Final Report for ECE 445, Senior Design, Spring 2024

TA: Selva Subramaniam

01 May 2024

Project No. 41

By

Alyssa Huang

Anthony Leapo

Ramsey van der Meer

Smart Insole

**­­­**

**Abstract**

This project aims to provide a solution to enhance a hiker’s experience. We aim to use an insole, integrated with pressure sensors, attached to an external module equipped with an additional accelerometer and gyroscope to provide analytics on the hiker’s journey. The device is designed to monitor dynamic pressure distribution across the foot, the intensity of impact, the orientation of the foot when the user is hiking, and spatial foot movements. It is designed to not encumber users and be packaged in as small of a space as possible. The insole can be used with either Bluetooth to connect to a web interface for real time in-hike analysis, or with a micro-SD card for post hike feedback and generated reports. Key metrics displayed would be areas of high impact and walking patterns, which both are visualized using heat maps and graphs within the interface.

Key findings show that the device’s ability to provide real-time feedback on hiking techniques allows users to correct form to mitigate the risk of injury during hiking. The following report details the development, testing, and verification of the smart insole, outlining its potential to help hikers by emphasizing good form.

[1. Introduction 1](#_Toc165499227)

[1.1 Problem 1](#_Toc165499228)

[1.2 Solution 1](#_Toc165499229)

[1.3 High Level Requirements 1](#_Toc165499230)

[1.3.1 Accurate Pressure and Sensor Values: 1](#_Toc165499231)

[1.3.2 Accurate and Intuitive Data Integration: 1](#_Toc165499232)

[1.3.3 Wearable/Modular Physical Implementation: 2](#_Toc165499233)

[2 Design 3](#_Toc165499234)

[2.1 Design Process 3](#_Toc165499235)

[2.1.1 Thin Film Pressure Sensors 3](#_Toc165499236)

[2.1.2 Capacitive Pressure Sensing 3](#_Toc165499237)

[2.1.3 Pressure-Sensitive Conductive Sheet 3](#_Toc165499238)

[2.2 Visual Aid 4](#_Toc165499239)

[2.3 Block Diagram 5](#_Toc165499240)

[2.4 Subsystems 5](#_Toc165499241)

[2.4.1 Remote Interface Subsystem 5](#_Toc165499242)

[2.4.2 Data Transmission Subsystem 6](#_Toc165499243)

[2.4.3 Data Storage Subsystem 6](#_Toc165499244)

[2.4.5 Status Subsystem 7](#_Toc165499245)

[2.4.6 Power Subsystem 8](#_Toc165499246)

[2.4.7 Sensing Subsystem 9](#_Toc165499247)

[3. Design Verification 10](#_Toc165499248)

[3.1 Sensors 10](#_Toc165499249)

[3.1.1 Accelerometer and Gyroscope Readings 10](#_Toc165499250)

[3.1.2 Pressure Sensor Readings 10](#_Toc165499251)

[3.1.3 SD Card Readings 11](#_Toc165499252)

[3.2 Power 11](#_Toc165499253)

[4. Costs 13](#_Toc165499254)

[4.1 Parts 13](#_Toc165499255)

[4.2 Labor 13](#_Toc165499256)

[4.3 Schedule 13](#_Toc165499257)

[5. Conclusion 15](#_Toc165499258)

[5.1 Accomplishments 15](#_Toc165499259)

[5.2 Uncertainties 15](#_Toc165499260)

[5.3 Ethical Considerations 15](#_Toc165499261)

[5.4 Future Work 15](#_Toc165499262)

[References 17](#_Toc165499263)

[Appendix A 18](#_Toc165499264)

[Requirement and Verification Table 18](#_Toc165499265)

# 1. Introduction

## Problem

Many people enjoy hiking since it allows people of all fitness levels to experience the outdoors. However, oftentimes the constant repetitive pounding on hikers' feet can lead to soreness or even injury. Many factors contribute to the injury risk factor, including a hiker's gait, fitness level, the amount of weight carried, terrain, and much more. Currently, there are no products on the market that can deliver personalized feedback on foot stresses that are experienced over the duration of a hike. This information can be crucial in selecting appropriate footwear and improving walking techniques to prevent injuries. Additionally, this information could be repurposed to provide a metric to measure the difficulties of hikes, as trails that place a lot of pressure on your feet can be shared amongst avid hikers.

## Solution

Our project’s goal is to enhance the hiking experience, by addressing a common issue faced by many hikers—foot soreness and potential injuries due to repetitive impact or improper foot dynamics. Our smart insole system aims to monitor and analyze the distribution of pressure across the foot, the intensity of impacts, and the orientation and movements during a hike. This system provides crucial data that aids in showing users areas of high impact and extraneous foot movements, helping users to potentially visualize and correct hiking form so that they can mitigate the risk of injuries.

