Switch Better Have My Money | Smart Shoe-Insole

Eric Csehoski, Kieran Humphreys, Ahmad Tamanna, Merron Tecleab, Xinyuan Zhu

December 17, 2021

Capstone Design ECE 4440 / ECE4991

Signatures

Eric Csehoski – digitally signed 12/17/21

Kieran Humphreys – digitally signed 12/17/21

Ahmad S Tamanna – digitally signed 12/17/21

Merron Tecleab - digitally signed 12/17/21

Xinyuan Zhu – digitally signed 12/17/21

Statement of work:

Kieran

My primary role for the project was to design the firmware contained on our microcontroller. At the beginning of the project, this required researching the right microcontroller and launchpad for our application. After getting our hands on the launchpad, I then went to work on the analog to digital conversion for our sensor values. In order to display the values stored in memory, I configured the backchannel UART communication so I could send the values to my computer and read them. I also had to calculate the total time it took to sample the channels, using an equation found in the datasheet. After the analog to digital conversion was understood, I moved on to configuring the Bluetooth communication.

After deciding on what Bluetooth module we were going to use in our final project, I configured the UART communication. To confirm it worked, I used a logic analyzer to read what the Bluetooth module responded with when sent "AT" commands. After this, I configured the Bluetooth connection with a separate computer, and figured out the actual format of data transmission. From here, I helped conduct system tests with a full data flow. I also assisted Xin Yuan in the design of the LabView program. Some other tasks I did for the group were spray paint the box that housed our hardware, research force sensors, determine power requirements and sampling frequency of the system, and decide where the sensors were to be placed on the sole.

Merron

During this Capstone project, my primary role was designing the overall look of the project. One of the main things was designing the 3D model case to put our hardware parts in it. Designing the case had to be done once we established all our hardware components that we would be using for our project. This case would be attached to the leg connected to the sole that I bought and had a part in designing where the sensors would be placed for usability purposes.

Xinyuan

I created the data-viewing software application for this project. Within this application, I implemented the Bluetooth receiver, data checking, CSV data storage, data visualization, and unit conversions. This was the front-piece of the project that people would be mainly viewing when demoing and testing our project. My secondary responsibility was helping with Bluetooth communication. I helped pick the specific module that we would use and configured it. Once we found that we had to use a 9600 baud rate for the current module, I performed extensive testing regarding sending and reading frequencies to find one that could minimize the transmission delay.

Ahmad

My primary task to contribute to this project was to design a PCB for analog signal processing, MSP430 connections, FSR 402 sensor connections, battery rechargeability, and

regulating the power for the three elements of the project, i.e., analog signal processing hardware, MSP430 launchpad, and HC06 Bluetooth module. For the PCB design, I put together the schematic and synthesized the board layout for manufacturing for the board send-outs as well as the revisions. At the meantime, I compiled the bill of materials (BOM) for the first part order set and assisted the in compiling the BOM for the second part order set in addition to assisting teammates to extensively research the best sensor for the project. After receiving the components for PCB and the board itself, I created the relevant paperwork needed for WWW electronics and delivered them for soldering. Moreover, I was also involved in the hardware testing and debugging process that led to the revision of the board schematics.

In addition to working on my primary duties, I was honored to assist the teammates in making the Bluetooth module booster pack, 3D print job, and brainstorming the solutions to optimize the project on the software side.

Eric

My role in this project was on the hardware side and was primarily related to the sensors. This involved researching pressure sensors and selecting the best one for our application. Additionally, I was responsible for designing the circuitry that connected our sensors to the analog input pins of the microcontroller. I was also responsible for the physical assembly of the shoe sole, which involved customizing a ribbon cable to match our sensor placement, soldering the sensors to the cable, then gluing and sewing the sensors and their wires to the sole. When we switched Bluetooth modules, I helped in the construction of the Bluetooth module booster pack. I also helped to design and select parts for the power circuitry, specifically on the circuitry that prevents the battery from over-discharging. We created two PCBs, as our first needed revision, and I was involved in the testing of both. I also maintained a spreadsheet of the total cost of our project.

Table of Contents

Table of Figures

List of Tables

Abstract

The project created by Switch Better Have My Money is a smart shoe insole. This insole has seven pressure sensors strategically placed throughout areas of the sole that experience the most pressure during physical activity. The data collected from these sensors then get transmitted using Bluetooth to an external software application, which will visually display the data in real time. The goal of this project will be to enable this smart insole to be used as a medical tool for use in physical therapy and diagnostics, as well as a tool to be used by athletes to evaluate and improve their performance.

Background

Foot pressure distribution (FPD) is a useful metric for the diagnosis of potential issues in a person's foot or gait. FPD can be used to identify foot deformities, diagnose gait disorders, and provide strategies for preventing foot ulcers in diabetes [1]. Outside of medical usage, the technology can be used by athletes and coaches; for example, golf players can use this data to assess the stability and the form of their swings [2]. This project was chosen because each member in the capstone team has either actively played a sport in the past or currently are engaged in sports. We wanted to provide a tool to benefit those who play sports. Upon doing further research about the benefits of knowing FPD, the scope of the project expanded to be both a sports improvement tool and a medical diagnostic tool.

Historically, the most common form of FPD analysis tools is the use of a kiosk, where users step on a stationary platform to take pressure measurements and use weight and height information to calculate biomechanical data. This is used by sole manufacturers such as Dr. Scholl's for selecting the best insole for a user given their needs [3]. In recent years, more portable methods of measuring FPD have emerged, one being a robotic shoe to measure ground reaction forces and the foot center of pressure. Other commercially available "smart insole" devices have also been developed to focus on balance and evaluate the gait of patients with neurological diseases.

There are several drawbacks in these pre-existing devices that we aim to improve upon to bring novelty to this capstone project. The first change is portability. The kiosks provide greater accuracy, at a cost of portability. This makes such a system infeasible to use outside of a laboratory setting. The capstone project will offer a smaller, condensed system to provide similar data. The second change is pricing. Existing systems were developed using proprietary hardware and this is reflected in the cost to consumers. By making the PCB the only piece of custom hardware developed, the total cost of the system is lower priced.

The largest novelty that we want to bring to the system is customizability. Everyone has different feet and therefore, manufacturing a one-size-fits-all solution is impossible. Because the sensors can be easily detached and reattached to an insole, users can further cater this system towards their needs by swapping out to their preferred shoe in-sole or adjusting the placement of sensors to specific areas that they need to focus more measurements on. Furthermore, an additional change not seen in other systems is making the software open source. This is a

consumer-driven decision that will allow other developers to create addons and features in order to expand the goal of customizability of the system.

Lastly, the team is firm believers in the right-to-repair [4]. This is a core flaw of other systems that have been studied prior to the development of this project. Because of the use of proprietary hardware and how the sensors are interfaced, having one sensor fail would result in the entire system having to be discarded and a new one purchased. With the modular design of this system, a faulty sensor can easily be replaced. This provides a longer product-life cycle and helps reduce the environmental impact.

This project is an interdisciplinary venture that combines previous coursework done in computer engineering, electrical engineering, and computer science.

The electrical engineering component of this project is the fabrication and testing of the PCB board, which involves interfacing the sensor array, creating the charging IC, and handing power delivery to all the components. For this, knowledge from the FUN series in various circuit schematics and creation of a PCB using Ultiboard was used. Additionally, the coursework in Electrical Energy Conversion (ECE 3250) was needed for the knowledge it provided in providing power to the entire system. Finally, Analog Integrated Circuits (ECE 4660) was used in order to provide a more elegant design to the hardware.

The computer engineering component is programming the embedded system, which involves configuration the analog-to-digital conversion of the circuit and transmitting data wirelessly through Bluetooth. The classes used in this were Introduction to Embedded Systems (ECE 3330) and the Embedded Computing series (ECE 3501/3502) in order to provide properly formatted code and knowledge of the microcontroller primitives. Additionally, Kieran had previous experience outside of the classroom at the High Performance-Low Power Laboratory to work with embedded systems.

The computer science component of this project is creating the software application that will be used to display the data that has been collected. For this, the experience where knowledge was derived from was previous laboratory experience as a LABVIEW developer. Xinyuan also had additional software development experience working as a C++ developer at the High Performance-Low Power lab.

Constraints

Design Constraints

Before planning anything in the project, there were restrictions imposed by the rules of the Capstone project. This involved the requirement to create a custom PCB board and the restriction on the CPU unit being that no hobby boards were allowed. Design constraints after the project were chosen were mainly influenced by team experience and the economic constraint.

The most apparent constraint faced was the total number of sensors that would be embedded in the sole. This number was strictly dictated by the number of Analog to Digital input pins that were available on the microcontroller chosen, the MSP430FR5969. This limited the total number of sensors available to be used to seven total. There was also an additional CPU constraint in that the one that was chosen either needed to have Bluetooth functionality built in or be compatible with a Bluetooth booster pack in order to make wireless communication possible.

The software that was available was largely limited to either software that had licenses provided from the University of Virginia, or free software that was available. This ultimately led to using KiCad to create the initial schematics, which were then redone in Multisim to be imported to Ultiboard in order to create the PCB. Code Composer Studio is a free software provided for programming the microcontroller so that was used. For the software application, the Eclipse Java IDE was used.

The manufacturing limitation of the project mainly stemmed from parts availability. Various components that the group originally planned to use, such as batteries or sensors were out of stock on Digikey, leading to a different component being chosen in the interest of time. Aside from this, there were several restrictions placed by the PCB manufacturer that our design had to fit.

The PCB manufacturer, Advanced Circuits laid out some constraints for manufacturing the board. A thickness of 62 mils and 9 mil core was required for 2-layer boards as well as minimum line per space of 5 mil and minimum drill hole size of 10 mils. As the PCB was intended to play the role of a booster pack for the MSP430, the Texas Instruments recommends having a width of 2000 mil for a standard module, but we were constraint with the size of Combicon connectors with 14 pins to fit cleanly at one side of board. Eventually, the board width was expanded to 2400 mil, so the extended section is barely noticeable and negligible for the project.

Economic and Cost Constraints

Cost constraints were not a significant factor in deciding the final design because the raw parts cost approximately $160 total, which was significantly under the budget limits. The sensors were the most expensive part of the project, with the microcontroller at the second most expensive. Therefore, a more sophisticated system could have been made with the budget. These changes could have been better sensors or a different microcontroller that had more IO slots. However, such a change would make PCB manufacturing significantly more complicated, so these changes were not pursued in the interest of time.

Environmental Impact

Since this project uses many electronic components and all of them have plastic, the environmental impact of this project would come from the manufacturing of the parts [5]. The use of a Lithium-ion battery in the project also introduces toxic chemicals to the environment, which will ultimately need to be discarded in the future due to a limited lifespan [6]. However, long-term use of the project will see a less negative environmental impact.

Sustainability

Since this system was designed to allow the easy interface of parts, long-term use of this system results in better sustainability. The modularity of the project allows easy repairs in the event a component malfunctions. This results in less landfill waste while also promoting product longevity. However, some parts such as the battery that will inevitably need to be replaced will have negative environmental consequences.

Health and Safety

The most significant safety risk that is present in the project is the Lithium-Ion battery. If this battery gets over-charged or punctured in any way, it will result in swelling and potentially cause a fire [7]. Therefore, extra precaution was exhibited when the charging IC was designed in order to prevent overcharging by shutting off the power. Since this battery is going to be used in a wearable device, another safety consideration made was to enclose the PCB, Microcontroller, and the battery into a housing for safety reasons. Additional operating conditions had to be put in place for the temperature to ensure the safety of the project.

Another safety risk was that sweat coming from wearing the project could result in a short circuit from the electronics. To address this, the microcontroller, PCB board, and battery were moved outside of the shoe and the sensors were placed under the in-sole rather than above it in order to provide more protection.

A final minor safety concern was to ensure that using the actual project itself was comfortable. Extra caution was made to ensure that the housing did not constantly rub against someone's skin, which could lead to sores after prolonged use.

External Standards

Barr Standards

The Barr coding standard [8] was used in the development of the embedded code for the MSP430. This convention gives a set of rules for code that minimizes bugs in firmware while letting the code be maintainable for future updates.

IPC Standards

IPC [9] standards were used as a basis for verification for the PCB board design of this project. These standards dictate part spacing, track spacing, and component placement. The goal of this is to improve the quality of PCB boards.

IEEE 802.15.1

Since this project used a Bluetooth module for wireless communication, the IEEE 802.15.1 [10] standard was dealt with, which provides resources for those who implement Bluetooth devices.

Universal Asynchronous Receiver-Transmitter

Universal Asynchronous Receiver-Transmitter (UART) [11] is a device-to-device communication protocol that follows standard procedure. This was the standard in the project that was used to create communication between the MSP430 and the HC-06 Bluetooth module.

1926.441 OSHA

Since the hardware component of the project involved charging and power delivery of a battery, 1926.441 OSHA [12] was applied to the project. This is for both the battery itself and the PCB design that was made to charge it on USB power.

Standard Triangle Language

Standard Triangle Language [13] is the file format that is native to the CAD files created by 3D software. This was used in the construction of the external housing.

Tools Employed

MATLAB

MATLAB [14] was used to create a model to see the output voltage of the sensor by varying the resistor connected to it in the voltage divider. This visual was then used to help the team analyze the tradeoff between sensitivity and range for each value, which was the deciding factor in choosing resistance values.

KiCad

KiCad [15] was initially used in the design of the LDO, charging IC, and the sensor layouts. The schematics were then transferred over to Multisim for the creation of the PCB board.

Multisim

Multisim [16] was used to create the final hardware interface for all of the electronic components on the board. For this software, we had to learn how to make custom layout parts for the block diagram.

Ultiboard

The Multisim files were exported to Ultiboard [17], which was then used to create a PCB board. We had to learn how to create custom layout components to get them laid out on the PCB board.

Code Composer Studio

Code Composer studio [18] was used for both programming the microcontroller and setting up the Bluetooth module. The UART and ADCs were configured within the software in order to gather the data that was collected by the sensors. Afterwards, it was then interfaced with the Bluetooth booster pack to transmit the data wirelessly. Both functions required extensive research in order to configure everything correctly. These involved looking through the programming manuals for each component and looking at the previous projects that used such boards in order to adapt them to our project.

LABVIEW

LABVIEW [19] was the chosen program to create the software application because it offered easy reading from the COM ports of a computer, which was taken up by the Bluetooth module.

Autodesk Inventor

To create a housing for the protection of the microcontroller and PCB board, Autodesk Inventor [20] was used to design it.

3D Printer

A 3D Printer was used to print out the housing for the final project.

Virtual Bench

The virtual bench [21] was vital in testing the manufactured PCB board to ensure that the voltage readings coming out of the sensor was correct.

Arduino

To help parallelize development, an Arduino board [22] was interfaced with the HC-06 Bluetooth module while the PCB was being constructed and tested with it in order to make the data-viewing software more testable.

Ultimaker

Ultimaker [23] was used as a 3D printing aid. This program converted the stl file that was made in Inventor to gcode for 3D printing.

