# iBand

By

Panav Munshi

Rutu Brahmbhatt

Saaniya Kapur

Final Report for ECE 445, Senior Design, Spring 2022

TA: Hojoon Ryu

4 May 2022

Team No. 34

## Abstract

The goal of this project is to build an armband that encompasses sensors that read biosignals generated by a user's movement and well being. Specifically, in our design we used sEMG, GSR, IMU, and Pulse sensors which will be discussed in further detail in the rest of our report. Overall, the armband needs to be able to read positional data, such as change in the yaw, pitch, and roll of the user's forearm. Furthermore, it is imperative that these goals are achieved by keeping the overall cost of the product less than $100.

These goals were motivated by the discontinuation of the Myo Armband - an armband that read biosignals to decipher gestures made by the user. Its discontinuation inspired the need to create a product that incorporated the Myo Armband's functionality and other, new functionalities, allowing the product to appeal to a larger audience.

After the development of the armband, aptly named as the iBand, it was found that it was correctly able to distinguish the electrical impulses generated by the user's muscles and was accurately able to read the user's pulse and GSR data, and the product met all of the defined high-level requirements.

## Table of Contents

# 1. Introduction

## 1.1 Problem Statement

Surface electromyography (sEMG) is a non-invasive computer based technique that utilizes electrodes placed on an user's forearm to record electrical impulses produced by the nerve's simulation of the skeletal muscle. Products that leverage the information provided through sEMG signals for rehabilitation, educational, and recreational purposes already exist in the market today. However, these devices are unable to withstand continuous use, do not provide the user with an ability to understand the data being collected and are expensive - ranging from thousands of dollars [1] with the cheapest option being discontinued [2]. Therefore, the options that are available in the market today are not ideal for rehabilitation use cases in which the nurses do not have a background in sEMG signal analysis. As well as, in scenarios where an expensive device is unobtainable and is often used with a greater frequency such as in educational and recreational use cases.

## 1.2 Solution Overview

Our solution to increase public accessibility and understanding of sEMG devices was to create an inexpensive, durable, and portable replacement to the now discontinued Myo Armband. This is done through the use of six sEMG sensors coupled with medical grade electrodes, an Inertial Measurement Unit (IMU), a Galvanic Skin Response sensor (GSR), and a Pulse sensor. The IMU combines the use of gyroscopes and accelerometers to measure angular and linear acceleration providing information as to position and velocity. The GSR measures the skin's conductivity which is directly proportional to sweat gland activity. The Pulse sensor measures the change in the amount of infrared light being transmitted through the body as dependent on the rate of blood flow. This design will then transmit the data collected by these sensors to a built-in user interface located on the user's laptop through bluetooth.

### 1.3 Block Diagram

Figure 1 illustrates the various subsystems that collectively create our solution. Specifically, we had a power subsystem that delivered power to the microcontroller and consequently the sensor unit as a whole. The information outputted by the sensor subsystem was then sent to the software subsystem which was ingested through the backend and portrayed on the frontend.

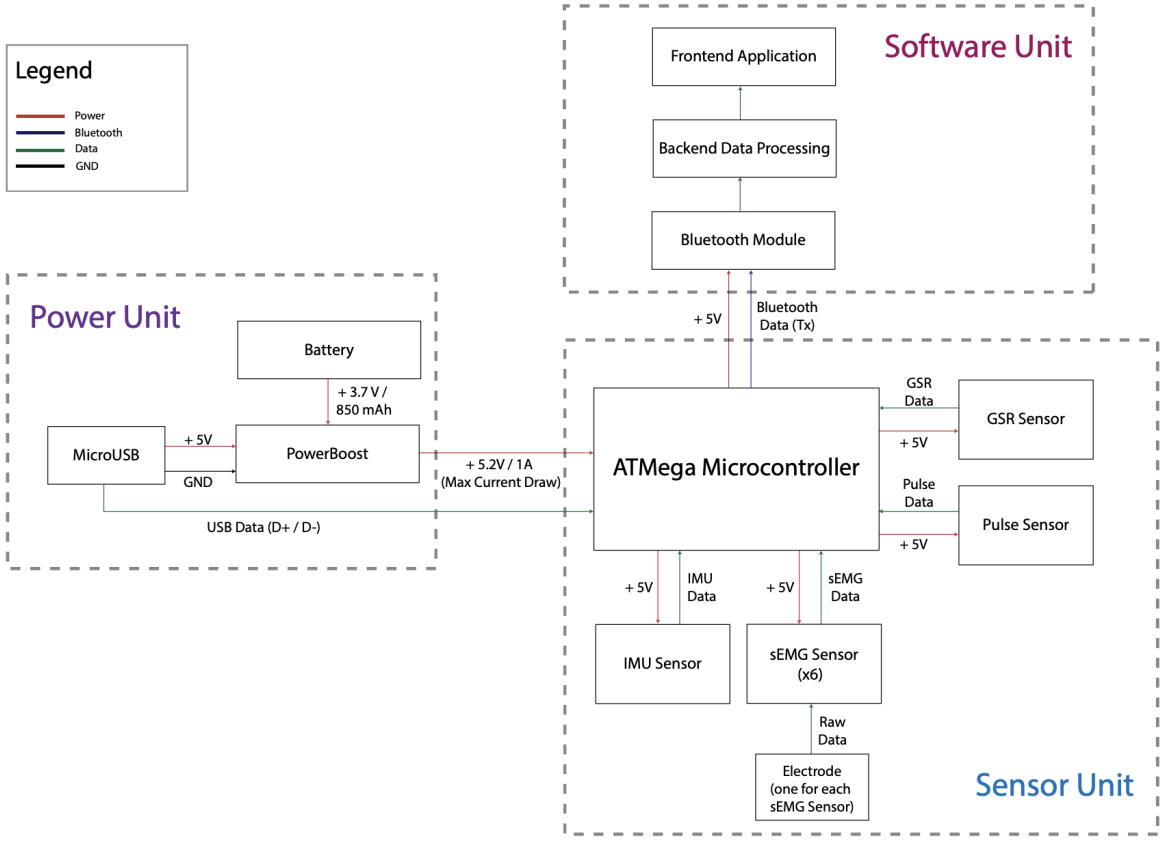


*Figure 1 : Block diagram of the subsystems in our proposed solution*

### 1.4 High - Level Requirements

1. **Sensing and Size:** The length of the armband must be between 22 cm and 44 cm and have the ability to change circumference without the relative position between sensors changing. Our six sEMG sensors (each 22mm long) , IMU sensor (25.4 mm long), GSR sensor (24mm long), and a Pulse sensor (15mm long) must fit within the constraints of the armband length.
2. **Compact Size:** The next quantitative characteristic this project must exhibit is that the new design must be smaller and more compact than the current design, which has the dimensions of 14.8cm wide x 14.5cm long x 1cm tall. To prevent a bulky design, the maximum size of the printed circuit board (PCB) must be less than half of this at 10 cm.
3. **Data Collection:** This project must be able to collect data for at least two hours with minimal to no discomfort put onto the user. The raw data from the sensors must be displaced in real-time and provide the user with some feedback as to how to obtain the most accurate data - such as where to place the armband.

# 2. Design

## 2.1 Design Procedure

### 2.1.1 Armband Design

The considerations that were made during the design process mainly resided in the fact that in order for the sEMG sensors to measure accurate readings they needed to be positioned on top of the various muscle groups. Specifically from a design perspective, this means that when the armband is stretched and contracted the relative position in between each of the sEMG sensors needs to remain constant.

The placement of the sensors was determined through an analysis in which we will be focusing on the error margins of the electrode placement with respect to the width of the muscle group in consideration. The muscle group in consideration for this analysis is the *Flexor Carpi Radialis*. As seen from a study performed by *B K Potu et al* [3], the average width of the muscle belly towards the distal end is about 1.99 ± 0.72 cm. Given that the electrodes that will be used in this project have a width of 1.545 cm, the error in the placement of the sensor with respect to the center of the muscle can vary from

$$\pm \frac{(1.99 - 1.545)}{2\ cm}\ to\ \pm \frac{((1.99 \pm 0.72) - 1.545)}{2\ cm}$$
$$= \pm 0.2225\ cm\ to \pm 0.5825\ cm.$$

However, as the diameter of the leads of the electrode is 0.9017cm, the maximum range of permissible error before the leads are not in contact with the center of the muscle (with respect to the width) is:

$$= \pm \frac{0.9017}{2\ cm}$$
$$= \pm 0.4509\ cm.$$

If the electrode is outside this range, the data collected from the sensor can produce data with high levels of noise, leading to substantial inaccuracies.

