Solegait

November 12, 2015

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: ENSC 440W Design Specification Document for Solegait Pods

Dear Dr. Rawicz,

Please find attached our design specification document for a mobile gait analyzer. We would like to build an insole with embedded pressure sensors to observe how a patient walks, which then sends data to a mobile device to see whether their planted foot rolls slightly inward or outward.

The design specification document details the system overview, the insole and circuit design, the data processor and the mobile application. It also discusses design and performance differences between proof-of-concept, prototype, and product stage devices.

Our company is founded by five engineering students in various fields of study. The members include Shaquile Nijjer, Zachary Nunn, Karsten Harder, Alexandra Talpalaru, and Ashley Lesperance. If you have any questions or concerns, feel free to contact me by phone at 604-939-1780 or by email at snijjer@sfu.ca.

Sincerely,

Shaquile Nijjer
CEO
Solegait

ENSCLOSED: Design Specification Document for Solegait Pods

Solegait

Design Specification for an Assistive Rehabilitation Device Named:

Team Members: Shaquile Nijjer
Alexandra Talpalaru
Zachary Nunn
Ashley Lesperance
Karsten Harder

Contact Person: Alexandra Talpalaru
atalpala@sfu.ca

Submitted to: Dr. Andrew Rawicz
Steve Whitmore
School of Engineering Science
Simon Fraser University

Issue date: November 12th, 2015

Revision: 3.1

Abstract

Gait, the manner in which a person walks, can be severely affected when someone is injured. Moreover, it is strongly related to a person's risk of injury, and abnormal gait can lead to serious degradation of bone mineral density and muscle mass as a person ages. The goal of Solegait is to create a viable and wearable device which plots the patients foot-ground interaction to help correct an injured patients gait, and reduce the risk of gait related injuries.

The design specification for the Solegait Pods device provides a detailed description of the design and development of a force sensitive insole used for the purpose of rehabilitation and injury prevention. The specifications within this document pertain to both the proof-of-concept model as well as the prototype model. Following a system overview, justifications regarding design and materials choices, as well as the verification and testing of the device will be thoroughly discussed throughout this document.

Solegait

Table of Contents

List of Tables

List of Figures

Glossary

MCU: Microcontroller Unit

MATLAB: A technical computing language for algorithm development

FSR: Force sensitive Resistor

Pedobarograph: A plot depicting the pressure distribution over the plantar surface of someone's foot

Gait: The manner in which someone walks

Eversion: When the plantar surface of the foot rotates laterally (outwards)

Inversion: When the plantar surface of the foot rotates medially (inwards)

Pronate: Eversion of the plantar surface of the foot during walking

Supinate: Inversion of the plantar surface of the foot during walking

Plantar: Of or relating to the sole of the foot

COP: Center of pressure

MVC: Mode-View-Controller

API: Application Program Interface

JSON: JavaScript Object Notation

Solegait

1 Introduction

The Solegait Pods device is a simple, lightweight device which provides meaningful pedobarograph data of the user in real time. Our device consists of a pressure-sensitive insole and a data processor to output the processed data in the form of a pedobarograph. The purpose of the Solegait Pods is to monitor the users gait and asses and remove any gait abnormalities. By referring to the pedobarograph, the user can identify what particular issues they struggle with during regular walking, and effectively correct these issues.

1.1 Scope

This document specifies the design of the Solegait Pods device, and will explain how the design meets functional requirements, outlined in *Functional Requirements for Solegait Pods*, revision 2.1 [1]. The design specification addresses all of the requirements for a proof-of-concept model, as well as for a prototype model. Justifications for design choices, as well as a system test plan will be thoroughly discussed throughout this document.

1.2 Intended Audience

This design specification is intended to be used by all members of Solegait. Design engineers are required to comply with this specification document. The project manager will use this document as a reference while assessing the projects progress. Test engineers will refer to this document while implementing the test plan, and ensuring the reliability of the device.

1.3 Background

In current practice, gait analysis involves tracking a patients lower limb dynamics using motion cameras and sensors. During post-processing, segment positions and joint angles are calculated at each frame of the recorded video and compared to the reference data. This process is often tedious for both the patients and the therapists analyzing this data, as it requires that the patient actively locates a laboratory with this specific software, then wait a significant period of time for the results. This technology is also very expensive as it requires multiple sophisticated components.

With a growing number of older adults suffering from gait related injuries, athletes sustaining injuries which affect gait, and individuals with neurological disorders affecting gait, there is an increasing need for fast, reliable and cost effective gait analysis software. By tracking a patient's foot-ground interaction using a simple pedobarograph, we can measure and quantify foot-pressure abnormalities in a variety of clinical conditions [1], [2].

As a result, current trends in gait analysis have moved towards analyzing the foot-ground interaction by use of pedobarograph data, then administering a rehabilitation program. However, this current technology is extremely expensive and only available to medical practices and large research groups.

Our device, utilizing a dense sensor matrix implanted into an insole, will provide a cost effective solution for measuring and analyzing a patient's foot-ground interaction. Each of our components are indented to be safe, reliable, and cost effective in order to provide the patient and their therapist with a simple, portable, and real-time gait analysis software.

2 System Overview

The Solegait Pods device consists of a shoe insole embedded with pressure sensors, a microcontroller unit (MCU) to collect the data, a Bluetooth communication system to send the data to a mobile phone, and a mobile phone application. A workflow diagram of the Solegait Pods device is illustrated in Figure 1 below. The input to the device are the users' steps. The pressure applied over the insole is detected by each respective sensor, and transmitted to the MCU.

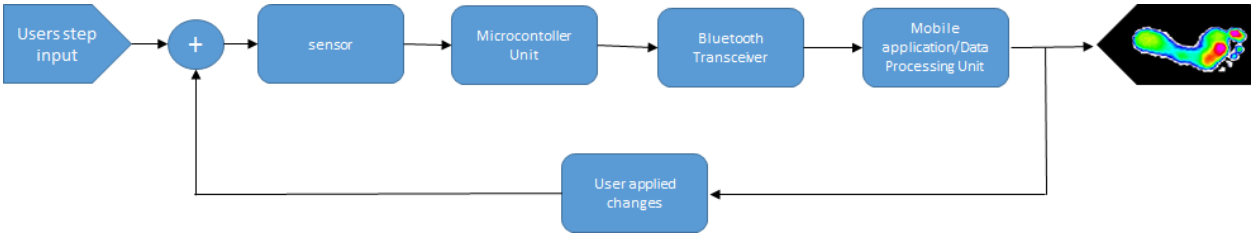

Figure 1: Graphical flowchart of Solegait Pods device

The Solegait insoles contain multiple layers, outlined in Figure 2. The top and bottom layers of the insole contain wires which are placed in opposite directions. In this configuration, the locations of the pressure sensors correspond to the areas where the horizontal and vertical wires overlap. The amount of pressure in these areas is dependent on how much the insole is compressed under the user's weight. The sensors are arranged on the insole in a pattern which ensures that more sensors are present in higher areas of interest, such as the calcaneus, arch, metatarsal heads, and the toes. As shown in the flowchart of Figure 1, the pressure sensors are then read by the MCU.

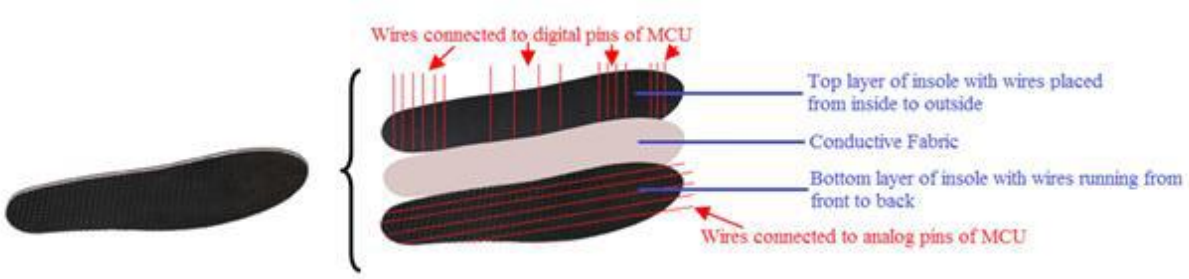

Figure 2: Layers of the Solegait Pods insole

When the FSR is compressed, it will transmit a current to the MCU, and the resulting voltage which is read by the MCU is in the range of 0V to 5V. Communication of the voltage data between the MCU and the user interface is facilitated by a Bluetooth system. This data is then processed and summarized in an application on the mobile phone. This application will then display the data as a real-time pressure plot (a pedobarograph) on the screen, allowing the user to fix input elements of gait prior to the next step. This document will outline the design of each component depicted within the flowchart of Figure 1.

The final product will consist of 64 pressure sensors embedded into the insole. Data will be collected by the MCU and transmitted to a mobile application via a Bluetooth transceiver. The application will process the data and display a real-time pedobarograph, as well as an average COP graph. Once sufficient data is collected, the application will provide the user with suggestions for gait correction.

3 Force Sensitive Resistor (FSR) and Electrical Design

In this section, we will discuss the description, proof-of-concept, prototype, and product for the electrical design phase.