Ethical, Social, and Economic Concerns

The use of our system could potentially result in someone's condition deteriorating. This would mainly be caused by if they used it without any influence from a professional, which could result in them trying to develop a worse posture just based on the data. This could also make some doctors and coaches lose customers as they try to perform a self-diagnosis and treatment plan without prior training.

Data privacy is important, especially medical data. This project at its current state is susceptible to a data breach by either having an external device connect to it or having the resulting data from the software breached. One way to help alleviate this problem would be to encrypt the data coming out from the Bluetooth module and have a unique encryption key stored by the software.

While the goal of the project is to create a more affordable and portal system of measuring FPD, there will always be a demographic that is unable to take advantage of our system. This is a significant issue because the project is design to be used long-term in order to be able to view trends in data that either point towards a patient improving their form or any further problems that may be caused.

Intellectual Property Issues

"Smart shoe module" is a patented device made for analyzing the motion of a wearer. The device uses digital signals to measure the motion of the wearer. One claim that relates to our project is "A shoe module comprising: a controller implemented to process the signal generated in the first circuit unit" [24]. Our project also uses a controller to process the signals generated by our sensor circuitry, but this claim is not enough to reduce the patentability of our project, as its part of a larger set of claims that are not very similar to our project. This patent also has motion sensors, comprising of an acceleration sensor and a gyro sensor, and only utilizes one digital pressure sensor. As a result of all of these differences, this patent does not reduce the patentability of our project.

More smart insole products could be found; in this case, it was a "Method and apparatus for customizing insoles for footwear" [25]. This device is described as a pressure plate with a grid of sensors and is used for generating a custom insole. One of their claims that relate to our project is "An apparatus comprising: a pressure plate having an array of pressure sensors for generating a two-dimensional pressure map of the sole of a foot" [25]. Our project also contains an array of sensors for generating a two-dimensional pressure map that is different in the term pressure plate. Ours is not a plate, but a sole made to be worn in a shoe. As a result, our project provides novelty, and patentability should not be affected by this patent.

The most similar patent was the "Footwear having sensor system" [26]. This patent is described as "An article of footwear includes an upper member and a sole structure, with a sensor system connected to the sole structure. The sensor system includes a plurality of sensors that are configured for detecting forces exerted by a user's foot on the sensor" [26]. This description would be accurate if used to describe our project. One claim described is dependent on their primary claim, the dependent one being "a sensor system comprising a first force sensor

connected to the insert member and a port in communication with the first force sensor configured for communication with an electronic device." The description also discusses including a processor, memory, software, TX/RX, etc. to accomplish storage or transmission of data to an outside source. These claims are closely related to our project and might make getting a patent unlikely.

After researching patents similar to our project, it is unlikely that our product would win a patent suit, but possible. No device is described as using the same communications protocols, nor is there one that includes a software application for viewing the data. However, the patent in source 26 may be similar enough to deny a patent for our project.

Detailed Technical Description of Project

A high-level block diagram of the project is shown in Figure 1.

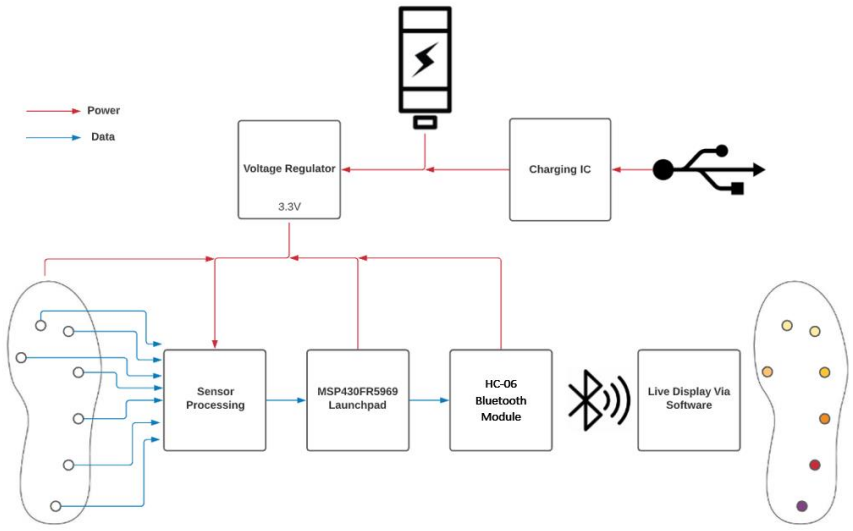


Figure 1: Block Diagram of the Project

Each block in the diagram shows a significant part of the system. This diagram shows both the hardware and software sides of the project, as well as how they interface. The software side of the project is broken down further with the flowchart in Figure 2, which describes the data's path and transformations. An important note to make is that the Bluetooth module displayed in the Figure lists it as the CC2650MODA, but the team changed the hardware to the HC-06 for the final project.

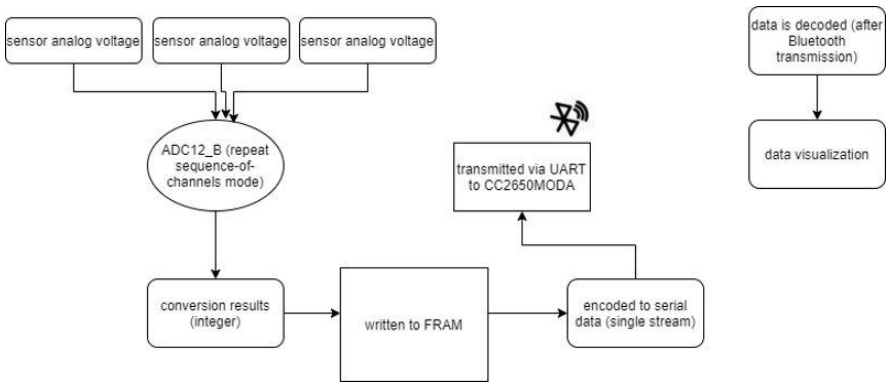


Figure 2: Software Flowchart

Hardware Design

The hardware components involved in our project can be categorized into two different parts that come together to form an interconnected circuit. First, the power unit involves a USB mini-B [27], a charging IC, a rechargeable 3.7V 1900mAh lithium polymer battery [28], and a fixed 3.3V output buck-boost converter [29]. The latter section of the circuit includes 7 force sensitive resistors (FSRs) i.e., pressure sensors [30], voltage divided and buffered from the microcontroller pins, meaning the outputs of the buffer (TLV274 Op-Amp) [31] are connected to Texas Instruments (TI) MSP430 [32] pins.

We have used “combicon” connectors [33] as analog signal entry connectors from the sensors, which allow us to remove the sensor wires from the board. In terms of the size of our header board, we originally planned to follow the TI booster pack standards. However, we needed a 14-pin connector (2 pins per FSR), which would not fit with these standards. Consequentially, we expanded the PCB outline to fit the “combicon” connectors, resulting in a perfect connection with the PCB. Figure 3 depicts the connection between the sensor blocks (TLV274 & Ckt), combicon connectors and modeled MSP430 pins.

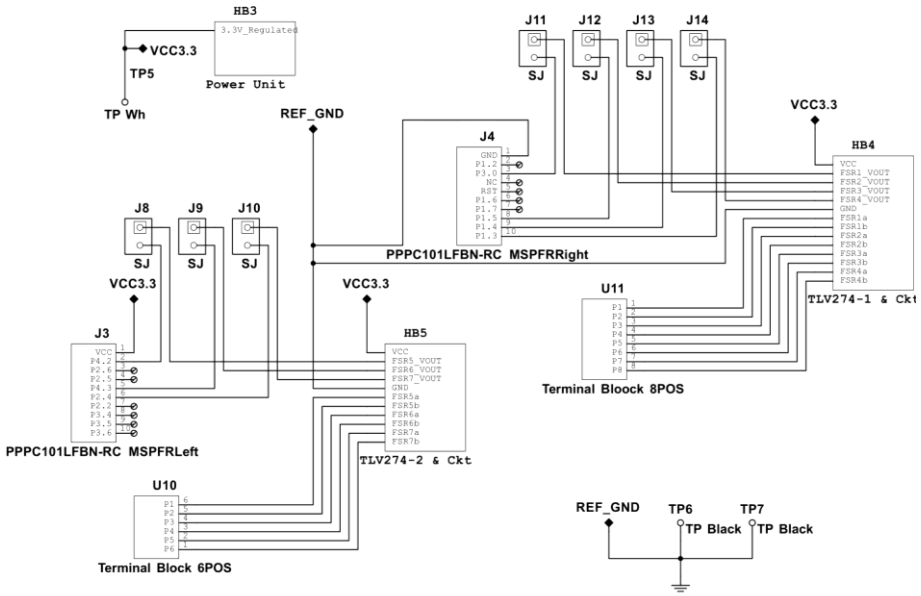


Figure 3: Vantage View Schematic of the Project

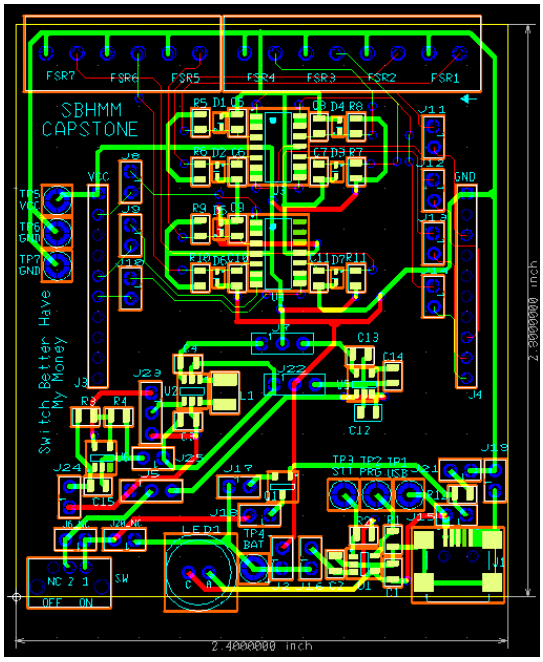


Figure 4: Overall Project Board Layout

1. Power Unit

The power unit design decisions have been made by centralizing the 3.7V rechargeable battery and all the parts that are selected-in depend on fulfilling the concept of having a rechargeable battery. As shown in the schematic below, a USB mini-B is connected to the charging IC block and the gate of a PMOS [34]. If the USB port is connected to a wall power source it will trigger the charging IC to recharge the battery and command the PMOS to act a switch and block the current between the battery and the voltage regulator. Once the USB is detached from the power source, the gate voltage is low, and PMOS allows the current flow from the battery to the voltage regulator, located inside the charging IC block. Next a sliding switch is used between the battery and the voltage regulator which acts as power on/off for the whole project units. And finally, if the switch is in contact, the battery voltage will be regulated by the voltage regulator providing a smooth 3.3V fixed output.

It can be noticed in Figure 5 that some extra number of shunt jumpers are used for board testability purposes, since the PCB had power issues in the first board send-out that couldn't be debugged even by the assistance of the course professors. The issue was later resolved by just using a new part footprint, known as SOT23-6.

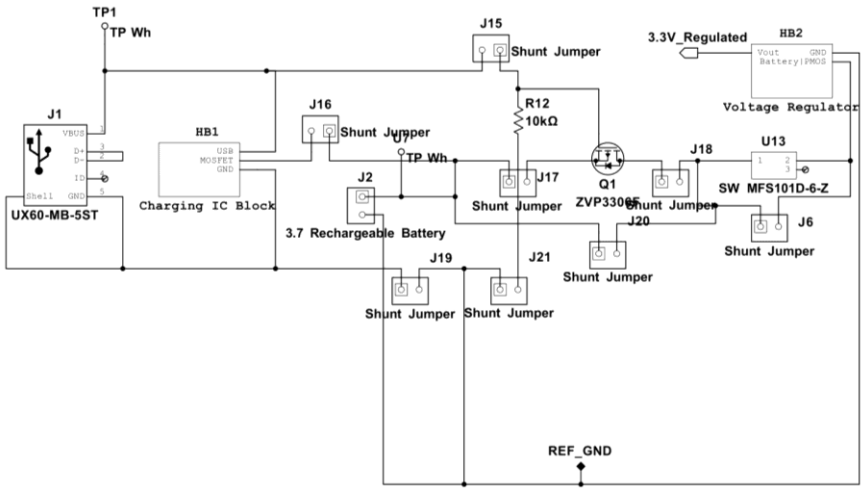


Figure 5: Power Unit Topology

i. USB Port

Despite having different USB port options, a UX60-MB-5ST USB mini-B [35] was chosen in the project for three reasons: smaller dimensions to save space on the board, simplicity of the USB type by having fewer pins, and having a footprint in the Multisim to save effort. The data pins in the USB are trivial since the port will be only used for the purpose of receiving power and not transmitting data. The shell or shield pin is connected to the ground to prevent electrostatic discharges (ESD) [36].

ii. Charging IC & Battery

While recharging the battery, the MCP73831 [37] acts as a charging controller, charging the battery with a voltage of 4.2V and a current of 500mA. We need to consider that for rechargeable batteries, the charging voltage differs from e discharging voltage. The MCP73831 was selected as it provides a charging voltage of 4.2V, matching the charging voltage of the battery. The recharge current is set at 500mA by the $R1 = 2k\Omega$ resistor. This value of the current is a safe charging rate since the max charging current for the battery is 0.5C meaning 950mA. Once the battery is fully charged, VBAT becomes an open circuit, so no current will flow into the battery. Additionally, STAT will signal the LED1 [38] to turn on, indicating to the user that the battery is full. Although, the LED is optional, but we preferred exploiting this option by putting an LED not connecting STAT to the microcontroller for having limited available pins. In

terms of battery selection, a lithium polymer battery was chosen over the lithium-ion, since the lithium polymer is made of solid material, making it safer for our wearable device.

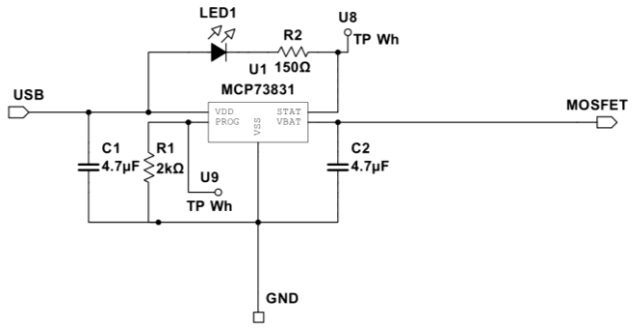


Figure 6: Charging IC Block Schematic

iii. Voltage regulator

Another important consideration is that battery voltages are not always constant. We expect the voltage to be 3.7V, as rated for our battery, but the voltage decreases as it is used. For our battery 3.0V is considered as fully drained. In this case, an LTC3531ES6 buck-boost converter [39] was selected to regulate the battery voltage. It functions in buck mode when the battery is above 3.3V, and boost mode when the battery is below 3.3V, providing our system with a constant 3.3V. The battery is considered dead once it hits 3.0V. To prevent over-discharge of the battery a TLV3031R5DBVR chip-enable comparator [40] is used to enable and disable the buck-boost regulator. The buck-boost enable is driven high by the comparator if the voltage is higher than 3.1V and if lower, the power unit shuts off. Additionally, a TC2014-3.3VCTTR [41] low dropout voltage regulator (LDO) is integrated in the circuit to provide the power using an LDO if the buck-boost fails to deliver the power.