Furthemore, the armband, itself, needed to be compact as specified in one of our high level requirements with the ability to withstand repetitive use. Therefore, when determining what type of material to use in our final design these requirements were of significance.

### 2.1.2. PCB Design

Our most important design consideration for the PCB was that the armband had to be downsized from the completed version in phase 2, which meant that it had to fit within the constraints mentioned in our second high-level requirement. Based on the circuits from phase 2, we determined the necessary components for our PCB and eliminated any unnecessary parts. We did this in order to reduce size, but also to significantly reduce the parasitic inductance in the long wires between the circuit and sensors in order to obtain more accurate data.

### 2.1.3. Software Design

There were several considerations that were made while designing the software for the iBand. As required by our sponsor, the iBand needed to collect data from the sensors in a parallel manner and package this data in order to be sent over to the Bluetooth Module. Furthermore, the compiled binary of the code needed to have a size of less than 32 kilobytes, and the size of all global variables - variables that can be accessed by all functions - needed to have a size less than 2 kilobytes. This is because the ATMega328P [4] microcontroller had a flash - the area where the program is stored - size of 32 kilobyte and a SRAM - the area where variable data is stored - size of 2 kilobytes. Lastly, the computer that received the packaged data from the iBand's Bluetooth Module needed to display the sensor data in realtime, preferably in the form of graphs.

## 2.2 Design Details

### 2.2.1 Armband Design

The design of the armband that encompassed our PCB, IMU, Pulse, GSR, and six sEMG sensors underwent three different design iteration cycles before we decided on a final prototype.

The first iteration consisted of designing a glove instead of an armband with the intention of being able to obtain more holistic data in regards to the nuanced movements being made by the user. This idea was inspired by one of the use cases that our sponsor had mentioned which involved using this prototype as an aid for surgical students when practicing their surgical technique. Upon further research, the proposed design was intended to look similar to Figure X but with medical grade material [5].

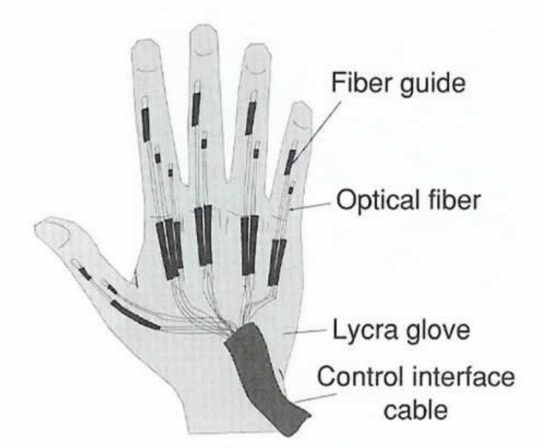


*Figure 2 : Image [6] of potential sensor embedded glove design*

The second design iteration consisted of six sEMG sensors clasped onto a rubber pipe [7] in which the individual wires from every sensor would be run through, as shown in Figure X. The sEMG sensors, themselves, would be encompassed in a chassis with the medical-grade electrodes being held to the user's forearm.

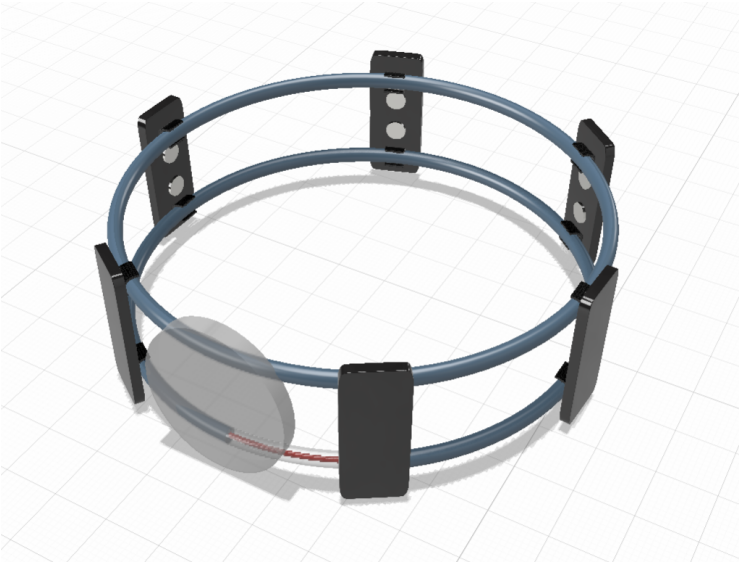

*Figure 3 : Design with rubber pipes and sEMG chassis and PCB encasing*

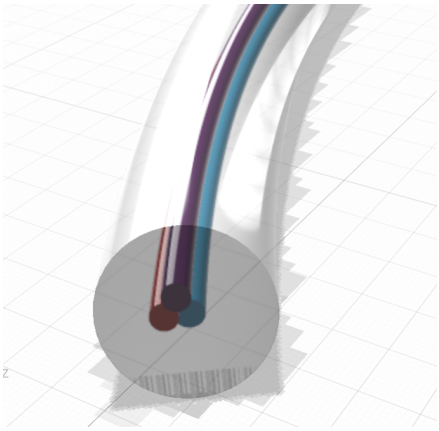

*Figure 4 : Wire management housed in the rubber pipe*

The third iteration, similarly, consisted of six encasings which encompassed the individual sEMG sensors and medical grade electrodes. However, each of these encasings were connected to each other through the use of rubber bands and zipties in order to ensure that the relative position of each sensor changed by a uniformly constant amount when the armband was expanded. The IMU sensor was then placed on top of one of these encasings as the sensor, itself, did not need to be placed directly on the user's skin in order to achieve its expected behavior. Due to the fact that the pulse sensor and GSR have a higher sensitivity as compared to the sEMG sensors, they needed to be placed on the user's fingertips in order to achieve the quality of measurements needed to see a discernible change in the output values. Lastly, in order to keep this design as compact as possible, the PCB and Bluetooth Module were placed in a box that was worn just above the armband.

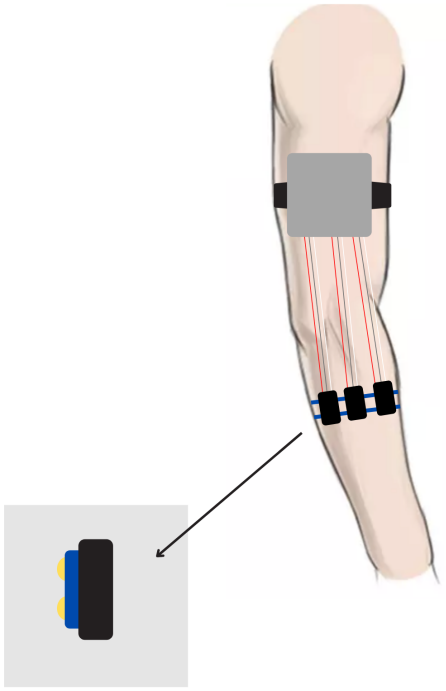

*Figure 5 : Demonstration of how the proposed solution will look on the user's forearm*