3.1 Description

Force sensitive resistors are made from polymer that has a low conductivity in an initial state, and when pressure is applied, the polymer is compressed which causes the particles inside to get closer together, thus increasing the conductivity where pressure is applied. This increase in conductivity, or decrease in resistivity, causes a change of voltage with pressure. In Figure 3, the drawings show an elementary description of how a force sensitive resistor works as pressure is applied.

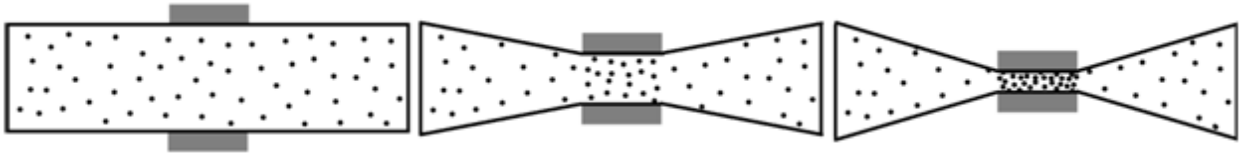

Figure 3: FSR when Pressure is applied

It is easily seen in Figure 3 that as more pressure is applied the particles between the wires get closer together which decreases the resistance between the wires.

Below in Figure 4 shows the circuit diagram used to measure the FSR.

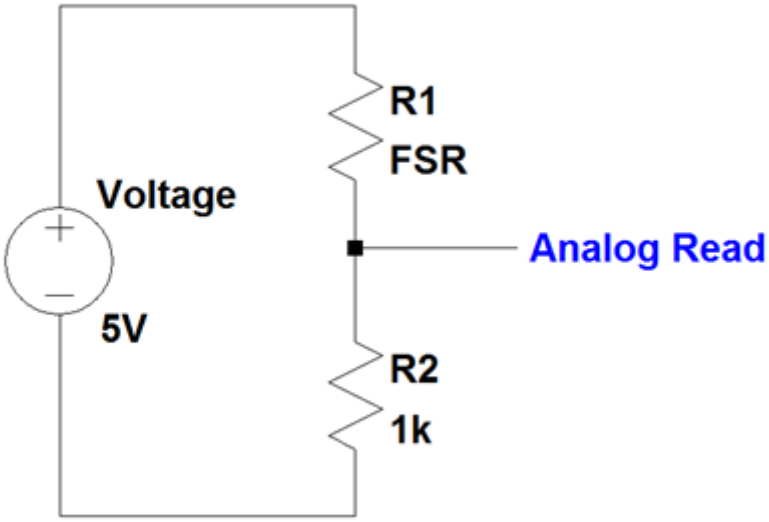

Figure 4: Circuit Diagram to measure FSR

A power of 5V will be used in this circuit. The pull up resistor represents the force sensitive resistor and a 1K resistor is the pull down resistor. Voltage will be read between the two resistors to find the change of voltage due to the force sensitive resistor.

When no pressure is applied to the FSR the total resistance is ideal infinite. This means that no current flows and the voltage across the 1K resistor equals zero. Resistance decreases as pressure is applied to the FSR, and current flows through the circuit creating a measurable voltage across the 1K resistor.

3.2 Proof-of-concept

We first placed ten force sensitive resistors along an insole as shown below.

Figure 5: 10 Force Sensitive Resistors on the Insole

All sensors were connected directly into the Arduino analog input using the circuit shown above in Figure 4. Data was collected and showed promising results for the pressure on a sensor through time. The amount of sensors needed to be increased for better data analysis, but using the circuit shown above, each individual sensor limits the sensor number to how many Arduino analog pins are available. To greatly increase the number of sensors a new circuit design must be implemented.

We expanded the amount of sensors we will use by implementing a matrix using both analog and digital pins on the Arduino. Figure 6 shows the case of a three by three matrix which equals a total of nine sensors.

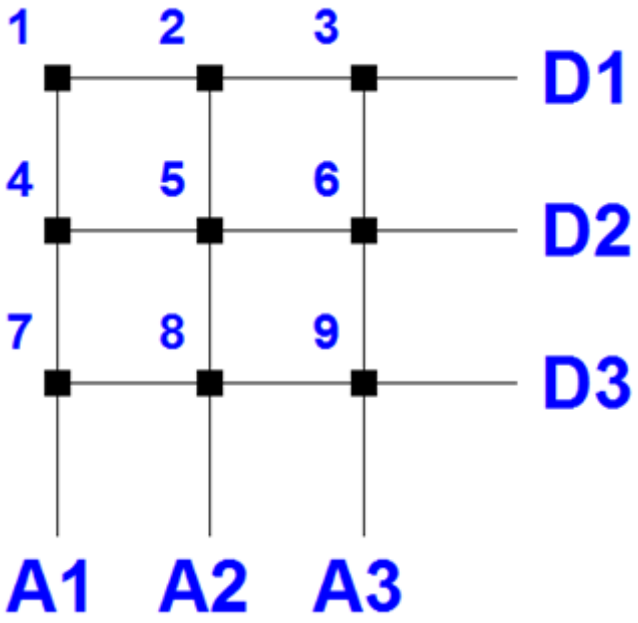

Figure 6: Analog and Digital Inputs

This matrix of sensors had to be proved before further tests, so the matrix was implemented into an insole in the same locations shown in Figure 5, excluding the center heel sensor because of the lower number of nine sensors. Just looking at the matrix above you can spot a serious flaw in that if D1,D2, and D3 are all powered and if sensors 1 and 7 are pressed, the value that A1 reads will make no sense because it will be some combination of two sensors. A multiplexer can fix this problem, however, because this is a wearable device, minimizing the size of the circuit is important for comfort, so this problem is solved in the Arduino code and will be explained in the MCU section. Non-insulated threaded wire was used for the proof of concept stage. The wire was chosen for it being the most flexible wire available, however because of it being non-insulated, the readings from the sensors had unwanted noise when no pressure is applied. For the prototyping stage and the rest of this project we now use 30 gauge insulated wire which reads clean values as well as being small enough to go across the insole without interfering with the user's comfort.

3.3 Prototype

The proof of concept stage of the project showed and confirmed that force sensitive resistor data can be used in pressure mapping of the insole and the amount of sensors used can be increased using a matrix layout. In the prototyping stage of the project, we expanded the amount of sensors to 64 implementing an eight by eight matrix. The schematic of the electrical circuit is shown below in Figure 7.