Our smart device integrates an insole covered with pressure sensors, an accelerometer, and a gyroscope into a comfortable, durable system that connects wirelessly to a web interface. This setup allows for real-time feedback and post-hike analysis through a user-friendly web application.

## 1.3 High Level Requirements

### 1.3.1 Accurate Pressure and Sensor Values:

* Integrate the pressure sensor to be able to track pressure changes in distinct regions of the foot, these regions having a maximum area of 3 inches squared.
* The sensors will need to track foot pressure distribution, impact intensity, and foot motion using the accelerometer, pressure sensor, and gyroscope. We expect sensors to be accurate within 10%.

### 1.3.2 Accurate and Intuitive Data Integration:

* Once we properly collect the data from the physical foot exerting pressure, we want to make sure we can properly store, transport, and intuitively represent the data on our web application. This will require proper use of the data we retrieve through sensible calculations and graphs.
* For storing and transporting, we will either store it within the microcontroller using an attached micro-SD card and send it in real time over Bluetooth to a separate device. We want real time transmission to not be significantly lossy, and only drop less than 5% of packets.
* For visualizing the data, we have chosen a heat map on our user interface to graphically show where the pressure distribution is. Other attributes regarding the overall hike, such as average pace and average orientation of the foot, can be extracted from the accelerometer and gyroscope data and displayed as normal graphs.

### 1.3.3 Wearable/Modular Physical Implementation:

* We want our device to be adaptable to various types of hiking boots, which is why we chose an insole that can be implemented very easily.
* We want our device to be wearable and enable our users to have 100% range of motion so that it does not deter from the hiking experience at all. Our objective of tracking analytics of the hike should augment and not bring down the hiking experience.

# 2 Design

In this section we will go over the design process and the final design details of our device.

## 2.1 Design Process

We started with the design of our insole. There were many options to consider when deciding the best pressure sensitive sensor to integrate into the insole. We had to consider cost, integrability, and viability. The following pressure sensing devices were considered:

### 2.1.1 Thin Film Pressure Sensors

Thin film pressure sensors act as variable resistors that lower resistance as pressure increases. I was able to find a small 19mm diameter sensor that would be suitable for integration in an insole. According to the resistor website [1], the variable resistor measures a range of weights from 20 grams to 6000 grams. Although this may seem like a small maximum, we can calculate the maximum pressure exerted over the size of the foot to see if the pressure distribution would allow for accurate sensing.

According to the diagrams from the paper “Real-time pressure mapping smart insole system based on a controllable vertical pore dielectric layer” [2], a human foot can exert up to 175 kPa of pressure. Given a diameter of 19mm, we can calculate the weight that would be felt by a single sensor placed under someone’s foot:

|  |  |
| --- | --- |
|  | (1) |

Thus, given the size of the pressure, a single sensor will have a maximum of around 5.057 kg of weight, which is under the 6 kg maximum that the sensor can handle.

However, the resistor cost is $6.90 per resistor, which is more expensive than we’re looking for.

### 2.1.2 Capacitive Pressure Sensing

Another option considered was using 3D printed conducive PLA and a dielectric to create a capacitive sensing device. This would provide flexibility to our design and would be a cheaper alternative, however, we decided that it would be more difficult and time-consuming to assemble compared to the next option.

### 2.1.3 Pressure-Sensitive Conductive Sheet

We considered Velostat, a pressure sensitive conductive sheet, as our last option. Velostat decreases resistance as the pressure increases, so it would allow us an easy way to measure pressure placed on an area. Velostat is easy to acquire and relatively inexpensive, while also provides us with flexibility when designing the insole. Using similar calculations as formula 1, we can choose a Velostat sheet size to cut out to implement in our insole.

We ended up settling with Velostat as the pressure sensitive sensor to use in our insole. Since Velostat acts as a variable resistor, we can design a simple voltage divider circuit and probe the voltage drop across each Velostat pad to find the pressure being placed on the Velostat pad. Below is a diagram of the circuit:

A diagram of a voltage diagram

Description automatically generated

Figure 1: Voltage Divider Used for Pressure Sensing Insole

## 2.2 Visual Aid

A hand holding a piece of electrical equipment

Description automatically generated

Figure 2: Visual aid for how the device would integrate into a shoe

Here we would insert the shown insole into a shoe in place of the existing insole. We would then attach the plastic enclosure to the shoe though a clip on the back allowing for our device to be fully mounted.