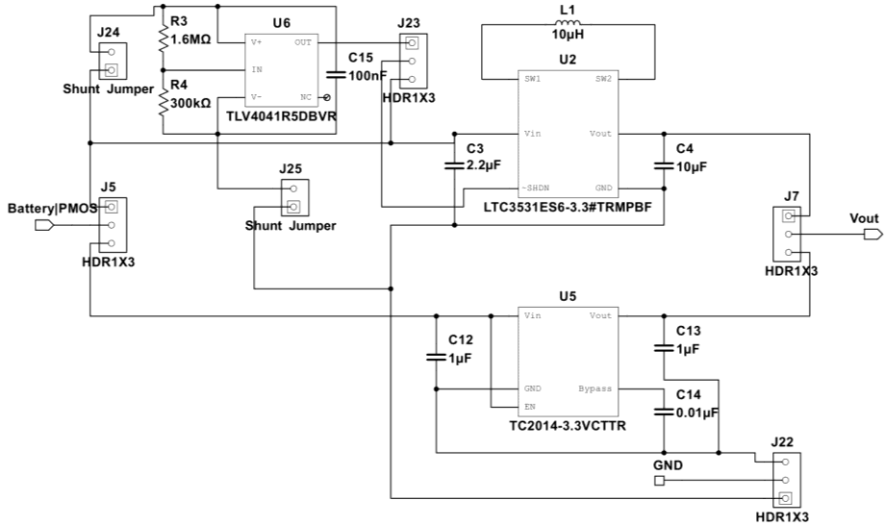


Figure 7: Voltage Regulator Block Schematic

2. Sensors and the Sensor Circuit

Overall, 7 Interlink Electronics 402 FSRs [42] were used by placing them in calculated locations on the sole. The FSR wires will be connected to the PCB by connecting one pin to the ground and other pin to the voltage dividers shown below. The original setup demonstrated in the datasheet of the FSRs connects the static-valued resistor to the ground and FSR's one pin to VCC instead of the ground. We altered the circuit by swapping the static-valued resistor and dynamic valued resistors (FSR) positions to protect the rest of the circuit in the case of the FSR shorting. The op-amps are used to buffer the circuit from the microcontroller and UClamp330 TVS diodes [43] are used to protect against transient voltages, which are sudden high voltage spikes. For the op-amp, we selected RRIO TLV274 [31] from Texas Instruments for its precision and better dynamic range. An RRIO op-amp is essential since the input and amp rail are powered from the same VCC and values are close. And finally, the output of the op-amp will be connected to the MSP430 analog pins and converted to digital using the pin ADCs.

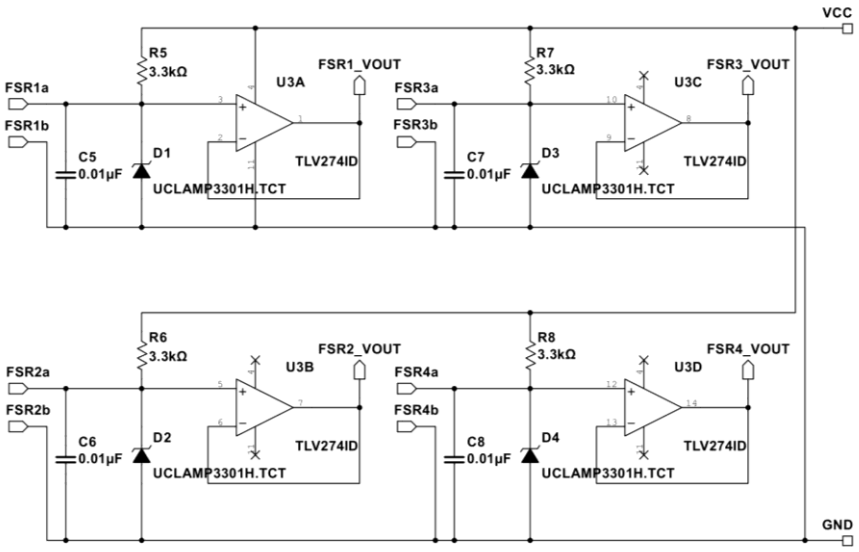


Figure 8: Sensor Block Schematic

To determine the value of the resistor to use that created the voltage divider, a MATLAB script was created to view the responsiveness of each resistor value. The team had to decide which value to use since there was a trade-off between the range of weights that could be easily detected and the sensitivity. A 3kΩ resistor was ultimately chosen because based on testing, it was found that when stepping on the shoe-sole, the weight applied to the sensor already placed it within the sensitive area. This would also help achieve better accuracy during pressure-sensitive activities such as jumping or running.

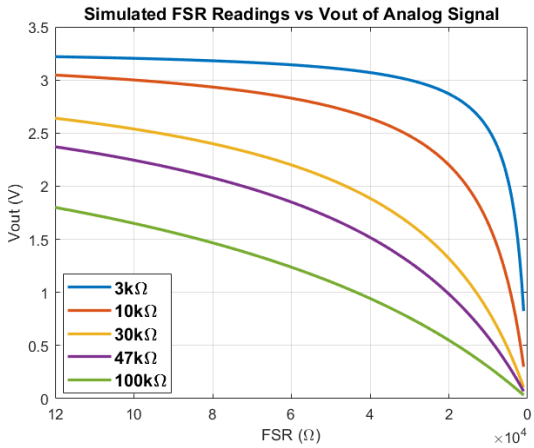


Figure 9: MATLAB Model of Resistors

Research was performed in order to determine the appropriate sampling frequency at which we would sample our sensors. The primary research of interest is an MIT study [44], which presents the results of a technique to determine the frequency content of gait. They used a pressure pad and sampled it at 2.5kHz. Figure 9 depicts the integral of power vs frequency for the cumulation of their data recording.

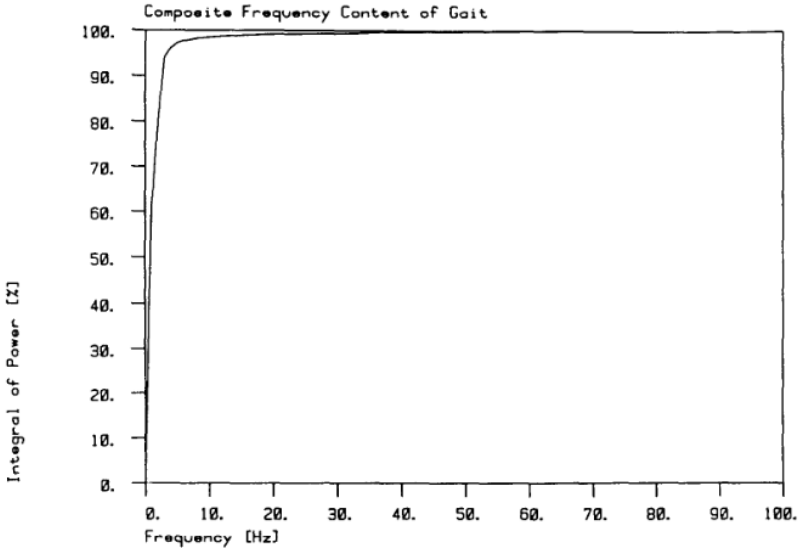


Figure 10: Integral of Power vs Frequency

As shown in the figure, the maximum frequency content of a walking gait is around 40Hz. Since we aim to provide a device that can be used by athletes, such as golfers evaluating the stability of their swing, we needed to accommodate for much higher frequencies. According to the Nyquist-Shannon sampling theorem, the sampling frequency must be greater than two times the maximum frequency content in your data. As we wish to view waveforms at a maximum of 250Hz, we set the sampling frequency at 1kHz.

Embedded Software Description

To determine what microcontroller to use, we researched for one that would satisfy a couple restraints for our project. The restraints were: enough analog inputs for our sensors, a voltage level suitable with our Bluetooth module, enough storage for our samples, and at least two UART modules. We decided to use the MSP430FR5969 [45], specifically the launchpad for this microcontroller, MSP-EXP430FR5969 [46]. The launchpad had several constraints with it. The maximum ratings for our microcontroller are shown below.

5.1 Absolute Maximum Ratings(1)

over operating free-air temperature range (unless otherwise noted)

	MIN	MAX	UNIT
Voltage applied at DVCC and AVCC pins to V_{SS}	–0.3	4.1	V
Voltage difference between DVCC and AVCC pins(2)		±0.3	V
Voltage applied to any pin (3)	–0.3	V_{CC} + 0.3 V (4.1 Max)	V
Diode current at any device pin		±2	mA
Storage temperature, T_{stg}(4)	–40	125	°C

Figure 11: Absolute Maximum Ratings for MSP430FR5969

Because the maximum voltage rating was 4.1 V, we set our voltage level to be 3.3 V (which is the value of voltage after the battery power is regulated). We also knew that the outputs of our sensors, which are inputted into the microcontroller, could not exceed 4.1 V. Our sensors were also powered with the 3.3 V regulated voltage to ensure we were within the operating voltage of the launchpad. As shown in figure 12, there are only 8 analog inputs on our launchpad (identifiable with the A[X] label). To be conservative, we decided on using 7 sensors for our sole.

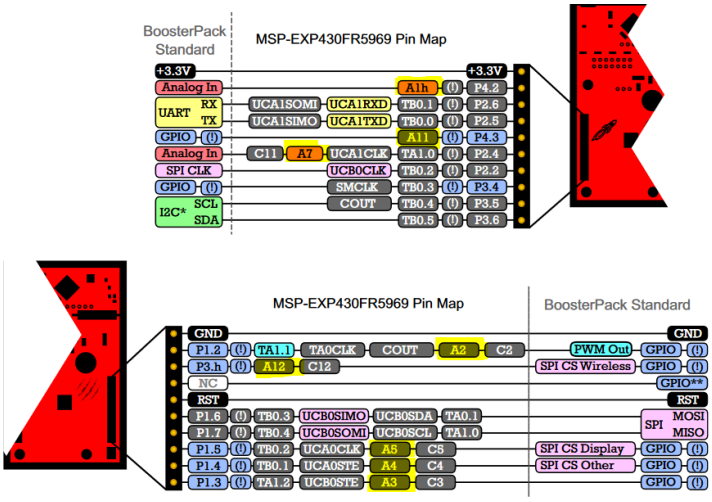


Figure 12: MSP-EXP430FR5969 Pinout

The process of reading the sensor data into the launchpad first begins with being given the sensor reading as an analog voltage, and each sensor is inputted to an analog pin. The next step is to take this analog voltage and convert it to a digital value, so it can be stored by our microcontroller. The MSP430 suite has a convenient Analog-to-Digital Converter (ADC) on board. For our application the idea was to sample many channels as fast as possible, and store the data. For this, the sequence-of-channels mode was selected for our ADC [45].

This mode operates by sampling a sequence of channels all at the same time and stopping the conversion at a set memory register. The state machine for this mode is shown below. Essentially, it relies on two bits being set to start the conversion: ADC12ON and ADC12ENC. I have a function called start_conversionADC() which sets these bits and is called at the beginning of our infinite loop. After waiting for all the channels to finish converting, the “end of sequence” (ADC12EOS) bit is set. After this, the values for each sensor (stored in ADC memory registers) are stored to memory as uint16_t’s.

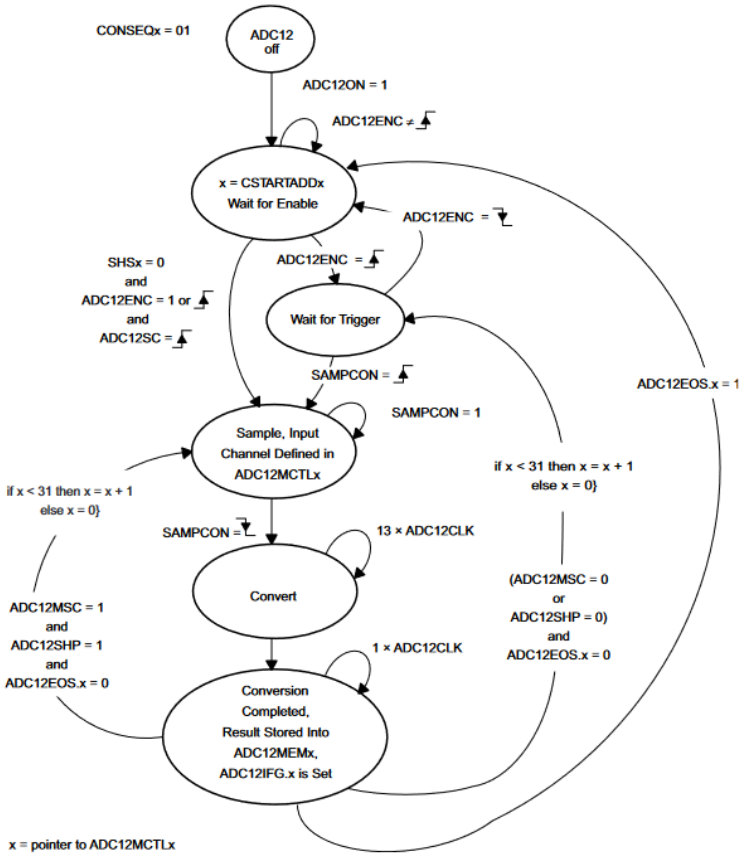


Figure 1313: Sequence-of-Channels Mode State Diagram

After completing the conversion for all channels and storing the conversion results, all that's left to do for our microcontroller is to send the data via UART to the bluetooth module. The bluetooth module we selected for our project is the HC-06 [47], which is a widely applicable module with a UART interface. To send the data over UART, this module had to first be configured on the MSP430. It was configured for a baud rate of 9600, with 8N1 encoding (8 data bits, one start bit, one stop bit, no parity bits). A function was written to send a string over the UART protocol, which is how our actual data was transmitted. The formatting of the string sent is shown below, with the dashes delineating the different sensor values.

```
char mem_str[120];

sprintf(mem_str, "%d-%d-%d-%d-%d-%d-%d-sbhmm\n", FSR_1, FSR_2, FSR_3, FSR_4, FSR_5, FSR_6, FSR_7);
UART1_WriteString(mem_str);
```

Figure 14: UART Transmission Format

Our code runs in an infinite while loop, so when we disconnect the launchpad from a USB connection it continues to run the logic. Before the while loop begins (executed once), the ADC and UART modules are configured. Then, at the beginning of the loop, the conversion begins. After the conversion is complete in the last ADC memory register an interrupt flag is set.

```
while(!(ADC12IV_ADC12IFG6)){ // wait for the last register in the sequence to finish conversion
    // do nothing
    ;
}
```

Figure 15: Waiting for Conversion to Finish

To ensure that all the memory registers are done with their conversion, we have a while loop in the infinite loop that does nothing while the interrupt flag is not set (shown above). After waiting for it to be set, the conversion is stopped.

```
case ADC12IV_ADC12IFG6:

    stop_conversionADC();                // stop converting ADCMEM registers after hitting interrupt for last analog input

    break;                               // Vector 24:  ADC12MEM6
```

Figure 16: IRQ Handler for ADC (ADC12 IFG6 FLAG)

The conversion is stopped in the Interrupt Request Handler (IRQ) for the ADC. This ensures that when the interrupt flag is set, the code immediately handles the interrupt by stopping the conversion. Then, the values are stored to memory and sent over UART.