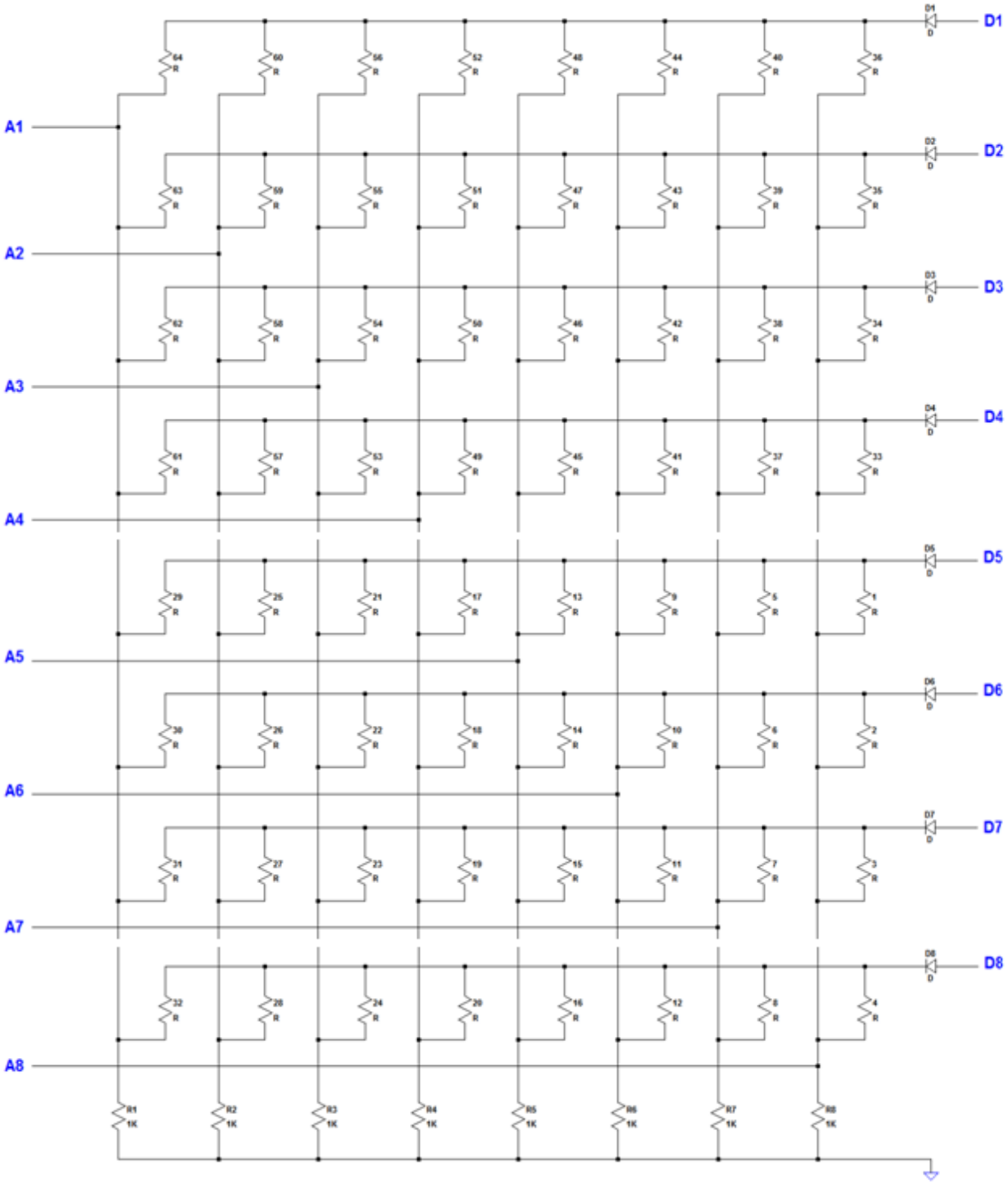

Figure 7: Electrical Circuit Schematic

The schematic above is for our 64 sensor matrix and each sensor is represented by a resistor labeled 1 through 64. The resistors R1, R2, ..., R8 have a value of 1K and are the pull down resistors for each digital output. Only one digital output will be powered at 5V and one period of time, so diodes are added to stop current flow to other digital pins which would be 0V and the same point in time; the details of the

digital pin power is explained in the MCU section. All circuit power is rated at 5V and comes from the digital pins of the Arduino.

The value of the 1K resistor was chosen because of power restrictions of the Arduino output pins. A digital output supplies power of 5V and a maximum of 0.04A. We are using a total of eight resistors in parallel at one period, and ignoring resistance due to the force sensitive resistor we can find the minimum resistance value from the equation:

$$R_{total} = \frac{V}{I}$$

Because we have a maximum power at V=5V and I=0.04A, we find that the minimum total resistance value must be equal to or greater than 125Ω. Now to find the total resistance we use the equation

$$R_{total} = \frac{R}{N}$$

where R is the value of each resistor and N is the number of resistors in parallel equaling 8. Solving the above equation we find that each resistor value must be equal to or greater than 1K each. Using this value for each resistor guarantees we will stay below the total power that an output can produce.

The schematic in Figure 7 was implemented into the insole and the sensor placement is shown below in Figure 8.

Figure 8: Mapped out Resistors on Insole

Sensor positions are spread or condensed in regions which are of higher important to understand the user's gait, such as condensed in the heel and ball of the foot and more spread in the large region of the arch.

3.4 Product

We have proven functionality of our final 64 sensor design in the prototype stage of the project, however, we must now implement the design into a shoe. To do this we must cover our circuit design in the sole using comfortable fabrics so that it is unnoticeable to the user. We must run the wire connections of the sensors up the ankle to the MCU and Bluetooth communication which will be located on the shoe or lower leg. Having the wires near the ankle will have the least amount of discomfort to the

user as we will use ribbon wire so that the wires stay flat up the user's shoe. It is crucial that the final product will match the comfort and flexibility of a normal insole, and because only wire will be placed in the shoe, comfort will be possible.

4 MCU and Bluetooth Communication

In this section, we will discuss the description, proof-of-concept, prototype, and product for the MCU and Bluetooth communication.

4.1 Description

Arduino was chosen as our MCU because of its power regulation of the 9V power supply, number of input and output pins, SPI connection for Bluetooth communication, and the ability to create an internal multiplexer within the code itself. Arduino MCU's come in many different sizes, which makes them ideal for our project because limiting the size of the device is crucial in a wearable device such as in this project.

To connect the data that the Arduino receives to the user's phone we will be using Bluetooth. The Bluetooth device connects to an Arduino through its SPI pinouts and will send data to an Android or IOS device. We will use a Bluetooth breakout called Bluefruit. Bluefruit uses Bluetooth low energy which will be important in the reduction of power loss for our device. Bluetooth was chosen as the wireless connection because the range of ten meters which will be more than enough distance to guarantee a connection from the Arduino to the user's phone, as well as the reliability of the connection.

4.2 Proof-of-concept

The first test which is explained in the previous section was to use ten sensors along the foot shown in Figure 5. Each sensor was connected to analog 1 through 10 pins on the Arduino mega. The Arduino mega code reads one analog value at a time and presents the pressure values with time in the serial monitor of the computer.

Next, we explored the matrix 3 by 3 circuit shown in Figure 6. Three wires were connected to separate digital outputs of the Arduino and the other three circuit wires were connected to separate analog inputs. The code is not as straight forward as the previous test because we cannot power multiple digital pins and expect accurate analog values. The Arduino code pulses each 5V digital pin separately and reads all analog pins for each digital pin. It is crucial that only one digital pin is on in an instance, and then turns off before the next digital pin turns on. Below shows a small portion of the code:

```
digitalWrite(30, HIGH);
FSR1 = analogRead(a1);
FSR2 = analogRead(a2);
FSR3 = analogRead(a3);
digitalWrite(30, LOW);
digitalWrite(31, HIGH);
FSR4 = analogRead(a1);
FSR5 = analogRead(a2);
FSR6 = analogRead(a3);
```

```
digitalWrite(31, LOW);
digitalWrite(32, HIGH);
FSR7 = analogRead(a1);
FSR8 = analogRead(a2);
FSR9 = analogRead(a3);
digitalWrite(32, LOW);
```

We confirmed Bluetooth communication between the Arduino and phone using a Bluetooth 4.0 breakout. The Bluetooth is powered through the 5V output pin on the Arduino as well as connect with SPI communication pins. The Bluetooth successfully transferred the pressure data to the phone.

4.3 Prototype

The proof of concept stage of the project showed and confirmed communication between the Arduino and phone via Bluetooth. We also showed that pressure data could be read through the analog input pins of the Arduino.

In the prototyping stage, we wanted to expand the amount of force sensitive resistor to 64 as explained in Section 3.3. This involves expanding the three by three matrix as in the proof of concept stage to an eight by eight matrix giving us a total of 64 sensors. Eight wires were connect to separate digital outputs of the Arduino which are labeled D1, D2, ..., D8 in the schematic shown in Figure 6. The other eight circuit wires were connected to separate analog inputs as labeled A1, A2, ..., A8 which are also shown in Figure 6. Just as in the proof of concept stage, the Arduino code pulses each 5V digital pin separately and reads all analog pins for each digital pin. The Arduino code for the 64 sensor layout is similar to the code shown in the proof of concept stage but is now reading for all 64 force sensitive resistor values.

4.4 Product

For the product stage of this project, we moved from using the Arduino mega to using the much smaller Arduino nano. The data shown in Figure 5 in Section 3.3 was collected from the Arduino through a hard wire connection to a computer. We have confirmed a Bluetooth connection between the Arduino and the phone, however, we have not yet implemented the Bluetooth to send real-time sensors data. Our product will be wireless which means future work will be implementing the sensors and Arduino to the Bluetooth so real-time data can be transferred to the phone. The Arduino code for reading analog values from the sensors is well defined as well as the code for connection to the low energy Bluetooth device. Our product will combine the code so that the Arduino can read sensor values and connect to the Bluetooth device simultaneously.

Our product will be worn on the shoe or lower leg of the user, so the size of the Arduino and Bluetooth is crucial for user comfort. We will be using the Arduino Nano and Bluefruit breakout in our product, both of which were chosen because of their small size. Our final product will fit into a box with a width of 15cm, height of 10cm, and depth of 10cm as shown below in Figure 9.

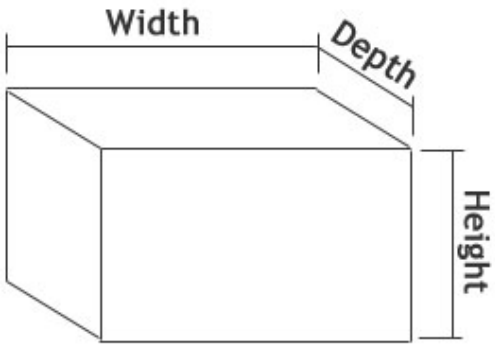

Figure 9: Maximum Size of Product

Ideally, our product will weigh less than 200g so that it can be unnoticed by the user. A 9V battery will be enclosed within the box and will power the total device, and because of the large size of the battery, the bulk of the weight from our device will be from the battery.

5 Data Processing

In this section, we will discuss the description, proof-of-concept, prototype, and product for data processing.

5.1 Description

Data collection occurred for the insole design at the three stages: proof-of-concept, prototype, and product (see section 3 for more details on insole design). Each data file contains gait information for the respective number of sensors over six steps, at a controlled cadence. Tests include normal gait as well as simulated gait pathologies, i.e. over pronation, and over supination. Table 1 summarizes the data collected for the proof-of-concept and prototype designs, and the plan for collection using the product design. Sections 5.2, 5.3, and 5.4 describe the analysis and processing performed on this data as well as some preliminary results. All graphs, step averaging, and analysis in sections 5.2 and 5.3 were performed using MATLAB software.