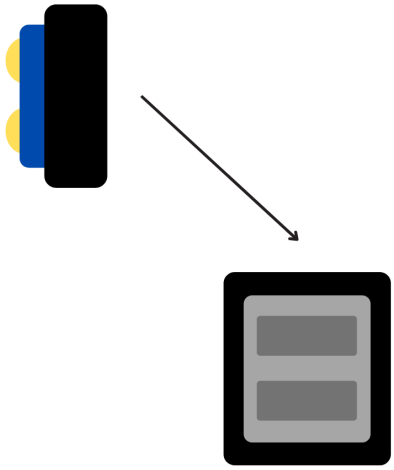

*Figure 6 : Demonstration of how the sEMG sensor will be placed in the chassis*

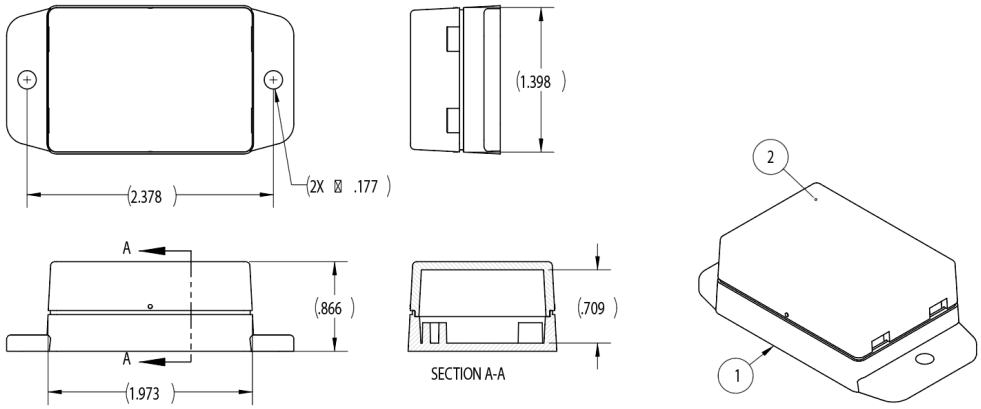


*Figure 7 : CAD drawing of the entire enclosure assembled*

*These images were taken from Polycase. com*

### 2.2.2. PCB Design

The second phase of the iBand utilized an Adafruit PowerBoost 1000C, which influenced the design of our power subsystem. We used the necessary input voltages of the sensors in the sensor subsystem as described in our block diagram in order to determine the necessary outputs of the power subsystem. The PowerBoost leveraged the MCP73871-3CC to regulate the temperature and current of the battery to prevent overheating, as well as the TPS61090RSAR, which is a boost converter. Our PCB also encompassed the MCP73871-3CC and a similar boost converter - the TPS61090. From the datasheets, we determined the necessary parts, connections and values for all components. Figure 8 shows the final schematic of the power subsystem.

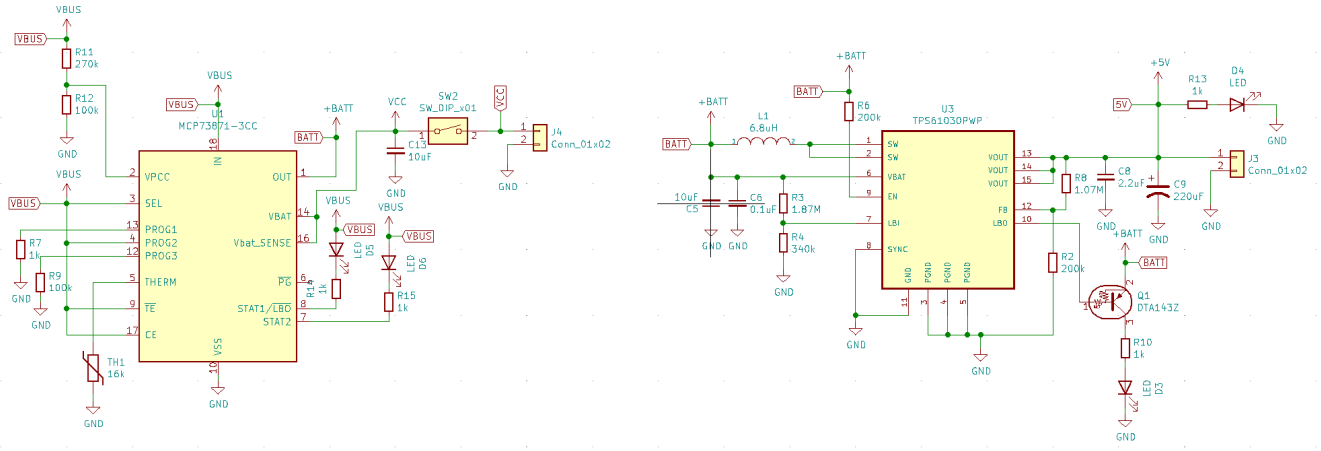


*Figure 8 : Boost converter and battery management system schematic (power subsystem)*

Furthermore, the phase 2 design used the Arduino Nano to record and display the raw data from a few of the sensors. The microcontroller and USB-to-UART Integrated Circuit (IC) were chosen because they were used in the Arduino Nano present in phase 2. The microcontroller was the ATMega328 and the USB-to-UART IC was the FT232RL. For the microcontroller, we used the X000048, which is the ATMega328 with a pre-installed arduino bootloader. For the USB-to-UART IC, we used the CY7C53225-28PVXC.

The first version of our PCB had the FT232RL, but it ran out of stock before we were able to order it so we had to order the CY7C53225-28PVX, instead, with a slightly different schematic. The CY7C64225-28PVXC provides the same functions as the FT232R, with 256-byte transmit and receive buffers [8] rather than 128-byte for receiving and 256-byte for transmitting [9]. The X000048 was mostly similar to the ATMEGA328 that we planned to use in the first version, but it lacked the necessary number of analog pins. Because of this, we had to add an Analog-to-Digital Converter (ADC) to the second version of our PCB. Other changes from version one of the PCB to two included the elimination of an external linear voltage regulator, the LM117, which is included in the Arduino Nano. We determined that this was not necessary as the TPS61090 provides the same constant output voltage as required by all of the external sources.

The PCB also includes headers that can be seen in Figure A.1 of Appendix A. These headers are used to connect all of the sensors to our PCB along with the Bluetooth Module.

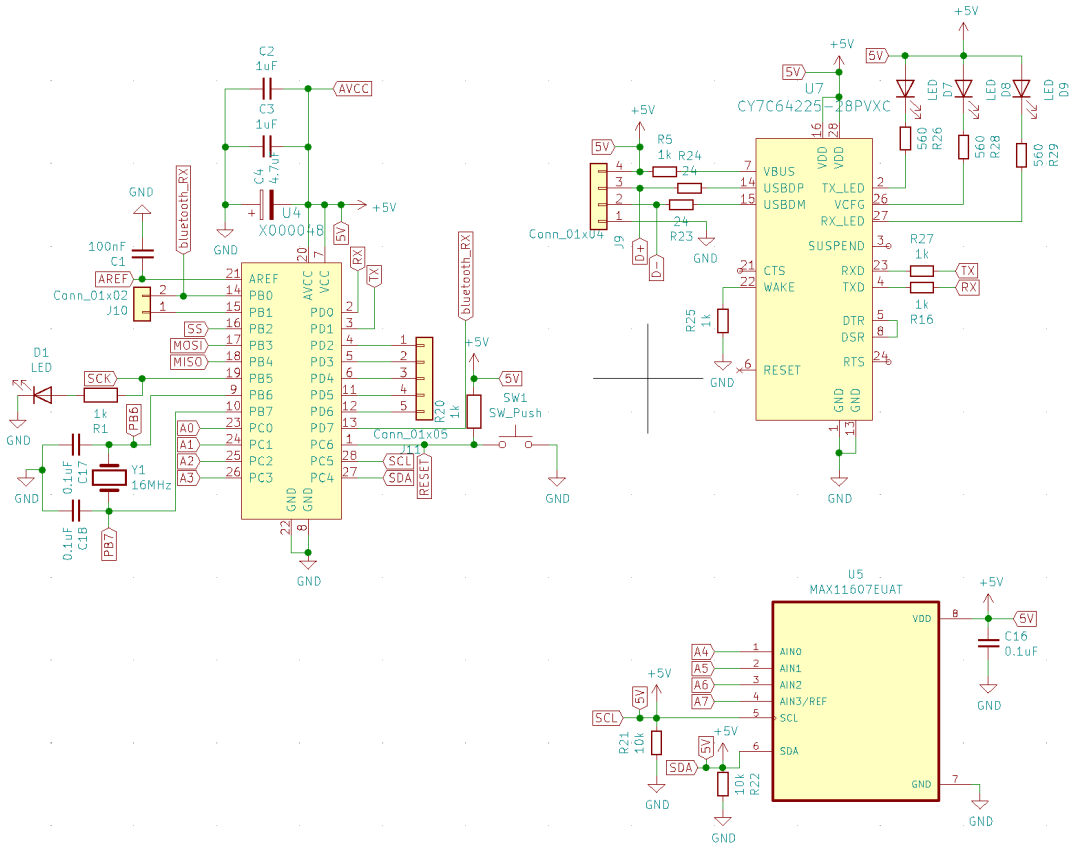


*Figure 9 : Microcontroller, Analog-to-Digital Converter, and USB-to-UART IC Schematic (sensor subsystem)*

### 2.2.2. Software Design

As the firmware code for the microcontroller was developed using the Arduino IDE, we were able to utilize libraries that were written for the Arduino, itself. This decision made it easier to reuse code that was publicly available, rather than having to start development from scratch.

In order to collect data from the sensors in a parallel manner, a multi-threaded approach for data collection was needed. The Arduino library known as *Arduino Threads* [10] provided the necessary functionality to achieve this. Essentially, the library implements proto-threading, which is a method of running tasks - or threads - sequentially at a set time-interval thereby creating an illusion of parallelism. True parallelism cannot be achieved on an ATMega328P microcontroller as it only has one processor core - a unit that reads instructions to perform specific actions. By wrapping the code for every individual sensor in threads, we were able to create an illusion that our microcontroller implemented the parallelism mentioned above.