## 2.3 Block Diagram

A diagram of a system

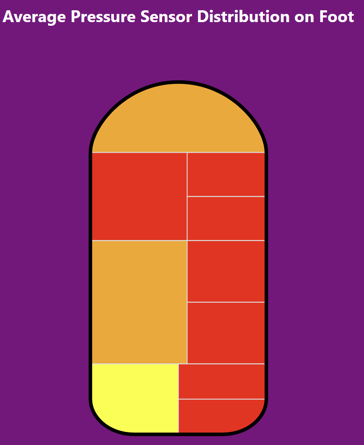
Description automatically generated

Figure 3 Smart Insole block diagram of different subsystems and their connections to each other

## 2.4 Subsystems

### 2.4.1 Remote Interface Subsystem

The remote interface subsystem was fully functional. It consisted of a web application built using the React framework and JavaScript. We chose to go with simplicity for this web application and give the user two clear choices of choosing a real time page or an upload page. The real time page consists of connecting to a device over a pop-up Bluetooth window, and once connected will update a heatmap of the foot and graphs of the elevation and the average pace of the user. These graphs are updated using the values from the pressure sensor and the readings from the accelerometer and gyroscope and are polled at roughly 64 hertz. The upload page allows users to upload a .txt file from the SD card reader from our device and will process that to give us a heatmap of the average pressure exerted on the foot and the elevation gain at various points throughout the hike. The heatmap is a shoe sectioned into ten areas and will light up in varying shades of red depending on the intensity of the pressure applied to it.

.

**Figure 4: Our heat map of the shoe displaying average pressure exerted post-hike.**

### 2.4.2 Data Transmission Subsystem

The data transmission subsystem was fully functional on our PCB. We were able to establish a Bluetooth Low Energy (BLE) connection between our microcontroller and our web interface. We chose to use the BLE protocol as it is designed to work with smart devices operating with minimal power and short range. This range limitation wasn’t an issue for us as users would be in the immediate vicinity of the device while in use. This ultimately allows us to use less energy than regular Bluetooth and allows our device to function for longer periods of time. Bluetooth BLE in code advertises itself out as a client and allows our web application to begin listening to its specific sensor characteristic ID, receiving those messages as a string to be used for further manipulation.

### 2.4.3 Data Storage Subsystem

The data storage subsystem was a crucial component which allowed us to store sensor data for ingestion later in our web interface. We couldn’t get this subsystem to function on our PCB as we made a mistake when connecting the pins according to the Serial Peripheral Interface (SPI) communication specification.

A screenshot of a computer

Description automatically generated

Figure 4: SD card connection legend with KiCad schematic for our PCB SD card connection.

As shown in the diagram above we connected our micro-SD card SMD module according to the SPI protocol of a regular SD card. We incorrectly assumed that both the SD card and micro-SD card would have the same pin mappings. If the flipped pins had all been data lines, it would have been easy to flip in software. However, the data in (DI) or master in slave out (MISO) pin according to the micro-SD protocol is supposed to map to pin 3 which we had connected to ground. This would be a huge obstacle as ground is connected to the rest of our components and we can’t just connect this to data. One solution we thought to implement would be to isolate the ground plane near the SD card and use that to connect to the GPIO on the ESP32. This never panned out and would have to be resolved in a future PCB order which would be easy to correct. Overall, the data storage subsystem on the PCB wasn’t functional.

On the bread boarded version of our smart insole, we were able to get micro-SD card writes to work as pins were clearly labeled on the dev board module we used. This allowed us to demonstrate that we would be able to write to the SD card in real time and later export this data to our web interface. We were able to write to the SD card at a rate of 100Hz on the breadboard, and were able to store all valuable pressure sensor, accelerometer, and gyroscope readings in a .txt file.

### 2.4.5 Status Subsystem

A hand holding a small plastic box with wires and buttons

Description automatically generated

Figure 5: Status subsystem showing user operated buttons and LEDs

The status subsystem was crucial to allow users to not only understand what the device is doing but be able to control it. We allowed users to control when they started/stopped a hike through an onboard button on the device. Additionally, we would be able to view when we were connected through Bluetooth to the web interface if the BT enabled LED was lit up. The other LED indicated if a hike was in progress and the sensors were recording or transmitting their data.

A screenshot of a computer program

Description automatically generated

Figure 6: Status subsystem schematic

As shown in the above schematic we have the capability to add another button into our device which would allow us to connect and disconnect Bluetooth. However, we were unable to get this functionality working in our Arduino code.

### 2.4.6 Power Subsystem

The power subsystem aimed to be able to supply constant power to the microcontroller through a lithium-ion battery. We choose a 2000 milliamp hours 3.7 volts lithium-ion battery as our power source for its small size and its ability to recharge. To convert this 3.7 volt battery into a usable 3.3 volts for our microcontroller we chose to use a low dropout (LDO). We chose to use the TLV758P500-mA SMD chip as it fits the needs of our project. It not only has a small package size but also allows for a variable input voltage.

Some design considerations we had to consider for the LDO were the low dropout voltage, meaning the minimum voltage at which the LDO would function properly and output the desired voltage. For us we needed to supply a constant 3.3 volts and according to the data sheet our chosen LDO would need 130 millivolts above our output voltage of 3.3 volts at 500 milliamps. This meant that our battery had a usable range of 3.7 volts to 3.43 volts. Overall, this would limit our battery’s ability to provide a sustained charge since it had a total range from 3.7 volts to 3.0 volts and would not be able to provide power once its voltage dipped below 3.43 volts.