Table 1: Data collected for different designs described in section 3

Stages	Gait type	Number of Steps	Number of Sensors	Test Number
Proof-of-concept	Normal	6	9	1
	Over-Pronation	6	9	2
	Over-Supination	6	9	3
Prototype	Normal	6	64	4
	Over-Pronation	6	64	5
	Over-Supination	6	64	6
Product	Normal	6	64	7
	Over-Pronation	6	64	8
	Over-Supination	6	64	9
	Low Arch	6	64	10
	Medium Arch	6	64	11
	High Arch	6	64	12

5.2 Proof-of-concept

Figure 10 illustrates one-step of a normal gait pressure pattern acquired using the proof-of-concept nine-sensor design described above. The graph, generated using MATLAB, shows the relative timings of the gait phases of the stance period, and the differences in pressure values across the cycle. From Figure 10 we can derive the relative timings of gait mechanics. The first pressure change occurs in the lateral and medial calcaneus, or heel region. This pattern represents the initial contact stage of the gait cycle when the heel makes contact with the ground. It is evident that the medial arch pressure values remain relatively low, which shows that there is little pressure on the raised inner part of the foot. We can see from the graph that the last sensors to reach their maximum values are the first and fifth phalanges, which are the toes. This final pressure peak represents the terminal stance of the gait cycle, when the subject's body weight shifts to the toes of the foot and continues to propel forward. The trends derived from the first stage of data acquisition and processing using the proof-of-concept design follow expected gait mechanics. However, the data in Figure 10 shows various problems with our initial insole design, such as low signal-to-noise ratio.

Noise pattern is obvious in the graph at around 1500ms when there is no pressure on the insole but pressure values are still detected. These values are significant in amplitude relative to the rest of the data and variable between sensors. One cause of this affect is the use of non-insulated wires in the proof-of-concept design. The distance from the voltage source, as well as other external factors, affects the resistance in non-insulated wires and creates fluctuations in the data. In Figure 10 it is also evident that the data was under sampled, which led to large jumps between data points with steep slopes. Changes were made to the proof-of-concept design to address the issues affecting the signal-to-noise ratio of our design.

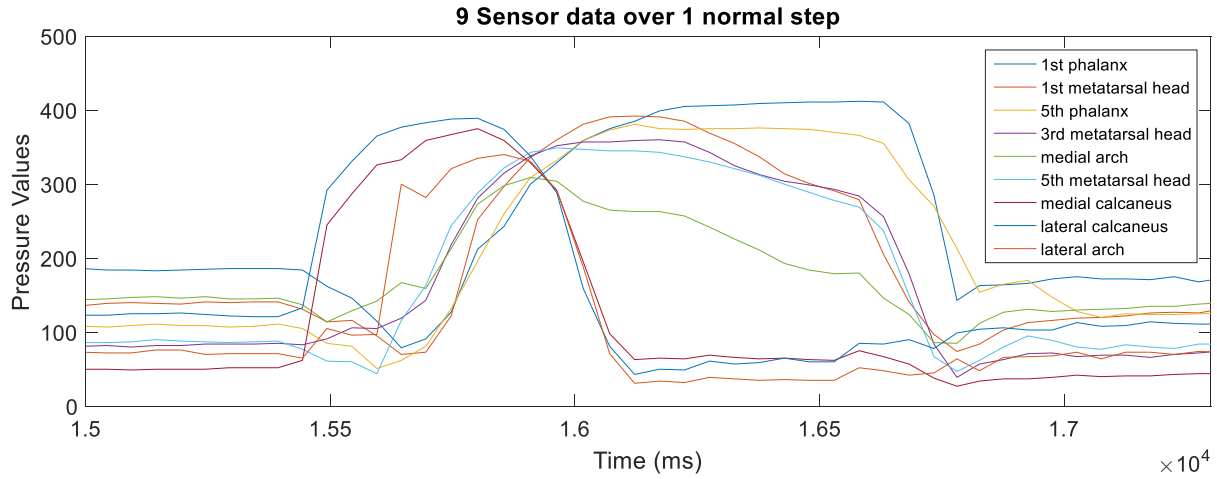

Figure 10: Plot of nine sensor data over one-step

5.3 Prototype

Figure 11 represents typical pressure data acquired over one-step using the prototype design described above over one gait cycle, for an individual with no gait abnormalities. Each of the trend lines in the graph represents the pressure over time detected by each sensor. The four colors represent different masks applied to the data. Red color represents the medial, lateral, and center calcaneus regions, or areas of the heel. Green data pertains to the medial, lateral and central arch regions of the foot. The

blue lines contain the sensor data from the metatarsal heads, or the ball of the foot. Lastly, yellow represents data collected from the phalanges, or toes. In comparison to Figure 10, Figure 11 has a much higher signal to noise ratio when pressure is not present on the sensors. This is evident in the figure around 1300ms, when no pressure is applied.

Following the pressure response of the sensors over one normal step, we can identify the gait stages in the stance period. Initial contact is evident in the early peaks of the red lines, loading response occurs approximately where the red, blue, and green trend lines overlap, mid stance occurs when the pressure increases in the blue lines, and terminal stance occurs when the yellow lines peak. Similar to Figure 10, the arch pressure values remain relatively low. We can see from the red trend lines below that during the initial contact phase, areas of the heel peak immediately remain high, only to decrease to zero with the other heel data. From this pattern, and similar patterns with the other gait stages, we can derive the length of the phases, the relative peak pressures, and the center of pressure. Figure 11 demonstrates that, using our prototype design, we can obtain expected pressure data for a gait cycle that has a high signal-to-noise ratio and sufficiently high sample rate to be able to extract the length, relative amplitudes and timings of specific phases.

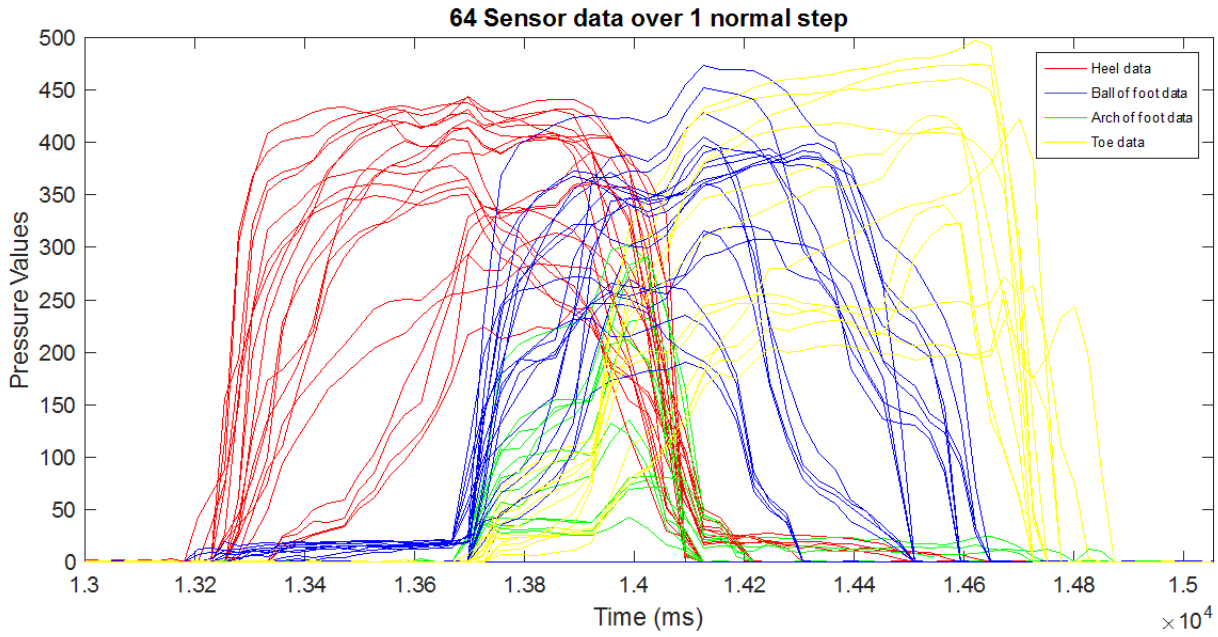

Figure 11: Plot of 64 sensor data over one step for 4 masks: heel (red), ball of foot (blue), arch of foot (green), and toes (yellow)

In addition to overall patterns, the amount of sensors in the prototype design also allows us to investigate gait pathologies more accurately. Data used in the following analysis illustrates the contrast between normal walking as well as simulated over-pronation and over-supination. The red trend lines in Figure 11 represent the average of the pressure data of all the medial heel sensors, averaged over six steps for the three different walking patterns. The green trend lines in illustrate the average of the pressure data of all the lateral heel sensors, averaged over six steps for the three different walking patterns. Similarly, Figure 12, Figure 13, and Figure 14 show the medial and lateral pressure trends for the arch, ball, and toe data respectively.