Next, the collected data had to be packaged into a format that could be sent over Bluetooth to a PC through the Bluetooth Module. The data was packaged in the form of a JavaScript Object Notation (JSON) for various reasons. Firstly, JSON is a lightweight and compact framework as it comprises only text. Secondly, JSON is a widely used format to send data and therefore there are many libraries that have built on top of it. Specifically, we used the ArduinoJSON library [11] to package our sensor data into a JSON as it helped us in keeping the flash and SRAM sizes within their constraints.

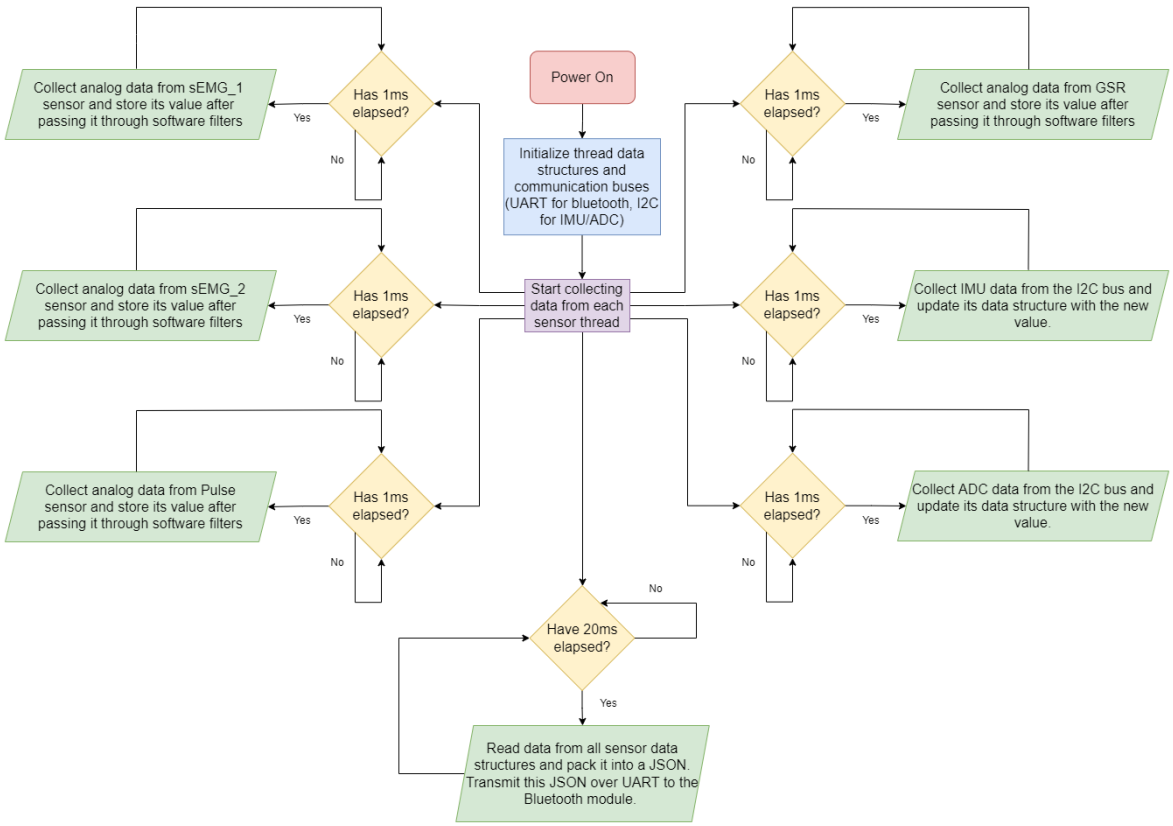


*Figure 10 : Pseudocode of the code for the microcontroller*

The flowchart in Figure 10 describes the state diagram written with the considerations discussed previously. The timing interval of 1 millisecond in the conditional statement was decided upon through the calculation of the difference between the time at the start of the execution and the end of the execution.

After the data was collected by the PC from the Bluetooth Module, it had to be processed into graphs. In order to do this, we ran a local backend server which parsed the JSON data received from the sensors and displayed them in real-time charts on our frontend application.

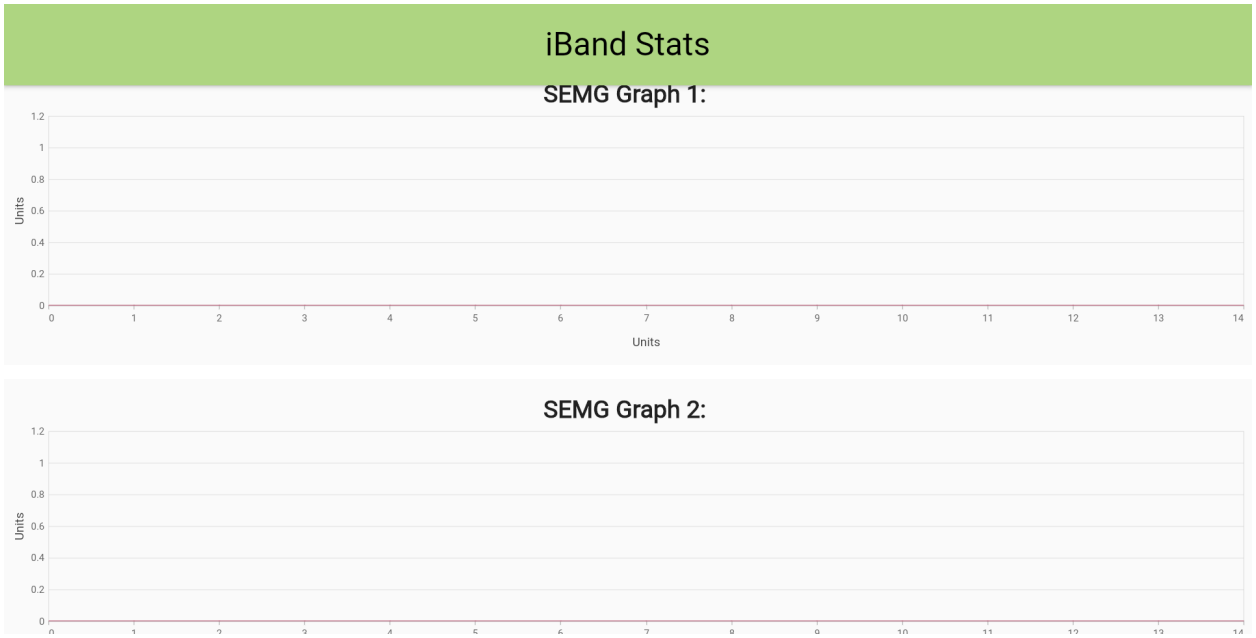


*Figure 11 : Frontend displaying sensor values on a real-time graph*

Lastly, in order to protect the data from side-channel security attacks, we employed the use of encryption. The JSON package was encrypted by the microcontroller using the Crypto [12] Arduino library before it was sent to the PC via Bluetooth. Next, the PC received the data via Bluetooth and isolated the encrypted data. Finally, we used Python's Crypto Library to decrypt the received data and proceeded with processing it.

## 3. Design Verification

We were able to perform separate tests on each of the subsystems of the project in order to verify their functionality. The Requirements and Verification tables for each subsystem are provided in Appendix A at the end of this document. Figure A.2 in Appendix A shows the final product for the armband that we have built for this project.

### 3.1. Power Delivery Unit

This subsystem supplies power to the microcontroller in the software subsystem and recharges the lithium-ion battery. It includes a boost converter and battery charge management controller, referred to as a PowerBoost, which handles both supplying power to the microcontroller and charging the battery. It has a voltage regulator which steps up the voltage, and a battery management system that is able to regulate the temperature and current of the battery to prevent overheating. The PowerBoost is powered by a 5V input and a lithium-ion battery at 3.7V for 850mAh. It is able to step up the voltage to provide an output voltage of 5.2V with a maximum current draw of 1A. A limitation of the PowerBoost is that it does not pass through the data bus lines from the MicroUSB header to the USB Power Out port. In order to alleviate this problem, we are directly connecting the D+/D- bus lines to the microcontroller. This allows us to use one single USB port for both charging and reprogramming the microcontroller.

