Design Report

Smart Sole

Version 2.0 • May 10, 2017

Nick Beezhold,
Garrett Schliebe,
Andrew Twining,
Modeste Niragire

Abstract

The Smart Sole team consists of a group of four engineers pursuing a Bachelor of Science in Engineering with an Electrical and Computer Concentration at Calvin College. They are expected to graduate in the Spring of 2017. The Smart Sole team originally set out to create a fitness tracking device that could be entirely powered by energy generated by piezoelectrics. That original goal was determined to be unfeasible after preliminary piezoelectric testing, so the scope of the product was modified. The team then decided to focus on making a fitness tracker that is more accurate and precise than comparable competition. This goal will be accomplished by integrating a piezoelectric comparator and accelerometer into a standard running shoe. Communications to a mobile application will be done via Bluetooth. This will allow the exporting and storage of user data for easy viewing and access. The application will also allow the user to track their exercises. The team will make sure that all features are implemented while preserving the comfort and aesthetics of the shoe. This document provides a proof of feasibility of the project as well as a review of the final prototype with test results. It also outlines the requirements, decision methodology, and testing methods employed by the Smart Sole team.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Initial inspiration for team Smart Sole was found by observing a loss of energy in athletic activities. Using the global push for capturing renewable energy as a lens, Team Smart Sole saw that activities such as recreational sports (Basketball, Soccer, Tennis etc) and running as having a great amount of mechanical energy underutilized. There is a lack of formal processes put in place to recapture this expended energy. Team Smart Sole decided to develop a method that would capture that energy and put it to use. The team decided to focus on running as the primary athletic activity examined. In addition to the harnessing and storing of energy, the team saw the potential to use the energy to power sensors that could be used for analysis to replace or augment analysis given by activity trackers.

The Smart Sole team performed a study on the feasibility of their concept and found that energy recaptured through the currently available piezoelectric technology was not enough to adequately power the circuitry they hoped to implement. The team adjusted the scope of their project to include two separate insole circuits. The first circuit would be a step counter that tracks steps through piezoelectrics and multiple feedback methods. This circuit also includes a Bluetooth Low Energy (Bluetooth LE or BLE) processor that sends recorded data to a mobile application. The second insole circuit would be a energy recapture circuit that is used to charge a spare battery that can replace the battery in the first circuit. Using these two separate circuits, the Smart Sole team was able to keep their original goal of sustainability.

The Smart Sole Team consists of Nicholas Beezhold, Modeste Niragire, Garrett Schliebe, and Andrew Twining. They are all Senior electrical and computer engineering students at Calvin College. The team members all share a passion for innovation and being on the cutting edge of technology. They all will graduate from Calvin College in the Spring of 2017 with a Bachelor of Science in Engineering, Electrical and Computer Concentration.

The Smart Sole project is part of the senior design course offered by Calvin College to engineering seniors. This is a required course that is intended to offer students the opportunity to work on an extended project that will challenge their abilities and demonstrate how a project may develop and function in an industry working environment. This involves researching, designing, prototyping, and testing as well as developing a potential business plan and marketing strategy for the product. The course also encourages students to develop their project management skills as the extended nature of the project makes planning and accountability a necessity. The course is offered over two semesters as ENGR 339 and ENGR 340. It is ultimately concluded with the Senior Design Banquet where the final project is presented.

Calvin College is “a top-ranked liberal arts college in Grand Rapids, Michigan that prepares students to be Christ’s agents of renewal in the world”.[1] Calvin College was founded in 1876 as a school of ministry training for the Christian Reformed Church. It has now grown to be one of the top liberal arts colleges in the Midwest, with over 4,000 students. Calvin College believes in integrating the Christian faith with learning to equip students with the skills necessary to live as agents of renewal in the world.

[1] "Calvin College | Grand Rapids, Michigan." 2015. 22 Apr. 2017 <https://calvin.edu/>

2 Project Description and Requirements

2.1 Project Description

The Smart Sole team formed around a desire to recapture energy lost through athletic activities and use this energy to power a device that is useful to the athlete. After exploring many options, including pressure sensing soccer balls and energy harvesting basketball courts, the team settled on the activity of walking and running. Walking is an activity that most people do every day. Providing a product that is incorporated into daily life will increase the potential customer base and encourage a large portion of society to practice healthy fitness habits. The final proposed design by the Smart Sole team incorporates piezoelectrics and other additional hardware into a running shoe. This product will be able to be used as a fitness tracker that is comparable to other similar products on the marketplace.

Piezoelectrics are materials that convert mechanical pressure into electrical energy. By implementing piezoelectrics into the sole of running shoes, the user can capture energy with each step they take. Rectifying the voltage and storing it in a battery or capacitor provides power that is necessary to power additional on-board sensors. The data that is acquired by the sensors will then be sent to a mobile application that will be able to store and analyze the data. The piezoelectrics can also be used to track steps as they will produce a voltage after each step a user takes. Through the use of piezoelectrics and other current step tracking technologies, the Smart Sole team believes that they can produce a fitness tracker that is more accurate and precise than a device that uses an accelerometer alone.

A mobile application will be created to allow a user to interface with the hardware. The mobile application should be able to export data from the in-shoe hardware via Bluetooth. The application will organize and store the data in a way that is easy for the user to understand. The application will be able to track addition exercises as well as running data. This will allow the user to store all of their data for any fitness activity in a single location, making it easier for them to track and review their overall progress. The scope of the application is easily adjustable, allowing the Smart Sole team to add new and interesting features as time permits. A depiction of the interaction between the hardware and software can be seen in Figure 1.

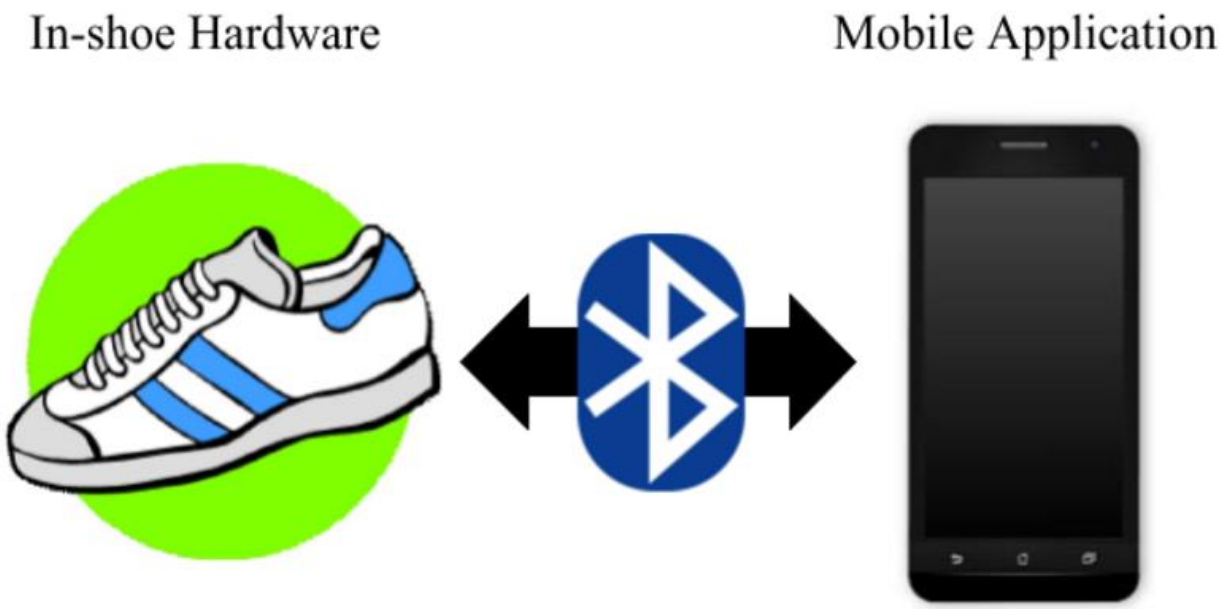

Figure 1: Simple Stylized System Diagram

2.2 Interface Requirements

The mobile application will provide a user-friendly interface that allows the end user to easily view data acquired by the device. The customer should be able to view current step count data as well as past archived step count data as desired. The application will be able to store exercise data as well as step count data. The user will be able to create and store their own exercises in the application for future reference and reuse.

The insole hardware should require minimum manual maintenance performed by the consumer. The lifetime of the product should be at least two years without maintenance. This number is consistent with the warranty period of a Fitbit device.[2] A complete list of the interface requirements can be found in Table 1 below. Requirements are separated into hardware and software requirements and listed based on their perceived importance. They are each given a letter and number combination that represents the type of requirement and the importance.

[2] "Returns and Warranty - Fitbit." 12 Feb. 2016, https://www.fitbit.com/legal/returns-and-warranty. Accessed 10 Dec. 2016.

Table 1: Interface Requirements

	Requirement
Software Interface	SI.1 The user interface of the mobile application should be intuitive and usable by any customer over the age of 8
	SI.2 The user should be able to view their step count data and exercise data
	SI.3 The user should be able to view data from up to 90 days prior, assuming a usage of once per day
	SI.4 The user should be able to create and save their own exercises in a predefined intuitive data format
	SI.5 The application should provide some form of interactive achievements/statistics to encourage regular usage
Hardware Interface	HI.1 The user should only need to interface with the in-shoe hardware via the battery or battery charging device
	HI.2 The device should function without manual user maintenance for two years

2.3 Functional Requirements

The functional requirements for both the hardware and software were defined by the Smart Sole team. The goals of this project are to capture energy from kinetic motion (through the energy recapture module), to power sensors for analysis, and to connect and successfully send data to a phone application. Requirements were defined to help achieve these goals. The requirements were once again divided into hardware and software requirements. Some important aspects considered when developing these requirements were ergonomics (size and weight), power consumption, voltage and current levels, durability, reliability, functional capabilities, cost, security, and safety. These aspects were all analyzed and condensed into a list of requirements that aims to address any functional concerns regarding the product. The list of requirements, in order of perceived importance, can be found in Table 2 (for hardware) and Table 3 (for software) below. The requirements are each given a letter and number combination that represents the type of requirement and its importance. These tables include the additional requirements that were defined by the team following the writing of the PPFS. These additional requirements are intended to better define the scope and purpose of the project.

Table 2: Hardware Functional Requirements

	Requirement
Hardware Functionality	HF.1 The device shall be able to record and export step count data
	HF.2 The device shall not cause any additional safety concerns such as cuts, blisters, electric shock, battery corrosion, or anything else that might harm a user
	HF.3 The device should be designed to minimize power consumption to a level comparable to other similar products
	HF.4 The device should be able to fit into a shoe without causing discomfort to a walker/runner
	HF.5 The cost of the product should be less than that of other comparable products on the market
	HF.6 The device should be water resistant and usable in wet and humid conditions

Table 3: Software Functional Requirements

	Requirement
Software Functionality	SF.1 The mobile application shall be able to export data from the Bluetooth processor in the hardware
	SF.2 The mobile application should have an idle mode that uses minimal processing power and phone resources
	SF.3 The mobile application should store user data for up to 90 days
	SF.4 The mobile application should be able used to store data during the run or after
	SF.5 The mobile application should be fun and easy to use
	SF.6 The mobile application should save user data locally
	SF.7 The mobile application should be secure and prevent third party access of customer information

2.4 Performance Requirements

Step counting was the primary performance objective of the Smart Sole device, and distance tracking and route mapping were secondary stretch objectives. Preliminary research showed that step counters currently on the marketplace all have some level of error in step counting and distance tracking; however, different studies showed differing levels of error. A study performed by various medical researchers found that differences in mean step count ranged from -22.7% to -1.5% for wearable fitness trackers and -6.7% to 6.2% for mobile application fitness trackers.[3] Data obtained from this study can be found in Figure 2. The data displayed is for the 500 step trial. According to the researchers, findings were "mostly consistent" between the 500 step and 1500 step trial. The counters with the highest accuracy, the Fitbit One and the Fitbit Zip, are both clip-on devices. In general, clip devices were shown to be more accurate than other wearables.

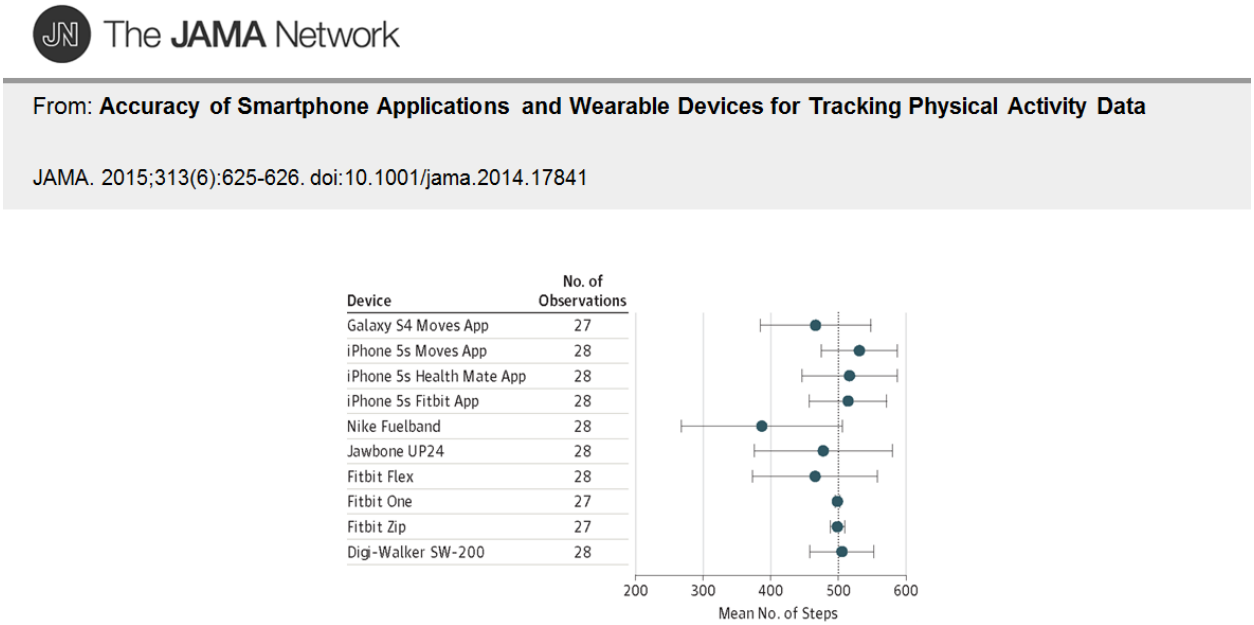

Figure 2: Step Counter Accuracy

[3] Case MA, Burwick HA, Volpp KG, Patel MS. Accuracy of Smartphone Applications and Wearable Devices for Tracking Physical Activity Data. *JAMA.* 2015;313(6):625-626. doi:10.1001/jama.2014.17841

A group of engineers from universities in Sheffield, United Kingdom also performed similar research in 2015.[4] This group examined the step count detection accuracy of seven separate commercially available activity monitors. The devices used in this study were all different forms of wearables that are worn on various locations on the body such as the lower back (Movemonitor), waist (Fitbit One), legs (Tractivity and ActivPAL), or arms. This study also observed the effects that walking speed could have on the accuracy of a step count reading. This became another important factor that the Smart Sole team began to consider when developing performance requirements. The data for this research was provided in the form of mean percentage error (MPE). Mean percentage error is found by taking the mean of the summation of the error percentages for each trial. A graphical representation of the data can be seen in Figure 3. The error bars displayed on the graph represent plus or minus one standard deviation. The data showed that most fitness trackers, while accurate at a high speeds, struggled with accuracy at a low walking speeds.

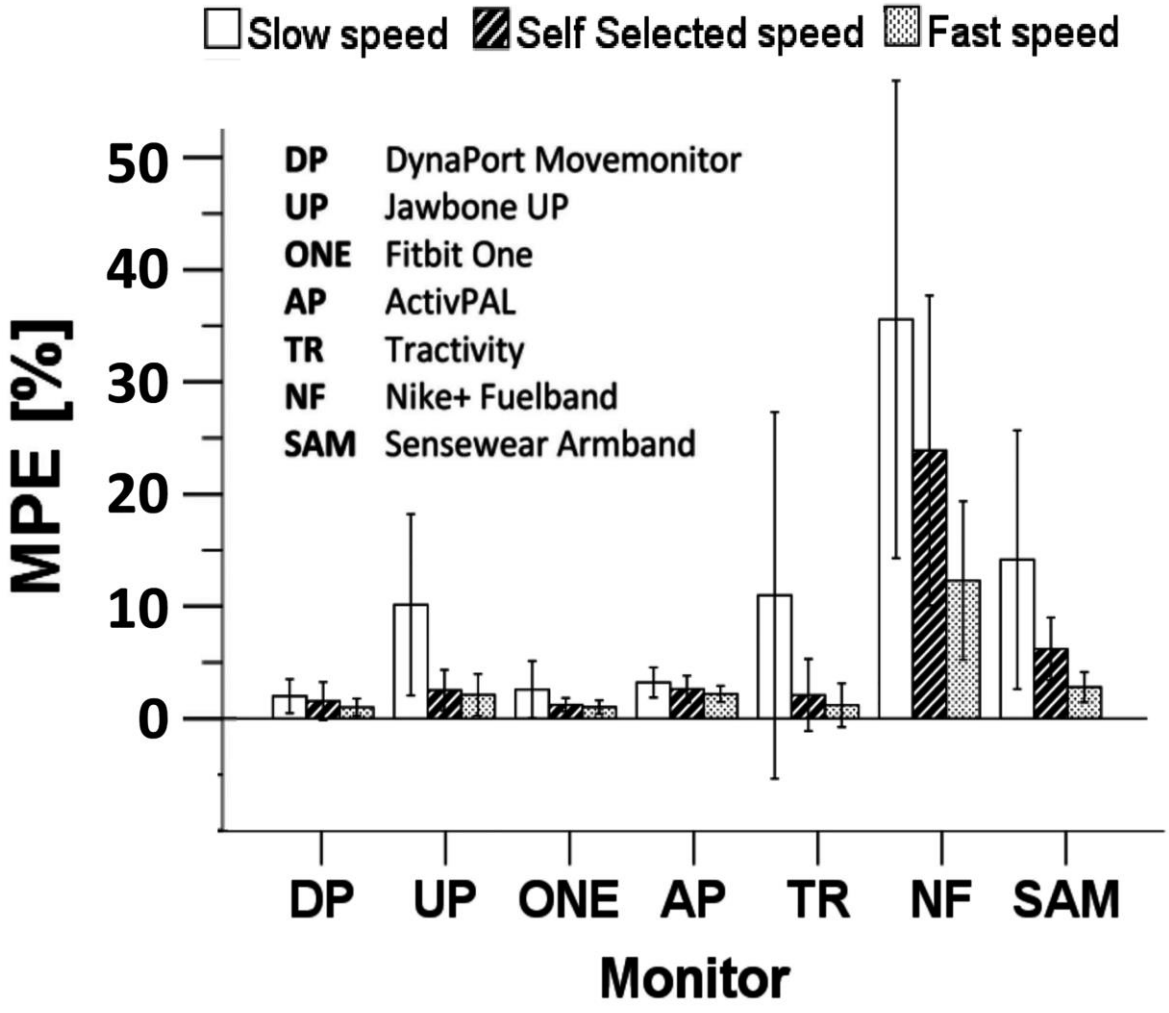

Figure 3: Step Counter Error at Varying Walking Speeds[4]

[4] "Step Detection and Activity Recognition Accuracy of Seven Physical ...." 19 Mar. 2015, http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118723. Accessed 11 Dec. 2016.

Distance accuracy is also a problem seen in wearable devices. A study performed in 2015 by Dan Graziano of CNET found that distance tracking error ranged anywhere from 0.3% (calibrated Apple Watch) to 16% (Pivotal Tracker 1).[5] The full results can be viewed in Figure 4. These results are based on the average of three walks. The distance was calculated by the distance measuring functionality of a treadmill. Additional research performed on the accuracy of fitness trackers showed similar results.

Activity trackers

Device	Steps	Distance (mi)	Difference*	Deviation*
Apple Watch (calibrated)	2,097 avg	1.003 avg	0.003	0.3%
Garmin Vivosmart	2,079 avg	1.01 avg	0.01	1%
Lifetrak Zone C410	2,120 avg	0.96 avg	-0.04	4%
iFit Active	2,166 avg	0.96 avg	-0.04	4%
Misfit Shine	2,102 avg	0.93 avg	-0.07	7%
Fitbit Charge	2,108 avg	0.91 avg	-0.09	9%
Apple Watch (out of the box)	2,107 avg	1.10 avg	0.1	10%
Pivotal Tracker 1	2,083 avg	0.84 avg	-0.16	16%

Figure 4: Distance Accuracy of Activity Trackers[5]

The performance requirements that the Smart Sole team developed were based on the research outlined above. The Smart Sole product should provide a step count and distance monitoring accuracy that is, at the very minimum, comparable to the current leader of the market competition. This threshold was set at less than plus or minus 5% for step count and less than plus or minus 4% for distance (if applicable based on design decisions).