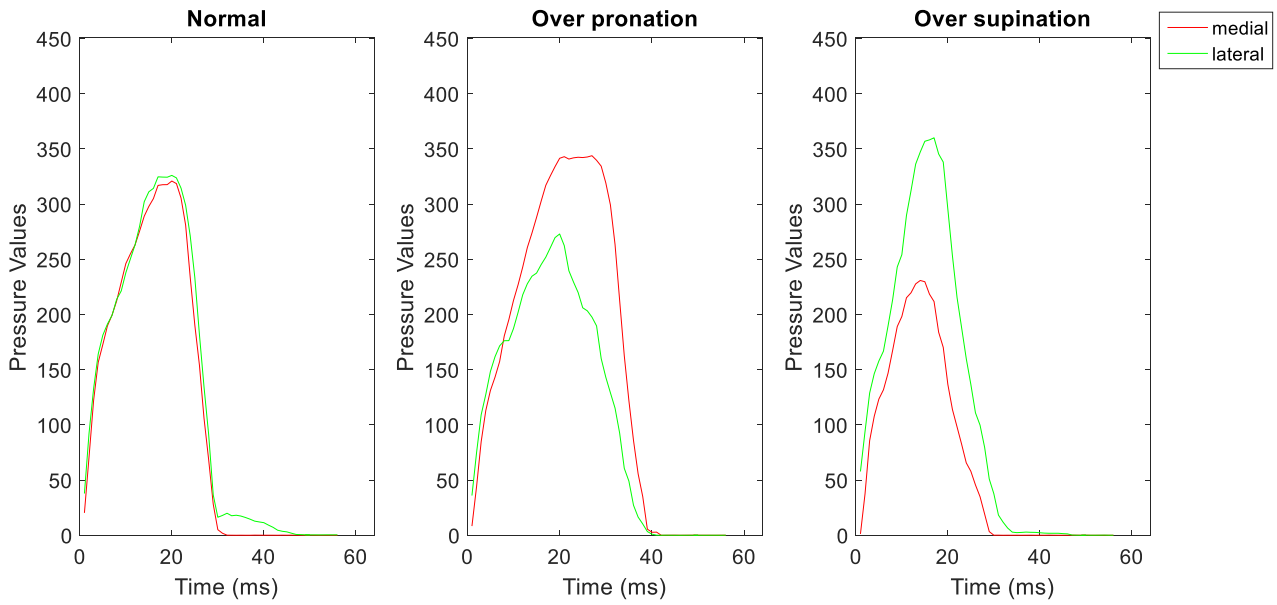

Figure 12: Average of medial and lateral pressure sensor data of the heel

From the normal plot of Figure 12, we can see that on average the subject applies even pressure over the heel during the initial phase of the walking cycle. In contrast, the over-pronation plot of Figure illustrates that if the subject is over-pronating, the pressure applied to the medial area of the heel is much higher than the lateral. The opposite pattern exists for a subject who is supinating, as shown in the supination plot of Figure 12. Similar patterns exist in the arch and ball data, as shown in Figure 13 and Figure 14 respectively.

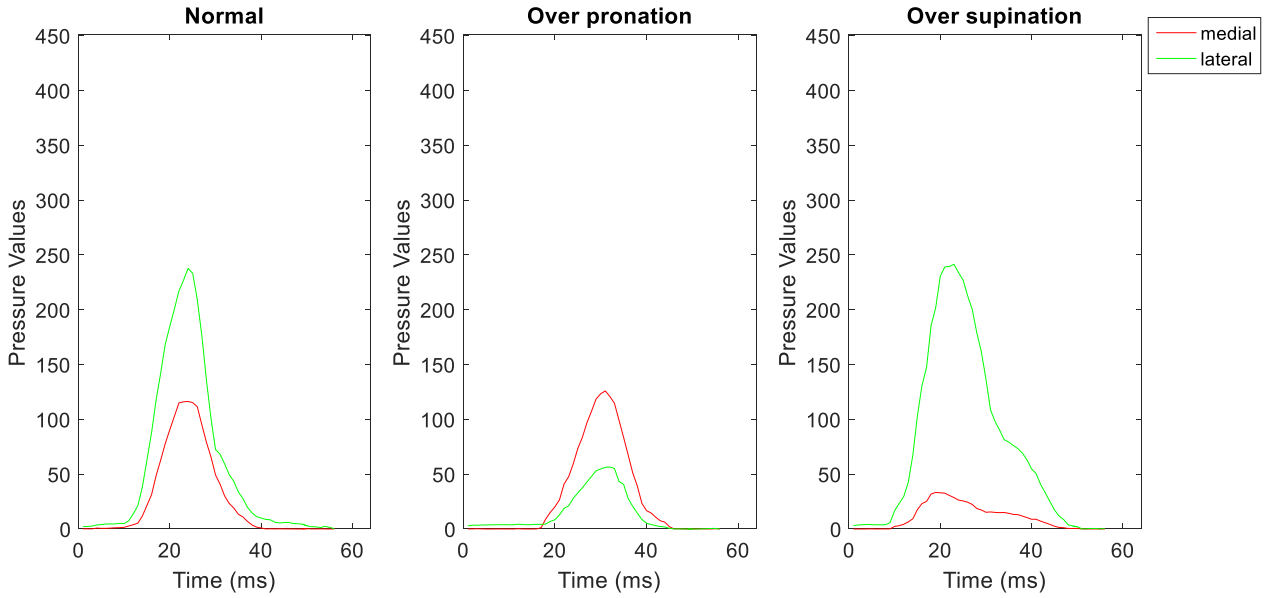

Figure 13: Average of medial and lateral pressure sensor data of the arch

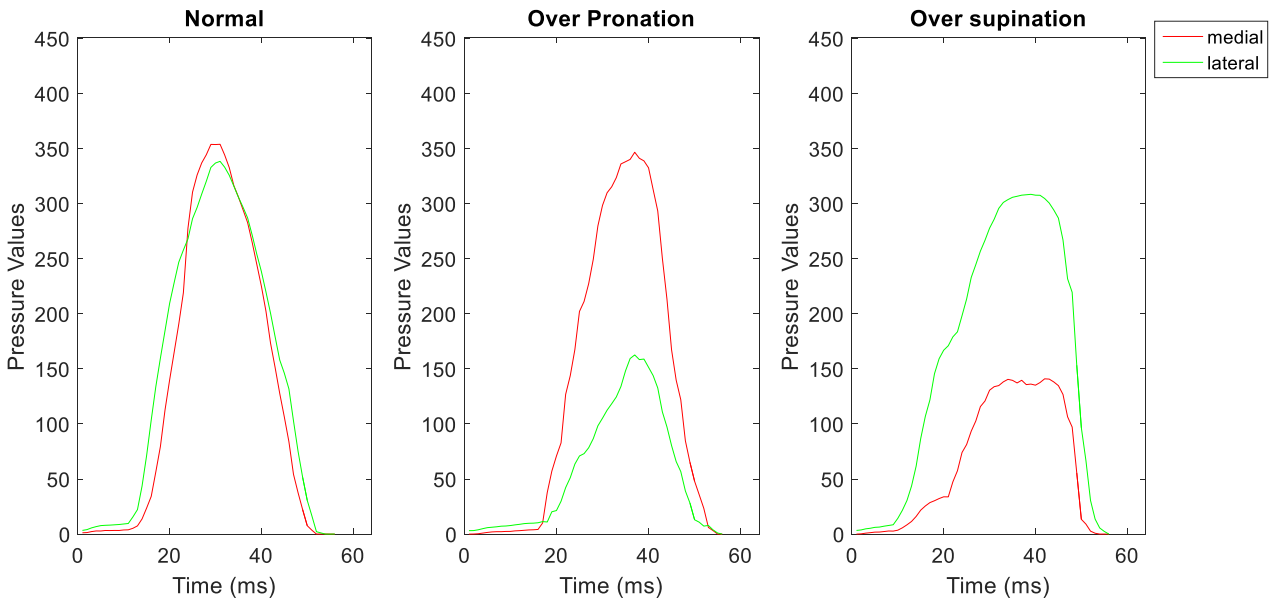

Figure 14: Average of medial and lateral pressure sensor data of the ball

Figure 15 represents the average of the pressure data for all the medial and lateral toe sensors, averaged over six steps for the three different walking patterns. The patterns aforementioned are not as evident in Figure 15 as the previous images. One reason for this discrepancy could be due to the fact that the individual is using the lateral or medial portion of the ball to propel forward instead of the toe regions. Another reason could be that the manner in which we simulated over-pronation and supination allowed the subject to place the swinging limb down sooner than the step was completed. Further testing is required to investigate the true cause of this phenomenon.