We performed the tests listed in Table A.1 of Appendix A in order to verify the success of our requirements. We were able to verify the output voltages and currents were within acceptable margins for the specified inputs. However, we were only able to verify that the rechargeable battery supplies voltage for 2 hours rather than 3 because our rechargeable battery was misplaced in the lab.

### 3.2. Software Unit

This subsystem consists of a Bluetooth Module, some backend processing, and a frontend user interface. The Bluetooth Module will receive a stream of serial information from the microcontroller through the connection from the $R_x$ to $T_x$ pin connection found on the Bluetooth Module and the microcontroller, respectively. This information will then be processed and cleaned by python scripts found in our backend system in order to enable a clear depiction of the individual signals produced by the sEMG, IMU, GSR, and Pulse sensors.

We were able to follow the verification process listed in Table A.2 to confirm the success of the requirements for the software subsystem. We met the requirement that the serial communication between the microcontroller and Bluetooth Module has to be at a rate of at least 9600 bauds (104.167 µs). In fact, the serial communication occurred at a rate of 115000 bauds (8.696 µs). We decided to proceed with a higher baud rate as it allowed for less noise and better communication.

### 3.3. Sensor Unit

This subsystem consists of all the sensors - six sEMG sensors, one IMU sensor, one GSR sensor, and one Pulse sensor - that will be used in this project. The sEMG sensors, as alluded to earlier, will be measuring electrical impulses from the muscles in the forearm. Using six sensors, several muscle groups can be

targeted, which allows the iBand to measure several different kinds of gestures. The IMU sensor aids in gesture recognition by providing the spatial position of one's arm. The GSR sensor allows the iBand to monitor the user's perspiration levels - an indirect indicator of one's emotions and stress levels. Lastly, the Pulse sensor aids the GSR sensor as heartbeats per minute can be added as an additional datapoint to measure a user's stress levels and emotions. All of the requirements for the sensor subsystem were tested as listed in Table A.3 of Appendix A, and all of them were met.

Figure 12 and 13 portray the sEMG signal data that is obtained when a certain muscle group is flexed. Figure 12 shows the basal sEMG signal when the muscle group associated with its sensor is at rest. Figure 13 shows the change in the amplitude of the sEMG signal when the muscle group associated with its sensor is flexed.

This form of data collection was repeated for each of the six sEMG sensors using a different motion that targeted each one of the muscle groups that a sensor was reading from, independently.

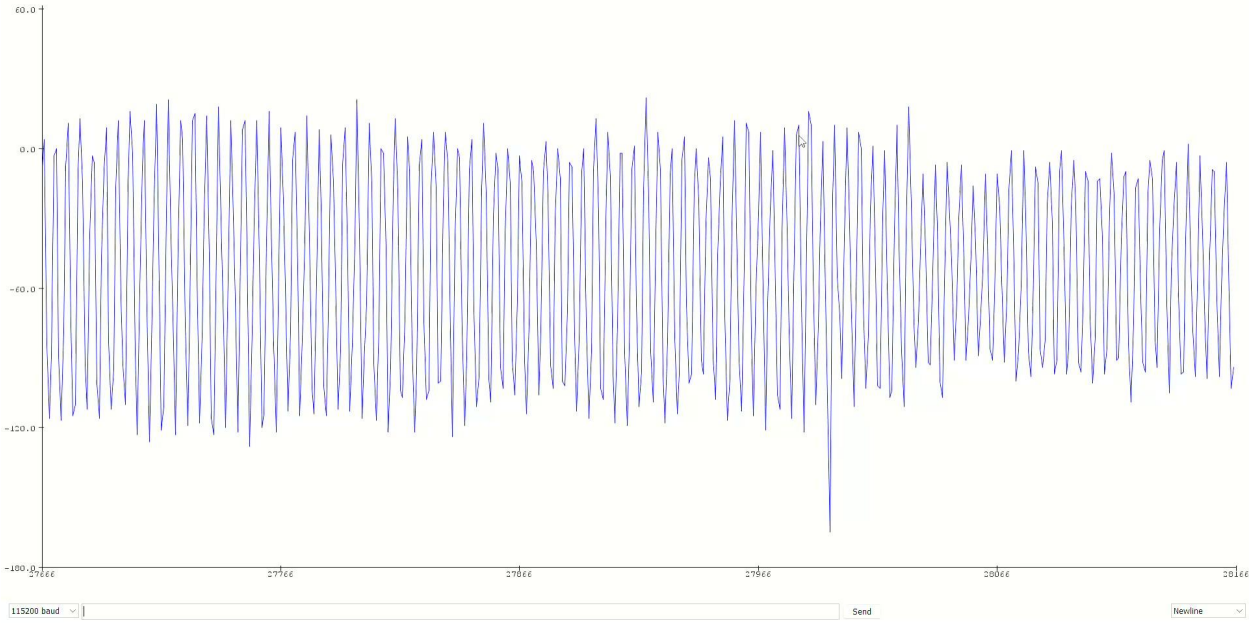


*Figure 12 : sEMG Sensor 1 reading when the muscle is unflexed*

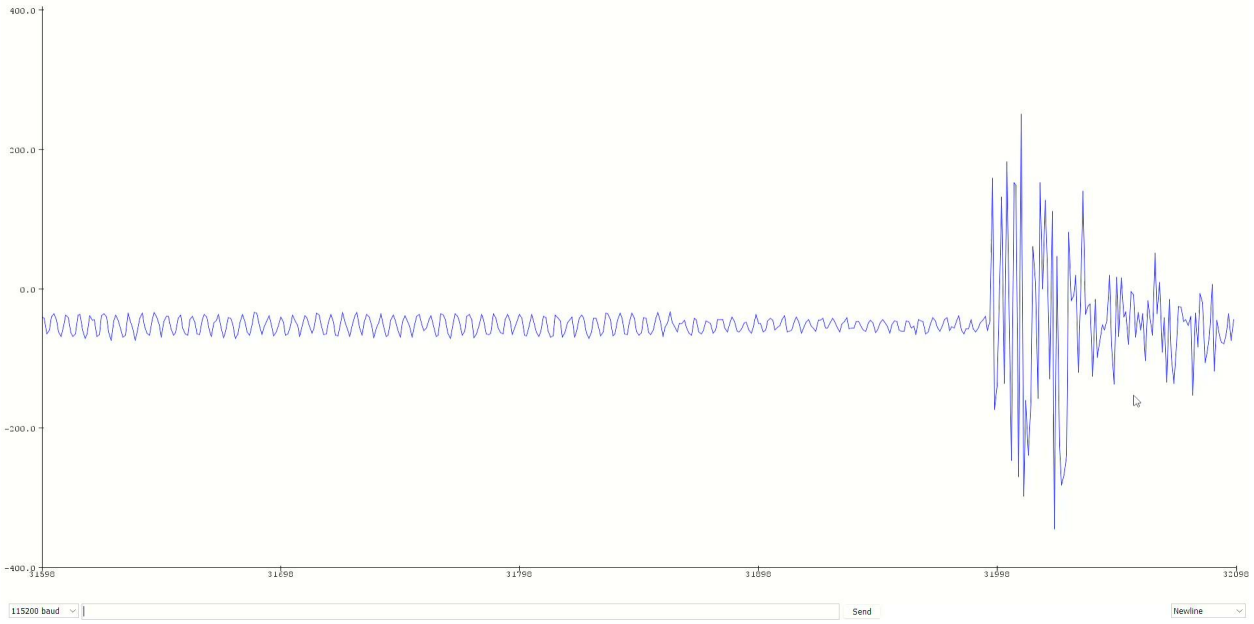


*Figure 13 : sEMG Sensor 1 reading when the muscle is flexed*

Figure 14 and Figure 15 portray the GSR signal measurements recorded from the subject's fingertips. Figure 14 shows the GSR signal recorded when the subject was at rest. Figure 15 shows the GSR signal recorded when the subject was under simulated stress - the user was asked to breathe fast in order to simulate a stressful situation.


*Figure 14 : GSR Sensor reading when the user is not stressed*


*Figure 15 : GSR sensor reading when the user is stressed*

Figure 16, Figure 17, and Figure 18 portray the IMU signal measurements recorded when a user moves his/her arm. Figure 16 shows the Pitch signal recorded. The sinusoidal wave-pattern was a result of displacing the IMU in a sinusoidal-like fashion. Similarly, Figure 17 shows the Roll signal recorded and Figure 18 shows the Yaw signal recorded with the same sinusoidal waves present.