Connecting Over Bluetooth

To connect the Bluetooth module to the computer, the first step is powering on the Bluetooth module in order to be able to pair to it. The Bluetooth module we are using is the HC-06 [47], the pinout for which is shown below. The module only has 4 pins: VCC, GND, Tx, and Rx. The VCC and GND pins are connected directly to the corresponding pins on the MSP430, while the Tx pin is connected to the Rx pin on the MSP430, and vice versa. The pinout for our MSP connection to the Bluetooth module is shown below.

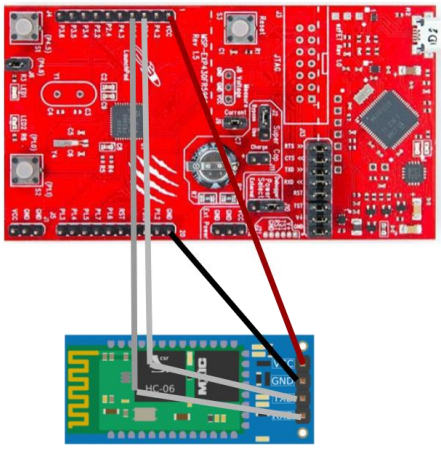


Figure 17: MSP Connection To HC-06

After the Bluetooth module is powered on, it will appear in the Bluetooth device list, usually in the settings application of a laptop computer. There is a red LED on the HC-06 that blinks corresponding to what mode it's in. After powering it on, the LED should blink steadily, indicating it is in "pairing mode." [48] Once paired with a device, the LED will still be blinking until you connect to it.

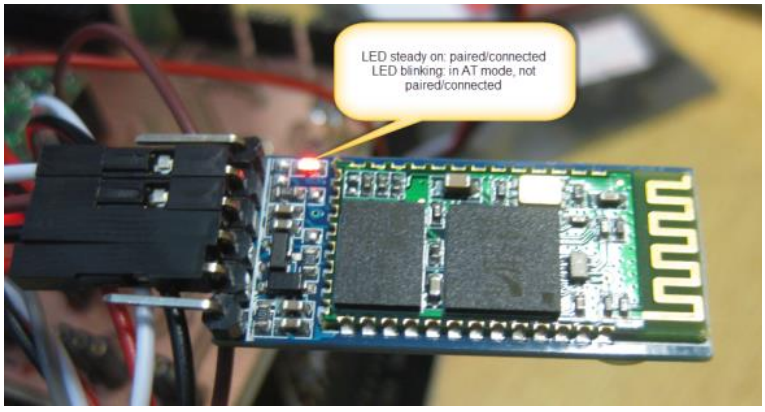


Figure 18: LED Modes

In order to connect to the Bluetooth module, we used a Bluetooth serial terminal. This software had the capability to connect to the Bluetooth module after being paired. For preliminary testing purposes, we used this terminal to receive data and check that our

transmission was working. In our LabVIEW [49] visualization program, the Bluetooth module will display as “paired” with the LED blinking until the program has been started.

Figure 19: Bluetooth Serial Terminal Layout

To change parameters of the Bluetooth module, “AT” commands can be sent to it while it is not connected to any devices. We were successfully able to communicate with our Bluetooth module using some of these commands, but not all. For example, we were able to change the name from “HC-06” to “SBHMM.” However, we were unable to configure the baud rate on this specific Bluetooth module, which constrained us at a baud rate of 9600. We ended up buying multiple HC-06 modules, with varying results for their performance.

1. Test communication

Send: AT (please send it every second)

Back: OK

Figure 20: Example of AT Command (from Datasheet)

The module we used for our project was unable to have its baud rate changed, but it was able to run on 3.3 V. The other modules we purchased were able to have their baud rate changed, but required at least ~4.2 V. So, we decided on the first Bluetooth module because it ran on 3.3 V which was the level of our regulated voltage. With the baud rate set as 9600, we were able to minimize the transmission delay as much as we could.

Data Display Software

Figure 21 displays the user interface of the project. In this interface, there is a setup where the Bluetooth device could be chosen. There are several other user controls here, which are to terminate the program, record the current input data, save the user data to a CSV file, and adjust the sensitivity of the color-display. In the middle, the data is visualized into a spectrum of colors, which is also placed under the shoe outline in order to give the user a better idea of intensity. Finally, the pressure readings in PSI are displayed.

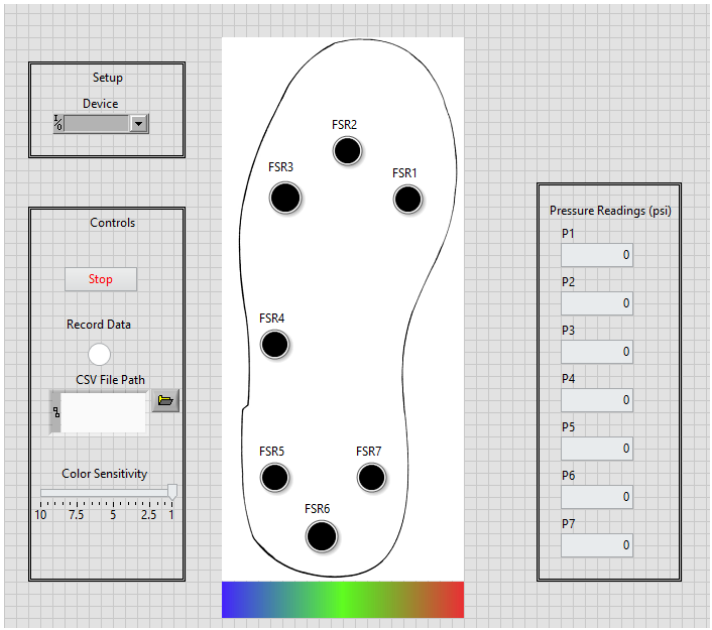


Figure 21: User Interface

The general flow chart for how the data display software works is shown in Figure 22. The software was created in LABVIEW because it allowed for the easiest pairing of the Bluetooth module through built-in libraries.

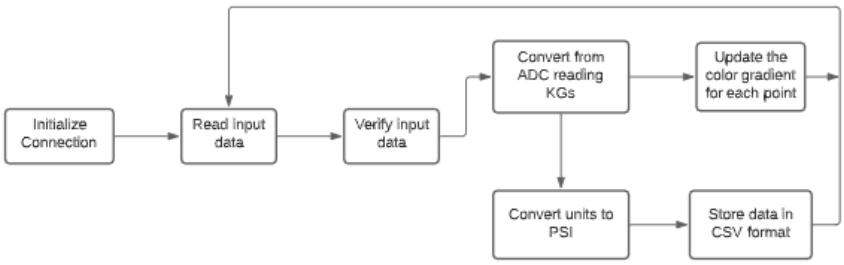


Figure 22: Data Display Flowchart

Connection Initialization

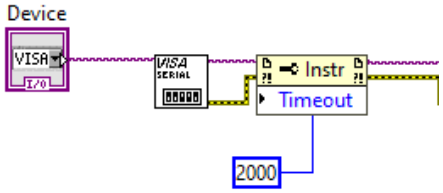


Figure 23: Connection Initialization

The part of the program involved in initializing the Bluetooth connection is shown in Figure 23. When the Bluetooth module is paired with the computer, a COM port is used. Within the LABVIEW software, the COM port that handles outgoing communication is selected.

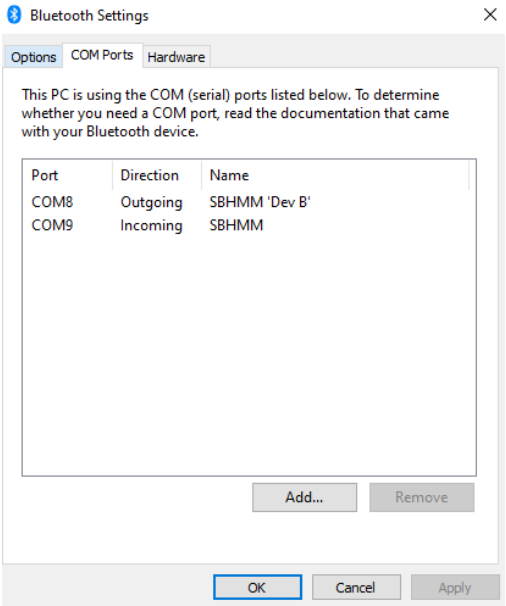


Figure 24: Example of Used COM Ports

Figure 24 demonstrates the paired device and COM ports used for selection. This logic also includes a time-out of 2 seconds, which will terminate the program if the device has not been connected within that time.

Reading Input Data

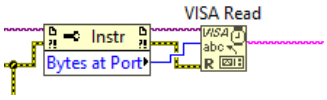


Figure 25: Input Data Reading

This part of the software reads data that is currently at the COM port. It was set up so that it would automatically read how many bytes were available. This was done to help in cases where the transmission string length was inconsistent (for instance, “4095-4095.....-SBHMM vs “500-600-… SBHMM). This stage and all subsequent stages were run in an indefinite while loop until either the program was stopped by the user, or the device was disconnected.

Data Verification

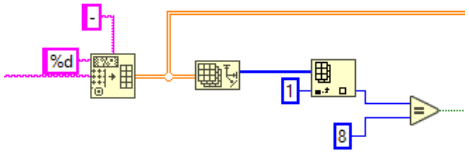


Figure 26: Data Verification

To ensure that the data received was valid for processing, the input string was first split into an array of doubles, delimited by the "-." Afterwards, the length of the array was checked to make sure that the total number of elements was 8 (7 from the sensors + SBHMM at the end). The purpose of this check was to ensure that no data strings were dropped during the transmission. The reason we check for an array size of 8 instead of 7 is because there were cases during testing where the last value would get cut off (example is when trying to transmit a value of 3000, it would get cut down to 300) so when there are 8 elements in the array, it can be assured that all the data readings were able to be transmitted.

At this stage, if the requirement of having an array size of 8 was fulfilled, then the program would move on to the next stage of data processing. If not, then the current data packet would be dropped and the program would wait for the next data string to come in to re-perform the verification.

ADC to Kilogram Conversion

The sensors were tested using the same circuit configuration as Figure 27 using weights from the physics department to find the value of VCC based on what the current weight was.

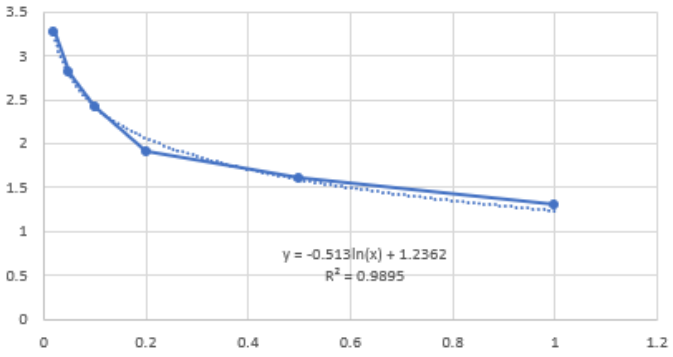


Figure 27: Regression Line for Sensors

In Figure 26, the x-axis is the weight put on the sensor while the y-axis is VCC. Based on the following graph, a trendline was made, with the equation relating weight to VCC as:

$$V_{CC} = -0.513 \ln(w) + 1.2362$$

Given the R^2 value of 0.9895, we were assured that the equation was accurate and would provide good data.

This equation was then converted to be based on what the weight would be given an input VCC to be

$$w = e^{\frac{V_{CC} - 1.2362}{-0.513}}$$

Since the ADC readings scaled from 4095 – 0 while VCC ranged from 3.3 – 0, the input reading from the ADC was divided by 1241 in order to make it fit within the formula range.

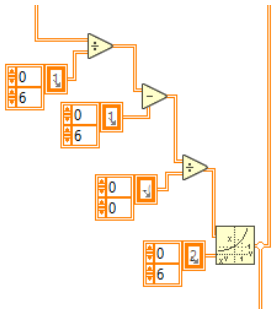


Figure 28: Software Unit Conversion

The final equation setup is shown in LABVIEW, where the input data (split into strings already) gets divided by 1241, subtracted by 1.2362, divided again by -0.513, then finally raised placed in an exponent by *e* in order to derive the weight, in kilograms, from the ADC reading.

Updating The Color Gradient

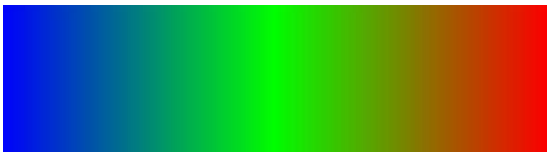


Figure 29: Software Color Spectrum

The chosen gradient would have blue representing no pressure, red representing the maximum amount of pressure, and green being the midpoint. In an effort to make the graphical feature of the data be available to a wider range of bodyweights, a sensitivity adjuster was created, which would control the specific weight that the data-display would max out (turn to red). With the

desired color spectrum chosen and given the inputs of the sensitivity and weight, the formula that represents each of the colors are as shown:

$$R = \max\left(0, 255 * \left(\left(2 * \frac{\max(w_{in}, sensitivity)}{sensitivity}\right) - 1\right)\right)$$

$$B = \max\left(0, 255 * \left(1 - 2 * \frac{max(w_{in}, sensitivity)}{sensitivity}\right)\right)$$

$$G = 255 - R - B$$

These would all give values for R, G, and B between 0 and 255. Once these values were calculated, they were condensed into a *uint* type because the input for the color point was of type uint and also it was to truncate any decimal points. The formula used to determine the proper uint value is

$$65536 * R + 256 * G + B$$

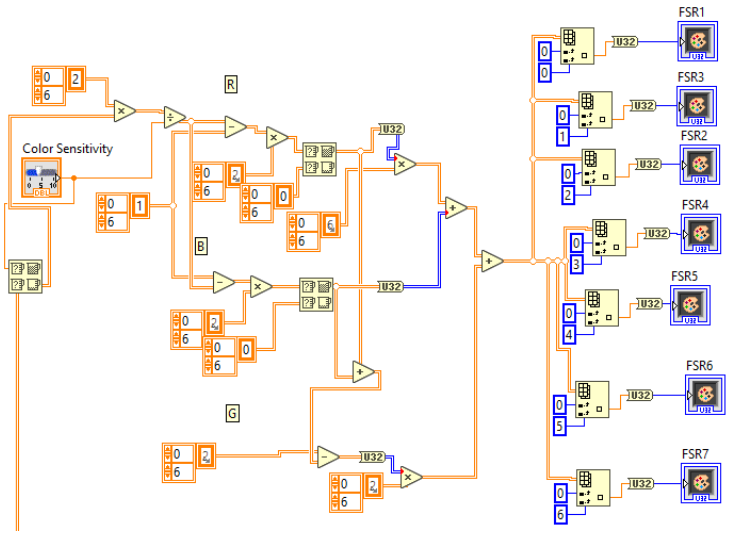


Figure 30: Software Color Updating Logic

This figure shows the program performing all of the calculations described, then assigned each color to the proper element in the data visualization.