A diagram of a circuit

Description automatically generated

Figure 7: Low Dropout Regulator circuit to output a consistent 3.3 volts for our microcontroller.

We decided to use the above circuit to provide power to our microcontroller. The 1M ohm and 200k ohm resistor combination was chosen according to the LDO’s datasheet so that the output would be as close to 3.3 volts as possible. We also used an LED to show if there was voltage supplied on the output of the LDO. This came in handy when debugging our circuit as it was a good indication there was an issue with our power subsystem when we noticed this LED wasn’t lit up. We also placed bypass capacitors in this design so that we could smooth the delivered voltage and hopefully supply more consistent power.

## 2.4.7 Sensing Subsystem

A screenshot of a computer

Description automatically generated

Figure 8: Sensing subsystem schematic

The sensing subsystem included the pressure sensor insole and the MPU6050, which encompasses both the gyroscope and the accelerometer. Since the insole is the second main large component other than the PCB and its enclosure, we made sure to design the insole so that it would be easily removable from the PCB, and conveniently attached to the PCB. From the schematic above, we see the second resistor half of the voltage dividers are off the board, to be connected through pin headers on the left side of the diagram shown above.

To the right of the diagram is the schematic we used for the MPU6050. While we connected the I2C ports, power, and ground accordingly, we were missing many connections on the rest of the pins, such as FSYNC and Clock. Not having these connections were detrimental to the success of our PCB, since we were unable to properly connect and communicate with the MPU 6050.

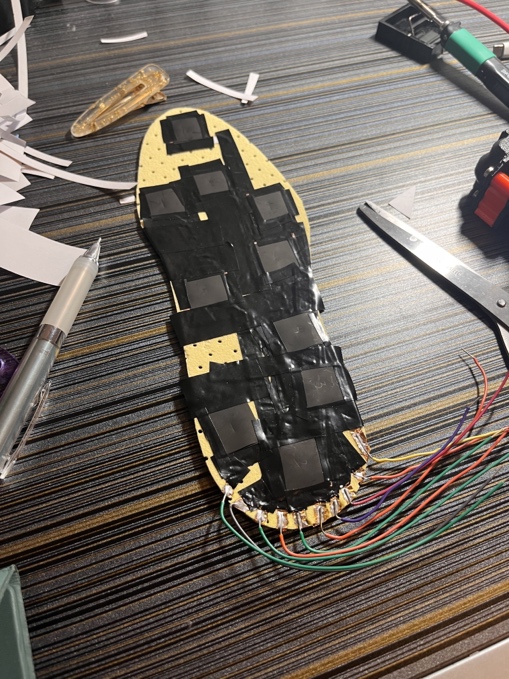


Figure 9: Prototype pressure sensing insole

Above is a prototype picture of our pressure sensing insole. Each of the wires coming out from the back of the insole plug into the pin headers on the PCB.

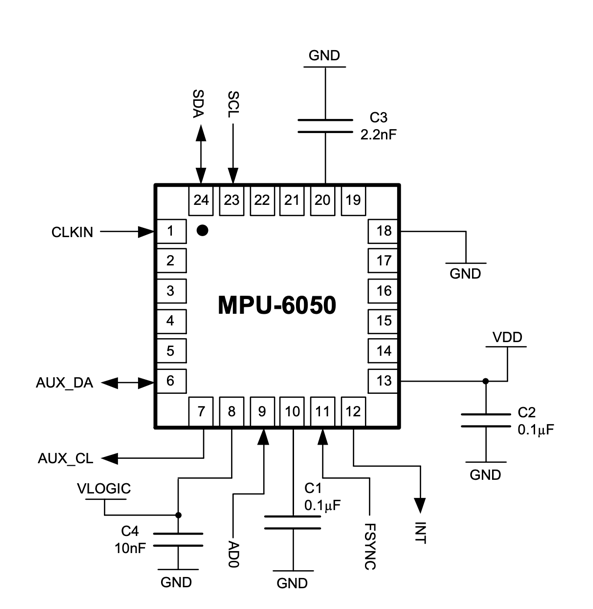


Figure 10: Sensing subsystem schematic [3]

Figure 10 shows a working circuit connection to the MPU6050 from the datasheet [3]. If we followed a similar design on our PCB, then it would’ve secured functionality of the MPU6050. However, we were only able to test MPU functionality through a complete breadboard circuit.

# 3. Design Verification

## 3.1 Sensors

### 3.1.1 Accelerometer and Gyroscope Readings

We wanted to make sure our sensor readings were accurate to within 10% of the ground truth so we measured the readings that it was giving us when kept static and discovered that they were relatively unstable but will eventually average out against each other. We additionally perform some post-processing on the post-hike analytics to smooth out some of the data. The post-processing consists of taking a windowed average of 15 sequential measurements. The data for the post-hike as a result is much smoother.