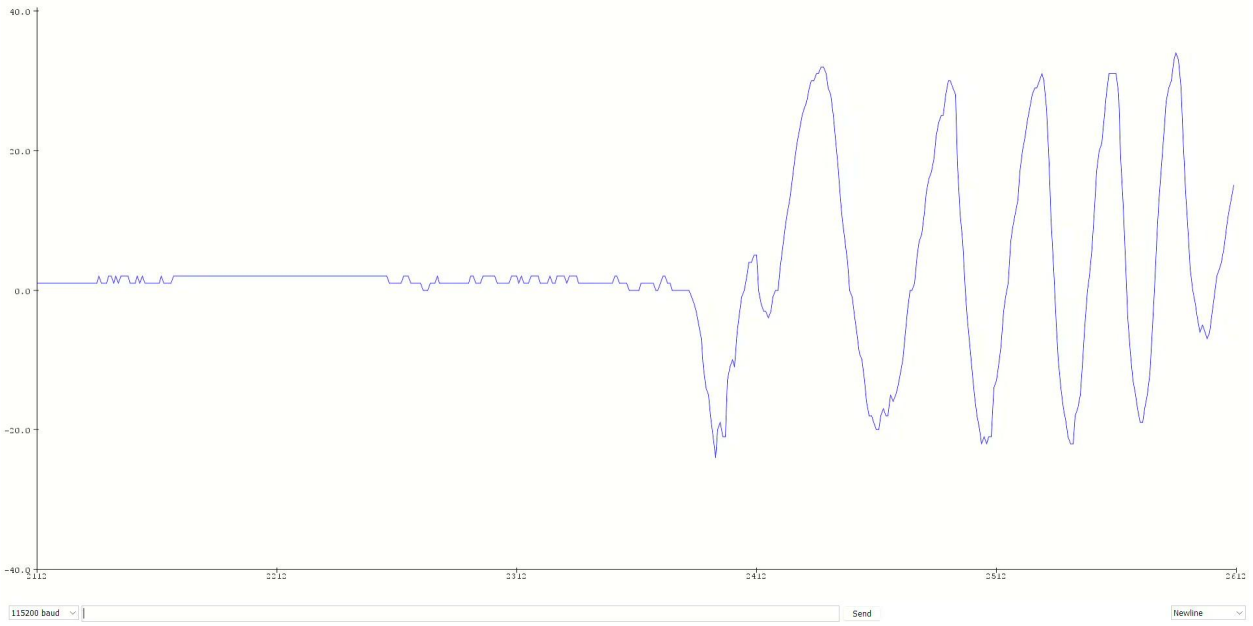


*Figure 16 : IMU pitch reading*

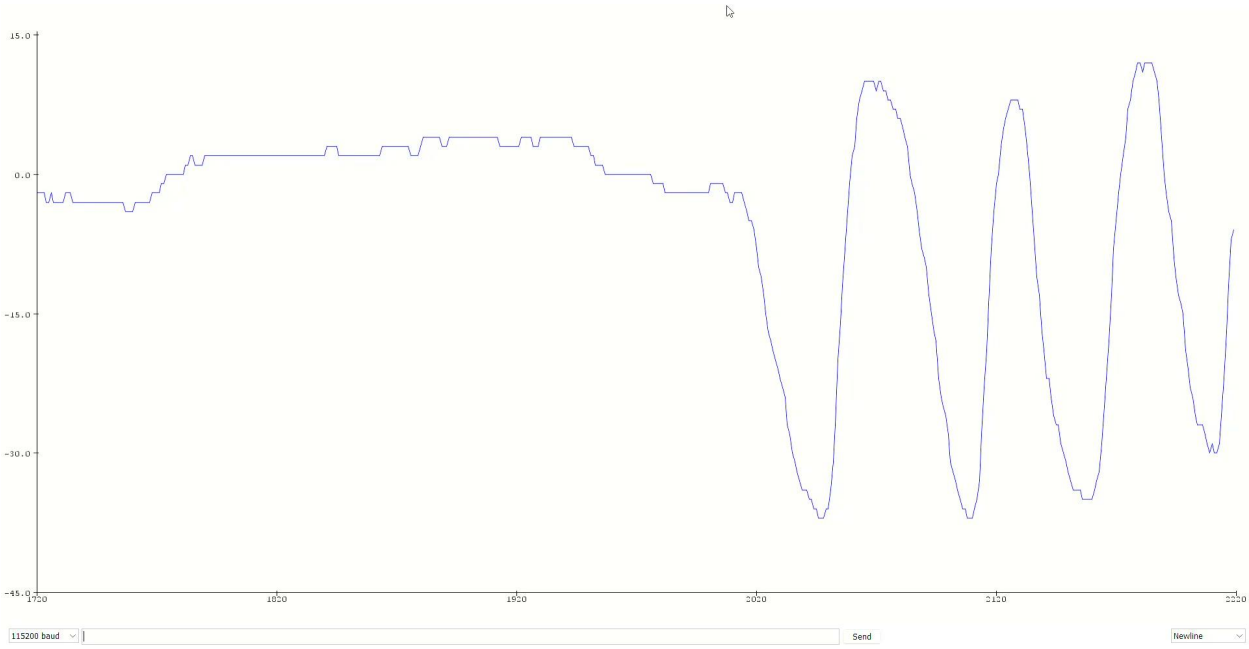


*Figure 17 : IMU roll reading*

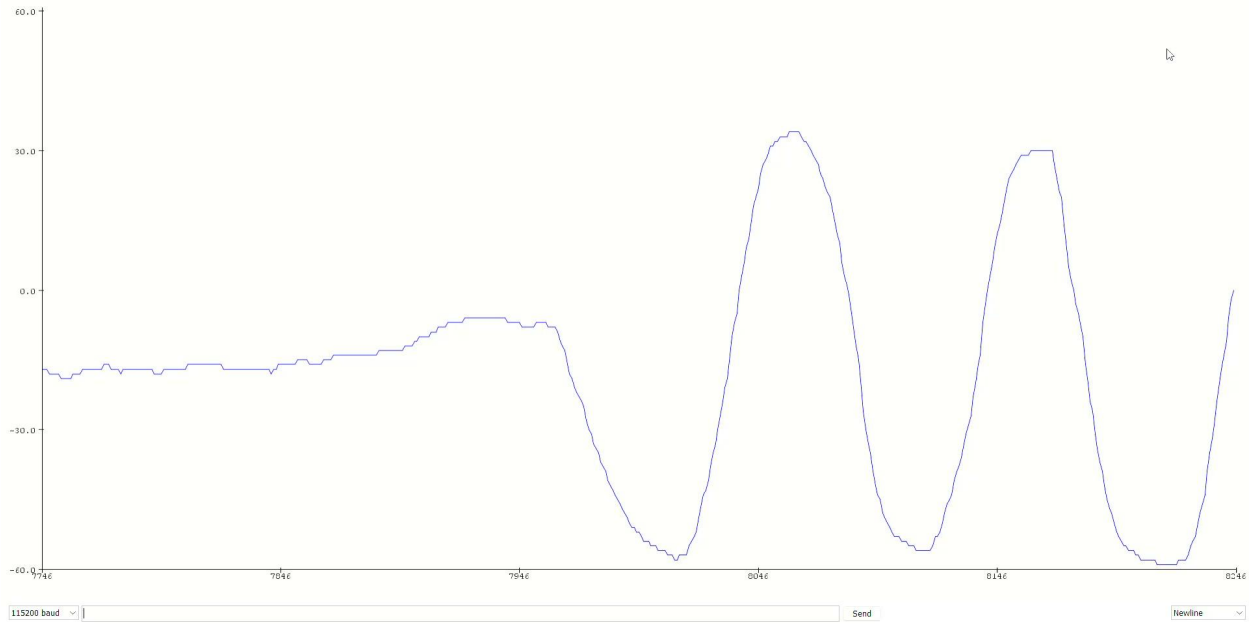


*Figure 18 : IMU yaw reading*

# 4. Costs

## 4.1. Parts

The cost of each part ordered for this project, not including the PCBs, which were provided to us by the ECE department, is presented in Table 1. The estimated labor costs for one electrical engineer and two software engineers working an average of 14 hours per week and making an average hourly wage is calculated and presented in Table 2.