Conversion to PSI

Based on previous scientific articles that were studied about Foot Pressure Distribution, we decided that pound per square inch (PSI) would be the most appropriate unit to show. To convert from Kilograms to PSI, the Kilogram unit was first converted to Pascals. Based on the datasheet, we knew that the sensors had a diameter of 13mm, which allows us to convert the reading to pascals. The conversion formula is shown

$$Pascals = \frac{w_{in} * (9.81)}{\pi * (6.5 * 10^{-3})^2}$$

Once the unit was derived in pascals, it was converted into PSI by just dividing it by 6895.

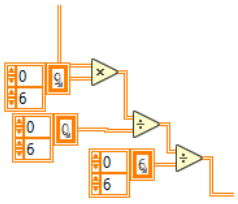


Figure 31: Unit Conversion to PSI

This part of the program shows the units being changed from kilograms (note, the calculation for area was pre-calculated for simplicity) to PSI.

CSV Storage

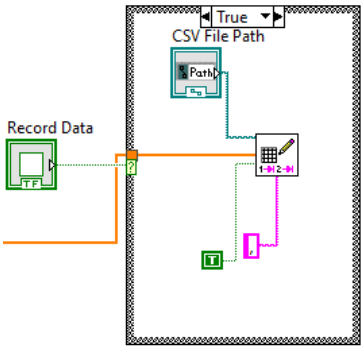


Figure 32: Software Data Storage Logic

To give the user more control, a design decision within the software was made so that you could selectively choose when to store data as a csv. To do this, another check was instituted where it

would see if the option to save as a CSV file was selected. If it was not, then the program would simply skip this part. When the option was chosen, all of the data that was converted into PSI would be appended to a new line of the CSV file, delimited by commas. If the file path that was chosen by the user already existed, then the new data would simply be added to the existing file. If not, then a new file would be created.

	A	B	C	D	E	F	G	H
1	425968	48.955	6.081	3.244	12.136	42.502	42.302	20.767
2	426039	48.955	6.081	3.244	12.136	42.502	42.302	20.767
3	426075	48.955	6.081	3.244	12.136	42.502	42.302	20.767
4	426110	48.955	6.081	3.244	12.136	42.502	42.302	20.767
5	426145	48.955	6.081	3.244	12.136	42.502	42.302	20.767
6	426180	48.955	6.081	3.244	12.136	42.502	42.302	20.767
7	426215	48.955	6.081	3.244	12.136	42.502	42.302	20.767
8	426250	48.955	6.081	3.244	2.407	42.502	42.302	20.767
9	426285	48.955	6.081	5.499	2.407	42.502	42.302	20.767

Figure 33: Example CSV File

Figure 29 displays an example of a CSV file that was generated by the software. Column A contains the time stamp, which is in units of milliseconds, the actual value is based on how much time passed midnight it was. This format for the timestamp was chosen because the team wanted the data to be easily manipulated and graphed, so this format was good for keeping an accurate time with one value. However, this means that if the user wants to keep track of the specific day, then they would have to name the file to that date in order to be organized.

External Design

Figures 34 and 35 depict the completed physical construction of the shoe-insole.

Figure 34: Complete Project Setup

Figure 35: Shoe Sole Sensor Placement

The size of the case was determined by the overall hardware components that included the PCB, the microcontroller, the battery pack, and the Bluetooth module. The size of it came out to be 65 mm in length, 46.5 mm in width, and 117 mm in height. The case also has an area that can slip a strap on for someone to attach around a leg. Adding caps to the case was essential in keeping all of the electronic hardware secure during usage of the project.

Figure 36: 3D Printed Case with Strap

Figure 37: 3D Printed Caps

The case and the lids were both 3D printed using software called MakerBot. It took multiple tries in 3D printing for a couple of reasons. One issue was with setting the MakerBot dimensions because the case was scaled too small. Another issue was that the model was too thin, resulting in the software not printing the entire case. Also, when printing a 3D model, the orientation of how it was printed affected the look of the case as printing the longer size would cause it to take a more curvature shape versus printing it on its shorter size would not have a curvature shape. The main reason for this problem was due deformities in the base of that the 3D printer extrudes on.

Figure 38: Close View of 3D Printed Case Interior

The lids are meant to slide onto each side of the case. The lids also provide space for the wires connected from the PCB to the sensors on the sole. There are 7 sensors on the sole that the PCB

is attached to. The sensors are placed strategically to represent more of how the weight was distributed on a person standing straight. According to [50], the heel carries around 60% of the load on a common foot, the “midfoot” (or bridge between top and heel of foot) carries around 8%, and the front of the foot carries around 28% of the load. Since most of the weight is distributed more on the heel area, three of the sensors were placed around that area of the foot to better represent that distribution. Furthermore, each sensor was sewed onto

Project Timeline

The original Gantt chart from the proposal is displayed in Figure 29 and the Gantt chart that was updated for the midterm design review is shown in Figure 30.

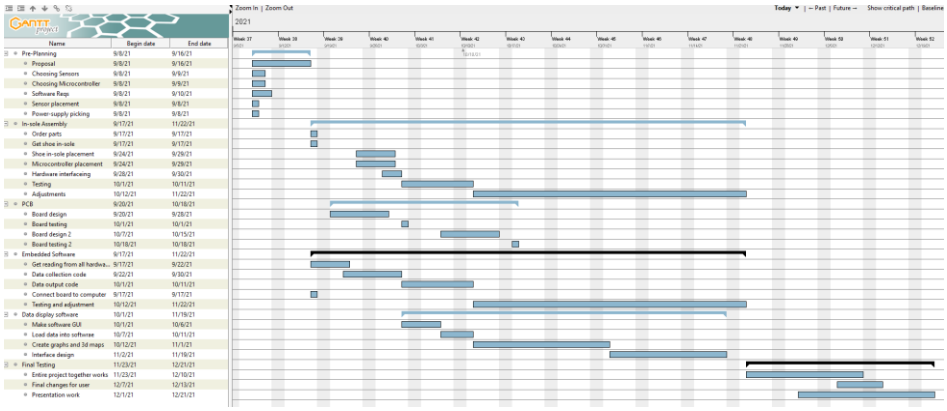


Figure 39: Original Gantt Chart

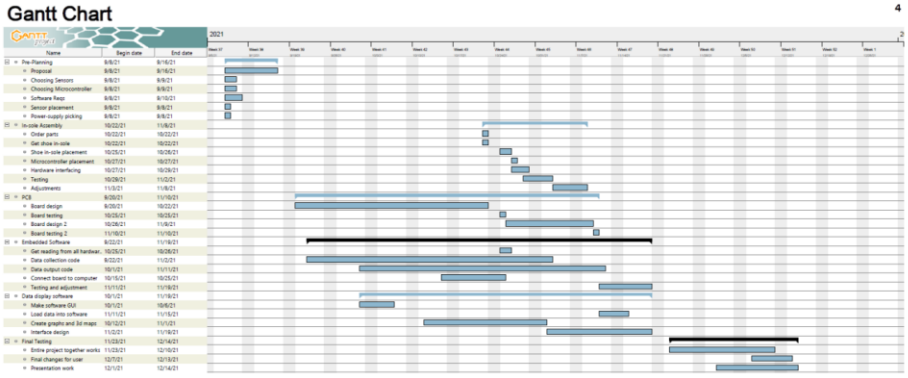


Figure 40: Midterm Gantt Chart

For these two charts, missing the first PCB board send-out date and then also realizing that some of the tasks, such as PCB testing and insole assembly, were able to be completed much faster than anticipated led to the pushing back of some of the items, such as the PCB testing and final

assembly and then the shortening of others like the in-sole assembly, PCB testing, and the final testing/verification. One important development that occurred around that time was a discussion within the team about how to further parallelize tasks. Initially, it was software and hardware would be developed in parallel, but within each subsection all of the tasks were serial. From the midterm design review, we learned that the tasks could be parallelized further within each subsection (hardware and software), which would be a trend going into the final Gantt chart.

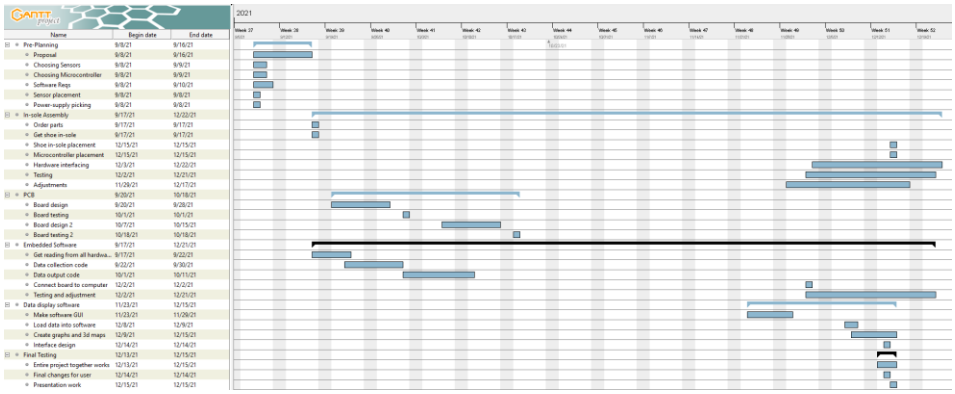


Figure 41: Final Gantt Chart

Figure 31 displays the final Gantt chart that followed the actual development timeline of the project. Most of the items were pushed back and completed on a much shorter timeline than originally planned. The main reason for the push back in many of the tasks was because development of the project was heavily held up by trying to get the Bluetooth communication functional, which was the hardest challenge. Originally, a CC2650MODA booster was planned on being used for wireless communications, but this had several problems. A lot of documentation from TI was completely removed from the website, leading to having to use code samples and the reference manual for help on programming it. The biggest issue faced with this module is that all of the sample code found was designed to be used on an MSP432, while we were using an MSP430, which made the existing code incompatible and attempts to modify it to fit the team's launchpad was unsuccessful. The reason that other tasks saw little to no progress during this time is because other team members were also taken off their responsibilities to try to help figure out how to work with the CC2650.

Once a new Bluetooth module was chosen, development quickly picked back up at a rapid pace and we were able to successfully complete the project on time for the final showcase. This was able to be done rapidly because after Bluetooth communication was established, all the other remaining tasks were much simpler and were with tools that team members were familiar with.

To further accelerate development at the end, tasks were further parallelized. Since the testing of the PCB was successful, the sensors were easily able to be placed on the in-sole. While that process was happening, the 3D model for the PCB boards and booster packs were completed. We were able to come up with ways in which testing and development could happen at the same

time for the same hardware. To test out the software with the MSP board while the shoe-insole was still being assembled, circuits that simulated the PCB board were built on breadboards. When the MSP was unavailable, an Arduino was used instead to help accelerate the development of the data-viewing software.

Serial Tasks

Despite the large overlap that could be done in many of the tasks, there were some project tasks that could not be completed until a previous stage was completed. Most prevalent in the hardware side is the process of testing and verification. These could not be done until either the PCB was manufactured or the custom booster pack for the Bluetooth module was completed. Other hardware related serial tasks was the CAD design, which could not accurately be modeled until the final hardware assembly was completed in order to get a more precise dimension. In the software side, sending data to the Bluetooth module via UART could not be completed until the ADCs and the UART for the MSP were configured. Finally, we could not perform entire system tasks until both the software and hardware components were completed.

Parallel Tasks

As previously stated, as the semester went on, it was realized that many of the tasks laid out could be parallelized. Initially, it was thought that the data-display software could not be successfully completed until the MSP430 programming was completed. However, by use of an Arduino, the completion of the data-display software could be done in parallel with the completion of the MSP430 code.

On the hardware side, testing of the PCB voltages and sensor interfacing could be completed while the sensors were being embedded into the shoe-insole through use of a breadboard circuit. Aside from this, the hardware and software components were heavily parallelized, resulting in both being completed around the same time for testing.

Team Roles

Eric Csehoski was primarily responsible for anything related to the force-sensitive resistors. These tasks included finding the correct sampling rate, picking the right force sensors to use for the project, making the sensor-reading circuitry, and sewing the force sensors on to the shoe insole. He had a secondary role in helping with the PCB design and verification.

Kieran Humphreys was primarily responsible for programming the firmware on the MSP430 board. This was configuring the UARTs and the ADC connections and also making the MSP430 interface with the HC-06 Bluetooth module to act as a sender. He had a secondary role in designing the software interface.

Ahmad Tamanna was primarily responsible for the PCB design and fabrication of the project. With this, he designed a system for power delivery through use of voltage regulators, battery charging, and analog signal processing. He had a secondary role in helping out with every other part of the project, by contributing to setting up Bluetooth communications, force sensor circuit design, and creating the 3D models.

Merron Tecleab was primarily responsible for all of the 3D printing that the team used in helping to design the appearance of the shoe-insole. He had a secondary role in sensor placement.

Xinyuan Zhu was primarily responsible for making the software application to view and process data while acting as a Bluetooth receiver. He created the features of the data visualization, sensitivity adjustment, unit conversions, and CSV file storage. He had a secondary responsibility in incorporating Bluetooth communication into the project.

Test Plan

Hardware Test Plan

Figure 42 shows the preliminary flowchart for our hardware test plan.

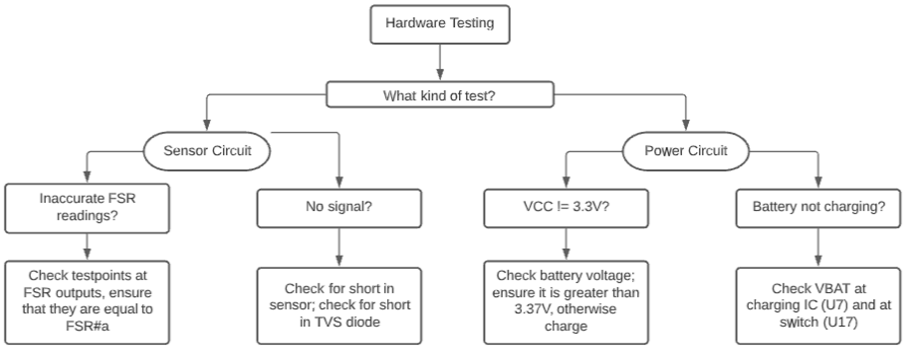


Figure 42: Hardware Test Plan

To test the hardware, we began with testing the power circuitry, as the sensor circuitry would not function without proper power. First, we checked the battery voltage to ensure that it was charged and within the expected bounds. Next, we connected the battery to the circuit, thereby connecting the battery to the buck/boost converter. If the system was functioning properly, it would output 3.3V to VCC. Moving forward, we tested the charging IC, and made sure that it prevented the battery from being charged once it reached 4.2V. We also tested the over-discharge circuitry using the VirtualBench. By applying 3.25V and stepping down, we can see if the buck/boost converter gets disabled by the comparator when the battery reaches 4.1V.