Additional performance considerations included power, charge time, standard deviation of error, and application performance. The original Smart Sole design intended to use all renewable energy, but the team later determined that this was unfeasible. Discussion of this begins in section 5.3.1 within this report. Battery and power requirements are dependent on the implementation of the system's power circuitry. Rechargeable batteries are seen as desirable as they have a longer lifetime and would not need to be replaced; however, requirements were developed for both options so the team would have objectives that were clearly defined when making a decision regarding the power supply circuitry. Requirements of the design option that is not used in the final product were eliminated when the proper decisions had been

[5] "How accurate is the Apple Watch's step counter and distance tracking ...." https://www.cnet.com/news/smartwatch-step-counter-and-distance-tracker-accuracy/. Accessed 11 Dec. 2016.

made. The requirements are each given a letter and number combination that represents the type of requirement and its importance (Table 4).

Table 4: Performance Requirements

	Requirement
Hardware Performance	HP.1 Step count error should be less than +/- 5%
	HP.2 The internal battery should be able to last at least 8 hours in use
	HP.3 The internal battery should charge from empty to full in under 2 hours (if rechargeable battery is used)
	HP.4 The rechargeable battery should last the lifetime of the product (~2 years)
	HP.5 The hardware should enter an idle low power mode if not in use to conserve battery life
Software Performance	SP.1 Data should be exported to the mobile application in less than 5 seconds
	SP.2 Application performance should be consistent with what is expected by users (verified through testing)

2.5 Environmental Requirements

One of the initial aims of the Smart Sole team was to produce a product that would have a minimal environmental impact. The team also hoped to promote sustainability by harnessing energy generated by the piezoelectrics. The initial primary environmental requirement was to use renewable energy as the primary source of power. This way, additional power is not needed to power sensors. When a completely self powering device was determined to be unfeasible, the Smart Sole team decided to test the energy recapturing capabilities of the piezoelectrics by designing a separate module that's primary purpose is to recapture the energy into a rechargeable battery. This modification allowed the Smart Sole team to continue their original goal of energy sustainability. The facilities and materials used to create the product should also be taken into account when considering sustainability. The facilities and materials used to produce the product should be carbon neutral and limit the amount of waste. This will be taken into consideration when developing a business plan and selecting vendors of the individual components.

The electronic hardware used within the product should be recyclable at any electronic recycling center. The components should also be environmentally safe if a user unknowingly disposes of the product in the trash. A concise list of all the environmental requirements can be found in Table 5. The requirements are each given a letter and number combination that represents the type of requirement and its importance.

Table 5: Environmental Requirements

Requirement
E.1 The insole hardware should be powered via a rechargeable battery to reduce waste
E.2 The vendors of components used in the Smart Sole product should incorporate sustainability practices into their business model
E.3 Smart Sole facilities used for production should minimize energy usage and waste in an attempt to remain carbon neutral
E.4 The electronics used in the product should be disposable at any electronic recycling center
E.5 The product should be trash safe

2.6 Project Deliverables

In addition to all of the requirements established for the functionality of the Smart Sole product, a list of deliverables was also generated. These deliverables are listed along with their status in Table 6. Certain deliverables such as the mobile application and website will continue to undergo maintenance whenever necessary.

Table 6: Project Deliverables

Deliverable	**Status**
Final PPFS	Completed 12/12/16
Final Report	Completed 5/10/17
Working Prototype	Completed 5/1/17
Mobile Application	Completed 5/3/17
Design Notebook	Completed 5/10/17
Team Website	Completed 5/10/17

3 Scope and Constraints

The Senior Design course offered by Calvin College is a two semester course; therefore, the total scope of the project is limited by the time available. Furthermore, although Calvin College provides excellent facilities for design work, there is certain equipment that will not be available for use, thus limiting the scope. The scope of the project is also limited by the available budget. Each senior design team had a budget of approximately $500 to spend on components, travel, and any other expenses associated with the project. If addition money is needed, that money would have to be acquired via fundraising or a third-party sponsor. An additional constraint on the project scope was noted as the team began developing the mobile application. Because Swift was chosen as the programming language for designing the application, the Smart Sole team was forced to develop the application using Apple computers. The computers owned by the students and the computers provided by the department either could not run the proper operating system or didn't have the proper hardware. This limited the team to only programming the application when they had access to Apple computers (weekdays from 9AM to 12AM). While this constraint did not have a major effect on meeting the project requirements, it did limit the amount of features that could be included within the application. These are the main constraints of the project from a high-level management perspective.

The biggest constraints of this project, from a functional perspective, are space, comfort, component resilience, budget, and power. The size of the shoe limits the number of components the team can fit into the shoe. This limits the size of the components that will be used as well as their arrangement within the shoe. In addition, the comfort of the user must be considered. Comfort is crucial as uncomfortable shoes can lead to injuries and other safety concerns. Furthermore, a customer is less likely to use a product that they consider to be uncomfortable or inconvenient. The additional components must therefore be small and not add points of pressure on the user's foot that would cause pain or discomfort.

Another problem to consider is the resilience of the components. As the components are stepped on repeatedly over time by a user's foot, the mechanical stress has the potential to cause damage. The components the team picks must be able to handle the oscillatory impact over a long period of time. With unlimited money, the team has the potential to get the smallest military grade components that can handle high levels of stress; however, the team budget is finite. The cost of higher quality components would transfer over to the consumer as a higher sale price.

Finally, the system's power budget must be considered. Renewable energy recapture was determined by the team to be infeasible based on research and analysis that can be found in the team's PPFS (Project Proposal and Feasibility Study). Because energy cannot be recaptured at a reliable rate, the power budget primarily concerns the device's battery life. A goal of the Smart Sole team is to create a device that

requires minimal end-user maintenance, so it is important to maximize battery life to provide a pleasant user experience.

With all of the previously mentioned constraints in mind, the Smart Sole team established the scope of their project. The team saw the scope of their project as addressing a problem, namely fitness tracking error, by creating a more accurate, integrated design than current market options. Furthermore, Smart Sole aims to utilize their abilities as electrical engineers and the principles of design to systematically create a solution to inaccuracy. The team accomplished this with set benchmarks and testing plans in place to measure their success.

4 Project Management

4.1 Team Organization

The team consists of four members: Nick Beezhold, Garrett Schliebe, Andrew Twining, and Modeste Niragire. All of the members are Electrical and Computer Engineering Majors. In addition, the team has one business minor and a math minor. The project is divided into four roles, including hardware, software, project management, and energy capture. Each team member was in charge of a one of the key roles in the project; however, this did not limit that member to working solely on the role their own area. Each team member is expected to contribute to any role, as needed.

Nick Beezhold was in charge of project management. He was responsible for assigning responsibilities to the members of the team, setting goals and creating a schedule, as well as tracking the team's progress. In addition, Nick handled all of the budgeting and the business plan. He also assisted in the research and design of the hardware portion of the project, specifically the calibration and setup of sensors and the management of their data.

Garrett Schliebe was in charge of the software portion of the project. This included designing and coding an app for the iPhone that will collect the data from the hardware through Bluetooth. Garrett headed up the data storage on the software side as well as how the app will display various measures. In addition, Garrett was the team webmaster.

Andrew Twining was in charge of the hardware portion as well as a liaison between the hardware and software. He was responsible for researching and designing the circuitry to capture and store the energy from the piezoelectrics. He also worked heavily with the microcontroller code and how it handled input. He also lead in implementing the Bluetooth interface and ensuring proper data transfer.

Modeste Niragire was in charge of the energy recapture portion of the project. This included researching and choosing the piezoelectrics used to generate the power. He also determined the power storage solution and the necessary energy needed to recharge the battery. Finally, Modeste was in charge of integrating the hardware into the shoe in a way that is ergonomic.

Additional contributors to the team included the team advisor, Mark Michmerhuizen, and the industrial consultant, Eric Walstra. Mark Michmerhuizen is an Electrical and Computer Engineering Professor at Calvin College and gave both industry experience as well as project management advice to the team. Eric Walstra is an Engineering Manager at Gentex and has more than 15 years of industry experience. A meeting with Eric Walstra was held on November 3 to discuss the project and risks involved. Figure 5 shows the roles of the project supervisors throughout the semester.

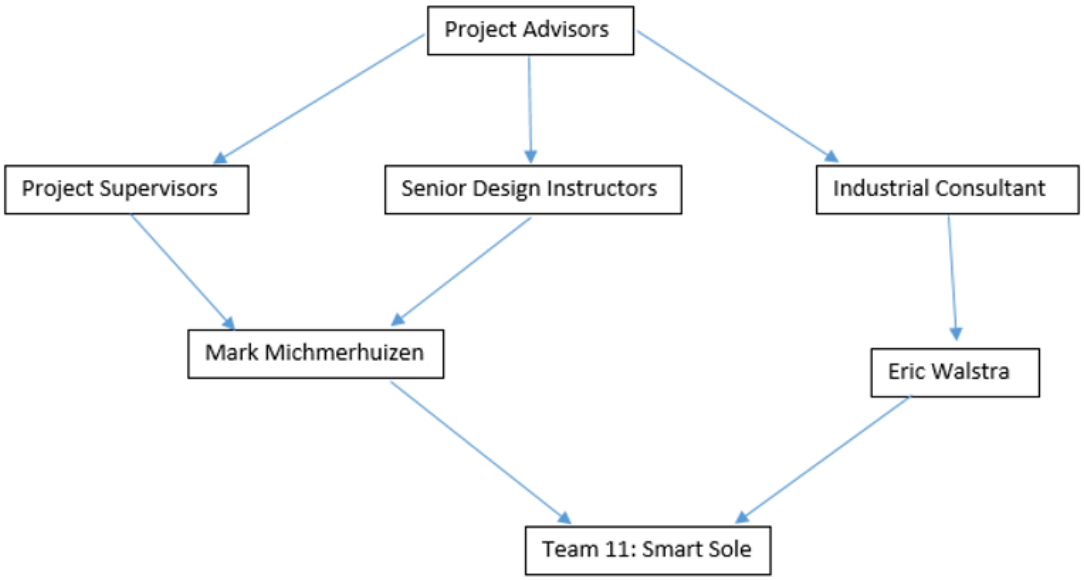

Figure 5: Project Supervisors

Team meetings were held on a weekly basis, with additional meetings during Senior Design class time occurring approximately once a week. These meetings typically include project research time, project planning, and project work days to generate required documents. Additionally, any team conflicts among members such as disagreements over design decisions were resolved with a group discussion before moving on. Input for each side was considered before making a joint decision. All team documents are held in a shared folder on Google Drive. This includes meeting minutes, research notes, presentations, schedules, testing results, and any other document or information relevant to the team. The folder is currently only shared between team members and the project supervisor Mark Michmerhuizen. Additionally, the final three weeks of lab time for a required computer engineering course were utilized to design portions of the step count module.

4.2 Schedule

Nick was in charge of maintaining the schedule and adjusting goals and deadlines weekly based upon the realized progress of the project. A work breakdown schedule was created to assist in determining what jobs and tasks needed to be done, how long these tasks would take, and when they should be completed. The complete work breakdown schedule is included in Appendix A. In addition, weekly progress reports were written and sent to the team advisor in order to notify the advisor of their progress and any issues, and to also maintain accountability. A table of schedule highlights is provided below in Table 7.

Table 7: Schedule Highlights

Project Milestone and Description	Date
Define and Propose the Project	September 14, 2016
Complete Project Feasibility Research	October 19, 2016
Post Website	October 26, 2016
Determine and Order Piezoelectrics for Testing	November 17, 2016
Step Count Module Prototype	December 8, 2016
PPFS Final Draft	December 12, 2016
Finish Prototype and Begin Testing and Troubleshooting Prototype	March 17, 2017
Consider Additional Hardware Features	April 14, 2017
CEAC Review	April 21, 2017
Develop Technical Documentation	May 2, 2017
Senior Design Project Presentation and Banquet	May 6, 2017

An order of tasks was determined by taking into account both time, risk, importance, and whether one task was a prerequisite for other tasks. For example, research was required before piezoelectrics could be ordered. Included in the weekly meetings is a discussion about whether the team is on schedule or if adjustments should be made. Some deadlines are static, such as the first major developmental requirement and the final PPFS, due on December 12. Most deadlines, however, could have been adjusted, as long as the project was completed in a timely matter. It was a priority to stay on schedule and not have to constantly readjust goals. It is for that reason that detailed scheduling took place, in order to avoid delays in the project.

As research and design advanced, reevaluations occurred more frequently based upon the determination that deadlines weren't always realistic. No major problems occurred, although there were many obstacles, which were to be expected. The amount of time spent on the project increased significantly in the second semester when compared to the first semester. The first semester consisted primarily of research, minimal design, documentation, and assignments. The second semester consisted mostly of design, construction, optimization, and testing. Time dedicated to the project in the final two weeks increased considerably. The team completed the project on time and left roughly two weeks dedicated to testing and optimizing the product as well as preparing for Senior Design Night. The total combined amount of time spent on the project was roughly 500 hours.

4.3 Budget

The budget was maintained by Nick. He was responsible for ensuring that the team remains on budget and that there were adequate funds for each area of the project requiring them. In group meetings, Nick discussed with each team member materials that would be required for their portion of the project, based upon their research and design. He then determined if the proposed materials were appropriate in relation to the budget or if there may need to be a change. The $500 budget provided by the Engineering Department was sufficient in funding the project. Major costs of the project included the piezoelectrics, Smart Bluetooth, microcontrollers and backup components. Total expenditures totaled $272.55, which included numerous components but also shipping. The detailed budget is located in Appendix C, and the developmental budget is included in section 7.2.

Each time a team member proposed a material to be purchased, Nick reviewed the budget and determined whether the purchase was appropriate in relation to the budget. The budget is updated each time permission was given to purchase an item. A comparison of the estimated budget and actual costs is done once a week. Nick filled out the order forms and sent them to the advisor to be approved. His goal was to order many parts from the same location or store at once, in order to limit shipping costs.

4.4 Method of Approach

The team started this project by identifying the objectives of the design. This involved determining a set of requirements that would provide identifiable objectives for the team. After establishing the purpose of the project and defining the problem, the alternative solutions to the problem were considered. The proposed solutions revolved around several ideas. First, that the shoe can be used to generate power that can subsequently be used to power the sensors such as an accelerometer and Bluetooth. Second, the shoe should be able to export information to portable electronics such as an iPods and cellphones through an application. When conducting research, there was immense concentration on satisfying the requirements of generating constant and uninterrupted energy and transferring information from the sensors to the user through the app. For energy generation, the use of piezoelectric devices was heavily relied on. It is known that with this technology, electrical energy can be converted from kinetic motion. The key research questions of this project were where and how should the piezoelectric devices be implemented to perform the work effectively; however, based on initial testing of piezoelectric materials, generating enough energy to power the sensors was ruled unfeasible.

As a team working together to accomplish a common goal, good communication amongst members was crucial. Most research and design work for the project was done on the individual or small group level, so holding weekly team meetings gave an opportunity for individuals to update the other members of the

team. In the status update meetings, the current status of the project was evaluated by highlighting what had been done and what tasks were priorities. The team established a Facebook group and mobile group message as the primary way of communicating, but interpersonal email was also used, especially for communicating with the project advisor.

The team understood that to achieve the project goals, they needed to make sure that fundamental principles of teamwork were followed. Those principles include cooperation with each other, dedication and commitment to the objectives, and effective communications. The team believes that we are agents of God's renewal, and we needed to ensure that we approached this project with kindness, honesty, and an appreciation of God's creation.

4.5 Task Specification

The Smart Sole team developed a list of tasks that must be completed in order to produce a prototype that is functional and marketable during their time on this project. The full work breakdown structure, in text form, can be found in Appendix A: Work Breakdown Structure.

5 Designing the Project

5.1 System Design and Architecture

The system design proposed by the Smart Sole team consists of two primary blocks: the in-shoe hardware and the mobile software. The basic block diagram of the system hardware can be seen in Figure 6.

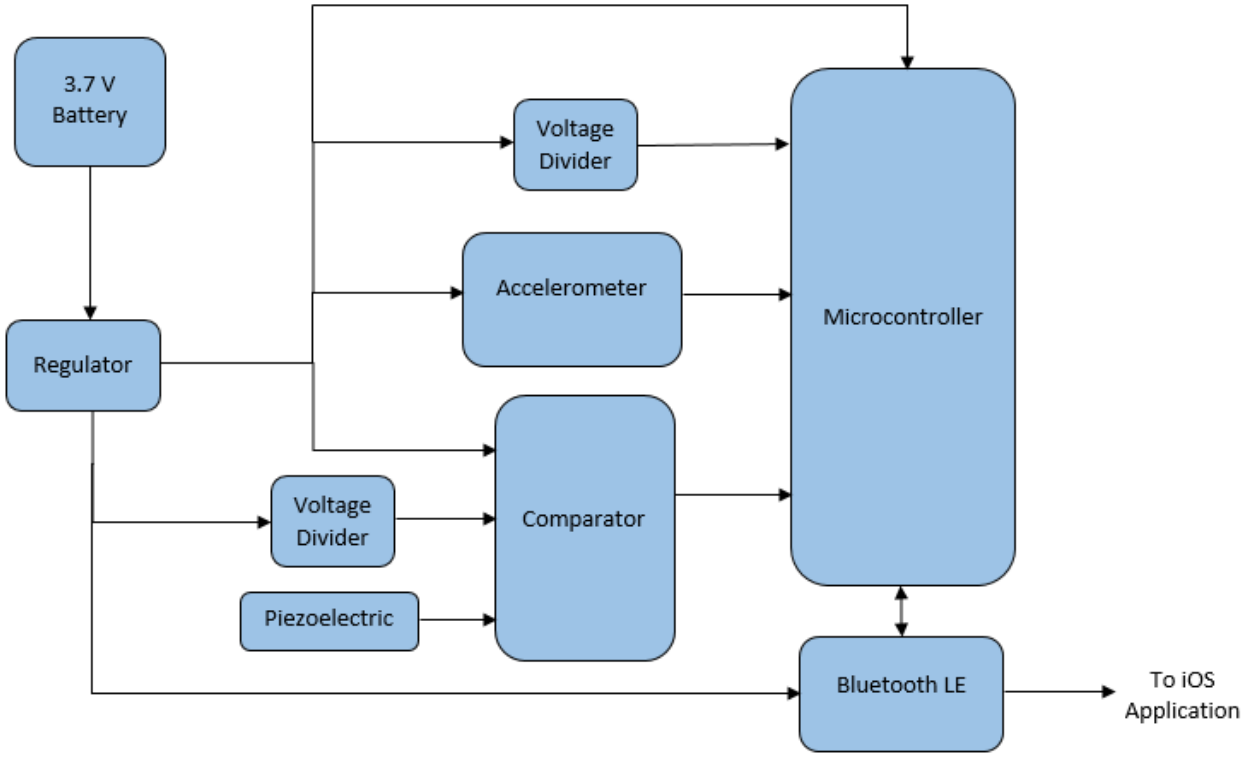

Figure 6: Block Diagram of Hardware Design

The primary hardware components used in the step counting module include a piezoelectric, a microcontroller, Bluetooth LE, a comparator, and an accelerometer. The two methods of data collection include the piezoelectric and accelerometer, with the piezoelectric being the driver. A seven state machine is used to determine when a step is taken. The piezoelectric is the primary driver, and the state machine will not advance to the next state unless a voltage spike from the piezoelectric is sensed. The final state is where the step count updating occurs as well as where the data is sent to the Bluetooth module. After the first state, there are multiple possible paths to reach the final state. This is necessary to take into account different walking environments, including walking up or down stairs, and how the accelerometer responds to each. The steps are continuously added and stored in the microcontroller. Because the module is only located in one shoe, every step sensed is multiplied by two to account for the other foot. This does lead to the possibility of overcounting if a user would stop and start with the same foot. Over the course

of a day, it is assumed most of this potential error balances out. The data is sent through Bluetooth to the app whenever the user requests input. The method of resetting the step counts to zero is done through the app, although a reset button is included on the microcontroller.

The accelerometer used is a triple-axis accelerometer, with a selectable scaling up to 16 g. A scaling of 4 g was selected, and only two axes were necessary: the y-axis for forward-backward motion and the z-axis for up and down motion. Side to side motion was not deemed useful enough to assist in determining steps. Originally, only the y-axis was used for the accelerometer. The brief forward acceleration of a step was enough to sense it. The threshold used is just over one gravity of acceleration in the forward motion. When the foot is lifted and about to move forward in the stepping motion, it is pointed down, so an additional effect of gravity impacts the acceleration of the y-axis. The use of one axis worked well in the ideal case of walking on flat paths, but was not enough to sense more realistic cases of walking up and down stairs, where the forward motion is minimal. A second axis, the z-axis, was added to account for these situations. The values of acceleration readings were analyzed to determine how best to set thresholds of z-axis accelerations. The effect of gravity on the accelerometer provided a unique challenge to overcome. Additional design details and decisions are located in Section 5.4.

Additional components for powering the device include a battery, regulator, and a separate battery charger via USB. A simple voltage divider connected to the microcontroller provides a method to read the voltage level of the battery to determine the remaining battery life. GPS was not included in the final project, although a later implementation would use the smart phone's GPS rather than an in-shoe GPS, due to both cost and power considerations.

The mobile application software design was done using Apple's Swift language in the Xcode integrated development environment (IDE). The criteria used to make this decision can be found in the following sections. The primary objectives when designing the mobile app were derived from the software requirements defined in Section 2 of this report. These objectives include making the application easy to use, designing it to perform efficiently and effectively, and allowing the user to store data. The final software design included a method to view and store step data as well as a method to view and store exercises. The additional exercise feature was added to provide the user with a "one-stop" service that allows them to track all aspects of exercise, instead of focusing solely on running. The mobile application was designed in parallel with the insole hardware. The application was modified when necessary since the software is easier to change and reconfigure than the hardware.