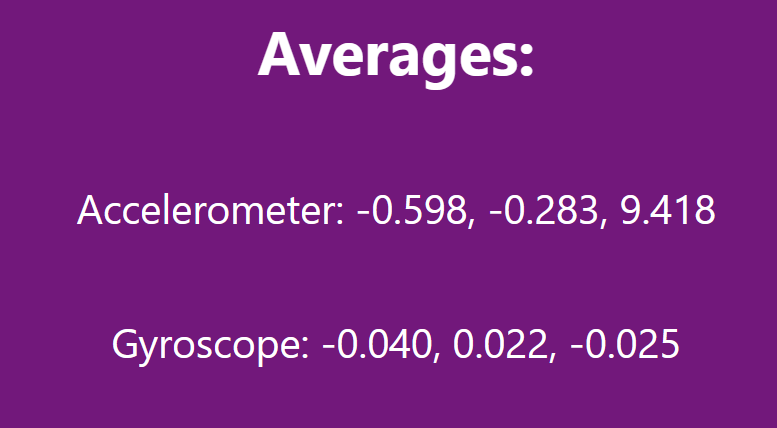


Figure 11: Average instantaneous accelerometer and gyroscope readings when kept flat on the ground.

### 3.1.2 Pressure Sensor Readings

We wanted to make the pressure distribution granular, which is why we sectioned the foot into 10 areas. We also wanted to make sure that the pressure sensor could distinguish between various levels of pressure intensity. For this we used four colors: black, yellow, orange, and red. We verified the pressure sensor readings by applying extremely different levels of pressure and making sure that the heatmap updated accordingly and that areas of the shoe that did not have pressure applied would indicate so.

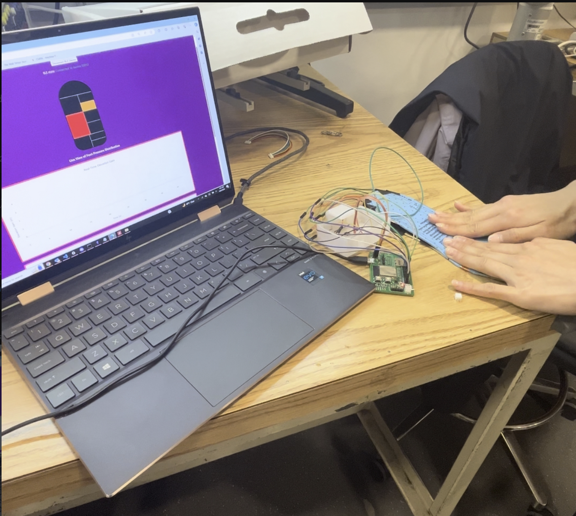


Figure 12: Our heat map updating in real time cross referenced with the pressure we are applying.

### 3.1.3 SD Card Readings

We verified the SD card was working by printing the accelerometer readings to a file at a specified rate and verifying that all the values were there. We additionally verified it was writing at the rate we wanted by printing to the serial monitor everything it did a write and how much time elapsed in between writes. After we were done writing to the SD card, we pulled it out of the connector, plugged it into the laptop, and verified that the readings were written to the file and could be viewed.



Figure 13: Output on serial monitor indicating how much time is elapsed between writes when set to 8Hz.

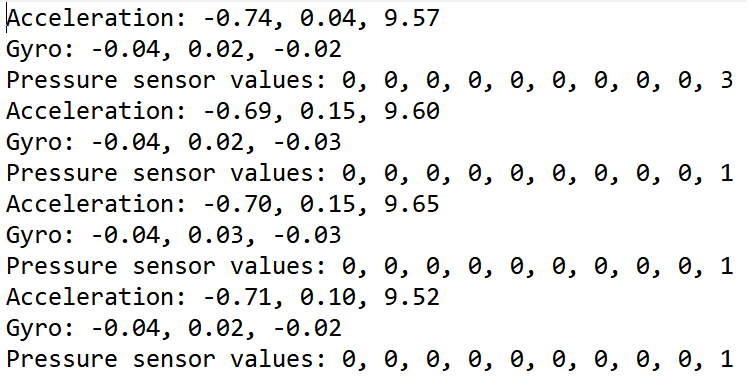


Figure 14: Sample output on the SD card file when the insole is sitting static.

## 3.2 Power

Once our PCB was assembled, we could test the functionality of the LDO and probe the output with our lab bench oscilloscope. The datasheet of the LDO claimed the output was accurate to 0.7% when within temperature specifications. We were able to test this with our lab set up and validated this claim.

