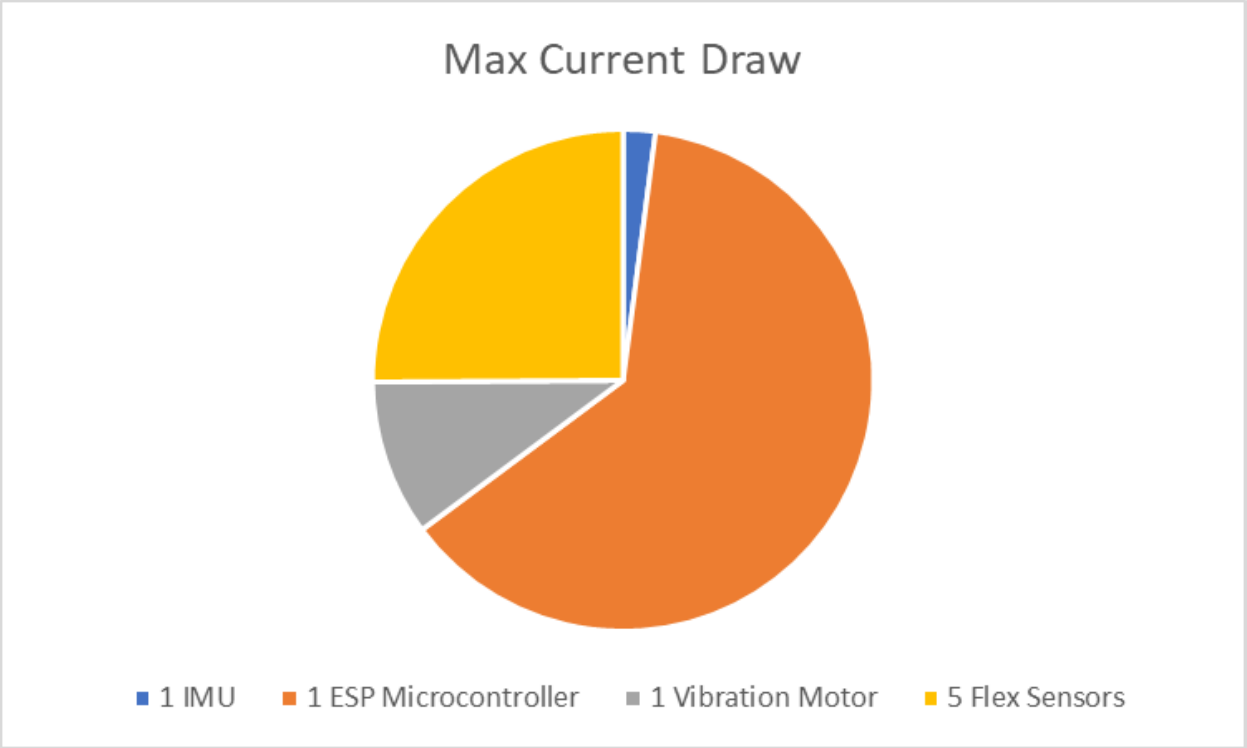
$$I_{avg} = \Sigma\, I_{component} \quad (1)$$

$$T_{bat} = C_{bat} / I_{avg} \quad (2)$$

Table 5: Max Current Draw of Each Component

Component	Max Current Draw (mA)
1 IMU	16.5 [5.5]
1 ESP Microcontroller	500 [5.4]
1 Vibration Motor	80 [5.3]
5 Flex Sensors	200 [5.2]
Total	796.5

Figure 15: Current Draw Component Breakdown



Looking at the pie chart, it seems the flex sensors take up around a quarter of the total power and the microcontroller takes up around two thirds. For power optimization, we could add a feature that turns the glove off when no motion is detected for a certain amount of time. Since the battery has a capacity of 3 Ah [5.1], the approximate minimum battery length is 3 hours, 46.2 minutes. In order to achieve our goal of 3 hours, we would need an average current draw of 1 A. This calculation was done assuming the highest possible current draw from all the components, so we have some wiggle room in terms of power.