*Table 1: Part Costs*

| Part | Manufacturer | Unit Cost ($) | Qty | Total Cost ($) |
|---|---|---|---|---|
| 4.7µF 50V Aluminum Electrolytic Capacitor | Nichicon | 0.48 | 2 | 0.96 |
| 220µF 25V Aluminum Electrolytic Capacitor | Nichicon | 0.57 | 2 | 1.14 |
| USB-A Connector | On Shore Technology Inc. | 0.50 | 2 | 1.00 |
| 6.8µH Fixed Inductor | Taiyo Yuden | 0.67 | 2 | 1.34 |
| DTA143Z Bipolar Transistor | onsemi | 0.31 | 3 | 0.93 |
| DTA143Z Bipolar Transistor | Rohm Semiconductor | 0.33 | 4 | 1.32 |
| Tactile Push Switch | CUI Devices | 0.21 | 2 | 0.42 |
| Dip Switch | APEM Inc. | 1.29 | 2 | 2.58 |
| 22 kOhm Thermistor | Kyocera AVX | 0.41 | 3 | 1.23 |
| MCP73871-3CC (Battery Manager) | Microchip Technology | 2.42 | 2 | 4.84 |
| LM117_TO3 (Linear Regulator) | Texas Instruments | 20.95 | 1 | 20.95 |
| TPS61030PWP (Boost Converter) | Texas Instruments | 3.75 | 2 | 7.50 |
| X000048 (8-bit MCU) | Arduino | 5.20 | 2 | 10.40 |
| MAX11607EUAT | Maxim Integrated | 5.83 | 2 | 11.66 |
| CY7C64225-28PVXC (USB to UART Bridge) | Cypress Semiconductor Corp | 3.99 | 2 | 7.98 |
| 16MHz Crystal Oscillator | ECS | 1.12 | 2 | 2.24 |
| Standard SMD LEDs | Harvatek Corporation | 0.08 | 25 | 2.10 |
| **TOTAL** | | | **60** | **$78.59** |

### 4.2. Labor

*Table 2: Labor Costs*

| Name | Ideal Engineer's Salary | Actual Hours Spent per Week | Number of Weeks | Total per Person |
|---|---|---|---|---|
| Rutu Brahmbhatt | $40 (Electrical Engineer) | 14 | 10 | $5600 |
| Saaniya Kapur | $45 (Software Engineer) | 14 | 10 | $6300 |
| Panav Munshi | $45 (Software Engineer) | 14 | 10 | $6300 |
| **TOTAL** | | **42** | | **$18,200** |

# 5. Conclusion

## 5.1 Accomplishments

In conclusion, we were able to accomplish quite a lot throughout this project. We were able to calibrate all the sensors, and collect reliable and accurate data from them. Furthermore, we were able to create an interface to monitor the data being collected from the iBand. This interface displayed the data in the form of graphs, allowing for better comprehension. Lastly, we were able to meet all the high-level requirements that were decided upon at the start of the development phase of the project.

## 5.2 Ethical Considerations

There were a few ethical considerations that were made during the design phase of this project. Firstly, we had to make sure that our iBand met the requirements laid out by HIPAA (Health Insurance Portability and Accountability Act of 1996) [13]. To abide by this, we made sure that all the data we collected from the biosensors were locally processed and not stored elsewhere without the patient's consent. Additionally, the data had to be encrypted in order to prevent side-channel attacks, such as MitM (Man In The Middle) [14], which could compromise the security of the patient's data. This was achieved by encrypting the packaged sensor data and decrypting it before processing it.

## 5.3 Future Work

There are a couple of things that can be worked upon in the future. Firstly, there were a couple of errors that were made while designing the PCB schematic. These consisted of the incorrect connections made for the I2C bus and the USB-to-UART IC. This could be worked upon by updating the schematic to purge the errors. Furthemore, the PCB, though downsized, can be made even smaller. This can be done by utilizing both sides of the PCB, using components with a smaller footprint, and by arranging the components compactly. Similarly, research can be done into what sensors will be ideal to include for the final revision of the iBand. This is because both the GSR and Pulse sensor had to be placed on the fingertips, rather than on the forearm as envisioned earlier. As the iBand aims to be a product shipped to medical professionals, it would be ideal to avoid keeping sensors on one's fingers. The hardware filters used to filter electrical signals from the muscles can also be replaced by software filters. This will result in a significant size reduction as the current sEMG enclosures were designed to include the large filter. Finally, the backend and frontend code can be worked upon to reduce the lag in displaying the real-time data received from the iBand.

## References


[1] B. Milosevic, S. Benatti and E. Farella, "Design challenges for wearable EMG applications," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, 2017, pp. 1432-1437, doi: 10.23919/DATE.2017.7927217.

[2] P. Sawers, "Amazon-backed wearables company Thalmic Labs kills its Myo armband, teases new product", *VentureBeat*, 2022. [Online]. Available: https://venturebeat.com/2018/10/12/amazon-backed-wearables-company-thalmic-labs-kills-its-myo-armband-teases-new-product/#:~:text=Share-,Amazon%2Dbacked%20wearables%20company%20Thalmic%20Labs%20kills,Myo%20armband%2C%20teases%20new%20product&text=Canadian%20wearables%20company%20Thalmic%20Labs,%2Dyet%2Dundisclosed%20new%20product. [Accessed: 04- May- 2022]

[3] B. Potu, V. Kumar, S. Annam and S. Sirasanagandla, "A morphometric study on flexor carpi radialis muscle of the forearm: A cadaveric study", *Morphologie*, vol. 100, no. 328, pp. 12-16, 2016.

[4] *Microchip.com*, 2022. [Online]. Available: https://www.microchip.com/en-us/product/ATmega328P. [Accessed: 04- May- 2022]

[5]"SHIELDEX® Metalized Conductive Fabric - V Technical Textiles", *V Technical Textiles*, 2022. [Online]. Available: https://www.vtechtextiles.com/conductive-fabric/. [Accessed: 04- May- 2022]

[6] “DATAGLOVES OVERVIEW,” *Yamaha VR gloves for musicians*. [Online]. Available: http://www.kobakant.at/DIY/?p=7114. [Accessed: 30-Mar-2022].

[7]"Composable Classic Bracelet with Number 18", *Nomination.com*, 2022. [Online]. Available: https://www.nomination.com/us_en/composable-classic-bracelet-with-number-18?utm_source=google&utm_medium=surfaces&utm_campaign=shopping_feed&utm_content=free_google_shopping&gclid=Cj0KCQiAgP6PBhDmARIsAPWMq6nSDaBObZhsx9zdTFdVwaBwwQKSSzdLGBmbG-zPsViFYyuF_hfVY6gaAgntEALw_wcB. [Accessed: 04- May- 2022]

[8]*Infineon.com*, 2022. [Online]. Available: https://www.infineon.com/dgdl/Infineon-CY7C64225_USB-to-UART_Bridge_Controller-DataSheet-v08_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ecca10346da&utm_source=cypress&utm_medium=referral&utm_campaign=202110_globe_en_all_integration-files. [Accessed: 04- May- 2022]

[9] *Ftdichip.com*, 2022. [Online]. Available: https://ftdichip.com/wp-content/uploads/2020/08/DS_FT232R.pdf. [Accessed: 04- May- 2022]

[10] "ArduinoThread - Arduino Reference", *Arduino.cc*, 2022. [Online]. Available: https://www.arduino.cc/reference/en/libraries/arduinothread/. [Accessed: 04- May- 2022]

[12] "ArduinoJson: Efficient JSON serialization for embedded C++", *ArduinoJson*, 2022. [Online]. Available: https://arduinojson.org/. [Accessed: 04- May- 2022]

[12] "Crypto - Arduino Reference", *Arduino.cc*, 2022. [Online]. Available: https://www.arduino.cc/reference/en/libraries/crypto/. [Accessed: 04- May- 2022]

[13] 2022. [Online]. Available: https://www.cdc.gov/phlp/publications/topic/hipaa.html. [Accessed: 04- May- 2022]

[14] "man-in-the-middle attack (MitM) - Glossary | CSRC", *Csrc.nist.gov*, 2022. [Online]. Available: https://csrc.nist.gov/glossary/term/man_in_the_middle_attack. [Accessed: 04- May- 2022]