For the LDO to operate properly and remain within a operable and comfortable temperature we are provided with an equation on the datasheet that calculates the max ambient temperature that the device can operate in.

|  |  |
| --- | --- |
|  | (2) |

According to this equation we can derive the max ambient temperature for the worst expected conditions where we are running on USB at 5v and there is 0.5A of current across the LDO.

|  |  |
| --- | --- |
|  | (3) |

Typical power draw is 0.4 amps, and we would operate normally off battery power of maximum 3.7v giving us the following max ambient temperature.

|  |  |
| --- | --- |
|  | (4) |

This Operating temperature will never occur even in the harshest of conditions and as such we can safely assume that the LDO will output at a desirable voltage and not burn users from high temperatures.

Looking into the time we can operate the device on battery power we can divide the battery capacity by the total current across the device. We probed our PCB and found it to draw around 0.4 amps of current when in full usage. However, this did not include the power drawn from the SD card reader and the IMU (Inertial Measurement Unit) as these components weren’t functional on our PCB. We instead used a shunt resistor to measure the current that these two components consumed in our circuit on our bread board. We found these measurements to be negligible with the IMU drawing 0.0017 amps and the SD card module consuming 0.000159 amps. As such assuming optimal conditions we have the following equation.

|  |  |
| --- | --- |
|  | (5) |

Overall, we were satisfied with the amount of time the battery would last allowing users to complete all but the most strenuous hikes. Looking at the reachability of the device we at first, we wanted to be able to charge our battery through our PCB. However, this would add a bunch more components on our PCB and as such we choose to fulfill the rechargeability requirement by using a removeable battery and charge it using a dedicated lithium-ion battery charging board.

Overall, we were able to achieve our requirements for the power subsystem as we were able to consistently and constantly deliver power, recharge our battery, and have our device powered for a whole hike.

# 4. Costs

## 4.1 Parts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 1 Parts Costs** | | | | |
| **Part** | **Manufacturer** | **Retail Cost ($)** | **Bulk Purchase Cost ($)** | **Actual Cost ($)** |
| BJT NPN Transistor | Comchip | 0.27 | 0.27 | 2.16 |
| Battery 3.7v 2000mAh | Pkcell | 12.50 | 12.50 | 12.50 |
| JST Connector | PH Connector | 0.17 | 0.17 | 0.68 |
| LDO | Texas Instruments | 1.01 | 1.01 | 2.02 |
| 5v 2.4A Lipo Charger | Seeed | 4.75 | 4.75 | 4.75 |
| TVS Diode Array | Littefuse | 0.78 | 0.78 | 3.12 |
| Sealed Tactile Switch | Omron | 0.65 | 0.65 | 7.76 |
| SD Card Adapter SMD | GCT | 1.09 | 1.09 | 4.36 |
| MPU-6050 IMU | InvenSense | 7.85 | 7.85 | 15.70 |
| 200k Ohm 805 SMD | KOA | 0.57 | 0.43 | 8.64 |
| 47 Ohm 805 SMD | Wurth | 0.14 | 0.08 | 3.95 |
| microSD Card Adapter | Sandisk | 10.00 | 10.00 | 10.00 |
| Insoles | Sintege | 2.66 | 2.66 | 8.00 |
| Copper Tape | Vaincre | 6.99 | 6.99 | 6.99 |
| Pressure Sensitive Velostat | Chibitronics | 10.00 | 10.00 | 10.00 |
| **Total** |  |  |  | **100.63** |

## 4.2 Labor

Assuming an average wage of $50/hour for ECE graduates from University of Illinois at Urbana Champaign, labor costs per person would amount up to $50/hour \* 2.5 (overhead factor) \* 125 hours (estimate number based on work we have done throughout the semester) = $15,625 per person. This totals to $46,875 worth of total labor costs throughout the semester for all group members.

## 4.3 Schedule

**Table 2 Work Schedule**

|  |  |  |
| --- | --- | --- |
| Week | Task | Team Member |
| 1/15 | Initial Web Board Post | Everyone |
| 1/22 | Laboratory safety training  CAD assignment | Everyone |
| 1/29 | Project approval | Everyone |
| 2/5 | Proposals  Team Contract | Everyone |
| 2/12 | Design Document Start | Everyone |
| 2/19 | Design Document Due | Everyone |
| 2/26 | PDB Design | Everyone |
| 3/4 | PDB Design | Alyssa + Ramsey |
| 3/11 | Spring break |  |
| 3/18 | 2nd PCB order due + control subsystem | Alyssa + Ramsey |
| 3/25 | Created the web application and got a heatmap uploaded to it | Tony |
| 4/1 | Got the IMU working on the dev board | Tony |
| 4/8 | Assemble and test PCB from 2nd round order  Finish making changed to PCB design for the 5th and final round order  Try and read from micro-SD card on dev board | Alyssa + Ramsey  Alyssa + Ramsey  Tony + Ramsey |
| 4/15 | Prototype the CAD the enclosure for our PCB  Work on getting 5th round PCB working  Integrate software with the PCB  Mock Demo | Ramsey  Ramsey + Alyssa  Ramsey + Tony  Everyone |
| 4/22 | Finish up the enclosure for our PCB  Work on trying to get IMU working on the PCB  Work on trying to get the SD card working on the PCB  Work on getting the button presses to register on our PCB  Final Demo | Ramsey  Alyssa  Ramsey + Tony  Tony  Everyone |
| 4/29 | Final Presentation | Everyone |