5.2 Design Criteria

5.2.1 Hardware Design Criteria

When designing the hardware, the team considered a number of factors. Three main sections were developed that contained these individual factors. The sections considered were renewable energy considerations, power management considerations, and user considerations. Renewable energy considerations include sustainability and the feasibility or impact of the renewable energy aspect of the project. Power management considerations concern intelligent distribution of power and minimization of power consumption. Finally, user considerations include any impact on the user during usage and both appearance and functionality. These design criteria are not for specific components but are instead intended for use when examining features on a macro level. The following descriptions are meant to give further insight as to how blocks of the overall block diagram were selected. In addition to the following categories, design norms played a big part in decision making. These design norms are discussed in further detail in Section 5.5.1. All criteria are given a weight out of 5 with 1 being the least important and 5 being critical.

Renewable Energy

Power output is an important factor for the product, because the more power available, the more analysis that can be performed and given to the user. The team gave this a weight of 4 out of 5 because the power issue is the foundation of many of our issues for this project.

Development Time

Development time is the amount of time it takes to create the product. This determines whether various parts of the project are feasible and whether the entirety of the project can be completed in the scope of the class. This criterion was given a weight of 3.

Capability

Capability is an important criterion for the team. Capability determines whether the product has the ability to gather energy and use the energy. The product must be capable of performing the tasks that are required of it; therefore, the team assigned this criterion a 4.

Availability

Availability is the ability to acquire all necessary parts and pieces for designing the energy harvesting component. The team assigned a 3 to criterion because, without access to needed hardware, the functionality of the final product could be hindered.

Ease of Building

Ease of building is determined by how difficult the system examined will be to develop. This criterion was not especially important because difficulty is expected. As long as feasibility is assured, difficulty can be negotiated. The team gave this criterion a 2.

Renewability

Renewability is whether or not the primary source of energy is derived from renewable locations. While the project could be powered by a number of other sources, the team desired renewable energy as a key source of power; therefore, the criterion received a 3.

Cost

Cost is the monetary cost to the team and the effect that it has on the budget. With a limited amount of money available, smart decisions must be made and cost analysis must be considered. Maintaining a less expensive production cost can be done through the use of inexpensive parts. The team assigned a 3 for this criterion because, while important, the team does not expect to go over budget.

Power Management

Power management is an ability to power all necessary components efficiently. This includes any additional sensors. By adding more analysis, the product increases in value to a potential consumer; therefore, a 5 was assigned to this criterion.

Feasibility

Feasibility is whether the added sensors will be able to be added or not. This is dependent on power and size considerations. It was given a 5 because if the project is possibly unfeasible, then there should be less consideration of it.

Comfort

Comfort is determined by how noticeable the hardware will be to the user when it is implemented inside or on the shoe. The team desires to make the hardware unnoticeable or as close to unnoticeable as possible. The team gave this consideration a 4.

Aesthetics

Aesthetics are how the shoe hardware makes the product look. Although not needed for functionality, it is needed for marketing and sales of the product, which generate revenue. Therefore, the team gave the consideration a 3.

Product Longevity

Product longevity determined by how long the hardware will last before it becomes unusable due to physical wear. The team desires the product to be long lasting so that the user can enjoy use it for a long time without having to have it replaced. It also will make the product more reliable; therefore, a 4 out of 5 was assigned.

Transferability

Transferability is whether the hardware is built into one pair of shoes or if it can be transferred to another shoe (e.g. an insole). This allows the product independence from any particular housing and makes the product interchangeable amongst shoes. The team assigned a 2 because, while it would be a nice addition, it is unnecessary for the first functional prototype.

Cost to User

Cost to user is how much the user will need to pay to offset the additional features added. To increase demand, the team wants the product to be as inexpensive as possible. Therefore, a 4 was given as a designated weight.

The final considerations, along with decision matrices, can be viewed below in Section 5.4.1.

5.2.2 Application Design Criteria

When creating a mobile application for our product, a mobile operating system on which to implement the software had to be selected. The criteria used for selecting a mobile platform on which to create the application were divided into two sections: considerations for the developer and considerations that would affect the end user. Developer considerations included any factors that would impact the research, design, and production of the mobile application. End user considerations included any impacts that the decision may have on the finalized product seen by the end user such as the functions available or the compatibility with the user's personal mobile phone. All application design criteria, including design norms, were given a weight out of 5 with 1 being the least important and 5 being critical.

Developer Considerations

Ease of Use

Ease of use of the programming software is one of the first developer consideration criterion established. Ease of use includes the difficulty of working with the language, the ease of obtaining the proper software to write the application, and the ability to add more features without having to completely rewrite the code. Ease of use is given a weight of 2 as it is important to consider but could be sacrificed assuming the person implementing the software is experienced in coding and able to learn quickly.

Development Time

Development time is considered when selecting the appropriate design. Development time includes the time to design, code, and implement the application on a mobile device. The time that the design takes to be implemented is important because the team is working on a strict deadline that cannot be adjusted if the design does not go as planned. Development time is therefore given a weight of 4 out of 5 as it is a crucial factor to consider but not the most important.

Feasibility

Feasibility of the software used to design the application is the next criterion considered. If the software is incapable of meeting the project requirement, it should not be considered as a viable option. Feasibility is given a weight of 4 since it is necessary to achieve the team's desired results.

Availability

Availability of software is the next criterion that was considered. The software that is used has to be easily acquired by the team. All platforms considered have free downloadable software; therefore, this criterion is given a weight of 3. This factor also included the availability of all equipment necessary to test the application.

Hardware Compatibility

Hardware compatibility is considered the most important consideration for the mobile application design. If the software is incompatible with the hardware, the product is completely non-functional. The product must be functional before any other requirements can be met regarding the application. Because compatibility is essential to accomplish all other requirements, it is deemed the most important criterion and given a weight of 5 out of 5.

Cost

Cost is the last developer consideration that was examined when selecting an alternative. One of the goals of the Smart Sole team is to create a product that is affordable and comparable to other similar products on the market. To achieve this goal, the cost of designing the mobile application must be kept to a minimum, Primary cost factors for the application were the cost of the design software, the labor cost of designing and implementing the application, and cost of the materials necessary to test the product. Cost is given a weight of 3 as the cost of the app development would be mostly fixed with the exception of labor to update and maintain the app.

User Considerations

A goal of the Smart Sole team is to create a product that is marketable and desirable to a wide consumer base; therefore, it is important to consider the needs of the end user when developing the product. Two primary needs were considered that fell under this category: functionality and compatibility.

Functionality

Functionality includes the ability of the application to perform all functions desired by the user. Functionality is defined as meeting the requirements for the application set by the Smart Sole team. Functionality also includes designing the application so that additional features based on potential customer feedback can be implemented in the future. Functionality is considered the most important user consideration and is therefore given a weight of 5 out of 5.

Compatibility

The final factor examined is compatibility with whatever mobile devices our target market may own. Our product is not useful if the consumer is unable to download the mobile application to their device. This factor is given a weight of 4 as it is important for us to keep our prospective market as large as possible.

Design norms are also considered when selecting the mobile operating system to program on. Design norms considered when proposing and selecting design criteria and alternatives can be found in Section 5.5.

5.3 Design Alternatives

5.3.1 Hardware Design Alternatives

One of the goals of the Smart Sole team was to increase the amount of energy captured through the running or walking process while still having a comfortable, marketable product. Smart Sole found through previous research that two different methods have been used for the energy capturing process. The first method uses a high powered generation device. Researchers Jingjing Zhao and You Zheng found they were able to harness an average power of 4 mW at 1 Hz (about walking speed) with this device. This gave them about 50 mW to work with. They determined that low power sensors could be added to provide additional data. The second method used a more comfortable mesh insole design. The same research team found that the more comfortable device could only give an output of 30 μW. They concluded that combining the power and comfort of the two prototypes would be the best way to approach the problem.[6] Originally, team Smart Sole intended to draw from the two prototypes in their own design and herald the work of Zhao and Zheng; however, after reconsidering the amount of energy available and the potential cost, the team decided an internal battery was a better option for a power source. The piezoelectrics were found to not provide enough power. The team will still use piezos in its step count module as a pressure sensor and potentially as a way to harness small amounts of power to extend the product's battery life.

5.3.2 Application Design Alternatives

One of the goals of the Smart Sole team was to make the final mobile application reach as large a consumer base as possible. Android OS and the Apple iPhone iOS were selected as possible platforms in which to develop the application. Android phones accounted for approximately 85% of all smartphone sales in the first half of 2016 with Apple selling the second most at approximately 13%.[7] This totals approximately 98% of the smartphone market. These sales have remained relatively stable for the past three years making Android OS and iOS the best design alternatives for reaching as large a market as possible. Furthermore, sales of Microsoft's Windows phone and other smart phones have been declining for the past two years. These phones also make up only approximately 2% of the market, making them unacceptable alternatives based on the team's desire to market to a large consumer base.

---

[6] Zhao, Jingjing, and Zheng You. "A Shoe-Embedded Piezoelectric Energy Harvester for Wearable Sensors." *Sensors* 14.7 (2014): 12497-510. *ProQuest.* Web. 13 Nov. 2016.

[7] "Mobile OS market share 2016 | Statista." 2016. 13 Nov. 2016 <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>

Programming in a language that is usable by both applications was another option that was considered. This can be achieved by using Oracle ADF Mobile, which used a Java and HTML5 based framework.[8] Oracle ADF Mobile has the ability to develop programs for both Android phones and iPhones; however, the development of these applications requires a developer certificate. Since the members of the Smart Sole team are not certified developers, this option was deemed unrealistic. Furthermore, none of the team members are familiar with programming in Java, making it necessary to learn a completely new language if Oracle ADF Mobile was selected as the development option.

5.4 Design Decisions

5.4.1 Hardware Design Decision

Hardware design decisions were based upon multiple decision matrices. These included renewable energy decisions, hardware decisions, and user preference decisions. As the project progressed, these matrices were reevaluated to represent the new goals and information the team obtained based upon research. Criteria were weighted on a scale of 1-5 with 5 being the most important. The solutions were given a number from 1-10 based on preliminary research, with 10 being the best. These design matrices are shown below in Table 8, Table 9, and Table 10. After reevaluating the design matrices, the team determined they would use an independent battery for powering the device, with a piezoelectric and accelerometer as the primary methods of collecting data. GPS would not be used in either hardware or software, but there would be potential for incorporating it in the smartphone in future work.

[8] "Building Mobile Applications with Oracle ADF ... - Oracle Help Center." http://docs.oracle.com/cd/E18941_01/tutorials/buildmobileappscontent/adfmobiletutorial_1.html. Accessed 10 Dec. 2016.

Table 8: Renewable Energy Decision Matrix

		Weight	Piezo Generator	Battery Charged by Piezo Generator	Rechargeable Battery
Renewable Energy Considerations	Power Output	4	1	4	10
	Development Time	2	6	4	8
	Capability	4	2	4	10
	Availability	2	6	6	6
	Ease Building	1	8	6	8
	Renewable	3	10	8	2
	Cost	3	5	2	6
	Stewardship	4	8	6	4
	Total		121	112	**156**

Table 9: Power Management Decision Matrix

		Weight	BLE, Step count	BLE, Step count, Accelerometer	BLE, Step count, Accelerometer, GPS
Power Management Considerations	Analysis Power	5	3	7	9
	Development Time	2	9	7	2
	Capability	4	4	7	9
	Availability	3	7	7	6
	Ease Building	1	8	6	4
	Feasibility	5	9	9	4
	Cost	4	8	8	6
	Stewardship	4	7	7	7
	Total		183	**209**	179

Table 10: User Preference Decision Matrix

		Weight	Generator Built in	Insole Generator	Mesh Insole
User Preference Considerations	Comfort	4	4	4	9
	Aesthetics	3	7	7	7
	Product Longevity	4	6	6	8
	Transferability	2	0	7	8
	Cost to user	4	8	7	5
	Trust	4	4	7	8
	Caring	4	4	5	9
	Integrity	4	7	7	7
	Total		153	179	**221**

After testing the potential of power generation by piezoelectrics, it was quickly determined that even a large amount of piezoelectrics would not be sufficient in generating enough power to continuously power the device. Testing showed that the power generation capabilities of a single piezoelectric was limited to the microwatt scale. Even with many piezoelectrics, it would still take an unrealistic amount of time to charge a separate battery. To offset the lack of renewable energy, the focus shifted primarily on functionality and analysis, in order to provide high accuracy to set the product apart from existing fitness trackers. The user preference decision matrix indicated that a mesh insole would be best, but since the team already concluded that power generation was unrealistic, the mesh insole could be used in future work to determine pressure points and other related features.

A small, 3.7 V, 500mAh lithium ion polymer battery was chosen to power the device. When the battery is fully charged, the actual voltage is 4.2 V. The battery is completely dead at 3.0 V. This information is used to assist in determining the battery life remaining, as is explained later. The capacity of the battery is 1.9 Wh, which is ample amount of energy needed to power the device for a full day. This size was chosen for the prototype to ensure a long battery life as well as lessening the charging time needed during testing. In production, though, a 150mAh would be chosen due to the smaller size and cheaper price. At 4.65

grams, the weight of the 150 mAh battery is less than half of the 10.5 gram, 500 mAh battery used in the prototype. It is also much smaller in all dimensions. Initially, two batteries were purchased, with the idea of one battery powering the step-count module in one shoe, while another battery would charge in the other shoe using a number of piezoelectrics. After rectifying the input voltage from the piezoelectrics to five volts, which would then be used in a charger that regulated the voltage to the proper level to power the 3.7 volt battery, the current was still only one tenth of a milliamp. Therefore, the final product would not use a power generating circuit in a shoe separate from the device, but rather forgo the power generation altogether. One of the goals of the team was to use renewable energy to power the device, but it was determined to be unfeasible and the team focused on their primary goal of increasing accuracy of the step counts, with a secondary goal of producing a low cost device. To charge the battery, the team purchased a lithium polymer charger with a micro USB jack to allow for easy charging with a USB cord through a laptop. The default charge current is 100 mA and is designed to work with the lithium polymer batteries used with the product.

When choosing the microcontroller, it was important to consider size, power consumption, and ease of use. The microcontroller was desired to be as small as possible while containing enough pins to support the external components of the accelerometer, comparator, and voltage readings. It was also desired to use the smallest microcontroller possible while still having enough processing power and memory to run the program. The small size was desired for both physical size but more importantly for the power consumption. It was very important for the microcontroller to be easily programmable. It was for that reason that Arduino microcontrollers were used. The Arduino Software Integrated Development Environment was readily available, and the microcontroller was easily programmable through micro USB using the serial port connection. The Arduino also came with its own breakout board for the microcontroller, as well as all the components necessary to run it. After researching a number of microcontrollers, the Adafruit Feather 32u4 Bluefruit LE microcontroller was chosen. As indicated in its name, this microcontroller also contains a Bluetooth LE module on the breakout board, so purchasing the independent Bluetooth LE module and soldering it together was not necessary. The actual microcontroller used on this board is Atmel's ATmega32U4, which is clocked at 8 MHz and 3.3V logic and has 32K of flash and 2K of RAM. The board contains 20 GPIO pins, a 3.3 V regulator, and a connection port for the lithium ion polymer battery. In addition, it measures just 2.0" by 0.9" by 0.28". At 5.7 grams, the device is very light. Adafruit provides many different development boards serving all different needs, but the Feather 32u4 was the best based upon size and capacity. There are more than enough pins, but many other boards had either too few pins or were too large. The ATmega32U4 processor was the perfect size so only several physical compromises had to be made. In the end, 72 percent of the microcontroller's memory was used.

A triple axis accelerometer was used as the other method of collecting data for step-counting. There are numerous accelerometers on the market, many with slight differences that aren't applicable to the project. Accelerometers come in both analog and digital versions, and the choice is typically dictated by the hardware used. In general, analog accelerometers are prone to more noise, and digital accelerometers are typically easier to work with. The reality is, though, that most digital accelerometers really just have an onboard analog to digital converter, and the sensors are analog at the core. A digital accelerometer was desired for the device. The number of axes in the accelerometer was another consideration. As mentioned previously, the team determined at least two axes were needed to determine a step. Although this was not known when the accelerometer was chosen, the team accounted for the potential need for multiple axes and chose a three axis accelerometer anyway. Additionally, most accelerometers were only available in three axes, and there was virtually no cost difference nor power consumption difference. A third consideration for the accelerometer was the sensitivity. Generally speaking, the more sensitivity the better. This was very important for the Smart Sole device, where even small changes in acceleration can be significant in determined steps. Since more sensitive accelerometers produce a larger change in the signal, they are easier to measure and the microcontroller will receive more accurate readings. A microcontroller with a 4 g scale was determined to be the most optimal for the project. Accelerations for walking and running were not expected to go beyond 4 g, and this was still a small enough scale for the readings to be sensitive enough. A 2 g scale would be even more accurate for readings within that range, but at the time of the decision, it was unknown whether or not thresholds would be set above this level. For this reason, it was a safer choice to use an accelerometer with a higher scale. A final consideration for the accelerometer was the power consumption. A small and low power accelerometer was desired to preserve as much energy as possible. Taking all of these factors into consideration, the team chose STMicroelectronics' LIS3DH accelerometer. This particular accelerometer was available from Adafruit on a breakout board, saving the team time from fabricating their own, which was a major factor in the choice. The accelerometer is a digital, three axis accelerometer with 10 bit precision. It provides both I2C and SPI interface options, although only the SPI interface was used for the project. The accelerometer allows for selectable scaling from 2 g to 16 g. 4 g scaling was chosen for reasons previously provided. Finally, this is an ultra low-power, high-performance accelerometer that has a current draw as low as two microamps. The accelerometer size itself is extremely small, and the breakout board is still less than one square inch, with a weight of just 1.5 grams.

Originally, piezoelectric generators were chosen to generate electricity and input data for a step. However, these generators were fairly large so small piezoelectric disks were chosen instead. The disk is roughly one inch in diameter, less than one quarter the width of a coin, and has virtually no weight.

Several methods were considered and tested when it came to determining how to read the piezoelectric data into the microcontroller. Every time the piezoelectric is compressed, a large voltage spike with

minimal current occurs. Due to the sensitivity of the piezoelectrics, some noise and bounce also occasionally occurs, which is undesirable. The first method tested was a direct analog read into the microcontroller. Since the piezoelectrics produce a large voltage spike, a certain threshold can be set to trigger a step, such as 3 volts. The microcontroller runs on 3.3 V, so usually extremely high voltages would be dangerous to input directly. A voltage divider could be used, but since the voltage spike values vary widely, this is not a consistent method to ensure the threshold level is passed. Additionally, a voltage divider with even small resistors in the single digit ohms is too much resistance for there to be any reading. The current produced by the piezoelectrics is extremely small already, and there is virtually no current produced with resistors. Since the current is so small, direct reading with the high voltage spikes was deemed alright. A small capacitor that could handle the high voltage was placed in parallel with the input and ground to maintain the voltage spike for a slightly longer period to assist in reading the value. After testing, the team determined that this method was not viable since the reading was inconsistent. It read well upon tapping the piezoelectric, but actual compression did not read correctly. The second method tested was a reverse biased zener diode voltage regulator. A current limiting resistor was not necessary due to the small current, and as mentioned previously, using one would eliminate the current. A 3.3 volt zener diode was used in reverse bias to provide a regulated 3.3 volts to the input of the microcontroller upon a voltage spike. This would in effect produce a digital high signal when the piezoelectric was compressed, or produce a digital low signal otherwise. This method did work a fair amount of the time, but it was not consistent enough to be reliable. The zener diode selected was one that required a low current to reach the breakdown point, but it was concluded that even with the specific diode the current was not always reached, which was the reason for the inconsistencies. Therefore, although this method was highly desired since there was no need to power the device as is the case with a comparator, this method was not used in the final prototype. The third method tested was using a comparator. A comparator is a circuit that compares an input voltage to a reference voltage. If the input voltage passes the reference voltage, the output voltage is pulled either high or low depending on the type of comparator. When the input voltage is below the reference value, the output voltage is just the opposite. This method worked well in eliminating noise while outputting a value that can be used as a digital high or low value. Different reference voltages were tested to find one that produced consistent readings without being too sensitive or not sensitive enough. Unfortunately, a comparator must be powered. However, due to the near perfect consistency, a comparator was still chosen. The output used a 3 kilo ohm pullup resistor from the regulated battery voltage to output either a digital low or digital high 3.3 V value. The comparator was powered by the regulated battery power, and a voltage divider using very large resistors to limit power consumption was used for the reference voltage. The optimal reference voltage was determined to be just 0.8 volts, so a voltage divider was designed accordingly to divide the 3.3 V down to 0.8 V. In order to improve the power consumption of the prototype, a smaller, single

comparator was used. The comparator itself worked the same as the other comparator previously used. However, the leads were very small and brittle, and broke upon testing.

One of the features on the app was for the battery life remaining to be displayed. This was accomplished by reading a voltage level from one of the input pins into the microcontroller. A voltage divider from the battery input to ground was used and split the voltage in half. The resistors were necessary to limit the current to the input pin, but two identical resistors were used in the divider so the voltage read into the microcontroller was simply doubled when calculating the actual battery life remaining. This data was sent to the Bluetooth module.