Next, we tested the sensor circuitry on our board. This section of testing is straight-forward; after connecting a sensor to the combicon and turning on the device, the voltage at each analog input pin should change with the sensor. When the sensor does not have any pressure applied, the voltage should sit around 3.2V; as pressure is applied, voltage should scale down to near 0V. Each input pin should function the exact same.

Software Test Plan

Our test plan for the software portion of our project will be presented sequentially. First, shown below is the test plan for checking if the ADC was converting properly, the first step in our data path. Ensuring that the ADC was converting all the proper channels at a reasonably fast rate (~15 µs) was sufficient to move on to the next step of the data flow.

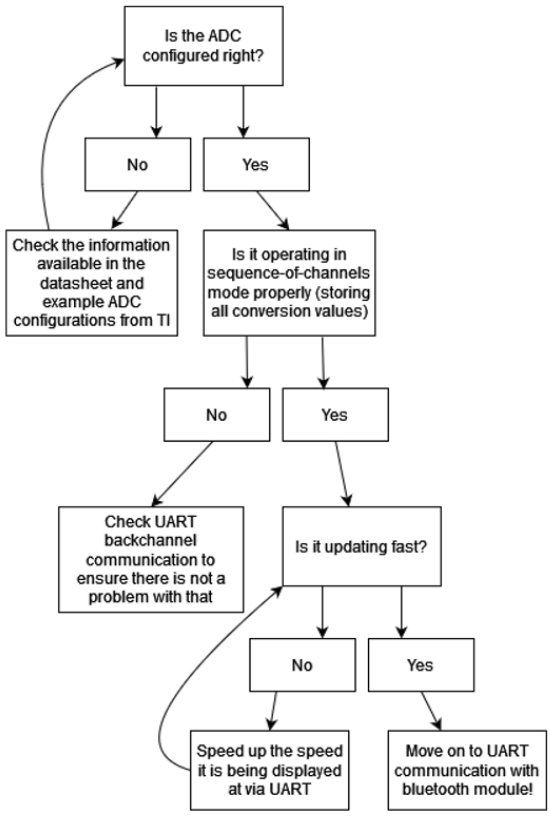


Figure 43: ADC Conversion Test Plan

Next, we had to configure the UART communication with the Bluetooth module, and the subsequent data transmission. The test plan for both of those steps is shown in figure 44. AT commands were sent to the module, which should respond with “OK.”

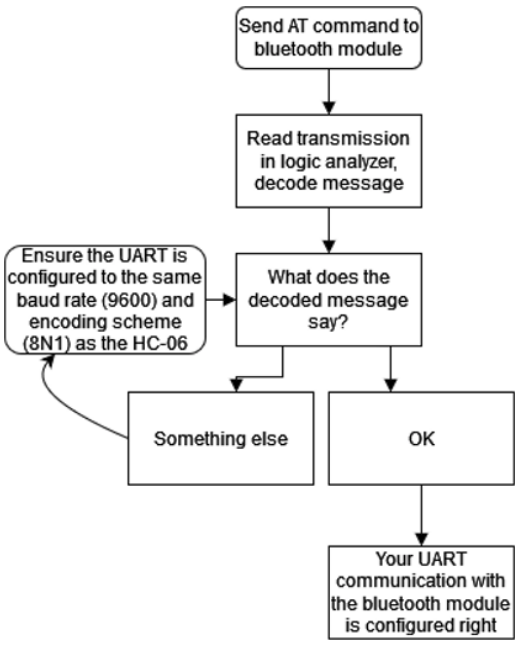


Figure 44: UART Communication Test Plan

Next, we moved on to sending values over Bluetooth from the microcontroller. The test plan for this is shown below. The process for this aspect of the project required a lot of troubleshooting. Often, the Bluetooth module was hard to connect to or would have to be unpaired and paired again.

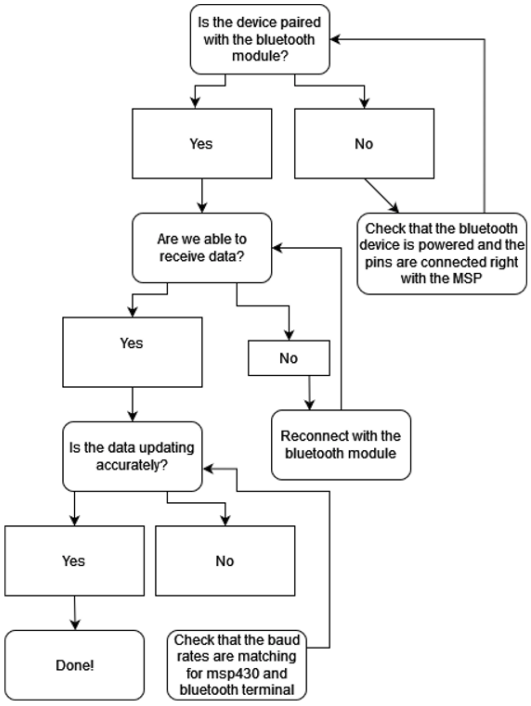


Figure 45: Bluetooth Transmission Test Plan

Once these tests were passed, we had confidence that the Bluetooth transmission module had correct behavior.

Data Display Module Test

Before the software was developed, an initial testbench was created to just read the Bluetooth buffer.

Figure 46: Read Buffer Test

This was done to verify that the software was behaving properly as a Bluetooth receiver. This test was important because prior to connecting it to a receiver, the team was under a presumption that the number of bytes transmitted would be the same every time, not considering that some readings would be shortened if there was a lot of pressure applied (a reading of 4095 vs a reading of 300). To rectify this, the number of bytes to read was automatically scaled with the number of bytes available to read.

Afterwards, the test program would be made to process the data into a color display, similarly to how it was done in the final program.

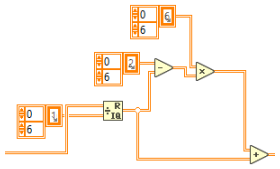


Figure 47: Test Program Data Processing

This process underwent multiple variations before the final programming was made for the data display. Initially, a MATLAB script embedded within LABVIEW was used to format and calculate the weight in kilograms based on the ADC reading. However, this method was problematic because tests showed that the script would often not update values where there was both an input and an output. To fix this, all MATLAB scripts were removed and were replaced by the LABVIEW math functions, which is seen throughout the final project.

When testing the final display software, test points were placed after every stage of data processing (as previously described in the technical description for the display software).

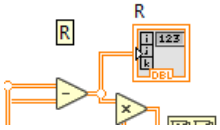


Figure 48: Test Point Sample

Figure 48 displays a test point that was created for the red value of the RGB color to make sure that the math created by a spectrum was correct.

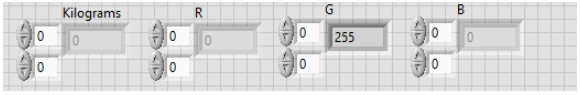


Figure 49: Test Points on Interface

Figure 49 displays a sample of test points as they showed up on the interface, which could then be easily examined during program execution to verify correctness.

Overall, a list of test points that were made in the software application were:

1. Checking the size of the array after the input string was split (int)
2. Seeing how often data passed the verification check. This was vital in determining a proper frequency to read the COM port (bool)
3. Checking the value of weight in kilograms after it was converted from the ADC reading. (int)
4. Checking the value of the conversion from kilograms to pascals (float)
5. Checking the value of the conversion from pascals to PSI (float)
6. Checking the value of each of the red, green, and blue values for the color spectrum (int)
7. Checking the value of converting from three RGB values that ranged from 0-255 to an unsigned integer

All these display test points were removed prior to the presentation session, once everything was verified.

When the MSP430 was unavailable because of either reconfiguring the firmware code or testing of the booster pack, an Arduino connected to the HC-06 module was used instead for the Bluetooth communication. During the tests with the Arduino, sample inputs were ones that scaled from 4095-0 linearly, ones that scaled from 4095-0 exponentially (which would translate to scaling linearly in weight), and ones where the values of data were completely randomized each time in order to simulate changing values from the sensor readings.

Final Results

Given all the difficulties faced in the construction of the project, we are proud that the project achieved everything that we set a goal for in the proposal.

Goal 1: Device wireless transmits data from the microcontroller to a computer

We were able to successfully achieve a wireless Bluetooth communication between the MSP430 and computer through use of the HC-06 Bluetooth module. The entire system was demonstrated to be able to function without any cables attached to the computer for data transfer.

Goal 2: Device and software show live updates for pressure

We felt that the system was reasonably responsive enough to live-updates when pressure was shifted. When measured, the software read data and processed it at a rate of 21 Hz. This could have been improved if the baud rate of the HC-06 could be increased, but we were satisfied at the result that we got being relegated to a baud rate of 9600.

Goal 3: Software displays data visualization

The software application was able to display the intensity of pressure on any given point by use of a color spectrum, which updated live as the weight was shifted.

Goal 4: Software demonstrates ability to store data in an Excel file

During testing, it was found that if data was constantly being stored while the system was active, the resulting CSV file would be almost illegible, so it was decided to give the user an extra control to pick whenever data was being stored. This feature was successful, as shown in the technical report.

Goal 5: Device can function without any attached cables for power or data transmission

This goal was met because during demonstration, all of the hardware was kept in a 3D printed case, using a battery for power delivery and the HC-06 module for data transmission.

Goal 6: Device is durable enough for use in repeated sections

Durability was a main concern since the project would be subject to repeated cases of being purposefully stepped on. During repeated testing and also the demonstration where many people were stepping on it to test out the data-viewing software, the device remained fully functional during that entire time.

Goal 7: Data taken from the sensor is accurate

When creating our own regression line to convert units of Voltage to Kilograms, the R^2 value was 0.9895, which demonstrated a high level of accuracy between the calculated weight and actual weight.

Even though the goals were met, we give ourselves an “A” over an “A+” for the functional part of the project. The reason behind this decision is because our project was limited by having to use a 9600 baud rate for the Bluetooth module. With this, the frequency at which we could

update the values in the sensor reading was much lower than what we originally wanted. In the final project, it was 21 while we wanted to push it up to 60-100. Another issue this caused was the delay in Bluetooth transmission. Tests with the Arduino showed that running on a higher baud rate decreased the delay between the sensor reading and software update. While we were able to get the delay down to 0.5 – 1 seconds, this was an area that we felt could have been improved on.

Costs

The final product of the Smart Sole ended up costing $177.89. However, there was more money that went into the development of the sole. We spent $88.83 on the first PCB board and its components, as well as $7 on an additional sensor for testing purposes. On top of this, we spent $34.80 on the CC2650 Bluetooth module, which ended up not being used in the project. We also made a mistake with our first attempt at 3D printing the casing, which cost $700. Figure 48 shows a summary of the project costs, and the appendix contains a more detailed spreadsheet of costs.

Smart Sole Product Cost and Design Expenses		
Product Costs		
Part	Cost ($)	Cost per product for 1,000 ($)
MSP430FR5969 Launchpad	19.19	4.53
Force Sensitive Resistors	53.03	29.63
Final PCB Manufacturing	33	0.65
Final PCB Components	57.67	42.56
3D Printed Casing	5	0.5
Miscellanous Parts	10	2
Total Product Cost	177.89	79.87
Design Expenses		
Part	Cost ($)	
Additional FSR	7	
First PCB Manufacturing	33	
First PCB Components	52.83	
CC2650 Bluetooth Module	34.8	
3D Printed Casing	700	
Total Design Expenses	827.63	
Total Project Cost	1005.52	

Figure 50: Smart Shoe-Insole Product Cost and Design Expenses

There was also an analysis of the product cost if mass produced. Every component benefits from discounts when bought in large quantities, but the microcontroller, force sensitive resistors, and PCB manufacturing benefit the most. As a result, the product would cost $79.87 in supplies. There would be added cost from assembly of the product, but this could be reduced through automatic assembly, with eligible parts including the PCB, sole, and 3D printed casing.

Future Work

We believe there are some key areas of our project which could be improved if a future team were to work on it. First, our hardware setup is kind of bulky, with two header boards and a microcontroller stacked on top of each other. This makes the box that users walk around in somewhat heavy and inconvenient. To correct this, a future team could choose a launchpad that has integrated Bluetooth or Wi-Fi capabilities. This would also reduce the power consumption of our overall system. Additionally, the header board and the launchpad could be integrated on one chip, reducing the system size and complexity.

Alternatively, a future team could opt to not use a microcontroller at all for this project. Because the microcontroller is only responsible for converting the sensor inputs, converting it to a digital value and storing that value, and then sending the data over UART, some of this logic could easily be offloaded to an FPGA. The analog-to-digital converter available on the MSP microcontrollers are handy, but an actual ADC could be used instead to limit power consumption.

In addition, the project could be worked on in the future to transmit and update readings at a higher rate. Currently, there is about a one second delay from when a user steps down on the sensor to when it changes color on the data display. Some things a future group could look at to reduce this delay is reducing the amount of data sent per packet and increasing the baud rate.

Currently, we are transmitting the sensor values separated by dashes so that the labview program can distinguish the different values. If a state machine were created for the receiver, this data could be sent seamlessly, with a guard page sent at the beginning of the transmission to indicate when a packet was incoming. After the receiver would read the guard page, it knows that each sensor value is coming in order, and how many bytes to read for each sensor.

Lastly, the security of our data transmission could be improved by a future group. Currently, our system transmits data over Bluetooth to a separate computer with no encryption or safety measures. A common encryption scheme used is AES. Since our project does use medical data, it is important to protect this information, and if our sole were to be used in actual medical applications, we would have to ensure the data transmission is secure.

References

[1] Hessert, M., Vyas, M., Leach, J., Hu, K., Lipsitz, L. and Novak, V., 2005. Foot pressure distribution during walking in young and old adults. [online] BMC. Available at: <https://doi.org/10.1186/1471-2318-5-8> [Accessed 10 September 2021].

[2] H. Odabas, C. Bulgan, B. Bingul and K. Sarpyener, The evaluation of foot pressure and postural structure of national golfers, AOTT, vol. 53, no. 2, pp. 150-153, 2019. Available: https://doi.org/10.1016/j.aott.2019.02.005. [Accessed 10 September 2021].

[3] B. Xia, H. Howlett and C. Lundy, Footcare product dispensing kiosk, 8,117,922, 2006.

[4] "Learn about the right to repair," *The Repair Association*. [Online]. Available: https://www.repair.org/stand-up. [Accessed: 17-Dec-2021].

[5] L. Parker, "Plastic pollution facts and information," *Environment*, 03-May-2021. [Online]. Available: https://www.nationalgeographic.com/environment/article/plastic-pollution. [Accessed: 17-Dec-2021].