3 Cost and Schedule

The average ECE graduate makes about $80,000 per year, which translates to roughly $40 per hour. We're planning on and have been working an average of 12 hours per week because this is a 4 credit hour class. Not including spring break, that's 15 weeks. Because there's three of us, the total labor costs add up to around $21,600.

A table of all parts we plan to use in the project along with their prices is in the following table.

Table 6: Overall Costs Table

Quantity	Part	Price
x1	ESP32 Dev Board	22.50
x1	ESP32 Microcontroller	2.97
x5	Flex Sensor	32.50
x1	IMU Dev Board	24.95
x1	Vibration Motor	2.25
x1	IMU	12.23
x1	Battery IC	.69
x1	Battery	14.95
x1	USB-C Port	.86
x1	Linear Voltage Regulator	.60
x1	Power Switch	.52
x1	micro-B USB Receptacle	.46
x1	USB-UART Bridge	5.70
x1	Glove	10
	Total	**131.18**

The next table is a timeline for the semester schedule.

Table 7: Schedule Table

Week	Goals	Person
3	1. Project Idea Initial Post Approved	1. Everyone
4	1. Finished Project Proposal 2. Put initial parts order 3. Talked with machine shop about design for enclosure	1. Everyone
5	1. Finish PCB schematic 2. All parts ordered	1. Mehul and Sai 2. Oilver
6	1. Design Document Finished 2. Edit Project Proposal for resubmission if needed 3. Finish PCB layout 4. Start breadboard testing components based on schematic (if parts arrive)	1. Everyone 2. Everyone 3. Mehul 4. Sai and Oliver
7	1. Finish up breadboard testing of parts 2. Find/debug issues with circuit and reflect changes in schematic/layout 3. Send out PCB gerber files	1. Sai and Oliver 2. Mehul 3. Mehul
8	1. Start microcontroller programming: a. interface with sensors b. Determining gestures c. send correct bluetooth commands	1. Everyone
9	Spring Break	
10	1. More Microcontroller Programming 2. Mount all components on PCB when it arrives 3. Test PCB functionality and edit design if required 4. Send out updated PCB gerber files 5. Give PCB to machine shop to start working on enclosure	1. Oliver and Sai 2. Mehul 3. Mehul 4. Mehul 5. Mehul
11	1. More Microcontroller Programming 2. Work on physical model of glove 3. Individual progress reports	1. Oliver and Sai 2. Mehul 3. Everyone individually

12	1. Fully functional software and hardware interface 2. Finish up mounting parts on glove and attach enclosure from machine shop	1. Everyone
13	1. Work on, finish Final Report 2. Team Contract Fulfillment	1. Everyone
14	1. Mock Demo 2. Work on, finish Final Presentation	1. Everyone
15	1. Final Demo 2. Mock Presentation	1. Everyone
16	1. Final Presentation 2. Final Report	1. Everyone

4 Ethics and Safety

For section I of the IEEE code of ethics [5.6], our project doesn't have too many issues. The gloves don't connect to the internet, so privacy shouldn't be an issue. The only thing that can theoretically be done maliciously is messing with a person's music, volume, or calls. This will be mitigated by requiring any established bluetooth connection to be consented to by the user of the phone. Our group will also uphold a high standard of safety, integrity, and responsible behavior in the process of working on the project.

For section II of the code of ethics, there are a couple safety concerns. Once concern is the risk of an accidental rapid increase in volume. This could cause ear damage to the user or disorientation that could lead to an accident. We will remedy this using a feedback system, which will notify the user when the glove is accepting inputs. Another safety issue is the PCB and other electronic components. These components can heat up, which can cause burns to the user if the components come in contact with the user's skin. To prevent this, we can mount the electronics

in a way that prevents skin-component contact. We also plan to treat each other as well as other groups fairly and with respect.

For section III, we plan to support and hold to account ourselves as well as other groups in following the IEEE code of ethics. Effective teamwork is essential in order for us to be able to successfully create our Bluetooth Enabled Gloves for Controlling Music. Thus, we will all act in a professional manner whilst not engaging in any discrimination and treating all people fairly and with respect.

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

[5.6] *IEEE code of Ethics* (no date) *IEEE*. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html (Accessed: February 4, 2023).