In regards to the Bluetooth service, two potential options were available. The first was to use existing architecture and protocols predefined for similar applications. This option provided the potential for added ease of development as well as more background information for debugging the service. The second option was to establish a custom service for our product. This method, although a more complicated process, allows much more adaptability. By providing specific custom characteristics for steps and battery percentage, a custom service provides a fitted service with no adaptations necessary. The team decided to go with a custom service. Although this was a challenge to establish, once connectivity was established, the stored characteristics were easy to access. This setup was done on the microcontroller through C code framework.

Two methods of designing and writing the program for the step count module were considered. The first method considered was a state machine. The second method considered was using interrupts. The team was much more familiar with creating a state machine using C code than using interrupts. It was for this reason that a state machine was written first to at least have a working product. The state machine has seven states. The first state sends the step count data to the Bluetooth module just once by using a flag. While it would be slightly more efficient to send the data in the seventh state, as is explained when talking about sending battery information, it sends data in the first state to be able to send the total step count if the user is done walking but has not imported the data yet. As mentioned, the piezoelectric is the main driver of the state machine, and therefore it reads values from the comparator multiple times in this state. If a certain amount of threshold values are met, then the state will advance. Otherwise, the state will remain at state one. The next state checks if the piezo voltage has gone back to low, meaning the foot has lifted off the ground. It checks this in a similar matter to the first state. The remaining states use similar methods but read accelerometer values from two different axis readings. Additional states were added to account for stair steps where a forward motion of the foot may not be evident. In effect, this caused a branching of states, where it is possible to reach the final state without going through every state. All of the threshold values were determined through extensive testing, as will be covered in Section 6. The seventh state is a brief state which simply updates the step count value, resets a flag related to Bluetooth

back to zero, calls the battery checking function, and adds a slight delay. It then immediately transitions back to state one. A function to read the battery voltage and send a percentage-remaining value to the microcontroller was written and is called in the seventh state. While this means the battery will only be read if the user takes a step, it was determined that this was better than constantly reading the voltage in the first state. The first reason is the loop happens so quickly that it sent too quickly to the Bluetooth module and the data would not send properly to the app. Secondly, if flags were used to only read the value once in the first state, this is the equivalent to reading once in the seventh state since the value will still be read once per step taken. However, reading in the final state eliminates a small amount of code needed for setting up the flag, making it more efficient to write in the seventh state. Using interrupts are far more efficient and can save energy by both less work for the microcontroller, but also for the ability to "sleep" the microcontroller in very small increments of time repeatedly. The end goal was to implement an interrupt method. However, this would have required writing a fundamentally different program from what was written for the state machine, and the amount of time this would've taken for both writing and testing the code would have been more than available from when the state machine was completed.

5.4.2 Application Design Decision

When creating a mobile application for the product, a mobile operating system in which to implement the software had to be selected. This decision was made using a variety of factors to determine the optimal choice (Table 11). Criteria are weighted on a scale of 1-5 with 5 being the most important. The solutions are given a number from 1 to 10, with 10 being the best. The design norms considered for software are also included in the decision matrix. The design norms are all weighted the same as they are all equally important to the decision. The design norms not listed were also considered when making the decision; however, they are not as applicable as the others and were therefore not included in the decision matrix.

Table 11: Mobile App Decision Matrix

		Weight	Swift (iPhone)	Android Studio	Both
Developer Considerations	Ease of Use	2	4	6	3
	Development Time	4	6	7	1
	Capability	4	8	8	9
	Availability	3	4	6	5
	Hardware Compatibility	5	8	8	8
	Cost	3	8	3	3
End User Considerations	Functionality	5	8	8	8
	Compatibility	4	9	8	10
Design Norms	Transparency	4	9	9	9
	Integrity	4	10	10	10
	Caring	4	9	9	9
	Trust	4	8	8	10
	Total		360	355	342

Ease of Use

Based on preliminary research, Android has more free resources readily available to aid an app designer. The primary programming language used to program for Android is more developed and older than Swift, which had its latest major version release in September of 2016 (Swift 3.0). This contributes to the amount of resources available online or in print. For this reason, Android is given a higher score than iOS for ease of use. Both design software take time to learn; therefore, the scores given are in the middle of the total range. Designing an application for both platforms would involve learning two separate languages, causing it to have the lowest score. Programming in a common language is another potential option; however, it is ruled out due to feasibility (see the end of this section).

Development Time

Development time of both Android and iOS depends mostly on the difficulty of the language. Because Android was found to be easier to use, it is given a higher score than iOS. Implementing the application for both platforms is given a score of 1, the lowest value, because it would take twice as much time in development and testing than just choosing one (assuming the programming will not be done through a common language as noted later).

Feasibility

Both Android and iOS devices are able to host applications that perform the same (or very similar) functions. For this reason, both Android and iOS are given the same score for capability. If there are any differences, implementing an application on both platforms would allow the team to provide the most completely capable software, giving the option of both the higher score.

Availability

Both Android and iOS have software available for download that can be used to design applications. The difference between the alternatives comes from the availability of equipment necessary for testing the application. All the members of the Smart Sole Team have Apple iPhones, so no further smart phone would need to be purchased if the app was developed for iOS. If the app were to be developed for Android, a smartphone for the purpose of testing would have to be added to the team's budget. Apple's iOS is given the higher score.

Hardware Compatibility

Both Android and iOS will be able to support the Bluetooth transmission the team hopes to implement. All alternatives are given the same score for hardware compatibility.

Cost

Because iOS Swift is available for free download through Apple's Xcode IDE and the members of the Smart Sole team own iPhones, using the iOS alternative would reduce the cost for testing and implementing the application. If the Android platform were to be used, an Android smartphone would need to be added to the budget, giving the Android alternative a lower score. Because cost of the iOS implementation is minimal, the alternative of doing both is given the same score as the cost of implementing Android.

Functionality

The requirements specified for the mobile application were created independent of the mobile operating system on which they would be implemented. It is assumed, due to the maturity of the operating systems

proposed as alternatives, that both platforms would be capable of supporting an application that meets all the defined requirements. All alternatives are therefore given the same score for functionality under these assumptions.

Compatibility

The score given for compatibility is based primarily on the ownership statistics of iPhones and Android devices. Because Android devices are purchased at a higher rate, Android is given a higher score. The option of both is given a perfect score as approximately 98% percent of new smartphone purchases are Androids or iPhones.[2]

Based on the results obtained from the decision matrix (Table 4), a decision was made by the team to move forward with the iOS Swift development of the application. The Android option was determined to be a suitable backup because of the small gap in score. This alternative would be selected if further research determined that iOS would be too difficult to implement. Both alternatives would be implemented if the iOS application is developed quickly, as time is the limiting factor that helped eliminate the option of both.

6 Integration and Testing

6.1 Software Development

The Smart Sole mobile application was developed using Apple's Xcode integrated development environment (version 8.2.1) and the Swift 3.0 programming language. These software and language versions were selected because they were the latest versions that could be run on the Mac computers provided by Calvin College. The development of the mobile application was done in parallel with the hardware design. Performing the development in parallel allowed both the hardware and software to adjust and adapt based on unexpected obstacles encountered during development.

The mobile application was developed from a basic single page application framework provided by Xcode. This framework was then edited to fit the requirements of the application. The base view controller was chosen to be a tabbed view controller because it allows the end user to view and choose from all the main viewcontrollers of the design.

The running tracker was the first tab of the final mobile application that was created. Development of the running tracker began with the creation of a new class for the method to discover the Bluetooth and a new class to "hold" the connected Bluetooth peripheral and all of the corresponding services and characteristics. These classes were generated using code from www.raywenderlich.com that was adapted to fit the functionality desired by the Smart Sole team. The running tracker view controller was set up to contain visualizations of the step count imported from the hardware as well as the goal step count established by the end user. A text field was also created to allow the user to input and edit their own step count goal. A separate progress bar was also implemented to allow the user to view the percentage of the battery in the insole device. Finally, an import button was included that, when pressed, would use the Bluetooth classes to import the step count and battery percentage saved on the insole microcontroller. If necessary, this button could also be configured to clear the microcontroller.

The second tab of the application was created to be an exercise tracker similar to other exercise trackers seen on the Apple App Store. This tab would be used by the end user to add and store exercises that they do in addition to running. Providing both exercise tracking and running tracking give the Smart Sole app an advantage over other applications that are only capable of doing one or the other. The exercise tracker was built with the help of an application creating tutorial provided by Apple Developer[9]. The application code in the tutorial was used as the base framework that the exercise tracker was then adapted from. The exercise tracker was designed to support adding and deleting exercises, editing additional exercises, and saving exercises even when the user exits from the app.

[9] "Apple Developer." https://developer.apple.com/. Accessed 9 May. 2017.

The final tab of the application is used to save the users running data. This tab was created to hold data the user saves in a table view. The data can then be accessed and viewed when desired. The data was saved in a data class which stored step count, goal step count, and the date the data was saved (using the NSDate class). A save button was imbedded in the navigation bar that would allow the user to save the data that was currently imported into the running tracker. The processing of saving this data would also clear the data in the exercise tracker. A method was implemented to allow the user to view data saved from previous days. Finally, an ImageView was created to notify the user if they had reached their goal.

The final most important and most time consuming part of the application design was error and crash prevention. Testing was run continuously during the development of the application to ensure that the entirety of the application functioned properly without crashing. The application was also tested when development had finished, ensuring that all error prevention methods were still functioning properly as expected.

6.2 Production Obstacles

6.2.1 Bluetooth Communication Obstacles

There were several Bluetooth LE network obstacles handled by the team during the course of this project. The first involved the setup of the UUIDs for the characteristics. With custom UUIDs, many particular protocols must be followed to create a working service. While attempting to create the characteristics, errors appeared saying the characteristics couldn't be established. In addition to time finding the correct commands to define services and characteristics, the team discovered that the last 7 bytes of the characteristics and the service had to match for them to relate.

The second obstacle was overflow problems involving the characteristics. While initializing characteristics, a specified number of bits needs to be given. The team set the battery bit numbers to 7. The team felt that this would allow a maximum value of 100 to be reached; however when trying to print the initial value to serial to see if it was set, the printed value read 13. After studying the error, the team determined that an overflow error may have occurred. The team allowed the default 20 bits to be set for the characteristics but altered the mobile application to not allow the value to be set higher than 100.

One final bug that was unresolved was when the battery value was updated, the application would always get a number between 49 and 51 percent. There may be multiple reasons for an error like this to occur. This problem was not resolved due to time constraints.

6.2.2 Software Obstacles

Many obstacles were encountered while developing the mobile application and software portion of the Smart Sole product. The first obstacle encountered when initially beginning the development of the software was the unavailability of computers to develop the software on. Because Xcode is an IDE developed and published by Apple, it is only available for download on systems running a version of OSX (Apple's operating system). Due to legal reasons, the team was unable to run a virtual operating system on the Windows computer provided by the engineering department. Two possible solutions involved working on MacBooks lent out by Calvin College's AV Department and working on the Mac computers in ITC (Calvin IT Department's computer lab). The first option was ruled out when the laptops were found to not have the hardware necessary to develop the application efficiently. The Smart Sole team chose to develop the application on the computers in ITC which limited the team's working hours to the hours when ITC was open. This forced the team to be proactive in setting deadlines as time would be a major limitation.

A second major obstacle occurred when the team discovered that the ITC computers were running an outdated version of Xcode that did not support Swift 3.0. The software on the computers had not been updated since the beginning of the Fall 2016 semester causing Xcode to be multiple versions out of date. Because all of the free resources provided by Apple online were for Swift 3.0 or higher, the team would have to spend additional time learning the language. Furthermore, if any errors were encountered, the team would have no databases to help debug and solve the error. The team determined it would be unfeasible to design the application in the outdated version of Xcode due to the previously mentioned time limitation. This issue was resolved through contacting the IT department and requesting an update of the software. Development of the application had to be stopped during this process.

The final major software obstacle involved connecting the mobile application to the hardware through Apple's Core Bluetooth framework and the Smart Sole team's custom Bluetooth service. As mentioned previously, the application's Bluetooth classes were based on code found online; however, because the application of the Bluetooth is not the same, the code had to be adjusted. The team kept the main framework of the classes and edited the function calls, creating their own distinct Bluetooth classes. These classes had to be paired with the hardware Bluetooth. Many of the hardware-side obstacles involving this can be found in Section 6.2.1 Hardware Obstacles. The primary issue from the application-side was the timing and method of connecting to the hardware. When the application loads, the Bluetooth begins to connect with the device. A delay function had to be added to the Bluetooth connection code to ensure that the loading of the main view controller did not interfere with the Bluetooth connection function. A second issue arose when trying to read the step count value send by the hardware to the application. The data received via Bluetooth was not in a form that the application found useful; instead,

the data was saved in a “Data” (class) object. The team could not find any defined protocol to convert the Data type to a UInt16 or any other simple data type. An extension of the Data class was written to resolve this problem. The extension took the Data object and converted it to any basic data type by accessing the object’s bytes.

Various bugs and errors were encountered when developing the application. All bugs or faults were recorded in a spreadsheet with the date they were encountered and a description of the problem. A table version can be found in Appendix B. Once the faults had be debugged, information about the cause and solution was recorded. This was done to help future programmers understand the errors that the team encountered and why specific resolutions were implemented. The documentation could also help future programmers debug their code if they encounter similar faults.

6.3 The Final Prototype

6.3.1 The Insole Device

The Smart Sole insole prototype is a foam insert similar to many cushioned insoles on the market but housed inside are the hardware elements used to track step count. Components are housed within cutouts of more dense purple foam and resting on softer green foam below. A tightly grouped comparator, microcontroller, and accelerometer are housed along the inside of the insole, along where the arch of a foot would rest. This placement, shown in Figure 7 below, is due to the lessened forces along the components. Due to the arch along the foot, contact is limited and therefore the components are more likely to remain protected. A piezo element is placed near the ball of the foot. This decision was made to allow the greatest amount of force be put on the piezo. The heel was considered for piezo placement, but was determined inconsistent due to a number of people who only make contact on the front of their foot while jogging or running. Finally a 3.7 V rechargeable battery was added on the outside with a deep cut to keep it level. The original sole of the shoe was added on top to cover the components. An added foam piece was added to the original insole over the area making contact with the microcontroller for added padding along the main components.

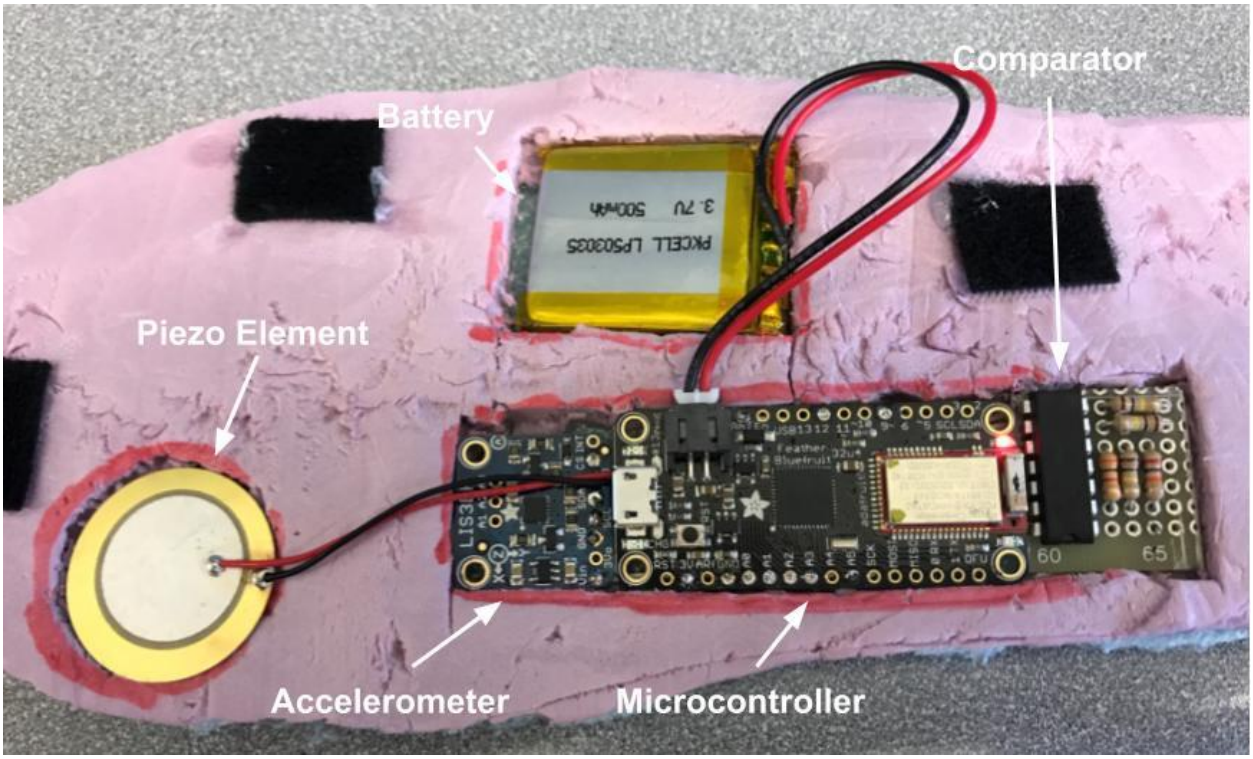

Figure 7: Labeled Insole Prototype

6.3.2 The Mobile Application

As mentioned in 6.1.2 Software Development, the Smart Sole mobile application was a tabbed application that consisted of three main tabs: a Running Tracker, an Exercise Tracker, and a table view to store Saved Data. The functionality of each of these main components will be covered in detail. The application code is not included within this document. This section assumes a basic understanding of the Swift programming language and the Xcode code development environment.

Running Tracker

Creating an app to import and store step count data is one of the primary objectives of the Smart Sole team. The final storyboard of the Running Tracker tab can be seen below in Figure 8. This view controller is also imbedded in the main tab view controller.

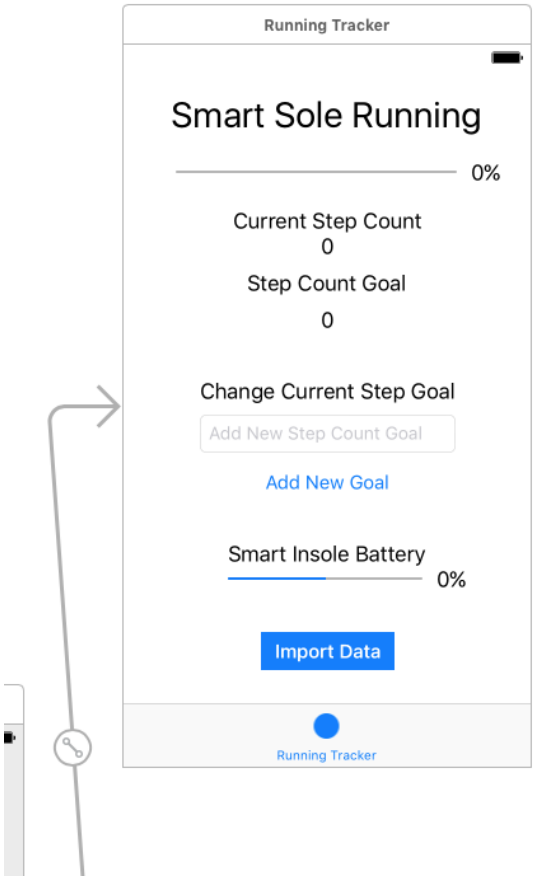

Figure 8: Running Tracker View Controller

The view controller consists of three major sections: the step count information, the battery information, and the import button. The first progress view (progress bar), above the "Current Step Count" label, displays the progress of the user towards their step count goal. This value is a percentage based on the "Current Step Count" label relative to the "Step Count Goal" label. If either of these labels changes value, the progress bar is updated along with the percentage label next to the bar. The value label under the "Current Step Count" label is updated only when the user imports data via the "Import Data" button; however, the "Step Count Goal" label can be updated at any time by the user. This label is updated via the text field and the "Add New Goal" button. The text field is restricted to a number pad to prevent invalid entries (such as Strings). The user can also close the keyboard at any time by touching anywhere else on the screen. Once a value is entered, the user must press the "Add New Goal" button to update the value in the "Step Count Goal" label. The text field is cleared and reset to display the default text when the goal value is successfully updated. The user is notified if they enter an invalid entry or press the button without inputting any value (Figure 9).

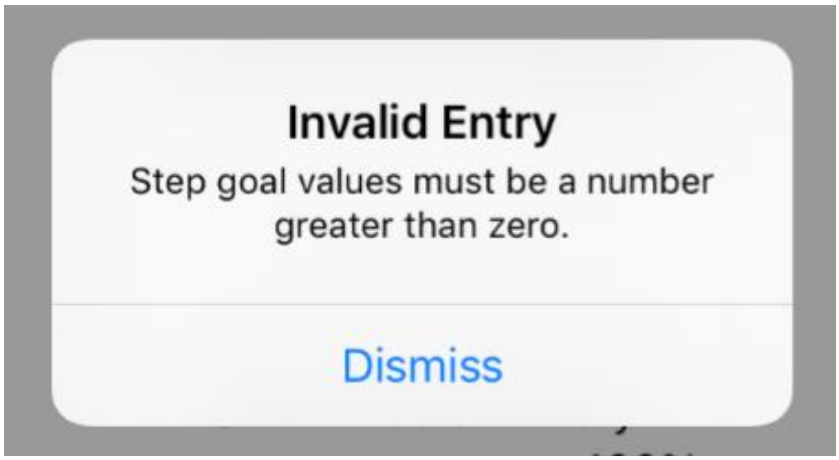

Figure 9: Invalid Entry Warning

The second section of the Running Tracker is the battery life tracker. The battery life tracker was implemented notify the user of the current battery charge of the device since the battery is not easily accessed from within the shoe. The battery life is displayed as a UIProgressView object with and corresponding label. The value of the progress bar and the label is updated whenever the "Import Data" button is pressed. The color of the progress bar is set to change based on the percentage of battery remaining.