[6] "The environmental impact of lithium batteries," *IER*, 12-Nov-2020. [Online]. Available: https://www.instituteforenergyresearch.org/renewable/the-environmental-impact-of-lithium-batteries/. [Accessed: 17-Dec-2021].

[7] J. Jhaveri, "[battery safety] top 5 reasons why lithium-ion batteries catch fire," *ION Energy*, 23-Apr-2020. [Online]. Available: https://www.ionenergy.co/resources/blogs/battery-safety/#:~:text=However%2C%20lithium%2Dion%20batteries%20are,and%20can%20cause%20widespread%20damage. [Accessed: 17-Dec-2021].

[8] "Barr_c_coding_standard_2018.pdf - keeps bugs out embedded C coding standard by Michael Barr \U00AE edition Barr-c 2018: Barrgroup.com Barr-C:2018 embedded: Course hero," *C*. [Online]. Available: https://www.coursehero.com/file/77005463/barr-c-coding-standard-2018pdf/. [Accessed: 17-Dec-2021].

[9] "Association connecting Electronics Industries IPC-2221A." [Online]. Available: http://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf. [Accessed: 17-Dec-2021].

[10] "IEEE 802.15 Wpan Task Group 1 (TG1)," *IEEE 802.15.1*. [Online]. Available: https://www.ieee802.org/15/pub/TG1.html. [Accessed: 17-Dec-2021].

[11] Person, "UART: A hardware communication protocol understanding universal asynchronous receiver/transmitter," *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices*. [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Accessed: 17-Dec-2021].

[12] “Department of Labor Logo United Statesdepartment of Labor,” *1926.441 - Batteries and battery charging. | Occupational Safety and Health Administration*. [Online]. Available: https://www.osha.gov/laws-regs/regulations/standardnumber/1926/1926.441. [Accessed: 17-Dec-2021].

[13] “What is an STL file?” *3D Systems*, 09-Jul-2020. [Online]. Available: https://www.3dsystems.com/quickparts/learning-center/what-is-stl-file. [Accessed: 17-Dec-2021].

[14] “Matlab,” *MathWorks*. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed: 17-Dec-2021].

[15] “KiCad Eda,” *Schematic Capture & PCB Design Software*. [Online]. Available: https://www.kicad.org/. [Accessed: 17-Dec-2021].

[16] “Multisim Live Online Circuit Simulator,” *NI Multisim Live*. [Online]. Available: https://www.multisim.com/. [Accessed: 17-Dec-2021].

[17] “Product,” *NI*. [Online]. Available: https://www.ni.com/en-us/shop/software/products/ultiboard.html. [Accessed: 17-Dec-2021].

[18] “CCSTUDIO,” *CCSTUDIO IDE, configuration, compiler or debugger | TI.com*. [Online]. Available: https://www.ti.com/tool/CCSTUDIO. [Accessed: 17-Dec-2021].

[19] “What is labview?,” *NI*. [Online]. Available: https://www.ni.com/en-us/shop/labview.html. [Accessed: 17-Dec-2021].

[20] “Inventor software: Get prices & buy official inventor 2022,” *Autodesk*, 15-Dec-2021. [Online]. Available: https://www.autodesk.com/products/inventor/overview?term=1-YEAR&tab=subscription. [Accessed: 17-Dec-2021].

[21] “VirtualBench software download,” *NI*. [Online]. Available: https://www.ni.com/en-us/support/downloads/drivers/download.virtualbench-software.html#360047. [Accessed: 17-Dec-2021].

[22] T. A. Team, “Software,” *Arduino*. [Online]. Available: https://www.arduino.cc/en/software. [Accessed: 17-Dec-2021].

[23] “Professional 3D printing made accessible: Ultimaker,” *ultimaker.com*. [Online]. Available: https://ultimaker.com/. [Accessed: 17-Dec-2021].

[24] “Smart shoe module,” by S. Shin, K. Kim, M. Kim, J. Kim, K. An, S. Jang, J. Cho, D. Han, C. Hwang. (2021, Jan 19). US10893719B2. Accessed on: Dec. 16, 2021. [Online]. Available: https://patents.google.com/patent/US10893719B2/en?q=smart+sole+pressure+sensor&country=US

[25] “Method and apparatus for customizing insoles for footware”, by E. Schwartz, M. Schwartz, R. Cahanap, L. Schwartz. (2011, Sep 28). EP1971946B1. Accessed on: Dec. 16, 2021. [Online]. Available: https://patents.google.com/patent/EP1971946B1/en?q=shoe+sole+pressure+sensor&oq=shoe+sole+pressure+sensor

[26] “Footwear having sensor system”, by M. Amos, A. Owings, A. Dean, A Schrock. (2019, Sep 3). US10398189B2. Accessed on: Dec 16, 2021. [Online]. Available: https://patents.google.com/patent/US10398189B2/en?q=smart+shoe+sole+sensor&country=US

[27] “USB Connector Type Guide.” Newnex, Accessed: Oct. 10, 2021. [Online]. Available: https://www.newnex.com/usb-connector-type-guide.php

[28] “Battery LP103450JH+PCM+PTC+2 WIRES 70MM.” Jauch Quartz, Nov. 2018, Accessed: Oct. 10, 2021. [Online]. Available: https://www.jauch.com/downloadfile/5bf5298855bac29715e11c4887bb44ba3/1900mah_-_lp103450jh_1s1p_2_wire_70mm.pdf

[29] “An Introduction to Buck, Boost, and Buck/Boost Converters.” Recom, Oct. 20, 2020, Accessed: Oct. 10, 2021. [Online]. Available: https://recom-power.com/en/rec-n-an-introduction-to-buck,-boost,-and-buck!sboost-converters-131.html?0

[30] “Force Sensing Resistor® (FSR) Sensor Series.” Interlink Electronics, Accessed: Oct. 10, 2021. [Online]. Available: https://www.interlinkelectronics.com/force-sensing-resistor

[31] “TLV274ID.” Texas Instruments, Nov. 2016, Accessed: Oct. 10, 2021. [Online]. Available: https://www.ti.com/lit/ds/slos351e/slos351e.pdf

[32] “MSP430FR5969 LaunchPad Development Kit (MSP-EXP430FR5969.” Texas Instruments, Jul. 2015, Accessed: Sep. 21, 2021. [Online]. Available: https://www.ti.com/lit/ug/slau535b/slau535b.pdf?ts=1632023701016

[33] “Combicon Connector 1803332.” Phoenix Contact, Jan .01, 2004, Accessed: Oct. 10, 2021. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Phoenix%20Contact%20PDFs/1803332.pdf

[34] “XP231P0201TR-G PMOS.” Torex Semiconductor Ltd., Accessed: Oct. 15, 2021. [Online]. Available: https://www.torexsemi.com/file/xp231p0201tr/XP231P0201TR.pdf

[35] “UX60-MB-5ST.” Hirose Connector, Aug. 2021, Accessed: Oct. 15, 2021. [Online]. Available: https://www.mouser.com/datasheet/2/185/doc_1806181827505-2576975.pdf

[36] “ESD Explained: What is Electrostatic Discharge?” Saline Electronics, Feb. 14, 2019, Accessed: Oct. 15, 2021. [Online]. Available: https://www.lectronics.net/esd-explained-what-is-electrostatic-discharge/

[37] “MCP73831T-2ATI/OT.” Microchip Technology, Feb. 28, 2020, Accessed: Oct. 15, 2021. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP73831-Family-Data-Sheet-DS20001984H.pdf

Formatted: French (France)

Formatted: French (France)

Formatted: French (France)

Formatted: French (France)

[38] "LED MT2118-G-A." Marktech Optoelectronics, Jul. 17, 2005, Accessed: Nov. 3, 2021. [Online]. Available: https://marktechopto.com/pdf/products/datasheet/MT2118-G-A.pdf

[39] "LTC3531ES6-3.3#TRMPBF." Analog Devices Inc., Accessed: Oct. 5, 2021. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/3531fb.pdf

[40] "TLV3031R5DBVR." Texas Instruments, Mar. 2019, Accessed: Nov. 12, 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/tlv4041.pdf?ts=1635170006955&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US%2526searchTerm%253DTLV4041R5DBVR%2526nr%253D6

[41] "TC2014-3.3VCTTR." Microchip Technology, Accessed: Nov. 12, 2021. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/21662F.pdf

[42] "30-81794, FSR 402" Interlink Electronics, Accessed: Oct. 5, 2021. [Online]. Available: https://cdn2.hubspot.net/hubfs/3899023/Interlinkelectronics%20November2017/Docs/Datasheet_FSR.pdf

[43] "UCLAMP3301H.TCT." Semtech Corporation, 2012, Accessed: Oct. 5, 2021. [Online]. Available: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/44000000MDWf/s0KJxyOHQyYOcvBXVBK9X3X8Cxj1gNltZ2vd2k1S0Wo

[44] E. Antonsson, R. Mann, "The frequency content of gait," Journal of Biomechanics, vol. 18, issue 1, 1985, Pages 39-47, ISSN 0021-9290, https://doi.org/10.1016/0021-9290(85)90043-0. (https://www.sciencedirect.com/science/article/pii/0021929085900430)

[45] "MSP430FR5969," *MSP430FR5969 data sheet, product information and support | TI.com.* [Online]. Available: https://www.ti.com/product/MSP430FR5969. [Accessed: 17-Dec-2021].

[46] "MSP-EXP430FR5969," *MSP-EXP430FR5969 Development kit | TI.com*. [Online]. Available: https://www.ti.com/tool/MSP-EXP430FR5969. [Accessed: 17-Dec-2021].

[47] "Guangzhou HC Information Technology Co., ltd ... - olimex," *Product Data Sheet*. [Online]. Available: https://www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf. [Accessed: 17-Dec-2021].

[48] E. Styger, "Using the HC-06 bluetooth module," *MCU On Eclipes*, 19-Jun-2013. [Online]. Available: https://mcuoneclipse.com/2013/06/19/using-the-hc-06-bluetooth-module/. [Accessed: 17-Dec-2021].

[49] *What is labview?* [Online]. Available: https://www.ni.com/en-us/shop/labview.html?cid=Paid_Search-7013q0000020cz6AAA-Consideration-GoogleSearch_LabVIEW_Brand&s_kwcid=AL%216304%213%21449107487685%21e%21%21g%21%21labview+ni&gclid=Cj0KCQiA5OuNBhCRARIsACgaiqWuT3wJcqrtNy

XHd_m6kDls9t-w70NrFceh7-q8fO9zVSFdCQWe4t4aApf0EALw_wcB. [Accessed: 17-Dec-2021].

[50] "Ultimaker 3D Printers & Filament," *Dynamism*. [Online]. Available: https://www.dynamism.com/ultimaker.html?position=&keyword=Ultimaker&device=c&network=s&matchtype=e&campaignid=388324046&adgroupid=1238050321885212&msclkid=2d3f96b0dc1f119a8460f7f778daa5bf&utm_source=bing&utm_medium=cpc&utm_campaign=_Brand_Ultimaker Exact&utm_term=Ultimaker&utm_content=Brand_Ultimaker. [Accessed: 19-Sep-2021].

Appendix

Table 1: Bill of Materials for Order Sendout 3

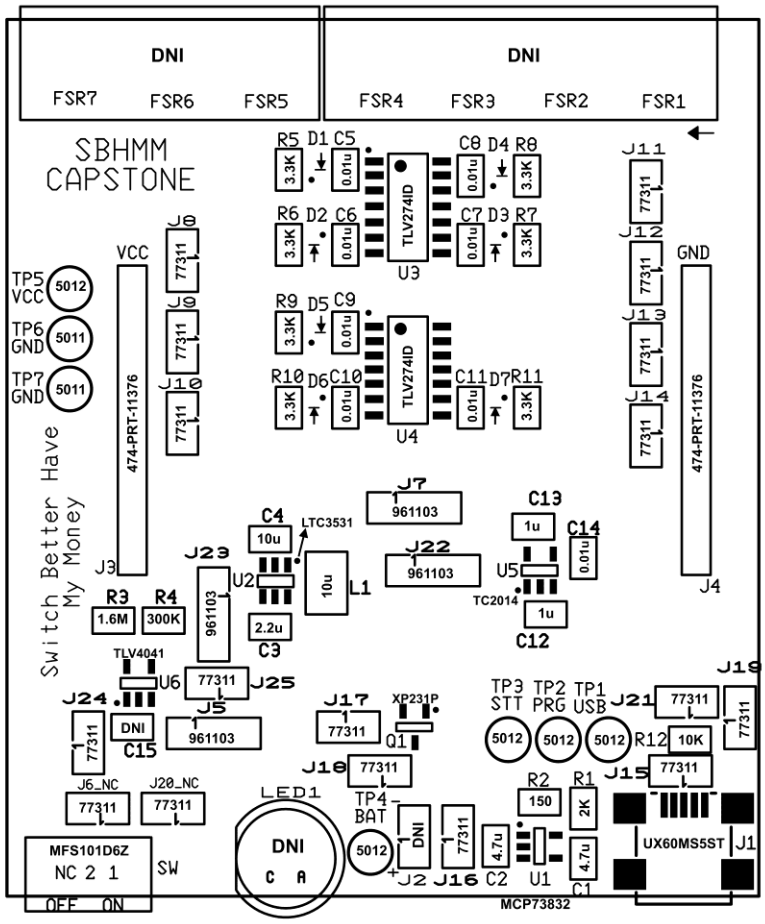
Index	Quantity	Part Numb	Manufactu	Descriptio	Customer	Available	Backorder	Unit Price	Extended I
1	2	296-26811	TLV274ID	IC OPAMP	SBHMM	2	0	1.67	3.34
2	7	UCLAMP3	UCLAMP3	TVS DIODE	SBHMM	7	0	0.7	4.9
3	20	390088-2-	390088-2	CONN SHU	SBHMM	20	0	0.702	14.04
4	1	563-1555-	MFS101D-	SWITCH SL	SBHMM	1	0	1.1	1.1
5	2	S7043-ND	PPPC101LI	CONN HDI	SBHMM	2	0	0.68	1.36
6	2	S1231E-10	PBC10SFCI	CONN HEA	SBHMM	2	0	1.32	2.64
7	1	P16044CT-	ERJ-P06D2	RES SMD 2	SBHMM	1	0	0.32	0.32
8	2	36-5011-N	5011	PC TEST PO	SBHMM	2	0	0.42	0.84
9	5	36-5012-N	5012	PC TEST PO	SBHMM	5	0	0.42	2.1
10	1	RHM150K(	ESR10EZPJ	RES SMD 1	SBHMM	1	0	0.1	0.1
11	1	893-XP231	XP231P02(	MOSFET P-	SBHMM	1	0	0.35	0.35
12	1	RHM10.0K	ESR10EZPF	RES SMD 1	SBHMM	1	0	0.16	0.16
13	8	399-17649	C0805Y10:	CAP CER S	SBHMM	8	0	0.43	3.44
14	1	AE08B-10-	AWG28-08	CBL RIBN :	SBHMM	1	0	7.05	7.05
15	1	445-6971-	CGA4J1X7I	CAP CER 1	SBHMM	1	0	0.4	0.4
16	1	296-TLV4(	TLV4041R:	LOW-POW	SBHMM	1	0	1.2	1.2
17	1	LTC3531E:	LTC3531E:	IC REG BCI	SBHMM	1	0	5.39	5.39
18	1	TC2014-3.	TC2014-3.	IC REG LIN	SBHMM	1	0	0.45	0.45
19	1	RHM1.60K	KTR10EZPF	RES SMD 1	SBHMM	1	0	0.18	0.18
20	1	445-17492	ADL3225V	FIXED IND	SBHMM	1	0	1.69	1.69
21	1	541-RCS08	RCS08053(	RES SMD 3	SBHMM	1	0	0.26	0.26
22	2	587-6522-	LMF212B7	CAP, MLC(	SBHMM	2	0	0.31	0.62
23	1	587-4932-	TMK212B7	CAP CER 2	SBHMM	1	0	0.31	0.31
24	2	445-14363	C2012X5R	CAP CER 4	SBHMM	2	0	0.31	0.62
25	7	541-RCS08	RCS08053I	RES SMD 3	SBHMM	7	0	0.2	1.4
26	20	609-4434-	77311-118	CONN HEA	SBHMM	20	0	0.201	4.02
27	4	3M155862	961103-68	CONN HEA	SBHMM	4	0	0.31	1.24
28	1	MCP73832	MCP73832	IC BATT CI	SBHMM	1	0	0.69	0.69
29	1	A113274-N	280358	CONN RCF	SBHMM	1	0	0.32	0.32
30	1	A106672-N	280370-1	CONN HEA	SBHMM	1	0	0.48	0.48