# 5. Conclusion

## 5.1 Accomplishments

Through the semester, we were able to successfully design and implement a pressure sensitive insole that can detect pressure differences across various pressure zones under the foot. This insole also could properly connect and communicate information with our custom PCB. Meanwhile, the PCB was mostly functional, as it could be programmed through micro-USB, successfully run a program on the microcontroller, connect to Bluetooth to a peripheral device, and send data to the peripheral device.

Aside from our device, we were able to get full functionality of all our desired requirements on a breadboard, which in addition to the functions mentioned above, could also properly write sensor data to an SD card, and log accelerometer and gyroscope data.

## 5.2 Uncertainties

Overall, our device was able to function in a satisfactory manner. However, it came up short in regard to the PCB design of the SD card and IMU. These integrations weren’t connected right as such were unable to communicate with our microcontroller. An area we were unable to achieve satisfactory performance was getting quantitative results from our pressure sensors. We were unable to correlate the sensor readings to physical pressures and as such displayed relative pressure to the user. An area of the foot which had more pressure would light up in a redder color. This was not specified in our design requirements but may have been some good data to provide to users. We believe that the pressure uncertainty is due to the given uncertainties found in resistor values, and most especially the differing resistance of our handmade insole. In order to provide this data, we may have to invest in calibrated sensors.

## 5.3 Ethical Considerations

Throughout our design process, we considered many ethical considerations that a wearable device should. For example, we ensured that the user is in full control of when the device begins and ends monitoring their data. We also implemented lights to display the status of the device to the user, so the user is always cognizant of the current state of the device, such as if it’s recording or if it’s Bluetooth connected to another device. We wanted to be sure we complied with the IEEE standards on data privacy, specifically those on personal data[[1]](#footnote-2). We also wanted to be compliant with the IEEE standards on wearable electronics and their security objectives, specifically the objectives regarding data processing terminal application software and wireless communications[[2]](#footnote-3)

## 5.4 Future Work

In the future, we would look forward to one more PCB iteration, which would bring us closer to the final product. The PCB needs minor changes to the SD card read as some pins were flipped in our created schematic. Additionally, we lacked some of the required connections on our IMU and as such would need to make this change in a future iteration of our PCB. An easy addition giving the state of our current PCB would be to add a way to turn on and off Bluetooth for the ESP32 as this would save a substantial amount of battery (around 25% or 130mA)[[3]](#footnote-4). Additionally, a way to recharge the devices battery though foot motion was brought up in the final presentation. This would be a great avenue to look into further down the line when we get everything functioning correctly and without issues.

# References

[1] Tao, J., Dong, M., Li, L. et al. Real-time pressure mapping smart insole system based on a controllable vertical pore dielectric layer. Microsyst Nanoeng 6, 62 (2020). <https://doi.org/10.1038/s41378-020-0171-1>

[2] “RP-C18.3-Lt Thin Film Pressure Sensor.” *DFRobot*, [www.dfrobot.com/product-1846.html](http://www.dfrobot.com/product-1846.html).

[3] *MPU-6000 and MPU-6050 Product Specification Revision ...*, [invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf](http://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf).

[4] “IEEE Privacy Policy.” *IEEE*, [www.ieee.org/security-privacy.html](http://www.ieee.org/security-privacy.html).

[5] "IEEE Standard for Wearable Consumer Electronic Devices--Overview and Architecture," in IEEE Std 360-2022 , vol., no., pp.1-35, 25 April 2022

[6] mywang espressif, “Ble taking too much power (IDFGH-769) · issue #947 · espressif/ESP-IDF,” GitHub, https://github.com/espressif/esp-idf/issues/947#issuecomment-500312453 (accessed May 1, 2024).