The final section of the Running Tracker is the "Import Data" button. Two classes were created to perform the desired Bluetooth functions: BTDiscovery and BTService. The first, BTDiscovery, begins the process of discovering Bluetooth peripherals that are advertising within the range of the phone. Once the correct peripheral is discovered, a BTService object is created to store the connection the peripheral's service. When pressed, the "Import Data" button calls the readSteps() and readBat() function on the BTService object. These functions read the value advertised by the corresponding Bluetooth characteristic and convert the value into an integer that can be used by the application. After this read process is completed, the "Current Step Count" label is updated along with both progress bars. If the button is pressed when the application is not connected to the insole device, a notification is sent to the user to prevent an application crash (Figure10). This data remains on the Running Tracker view controller until it is either replaced with a new value or erased after saving.

Figure 10: Connection Error Warning

Exercise Tracker

The Exercise Tracker tab of the mobile application was created in an attempt to make a product that can attract users that have fitness interests beyond just running. This view controller is also imbedded in the main tab view controller. The final storyboard of the Exercise Tracker tab can be seen below in Figure 11.

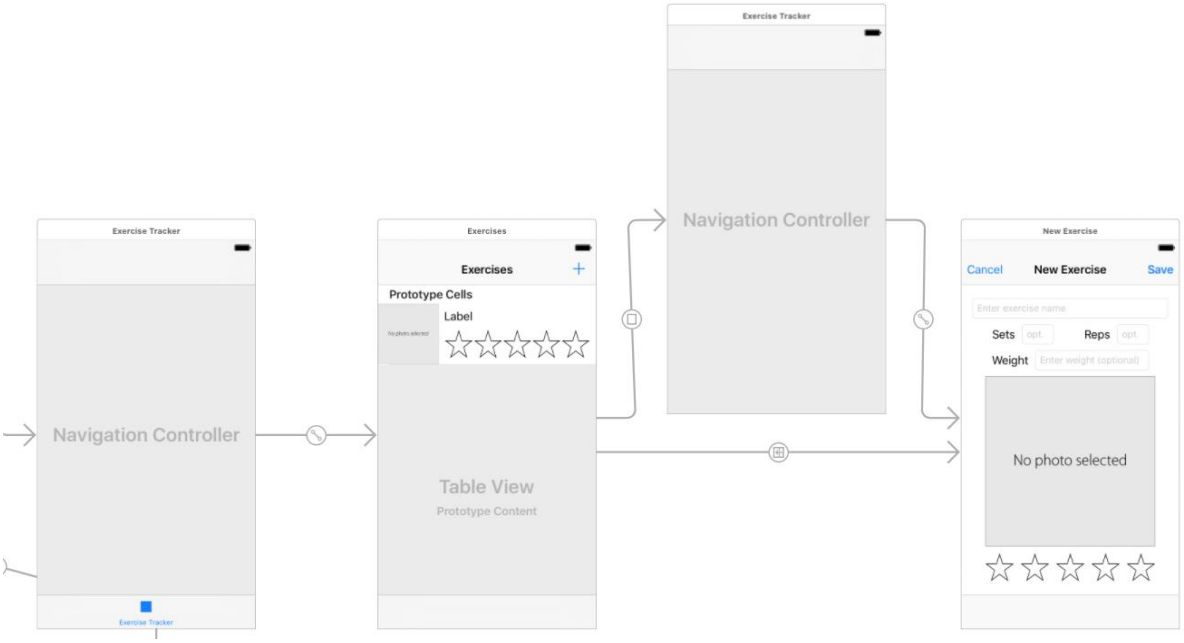

Figure 11: Exercise Tracker Storyboard

The Exercise Tracker is created by embedding a navigation controller (leftmost view controller) within the original tabbed view controller. The navigation controller is used to control the navigation between the table view controller which stores the exercise (left center view controller) and the rightmost view

controller which is used to add and edit meals. The “ExerciseViewController” (rightmost controller) is connected to the table view controller via two segues because it can either be presented modally (adding an exercise) or pushed (viewing an exercise). The modal presentation method embeds the ExerciseViewController in another navigation controller (right center). Care is taken when moving between view controllers to prevent any errors or crashes.

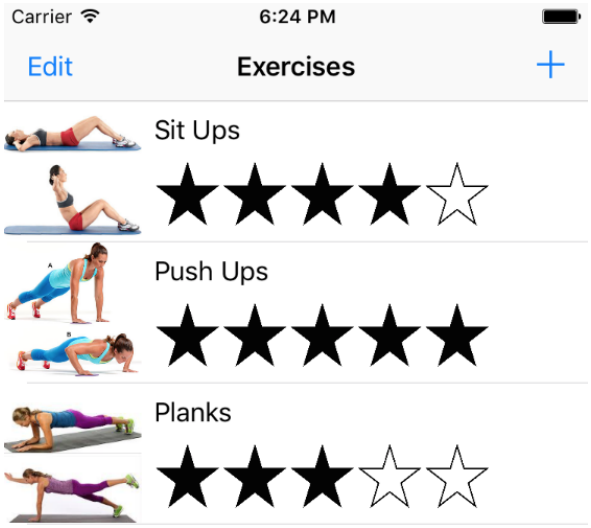

Figure 12: Exercise Table View Controller

The table view controller is the first important part of the Exercise Tracker. The table view controller contains cells of a custom class that display a photo, a name, and a rating based on the user input. Tapping on an exercise allows for editing. The user can also create a new exercise using the “+” button in the top left corner of the view controller. The exercises can also be deleted by swiping horizontally or pressing the edit button. A screenshot of this view controller is seen in Figure 12.

The second major view controller of the Exercise Tracker is the ExerciseViewController. This view controller is used to edit or add exercises based on how it was presented. If the user is adding a new exercise, the view controller is presented using modal presentation. The view controller is presented with empty fields and the default image (Figure 13). The user is then able to fill in each field with the corresponding information. All fields are optional with the exception of the name field. The save button in the upper right corner is disabled until the user enters and exercise name. When the name is entered, it appears in the navigation title and the save button is enabled. The user is able to insert an image of the exercise by tapping on the image view. The image view is set to access the phone’s photo library.

Additional functionality can be added to support taking new photos to use. The stars along the bottom of the view controller are created using a custom class called “RatingControl”. This class behaves similarly to the rating systems of many other apps including the iPhone’s default music player. Pressing and releasing on a star fills in all stars from the left, up to and including the star that was pressed. If the rightmost star that is filled is pressed, the rating returns to zero and all stars are reset. The remaining text fields on this view controller are configured to accept any string input and return to the default text when no entry is made. When the save button is pressed, all information on the view controller is saved to a custom “Exercise” class. The cancel button can be used to return to the table view controller at any time.

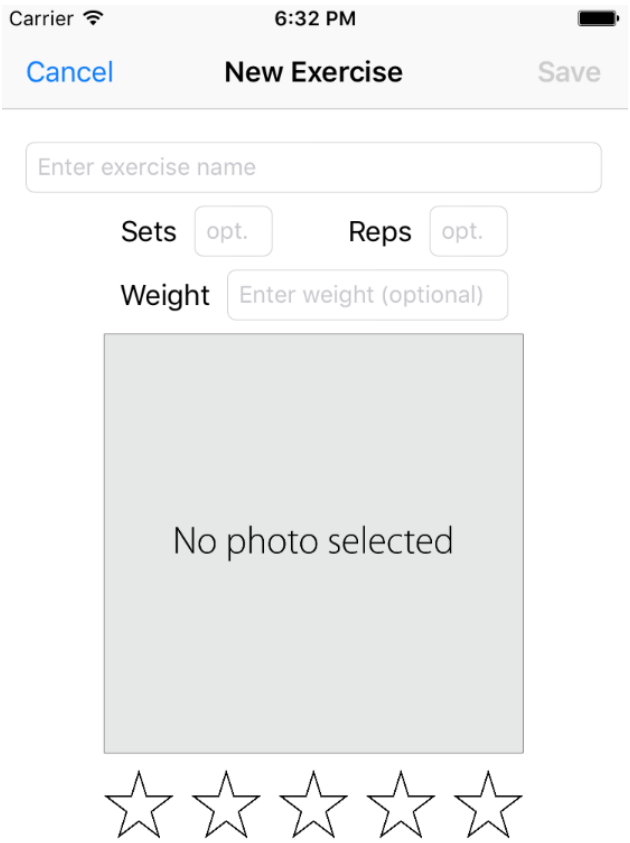

Figure 13: Adding an Exercise

The ExerciseViewController can also be presented using push presentation. This occurs when the user taps on an exercise in the table view controller. The view controller is presented in a way that allows the user to view and edit their preexisting exercise (Figure 14).

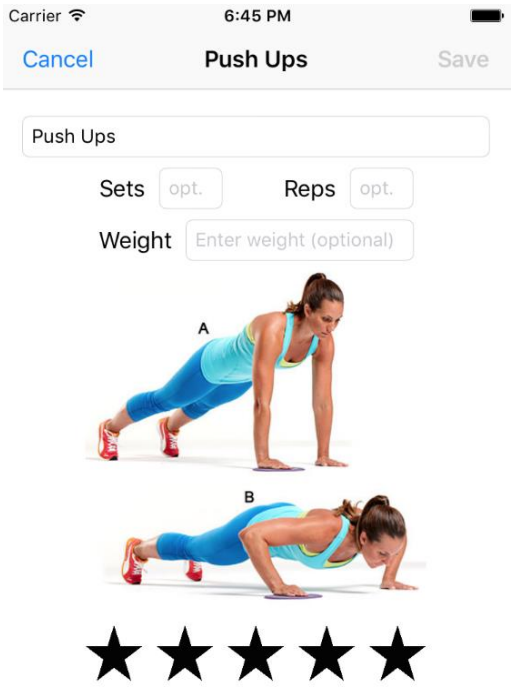

Figure 14: Viewing and Editing an Exercise

The save button is only enabled when changes have been made to the exercise. The save button is configured to replace the old exercise when saving to prevent duplicates of the same exercise. The cancel button can be pressed at any time to return to the table view controller. The code for the cancel button determines how the view controller was presented and returns safely to the previous controller.

Saved Data

The final tab of the application is used to save and store running data. This tab is configured similarly to the exercise tracker with a table view controller that is embedded within a navigation controller. The segues between view controllers are handled similarly to the segues in the Exercise Tracker. The Saved Data tab also has a view controller to save and view running data (RunningViewController); however, the editing capabilities have been disabled. The user should not need to edit previous running data, justifying this decision. The final storyboard of the Exercise Tracker tab can be seen below in Figure 15.

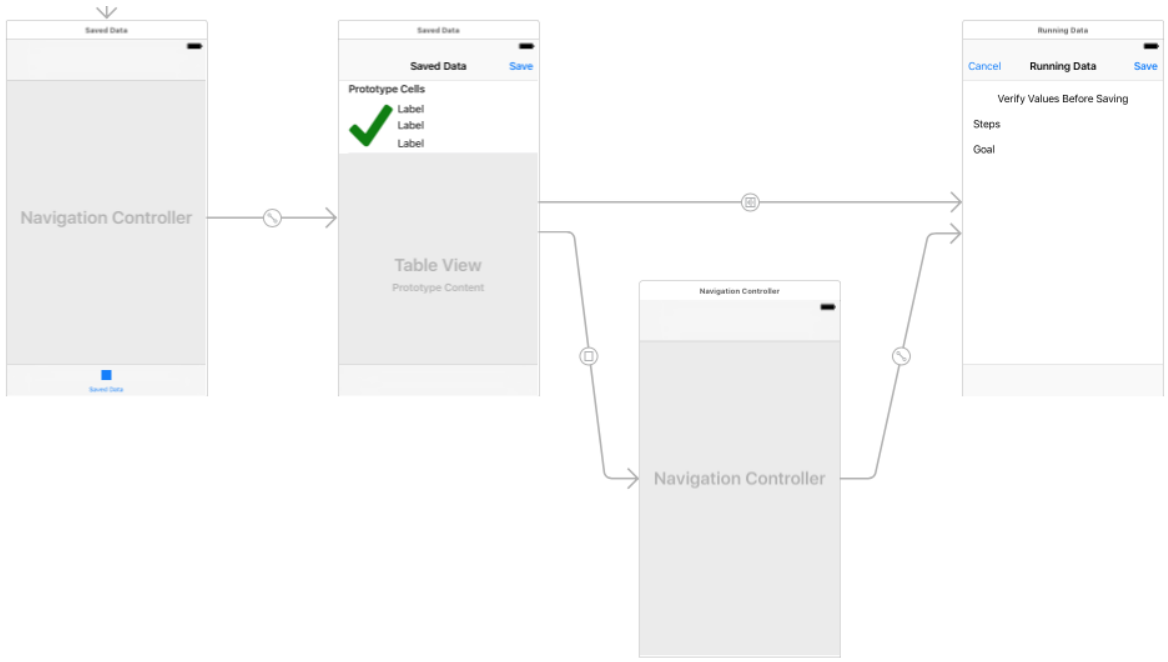

Figure 15: Saved Data Storyboard

Just like the Exercise Tracker, the Saved Data tab uses a table view controller with custom table cells; however, the cell contents differ. The RunningTableViewCells contain three labels and an image view. The first label displays the date the data was saved, the second shows the amount of steps taken, and the third shows the goal step count. The contents of the image view change based on whether the step count goal was met. If the user has met or exceeded their step count goal, the image view displays a green checkmark. If the user has not met their step count goal, the image view is left white. This system allows the user to easily determine whether they have met their goals. The navigation buttons of the table view controller function similarly to the Exercise Tracker. The user can delete data by swiping horizontally or pressing the edit button in the upper left corner. The user can also save the data that is currently on the Running Tracker screen by pressing the save button in the upper right corner. A screenshot of this view controller with preloaded sample data can be seen in Figure 16.

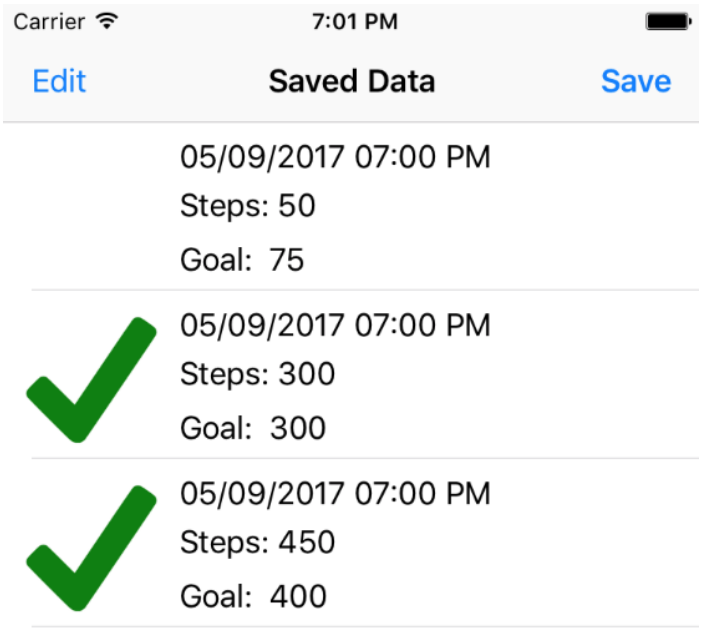

Figure 16: Saved Data Table View Controller

The Saved Data tab contains a view controller (RunningViewController) that is used to save and view step count data. The view controller is presented modally when saving data. The RunningViewController takes the step count and goal step count data from the Running Tracker and displays it for the user to review prior to saving, as seen in Figure 17.

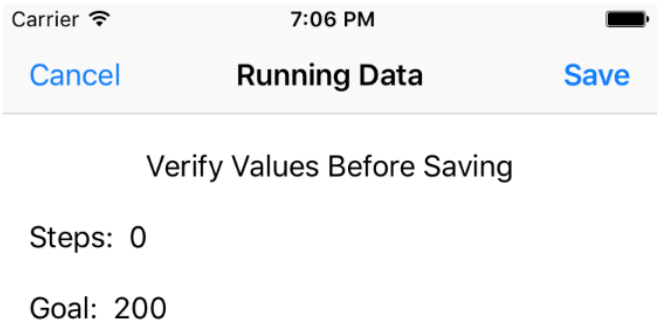

Figure 17: Saving Running Data

The data can be saved permanently by pressing the save button. The data is saved in a custom class that contains the steps, the goal, and the date and time the data was saved. The date is determined using a NSDate object that uses the built in clock hardware of the iPhone. Pressing the save button will also reset all values on the Running Tracker view controller. If the user does not wish to save the data, they can return by pressing the cancel button.

If the user taps on previously saved data, they are able to view it using the RunningViewController (push presentation). All aspects of the data can be reviewed as seen in Figure 18.

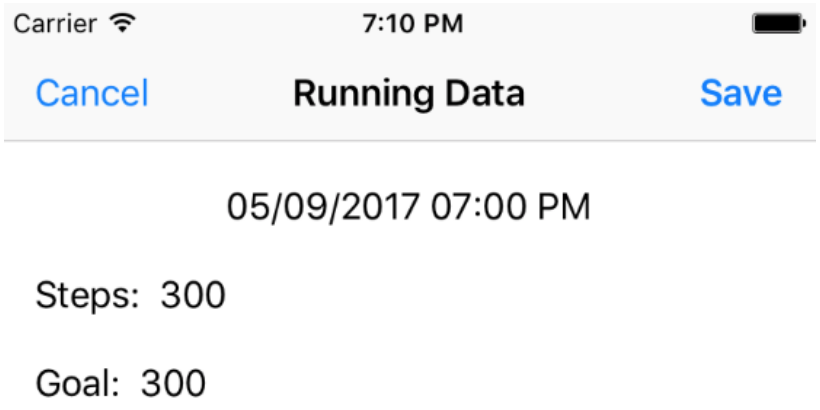

Figure 18: Viewing Running Data

There are no fields that can be modified by the user, effectively preventing them from changing the data. If the user does press the save button, the data is safely returned to the table view controller. The cancel button can also be used at any time to return to the table view controller. Both the Saved Data tab and the Exercise controller support data persistence. Even if the user exits the Smart Sole application or shuts down their phone, the data remains in the state that they left it. The sample exercises and running data can easily be removed by the user. Because of data persistence, they will not return when the user reloads the app.

6.4 Testing and Results

The hardware that was tested included the piezoelectric and comparator portion, the accelerometer portion, and the Bluetooth module. These were tested in several stages. Since both the piezoelectric and accelerometer were used to collect data, programs were first written for them separately and tested separately before they were combined and tested. A program for the piezoelectric was written and tested originally by simply compressing the piezo with a finger and releasing. Different amounts of pressure was used for different amounts of time. An LED was attached and instructed to light up for a brief period of time whenever the piezoelectric produced a voltage spike of sufficient magnitude, based upon a level set with the comparator. The serial monitor, which is part of the Arduino IDE, was useful in analyzing the values being read by the microcontroller. Additionally, an oscilloscope was used to visualize the length and magnitude of the pulse. Rapid compression was performed as well to determine the response time. Once it was determined the piezoelectric would produce consistent results under compression, the piezoelectric circuit was then placed on the insole of the shoe and tested when walking, with the

microcontroller taped on top of the shoe to see the LED light up upon each step. The threshold voltage had to be recalculated and a new voltage divider constructed for the comparator until the piezoelectric response was consistent upon every step. Adjusting the amount of times the input values were read and checked was also necessary. The comparator was tested in conjunction with the piezoelectric. An oscilloscope was used to test and check the comparator output. A separate program for measuring accelerometer values was also created. The serial monitor assisted in analyzing the different acceleration values for each axis, as well as the effect gravity had when the accelerometer was tilted a certain amount of degrees. A program to read and store a large array of acceleration values each step was also created to get a better feel for the threshold level needed, at least for ideal steps. The accelerometer device was tested in a similar manner to the piezoelectric testing. The microcontroller, along with the accelerometer and LED, was placed on top of the shoe with the accelerometer in the same position it would be when in the insole of the shoe. The LED would light up when a certain threshold value was met. Finally, the two components were put together in a final program. Additional states and logic was needed to test these components together, as was explained previously in section 5.4.

The mobile application was tested for errors and bugs during development. Testing was performed by determining all potential situations that could occur for any given action. For example, if the application contains a text field, a user is able to enter a wide range of inputs including Strings, Floats, Ints, and many other potential data classes. All of these scenarios must be accounted for to prevent the application from crashing. No formal test suite was developed as there is no simple way to test all potential cases. Once the application was complete, it was distributed to the members of the Smart Sole team for use and validation. Any errors encountered during this time were recorded and resolved. This final status of all bugs and errors can be seen in Appendix C.