Table 2: Bill of Materials for WWW Electronics Solding

SBHMM BOM for WWW Electronics

Index	Quantity	Reference	Part	MANU PART#	MANU
1	1	U1	MCP73832	MCP73832T-2DFI/OT	Microchip Technology
2	1	U2	LTC3531	LTC3531ES6-3.3#TRMPBF	Analog Devices Inc.
3	2	U3, U4	TLV274ID	TLV274ID	Texas Instruments
4	1	U5	TC2014	TC2014-3.3VCTTR	Microchip Technology
5	1	U6	TLV4041	TLV4041R5DBVR	Texas Instruments
6	1	Q1	XP231P	XP231P0201TR-G	Torex Semiconductor Ltd
7	7	D1, D2, D3, D4, D5, D6, D7	UCLAM	UCLAMP3301H.TCT	Semtech Corporation
8	1	J1	UX60MS5ST	UX60-MB-5ST(70)	Hirose Electric
9	1	SW	MFS101D6Z	MFS101D-6-Z	Nidec Copal Electronics
10	1	L1	10uH	ADL3225VT-100M-TL000	TDK Corporation
11	2	C1, C2	4.7uF	C2012X5R1A475K060AB	TDK Corporation
12	1	C3	2.2uF	TMK212B7225MG-TR	Taiyo Yuden
13	1	C4	10uF	CGA4J1X7R0J106K125AC	TDK Corporation
14	2	C12, C13	1uF	LMF212B7105KGHT	Taiyo Yuden
15	8	C5, C6, C7, C8, C9, C10, C11, C14	0.01uF	C0805Y103K9RAC7800	KEMET
16	1	R1	2K	ERJ-P06D2001V	Panasonic Electronics
17	1	R2	150	ESR10EZPJ151	Rohm Semiconductor
18	1	R3	1.6M	KTR10EZPF1604	Rohm Semiconductor
19	1	R4	300K	RCS0805300KFKEA	Vishay Dale
20	7	R5, R6, R7, R8, R9, R10, R11	3.3K	RCS08053K30JNEA	Vishay Dale
21	1	R12	10K	ESR10EZPF1002	Rohm Semiconductor
22	2	J3, J4	474-PRT-11376	474-PRT-11376	Sparkfun
23	4	J5, J7, J22, J23	961103	961103-6804-AR	3M
24	17	J6_NC, J15, J16, J17, J18, J19, J20_NC J8, J9, J10, J11, J12, J13, J14, J21, J24, J25	77311	77311-118-02LF	Amphenol ICC (FCI)
25	2	TP6, TP7	5011	5011	Keystone Electronics
26	5	TP1, TP2, TP3, TP4, TP5	5012	5012	Keystone Electronics
27	1	C15	**DNI**	C0805T104J5RAL7800	KEMET
28	1	FSR1, FSR2, FSR3, FSR4	**DNI**	8POS TermBlock	Phoenix Contacts
29	1	FSR5, FSR6, FSR7	**DNI**	6POS TermBlock	Phoenix Contacts
30	1	J2	**DNI**	Battery3.7V	Jauch Quartz
31	1	LED1	**DNI**	MT2118-G-A	Marktech Optoelectronics

Note: Please solder J3 and J4 from bottom, meaning female pins should be facing bottom and don't cut the leads after soldering. Thanks.



Assembly Top Side

Figure 51: Soldering Layout for the WWW Electronics

Table 3: Budget Breakdown 1/3

1	SBHMM Budget						
2							
3	Before 10/24:						
4	Part Number	Manufacturer Part Numbe	Description	Quantity	Unit Price	Extended Price	Running Total
5	296-44946-ND	BOOSTXL-CC2650MA	CC2650 BLE BOOSTERPAC	1	34.8	34.8	34.8
6	296-35647-ND	MSP-EXP430FR5969	LAUNCHPAD MSP430FR59	1	19.19	19.19	53.99
7	485-166	166	PRESSURE SENSOR DEVEL	1	7	7	60.99
8							
9	Order #2:						
10	Digikey						
11	Part Number	Manufacturer Part Numbe	Description	Quantity	Unit Price	Extended Price	Running Total
12	1908-LP103450JH+PCM+PTC	LP103450JH+PCM+PTC+2 '	BATT LITH POLY 1S1P 190	1	17.33	17.33	78.32
13	MCP73831T-2ATI/OTCT-ND	MCP73831T-2ATI/OT	IC BATT CNTL LI-ION 1CEL	1	0.69	0.69	79.01
14	ADP2503ACPZ-3.3-R7CT-ND	ADP2503ACPZ-3.3-R7	IC REG BCK BST 3.3V 10LF	1	3.76	3.76	82.77
15	296-26811-5-ND	TLV274ID	IC OPAMP GP 4 CIRCUIT 1	2	1.67	3.34	86.11
16	UCLAMP3301HCT-ND	UCLAMP3301H.TCT	TVS DIODE 3.3VWM 8VC S	7	0.7	4.9	91.01
17	490-10965-1-ND	LQM2HPN1R5MEHL	FIXED IND 1.5UH 1.45A 15	1	0.47	0.47	91.48
18	445-6971-1-ND	CGA4J1X7R0J106K125AC	CAP CER 10UF 6.3V X7R 08	1	0.4	0.4	91.88
19	445-7679-1-ND	C2012X5R0J226K125AB	CAP CER 22UF 6.3V X5R 08	1	0.67	0.67	92.55
20	296-TLV4041R5DBVRCT-ND	TLV4041R5DBVR	LOW-POWER COMPARAT	1	1.2	1.2	93.75
21	390088-2-ND	390088-2	CONN SHUNT DUAL BEAM	10	0.702	7.02	100.77
22	563-1555-ND	MFS101D-6-Z	SWITCH SLIDE SPDT 300M	1	1.1	1.1	101.87
23	S7043-ND	PPPC101LFBN-RC	CONN HDR 10POS 0.1 GOL	2	0.68	1.36	103.23
24	S1231E-10-ND	PBC10SFCN	CONN HEADER VERT 10PC	2	1.32	2.64	105.87
25	399-7420-1-ND	C0805C475M9RACTU	CAP CER 4.7UF 6.3V X7R 0	2	0.56	1.12	106.99
26	541-RCS080560K4FKEACT-ND	RCS080560K4FKEA	RES SMD 60.4K OHM 1% 0	1	0.26	0.26	107.25
27	P20881CT-ND	ERJ-PB6B3403V	RES SMD 340K OHM 0.1%	1	0.33	0.33	107.58
28	P16044CT-ND	ERJ-P06D2001V	RES SMD 2K OHM 0.5% 1/	1	0.32	0.32	107.9
29	P21336CT-ND	ERJ-PB6D6043V	RES SMD 604K OHM 0.5%	1	0.23	0.23	108.13
30	P20890CT-ND	ERJ-PB6B4023V	RES SMD 402K OHM 0.1%	1	0.33	0.33	108.46
31	36-5011-ND	5011	PC TEST POINT MULTIPUR	2	0.42	0.84	109.3
32	36-5012-ND	5012	PC TEST POINT MULTIPUR	5	0.42	2.1	111.4

Table 4: Budget Breakdown 2/3

33	RHM150KCT-ND	ESR10EZPJ151	RES SMD 150 OHM 5% 0.4	1	0.1	0.1	111.5
34	893-XP231P0201TR-GCT-ND	XP231P0201TR-G	MOSFET P-CH 30V 200MA	1	0.35	0.35	111.85
35	277-1167-ND	1803633	TERM BLOCK PLUG 8POS 5	1	9.1	9.1	120.95
36	277-1210-ND	1803316	TERM BLOCK HDR 6POS 9(	1	2.55	2.55	123.5
37	277-1212-ND	1803332	TERM BLOCK HDR 8POS 9(	1	3.36	3.36	126.86
38	399-17649-1-ND	C0805Y103K9RAC7800	CAP CER SMD 0805 .01UF	7	0.43	3.01	129.87
39	RHM10.0KAECT-ND	ESR10EZPF1002	RES SMD 10K OHM 1% 0.4	8	0.16	1.28	131.15
40	1027-1001-ND	30-81794	SENSOR FORCE RES 0.04-4	7	7.576	53.03	184.18
41							
42	Mouser						
43	Mouser #	Manufacturer Part Number					
44	798-UX60-MB-5ST70	UX60-MB-5ST(70)	USB CONNECTORS	1	0.85	0.85	185.03
45							
46	Order #3						
47	Digikey						
48	Part Number	Manufacturer Part Numbe	Description	Quantity	Unit Price	Extended Price USD	
49	296-26811-5-ND	TLV274ID	IC OPAMP GP 4 CIRCUIT 1	2	1.67	3.34	188.37
50	UCLAMP3301HCT-ND	UCLAMP3301H.TCT	TVS DIODE 3.3VWM 8VC S	7	0.7	4.9	193.27
51	390088-2-ND	390088-2	CONN SHUNT DUAL BEAV	20	0.702	14.04	207.31
52	563-1555-ND	MFS101D-6-Z	SWITCH SLIDE SPDT 300M	1	1.1	1.1	208.41
53	S7043-ND	PPPC101LFBN-RC	CONN HDR 10POS 0.1 GOL	2	0.68	1.36	209.77
54	S1231E-10-ND	PBC10SFCN	CONN HEADER VERT 10PC	2	1.32	2.64	212.41
55	P16044CT-ND	ERJ-P06D2001V	RES SMD 2K OHM 0.5% 1/	1	0.32	0.32	212.73
56	36-5011-ND	5011	PC TEST POINT MULTIPUR	2	0.42	0.84	213.57
57	36-5012-ND	5012	PC TEST POINT MULTIPUR	5	0.42	2.1	215.67
58	RHM150KCT-ND	ESR10EZPJ151	RES SMD 150 OHM 5% 0.4	1	0.1	0.1	215.77
59	893-XP231P0201TR-GCT-ND	XP231P0201TR-G	MOSFET P-CH 30V 200MA	1	0.35	0.35	216.12
60	RHM10.0KAECT-ND	ESR10EZPF1002	RES SMD 10K OHM 1% 0.4	1	0.16	0.16	216.28
61	399-17649-1-ND	C0805Y103K9RAC7800	CAP CER SMD 0805 .01UF	8	0.43	3.44	219.72
62	AE08B-10-ND	AWG28-08/F-1/300	CBL RIBN 8COND 0.039 M	1	7.05	7.05	226.77
63	445-6971-1-ND	CGA4J1X7R0J106K125AC	CAP CER 10UF 6.3V X7R 08	1	0.4	0.4	227.17

Table 5: Budget Breakdown 3/3

63	445-6971-1-ND	CGA4J1X7R0J106K125AC	CAP CER 10UF 6.3V X7R 08	1	0.4	0.4	227.17
64	296-TLV4041R5DBVRCT-ND	TLV4041R5DBVR	LOW-POWER COMPARAT(	1	1.2	1.2	228.37
65	LTC3531ES6-3.3#TRMPBFCT-	LTC3531ES6-3.3#TRMPBF	IC REG BCK BST 3.32V TSO	1	5.39	5.39	233.76
66	TC2014-3.3VCTCT-ND	TC2014-3.3VCTTR	IC REG LINEAR 3.3V 50MA	1	0.45	0.45	234.21
67	RHM1.60MAHCT-ND	KTR10EZPF1604	RES SMD 1.6M OHM 1% 1/	1	0.18	0.18	234.39
68	445-174925-1-ND	ADL3225VT-100M-TL000	FIXED IND 10UH 450MA 1!	1	1.69	1.69	236.08
69	541-RCS0805300KFKEACT-NI	RCS0805300KFKEA	RES SMD 300K OHM 1% 0.:	1	0.26	0.26	236.34
70	587-6522-1-ND	LMF212B7105KGHT	CAP, MLCC, 0805/2012, 10	2	0.31	0.62	236.96
71	587-4932-1-ND	TMK212B7225MG-TR	CAP CER 2.2UF 25V X7R 08	1	0.31	0.31	237.27
72	445-14363-1-ND	C2012X5R1A475K060AB	CAP CER 4.7UF 10V X5R 08	2	0.31	0.62	237.89
73	541-RCS08053K30JNEACT-NI	RCS08053K30JNEA	RES SMD 3.3K OHM 5% 0.4	7	0.2	1.4	239.29
74	609-4434-ND	77311-118-02LF	CONN HEADER VERT 2POS	20	0.201	4.02	243.31
75	3M155862-03-ND	961103-6804-AR	CONN HEADER VERT 3POS	4	0.31	1.24	244.55
76	MCP73832T-2DFI/OTCT-ND	MCP73832T-2DFI/OT	IC BATT CNTL LI-ION 1CEL	1	0.69	0.69	245.24
77	A113274-ND	280358	CONN RCPT HSG 2POS 2.5	1	0.32	0.32	245.56
78	A106672-ND	280370-1	CONN HEADER VERT 2POS	1	0.48	0.48	246.04
79							
80							
81	Mouser						
82	Mouser #	Mfr. #	Description	Order Qty.	Price (USD	Ext.: (USD)	
83	798-UX60-MB-5ST70	UX60-MB-5ST(70)	USB Connectors	1	$0.85	$0.85	246.89
84	474-PRT-11376	PRT-11376	SparkFun Accessories Ardu	2	$0.50	$1.00	247.89