# Appendix A

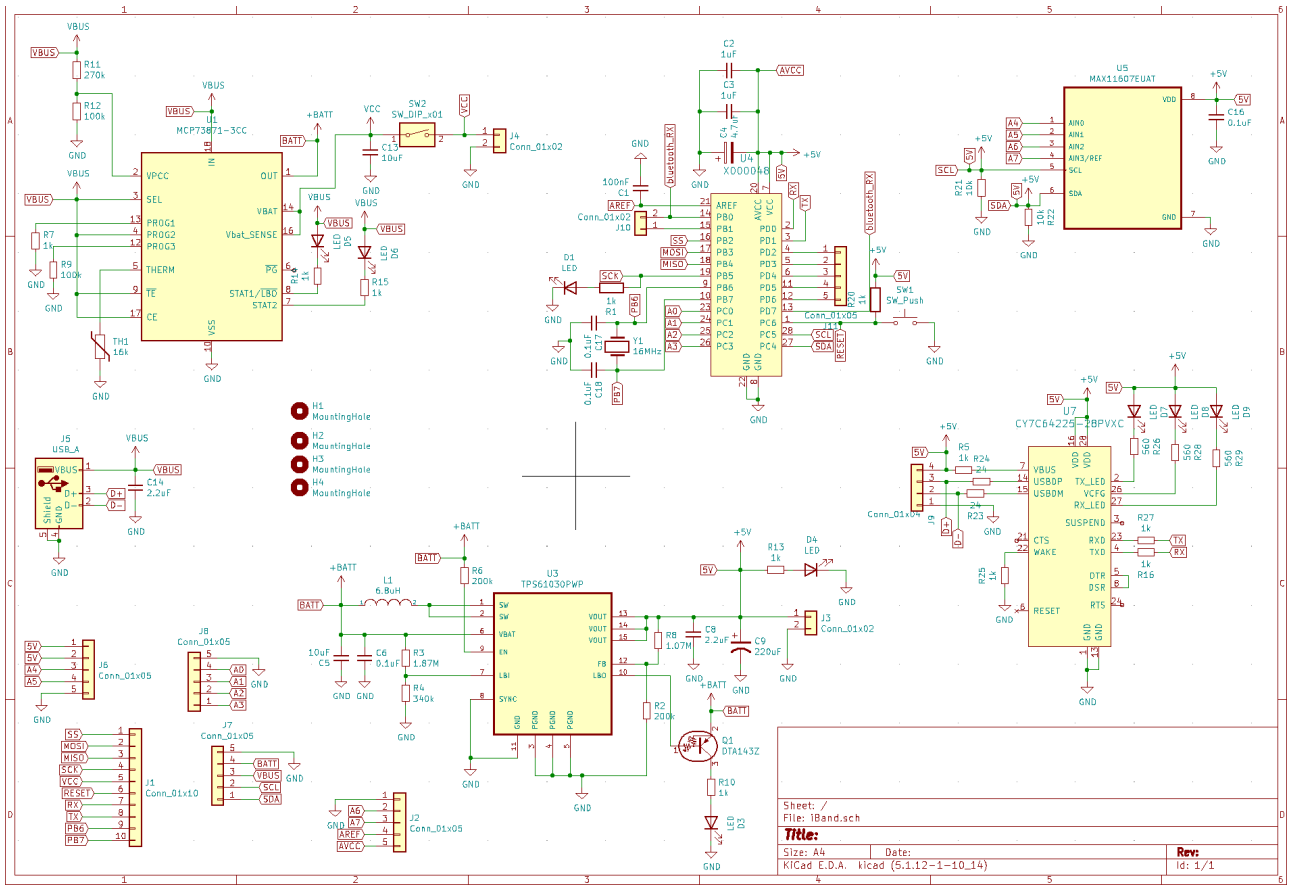


*Figure A.1: PCB Circuit Schematic*

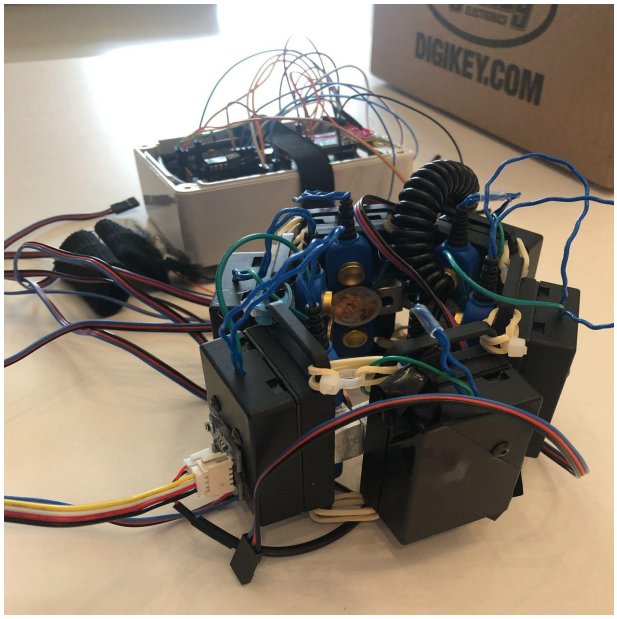


*Figure A.2: Armband Final Product*

*Table A.1 : Requirements and Verification Table for Power Delivery Subsystem*

| Requirements | Verification |
|---|---|
| At an input voltage of 5V, the PowerBoost will charge the lithium-ion battery at 3.7V±50mV for 850 mAh. | 1) Discharge the 3.7V lithium-ion battery.<br>2) Connect the microUSB to the Printed Circuit Board (PCB) to provide an input voltage of 5V and recharge the battery completely. Verify that the output voltage of the lithium-ion battery is between 4.2V and 4.5V. |
| The PowerBoost will provide an output voltage of 5.2V±50mV and output current ranging between 0A to 1A.<br><br>It will have an input voltage of 3.7V-4.5V from the lithium-ion battery. | 1) Connect the lithium-ion battery to the PowerBoost. Verify that the output voltage of the PowerBoost is 5.2V±50mV using an oscilloscope and voltage probe.<br>2) Connect the output of the PowerBoost to the microcontroller and use the oscilloscope and current probe to verify that the output current is 1A±50mA. |
| The power subsystem will include a rechargeable battery which supplies voltage for at least 3 hours. | 1) Discharge the battery to 3.0V±50mV. Then charge it completely until the voltage of the lithium-ion battery is between 4.2V and 4.5V.<br>2) Connect the battery to the PowerBoost for 3 hours to discharge it. Check that the voltage of the lithium-ion battery is between 4.2V±50mV to 4.5V±50mV. |

*Table A.2 : Requirements and Verification Table for Software Subsystem*

| Requirements | Verification |
|---|---|
| The serial communication between the microcontroller and Bluetooth Module must occur at a rate of at least 9600 bauds[3] (104.167 µs). | 1) Connect an oscilloscope to the $T_x$ pin of the microcontroller.<br>2) Set the oscilloscope to trigger on a pulse.<br>3) Take the reciprocal of the shortest pulse.<br>4) Verify that it is 104.167 µs. |
| The input voltage to the Bluetooth Module must be 5 V ± 1V[4] | 1) Connect one lead of the voltmeter to the input $T_x$ pin of the microcontroller |

| | |
|---|---|
| | 2) Connect the other lead of the voltmeter to the $R_x$ pin of the Bluetooth Module<br><br>3) Verify that there is an electrical potential difference of 0 V ± 1V |
| Bluetooth connection must be enabled and exposed to external devices[5] | 1) Prior to any external device connecting to the Bluetooth Module, the red LED should blink continuously.<br><br>2) Once connection to an external device is successful the red LED's blinking rate decreases. |

*Table A.3 : Requirements and Verification Table for Sensor Subsystem*

| **Requirements** | **Verification** |
|---|---|
| The sEMG sensor outputs a filtered sEMG signal between 0-1000 units[6]. | 1) Connect the sEMG sensor to an Arduino and place the electrode longitudinally over a team member's forearm.<br><br>2) Open the Serial Plotter in the Arduino IDE to view the signal.<br><br>3) Verify the signal by observing the max values of the peaks generated.<br><br>4) If the output is above 1500, or has noise (values might go up to 4 x 10e9), check the connection to the board and check the placement of the sensor over the arm. |
| The IMU sensor needs to correctly reflect changes in spatial position and the changes should be within ± 3% of the expected value[7]. | 1) Connect the IMU sensor to an arduino and place it on a flat surface.<br><br>2) Open the serial monitor in the Arduino IDE to view the signal.<br><br>3) Rotate it by 90 degrees in each axis (x, y, z) and check the changes in spatial angles.<br><br>4) Check if the changes in spatial position are within ± 3% of the expected value (here, 90 degrees). |
| The Pulse sensor's BPM (Beats Per Minute) output should be within ± 5% of the actual heart | 1) Connect the Pulse sensor to an arduino and place it over a team member's |

| | |
|---|---|
| rate of the user. | forearm.<br><br>2) Open the serial monitor in the Arduino IDE to view the signal.<br><br>3) Either manually count or use medical grade equipment to measure the user's heartbeat.<br><br>4) Check if the value measured by the sensor is within ± 5% of the value obtained from manual counting / medical equipment.<br><br>5) Repeat Steps 3-4 five times to verify the accuracy of the result. |