The final prototype was tested for accuracy under multiple conditions. Much of the testing resulted in either adjusting threshold values for the accelerometer or the number readings for each state. The first condition tested was the ideal case. This consisted of walking at a moderate pace with no stops on a hard and level surface, including both school halls and sidewalks. Since the individual hardware components were previously tested and adjusted separately under the ideal case, most of the thresholds were relatively reliable when they were combined and integrated together. The most major adjustments included timing changes. The results for this case are shown in Figure 19 and Figure 20. The accuracy of the ideal case neared 100 percent under testing. The second test condition consisted of jogging on a hard, mostly flat surface at a quick pace. Again, this included school halls as well as sidewalks and the road. The accuracy of the jogging case had a very low outlier at 65.4 percent. With some minor adjustments in timing, the accuracy was consistently close to 100 percent. The results are shown in Figure 21 and Figure 22. The third test condition consisted of walking at various paces in changing terrains and environments. This included walking in the grass, woodchips, paths, as well as various slopes. The accuracy of this test was

near 100 percent also, as shown in Figure 23 and Figure 24. The largest concern was the piezoelectric sensing in the softer conditions, especially low areas that were still soft from previous rain. However, the piezoelectric sensing device was sufficient in picking up steps. The accelerometer values were close to the ideal walking condition, and the slopes and related gravity effects were previously accounted for in the design. The final test condition was the non-ideal case, which most closely emulated real life conditions. This test included walking up and down stairs, stopping at starting, and partaking in everyday activities such as stopping to get a drink of water or turning around. In addition, some portions of the test included slow stepping, rapid stepping, and other methods to attempt to trick the sensing devices. The accuracy of the tests was close to the desired goal of 95 percent accuracy at 94.3 percent. The accuracy improved in further tests as the device was optimized, and continued tests would raise the average above the goal of 95 percent. The testing results are shown in Figure 25 and Figure 26.

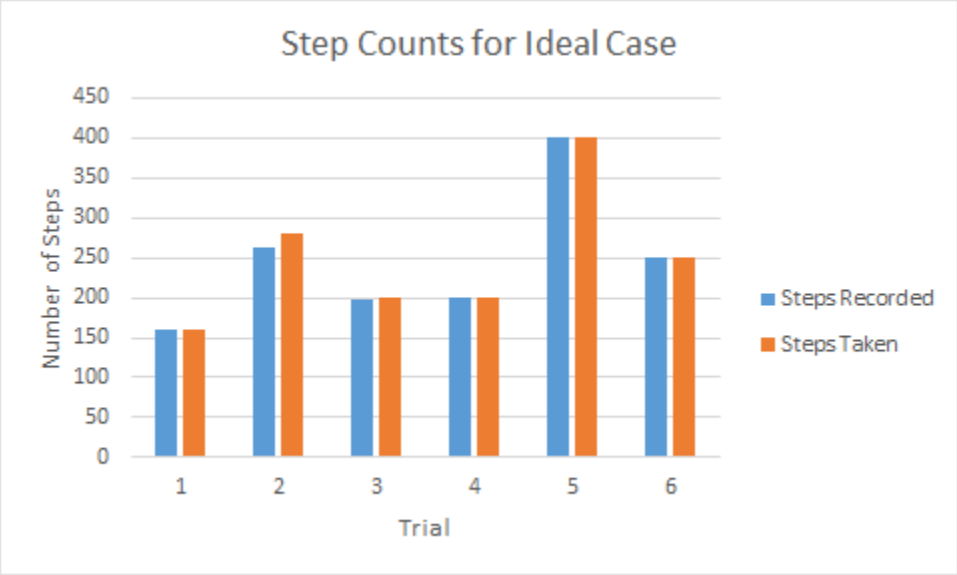

Figure 19: Step Counts for Ideal Case

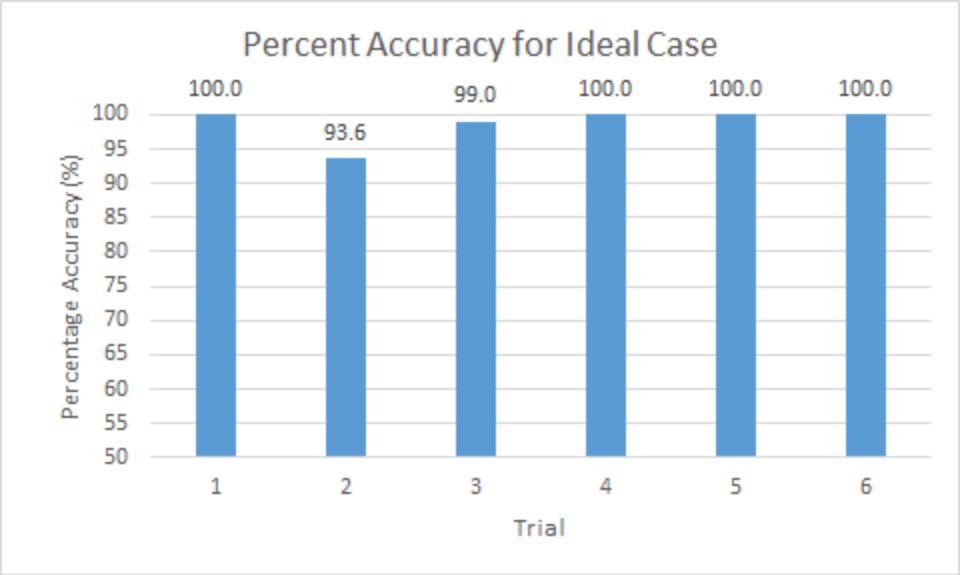

Figure 20: Percent Accuracy for Ideal Case

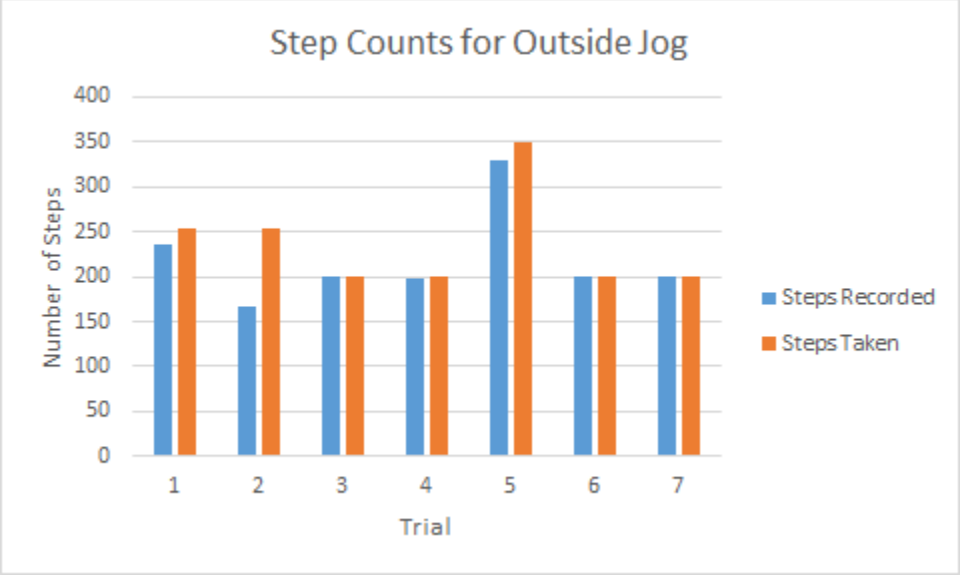

Figure 21: Step Counts for Outside Jog

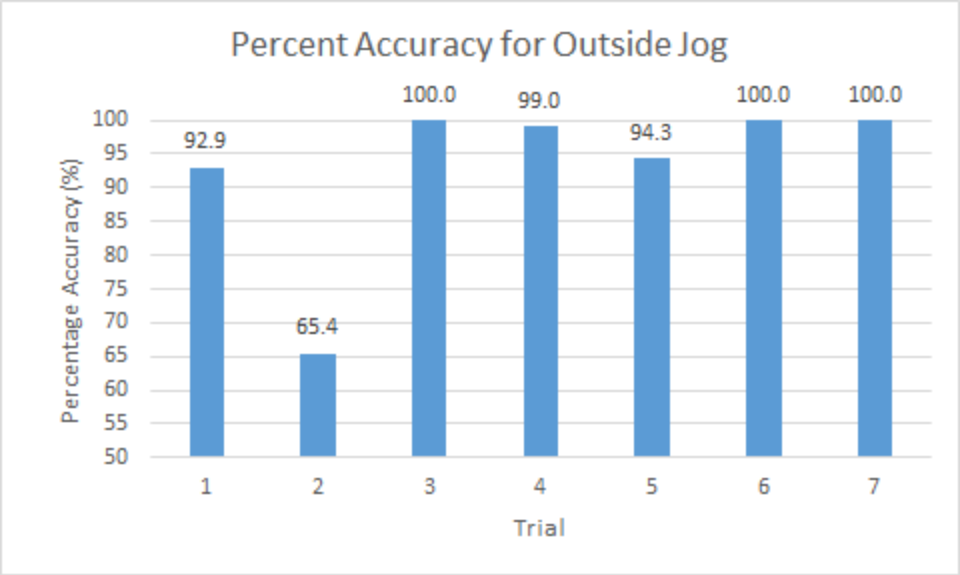

Figure 22: Percent Accuracy for Outside Jog

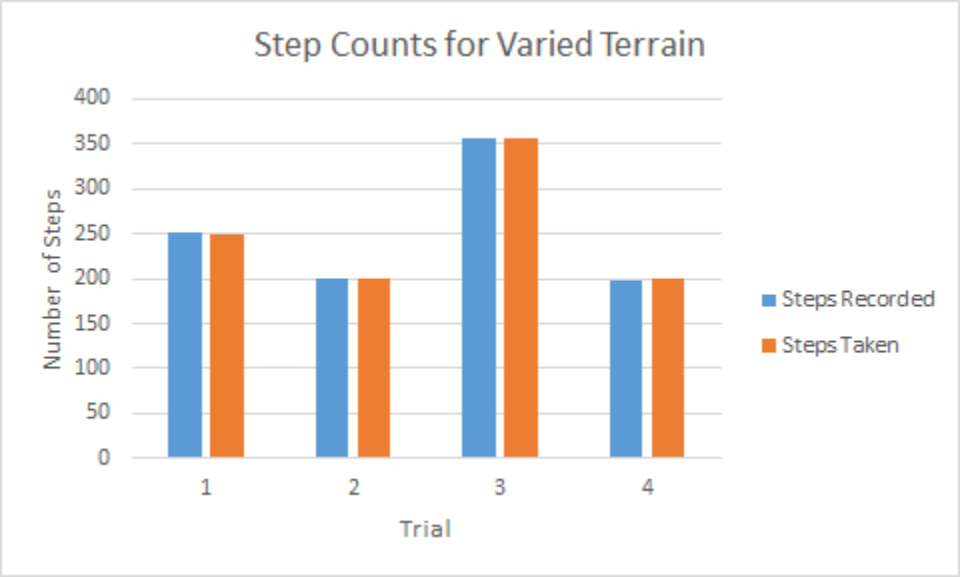

Figure 23: Step Counts for Varied Terrain

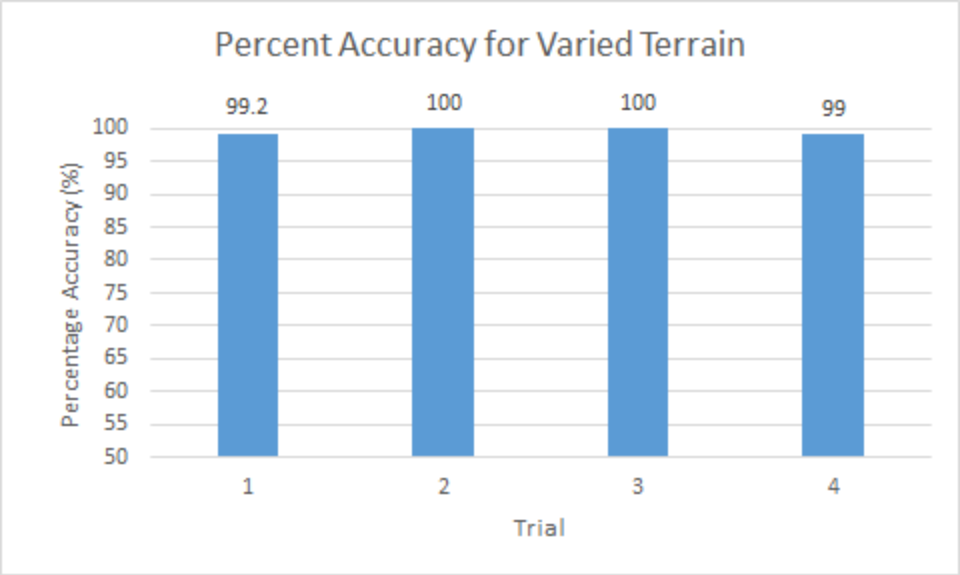

Figure 24: Percent Accuracy for Varied Terrain

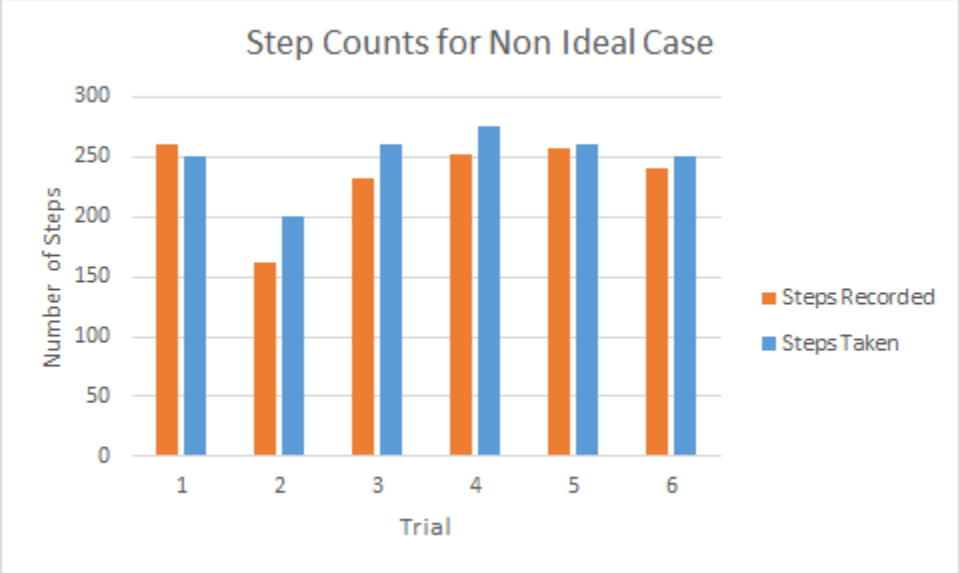

Figure 25: Step Counts for Non Ideal Case

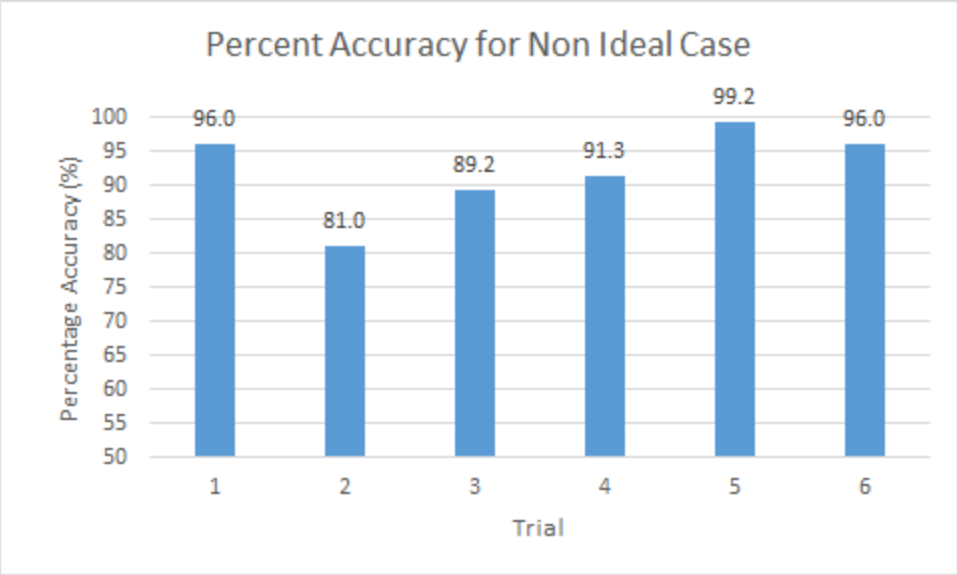

Figure 26: Percent Accuracy for Non Ideal Case

Overall, the product was very accurate in most ideal conditions, and the goal of 95 percent accuracy was nearly reached for the realistic case as well. Removing the outlier from the data, the average accuracy for the realistic case was 94.34 percent. The device was designed to avoid errors with stairs and other obstacles as well. The leading causes of inaccuracies were found to be walking down stairs, stopping and starting, and turning around. Walking down stairs was more difficult to program than walking up stairs due to the effect of gravity. While walking up stairs will lead to a large decrease in the downward acceleration of gravity for a brief period of time, walking down the stairs does not always have the opposite effect. In fact, in many cases, the downward acceleration rarely changes significantly at all. The source of inaccuracy from starting and stopping is mostly due to the potential to either double count if the same foot moves for starting and stopping, or miscounting due to if the opposite foot moves multiple times instead. Finally, turning around can lead to inaccuracies due to minimal changes in acceleration in either of the two sensing axes.

6.5 Analysis

6.5.1 Hardware Analysis

The device hardware requirements consisted of functional requirements and performance requirements. All of the functional requirements were met, and most of the performance requirements were met. For those that weren't, they were generally very close to the goal or met the requirement under certain conditions.

Functional Requirements

HF.1 The device shall be able to record and export step count data.

The requirement was met. The microcontroller counts and stores the step count data. Each cycle in the state machine sends the data to the Bluetooth module, which can then be sent to the app whenever the user imports that data from their smartphone.

HF.2 The device shall not cause any additional safety concerns such as cuts, blisters, electric shock, battery corrosion, or anything else that might harm a user.

This requirement was met. The device is located in a foam insole beneath the actual shoe insole with multiple layers of protection. The hardware would be coated with epoxy for additional protection from sharp edges, although the decision was made to not coat the actual prototype in order to save the microcontroller for future projects.

HF.3 The device should be designed to minimize power consumption to a level comparable to other similar products.

This requirement was met. The hardware used low energy components as well as the smallest feasible microcontroller to limit power consumption. There is potential to lower power consumption even more by using interrupts and custom parts as mentioned, but the power consumption level is comparable or even better than many other similar products.

HF.4 The device should be able to fit into a shoe without causing discomfort to a walker or runner.

This requirement was met. The device fits into a foam insole with cutouts for the hardware. The shoe's original insole is placed on top of the foam insole containing the hardware. The hardware is strategically placed near the arch of the shoe, where the least amount of force from each step is generated. This allows for the device to be nearly undetectable to the user.

HF.5 The cost of the product should be less than that of other comparable products on the market.

This requirement was met. Most comparable devices cost close to $100, while the Smart Sole device costs $45.

HF.6 The device should be water resistant and usable in wet and humid conditions.

This requirement was met in theory. The waterproofing technique was researched and the materials required were purchased. Coating the circuit in epoxy and covering the battery connection was determined to be an effective waterproofing technique. However, as mentioned previously, the team

decided not to coat the prototype with epoxy. The team decided to save the microcontroller in case it would be used in the future for different applications.

Performance Requirements

HP.1 Step count error should be less than +/- 5%.

This requirement was met in some situations. In each testing condition except the non-ideal case, the step count error was less than five percent. In ideal cases, the step count error was nearly zero. The step count error was 5.66 percent in the most realistic case, although the later trials had less error after design improvements were made in the code.

HP.2 The internal battery should be able to last at least eight hours in use.

This requirement was met. The battery selected has a large capacity of 500 mAh, and can last well over 10 hours, based upon the limited battery testing and forecasting.

HP.3 The internal battery should charge in under 2 hours.

This requirement is met if the user charges the battery each day. The battery itself has such a large capacity that it takes 5 hours to completely charge it based upon the current charger with a charging rate of 100mA. However, after a full day's use, the battery will not be completely drained.

HP.4 The rechargeable battery should last the lifetime of the product (2 years).

This requirement was met, based upon the data provided for the rechargeable battery that was ordered. The battery should last more than two years based upon the number of recharges it contains, which is stated online for the battery. In addition, most lithium ion batteries last longer than what is rated, although at a dimensioning capacity.

HP.5 The hardware should enter an idle low power mode if not in use to conserve battery life

This requirement was met, although there are certainly areas to improve this even more with the use of interrupts as mentioned. Still, when the device is in idle, the program will remain in state one where there is only simple looping and checking for a piezoelectric spike. This means that there is no data sending, no accelerometer reading, or any other work-heavy process. Therefore, when idling, the device uses less power.

6.5.2 Software Analysis

The final mobile application met all of the requirements established by the Smart Sole team. The following section reviews the requirements and how they were met.

Interface Requirements

SI.1 The user interface of the mobile application should be intuitive and usable by any customer over the age of 8.

This requirement has been met. The software minimizes the amount of necessary input by the user and maintains a simple and straightforward interface. People that were shown the application were able to use it to perform its basic functions without the need for instruction.