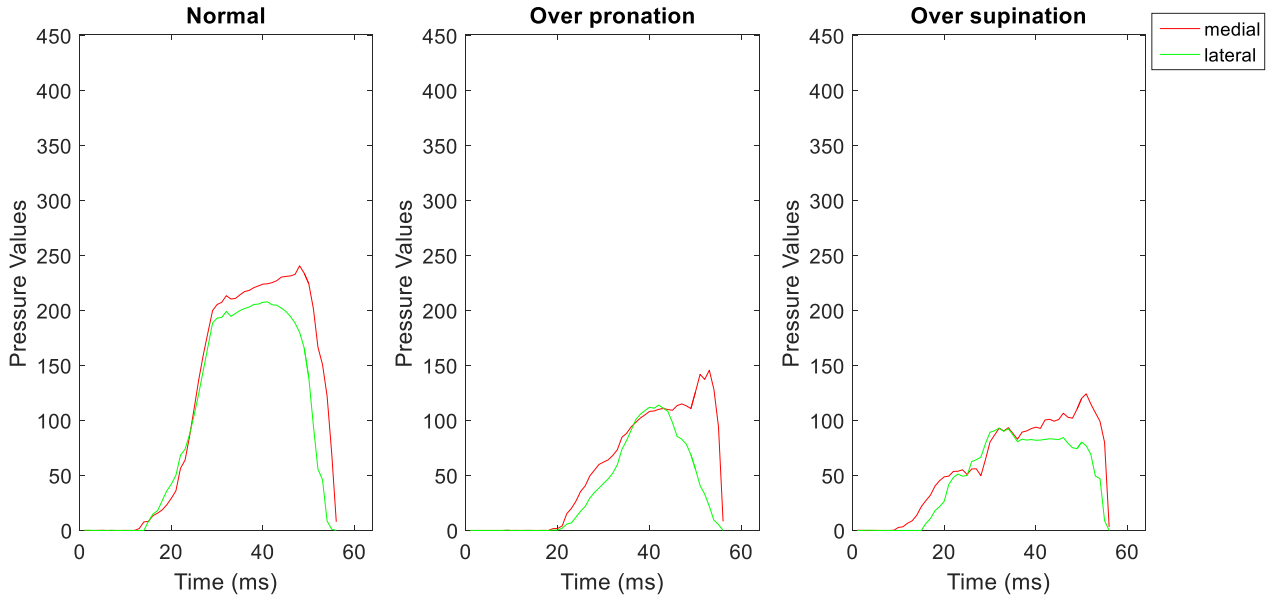

Figure 15: Average of medial and lateral pressure sensor data of the toes

Figure 16 represents eight instances of the real-time pressure data generated over one normal step cycle on the insole. The data in Figure 16, 17 and 18 represents the average of the pressure data over six individual steps acquired while the individual walked normally, over-pronated, and over-supinated respectively. The colors on this image represent intensity values; zero symbolizes minimum pressure and one denotes maximum pressure. The first insole image in Figure 16 is the initial heel strike and the last insole image is the final toe off. Figure 17 and Figure 18 show the differences between normal gait and over-pronation or over-supination. From Figure 17, we can see that more pressure is applied to the medial part of the foot, indicating over-pronation. In contrast, Figure 18 shows that there exists more pressure on the lateral part of the foot, indicating over-supination.

Figure 16: Real-time pressure data generated from the average of six normal steps

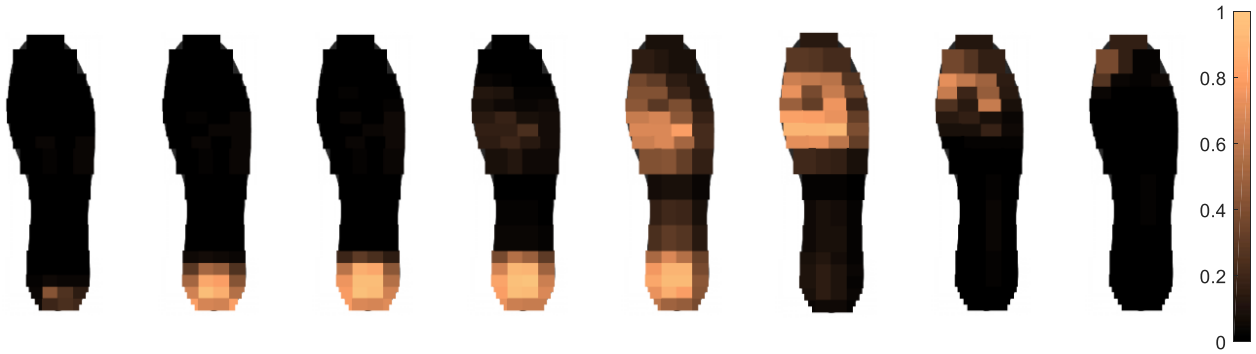

Figure 17: Real-time pressure data generated from the average of six over pronated steps

Figure 18: Real-time pressure data generated from the average of six over supinated steps

Figure 19 illustrates the peak values that each of the sensors reached, on average, over six steps of normal walking, over-pronation, and over-supination. From this figure, we can see that during normal walking, a subject evenly applies pressure over the medial and lateral parts of the foot, with the exception of the arch as previously discussed. When pronating, less pressure is applied on the lateral portions of the foot while the opposite is true during supination.

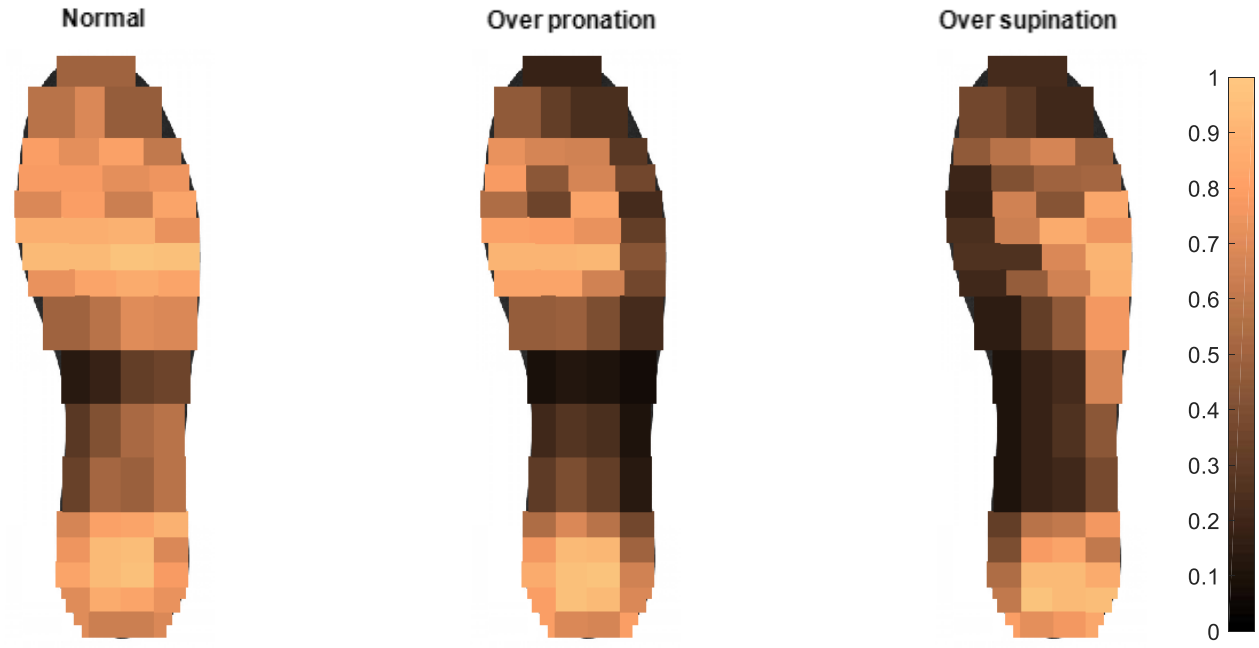

Figure 19: Peaks of sensor data averaged over six steps

Pseudocode for the data processing is provided in the Appendix section of document.

5.4 Product

Using the product design outlined above, wireless transfer of the data from the Arduino to the mobile phone allows for real-time processing and analysis. The mobile application is developed in Java to be used on the Android network. The MATLAB algorithms are converted to Java for the final design and pseudocode to measure the center of pressure is provided in the Appendix. A real-time pedobarograph plot of the user's gait as exemplified in Figure 16, as well as a static average, or peak plot, as displayed in Figure 19, are the feedback given to the user through the mobile application.

6 Software Design

The Pods Mobile application will be the main point of interaction for the end user to the system. It is here that the user can interact with the Pods and analyze the data. In the background, the application will do all the heavy lifting, getting data from the Pods and processing so that it can be displayed to the user in real-time and afterwards.

The application will be developed for the Android platform and therefore will be primarily developed in Java and XML. The dataflow will be as follows:

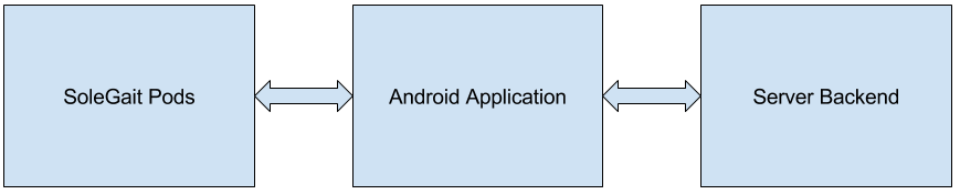

Figure 20: Dataflow Diagram

As per Android recommended guidelines, the application will be developed using the MVC (Mode-View-Controller) model and strictly follow the Object Oriented paradigm which is inherited from Java. This is especially beneficial to an application that has many moving parts and needs to interface with the world beyond the device on which it is running.