# Appendix A

# Requirement and Verification Table

**Table 3 Requirements and Verification Table**

|  |  |  |
| --- | --- | --- |
| Requirements | Verification | Verification status |
| Consistent power delivery | Measure voltage delivered to microcontroller and validated that it is consistent at 3.3v with a margin of error of 5%. This can be done with a voltmeter. | Y |
| Rechargeability | Use the device until battery isn’t full. Then measure battery voltage. Charge up the battery until battery charger indicates full. Finally measure the voltage of the battery and make sure it is fully charged according to the data sheet at 3.7v.  Show that we can run the device on the recharged battery | Y |
| Last all hike | Monitor the current draw of the system while running and calculate the total time it can run. Max battery capacity mAh / current of system mA à total time system can operate | Y |
| Users can tell status of device | Make sure LEDs light up at the right times to indicate status of device. This includes the Bluetooth and Start Hike LED | Y |
| Start and end hike button | Make sure data isn’t being recorded by probing SD card connection and shouldn’t be any data written. Then press start hike button. After which make sure data is being recorded by making sure a new file is being created and written to. Press end hike. Finally make sure data isn’t being recorded and a hike file has been written to SD card and completed with sensor data included. | Y |
| Bluetooth button | Make sure we can press this button and connect from our web interface to our device.  *We are constantly broadcasting and allows users to connect to the device automatically.* | N |
| Accelerometer can track velocity and acceleration | Move accelerometer around and make sure that the movement is read by checking accelerometer output in the web interface. Accurate to within 5%, can verify with gravity measurements. | Y |
| Pressure sensors can track pressure in relation to each other accurately. | Use set of 3 different weights. Place weights on all permutations of pressure pads on the insole. Make sure that in the interface each pad is calibrated right so that the weights in relation to each other are right. Only want them to be able to measure pressure accurately in relation to other sensors as sensors work when bent. Since some are already bent a bit to be placed on an insole just need to calibrate them accordingly. | Y |
| Gyroscope can track direction | Change the orientation of the device. Make sure that this is picked up by the microcontroller. Accurate to within 5%, we can verify by rotating the shoe 90 degrees and verifying that the gyroscope registers this exactly. | Y |
| Processing delay can keep up with sensor read rate | Validate that all sensor data is being written to SD card and none is lost due to overflow of write buffer due to reading too fast. To verify we can record for 60 seconds specified in a test software program and check we have 60 \* 8 samples for each sensor | Y |
| The enclosure that the sensing subsystem will be housed in can withstand the hiking experience/slightly wet and non-ideal weather | Validate that the sensors do not get damaged, and the sensor box can withstand small samples of water being poured on it.  *We could waterproof our device a coating. However we didn’t water proof our device since this would not allow access and the ability to modify our PCB.* | N |
| Data can be transmitted to another device. | Validate that when sensors are stimulated the corresponding data shows up in the web interface | Y |
| Information not lost when transferring from sensor -> microcontroller -> Bluetooth | Validate that all data points are collected and sent by applying pressure to various different parts of the insole rapidly and then cross referencing the received output with your known input. | Y |
| Receive data transmitted from data transmission subsystem with low latency | View data acquired through Bluetooth connection on the web interface. Data acquired for real-time must be displayed within 4 seconds. | Y |
| Display data to users | Show users acquired data in heat map, elevation gain and average pace. For these charts, the x dimension is time, and the y dimension are the listed attributes. Data must be accurate. We can test this by applying pressure to one side of the shoe and then monitoring its corresponding display on our interface. If the data is not accurate, we will explore data post processing options to reduce noise, such as applying filters or taking a weighted average of numerous measurements. | Y |

A red circuit board with blue lines

Description automatically generated

Figure 13: Final PCB Layout

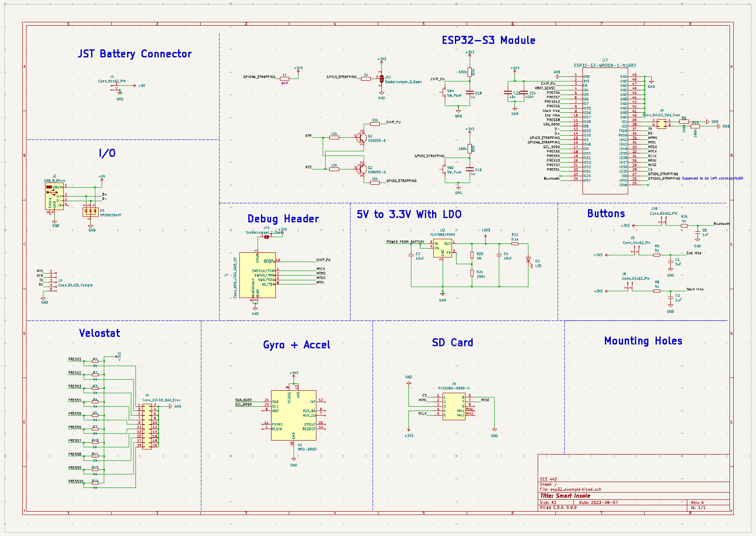


Figure : Final PCB Schematic

1. [2] www.ieee.org/security-privacy.html [↑](#footnote-ref-2)
2. [3] https://standards.ieee.org/ieee/360/6244/ [↑](#footnote-ref-3)
3. [4] https://github.com/espressif/esp-idf/issues/947 [↑](#footnote-ref-4)