SI.2 The user should be able to view their step count data and exercise data.

This requirement has been met. The application implements table view controllers to manage the user's step count and exercise data. The information is displayed in a concise and clear manner.

SI.3 The user should be able to view data from up to 90 days prior, assuming a usage of once per day.

This requirement has been met and exceeded. The current build of the application supports total data persistence. Data will be saved until the user removes it or removes the app. The data is stored in simplified data classes that should not use up a significant portion of phone memory.

SI.4 The user should be able to create and save their own exercises in a predefined and intuitive data format.

This requirement has been met. As mentioned previously, exercises can be saved permanently within the application. The exercise data class contains all intuitive fields that are necessary to store many forms of exercise. This includes exercise name, sets, reps, weight, an image of the exercise, and a rating. If production on the application continues, a method can be created to sort the data by a specific field.

SI.5 The application should provide some form of interactive achievements/statistics to encourage regular usage.

This requirement has been met. Currently, the only form of "achievement" is the green checkmark that is displayed next to saved data when the step count goal has been met. More achievements or statistics may be added if production continues. This requirement was considered one of the least crucial due to its nature.

Functional Requirements

SF.1 The mobile application shall be able to export data from the Bluetooth processor in the hardware.

This requirement has been met. The application successfully imports step count and battery data from the insole hardware when the “Import Data” button is pressed

SF.2 The mobile application should have an idle mode that uses minimal processing power and phone resources.

This requirement is indirectly met. The step count data is stored within the insole microprocessor. The data only needs to be exported when the day is completed; therefore, the mobile application does not need to be on, except when the user wishes to import and save data.

SF.3 The mobile application should store user data for up to 90 days.

This requirement has been met and exceeded. The current build of the application supports total data persistence. Data will be saved until the user removes it or removes the app. The data is stored in simplified data classes that should not use up a significant portion of phone memory.

SF.4 The mobile application should be able to store data during the run or after.

This requirement has been met. The timing of the data importation and storage is dependent solely on the user. The user is able to import the data at any time, as long as the Bluetooth connection can be made.

SF.5 The mobile application should be fun and easy to use.

This requirement has been met. As mentioned in SI.1, the application is easy for anyone to use.

SF.6 The mobile application should save user data locally.

This requirement has been met. The application saves all data on the user’s phone. No cloud storage is used. The app is designed this way to protect the user’s identity and promote confidentiality.

SF.7 The mobile application should be secure and prevent third party access of customer information.

This requirement has been met. All data acquired within the application is saved locally on the user’s device. It cannot be accessed by any other applications as these other applications do not have access to the datapath on which the data is stored. The only was a third party can access this data is by hacking into the device’s memory.

Performance Requirements

SP.1 Data should be exported to the mobile application in less than 5 seconds.

This requirement has been met. The data is imported to the application almost instantaneously (assuming the Bluetooth is connected). The time to import data was measured to be under 200ms from when the "Import Data" button is pressed. The Bluetooth connection process takes 3 seconds from when the user first opens the application. The total time from application startup to data importation is ~3 seconds.

SP.2 Application performance should be consistent with what is expected by users.

This requirement has been met. The expectation is that all features should function properly and the application should not crash when undergoing normal operating conditions. This was verified through the testing and debugging process. All bugs that remain are listed in Appendix B.

7 Business Plan

7.1 Marketing Plan

While a step counting device is an interesting idea, fitness tracking devices have been around for many years. They have become very popular only within the past few years though, thanks in part to the rise of smartwatches and the rising awareness of the benefits of exercise. Many new devices come out each year, most with similar features and accuracy. The Smart Sole device provides the same functionality but at a cheaper price, a more accurate measurement of step count, and no requirement to wear a watch or wristband as most other devices require.

7.1.1 Competition

Many companies produce a variety of fitness trackers. The four main competitors in the fitness tracker market include Fitbit, Garmin, Misfit, and Apple. Each of these companies, with the exception of Apple, release a wide variety of fitness trackers.

Fitbit offers sports watches and wristbands starting as low as $100. The Fitbit Flex 2, released in the fall of 2016, is the cheapest product, starting at $100 and offering basic activity tracking.[10] It does, however, require the user to wear the fitness band.

Garmin offers sports watches at prices ranging from $100 to over $450. The cheapest model Garmin offers is the Garmin vivofit® 3, which starts at $100, offers basic fitness tracking and can connect to a capable smartphone.[11] The vivofit® 3 is a fitness band with a small display screen.

Misfit offers cheaper wristbands in general, with models starting at $100. The Misfit Ray costs $100 and is a waterproof wristband that provides basic fitness tracking features.[12] The Misfit Ray is compatible with the Misfit smartphone app.

Apple's Watch offers fitness tracking apps, but this comes at the price of the Apple Watch itself. The price of an Apple Watch starts at $370. The Apple Watch Series 2 comes with built in GPS and a display.

---

[10] "Fitbit Flex 2™ Fitness Wristband." https://www.fitbit.com/flex2. Accessed 12 Dec. 2016.
[11] "vivofit 3 | Garmin | Fitness Tracker." https://buy.garmin.com/en-US/US/into-sports/health-fitness/vivofit-3/prod539963.html. Accessed 12 Dec. 2016.
[12] "Misfit Ray Premium Fitness + Sleep Tracker - Misfit." https://misfit.com/products/misfit-ray/. Accessed 12 Dec. 2016.

It is also waterproof.[13] The user's smartphone is not needed for fitness tracking. Third party apps may be used with the Apple Watch.

Most fitness trackers can have step counting error of ten percent or greater, but the Smart Sole reduces the error to within roughly five percent through the use of piezoelectrics. None of the fitness tracking companies offer reliable fitness trackers for less than $100. The Smart Sole costs well under $100 (see Section 7.3) while offering similar tracking capabilities and more accuracy. Table 12 provides a summary of Smart Sole and its competitors.

Table 12: Summary of Smart Sole and Competitors

Product	Features	Price
Smart Sole	Basic step counting, waterproof, smartphone app	$45
Fitbit Flex 2	Basic fitness tracking, waterproof, smartphone app	$100
Garmin vivofit® 3	Basic fitness tracking, waterproof, display, smartphone app	$100
Misfit Ray	Basic fitness tracking, waterproof, smartphone app	$100
Apple Watch Series 2	Advanced fitness tracking, waterproof, built in GPS, display, smartphone app	$370

7.1.2 Market Survey

The market for Smart Sole is large and growing. The fitness tracking industry is expected to grow from $2 billion, in 2014, to $5.4 billion by 2019.[14] Fitness trackers are becoming popular among all age groups, especially the Millennial generation. Prospective customers include those both with and without smart watches. Many customers with smart watches do not have a need for a stand alone fitness tracker, as the watch provides tracking abilities. However, some customers still desire a stand alone tracker that is more rugged or suitable in conditions that could ruin a smart watch. The other set of customers, those currently without a smartwatch, desire a cheaper product.

[13] "Apple Watch Series 2 - Apple." http://www.apple.com/apple-watch-series-2/. Accessed 12 Dec. 2016.
[14] "Fitness tracker market to top $5bn by 2019 - Wareable." 26 Mar. 2015, http://www.wareable.com/fitness-trackers/fitness-tracker-market-to-top-dollar-5-billion-by-2019-995. Accessed 11 Dec. 2016.

7.2 Cost Estimate

Bringing the Smart Sole product to the market would entail some additional costs and require further design to improve comfort and aesthetics. The costs of Smart Sole can be broken up into development costs and production costs.

7.2.1 Development

Development costs consist of preliminary research, design, and testing. In addition, if the product was taken to the market, other costs would be incurred such as the use of Calvin's facilities, which previously have not been taken into consideration.

Preliminary research costs include all of the time taken among the four team members to research project feasibility, which includes researching piezoelectrics, Smart Bluetooth, potential components and devices which would be used, and other early project-related research. This includes the selection of piezoelectrics. Preliminary research does not include design costs, which in many cases might be considered a type of research. Preliminary research only includes the time spent determining project feasibility and all aspects related to that. Because it is hard to break up time researching app design, the development of the app is included in the design costs. Between the four team members, roughly 30 hours of preliminary research was done. The team values their time at a rate similar to that of an entry level engineer. The compensation cost for an entry level engineer is based upon the cost to the company, not simply the salary of the engineer. Typically, the actual cost of an engineer can be estimated to be around double or slightly more than double that of their salary. This is because besides the engineer's salary, other benefits such as health and life insurance, retirement, holiday pay, education, traveling, and many more factors must be accounted for. In addition, there exist other costs in an organization, such as non-revenue generating departments like management, finance, customer service, quality, and human resources. Organizations must also pay for property, plant, equipment, and utilities. All of these variables factor into the cost per employee. With this in mind, a $70 thousand salary equates to about $35 per hour. A suitable cost to the company based upon this is $80 per hour. Therefore, a total of $2,400 was spent on research.

Design costs primarily consist of engineering design and construction time, which is also the bulk of the total time spent on the project. One other design cost is the cost of making the first prototype and any refinements made before moving the product towards production. The total time dedicated to designing the product is 450 hours. Project management and documentation time increases this to 500 hours. This equates to $40,000 in design costs. The cost of making one product has a minimal impact on this design cost. The prototype itself, including the battery and charger, costs $45.37. Data for prototype costs are

located in Appendix D. However, the actual cost of the project was significantly higher due to shipping costs, parts purchased that were purchased for safety, and other components that did not end up getting used. The total amount spent on the project was $272.55. The complete budget is provided in Appendix C. This brings the total design costs to $41,072.55.

Testing costs include the time taken to test the hardware, piezoelectric generation, app, and complete prototype. In addition, fixed testing costs include the equipment used to test the product. This includes a Fluke Multimeter, oscilloscope, function generator, power supply, and miscellaneous costs related to testing. These include probes, breadboards, and jumper wires. The costs for the following equipment are: $400 for a Fluke Multimeter, $520 for a basic Tektronix oscilloscope, $300 for a Keysight (Agilent) function generator, $200 for an Instek power supply, and $100 for probes, a breadboard, and jumper wires. The amount of time spent testing the product includes both testing of the final prototype and the numerous tests between each design iteration. Although testing time was integrated throughout the design process, it is separated into an individual cost category. Approximately 20 hours of testing occurred, resulting in $1600 of costs. Total testing costs add up to $3,120.

Additional costs include facility costs, which includes the previously unaccounted costs of using Calvin's facilities for development and business functions. Kentwood has one of the cheapest office space rental markets in the United States. A small office in Kentwood located near 28th Street and East Beltline can be rented for $9.00 per square foot per year. This includes utilities. Since the team will be outsourcing production, the size of the office space does not need to be large. The team plans to use the same facility in the future for production related overhead, providing space primarily for administration and future product research. A facility of 1500 square feet would accommodate the team's needs. This results in an annual cost of $13,500, or $1,125 monthly. With 8 months of rental for development and testing, the cost is $9,000.

Total development costs come to $54,792.55, which is rounded up to $55,000 to account for unforeseen issues or costs.

7.2.2 Production

Production costs are the other types of costs associated with bringing the product to the market. The product in production would be slightly different than the prototype. It was determined that purchasing the components individually and building the breakout boards as a company was cheaper than purchasing a premade breakout board such as those provided by Adafruit. This will increase labor costs slightly since more time will be needed to produce the boards. However, the increase in the outsourced labor cost per

product would only be a fraction of the cost savings of component costs for each product. Production costs can be broken down into both fixed and variable costs.

7.2.2.1 Fixed Costs

Fixed costs include property and equipment, general and administrative expenses, and marketing costs.

Property costs include the cost of renting facilities for business administration and troubleshooting. The same facility used in development will be used afterwards and rented for $13,500 annually.

Equipment used in development, such as the oscilloscope, function generator, multimeter, and power supply can be used during production, if troubleshooting is necessary. Therefore, the only equipment costs include annual replacements of damaged components.

General and administrative expenses are expected to be $1600 a week, or $83,200 annually.

Marketing costs are expected to be online marketing at a cost of $40,000 a year.

The total fixed costs are therefore $136,700 annually.

7.2.2.1 Variable Costs

Variable costs include component costs, labor costs, and shipping costs. The components and their estimated prices are included in Table 13. Purchasing components in bulk, in contrast to just one for the prototype, will lead to substantial savings in variable costs. Most retailers provide costs for a single unit and bulk purchases. Some bulk purchases only go up to 100, and it's possible that purchasing in even larger amounts would result in even lower unit costs, although at diminishing returns. The price of purchasing in bulk is used in the analysis. A three-axis accelerometer was chosen from Mouser. Although a single unit costs $1.21 alone, the unit cost is $0.48 when over 5000 are purchased[15]. The Bluetooth LE chosen costs only $2.18 when bought in orders of 2000 or more[16].

[15] "KXTJ3-1057 Kionix | Mouser." http://www.mouser.com/ProductDetail/Kionix/KXTJ3-1057/?qs=hicmRmzGcNR8WipH%252bLEjqg%3D%3D. Accessed 10 May. 2017.
[16] "nRF51822-QFAA-R7 Nordic Semiconductor | Mouser." http://www.mouser.com/ProductDetail/Nordic-Semiconductor/nRF51822-QFAA-R7/?qs=7xCC5jQa1OLeUlPjz6tWmQ%3D%3D. Accessed 10 May. 2017.

Table 13: Component Costs

Component	Price	Number	Cost
Accelerometer	$0.47	1	$0.48
Piezoelectric	$0.50	1	$0.50
Microcontroller	$2.96	1	$2.96
Bluetooth LE	$2.18	1	$2.18
Comparator	$0.45	1	$0.45
Small Components	$0.08	40	$3.20
Battery	$0.75	1	$0.75
Charger	$2.02	1	$2.02
Packaging	$0.35	1	$0.35
Total			**$12.89**

Labor and assembly of Smart Sole will be outsourced due to the lack of resources and quantity to allow assembly to be performed on such a small scale basis. The total hourly cost of labor to the company is $100 per hour. Breakout boards can be assembled using robotic surface mount technology at a rate of 360 per hour. This equates to $0.28 of labor per board. A Smart Sole can be assembled and packaged in two minutes, meaning 30 Smart Soles can be manufactured per hour. This equates to a labor cost of $3.33 per unit. The total labor cost per Smart Sole is $3.61.

Shipping costs are expected to average $1.50 based upon a mix of bulk shipping to distributors but also direct shipping from the business. Adding all of these individual costs results in a total variable cost of $18.00 per Smart Sole unit.

7.3 Summary of Financials and Profitability

Based upon the total variable costs, it has been determined that the Smart Sole will sell for $45. This is a 150 percent markup. This is a very large markup, but it is necessary to recover high development and fixed costs. A high volume of sales must also still occur. It is essential to maintain a lower price point to beat the competition, and with the 150 percent market, the product still accomplishes this. The volume of the product sold is expected to increase for the first three years, due to increased product awareness, and then taper off due to improved alternative products and models. At this time it is expected a newer

revision will be released. The profitability looks at only the first generation Smart Sole product. The business becomes profitable after three years, and the total investment is recovered during the third year as well. A summary of the profitability of Smart Sole is provided in Table 14.

Development costs are included as fixed costs in the first year only.

Table 14: Profitability of Smart Sole for Five Years of Operation

Year	Volume	Revenue	Fixed Costs	Variable Costs	Profit
1	3,000	$135,000	$191,700	$54,000	($110,700)
2	5,000	$225,000	$136,700	$90,000	($1,700)
3	10,000	$450,000	$136,700	$180,000	$133,300
4	9,000	$405,000	$136,700	$162,000	$106,300
5	6,000	$270,000	$136,700	$108,000	$25,300

8 Conclusion

The Smart Sole team set out to create a fitness tracker that functioned on renewable energy and would be able to compete with similar fitness trackers on the market. After researching and testing, the team determined that it was unfeasible to power the device solely on energy generated through the use of piezoelectrics. The team created hardware and software modules separately, making sure that effective testing was implemented before joining them into one unified product. A prototype for the team as well as an iOS application was developed, tested and improved based on test data. The prototype the team developed met the error standards they set out to achieve. The insole tracked at 95% accuracy on average, which improved on the market average of 10%. By performing more testing, responsive development, and alterations, and utilizing the given business plan, the Smart Sole team believes they can create a product that could be brought to market and succeed. This success could be heightened by adding more functionality to the insole and application, such as GPS, weight goal tracking, gait analysis, and fitness features. The Smart Sole team has learned a great deal from this project and hopes to utilize this knowledge in future work in their careers.

9 References

1 "Calvin College | Grand Rapids, Michigan." 2015. 12 Nov. 2016 <https://calvin.edu/>

2 "Returns and Warranty - Fitbit." 12 Feb. 2016, https://www.fitbit.com/legal/returns-and-warranty. Accessed 10 Dec. 2016.

3 Case MA, Burwick HA, Volpp KG, Patel MS. Accuracy of Smartphone Applications and Wearable Devices for Tracking Physical Activity Data. *JAMA.* 2015;313(6):625-626. doi:10.1001/jama.2014.17841

4 "Step Detection and Activity Recognition Accuracy of Seven Physical ...." 19 Mar. 2015, http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118723. Accessed 11 Dec. 2016.

5 "How accurate is the Apple Watch's step counter and distance tracking ...." https://www.cnet.com/news/smartwatch-step-counter-and-distance-tracker-accuracy/. Accessed 11 Dec. 2016.

6 Zhao, Jingjing, and Zheng You. "A Shoe-Embedded Piezoelectric Energy Harvester for Wearable Sensors." *Sensors* 14.7 (2014): 12497-510. *ProQuest.* Web. 13 Nov. 2016.

7 "Mobile OS market share 2016 | Statista." 2016. 13 Nov. 2016 <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>

8 "Building Mobile Applications with Oracle ADF ... - Oracle Help Center." http://docs.oracle.com/cd/E18941_01/tutorials/buildmobileappscontent/adfmobiletutorial_1.html. Accessed 10 Dec. 2016.

9 "Apple Developer." https://developer.apple.com/. Accessed 9 May. 2017.

10 "Fitbit Flex 2™ Fitness Wristband." https://www.fitbit.com/flex2. Accessed 12 Dec. 2016.

11 "vivofit 3 | Garmin | Fitness Tracker." https://buy.garmin.com/en-US/US/into-sports/health-fitness/vivofit-3/prod539963.html. Accessed 12 Dec. 2016.

12 "Misfit Ray Premium Fitness + Sleep Tracker - Misfit." https://misfit.com/products/misfit-ray/. Accessed 12 Dec. 2016.

13 "Apple Watch Series 2 - Apple." http://www.apple.com/apple-watch-series-2/. Accessed 12 Dec. 2016.

14 "Fitness tracker market to top $5bn by 2019 - Wareable." 26 Mar. 2015, http://www.wareable.com/fitness-trackers/fitness-tracker-market-to-top-dollar-5-billion-by-2019-995. Accessed 11 Dec. 2016.

15 "KXTJ3-1057 Kionix | Mouser." http://www.mouser.com/ProductDetail/Kionix/KXTJ3-1057/?qs=hicmRmzGcNR8WipH%252bLEjqg%3D%3D. Accessed 10 May. 2017.

16 "nRF51822-QFAA-R7 Nordic Semiconductor | Mouser." http://www.mouser.com/ProductDetail/Nordic-Semiconductor/nRF51822-QFAA-R7/?qs=7xCC5jQa1OLeUlPjz6tWmQ%3D%3D. Accessed 10 May. 2017.

10 Acknowledgements

Team Smart Sole acknowledges the support of several people and entities that made this project possible. This includes Mark Michmerhuizen, the professor and advisor for the team, the industrial consultant Eric Walstra of Gentex, the Calvin College Engineering Department, and anyone else who helped the team throughout the year.

Appendix A: Work Breakdown Structure

My Tasks in Calvin College Senior Design Team 11

- ~~1.2 Software~~ — due May 5
  Generate the software associated with our project including a mobile application and a team website.
  - ~~1.2.1 Mobile Application~~ — due April 28
    This task involves the generation of the mobile application for our product including data acquisition and storage as well as possible GPS applications.
    - ~~1.2.1.1 Research~~ — due November 25
      This task includes research how to design and create an effective mobile application.
    - ~~1.2.1.2 Requirement Definition~~ — due December 9
      This task includes defining functions and user interface that is relevant for our mobile application. This includes functional definitions as well as providing an easy to use interface. Dependent on 1.2.1.1 Research.
    - ~~1.2.1.3 Application Development~~ — due April 28
      This task involves the actual creation and design of the mobile application. This includes coding and testing. This task is dependent on both 1.2.1.1 Research and 1.2.1.2 Requirement Definition.
  - ~~1.2.2 Website Design~~ — due May 5
    This task involves designing and maintaining a team website that meets the requirements laid out by the engineering department.
    - ~~1.2.2.1 Research~~ — due May 5
      This task involves performing any research necessary to design a professional website. This task is continuously ongoing as necessary and can be performed simultaneously with the other tasks in this branch.
    - ~~1.2.2.2 Design Implementation~~ — due December 9
      This task involves creating the interface as well as the content associated with the website.
      - ~~1.2.2.2.1 Content~~ — due December 9
        The purpose of this task is to implement all of the required content onto our website. Can be performed simultaneously with 1.2.2.2.2 Interface.
      - ~~1.2.2.2.2 Interface~~ — due December 9
        The purpose of this task is to make our website attractive and easy to use. Can be performed simultaneously with 1.2.2.2.1 Content.
    - ~~1.2.2.3 Website Upkeep~~ — due May 5
      This goal of this task is to maintain the website and update the content and interface as necessary as the project continues. Can be performed at the same time as other website tasks.
  - ~~1.2.3 Software Budget Analysis~~ — due May 5
    This task includes budgeting the design and development of our prototypes as well as budgeting and analyzing long term options covered in our business plan. This data will be used along with 1.1.3 Hardware Budget Analysis to help generate a business plan and a overall project budget for the year.
- ~~1.1 Hardware~~ — due May 5
  Design and implement the hardware associated with our project
  - ~~1.1.1 Research~~ — due December 9
    This task includes all research and research related work that need to be done to make informed decisions when prototyping and testing. Research can be broken down into mechanical and electrical considerations.
    - ~~1.1.1.1 Electrical Research~~ — due December 9
      This sub-task includes all research regarding electronics and their implementation in our project.
      - ~~1.1.1.1.1 Component Analysis~~ — due December 9
        This task includes researching and choosing component for our design. May include some library research as well as preliminary ordering and testing of parts. Dependent on 1.1.1.1.2 Power Generation, 1.1.1.1.3 Step Counting, and 1.1.1.1.4 Wireless Communication as well some dependencies on 1.1.1.2 Mechanical Analysis.
      - ~~1.1.1.1.2 Power Generation~~ — due November 25
        This task includes researching power generation capabilities as well as the materials and products that would be most effective.