The following is an overview of the objects and their interactions:

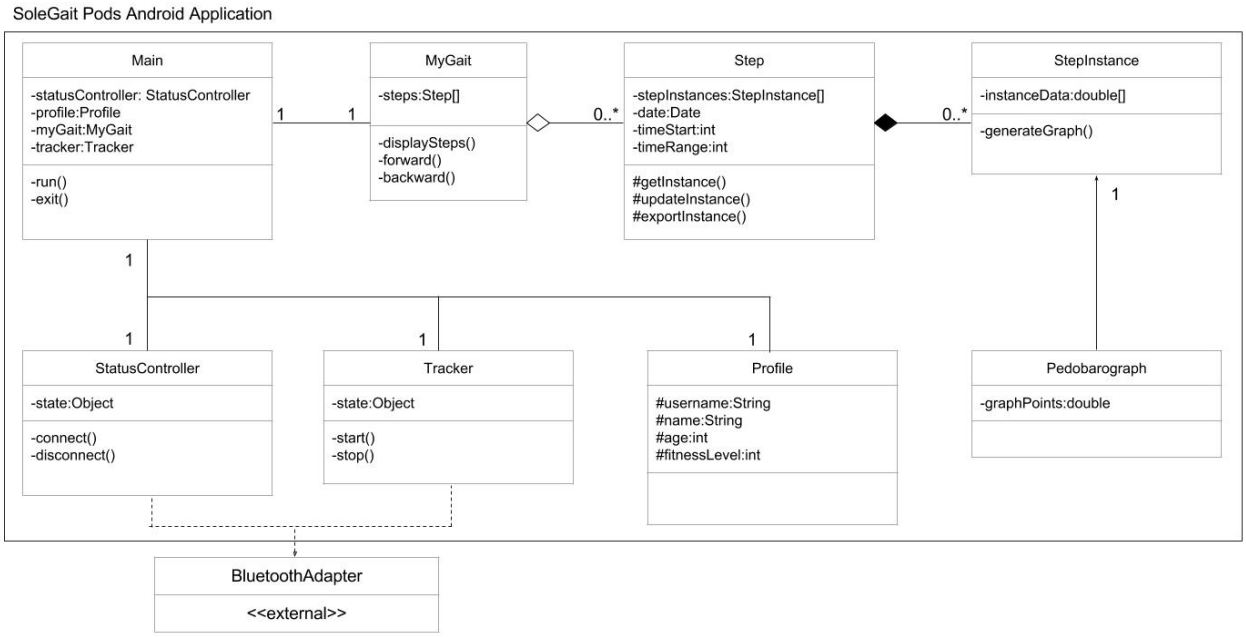

Figure 21: UML Diagram

On the device side, the data store will be SQLite, the most reliable way to store large amounts of data for the Android Runtime (ART). On the server backend, SQLite will also be used to prevent any incompatibilities that may arise between different database implementations. The following is an Entity-Relationship Diagram modeling the database structure:

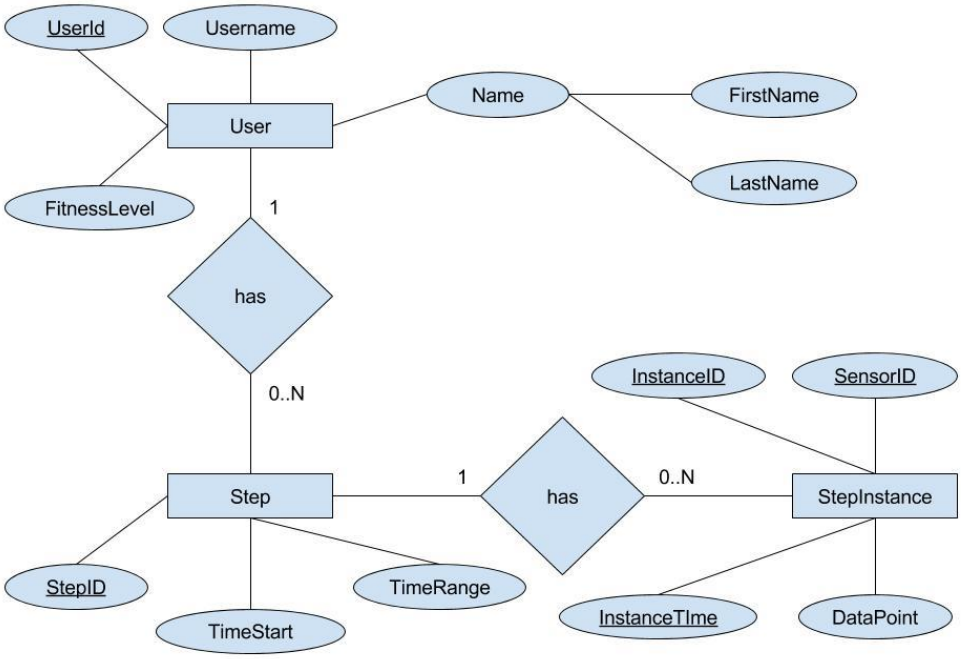

Figure 22: ER Diagram

6.1 Splash Screen

The Splash Screen displays relevant and concise application information, company information and disclaimers while the application loads. The following figure displays the wireframe:

Figure 23: Wireframe of Splash Screen

6.2 Status Screen

The Status Screen serves as the landing for the user. It is here that the status of the Pods can be viewed and changed. The possible states that will be displayed are disconnected, connected and tracking, which will be contextual. The following is the wireframe:

Figure 24: Status Screen with Connect

Figure 25: Status Screen with Disconnect

6.3 Tracking Screen

To allow the user to collect data and view what is happening with the pods in real time, there is the Tracking Screen. The following is the wireframe:

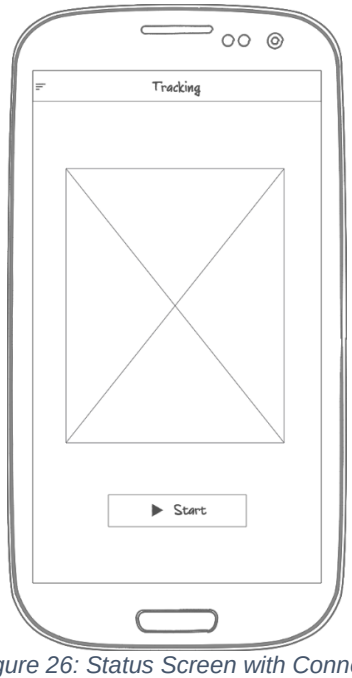

Figure 26: Status Screen with Connect

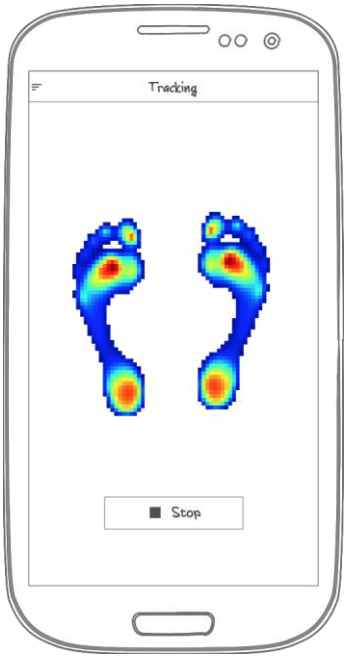

Figure 27: Tracking Screen with Stop

6.4 MyGait Screen

To provide a personalized experience for the user, the MyGait Screen will be populated with useful statistics based on the user's profile. The following is the wireframe:

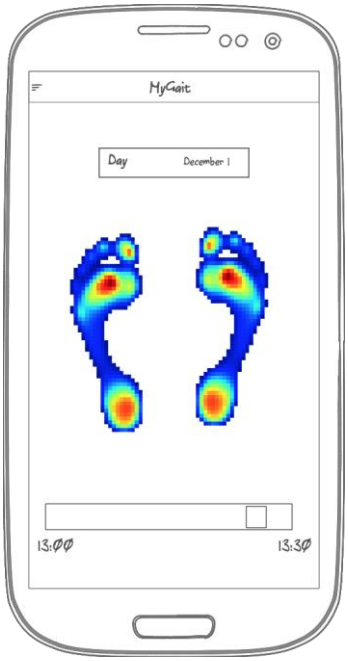

Figure 28: Wireframe of MyGait Screen

6.5 Profile Screen

The profile screen enables the user to enter personal data and preferences to better personalize their Pods experience. The following is the wireframe:

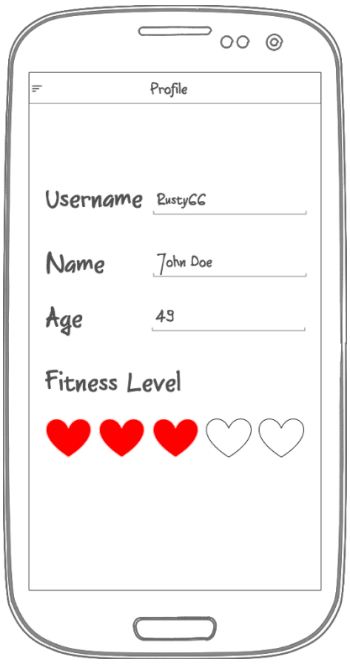

Figure 29: Wireframe of ProfileScreen

6.6 App Menu Drawer

At any location in the app, the user will be able to open the Menu Drawer and access the other parts of the application. The following is the wireframe:

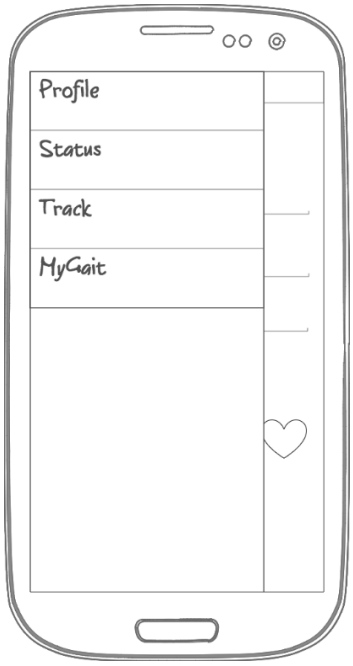

Figure 30: Wireframe of App Menu Drawer

6.7 Application Backend

The application backend will mainly be used as a data store to prevent large amounts of data from filling up the end users' device. Having all the data collected on one server has other benefits such as its use for research. The backend could be extended with a set of APIs (Application Program Interfaces) that can provide anonymous data to a researcher.