- [x] ~~1.1.1.1.3 Step Counting~~ due November 25
  This task includes research into step counting circuits as well as low power data storage options.
- [x] ~~1.1.1.1.4 Wireless Communication~~ due November 25
  This task includes researching low power wireless communication methods.

- [x] ~~1.1.1.2 Mechanical Research~~ due December 9
  This task includes researching all mechanical topics that must be taken into consideration for our design.
- [x] ~~1.1.1.2.1 Stress/Impact Analysis~~ due November 25
  The goal of this task is to perform research regarding stress/impact and the effects that it will have long term on our product.
- [x] ~~1.1.1.2.2 Ergonomic Considerations~~ due November 25
  This task includes taking into account user comfort as our product should feel like a standard running shoe.

- [x] ~~1.1.2 Product Generation~~ due May 5
  This task includes using research to generate possible prototype ideas as well as testing and implementing said ideas. Dependent on 1.1.1 Hardware Research.
- [x] ~~1.1.2.1 Product Testing~~ due April 21
  This task includes testing our prototype(s) for functionality and comfort. This task will be ongoing as long as necessary in multiple stages. Full test plan will be generated.
- [x] ~~1.1.2.1.1 Prototyping~~ due March 24
  This task involves implementing research to create and a design a prototype of our product. May later be divided into further sub-tasks once research is complete. Dependent on 1.1.3 Hardware Research.
- [x] ~~1.1.2.1.1.1 Comparator Design~~ due February 3
  Design the comparator component of the device hardware
- [x] ~~1.1.2.1.1.2 Accelerometer Design~~ due April 3
  Design the accelerometer hardware components of the step count module.
- [x] ~~1.1.2.1.1.3 GPS System Design~~ due April 3
  Design the GPS component of the step count module (if necessary)
- [x] ~~1.1.2.1.1.4 Bluetooth and Processor Design~~ due February 3
  Design the Bluetooth circuit and all circuitry associated. May undergo changes as the requirements are modified
- [x] ~~1.1.2.1.2 Test Plan Generation~~ due March 24
  This task involves the creation of a test plan to effective and efficiently test all aspects of our prototype design. Dependent on 1.1.3 Hardware Research.

- [x] ~~1.1.3 Hardware Budget Analysis~~ due May 5
  This task includes budgeting the design and development of our prototypes as well as budgeting and analyzing long term options covered in our business plan. This data will be used along with 1.2.3 Software Budget Analysis to help generate a business plan and a overall project budget for the year.

- [x] ~~1.4 Project Management~~ due May 5
  This task includes all functions associated with managing personnel, scheduling, and finances. This task will be ongoing throughout the entirety of the project.
- [x] ~~1.4.1 Team Organization~~ due May 5
  Includes general team and work organization tasks including the WBS, meeting minutes and team documentation.
- [x] ~~1.4.1.1 Work Breakdown Structure~~ due October 31
  Generate a basic WBS to divide tasks and assign due dates.
- [x] ~~1.4.1.2 Meeting Minutes~~ due May 5
  Keep a good record of team meetings and member contributions to increase accountability and meeting effectiveness.
- [x] ~~1.4.1.3 Documentation~~ due May 5
  Keep an organized and easy to access set of research notes and other documentation.

- [x] ~~1.4.2 Scheduling/Gantt Chart~~
  Create a basic schedule of tasks to better manage team time and organize due dates. Incorporated within WBS and subject to change.

- [x] ~~1.4.3 Financial Management~~ due May 5
  Manage costs of materials and other group functions. Dependent on 1.1.3 Hardware Budget Analysis and 1.2.3 Software Budget Analysis. This task is ongoing throughout the project.

- [x] ~~1.3 Business Plan~~ due April 7
  Manage all task associated with marketing and planning or project. Current due date is subject to change based on requirements

  - [x] ~~1.3.1 Executive Summary~~ due December 9
    This task includes developing the finalized business plan as well as recording and presenting any material that a executive would find important when considering our project. This includes the PPFS. Dependent on 1.3.2 Marketing and Sales and 1.3.3 Financial Projection.

    - [x] ~~1.3.1.1 Business Plan~~ due December 9
      This task includes any business related research and decisions that will go into the final PPFS. The primary focus of this task is on business related issues and not technical design ideas.

    - [x] ~~1.3.1.2 Design Plan~~ due December 9
      This task focuses on the design and engineering aspects that must be included in our executive summary. This information will be simplified and condensed for corporate purposes.

  - [x] ~~1.3.2 Marketing and Sales~~ due December 9
    This task includes taking information gathered through market analysis and applying it to develop a market and sales strategy. Dependent on 1.3.4 Market Analysis.

    - [x] ~~1.3.2.1 Marketing Strategy~~ due December 9
      The purpose of this task is to create a plan to promote our product based on data found through market research and analysis.

    - [x] ~~1.3.2.2 Sales Strategy~~ due December 9
      This task is to develop a plan to strategically distribute and sell our product in the most effective way possible.

  - [x] ~~1.3.3 Financial Projection~~ due April 7
    This task includes developing a big picture financial projection of our products including cost and revenue analysis against market trends to show profitability.

    - [x] ~~1.3.3.1 Financial Data~~ due April 7
      This task involves acquiring data regarding market projections, general sales trends, and any other information essential to developing an accurate financial projection for our overall project.

  - [x] ~~1.3.4 Market Analysis~~ due November 25
    This task involves performing research regarding the market for our product which will be used to help develop a potential business plan and product strategy.

    - [x] ~~1.3.4.1 Industry Outlook~~ due November 25
      The goal of this task is to generate an outlook on the revenue that could potentially be generated in the market we are entering. The primary purpose is to determine the future potential profitability of our product based on industry trends.

    - [x] ~~1.3.4.2 Target Market~~ due November 25
      The goal of this task is to find a target demographic/market that we would like to sell our product to. This will have a major impact on our marketing strategy.

    - [x] ~~1.3.4.3 Pricing~~ due November 25
      This task involves doing the research necessary to determine the selling price of our product. This will involve taking into account cost of materials as well as performing an economic analysis to determine a price to maximize profit. This task will be ongoing as materials used may change.

    - [x] ~~1.3.4.4 Competitive Analysis~~ due November 25
      This task involves analyzing and finding potential competition that our product may have and ensuring that our product will be able to compete in a competitive market.

Appendix B: Debugging Log

Date	Issue	Date Resolved	Reason for Issue	Resolution
4/10 2:30 PM	readSteps function was not working when the readValue command was implemented	4/12 3:00 PM	peripheral.readValue returns void.	Used .value on the CBCharacteristic object to return Data?
4/10 3:00 PM	App crashing when non numerical values are entered as a goal	4/11 5:00 PM	Floating point calculations can not be performed on a Float? containing nil	Made the default keyboard into a numpad
4/10 3:00 PM	setProgress Command is crashing the application	4/11 5:15 PM	The progressView bar was changed manually to a UIProgressView when it was still of type UIView.	Deleted and recreated outlet with type UIProgressView
4/11 5:25 PM	Program was not starting. Received "this class is not key value coding-compliant for the key" error	4/11 5:40 PM	A new outlet for the Battery percentage progress view was created. The old one was not deleted	Deleted the old outlet in the Storyboard connection inspector
4/11 5:00 PM	App is crashing when goal is added with nothing in text box	4/18 11:40 PM	The value of the goal label is being force unwrapped when it is equivalent to nil	A guard command was inserted to protect against empty strings ("") and zeros
4/18 2:00 PM	Numpad is stuck on the screen covering buttons and other functions	4/18 10:15 PM	The numpad is only set to dismiss if the AddGoal button is pressed	Used a Tap Gesture Recognizer to dismiss the text box if the user taps outside of the numpad
4/18 11:30 PM	User is not alerted if they enter an invalid value into the AddGoalTextBox	4/25 6:15 PM	UIAlertController has not been implemented	Added an UIAlertController object to notify the user of invalid entries
4/19 2:30 PM	Import button is crashing app when it is finding nil	4/24 3:00 PM	The Bluetooth Peripheral is not being connected leading to a peripheral object value of nil.	The Bluetooth was not configured properly on the hardware side.

4/24 2:30 PM	Cancel button in add exercise screen is not working	4/25 6:00 PM	The view was not being dismissed properly. The push dismissal method was always being called	Looked for any value in presentingViewController instead of a specific one (UINavigationController)
4/24 2:30 PM	Edit button is missing in the exercise tracker ver 4 24	4/24 2:40 PM	Accidentally deleted the code to implement the edit bar	Replaced the missing code
4/24 3:10 PM	Application only connects to peripheral some times, specifically when a breakpoint is set before the scan starts	4/26 12:45 PM	A race condition was (most likely) occuring between the views loading and the Bluetooth scanning process	Added a delay to the scanning command to wait until the initial viewcontroller loads
4/24 3:20 PM	didDiscover peripheral command is continuously calling	4/26 12:55 PM	The device appears to take multiple iterations of this function to connect	This is likely a non-issue. No resolutions have been implemented
4/25 5:15 PM	Peripheral is connected but services cannot be found even though did discover services is calling	4/28 10:30 AM	Services and characteristics were not being set properly from the hardware side	The services and characteristics were recreated using updated code
4/27 12:00 AM	Service is connected properly but the characteristic data is not being interpreted properly	4/28 12:00 PM	The step count value is stored as a "Data" type instead of a usable type	Implemented an extension on the Data class that allows a Data object to be converted to a UInt16 or any other basic data type
4/28 12:15 PM	Have to press Import Data twice to see imported value	N/A	Unknown. May need to call the read value function twice	Unresolved. Does not majorly affect performance
5/1 1:30 PM	Application crashes if import button is pressed with no peripheral connected	5/4 2:30 PM	The read functions are force unwrapping optionals that contain nil	Added a guard statement to verify a peripheral is connected before trying to read from it
5/1 2:15 PM	Sets Reps and Weight are not configured with proper identifiers in the Exercise view	5/1 9:30 PM	No labels were created to display identifying text	Added labels in a horizontal stack view to clearly establish the purpose of the text boxs
5/1 9:30 PM	User can type Strings into the Sets, Reps, and Weight boxes	5/4 2:30 PM	No restrictions are put on valid text box entries	This is a non-issue. Data is saved as a string so it should not cause any problems and it will allow users more freedom

5/1 9:30 PM	The percentage label for the battery percent is not properly placed when running some iPhone simulators	5/6 1:00 PM	Constraints were not configured properly	Reconfigured constraints
5/1 10:30 PM	The percentage bar is not updating when a new step count value is being imported	5/1 11:00 PM	The step progress value is never changed when importing the data	Added a line to the import button function to update the value of the step progress percentage
5/1 10:30 PM	Set, Reps, and Weight text fields are not closing keyboards when done is pressed	5/1 10:50 PM	The textFields delegates had not been set to the view controller	Set the text field delegates
5/2 6:30 PM	The imported step count data is not cleared when the user saves the data	5/4 2:45 PM	There was no function implemented to reset the step values	Created a function within viewDidAppear to determine whether the values should be reset and then reset them
5/3 9:00 PM	The stars label in the exercise tracker is stretching off screen on some devices	5/4 2:00 PM	This RatingControl object in the table view was extended off the screen in te storyboard view	Shrank the object size and adjusted the stars to be sized and distributed evenly
5/4 2:15 PM	Step counts get deleted when switching between tabs	5/4 2:45 PM	The function to reset the steps was called every time the view controller loaded	Added an additional variable to determine if steps had been saved. Implemented it in the viewDidAppear function
5/4 3:00 PM	Checkmark image is showing up for all data entries	5/4 3:00 PM	The table view cells were all set to the default image	Added code to the TableViewController to set desired image
5/5 1:30 PM	Step count display can only go up to 256	N/A	Only getting the first byte of the data when reading from the Bluetooth characteristic	Most likely a hardware side issue. Current code should be able to read multiple bytes

Appendix C: Budget

Date	Parts Ordered	Cost	Starting Budget	Remaining Budget
-	-	-	-	$500.00
11/17/16	2 Piezoelectric Generators	$27.00	$500.00	$473.00
2/17/17	15 Small Piezo Elements	$15.99	$473.00	$457.01
2/17/17	Mini Micro	$9.93	$457.01	$447.08
2/17/17	Adafruit Feather BLE	$23.05	$447.08	$424.03
2/17/17	Adafruit Accelerometer	$4.95	$424.03	$419.08
2/17/17	Adafruit Pro Trinket	$9.95	$419.08	$409.13
2/17/17	BLE 4.0	$19.95	$409.13	$389.18
2/17/17	BLE UART	$17.50	$389.18	$371.68
2/17/17	Accelerometers	$11.72	$371.68	$359.96
2/17/17	Shipping	$14.10	$359.96	$345.86
4/3/17	(2) Adafruit Accelerometer	$9.90	$345.86	$335.96
4/3/17	Adafruit Feather BLE	$29.95	$335.96	$306.01
4/3/17	Adafruit Shipping Est.	$7.14	$306.01	$298.87
4/3/17	(10) 3.3 V Zener Diode	$2.75	$298.87	$296.12
4/3/17	(2) Comparator	$2.96	$296.12	$293.16
4/3/17	(2) Comparator	$1.00	$293.16	$292.16
4/3/17	Mouser Shipping Est.	$4.99	$292.16	$287.17
4/7/17	Li-Ion Battery	$15.90	$287.17	$271.27
~~*4/7/17*~~	~~*3 V Li-ion coin battery*~~	-	*$271.27*	*$271.27*
4/7/17	3.6V Li-ion battery	$11.80	$271.27	$259.47
4/7/17	Charger	$9.95	$259.47	$249.52
4/7/17	IC chip	$3.00	$249.52	$246.52
4/7/17	Epoxy	$4.98	$246.52	$241.54
4/7/17	Combined Shipping	$14.09	$241.54	$227.45
			Total Spent:	$272.55

Appendix D: Prototype and Production Component Costs

Prototype Cost (No Charging Circuit)			
Component	Price	Number	Cost
Piezoelectrics	$1.07	1	$1.07
Microcontroller	$23.05	1	$23.05
Accelerometer	$4.95	1	$4.95
Comparator	$0.50	1	$0.50
Small components	$0.10	5	$0.50
Battery	$7.95	1	$7.95
Charger	$6.95	1	$6.95
Packaging	$0.40	1	$0.40
Total			**$45.37**

Prototype Cost (Charging Circuit)			
Component	Price	Number	Cost
Piezoelectrics	$1.07	11	$11.77
Microcontroller	$23.05	1	$23.05
Accelerometer	$4.95	1	$4.95
Comparator	$0.50	1	$0.50
Small components	$0.10	5	$0.50
Battery	$7.95	2	$15.90
Charger	$6.95	1	$6.95
Packaging	$0.40	1	$0.40
Total			**$64.02**

Production Cost (No Charging Circuit)			
Component	Price	Number	Cost
Piezoelectrics	$0.50	1	$0.50
Microcontroller	$2.96	1	$2.96

Accelerometer	$0.48	1	$0.48
Comparator	$0.45	1	$0.45
Small components	$0.08	40	$3.20
Battery	$0.75	1	$0.75
Charger	$2.02	1	$2.02
Packaging	$0.35	1	$0.35
Bluetooth	$2.18	1	$2.18
Total			**$12.89**

Production Cost (Charging Circuit)			
Component	Price	Number	Cost
Piezoelectrics	$0.50	10	$5.00
Microcontroller	$2.95	1	$2.95
Accelerometer	$0.48	1	$0.48
Comparator	$0.45	1	$0.45
Small components	$0.08	40	$3.20
Battery	$0.75	1	$0.75
Charger	$2.02	1	$2.02
Packaging	$0.35	1	$0.35
Bluetooth	$2.18	1	$2.18
Total			**$15.20**

Appendix E: Major Meeting Minutes

*All team members are present at the meeting, unless noted

Wednesday, September 14
- Continued brainstorming project ideas
- Researched feasibility of existing project ideas

Monday, September 19
- Listed project ideas in Google doc
- Compared ideas and researched requirements for project

Wednesday, September 21
- Agreed upon piezoelectric step tracker for project
- Defined project objectives and requirements

Wednesday, September 28
- Determined how project could be broken up
- Split project up with hardware, software, piezo, management
- Designated a leader to each portion of project
- Set some future goals

Tuesday, October 4
- Worked on PPFS outline

Monday, October 10
- Looked at power generation from piezoelectrics
- Look at energy harvesting circuits

Friday, October 14
- Worked on project brief for industrial consultant

Monday, October 17
- Created powerpoint for first presentation (Nick)
- Additional research on piezoelectrics for project presentation (Nick)

Wednesday, October 19
- Worked on project presentation (Nick and Modeste)
- Practiced presentations

Sunday, October 23
- Team meeting for website
- Team pictures

Wednesday, November 2
- Worked on project poster
- Discussed the upcoming Friday's at Calvin

Friday, November 4
- Talked at station for Friday's at Calvin

Sunday, November 6
- Began rough draft of PPFS based upon the outline

Monday, November 7
- Broke up sections to write of PPFS further

Wednesday, November 9

- Discussed PPFS progress
- Broke down additional sections among team members

Thursday, November 17

- Discussed devices to order (Nick and Andrew)
- Ordered piezoelectrics and smart bluetooth (Nick and Andrew)
- Worked on comparator circuit in lab (Nick and Andrew)

Wednesday, November 30 (2:30-3:30):

- Began creation of a power point for the second oral presentation (Garrett and Andrew)
- Updated action items list (Garrett)
- Planned lab work for the following day, including Bluetooth and Arduino implementation (Nick)
- Improve PPFS (Nick)
- Discussed potential system upgrades (accelerometer, GPS, micro USB chargeable)

Monday, December 5

- Weekly meeting
- Talked about using Bluetooth processor for implementation

Wednesday, December 8

- Meeting about step count module (Nick and Andrew)

Monday, December 12

- Brief meeting for finalizing thoughts and comments on PPFS final draft
- Discussed Bluetooth LE processor
- Discussed plans looking ahead over Christmas Break

Friday, February 3

- Discussed semester goals
- Reviewed budget
- Resumed researching specific components

Friday, February 10

- Met with IT for software considerations (Garrett)
- Worked on progress report (Nick)
- Researched microcontrollers
- Asked Professor Michmerhuizen about input on microprocessor/microcontroller, as well as development kits

Monday, February 13

- Brief meeting to set dates to finalize primary component selection, and order deadline
- Discussed methods to communicate individual research and progress

Wednesday, February 15

- Chose microcontroller, Bluetooth LE, battery

Friday, February 17

- Discussed project progress among each team member to be recorded in progress report
- Worked on block diagrams
- Hardware and software research and discussion

- Discussed adding power supply / energy capture circuit

Wednesday, February 22

- Received and reviewed first batch of parts
- Tested piezoelectrics
- Installed and tested connection with Arduino

Friday, February 24

- Discussed project goals and accomplishments
- Worked on supply and comparator circuit implementations
- Research and programming for Arduino Bluetooth

Sunday, February 25

- Worked on updating industrial consultant report

Monday, March 6

- Weekly meeting, discussed hardware and software progress

Monday, March 13

- Weekly meeting
- Discussed content for upcoming presentation on Friday

Thursday, March 16

- Finished slideshow and practiced verbal presentation 3 (Nick and Modeste)

Monday, March 27

- Discussed (minimal) progress made over spring break

Wednesday, March 29

- Finished team description for banquet

Weekend, April 7-9

- Divide and work on CEAC report

Monday, April 10

- Finished CEAC report
- Discussed and set progress deadlines for Easter break for each portion of project

Wednesday, April 19

- Discussed roles for CEAC presentation
- Major workday to have a working product

Friday, April 21

- Review and practiced CEAC presentation
- CEAC presentation and feedback

Monday, April 24

- Finished final report rough draft / outline
- General workday and meeting on week's objectives

Thursday, April 27

- Prepare presentation 4

Wednesday, May 3

- Group meeting over progress
- Talk about final report

Friday, May 5

- Go over Senior Design Presentation
- Plan Senior Design Open House