The backend will be written primarily in Python using the CherryPy Web Framework and will use an SQLite database that will preserve the same structure as the mobile device side database. The Android application will communicate with the API through HTTP GET and HTTP POST requests. Any data that will be transferred will be in JSON (JavaScript Object Notation) format.

7 Test Plan

The following test plan is created to ensure our final product meets the requirements of our design specifications. We will be testing our product with various types of testing through different stages. First, we will conduct tests on our individual components in order to isolate ideal functionality of each part of the system before being assembled together. Second, we will discuss the data processing unit and testing a user's extreme cases of over-pronating and supinating. Finally, we will perform different tests of the software application design and the overall system functionality.

7.1 Unit testing

This part of testing will test each of our components separately to insure each component works as expected.

7.1.1 FSR

There are a lot of areas that need to be tested with the FSR. First, connecting the sensor to a multimeter and apply a force to the FSR. The expected outcome is that there is change in resistance, which causes a change of voltage with pressure. Testing each of the 64 sensors individually is required to ensure that there are no errors or faulty sensors when force is applied. Another area to test is to ensure a heavy weight can be applied to the FSR, while still producing a required output. Placing the sensor under the heel and applying the maximum amount of pressure, the FSR can still detect the force applied by the body weight.

7.1.2 MCU

In order to test the microcontroller, first, connecting the microcontroller to a power supply and observe that the lights are responding to ensure the MCU powers on correctly. Next, we tested that each analog and digital pin functions correctly. Using the sensors to connect to each pin individually and observe the data to ensure that the pins are working properly. Additionally, testing that the microcontroller will be integrated with the Bluetooth chip correctly in order to collect the required data.

7.1.3 Battery

Testing the battery will be done in a couple of tests. First, to power on the device, connecting the battery to the microcontroller in order for the MCU to function without the use of an external power source. To test how long the battery will last, using the system for at least 1-2 hours without an external power source will be desired.

7.1.4 Bluetooth Chip

Testing that there is a valid connection with the mobile application and the microcontroller is required. A couple basic commands were sent from the cell phone to the microcontroller in order to test that the Bluetooth chip functioned properly.

7.2 Data Processing

7.2.1 Pedobarograph

The data processing unit will be tested and verified using a variety of data sets. Preliminary data sets will consist of a normal subject using the insole. Data processing as described in section 5.3 of this document will be completed to ensure that no abnormalities exist in the pedobarograph or the code. Obtained pedobarograph data will also be compared with literature, to ensure that similar products have shown to provide the same pressure plots. Once the pedobarograph is verified and approved, we will continue into the next phase of testing.

The following phase consists of the user providing two types of 'extreme' data. For the first type of extreme, the user will perform ankle eversion during walking in order to over supinate. The next extreme, the user will invert their foot as much as possible so as to over pronate. The pedobarographs of these two data sets will be compared against their respective literary equivalents.

Finally, once the Data Processing Unit has been extensively tested using normal and simulated data, we will collect various test subjects who have no affiliation with Solegait. These subjects will be instructed to wear the insole and walk at a comfortable pace for at least six strides. This new data will then be verified and approved using the same method mentioned above. Following the experiment, the users

will be asked to provide their opinion regarding the wearability and comfortability of the device. This will provide us with more feedback regarding the physical design, and usability of our device.

7.2.2 Center of Pressure

After the pedobarograph data is depicted, the data processing unit will then output a plot displaying the average center of pressure (COP) over the plantar surface for one data acquisition session. We will test the COP function using the same data sets produced by our users outlined in the previous section. The COP plot will be verified by manually calculating the COP for various regions of the foot, and comparing these values with the plots. A typical COP plot is depicted within the following figure.

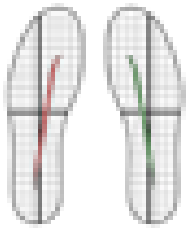

Figure 31: Center of Pressure plot

7.3 Application

Testing is considered one of the most important parts of software engineering and the Solegait Pods Application will be rigorously tested at all stages using the methodologies that have become standard in the industry.

Following a bottom up approach, at the very lowest level, white-box unit testing will be employed to ensure correct and predictable algorithm behavior. More focus will be placed on dynamic and data flow analysis, as these are what are of main importance for the application that interfaces with a Bluetooth device. Code reviews will be limited to certain parts of the application.

On the upper level of testing, integration and system testing will be performed. This will be performed in black-box style so as to examine proper interface interaction and end-to-end reliability. As the application is developed the usual smoke and sanity tests will be continuously done on the application to detect any gating defects or regressions.

7.4 System Testing

At the end of the individual testing, data processing, and application testing, it is expected that all testing produces expected results and can now test the whole system. The first thing we need to test is the overall comfortability of the insole. The insole is worn by all team members as well as other individuals to identify the right comfort level.

We will then connect the FSR sensors with the microcontroller and test the accuracy of the data. The data will be analyzed from the MCU to ensure that it was successfully logged and saved, and that it can be read properly. Furthermore, the MCU will extract the data from the sensors and compares it with the individual testing as described in 7.1.1.

The Bluetooth chip will then be added, in conjunction with the mobile app, and further testing with commands from the Bluetooth will be performed. The mobile app will then be integrated and the GUI

will be tested to make sure it is user friendly and self-explanatory. Finally, the data in the GUI will be compared with the extracted data from the MCU to ensure the data is depicted correctly.

8 Conclusion

The design specifications outlined within this document will be adhered to as strongly as possible during the development of the Solegait Pods device. Moreover, the design of each component follows the requirements outlined within *Functional Specification for an Assistive Rehabilitation Device Named: Pods* [2] to ensure that Solegait Pods is a safe, and reliable device.

Solegait is a team of engineers who are highly dedicated to the development of an assistive device to aid a patient's rehabilitation process and correct their foot dynamics. The Solegait Pods is a low cost solution to more accurately define imbalances in the user's walking patterns, and as a result, inhibit any chronic injuries in athletes, or bone and muscle degradation of older adults.

The Solegait Pods is designed to provide therapists, physicians, and everyday users a cost effective solution to solve their gait related injuries. Furthermore, Solegait is a company that supports the advancement of medical research; therefore, with the added benefit of a cloud server, researchers have the opportunity to gather open source data for their own benefits.

References

[1] V. AJ, 'The optical pedobarograph. - PubMed - NCBI', *Ncbi.nlm.nih.gov*, 2015. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/8364849. [Accessed: 13- Nov- 2015].

[2]J. Riad, S. Coleman, J. Henley and F. Miller, 'Reliability of pediobarographs for paediatric foot deformity', *J Child Orthop*, vol. 1, no. 5, pp. 307-312, 2007.

[3] Solegait, “Functional Specification for an Assistive Rehabilitation Device Named: Pods”, Simon Fraser University, Burnaby, BC, Canada, October 2015.

[4] S. User, 'Physio & Pro Sports - Moticon', *Moticon.de*, 2015. [Online]. Available: http://www.moticon.de/products/physio-pro-sports. [Accessed: 13- Nov- 2015].

9 Appendix

9.1 Pseudocode for prototype stage data processing

1. Import sensor data into matrix in MATLAB
2. Split data into four masks for the heel, ball, arch, and toes
3. Locate the points in time at which the subject begins and ends a step
    a. Set a threshold for determining the location of heel touch
        i. Heel sensor data > 10 pressure units
    b. Set a threshold for determining the locations of toe off
        i. Time of toe off > time of heel touch
        ii. All sensor data < 50 pressure units
4. Assign a sensor value to every pixel value of insole image to display real time plot
5. Average sensor values over a number of steps and display static insole image

9.2 Pseudocode for calculating center of pressure in product stage of data processing

6. For each row of sensor data

    COP = [sum from
    0:n](sensor(0)*loc(0)+sensor(1)*loc(1)+...+sensor(n)*loc(n))/(sensor(0)+sensor(1)+…+sensor(n)]

    End

7. Plot COP over entire insole

OUTPUT: