



WPI

MAGNEFORCE: VALIDATION OF A MODULAR TRI-AXIAL FORCE SENSOR FOR GAIT ANALYSIS

A Major Qualifying Project Report

Submitted By:

Brandon Lam (RBE/ME)

David Laovoravit (RBE)

Nathan Stomberg (BME)

Project Advisors:

Professor Çağdaş Önal (Lead Advisor)

Professor Karen Troy (Co-Advisor)

April 27, 2017

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on its web site without editorial or peer review.

Acknowledgements

We would like to thank Professor Çağdaş Önal and Professor Karen Troy for overseeing and guiding us through this project. We would also like to thank Professor James Duckworth and Joe St. Germain for their expertise and help with the circuitry of the project. We would also like to thank Katherine Creighton and Lisa Wall for their assistance with lab materials and helping with the procure supplies for this project. Thanks to Selim Ozel, for helping with last year's code, which already had a look-up table and data request. We would like to extend a special thanks to Anand Ramakrishnan and Anany Dwivedi for their assistance in creating and calibrating the sensors used.

Table of Contents

Acknowledgements.....	i
Table of Contents.....	ii
Table of Figures.....	v
List of Tables.....	vi
Glossary.....	vii
Abstract.....	ix
Chapter 1: Introduction.....	1
Chapter 2: Background.....	4
2.1 Diabetes and the Importance of Shear Forces.....	4
2.2 The Gait Cycle and Plantar Forces.....	5
2.3 Current Devices.....	6
2.3.1 Accelerometer Based Systems.....	7
2.3.2 Force Based Systems.....	8
2.3.3 Switch Based Systems.....	9
2.3.4 Video Based Systems.....	10
2.3.5 Multiple-Approach Systems.....	10
2.4 Gap in Technology.....	11
2.5 Measuring Shear Forces.....	11
2.6 Previous Project Iterations.....	13
Chapter 3: Project Strategy.....	14
3.1 Client Statement and Goal Statement.....	14
3.2 Objectives and Constraints.....	14
3.3 Project Approach.....	17
Chapter 4: Design Process.....	18
4.1 Needs Analysis.....	18
4.1.1 Detecting Normal and Shear Forces.....	18
4.1.2 Creating a Modular System.....	19
4.1.3 Device Safety.....	19
4.2 Design Alternatives.....	19

4.2.1 3-by-3 Module	20
4.2.2 Protrusions around the magnets.....	21
4.2.3 Multiple Sensors per Magnet.....	22
4.3 COMSOL Simulation.....	22
Chapter 5: Final Design	24
5.1 Electrical Component.....	24
5.2 Mechanical Component.....	27
5.3 Software Component.....	28
Chapter 6: Methodology	31
6.1 Load Cell Description	31
6.2 COMSOL Simulation.....	31
6.3 Force Testing.....	33
6.3.1 Force Calculations	33
6.3.2 Normal Force Testing.....	33
6.3.3 Shear Force Testing	34
6.4 Testing for Modularity	35
6.5 Data Analysis	36
6.5.1 Normal Force Data	36
Chapter 7: Results	38
7.1 COMSOL Results	38
7.2 Normal Force Results.....	39
7.3 Shear Force Results.....	40
7.4 Modularity Results	42
Chapter 8: Discussion	43
8.1 COMSOL Model.....	43
8.2 Normal Force Data	43
8.3 Shear Force Data	44
8.4 Discrepancies in Data.....	45
8.5 MagneForce as a Proof-of-Concept	45
8.6 Cost Analysis.....	45
8.7 Evaluation of Project Objectives.....	47

8.7.1 Safety	47
8.7.2 Sensor Functionality	47
8.7.3 Affordability	47
8.7.4 Gait Monitoring	47
8.7.5 Non-invasiveness	48
8.7.6 Reliability	48
Chapter 9: Recommendations and Conclusion	49
9.1 Future Recommendations.....	49
9.2 Conclusion.....	51
Works Cited	53
Appendix A: Gantt Chart	56
Appendix B: Magnet Protrusion CAD Model and Drawings	59
Isometric View	59
Protrusion Drawing	60
Protrusion Explosion Drawing	61
Appendix B: 3-by-3 Position Drawing	62
Appendix C: Silicone Molding Instructions	63
Appendix D: Final Mold Design.....	66
Appendix E: COMSOL Iterations	73
Appendix F: Current COMSOL Model Parameters	74
Appendix G: Foot Area Approximation	76
Appendix H: Foot Pressure Calculations	77
Appendix I: Shear Force Apparatus.....	78
Appendix J: Arduino Code for Modularity Test.....	81
Appendix K: Code for JAVA GUI	85
Java Code GUI display (Back End)	85
Java Code GUI display (Front End).....	88
Appendix L: MATLAB Code.....	95
Appendix M: Offset Filter Code	97
Appendix N: 3rd Iteration COMSOL Data Table.....	101

Table of Figures

Figure 1: Location of Foot Ulcers [10]	2
Figure 2: System Architecture of the MagneForce Device	3
Figure 3: Normal Gait Cycle [17].....	5
Figure 4: Gait Cycle Reaction Forces	6
Figure 5: IDEEA Monitor and Accelerometer Sensors [23]	7
Figure 6: F-Scan Show Insert [26].....	8
Figure 7: GAITRite Force Sensing Carpet [31].....	9
Figure 8: 3-D Gait Camera System [33]	10
Figure 9: University of Auckland's Strain Gauge Specification [41]	12
Figure 10: Early 3-by-3 Sensor Unit Arrangement.....	20
Figure 11: Design of Protrusions Holding the Magnets	21
Figure 12: Sensor Arrangement CAD Model Representation	22
Figure 13: Final MagneForce Module and System Architecture.....	24
Figure 14: Schematic of Circuit Wiring (left) and Board Layout (right) in Eagle	26
Figure 15: Unpopulated (left) and Populated (right) PCB.....	27
Figure 16: 3D Printed Silicone Mold.....	28
Figure 17: Screenshot of Java GUI in Action.....	30
Figure 18: Instron 5544 Uniaxial Testing Device.....	31
Figure 19: Simplified COMSOL Model of the Sensor Module	32
Figure 20: Normal Force Testing (left) and FBD of Normal Force Testing (right)	34
Figure 21: FBDs of Shear Force Testing	35
Figure 22: Apparatus used for Shear Force Testing	35
Figure 23: 2 Modules Connected for Modularity Testing	36
Figure 24: COMSOL Model under a 60N Load.....	38
Figure 25: Simulated Magnetic Flux (Ecoflex 0030 Silicone) Rubber	38
Figure 26: Data from 1 Sensor during Normal Force Testing (Dragon Skin 30 Silicone Rubber)	39
Figure 27: Normal Force Testing Data for all 4 sensors on the module during 1 trial.....	40
Figure 28: Shear Force Graphs in the Negative x (left) and Negative y (right) Directions. Boxes Show when Force was Applied and Removed	41
Figure 29: Shear Force Data for the Negative (left) and Positive (right) x Direction	41
Figure 30: Data Log of 2 Modules Running in Series	42

Permission is pending for the use of sourced figures

List of Tables

Table 1: Objective Breakdown Chart	15
Table 2: Project Objective Pairwise Comparison	16
Table 3: Bill of Materials for 7 Professionally Made Modules	46
Table 4: Cost Comparison of MagneForce and its Competitors.....	46

Glossary

3D	Three Dimensions
ADC	Analog-to-Digital Converter
CAD	Computer Aided Design
Cm ²	Centimeter squared
CRC	Cyclic Redundancy Check
FBD	Free Body Diagram
FPGA	Field Programmable Gate Array
GRFs	Ground Reaction Forces
GUI	Graphic User Interface
Hz	Hertz
IDEEA	Intelligent Device for Energy Expenditure and Activity
IMU	Inertia Measurement Unit
kPa	Kilopascals (Pressure)
MISO	Master-In, Slave-Out
mm	Millimeter
MOSI	Master-Out, Slave-In
MQP	Major Qualifying Project
MUX	Multiplexer
N	Newton
nF	Nano-Farad
OMD	OptoForce 3D Force Sensor
Pa	Pascals (Pressure)
PAGAS	Portable and Accurate Gait Analysis System
PCB	Printed Circuit Board
SPI	Serial Peripheral Interface
SSH	Secure Shell
UCT	University of Cape Town
μT	microtesla
US	United States
USD	United States Dollars

V	Volts
WPI	Worcester Polytechnic Institute
XOR	Exclusive OR

Abstract

Monitoring loading in both normal and shear directions is essential for properly treating diabetic foot ulcers, which arise from abnormal loading on the bottoms of the feet. Current technologies do not offer an affordable device that measures both normal and shear forces. To address this need, a novel device was created, consisting of four magnets in silicone rubber directly above four Hall-effect sensors. Applied forces displace the magnets and change magnetic flux data read by the sensors, indicating the magnitude and direction of the force. The sensor module was tested under a normal force of 200 N, and a 29.7 N force applied at 45 degrees. The z-axis data increased with normal force applied, at a rate of $4 \mu\text{T}/\text{N}$. For shear, the magnetic flux data for each axis trended in the same direction as the force applied on that axis. Two modules in series achieved an update frequency of 70 Hz. Overall, this device provides a proof-of-concept for relating displacement in magnetic fields to applied force in three dimensions.

Chapter 1: Introduction

Diabetes is extremely prevalent in the United States, and is a disease of great concern. As of 2014, just over 9.3% of the American population had diabetes, amounting to 29.1 million people with this disease [1]. Diabetes is an illness that inhibits the body from using blood glucose effectively. There are two major types of diabetes. Type 1 Diabetes, an inherited disease, prevents the body from producing the insulin that is needed for cells to take up the blood glucose. Type 2 Diabetes is the more common form of the disease, and is not hereditary, but is instead acquired from lifestyle-related factors [2]. With Type 2 Diabetes, the insulin receptors on the cells wear out, and the body no longer takes up blood glucose effectively [2].

Over time, diabetes exposes the body to high levels of blood glucose, which can have detrimental effects on a person's health. High blood glucose causes problems in many places throughout the body, including the heart, brain, nerves, kidneys, and eyes [2]. One particularly serious effect of long term Type 2 Diabetes is diabetic neuropathy, which arises from damage to the nerves that causes patients to lose sensation in their extremities, especially their feet [3]. When experienced in the feet, the condition is known as diabetic plantar neuropathy, a harmful effect of diabetes that can lead to ulcerations, or open sores, on the foot. In fact, about 15% of people with diabetes end up developing foot ulcers [4] and as such the treatment of ulcers deserves attention. Indeed, in 25% of cases, diabetic foot ulcers ultimately end in amputation of the foot or bone removal [5]. A study by Katoulis et. al., shows that understanding the forces acting upon the foot is necessary for treatment of foot ulcers, as well as managing pain, preventing ulcers, and avoiding surgery [6].

In particular, shear forces on the bottom of the foot pose a significant risk of causing and exacerbating foot ulcers in diabetes patients. Unfortunately, a proportional amount of attention has not been given to plantar shear, with many earlier studies focusing instead on the ground reaction forces (GRFs) through the foot. Indeed, the majority of easily accessible devices used for the characterization of plantar forces only measure the normal forces through the foot, as found in a 2014 review of plantar shear in diabetes patients [7]. Despite the maximum pressures exerted by shear forces being lower in magnitude than GRFs [8], repeated exposure to shear forces in the same area over time place patients at a high risk of plantar ulceration [9]. In that regard, a method to observe both normal and shear forces acting on the foot is needed in order to

evaluate, treat, and prevent diabetic ulcers.

The medical field already has devices that can measure a patient's gait cycle, normal forces, and even shear forces, that act upon the patient's feet; however, these devices are either too invasive, too bulky for a patient to carry around, too expensive for a patient to buy, or cannot measure both normal and shear forces simultaneously. In order to monitor a patient with diabetic neuropathy, and to understand the full range of forces on the patient's foot, the device must be able to measure both normal and shear forces. There is, therefore, a need for a noninvasive device capable of both measuring shear force and normal force on the bottom of the foot.

As a result, we have developed a modular force sensing device that uses Hall Effect sensors and magnets to relate displacements in magnetic fields to forces applied in three dimensions. A small circuit board is populated with four Hall Effect sensors, with a magnet suspended above each sensor in silicone rubber, which encases the whole device. The sensors are able to determine the position of the magnet by measuring the magnetic field in three dimensions. This allows for the normal force (the z axis, perpendicular to the surface of the sensor) as well as the shear force (the x and y axes, on the same plane as the surface of the sensor) to be measured when the magnet is displaced by a force. The small size of the circuit board allows for the number of modules to be customized to the size of the shoe, covering the areas of the foot most prone to ulcers: The bottoms of the toes, the pads of the feet, and the heels, as seen in Figure 1 [10].



Figure 1: Location of Foot Ulcers [10]

The sensors on the module record magnetic flux data, which changes as applied forces displace the magnets. The data is then processed through an Arduino, after which it is sent to a

computer via the computer's serial port. From there, the data can be post-processed using Microsoft Excel or MATLAB as seen in Figure 2.

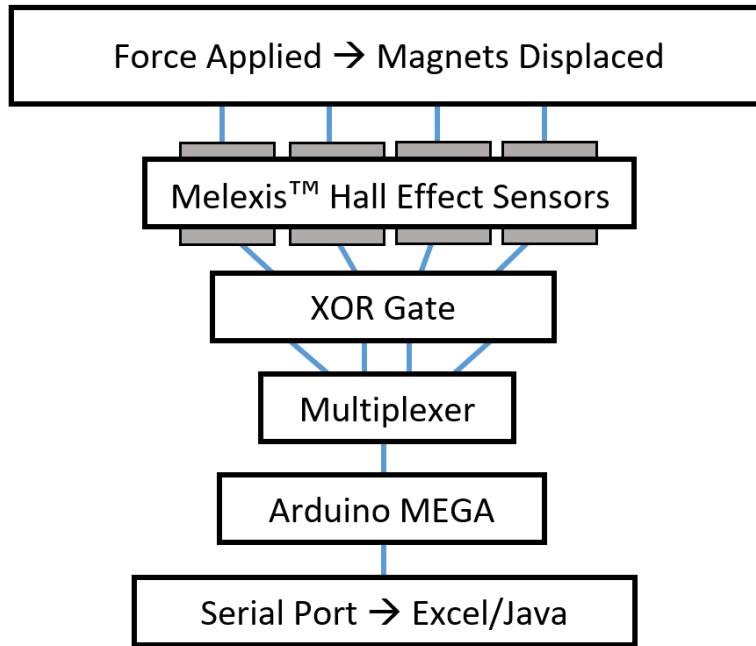


Figure 2: System Architecture of the MagneForce Device

Overall, our modular force sensing device (referred to as MagneForce) serves as a proof-of-concept for relating displacements in magnetic field to applied force in three dimensions. This concept can be used in a shoe pad to measure both normal and shear forces on a patient's foot, the data from which can be given to a doctor to determine if the patient is at risk for a dangerous diabetic foot ulcer, or to determine the best treatment option for managing or preventing diabetic foot ulcers.

Chapter 2: Background

2.1 Diabetes and the Importance of Shear Forces

Diabetes is a serious disease, and exists in high prevalence in the United States, [1]. With diabetes, the body is unable to uptake blood glucose, which is normally used as a source of fuel for cells. Left untreated, high levels of blood glucose can lead to a wide range of detrimental health effects over time. Present in nearly 50% of people with diabetes, nerve damage stands out as one of the most noticeable effects of high blood glucose. This nerve damage is characterized as diabetic neuropathy [11].

Diabetic neuropathy manifests itself through a variety of symptoms, all of which stem from nerve damage. Of the most recognizable of these symptoms is numbness in the hands and feet. The numbness is often accompanied by sharp pains, burning sensations, and bouts of high sensitivity to touch. In addition, nerve damage in the feet leads to ulcerations, sores, and infections [12], making plantar (foot) neuropathy particularly serious.

With regard to ulcerations, plantar shear plays a large role in their formation and aggravation in diabetes patients. For one, repeated shear forces actually leads to faster breakdown of the skin than normal forces [13]. Plantar shear also causes callouses and hyperkeratosis (abnormal skin thickening), both of which have been found to greatly increase the likelihood of diabetic ulcer formation [14]. In the context of diabetic neuropathy, understanding plantar shear takes on great importance.

Since diabetic plantar neuropathy causes mobility and coordination issues that lead to an atypical weight distribution, people with diabetic neuropathy are at a very high risk for developing foot ulcers, because of the increased exposure to plantar shear [9]. For diabetes patients, the importance of monitoring plantar forces is twofold. First, better understanding plantar pressure and shear force can help to create profiles that identify individuals that are at the greatest risk of foot ulcers [7]. Second, treatment of diabetic foot ulcers requires constant care and attention, in order to keep pressure off of the wound [15], and understanding the forces acting at the bottom of the foot helps with that treatment. As a result, the ability to monitor the pressures exerted on the foot is of great importance for the management and prevention of foot ulcers and wounds caused by diabetic neuropathy.

2.2 The Gait Cycle and Plantar Forces

Understanding the principles of human gait is essential to understanding the devices used to monitor gait, as well as how plantar forces relate to diabetic ulcers. A typical human walking cycle is defined by two major phases, following the motion of a foot from when it touches the ground for one step (heel strike) until it touches the ground again for its next step. The two phases are the stance phase (when the foot is on the ground) and the swing phase (when the foot moves through the air), both of which occur between heel strikes of the same foot [16]. Figure 3 shows a basic diagram of the normal gait cycle.

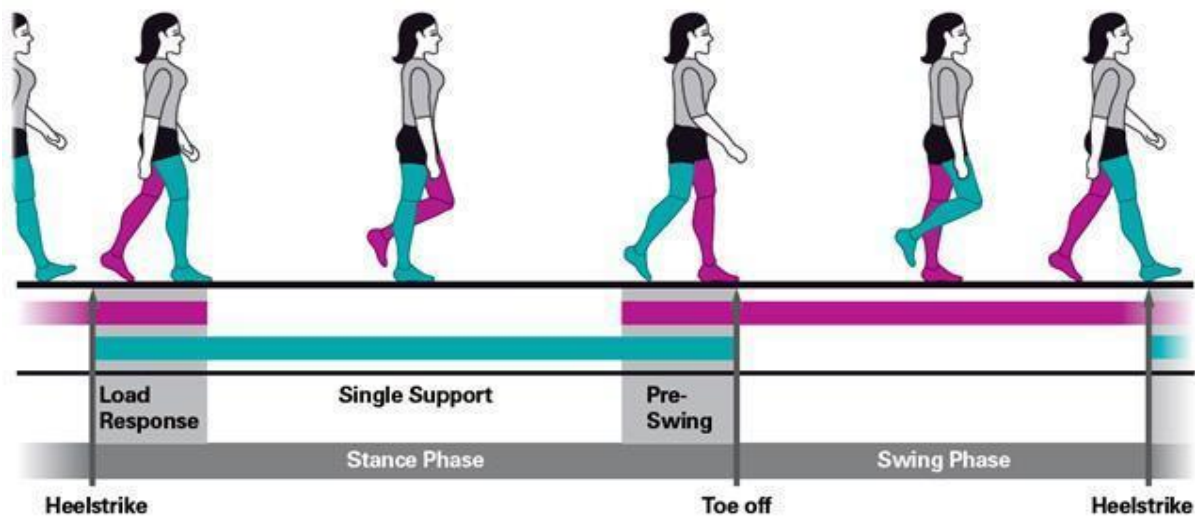


Figure 3: Normal Gait Cycle [17]

Of course, as a person walks, forces are experienced throughout the bottom of the foot as the pressure shifts with the person's position to support his body weight. While the foot of the average, healthy person usually experiences a center of pressure vector that distributes weight rather evenly [18, 19], this does not hold true if a person has foot problems, plays sports, or experiences any deviation from normal walking on a flat surface. In other words, gait varies widely with illness and injury.

Understanding the gait cycle is also important in terms of understanding the impact of shear forces on ulcer formation in diabetes patients. Even though the greatest maximum pressure on the foot is exerted by normal forces, the foot actually experiences plantar shear forces twice during the stance phase, exposing the foot surface much more frequently to shear forces than to

normal forces. This phenomenon is known as the biphasic nature of gait, which can be seen in Figure 4.

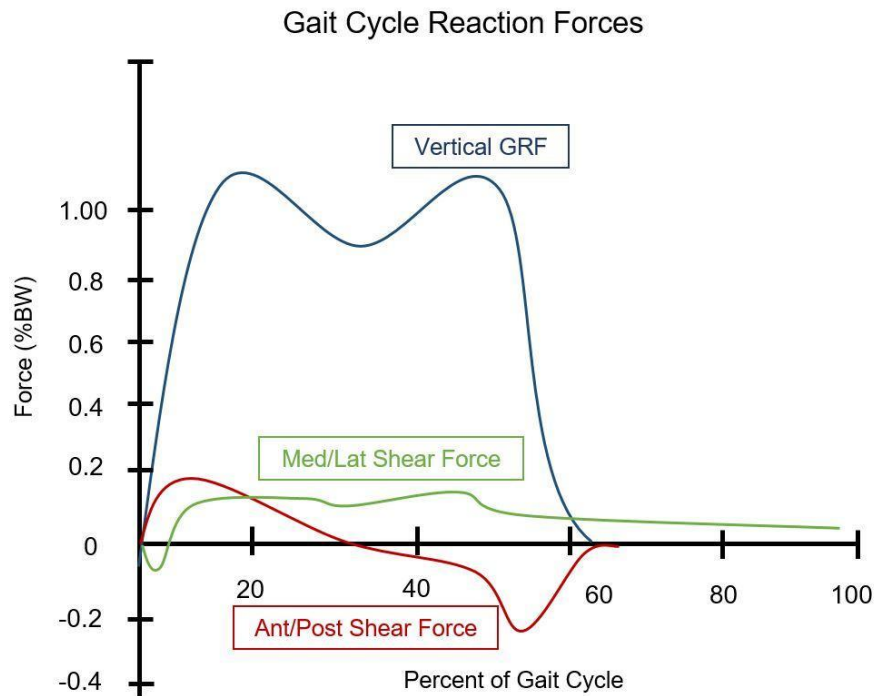


Figure 4: Gait Cycle Reaction Forces

In the case of patients with diabetic plantar neuropathy, abnormalities exist in the gait cycle that cause repeated shear forces of greater magnitude than seen above, stemming from the lack of sensation in the feet. The abnormal loading of the foot is what contributes to the increased risk of foot ulcers. A large number of devices already exist to study plantar forces for purposes of rehabilitation, clinical diagnosis, sports, and robotics [20]. As a result, the examination of current technologies will provide context and reveal the needs that we can fill with our device.

2.3 Current Devices

Gait analysis is an extremely important aspect of monitoring one's health. Many entrepreneurs and businesses have seen this need and have created devices in an attempt to fill that need. The devices range from technologies that fit inside the shoe to external set ups.

2.3.1 Accelerometer Based Systems

IDEAA

The IDEEA system is a Gait analysis system developed by MiniSun to help patients with cerebral palsy (a neuromuscular disease) as well as with sports rehabilitation [21, 22]. The device is comprised of 5 accelerometers that are taped on the user's thighs, ankles, and the chest. These sensors are pretty small and are not wireless as seen in Figure 5. The device measures the accelerations across the body and sends the data to a monitoring microcontroller that is attached to the hip of the user. That data from the monitor is then sent to the computer where it is processed in a program called ActView (GaitView) so that activity and gait can be analyzed. The system was shown to have a fairly high amount of reliability when being used for rehabilitation, [21], but unfortunately seemed to have less reliability with patients with cerebral palsy, having specific issues with walking velocity and stride length [22].



Figure 5: IDEEA Monitor and Accelerometer Sensors [23]

2.3.2 Force Based Systems

F-Scan

The F-Scan System, seen in Figure 6, is by far one of the best systems on the market. It uses 960 sensing elements in order to collect sensor data like pressure, force and the timing of a gait cycle [24]. The F-Scan is useful in many different applications and industries such as; analyzing foot function and gait abnormalities, monitoring foot disorders, evaluating athletic footwear, and identifying areas of potential ulceration [24]. There are 3 different methods that the F-Scan can record data in: tethered, wireless, and data-logger. These 3 modes each have their own scanning rate and distance associated with them: the tethered mode has a scan rate of 750 Hz within a 100 foot radius, the wireless mode has a scan rate of 100 Hz within a 328 foot radius, and the data-logger mode has a scan rate of 750 Hz, onto internal data, without a distance limit [24]. With such a large sensor array it is able to pick up forces at many of locations that many other systems will not be able to. Unfortunately the system lacks durability and repeatability and often has a higher with regards to calibration, creep and hysteresis [25].

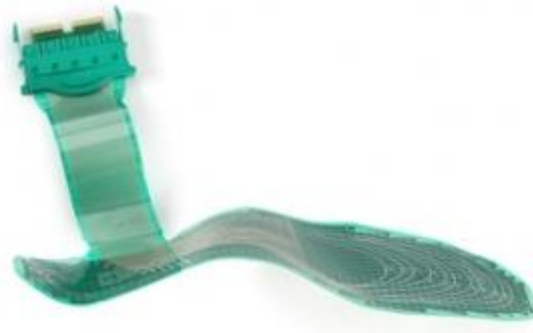


Figure 6: F-Scan Shoe Insert [26]

Pedar by Novel

The Pedar system, developed by Danish company Novel, is a product similar to the F-Scan device. The Pedar system is a shoe insert that uses 85 sensor elements to detect pressures of up to 1200 kPa. The Pedar system boasts both tethered and Bluetooth capabilities, as well as local flash memory for data logging [27]. While this device reliably measures plantar pressures, it does not measure shear forces, and costs as much as \$5000 [28].

Smart Shoes

The “Smart Shoes” system is one that measures force based on the pressure changes in

air bladders that have been built into the shoe. The pressure changes in the silicone air bladders are converted into measurable voltages. A strength of this system is that it has over a 97% repeatability for measuring normal forces. However, the “Smart Shoes” system does not allow for measurements of shear force and so would not be a suitable method for diagnosing diabetic neuropathy [29].

GAITRite

The GAITRite system is a force sensing carpet, shown in Figure 5, for modeling the distance of user’s stride as well as the timing between each heel strike and toe off. A drawback to GAITRite is that it cannot measure the user’s foot angles during the gait cycle [30]. The carpet itself is 60 cm wide by 360 cm long [31], which is far too large to be practical for universal, everyday use. Overall, the flaws in the GAITRite system make it a non-ideal system for helping with diabetic neuropathy.



Figure 7: GAITRite Force Sensing Carpet [31]

2.3.3 Switch Based Systems

PAGAS System

The PAGAS System relies on switches at the toe and heel of the foot in order to determine the gait of the user [32]. The heel strike and toe off timings are measured when only the respective switch is activated. When both are activated, the foot is in the stance phase, and when neither of the switches are activated, the leg is in swing phase. The device’s primary focus

is measuring the different phases of the gait cycle, because of this device cannot determine the forces being applied or the angles at which the heel and toe strike occur, making it insufficient for our application.

2.3.4 Video Based Systems

3-D Gait and Run³

Running Injury Clinic's "3-D Gait and Run³" is a video-based motion capture device used to track the gait cycle [33]. Using cameras, like the system in Figure 8, and specialized software, "3-D Gait and run" measures rotation and flex of different parts of the body, such as the ankle, knee, hip, and pelvis [33]. Because Gait and run only records information using video data, it cannot measure the forces exerted on the feet. Operating this technology properly would require a fair amount of expertise and cost, as well.



Figure 8: 3-D Gait Camera System [33]

2.3.5 Multiple-Approach Systems

Shoe-Integrated Wireless Sensor System

The Shoe-Integrated Wireless Sensor System includes 3 orthogonal accelerometers, 3 orthogonal gyroscopes, 4 forces sensors, 2 bi-directional bend sensors, dynamic pressure sensors, and an electric field height sensor [34], all of which work together to collect a comprehensive

amount of data. Using sensors inside and outside the shoe, the device provides a detailed picture of the user's gait cycle. The system is similar in concept to our shoe insole, incorporating many different sensors into a single package in the shoe to measure both the gait cycle and the forces applied. The system's drawbacks, however, stem from its use of force sensitive resistors (or piezoresistive force sensors) which are subject to creep, high hysteresis error, and low repeatability for force measurements. The piezoresistive force sensors are also unable to measure shear forces.

2.4 Gap in Technology

As discussed previously, much of the technology used in gait analysis focuses on measuring the stride and positions of the body as it goes through the gait cycle. While this is an important aspect of gait analysis, there are other aspects of the gait cycle that are often left out, such as the measuring of shear forces applied during the gait cycle. More often than not, devices only take into account the normal forces acting on the foot [35]. Yet, shear forces are important for detecting and preventing many foot maladies, such as ulcerations, arising during diabetic neuropathy [9]. Many of the devices that are capable of being used to measure and monitor diabetic neuropathy are either too expensive, too large, or too invasive for a patient to use in the comfort of his or her home easily, and none measure shear forces in a compact device.

2.5 Measuring Shear Forces

Although there are many gait monitoring devices, most of the devices fail to measure shear forces, are expensive, or do not measure shear forces in an array. Diabetic ulcers arise when there are high pressures inside the foot, in part due to shear forces. Patients with diabetic plantar neuropathy are unable to feel their feet and thus do not know how much pressure they could be applying to their feet, causing physical damage. In order to moderate and prevent the development of ulcers in people with this ailment, the ability to measure shear force within a person's foot is highly important [36].

Currently, there are a limited number of ways to accurately measure shear forces. However, the current solutions are either too expensive, or have problems with creep and deterioration over time, needing chronic recalibration. For example, the OptoForce 3D Force Sensor (called OMD) [37] is able to measure both force and rotation in all three dimensions, but

each sensor costs over 900 dollars [38].

Another implemented solution is the use of strain gauges [39]. At the University of Auckland, researchers developed a triaxial-measurement shear-test device for measuring soft biological tissues, see Figure 9. Although effective, their device is about 25mm in size and the strain gauges are attached to flexible steel beams, which could lose its accuracy over time due to constant bending. Strain gauges themselves also need to be recalibrated regularly as their stiffness increases over time [40]. For our project's purposes, it is not ideal to have regular recalibration, because it makes the device less user-friendly.

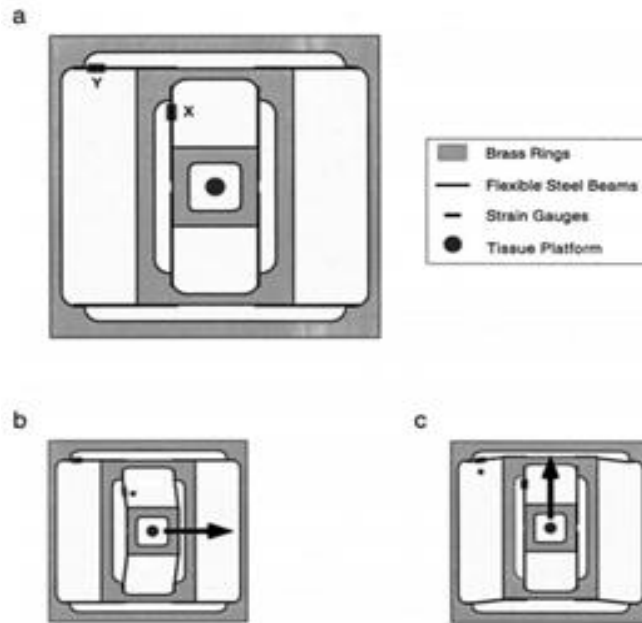


Figure 9: University of Auckland's Strain Gauge Specification [41]

Another solution that has been implemented is the use of strain gauge rosettes, investigated at the University of Cape Town (UCT) [42]. While strain gauge rosettes can measure shear force, they do not provide the resolution our project needs. As such, UCT uses the strain gauge rosettes in four main locations instead of having them in an array. As with the University of Auckland's device, the rosettes use strain gauges, which require regular recalibrations.

2.6 Previous Project Iterations

The previous iteration of this device was made by a Major Qualifying Project (MQP) team at Worcester Polytechnic Institute (WPI), which investigated creating a shoe pad that is capable of measuring normal and shear forces with Hall Effect sensors. Their main objective was to be able to “measure normal forces along the foot at as many points as possible” [43]. Their secondary objective was to measure the “shear forces at each of the points normal force is measured” as well as to track the foot motion in the air to determine the gait cycle.

This previous project did accomplish the primary task of detecting normal forces with the Hall Effect sensors. Using Dragon Skin 10 silicone rubber, the project team was able to measure the normal forces acting upon the sensors with a 5.85% hysteresis level [43]. However, their attempt of measuring shear forces did not produce any conclusive data, and the team was “unable to successfully relate pressure applied to the device to the magnetic fields” [43]. The team was able to achieve an update frequency of 33 Hz with a sensor density of 1 sensor per cm². In order for them to test for shear and normal forces, the team also designed a calibration system.

The main area for improvement from last year’s project is the measurement of shear forces. Last year’s team focused mainly on the measurement of normal forces and therefore did not measure shear forces accurately. Since our project focuses on helping people with diabetic plantar neuropathy, we will be focusing on acquiring both normal and shear force measurements. Overall, the research on existing devices and the lessons learned from the previous project both established the gap in technology and contributed to our design process.

Chapter 3: Project Strategy

3.1 Client Statement and Goal Statement

The scope of this project stems from the client statement, which was made by the project team after receiving the initial task from the advisors:

Develop and test a portable, non-invasive force-sensing array to be embedded in a shoe pad that can reliably measure normal and shear forces on the bottom of the foot, track the gait cycle, and transmit the information in a meaningful way for use in clinical and robotic applications.

Using this client statement as a starting point, the project group developed a focused goal statement to refine the scope of the project. The project goal is as follows:

Develop and test a portable, non-invasive gait-monitoring device that reliably measures normal and shear forces on the bottom of the foot, to gather information for use in treatment and prevention of foot ulcers resulting from diabetic neuropathy.

The group chose to focus the scope of the device on the monitoring of diabetic foot ulcers due to the serious nature of the disease. Additionally, there are a large number of applications for a portable gait monitoring device that effectively measures shear; choosing to focus on the application relating to diabetic neuropathy helped to narrow down the scope of both the research and the design process.

3.2 Objectives and Constraints

This project is building upon progress made by a previous MQP team with the same task. The previous group developed a pair of 3-by-3 sensor arrays in silicone rubber that reliably measured normal forces, but did not reliably measure shear forces. Keeping the client statement and the progress from the previous project in mind, our team developed the following objectives: The device must have improved sensor functionality over the previous project with regard to shear force measurements and sensor density; it must have gait monitoring capability; it must also be non-invasive, reliable, safe, and affordable. **Table 1** shows the objective breakdown.

Table 1: Objective Breakdown Chart

Objective Breakdown Chart					
Sensor Functionality	Gait Monitoring	Non-Invasive	Reliable	Safe	Affordable
Improved Sensor Density	IMU to detect heel and toe strikes, foot position	Thin	Low Hysteresis Error & High repeatability	Durable Materials	Cost no more than \$750 to build
Reliable Normal Force Measurements		Wireless	Data comparable to other devices	Low User Risk	
Reliable Shear Force Measurements		Portable	Low Material Creep		

The major area for improvement from the previous iteration of this project is the ability to measure shear forces. Given the necessity of reliably measuring shear forces on the sole of the foot to accurately treat diabetic plantar neuropathy, our group’s primary objective, after safety, is to ensure that the sensor array can accurately and reliably measure shear forces. A pairwise comparison was done to prioritize our objectives, as seen below in **Table 2**.

Table 2: Project Objective Pairwise Comparison

Project Objective Pairwise Comparison							
	Sensor Functionality	Gait Monitoring	Non-Invasive	Reliable	Safe	Affordable	Totals
Sensor Functionality		0.5	1	0.5	0	0.5	2.5
Gait Monitoring	0.5		0.5	0.5	0	0.5	2
Non-Invasive	0	0.5		0	0	0.5	1
Reliable	0.5	0.5	1		0	0.5	2.5
Safe	1	1	1	1		1	5
Affordable	0.5	0.5	0.5	0.5	0		2

The objectives in each of the rows above were evaluated against the objectives in each column. A score of “1” means the objective is more important than that against which it is compared, a score of “0.5” means the objectives are equally important, and a score of 0 means the objective is less important. Upon evaluation of the project objectives, safety is the most important, followed by sensor functionality and reliability of the device. The “non-invasive” objective ranked last because without the capacity to accurately and reliably measure the forces on the foot, the size of the device is almost irrelevant.

A number of constraints play a part in this project as well. For one, the size of the final product is limited by the size of the shoe that it is designed for. Additionally, the price for development of the project must fit within the team’s budget, which is \$750. Time is also a constraint, as the team must complete all the research and development into the time frame of one school year. Finally, the device must be safe to use above all else, ruling out solutions that may appear easier but involve greater user risk.

3.3 Project Approach

Our project is divided into three major phases: research, prototyping, and testing. The tasks planned throughout these phases were plotted in a Gantt chart (Appendix A). The first stage of our project was dedicated to researching and understanding the problem, as well as understanding the progress made by the previous MQP team. The second phase of our project involved assembling and troubleshooting the circuit board used for the sensor module. This phase involved assembling a 5V circuit for a 2-by-2 sensor array, and troubleshooting iterations with different circuit components (i.e. multiplexer and XOR gate). The building phase also included iterating through several versions of a mold used to encase the circuit board and magnets in silicone rubber. The final testing phase included designing tests to validate the sensor functionality. The tests were directed at showing the sensor module could serve as a proof-of-concept for detecting normal and shear forces. This phase also included a brief review of the design objectives and a cost analysis in preparation for the final presentation. The following section describes the product design process in detail.

Chapter 4: Design Process

Our goal was to create a portable, non-invasive force sensing device that reliably measures normal and shear forces on the bottom of the foot, to aid in the treatment and prevention of diabetic foot ulcers. We are aiming to address the needs brought to attention by the previous project, while adding innovations to increase reliability and accuracy. This chapter details the processes behind the design of the product, as well as the thoughts behind their alternate variations.

4.1 Needs Analysis

Keeping the project objectives in mind, the key features of the shoe pad device were considered, as a way of evaluating what type of work needs to be accomplished, and also as a way of driving the design of the product. The three major needs discussed here are measuring normal and shear forces, creating a modular system, and ensuring the device is safe.

4.1.1 Detecting Normal and Shear Forces

Working with the knowledge of the system created by the previous MQP group [43], it was clear that shear forces must be prioritized as the focus of our project in order to properly measure them. Normal force required attention as it also plays a part in correctly diagnosing foot ulcers. The device for this project will use Hall Effect sensors to measure the displacement of a magnetic field to detect normal and shear forces. While normal forces act perpendicular to the bottom of the foot, shear forces act in the same plane as the bottom of the foot. When occurring at the same time, the combination of forces are applied in the x-, y-, and z-planes. The application of these forces to a magnet will thus cause displacements in the magnetic field in three dimensions, which are picked up by the Hall Effect sensors.

Magnetic interference plays a large part in ensuring the sensors are able to send read the correct data values. If one of the magnets affect the sensors that are not directly below it, then it will interfere with the device's ability to read multiple points of data. In order to ensure this would not affect the device, the minimum distance of separation between sensors and magnets needed to be determined.

The rotation of the magnet also needed to be controlled. If the magnets rotate within the

silicone rubber, then the consistency of the data is diminished. To address this, anti-rotation measures were taken during the design of the device. This is especially important so that the magnet does not change angles whenever force is applied to the system.

4.1.2 Creating a Modular System

A modular design is desirable as it allows for greater flexibility in application. In order for this modular design to be achieved, the device must take up a small area and it must have a circuit that can support such a design.

A small module allows for many sensors to be placed in a single area. This would allow a custom design for the array of the sensors and maximize the sensor density without having to conform to only one size. This is extremely useful in the prototyping stage before a marketable device has been decided upon and allows for many different applications outside of just in a shoe pad.

An ideal circuit design allowed for all of the modules to be run in parallel as more sensors will be reading data at once, thus increasing the frequency rate that the device send data. Another ideal trait of this circuit is that it would not add many additional input/output lines to the total system. This is because it would allow for the maximum number of modules to be connected.

4.1.3 Device Safety

A safe device was designed so that it can support the weight and forces that would be applied on the device without injuring the user or affecting the device's ability to properly function. In order to do this, the circuit was encased in silicone rubber to protect both the user and the device. The best silicone would have a stiffness that allowed for the applied forces to create a displacement in the magnets without compromising the integrity of the circuitry.

4.2 Design Alternatives

Many designs and ideas were considered during the brainstorm of our project. While they may be an excellent ideas not all of them made it to the final design. In this section we discuss some of the alternative designs that had been discussed over the course of the project.

4.2.1 3-by-3 Module

Last year's MQP team had developed a large 3-by-3 array, seen in Figure 10 (see Appendix B for more details), which would be placed at 2 different locations of the shoe pad, the front and the back of the foot, allowing for a total of 18 data points at any given time. These two locations were likely prioritized to help detect the toe off as well as the heel strike phases of the gait cycle.

This sensor array comprised of both a rigid circuit board and a piece of silicone with magnets embedded inside. The circuit used Melexis position Hall Effect sensors (MLX90363EDC-ABB-000-SP) in order to detect the changes in magnetic field as the user applies forces to the silicone. The silicone would act as a cushion to help provide a more comfortable insole for the user as well as protect the circuit underneath.

The circuit board was created using the 3.3-volt design found in the Melexis sensor datasheet as a reference. For the 3-by-3 array the team wired all 9 sensors in parallel allowing all of the sensors to gain the required 3.3 volts to operate.

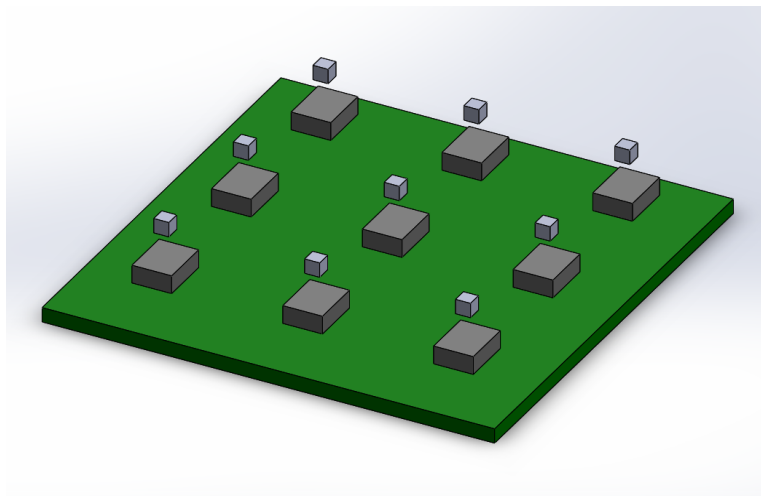


Figure 10: Early 3-by-3 Sensor Unit Arrangement

The decision to decrease the array from a 3-by-3 to a 2-by-2 module was mostly guided by the area the array would take up. Using last year's 3-by-3 array as a reference, we noticed that the size of the 3-by-3 circuit board did not maximize the amount of sensors that could potentially be placed in a shoe. By decreasing to a 2-by-2 module, the size of the PCB decreased dramatically, allowing for greater potential to at least place modules in the areas of most frequent

ulceration [10], see Figure 1 in Chapter 1.

4.2.2 Protrusions around the magnets

As previously mentioned the rotations of the magnets was going to be an issue that needed to be addressed. One possible solution to prevent magnet rotation is to create protrusions around the magnet, see Figure 11 for an image of the protrusions. These acrylic protrusions would be positioned along the centers of each face of the magnet, see Appendix B for more details. The acrylic protrusions could have holes or porosities, allowing the magnets to “grip” the silicone around it, and preventing the magnets from rotating within the silicone. With this design, the magnets’ positions are not restricted, because the movement along the magnets’ axes are not restrained. Since the material of these protrusions are made from acrylic, the structure would also not interfere with the magnetic fields being detected by the Hall Effect sensors

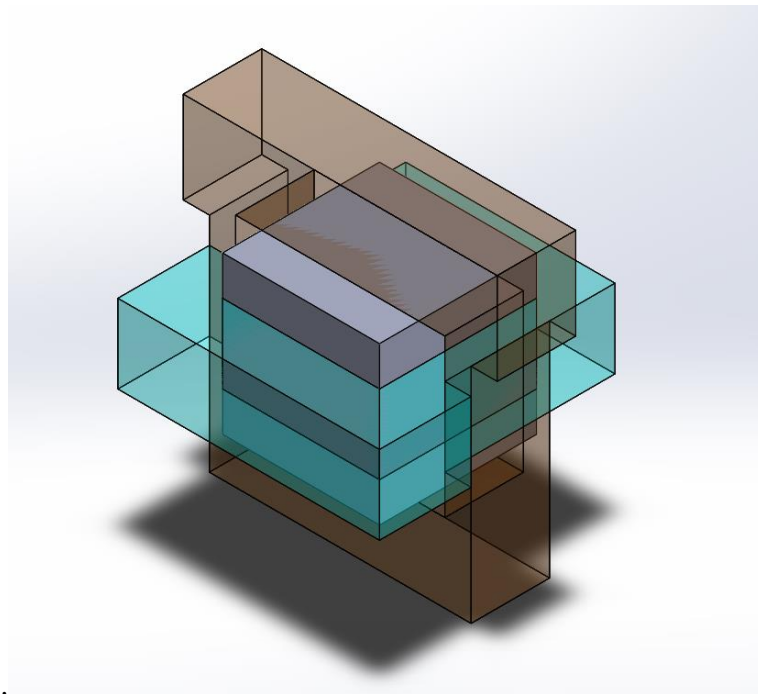


Figure 11: Design of Protrusions Holding the Magnets

While testing the previous MQP’s design we found that the acrylic plates they used to surround the magnet in one plane was enough to keep the magnet from rotating and misaligning. The design of using a single plate is much more simplified than the using the multiple protrusions and so was the design used in the final design of the device.

4.2.3 Multiple Sensors per Magnet

Another design that was explored was allocating more sensors to each magnet. This design was an attempt to detect shear forces more accurately. Magnetic fields are polar in that they have a North and South poles. If sensors are strategically placed around the magnet on the circuit board, then the change in x and y axes (parallel with the circuit board) can be detected more easily. As the magnets leaned to one direction at least one of the Hall Effect sensors would detect a larger value than the others. By comparing these values we may determine the direction of the shear force as well as the magnitude of the force.

Some of the orientations can be seen in Figure 12. 1-4 sensors per magnet designs were considered as well as their potential orientations. A greater sensor to magnet ratio seemed to be superfluous and unnecessary.

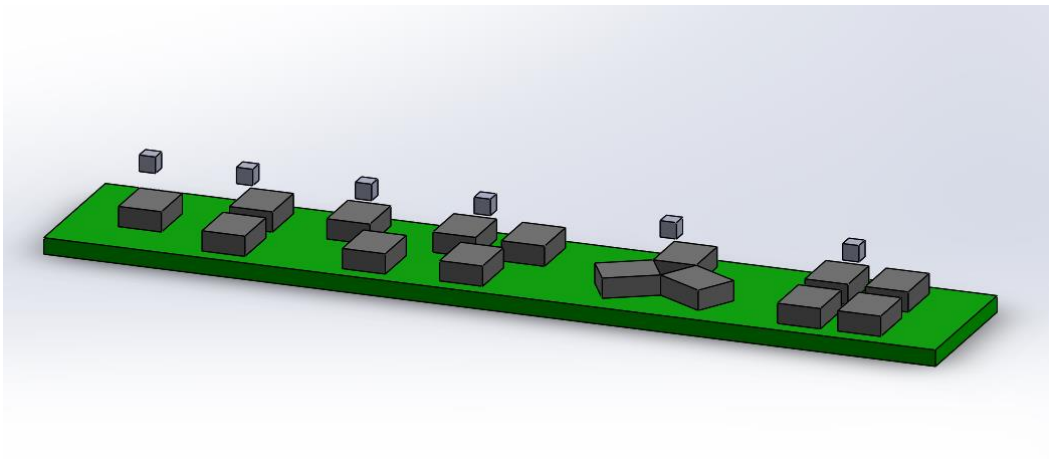


Figure 12: Sensor Arrangement CAD Model Representation

Through initial testing, it was discovered that one sensor was sufficient in detecting the changes in magnetic flux values as long as the north and south poles were perpendicular with the surface of the sensor. This allowed for the best case scenario of having a 1-to-1 sensor to magnet ratio, thus allowing for the greatest force measurement density possible.

4.3 COMSOL Simulation

We used COMSOL Multiphysics version 5.2 to validate our designs by simulating both the physical and magnetic properties of the sensor assembly. By applying forces in the model, the displacement of the magnets can be simulated, and the expected magnetic field

measurements at the sensor locations can be recorded. In this way, the “ideal” response can be compared to the actual response from raw testing data. A detailed description of the numerical simulation we used for testing can be found in Chapter 6.

Chapter 5: Final Design

Our final design is a modular 5V circuit design with a 2-by-2 array of Hall Effect sensors on top, making use of a multiplexer and XOR gates to connect all 4 sensors to a single output. The following sections detail the decisions made to go from the alternative designs to the final 2-by-2 modular design. The final module, along with the system architecture, are in Figure 13 below.

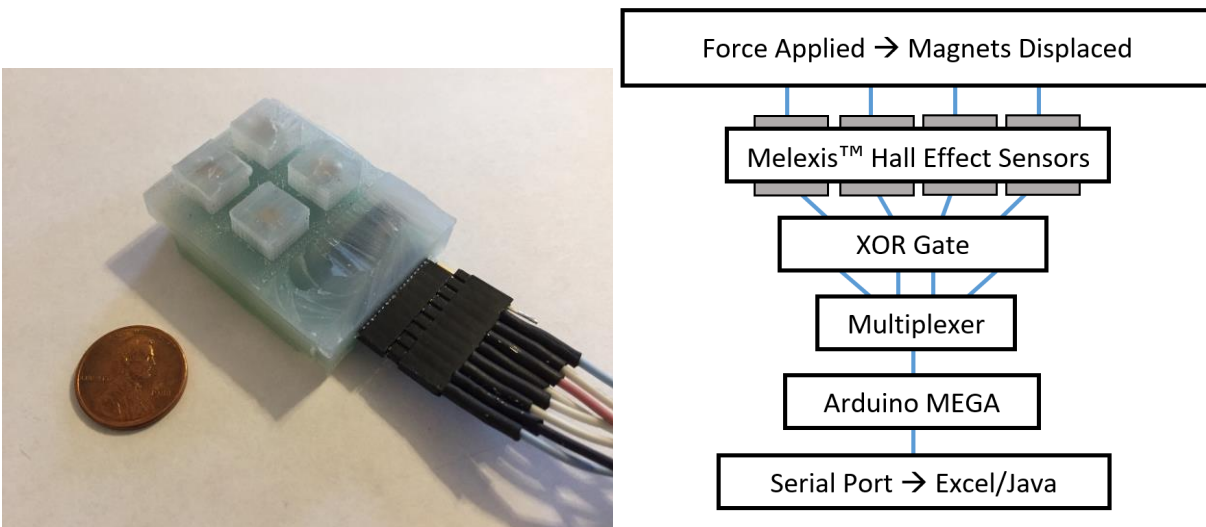


Figure 13: Final MagneForce Module and System Architecture

5.1 Electrical Component

Circuit Model

Found in the datasheet for the Melexis Hall Effect Sensors [44] are circuit designs for using either a 3.3 V or 5 V power input. The main difference between the two circuits, besides the input voltage, is the wiring of the capacitors. The 3.3 V circuit uses a single 100 nF capacitor to prevent power surges. The 5 V circuit recommends a 47 nF capacitor to prevent power surges as well as a 100 nF capacitor to act as a decoupling capacitor which filters the signals.

The 5 V circuit was chosen for the final design because of the advantage of decoupled signals and because 5 V is more commonly used than 3.3V for logical operations in microcontroller units.

Analog Multiplexer usage

An analog multiplexer (MUX) (ADG704) was added to the final design of circuit board.

This decision was made to minimize the amount of inputs and outputs that would be required for the device. The MUX controls which sensor the controller gathers data from. The MUX allows us to send one signal for the slave select from the controller and then program which sensor it will activate. This is controlled through the A1 and A0 lines on the MUX. These take in a binary command to switch from the first to the fourth sensor. These lines can be wired up with other A1 A0 lines from other modules as they simply allow the sensor data to be cycled from one to the other.

To increase the modularity of the circuit, the enable pin was wired to the input voltage. When the enable pin is powered the MUX is given power, but does not send data until the slave select pin is also given power. This allows for the same functionality without needing as many inputs and outputs to the system.

The integration of the MUX allows the initial module to require 3 pins (A0, A1, and the Slave Select pin) and only one additional pin, the slave select pin, for each subsequent module. This is a big improvement from requiring 4 pins for each module, one from each additional Hall Effect sensor.

XOR Gate

An XOR gate was necessary in order for the sensor signals to be properly read. This is because all of the Hall Effect sensors are active LOW meaning they are turned on when their slave select pin is not given power and turned off when they are given power, With just a MUX, we are only able to send a logic HIGH signal to a single sensor while the other three sensors are not given a signal. This resulted in 3 sensors being active at the same time.

An XOR gate was implemented between the sensors and the MUX to reverse the sensor inputs that were going to the Hall Effect sensors. This is done by powering the other side of the XOR gate so that when a positive signal is sent from the MUX it results in a LOW signal to be sent to the desired sensor, while the others will be getting a HIGH signal. This results in only one sensor reporting data while the others remain inactive.

Final Circuit Design

The final circuit was created using Eagle. This board is 29.8 mm in width and 39.0 mm in length. Figure 14 shows an image of the schematic used for the wiring of the circuit as well as

the board layout. One of the largest guiding factors in determining the size of the module was the distance the magnets needed to be from each other in order to prevent magnetic interference. Based on calculations made by previous groups (last year's MQP team as well as MSc students within WPI Soft Robotics Lab working on a similar project, Anand Ramakrishnan and Anany Dwivedi) the magnets need a separation of at least 15 millimeters in order to avoid magnetic interference.

The following package sizes were used for their respective components on the board: a SOIC-8 package was used for the footprint of the Melexis sensors, a modified μ SOIC-10 was used for the MUX, a DIL14 package was used for the XOR gates and a 0805 package was used for the capacitors. The circuit board includes a section for an 8 pin through-hole female connector allowing for it to be more easily wired to the controller.

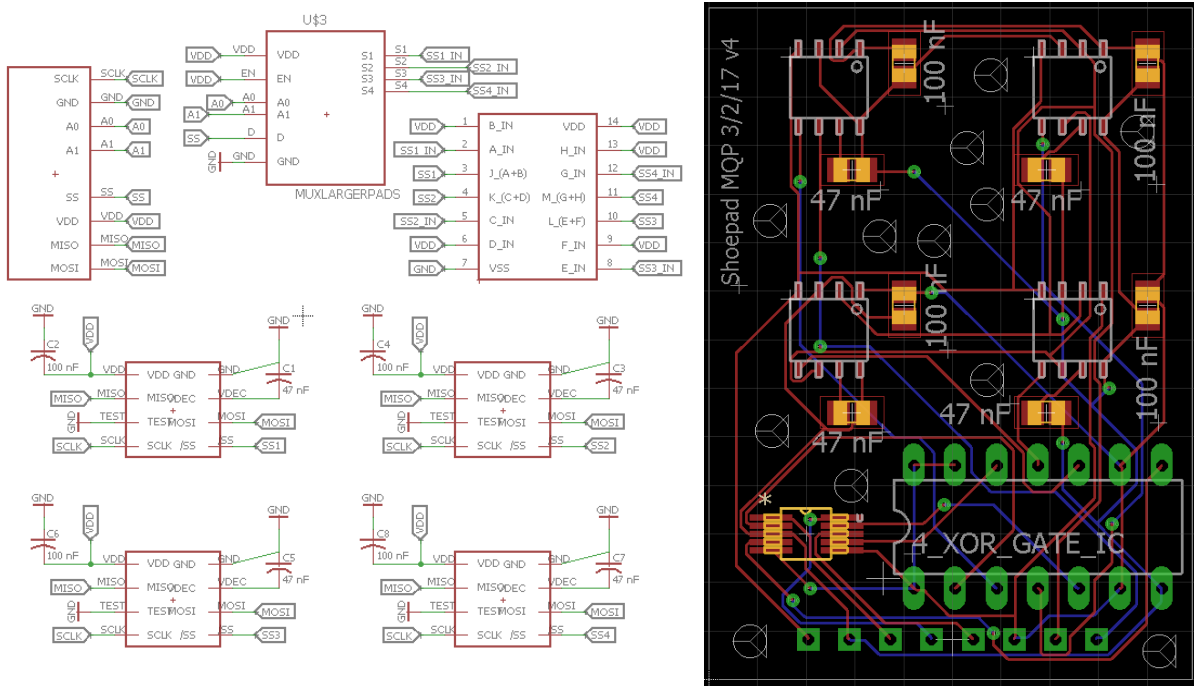


Figure 14: Schematic of Circuit Wiring (left) and Board Layout (right) in Eagle

As mentioned previously the magnets must be at least 15mm from each other. This in turn means that the Hall Effect sensors must also be at least 15mm apart from each other. This was the main factor that contributed to the size of the circuit board.

Once the circuit board had been designed, Gerber files were created and sent to SeeedStudio, a PCB manufacturing company, and the electrical components (Melexis Hall Effect

sensors, MUX, XOR gates, 47 nF and 100 nF capacitors, and connector pins) were ordered from Digikey, a distributor of electrical components. When all the components had come together they were soldered onto the board, seen in Figure 15.

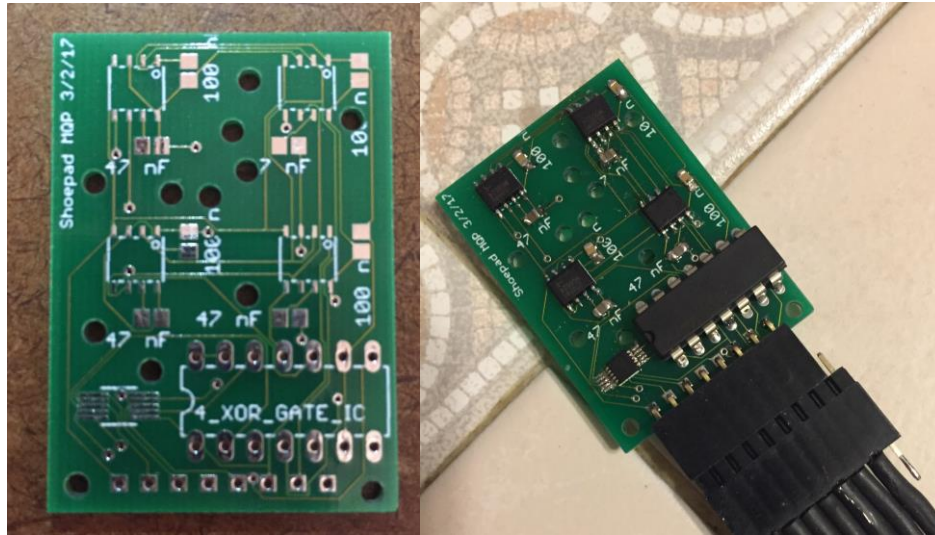


Figure 15: Unpopulated (left) and Populated (right) PCB

5.2 Mechanical Component

In order to house the magnets needed for the device, as well as to protect the device from external forces, Dragon Skin 30 silicone rubber is used to encase the magnets and the board. In this section the design and molding processes are described.

Molding Process

In order to create the silicone used for the module, a mold must first be created. A 3D printed mold was also created so that the silicone piece of the device could be molded. The silicone requires a multi-step process in order to be properly made. The first step creates most of the mold which includes a base as well as wells, or holes, in the silicone. Once this step is completed, the top of the mold would be removed, the magnets would be placed inside and more silicone would be molded on top, locking the magnet in place. The full procedure can be found in Appendix C.

Final Mold Design

The final mold design can be seen in Figure 16 (see Appendix D for more details). It was largely modified from the previous MQP team's 3-by-3 mold. The spacing between the wells is

the same as spacing required for the magnets (15mm). A magnet embedded in a 5mm square acrylic plate will be placed inside the well before it is sealed back up with silicone during the 2nd stage of the molding process. The mold was created so that there would be at least 6mm of silicone between the Hall Effect sensor and the magnet. This was done to ensure that the Hall Effect sensors would not become over saturated when force is applied to them, as they were in previous iterations of the silicone.

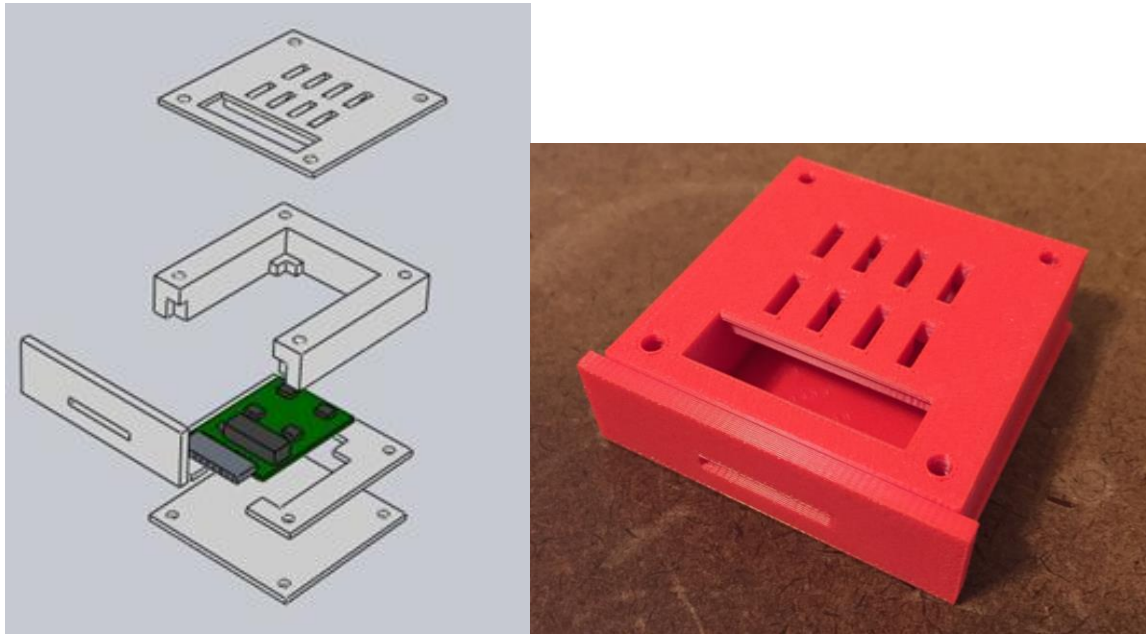


Figure 16: 3D Printed Silicone Mold

5.3 Software Component

The software is the third key component to this project. The software portion is divided into two main sections: Reading data onto the Arduino microcontroller board, and storing and displaying data.

Arduino is used to initialize and manipulate the controller and sensors in order for them to function properly and collect the data from the sensors as intended. Since this project is based on last year's project, many properties and code sections were inherited from them.

One such example is the CRC (Cyclic Redundancy Check) that was written by Selim Ozel (a PhD student in WPI Soft Robotics Lab) and was used in last year's code. The CRC is an error detection code that helps prevent data corruption. CRC works by having a set number to divide by and the remainder is used to check if the output has the same remainder upon

calculation, which is all done in binary. A lookup table makes the check much faster and none of the calculations have to be performed in real time.

The other section that was reused from last year's project was the message sent and conversion that splits the receiving data into the three different axes, Bx, By, and Bz. The message itself follows the initialization and data query from Hall Effect datasheet [44].

Originally the code was written using a for-loop to iterate through the different slave select pins and the sensors as a result; however, our team decided to use a state-machine instead, switching between the states which corresponds to each sensor. This helped with our final iteration in which we moved the slave select switching into the hardware with a MUX for each module. Each of the four slaves routes to each of the four states. Data collection is done at this stage of the process and are retrieved through the MISO line. This data is then filtered using a bandpass filter as well as an offset filter. The processed data that enter the computer through the Serial line is then parsed through one of the three options.

In one instance, the data can be retrieved from the Serial Line and be shown on the Arduino's Serial Monitor. This was used mainly for debugging the code and being able to see the output stream. It's simple; however it provides no functionalities other than displaying data.

In another case, the data can be retrieved through the Serial Port and pushed through PuTTY. PuTTY is a terminal emulator application software that can help display and record data as well as other things like SSH (Secure Shell) into a network, like WPI's CCC machine. But using this program to read the data coming in from the Serial Port with the same Baud Rate, the data can also be stored straight into a .txt document, by logging the data. We import the data from the .txt file into a graphing tool like Microsoft Excel or MATLAB. By graphing the logged data, we are able to determine the trends that are not as clearly seen in a Serial Monitor display.

Finally, the data can be parsed from the Serial line into a Java GUI (Graphic User Interface) and displayed using graphical means as seen in Figure 17. In our case, we used a progress bar to show how the data is responding to the forces applied.

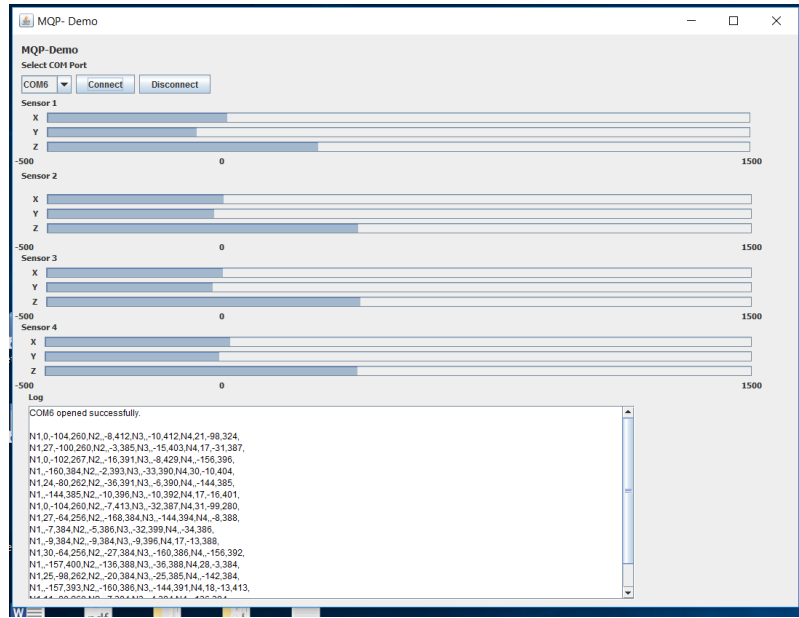


Figure 17: Screenshot of Java GUI in Action

Chapter 6: Methodology

The purpose of testing and calibrating the sensors was to determine the viability of the 2x2 module to serve as a proof of concept for relating displacement of embedded magnets to a force vector applied in three dimensions. The results were analyzed qualitatively to confirm trends in data that correlated to the direction and relative magnitude of the force applied. The normal force data was also compared to a COMSOL simulation to confirm its accuracy.

6.1 Load Cell Description

Force testing was performed on an Instron 5544 electromechanical uniaxial testing device equipped with a 2000 Newton (N) load cell with a linearity of 0.1% (i.e., ± 2 N). The Instron allowed for a uniform distributed force to be applied to the top face of the sensor module. An image of the testing device can be seen in Figure 18.



Figure 18: Instron 5544 Uniaxial Testing Device

6.2 COMSOL Simulation

As a means of validating our designs, we used COMSOL Multiphysics version 5.2 (Classkit License) to simulate the response of the sensor array. Using the Multiphysics model to

simulate both the physical deformation and magnetic fields involved with the sensor array, the “ideal” behavior of the sensors under various loads could be observed and compared to the actual raw data collected from the sensors. In general, the model steps through different loads applied to the magnets in the silicone rubber body, recording their physical displacement, along with the resulting magnetic flux values at the point where the Hall Effect sensor is located. By comparing the simulated magnetic flux values to the actual data given by the sensors, the accuracy of the device could be confirmed during calibration.

The COMSOL model was made using the CAD tools from within the program. Because the program slows down when the complexity of the model increases, a simplified version of our final silicone mold was used, while retaining the proper spacing between the magnets and the sensors. As determined by the WPI Human Augmentation Lab, the magnets were kept 15 mm apart from each other, and suspended 6 mm above the Hall Effect sensors to help prevent data saturation. The early iterations of the COMSOL model design can be seen in Appendix E. The most recent model can be seen in Figure 19.

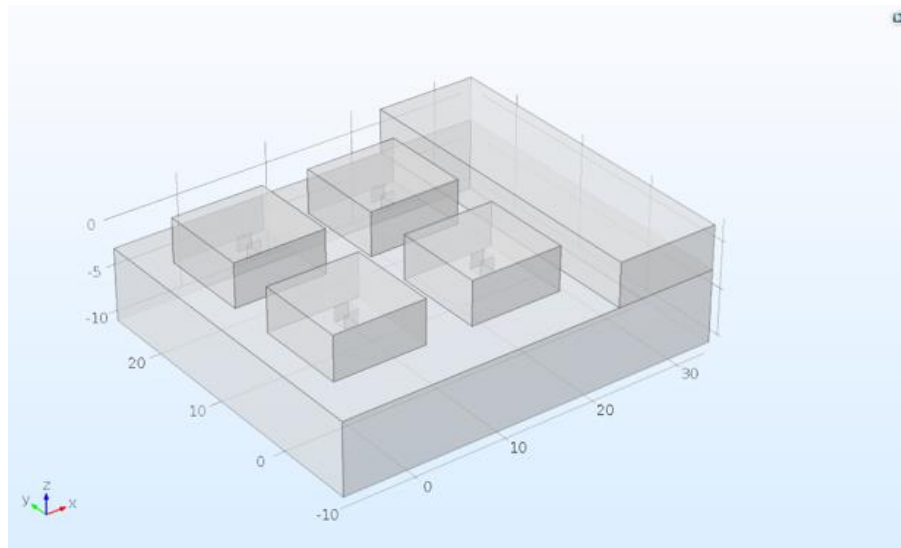


Figure 19: Simplified COMSOL Model of the Sensor Module

With the geometry of the sensor array created, physical and magnetic properties were assigned to the model for the silicone and the neodymium magnets. The hyperelastic behavior of the silicone was defined using a 3-parameter Ogden material model, with coefficients obtained from Ecoflex 0030 silicone rubber. To simulate the behavior of the array under varying forces, two studies were set up in the model. The first study used a parametric sweep to step through a

range of specified loads, while the second study plotted magnetic flux density at each of the same loads. A full list of material parameters and properties used in the COMSOL simulation is located in Appendix F. For each of the following testing protocols, an identical simulation will be run in COMSOL to provide results to compare against the raw data.

6.3 Force Testing

Different procedures were followed for normal force testing and shear force testing. The forces for each test were determined from weight calculations based on an overweight male of height 5'10" and a US size 10 shoe. In both cases, once the appropriate forces were applied, data was recorded using the Arduino Mega and logged by PuTTY, a data logging software. This data was then sent out as a text file (.txt), after which it was processed using Microsoft Excel.

6.3.1 Force Calculations

Taking the average height of a male in the US to be about 5'10" [45], the weight of an overweight individual was determined to be about 90 kg or 883 N, based on ideal body mass index for a male of that height [46]. The area of the bottom of a size 10 foot was estimated to be 523.75 cm², using the bottom of a US men's size 10 shoe as reference [47]. The details of the foot area calculation can be found in Appendix G. Two different calculations were performed to determine the maximum normal force and shear force to apply to the sensor module. Using the determined body weight and foot area, the pressure on the foot due to normal force was calculated. This pressure was multiplied by the area of the 2-by-2 sensor module to determine the maximum normal force to apply. A safety factor was applied to account for non-even distribution of forces under the foot. The normal force determined for testing using this method was 200 N.

For shear forces, the max shear force was taken to be 15% of the person's body weight [48, 49], and assumed to cover 25% of the sole of the foot at one time. This information was then used to find the shear pressure exerted on the foot, which was used with the area of the 2-by-2 module to calculate the maximum shear force to apply. The shear force determined for testing using this method was 19.56 N. The maximum force calculations can be seen in Appendix H.

6.3.2 Normal Force Testing

Normal force testing was performed by applying an evenly distributed normal force on

top of the module using the Instron 5544 Uniaxial Testing Device. This force ramped up from 0 N to 200 N at a rate of 200 N/min. Data was collected from each of the four Hall Effect sensors, and the relationship between magnetic flux and applied force was observed. This test was performed three times, and the data was compared to a simulation run in COMSOL. Due to restraints on the finite element model, the COMSOL simulation was only able to run a 60 N normal force test using a ‘coarser’ mesh. Figure 20 shows how the normal force testing was set up.

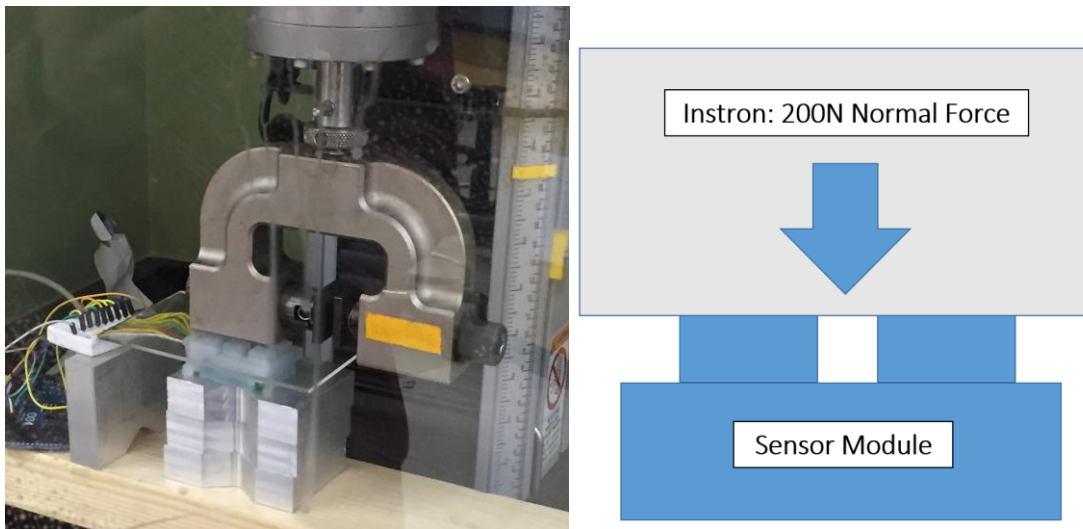


Figure 20: Normal Force Testing (left) and FBD of Normal Force Testing (right)

6.3.3 Shear Force Testing

Shear force testing was done by applying a 21 N normal force and using weights to create a 21 N shearing force. The weights were hung over the edge of testing area so that they would be pulling on the module in a perpendicular angle from the normal force. With the 21N normal force and horizontal force applied simultaneously, the resultant force acting upon the sensor was 29.7 N at a 45 degree angle. During the shear force testing, the applied force was held constant for 5 seconds. While the applied force for shear testing was greater than the maximum force calculated in Section 5.3.1, it was the closest option given the limited amount of weights available for testing. Figure 21 show the free body diagram and setup used for shear force testing. Near the end of the project, arbitrary shear forces were also exerted on the sensor module to confirm the validity of the x-axis readings, which were incorrectly recorded during one testing session. These results are also included in the next chapter.

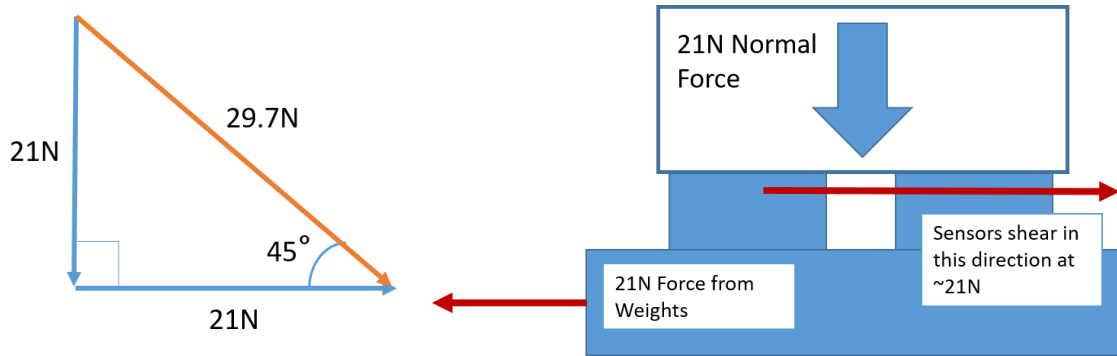


Figure 21: FBDs of Shear Force Testing

An apparatus was created to ensure that the module would only move in the desired direction. Figure 22 shows the apparatus (see Appendix I for more details) . This was designed so that the module can be placed in multiple orientations, allowing for testing in the positive and negative directions of the x and y axes. Due to difficulties with the COMSOL model becoming unstable when simulating shear forces, no COMSOL data was available for this test.

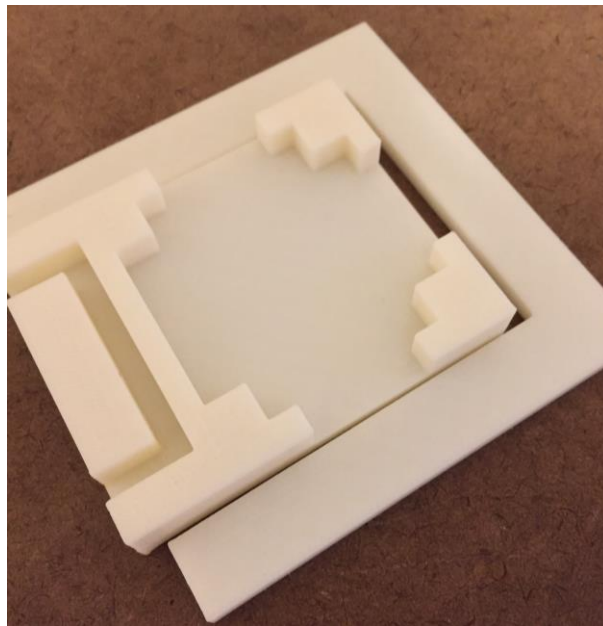


Figure 22: Apparatus used for Shear Force Testing

6.4 Testing for Modularity

In order to test the modular design of the MagneForce circuit, 2 modules were connected by wiring all of the input and output lines of the modules together, except for the slave select line on the MUX. This allowed for all the data to be received from all of the different sensors, which

would not report data unless their slave select pin was chosen. This test was run to confirm if multiple modules could collect data in series. An image of the experimental setup is found in Figure 23 and the Arduino code for the test can be found in Appendix J.



Figure 23: 2 Modules Connected for Modularity Testing

6.5 Data Analysis

6.5.1 Normal Force Data

Normal force data was processed first using Excel to visualize the magnetic flux data graphically. To do this, data was parsed from Arduino into a PC through the Serial line. The data was then collected from the Serial line using PuTTY, which then logged the data into a .txt file upon closing. From there, the .txt file was imported to Excel. This imported data was separated by columns using the ‘,’ as a delimiter and separated by rows using the new line character ‘\n’ for the delimiter. Once the data was in the Excel file, a line graph was then plotted for each axis of each sensor, plotting magnetic flux (in microtesla units) against applied normal force (in N). The trends from the graphs were compared qualitatively to the 60 N normal force simulation produced by the COMSOL model. A Java GUI was also developed to display the data; however, this program had a high input lag and could not be used in real time. A description of the program, along with the code, can be found in Appendix K.

To further confirm the accuracy of the normal force data, the data were processed using a custom code to graph each test and run a linear regression on the data. From the linear regression, R-squared values for both the raw data and COMSOL model were calculated. The R-squared values were used to compare the similarity in trends between the two data sets. The

MATLAB code used in this analysis can be found in Appendix L.

6.5.2 Shear Force Data

Shear force data was also processed using Excel. Magnetic flux was graphed against time for the shear force tests, as only a single force was applied for a period of time. The trends in the data were observed qualitatively to determine if the direction of the shear force applied was represented.

6.5.3 Data Filtering

Due to the amount of noise in the system, a band pass filter is applied when data is collected by Arduino. This filter eliminates spikes that are over 2000 μT and under -2000 μT . These numbers are chosen because values over and under 2000 μT indicate saturation of the sensors, and such values are unusable.

Data from the Z-axis showed irregular shifts in the value output. To remedy that, an offset filter was applied right after the band pass filter to adjust for shifts in z-axis data, see Appendix M. This filter offsets values that represent jumps larger in magnitude than 150 μT (determined manually as a value unreasonably high to represent natural progression of loading) and this reduced the effect of the jumps. This value used, however, will need to be tuned for each module, as each set of sensors behaves differently.

Chapter 7: Results

This section details the results of the normal force and shear force tests performed on the final sensor module. These results include data from the 60 N COMSOL simulation, the 200 N normal force tests, and the shear force tests with the 29.7 N force applied at a 45 degree angle.

7.1 COMSOL Results

As mentioned in the previous chapter, the COMSOL simulation was only run up to a 60 N normal force, as the model would crash at greater forces, as well as under shear loading. While unfortunate, the latest iteration provided a sufficient trend to compare to the raw data as well as to run a linear regression. Figure 24 shows the model under a 60 N normal force, and Figure 25 gives the graph of simulated magnetic flux in microtesla versus applied force.

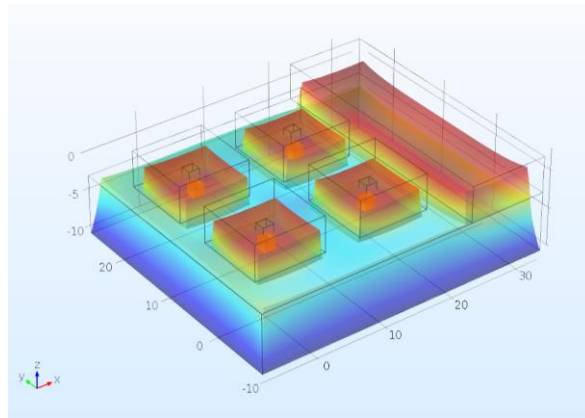


Figure 24: COMSOL Model under a 60N Load

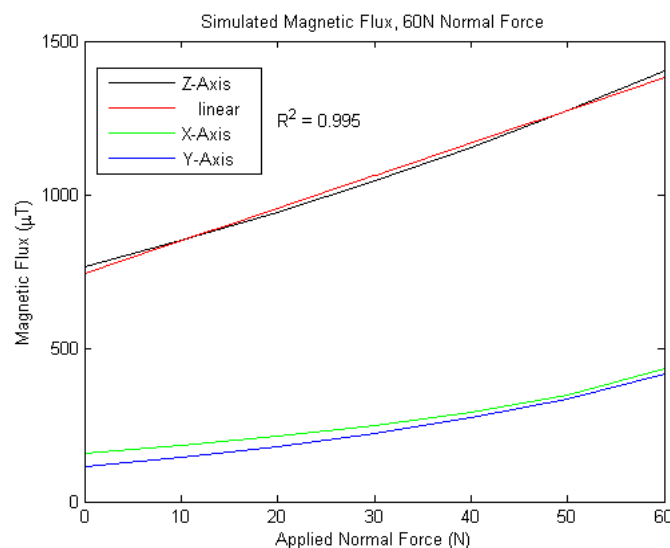


Figure 25: Simulated Magnetic Flux (Ecoflex 0030 Silicone) Rubber

As seen in the graph above, magnetic flux increases from roughly 750 μT at 0 N to roughly 1400 μT at 60 N of normal force. The x- and y- axes also increase to a much lesser degree, both from roughly 100 μT at 0 N to about 400 μT at 60 N. The red line on the graph indicates the linear regression on the z-axis data, and the R-squared value for the z-data is 0.995, indicating a strong linear trend between magnetic flux in the z-axis and the amount of force applied. Data from the COMSOL simulation can be found in Appendix N.

7.2 Normal Force Results

For the 200 N normal force test performed on the sensor module, magnetic flux in the z-axis also increased with increasing normal force applied. A representative graph from a single sensor is shown in Figure 26.

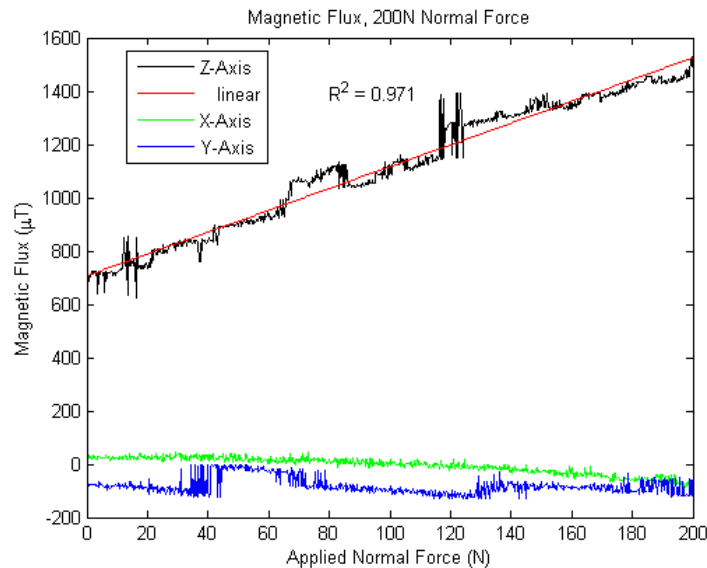


Figure 26: Data from 1 Sensor during Normal Force Testing (Dragon Skin 30 Silicone Rubber)

As shown by the graph above, magnetic flux in the z-axis of the sensor increases from roughly 700 μT at 0 N of applied force to about 1500 μT at 200 N of applied normal force. The x- and y- axis data experienced minimal change with increasing force, with the y-axis staying close to 0 μT and the x-axis close to 75 μT . The red line on the above graph indicates the linear regression, with the calculated R-squared value being 0.971, indicating a strong linear increase in z-axis magnetic flux with increasing normal force. In addition, the data experiences small jumps of roughly 50 μT throughout. The magnetic flux response was also consistent across all four sensors, as seen in Figure 27.

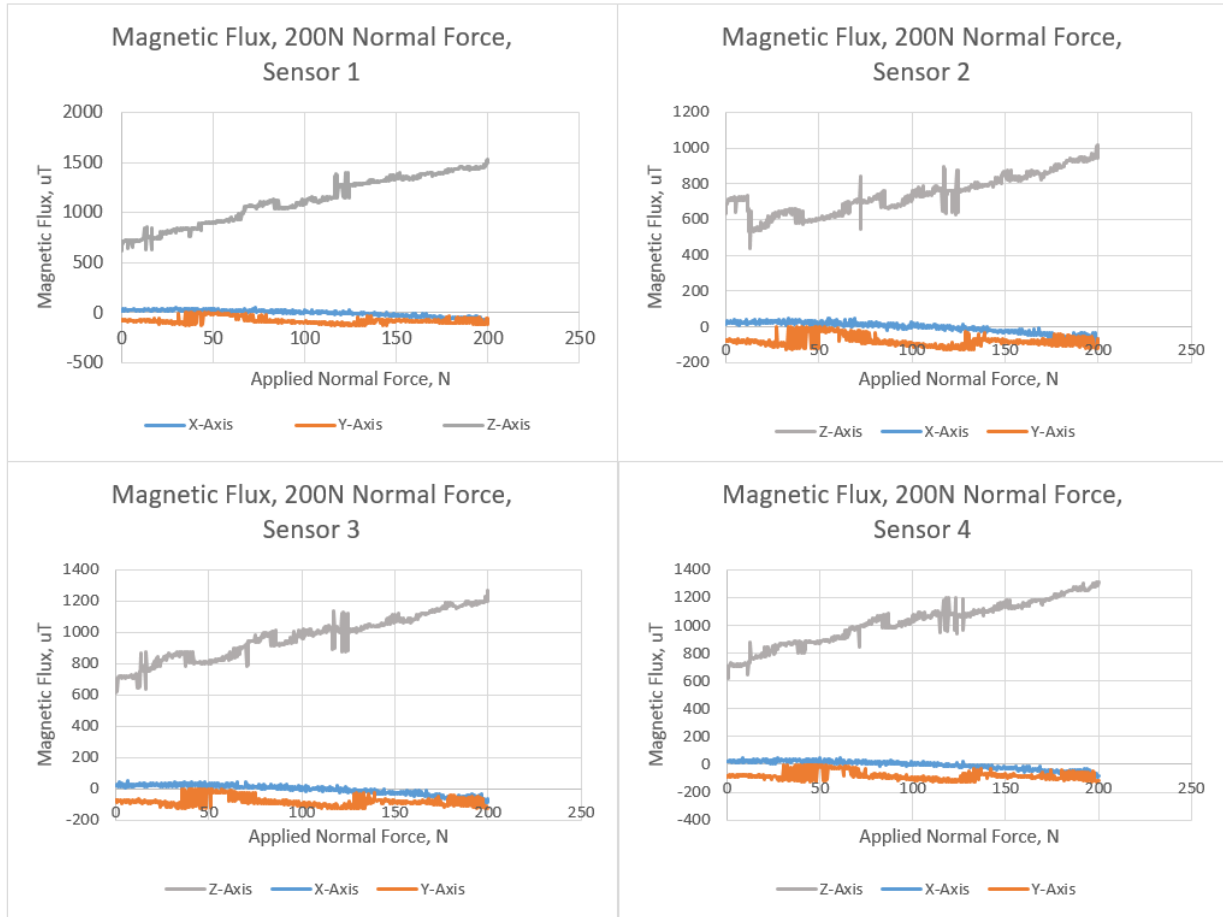


Figure 27: Normal Force Testing Data for all 4 sensors on the module during 1 trial

Across all four sensors for the same normal force trial, magnetic flux in z increased with increasing normal force applied, while the x and y axes remained relatively constant.

7.3 Shear Force Results

For the shear tests performed with a constant 29.7 N force applied at a 45 degree angle to the XY-plane, there were consistent trends in magnetic flux indicating the direction of the applied force. Representative graphs for one sensor tested in the negative y and the negative x directions are shown in Figure 28 A graph for the arbitrary shear force applied in the positive x-direction (to confirm functionality in that direction as described in Section 5.3.3) for the same sensor can also be found in Figure 29.

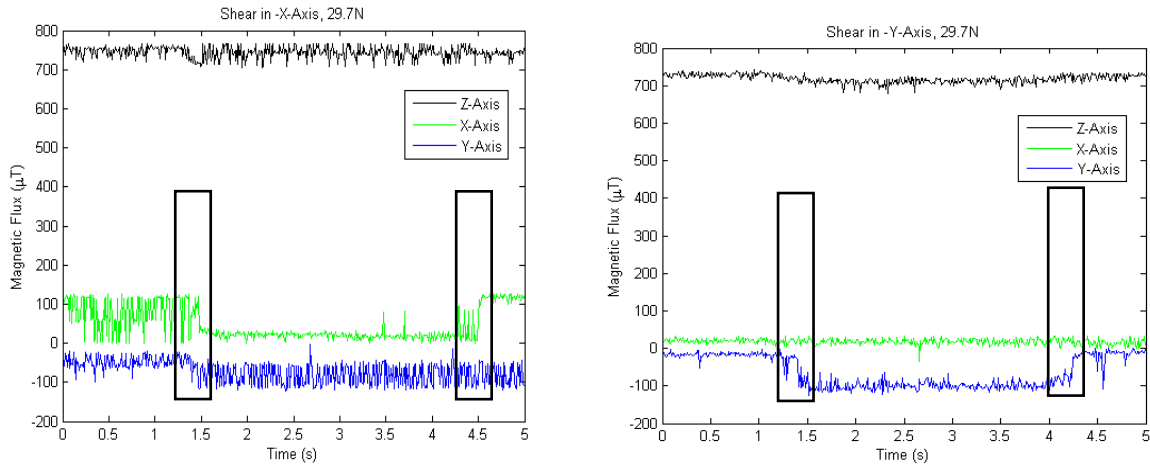


Figure 28: Shear Force Graphs in the Negative x (left) and Negative y (right) Directions. Boxes Show when Force was Applied and Removed

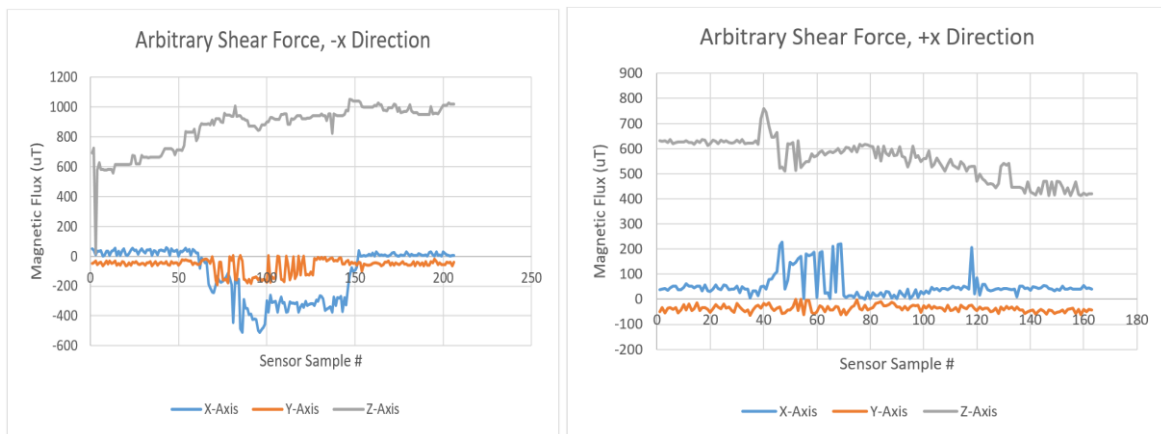


Figure 29: Shear Force Data for the Negative (left) and Positive (right) x Direction

The data from the shear testing shows a consistent trend in magnetic flux in the direction of the applied force. This is true in both the positive and negative directions along both the X- and Y-axis. In each direction (for the 29.7 N force), the change in magnetic flux when the shear force was exerted was about $100 \mu\text{T}$ in each respective axis independently. Moreover, each axis that does not have forces applied does not show any significant changes in terms of magnitude of their flux readings. For example, when the shear force was applied in the negative x -direction, the magnetic flux readings along the x -axis decrease by about $100 \mu\text{T}$; the y - and z - axes stay relatively consistent, with minimal changes of about $10 \mu\text{T}$ occurring in the z -axis. The jumps in sensor data are further discussed in Chapter 8.7.2.

7.4 Modularity Results

The modules were tested to see if all 8 sensors on two modules would be able to report data individually. These modules were wired together as previously stated in Chapter 6. When tested, the sensors all reported data independently of each other when they were called upon to do so, as illustrated in Figure 30.

```
COM3 (Arduino/Genuino Mega or Mega 2560)
PinSS1: N1,-28,-12,-96,PinSS1: N2,-16,-14,-102,PinSS1: N3,-20,-15,-76,PinSS1: N4,-17,-16,-95,
PinSS2: N1,-7444,-157,349,PinSS2: N2,-7680,-156,272,PinSS2: N3,-16,-159,256,PinSS2: N4,-2048,-31,341,
PinSS1: N1,-17,-14,-127,PinSS1: N2,-32,4832,-98,PinSS1: N3,-14,-15,-75,PinSS1: N4,-7,-16,-128,
PinSS2: N1,-7454,-149,343,PinSS2: N2,-7673,-160,256,PinSS2: N3,1,-156,263,PinSS2: N4,-2046,-29,339,
PinSS1: N1,-16,-8,-108,PinSS1: N2,-27,-3,-70,PinSS1: N3,-8,-13,-99,PinSS1: N4,-24,-15,-100,
PinSS2: N1,-7448,-141,341,PinSS2: N2,-7677,-159,267,PinSS2: N3,1,-154,256,PinSS2: N4,-20,-29,346,
PinSS1: N1,-20,-12,-92,PinSS1: N2,-32,-13,-60,PinSS1: N3,-23,-13,-60,PinSS1: N4,-15,-16,-90,
PinSS2: N1,-7440,-141,337,PinSS2: N2,-7677,-155,264,PinSS2: N3,6,-160,264,PinSS2: N4,-2039,-29,346,
PinSS1: N1,-26,-15,-126,PinSS1: N2,-14,-13,-102,PinSS1: N3,-8,-1,-67,PinSS1: N4,-16,-14,-124,
PinSS2: N1,-7447,-140,341,PinSS2: N2,-7663,-157,260,PinSS2: N3,8,-153,266,PinSS2: N4,-2042,-32,351,
PinSS1: N1,-31,-32,-127,PinSS1: N2,-19,-23,-102,PinSS1: N3,-2,-15,-100,PinSS1: N4,-19,-14,-128,
PinSS2: N1,-7456,-141,342,PinSS2: N2,-7677,-160,266,PinSS2: N3,1,-159,256,PinSS2: N4,-2048,-23,342,
```

Figure 30: Data Log of 2 Modules Running in Series

Chapter 8: Discussion

Overall, our results indicate that our MagneForce module serves as a proof-of-concept for relating change in magnetic field to the magnitude and direction of force applied. This section discusses the biggest implications of our results as well as how the final product matched up to our matrix of objectives from Chapter 3.

8.1 COMSOL Model

As mentioned in the previous two chapters, the COMSOL model was somewhat limited in that it would crash under high loads and shear forces. While working with the model, it became clear that as the complexity of the model increased, the simulation had greater difficulty running through completion. As a result, the geometry needed to be simplified, and the simulation needed to be run using a ‘coarser’ mesh in order to acquire a satisfactory trend. While these modifications reduce the accuracy of the simulation, they were necessary to create a functional model in the first place. Additionally, the resulting simulation was appropriate for an initial validation of our device. Future iterations of this project will require more time to be spent with the COMSOL program in order to build a more stable model in COMSOL that can simulate higher normal forces as well as shear forces.

8.2 Normal Force Data

As demonstrated by both the COMSOL simulation and the experimental normal force data, the magnetic flux in the z-axis increases with increasing amounts of normal force applied. For both the COMSOL model and the sensor data, the R-squared values were 0.995 and 0.971, respectively, indicating strong linear trends for both sets of z-axis data. The similarity of the R-squared values confirms the similarity in z-axis trends between the experimental data and the COMSOL model.

However, while the R-squared values are close, indicating similarity, the slopes of the z-axis between the experimental data and the COMSOL data are not the same. The COMSOL simulation increases from 750 μT at 0 N to roughly 1400 μT at 60 N ($\sim 10.8 \mu\text{T}/\text{N}$), while the data in Figure 26 increases from 700 μT at 0 N to roughly 1500 μT at 200 N ($\sim 4 \mu\text{T}/\text{N}$). This difference likely results from the material properties of the COMSOL simulation. The COMSOL model used the mechanical properties Ecoflex 0030 silicone rubber, as the coefficients for the

Ogden hyperelastic model for that material were readily available. Our MagneForce module, on the other hand, used Dragonskin 30 silicone rubber, which is stiffer than the Ecoflex. Dragonskin 30 was used for the product as the material properties were much more desirable; the increased stiffness held the magnets in place well, and did not deform too much under high loads, protecting the circuitry. Considering all these factors, the differences in slope are sensible, as the stiffer Dragonskin 30 did not deform as quickly, resulting in a slower increase in magnetic flux with increasing applied force. The lower stiffness of the Ecoflex 0030 also explains the higher magnetic flux values in the COMSOL simulation, as it would deform more under at lower forces, giving higher magnetic flux readings.

When comparing the two other axes (X and Y), the readings from the actual test differs from the simulated COMSOL values. In the simulation, readings from both X and Y trends upwards. This is not the case for the actual reading, however. The data trend of the sensor module is more constant without much deviation. These differences are likely due to the limitations of the COMSOL simulation, as it often had difficulty modeling the hyperelastic behavior of silicone rubber. However, the x- and y- data from the normal force experiments remained relatively constant, indicating that little to no shearing occurred for an applied normal force, as expected.

8.3 Shear Force Data

Looking at data produced through the shear force test, there is a consistent relationship between the changes in the magnetic flux in X and Y axes and the forces applied in the same directions. The changes in the positive and negative X and Y position are as expected. As the force is applied to the positive X, the magnetic flux reading in the X direction increases, as force is then applied in the opposite direction, the magnetic flux reading decreases and shifts the other way. Furthermore, the magnetic flux does not change for axes in which force is not applied. The normal force was applied from the beginning of the experiment to create a frictional force, thus a change in Z should not be observed. This holds true with the results of our test. Overall, these results show that the magnetic flux data can be related to the direction of the force applied, which can be used to detect shear forces.

8.4 Discrepancies in Data

Across all experimental data, small jumps in the data collected were observed, ranging from $\sim 10 \mu\text{T}$ to $\sim 50 \mu\text{T}$. Because the size of the jumps were consistent in magnitude, the band-pass filter we coded worked for adjusting most of the data jumps. The jumps in data could possibly be occurring as a result of the soldering connections on our circuit board; applied force could be causing components to shift slightly, resulting in the jumps in data. Alternatively, the error could be occurring during the communication over the computer's serial port; the code used to read and parse through the sensor data may not be optimally reading data from the sensors.

8.5 MagneForce as a Proof-of-Concept

8.5.1 Functionality

We were able to improve upon the previous iteration of this device by ensuring it would function properly and be able to detect normal and shear forces. This is greatly due to ensuring there would be no magnetic interference created by the other magnets in the module. Though this became the limiting factor in determining the size of our module, it was necessary in order to get accurate data for both normal and shear forces up to 200 N and 21 N respectively. By spacing the sensors 15 mm apart from each other and placing the magnets directly above the sensors, interference and crosstalk between the sensors was successfully avoided.

8.5.2 Modularity

The final size of the module is 36mm x 46.6 x 17.4 mm. This size allows for many different modules to be placed in a single shoe pad allowing for customization based on the shoe size. The modular concept was also proven when all 8 sensors were producing data when 2 modules were wired together. This allows the system to be adaptive to any size that may be required for the patient.

8.6 Cost Analysis

It was determined that 7 modules can fit inside a size 10 shoe. From this value we can determine the amount of parts required for a shoe pad made of our MagneForce modules. The following table outlines the parts needed for 7 modules, as well as their quantity and cost. The total cost for assembling this shoe pad comes out to be \$164.03.

Table 3: Bill of Materials for 7 Professionally Made Modules

7 Professionally Made Modules		
Quantity	Description	Cost
10	PCBs	\$9.99
28	Hall Effect Sensors	\$105.84
28	47nF Caps.	7.34
28	100nF Caps.	1.85
7	XOR Gates	2.73
7	Multiplexers	\$19.67
7	8 pin header	\$6.58
	Dragon Skin 30	\$10.03
	Total:	164.03

The cost of parts for the MagneForce sensor is \$164.03 per shoe pad. This cost is significantly cheaper than the price of our competitors, which can be seen in the following table:

Table 4: Cost Comparison of MagneForce and its Competitors

Product	Description	Price	Shear Force?
MagneForce	7 modules in a shoe pad	\$164.03	Yes
Tekscan F-Scan	Resistive force sensors in a shoe pad	~\$4,000	No
Pedar System	Force sensors in a shoe pad	~\$5,000	No

Tekscan’s F-Scan System costs roughly \$4000 USD [28], and the Pedar System by Novel costs \$5000 [28]. While our price does not include labor cost or profit margins, after scaling for bulk manufacturing, we expect that the price of the proposed system will still be an order of magnitude lower than existing commercial systems. Additionally, existing systems only measure

normal forces while the MagneForce sensor can measure both normal and shear forces. This gives the product an edge over the competitors in both price and functionality.

8.7 Evaluation of Project Objectives

The 6 most important project objectives we were looking to accomplish were: safety, sensor functionality, affordability, gait monitoring, non-invasiveness, and reliability.

8.7.1 Safety

Safety was rated as the most important objective. Our device is extremely safe as it is encased in silicone, providing the circuitry protection from the user and the user protection from the circuitry. Because of this, this objective has been properly accomplished.

8.7.2 Sensor Functionality

One of the next most important objectives was to have sensor functionality. Given our results, our sensors can be used to determine the normal force applied based on the magnetic field of the magnet, as well as the direction of shear forces. Given these two factors as well as the fact that the sensors are modular and can be combined, this objective has also been met.

We also achieved a refresh rate of 70 Hz for a single module, along with a spatial density of 1 sensor per 15 mm². The refresh rate is higher than the rate achieved during last year's project iteration (33 Hz), but an even higher refresh rate will be useful in future iterations for better data collection. The spatial density also has room for improvement. While the sensor density is currently limited by the minimum distance to avoid magnetic cross-talk, using weaker or smaller magnets could solve this issue.

8.7.3 Affordability

The objective of affordability was also met. Given our cost analysis in the previous section, our device is quite affordable, especially when compared to the competitors' prices, meeting our objective of affordability.

8.7.4 Gait Monitoring

The objective of gait monitoring was not met. Time constraints and time spent troubleshooting the module functionality prevented us from successfully integrating an IMU into the device. Bluetooth communication was also not achieved, as proving the ability of the device to relate displacement in magnetic field to applied force was of higher priority.

8.7.5 Non-invasiveness

As for non-invasiveness, this objective was met in part. In its current state, the sensor module is rather thick, and a shoe would need to be modified or custom-made in order to house the modules. However, future iterations can reduce the size to make it more accommodating.

8.7.6 Reliability

The goal of reliability was also not fully realized, as time constraints on the project limited the amount of experiments that could be run. As a result, the body of results was not large enough to determine reliability of sensors over a large number of cycles.

Chapter 9: Recommendations and Conclusion

9.1 Future Recommendations

While the MagneForce module currently serves as a proof of concept for relating displacement of magnetic fields to applied forces in three dimensions, there is still potential for improvement to make the device a more complete product. Improvements can be made to the module itself, and gait monitoring features can be added to increase the device's versatility.

9.1.1 Module Improvements

As with any device, modifications can always be made to improve upon the module. Five of the major improvements are as such: refining the circuit design to accommodate for a more dynamic change in force, reducing the signal noise, reducing the thickness of the modules, and testing reliability over time with a large number of cycles.

Currently, the modules cannot handle highly dynamic forces. That is, rapidly changing forces at high magnitudes (such as with running or tripping) are liable to be filtered out. For example, if a person were to stomp on the module, the device would likely filter out the stomp because it created a data spike close to a saturation value. Modifying the device to better handle these types of forces would make the product much more robust. This could potentially be accomplished through improving the filtering run through the software, or improving the circuit design to limit noise so that the filter does not need to be as sensitive.

As just previously mentioned, noise in the data collected by the sensors is an area with potential for improvement. While the signal noise was greatly reduced from bypassing the breadboard, high spikes in the data still appeared that needed to be filtered out. The greatest source of the signal noise is probably the inconsistency that is inherent with hand-soldered circuits. Because all of the circuit components were soldered on manually, inconsistencies occurred with wires and components coming loose, introducing noise. Having the circuits professionally assembled would likely solve this issue, and vastly improve the consistency of the circuit construction.

Additionally, the current module is a bit too thick to be added as an insole, and the sole of the shoe would have to be carved out in order for the modules to fit. If improvements can be made to reduce the module's thickness, but still keep the encasement of the PCB in silicone to a

size that can be fit comfortably inside a regular shoe, this would improve the module's ease of use. This modification may be difficult given that moving the magnets too close to the sensors would cause more saturated readings, but this could be remedied by using weaker magnets.

In order to prove that the current modules would be suitable for frequent use, reliability over time must be tested. The device would benefit from testing that it can withstand strains over a large number of cycles and over a long period of time. In addition, the consistency of the data over many cycles needs to be confirmed as well, to ensure that the readings remain accurate over an extended period of time. Using a dynamic load cell that can cycle through a test multiple times could be useful, and the hysteresis of the measured data could be observed.

9.1.2 Added Features

Aside from improving the circuit module itself, there are other features to add to the device that would make it more complete as a product. The first such feature is a program to graph or visualize collected data in real time. The current method of data collection involves logging the data and graphing it manually in Excel. This approach is inefficient, and unnecessarily complicates the process of collecting data and troubleshooting the device. Having a program to display a graph of the force data collected would not only improve how the force sensors are tested, but would also provide a platform for clinicians and patients to view the data in the future. Later generations of the product could also have a user-friendly app, which could allow patients to view their own data while also allowing them to share it with their doctor. Having an app that can be integrated to monitor what is going on in real-time would greatly improve both functionality of the device as well as the ease of use for the user.

Another useful feature to add would be an IMU, to monitor the position and movement of the patient's foot and leg throughout the gait cycle. While the sensor module itself would work to detect normal and shear forces, adding an IMU would give it the functionality of a complete gait monitoring device.

9.1.3 Future Regulatory Considerations

Since future iterations of this device will likely be used to aid in diagnosis and treatment of diabetic ulcers, it is probable that the MagneForce product will need to be classified as a medical device. Medical device classification requires a variety of standards to be met, both in fabrication of the device as well as in testing and clinical trials. Here we discuss a few of these

potential considerations.

First, because the sensor module is non-invasive (external to the skin) and thereby poses minimal health risk to the patient, yet more complicated than simpler devices (like bandages), future iterations of the device would likely be classified as an FDA Class 2 medical device [50]. By virtue of being a Class 2 device, our product would be subject to FDA evaluation, and would be subject to FDA regulations for proper use following that evaluation.

Additionally, the medical device classification requires that standards be met regarding biocompatibility and safety to the patient. The biggest standard to meet with regard to biocompatibility is ISO standard 10993, which governs evaluation of medical device biocompatibility [51]. This standard covers an extensive range of device types and considerations for both internal and external products. Specifically, ISO 10993 includes an assessment to identify and mitigate potential biocompatibility risks. This assessment would include a literature review of related devices, a review of clinical experience of similar devices, and a review of results on animal testing of similar devices.

Again, biocompatibility and safety should be of the highest importance for future product iterations. Another standard we can use to ensure patient safety is ASTM F720-13, which details protocols for materials testing on guinea pigs for contact allergens [52]. While silicone rubber is generally regarded as safe for contact with skin, performing tests on guinea pigs with samples of the Dragonskin 30 will confirm if prolonged contact with the silicone would cause any adverse reactions with the skin. Overall, end user safety is a subject that cannot be overlooked in future project iterations.

9.2 Conclusion

Overall, the MagneForce module currently serves as a proof-of-concept for relating displacements in magnetic fields to applied forces in three dimensions. Using this concept, the sensor module can be used to measure both normal and shear forces. With the proper modifications and improvements, MagneForce can be integrated into a shoe pad to measure the forces on the bottom of the foot, helping diabetes patients prevent and manage foot ulcers. As a whole, this device has great potential to become a successful household or clinical product.

Works Cited

- [1] Centers for Disease Control and Prevention. (2014, October 4). *2014 Statistics Report | Data & Statistics | Diabetes | CDC*. Available: <http://www.cdc.gov/diabetes/data/statistics/2014statisticsreport.html>
- [2] National Institute of Diabetes and Digestive and Kidney Diseases. (2016, October 4). *Choose More than 50 Ways to Prevent Type 2 Diabetes | NIDDK*. Available: <https://www.ncbi.nlm.nih.gov/pubmed/>
- [3] Mayo Clinic. (2015, October 4). *Diabetic neuropathy*. Available: <http://www.mayoclinic.org/diseases-conditions/diabetic-neuropathy/basics/definition/con-20033336>
- [4] American Podiatric Medical Association. (2016, October 4). *Diabetic Wound Care | Foot Health | Learn About Feet | APMA*. Available: <http://www.apma.org/Learn/FootHealth.cfm?ItemNumber=981>
- [5] W. J. Jeffcoate and K. G. Harding, "Diabetic foot ulcers," *The Lancet*, vol. 361, no. 9368, pp. 1545-1551, 5/3/ 2003.
- [6] E. C. Katoulis, A. J. Boulton, and S. A. Raptis, "The role of diabetic neuropathy and high plantar pressures in the pathogenesis of foot ulceration," (in eng), *Horm Metab Res*, vol. 28, no. 4, pp. 159-64, Apr 1996.
- [7] J. E. Perry, J. O. Hall, and B. L. Davis, "Simultaneous measurement of plantar pressure and shear forces in diabetic individuals," *Gait & Posture*, vol. 15, no. 1, pp. 101-107, 2// 2002.
- [8] M. Yavuz, A. Tajaddini, G. Botek, and B. L. Davis, "Temporal characteristics of plantar shear distribution: Relevance to diabetic patients," *Journal of Biomechanics*, vol. 41, no. 3, pp. 556-559, // 2008.
- [9] S. C. Wu, V. R. Driver, J. S. Wrobel, and D. G. Armstrong, "Foot ulcers in the diabetic patient, prevention and treatment," *Vascular Health and Risk Management*, vol. 3, no. 1, pp. 65-76, 2007.
- [10] Healthwise Staff, "Locations of Foot Ulcers," 2015.
- [11] D. Quan, "Diabetic Neuropathy: Practice Essentials," 2016.
- [12] National Institute of Diabetes and Digestive and Kidney Diseases, "Nerve Damage (Diabetic Neuropathies) | NIDDK," 2013.
- [13] B. Goldstein and J. Sanders, "Skin response to repetitive mechanical stress: A new experimental model in pig," *Archives of Physical Medicine and Rehabilitation*, vol. 79, no. 3, pp. 265-272, 1998/03/01 1998.
- [14] H. J. Murray, M. J. Young, S. Hollis, and A. J. M. Boulton, "The association between callus formation, high pressures and neuropathy in diabetic foot ulceration," *Diabetic Medicine*, vol. 13, no. 11, pp. 979-982, 1996.
- [15] B. Wisse, "Diabetes - foot ulcers," 2014.
- [16] University of Texas-Pan American. (2008, October 7). *Phases of the Normal Gait Cycle*. Available: http://faculty.utpa.edu/rafree/res/biomechanics/Virtual%20Biomechanics%20Laboratory/phases_of_the_normal_gait_cycle.html
- [17] OptoGait. (2012). *Microgate OptoGait - Gait Analysis*. Available: <http://www.optogait.com/Applications/Gait-Analysis>
- [18] Footmaxx. (2016, October 7). *Basic Biomechanics*. Available: <http://www.footmaxx.com/pathologies-sp-484/basic-biomechanics/basic-biomechanics>
- [19] Footmaxx. (2016, October 7). *Pathologies*. Available: <http://www.footmaxx.com/pathologies-sp-484?start=28>
- [20] W. Tao, T. Liu, R. Zheng, and H. Feng, "Gait Analysis Using Wearable Sensors," *Sensors (Basel, Switzerland)*, vol. 12, no. 2, pp. 2255-2283, 2012.
- [21] MiniSun. (2016, October 7). *Intelligent Device for Energy Expenditure and Activity*. Available:

- <http://www.minisun.com/ideea.htm>
- [22] A. H. Mackey, N. S. Stott, and S. E. Walt, "Reliability and Validity of an Activity Monitor (IDEEA) in the Determination of Temporal-Spatial Gait Parameters in Individuals With Cerebral Palsy," *Gait & Posture* 28(4), vol. 28, no. 4, pp. 634-639, 2008.
- [23] MiniSun. (2017). *MiniSun_IDEEA*. Available: <http://www.minisun.com/LifeGait.htm>
- [24] Tekscan Inc. (2016, October 7). *F-Scan System*. Available: <https://www.tekscan.com/products-solutions/systems/f-scan-system>
- [25] J. Woodburn and P. S. Helliwell, "Observations on the F-Scan in-shoe pressure measuring system," *Clinical Biomechanics*, vol. 11, no. 5, pp. 301-304, 1996/07/01 1996.
- [26] Tekscan Inc, "Footwear Research and Development with F-Scan," 2013-12-10 2013.
- [27] Novel USA. (2012). *Pedar, Mobile Pedography*. Available: http://www.novelusa.com/assets/pdf/pedar/pedar_mobile-pedography_web.pdf
- [28] ResearchGate. (2017, April 24). *Can anyone recommend some pressure insoles for gait analysis in adults?* Available: https://www.researchgate.net/post/Can_anyone_recommend_some_pressure_insoles_for_gait_analysis_in_adults
- [29] K. Kong and M. Tomizuka, "A Gait Monitoring System Based on Air Pressure Sensors Embedded in a Shoe," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 3, pp. 358-370, 2009.
- [30] A. L. McDonough, M. Batavia, F. C. Chen, S. Kwon, and J. Ziai, "The validity and reliability of the GAITRite system's measurements: A preliminary evaluation," *Archives of Physical Medicine and Rehabilitation*, vol. 82, no. 3, pp. 419-425, 3// 2001.
- [31] GAITRite. (2016, October 7). *GAITRite Systems - The One and Only*. Available: <http://www.gaitrite.com/>
- [32] R. Wagner and A. Ganz, "PAGAS: Portable and accurate gait analysis system," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 280-283.
- [33] R. I. Clinic. (2016, October 7). *3D Gait Analysis*. Available: <http://3dgaitanalysis.com/>
- [34] S. J. M. Bamberg, A. Y. Benbasat, D. M. Scarborough, D. E. Krebs, and J. A. Paradiso, "Gait Analysis Using a Shoe-Integrated Wireless Sensor System," *IEEE Transactions on Information Technology in Biomedicine*, vol. 12, no. 4, pp. 413-423, 2008.
- [35] M. Yavuz, "Plantar shear stress distributions in diabetic patients with and without neuropathy," *Clinical biomechanics (Bristol, Avon)*, vol. 29, no. 2, pp. 223-229, 11/15 2014.
- [36] M. E. Fernando *et al.*, "Plantar Pressure in Diabetic Peripheral Neuropathy Patients with Active Foot Ulceration, Previous Ulceration and No History of Ulceration: A Meta-Analysis of Observational Studies," (in eng), *PLoS One*, vol. 9, no. 6, 2014.
- [37] OptoForce. (2016, October 11). *3D Force Sensor (OMD)*. Available: <http://optoforce.com/3dsensor/>
- [38] RobotShop. (2016, October 11). *100N 3D Force Sensor*. Available: <http://www.robotshop.com/en/100n-3d-force-sensor.html>
- [39] S. Dokos, I. J. LeGrice, B. H. Smaill, J. Kar, and A. A. Young, "A Triaxial-Measurement Shear-Test Device for Soft Biological Tissues," *Journal of Biomechanical Engineering*, vol. 122, no. 5, pp. 471-478, 2000.
- [40] Konux Inc. (2016, October 12). *Sensors with Strain Gauge | KONUX*. Available: <https://www.konux.com/sensors-with-strain-gauge/>
- [41] !!! INVALID CITATION !!! {}.
- [42] M. P. Clarke, "An in-shoe gait analysis device to measure the maximum shear stresses at the first metatarsal head," (in eng), MSc 1996 1996.
- [43] J. G. Johnson and S. S. Ozkan, "Three Dimensional Force Sensing Array with Applications in Robotics and Biomechanics," 2016, Available: <https://web.wpi.edu/Pubs/E-project/Available/E-project-042816-121258/>.

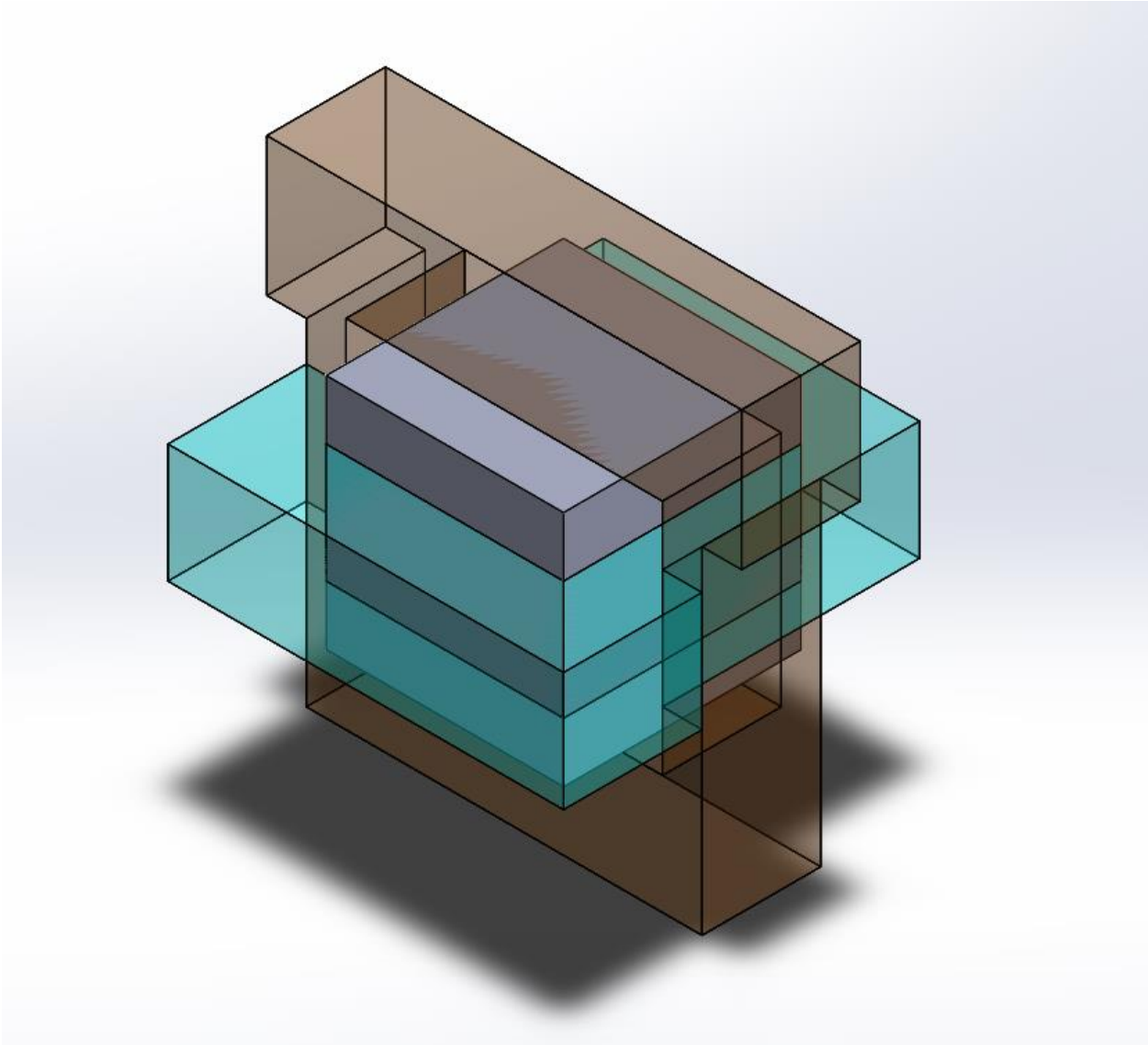
- [44] Melexis, "MLX90363 Triaxis Magnetometer IC Data sheet," ed, 2013.
- [45] Centers for Disease Control and Prevention, "FastStats - Body Measurements," 2016.
- [46] Rush University Medical Center, "Ideal Height and Weight Chart," 2016.
- [47] E. Giles, "Height Estimation from Foot and Shoeprint Length," *Journal of forensic sciences*, vol. 36, no. 4, p. 13129J, 07 1991.
- [48] S. J. Lee and J. Hidler, "Biomechanics of overground vs. treadmill walking in healthy individuals," *Journal of Applied Physiology*, 10.1152/jappphysiol.01380.2006 vol. 104, no. 3, p. 747, 2008.
- [49] B. R. Umberger and P. E. Martin, "Mechanical power and efficiency of level walking with different stride rates," *Journal of Experimental Biology*, 10.1242/jeb.000950 vol. 210, no. 18, p. 3255, 2007.
- [50] Center for Devices and Radiological Health, "Consumers (Medical Devices) - Learn if a Medical Device Has Been Cleared by FDA for Marketing," (in en), WebContent 2014.
- [51] Center for Devices and Radiological Health, "Use of International Standard ISO 10993-1, "Biological evaluation of medical devices - Part 1: Evaluation and testing within a risk management process", " Available:
<https://www.fda.gov/downloads/medicaldevices/deviceregulationandguidance/guidancedocuments/ucm348890.pdf>
- [52] ASTM, "Standard Practice for Testing Guinea Pigs for Contact Allergens: Guinea Pig Maximization Test,"

ID	Task Name	Jan 2017																											
		31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	Troubleshoot Modular Circuit Board																												
2	Contact Anany																												
3	Re-design Circuit if Necessary																												
4	Order New Circuit Board																												
5	Assemble New Circuit Board																												
6	Fix COMSOL Model																												
7	Force Testing Initial Circuit Design																												
8	Mold Silicone for First Circuit (Ecoflex)																												
9	Laser Cut Acrylic Plates																												
10	Data Collection on Load Cell																												
11	Force Testing Second Design (Dragonskin + Ecoflex)																												
12	Mold Second Circuit																												
13	Data Collection on Load Cell																												
14	Force Testing Third Circuit Design (Magnet Protrusions)																												
15	Laser Cut Acrylic Protrusions																												
16	Mold Third Circuit																												
17	Data Collection on Load Cell																												
18	Arrange Modules in Shoe-Pad																												
19	Silicone Mold Shoe-pad/Modules																												
20	Update code to accommodate multiple modules																												
21	Test/Calibrate Shoe-Pad Device																												
22	Analyze Data/Finish Paper																												

ID	Task Name	Feb 2017																												Mar 2017								
		30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4			
1	Troubleshoot Modular Circuit Board																																					
2	Contact Anany																																					
3	Re-design Circuit if Necessary																																					
4	Order New Circuit Board																																					
5	Assemble New Circuit Board																																					
6	Fix COMSOL Model																																					
7	Force Testing Initial Circuit Design																																					
8	Mold Silicone for First Circuit (Ecoflex)																																					
9	Laser Cut Acrylic Plates																																					
10	Data Collection on Load Cell																																					
11	Force Testing Second Design (Dragonskin + Ecoflex)																																					
12	Mold Second Circuit																																					
13	Data Collection on Load Cell																																					
14	Force Testing Third Circuit Design (Magnet Protrusions)																																					
15	Laser Cut Acrylic Protrusions																																					
16	Mold Third Circuit																																					
17	Data Collection on Load Cell																																					
18	Arrange Modules in Shoe-Pad																																					
19	Silicone Mold Shoe-pad/Modules																																					
20	Update code to accommodate multiple modules																																					
21	Test/Calibrate Shoe-Pad Device																																					
22	Analyze Data/Finish Paper																																					

Appendix B: Magnet Protrusion CAD Model and Drawings

Isometric View



Protrusion Explosion Drawing

ITEM NO.	PART NUMBER	MATERIAL	DESCRIPTION	QTY.
1	guard	ACRYLIC	PROTRUSION	1
2	Magnet	MAGNET	MAGNET	1
3	guard2	ACRYLIC	PROTRUSION	1

PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF WPI. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF WPI IS PROHIBITED.

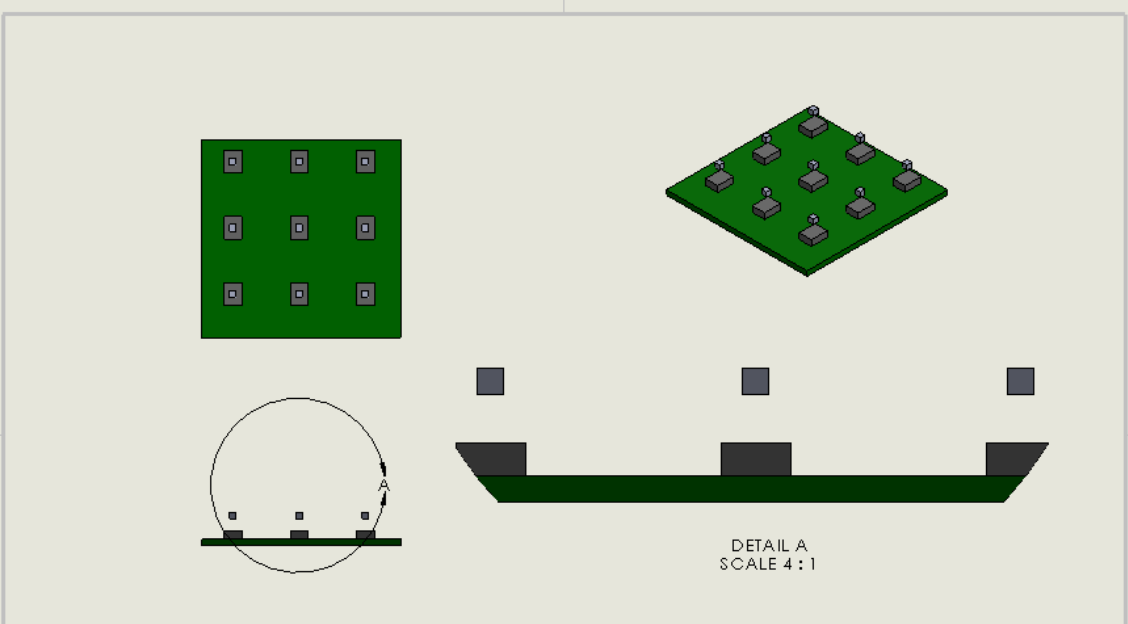
UNLESS OTHERWISE SPECIFIED		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS DRAWN		TDL	10/12/2016
TOLERANCES:		CHECKED	
ANGLES: ± 0.1°		ENG APPR.	
ONE PLACE DECIMAL ±0.1		MFG APPR.	
TWO PLACE DECIMAL ±0.12		QA	
THREE PLACE DECIMAL ±0.25		COMMENTS:	
INTERPRETATION TOLERANCES PER:			
MATERIAL:			
FINISH:			
NEXT ASSEMBLY	USED ON	SCALE: 10:1 WEIGHT:	
APPLICATION		SHEET 1 OF 1	

Worcester Polytechnic Institute


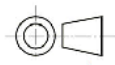
TITLE:
PROTRUSION

SIZE **A** DWG. NO. magnet and guards_ REV **2**

Appendix B: 3-by-3 Position Drawing



The drawing shows a 3x3 grid of components on a green PCB. It includes a top-down view, a perspective view, and a detail view labeled 'DETAIL A' at a scale of 4:1. The detail view shows a component with three pins. A circular callout with arrows indicates the location of the detail on the top-down view.

 <p>Worcester Polytechnic Institute</p> <p>PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.</p>			DIMENSIONS AS SPECIFIED DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ANGULAR: ± 0.1° ONE PLACE DECIMAL: ±0.1 TWO PLACE DECIMAL: ±0.12 THREE PLACE DECIMAL: ±0.123		NAME: TDL DATE: 10/12/2016	TITLE: 3X3 CIRCUIT PLACEMENT SIZE: A DWG. NO.: 3x3 circuit REV: 1 SCALE: 1:1 WEIGHT: SHEET 1 OF 1
			INTERFERING GEOMETRIC TOLERANCING PER: MATERIAL: FINISH: APPLICATION: DO NOT SCALE DRAWING	CHECKED: ENG. APPR.: MFG. APPR.: Q.A.: COMMENTS:		
	NEXT ASSY: USED ON:					

Appendix C: Silicone Molding Instructions

The silicone molding in this procedure is done in the WPI Soft Robotics Lab in Higgins Labs. Silicone is very sticky, so gloves are preferable. However, the silicone is not harmful to skin, so if you do get some on you it is not the end of the world.

1. A rigid mold (probably 3D printed) is needed for setting and baking the silicone. Make sure to have this beforehand. Make sure that the sensor is secured into the mold.
2. Find a cup or container in which to mix the silicone, along with the proper bottles of silicone mix (the yellow and blue bottles of Ecoflex 0030 for this example). Find the scale for weighing the silicone mixture.



FIGURE A: THE ECOFLEX 0030 MIX

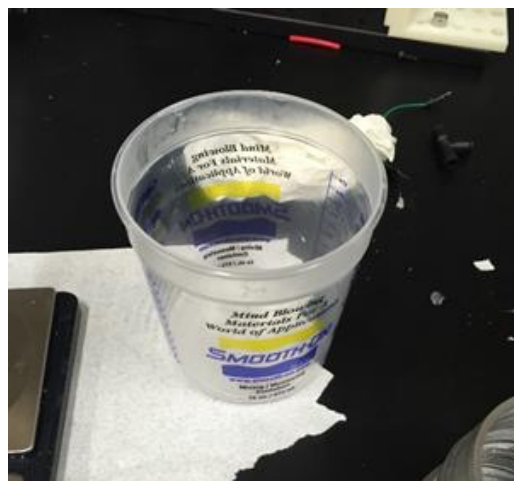


FIGURE B: EXAMPLE OF MIXING CUP

3. Place the cup on the scale and tare it. Mix equal parts of the blue and yellow components of

the Ecoflex 0030 silicone in the cup. (A different ratio is required for different types of silicone, like Dragon skin). For purposes of a single magnet and sensor, **5g of each** should be enough. Always mix a little more than is actually needed to account for air bubbles.

4. Once weighed, mix the two parts together using a mixing stick. Mix thoroughly to remove as many air bubbles as possible.
5. After mixing, place the container of silicone in the vacuum chamber on the lab bench. Secure the top and make sure the valves are opened into the chamber. Turn on the vacuum switch on the wall. For this amount of Ecoflex, leave in the chamber for **2 to 3 minutes**.
6. When the mixture is done in the vacuum chamber, use the second valve on the cap to equalize the air pressure before removing the cap. To prepare for setting the silicone, place down paper towels where you will be working! The others in the lab will give you a hard time if you make a mess. Silicone is very messy.
7. Before moving on **make sure that there are no air bubbles in the silicone!** Any air bubbles in the mold will ruin the integrity of the silicone once it is baked.
8. Find a clamp, and clamp down the mold/sensor assembly. Place on top of the paper towels. Carefully pour the silicone into the mold. Be patient and wait for the silicone to settle. As you pour, air bubbles will rise to the top, so **make sure to pour more than is necessary – the only way to remove air bubbles is through volumetric displacement.** Don't worry if the silicone overflows – it is easy to cut off excess afterward

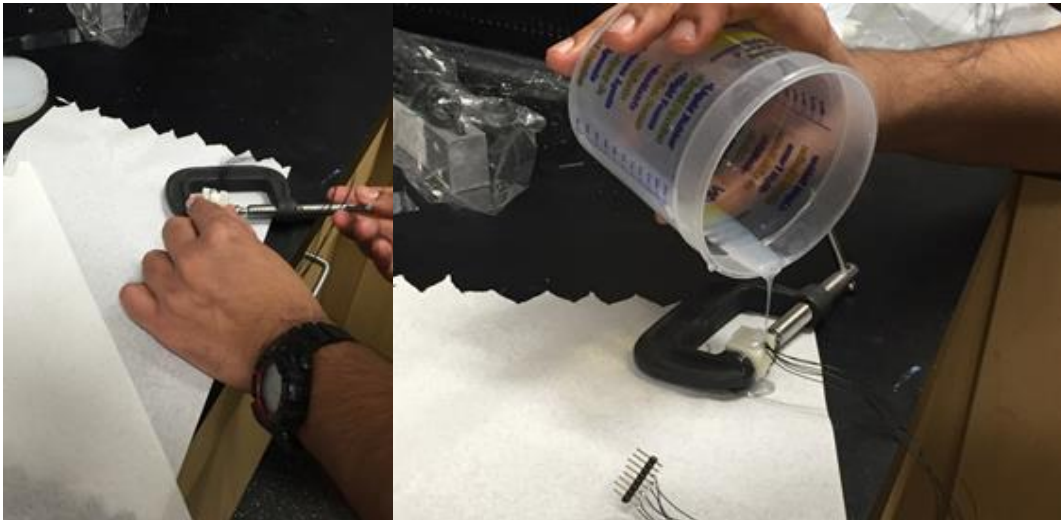



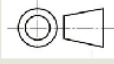
FIGURE C: CLAMPING THE MOLD AND POURING THE SILICONE

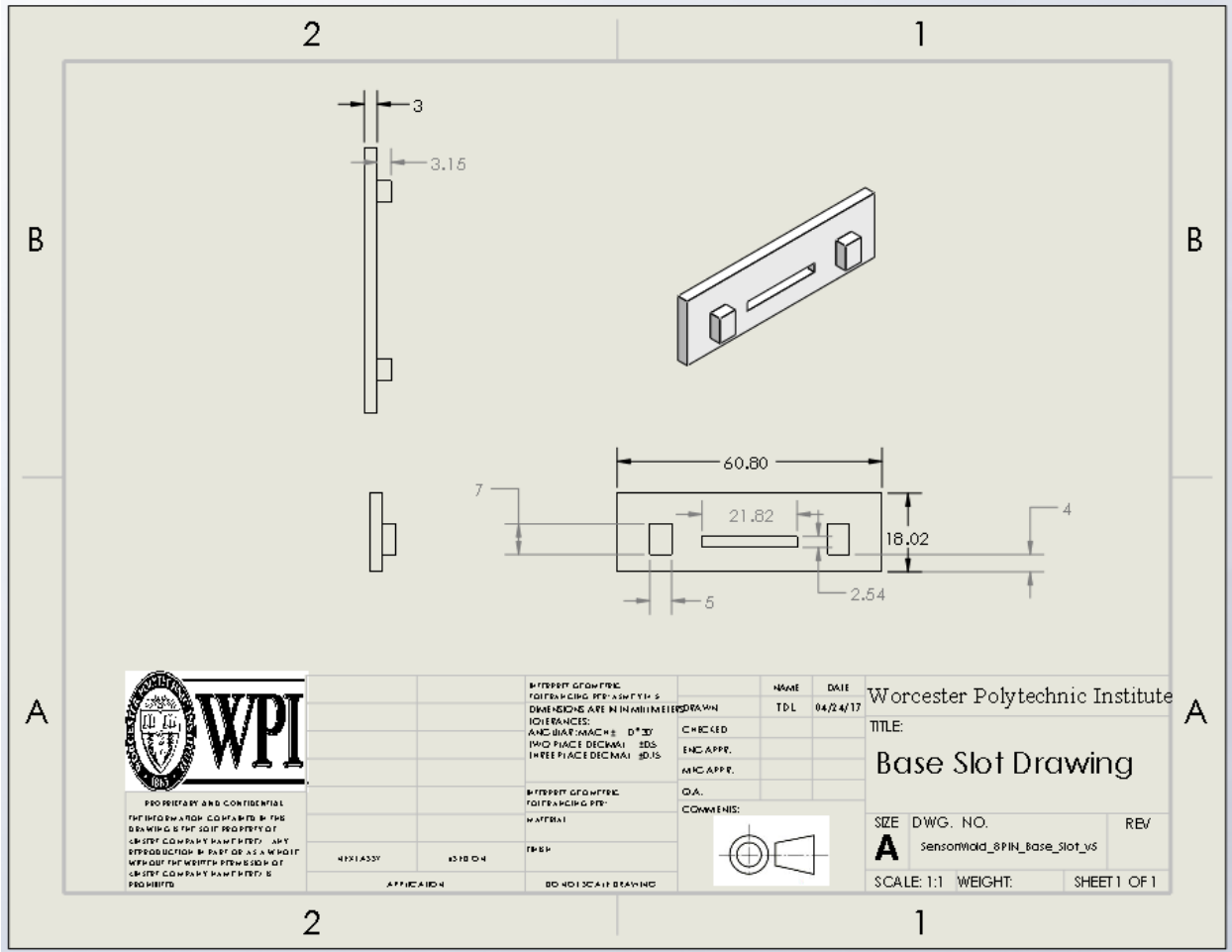
9. Place the mold into the oven on the bench. The oven technically has three stages to set. For

this silicone, only the first stage is needed. **Set the temperature for the first stage at 60 C,** and the other two stages to 0C (off). **Set the rate of heating to 40 C/minute. Bake the silicone in the oven for 20 minutes.**

10. After the time is up, remove the mold from the oven, and break the mold off, revealing the final silicone. Trim off excess silicone as necessary.

Appendix D: Final Mold Design

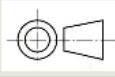
 <p>PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF [REDACTED] COMPANY NAME HERE. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF [REDACTED] COMPANY NAME HERE IS PROHIBITED.</p>			INTERPRETIVE TOLERANCING PER: ASME Y14.5 DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED TOLERANCES: ANGULAR: MAX CHG: 0°30' TWO PLACE DECIMAL: ±0.5 THREE PLACE DECIMAL: ±0.15	NAME: TDL DATE: 04/24/17	TITLE: Worcester Polytechnic Institute Mold Assembly
	NEXT ASSY: APPLICATION:	USED ON: DO NOT SCALE DRAWING	INTERPRETIVE TOLERANCING PER: MATERIAL: FINISH:	CHECKED: ENG APPR.: MFG APPR.: Q.A.: COMMENTS: 	SIZE: A DWG. NO.: Mold Assembly SCALE: 1:1 WEIGHT:

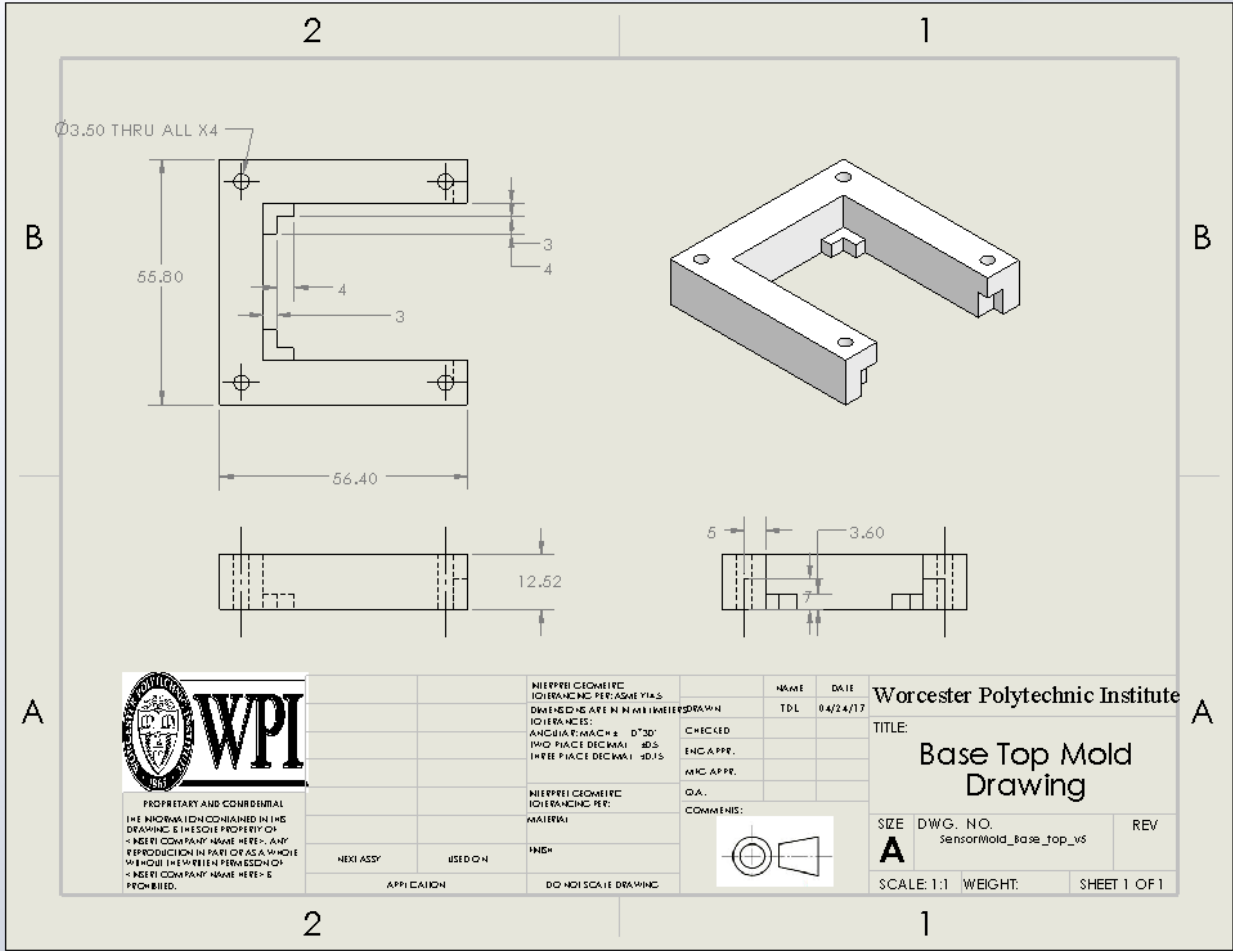


PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS
 DRAWING IS THE SOLE PROPERTY OF
 CREDIT COMPANY BANK FIDELITY. ANY
 REPRODUCTION IN WHOLE OR IN PART
 WITHOUT THE WRITTEN PERMISSION OF
 CREDIT COMPANY BANK FIDELITY IS
 PROHIBITED.

MATERIAL T304	FINISH 160	QUANTITY 1000	DATE 04/24/17	NAME TDL	DRAWN TDL
APPLICATION DO NOT SCALE DRAWING			CHECKED ENC APPR. ENC APPR. Q.A. COMMENTS:		

Worcester Polytechnic Institute	
TITLE: Base Slot Drawing	
SIZE A	DWG. NO. Sensorwld_8PIN_base_slot_v5
SCALE: 1:1 WEIGHT: SHEET 1 OF 1	

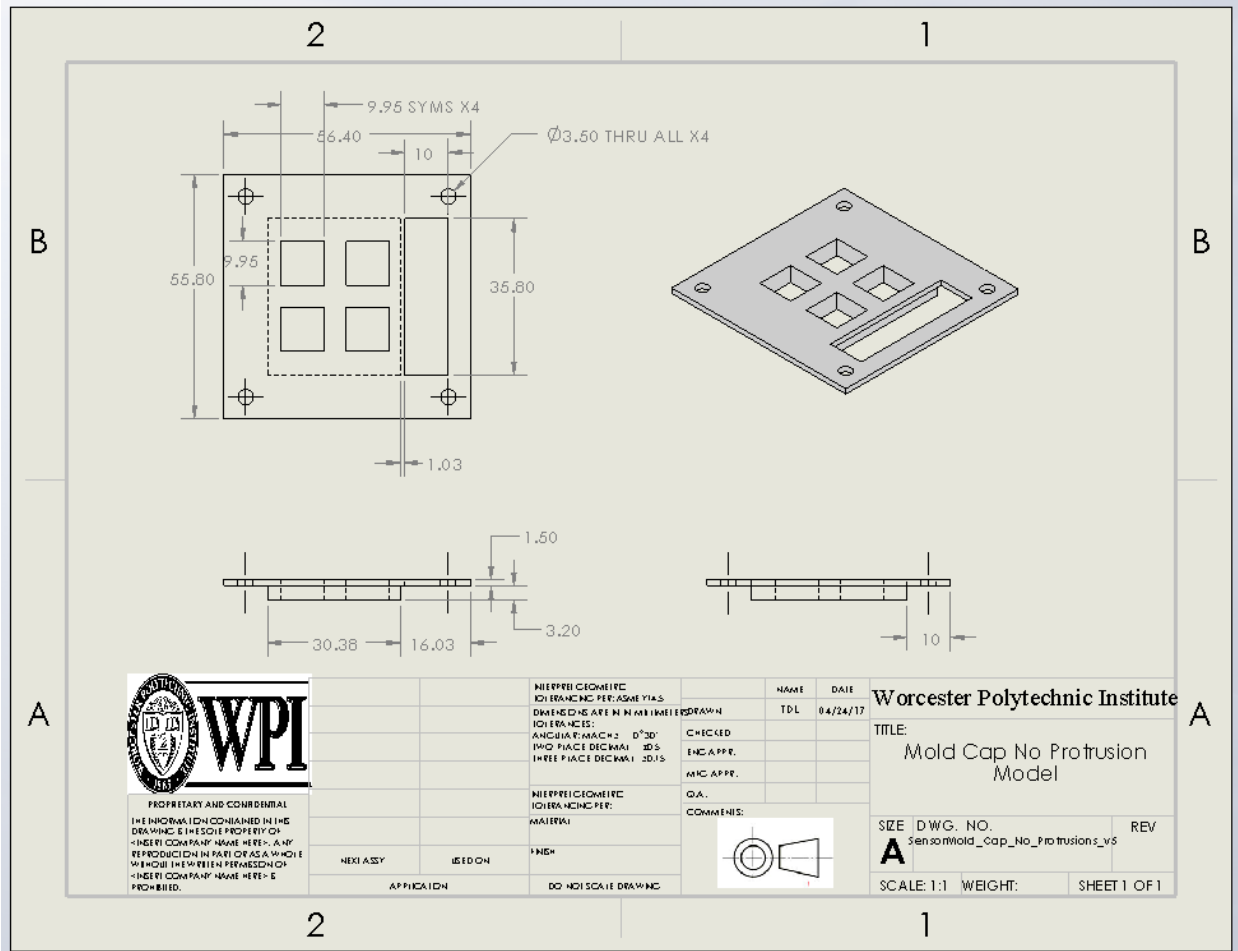




PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS
 DRAWING IS THE SOLE PROPERTY OF
 WPI. NO PART OF THIS DRAWING IS TO
 BE REPRODUCED OR TRANSMITTED IN
 ANY FORM OR BY ANY MEANS, ELECTRONIC
 OR MECHANICAL, INCLUDING PHOTOCOPYING,
 RECORDING, OR BY ANY INFORMATION
 STORAGE AND RETRIEVAL SYSTEM, WITHOUT
 THE WRITTEN PERMISSION OF
 WPI. NO PART OF THIS DRAWING IS TO
 BE REPRODUCED OR TRANSMITTED IN
 ANY FORM OR BY ANY MEANS, ELECTRONIC
 OR MECHANICAL, INCLUDING PHOTOCOPYING,
 RECORDING, OR BY ANY INFORMATION
 STORAGE AND RETRIEVAL SYSTEM, WITHOUT
 THE WRITTEN PERMISSION OF
 WPI.

NIERPRE GEOMETRIC TOLERANCING PER: ASME Y14.5		NAME	DATE
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		TDL	04/24/17
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		CHECKED	
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		ENCL APPR.	
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		MFG APPR.	
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		D.A.	
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED		COMMENTS:	
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED			
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED			
NEXT ASSY	USED ON		
APPLICATION		DO NOT SCALE DRAWING	

Worcester Polytechnic Institute		
TITLE: Base Top Mold Drawing		
SIZE	DWG. NO.	REV
A	SensorMold_base_top_vs	
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

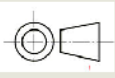


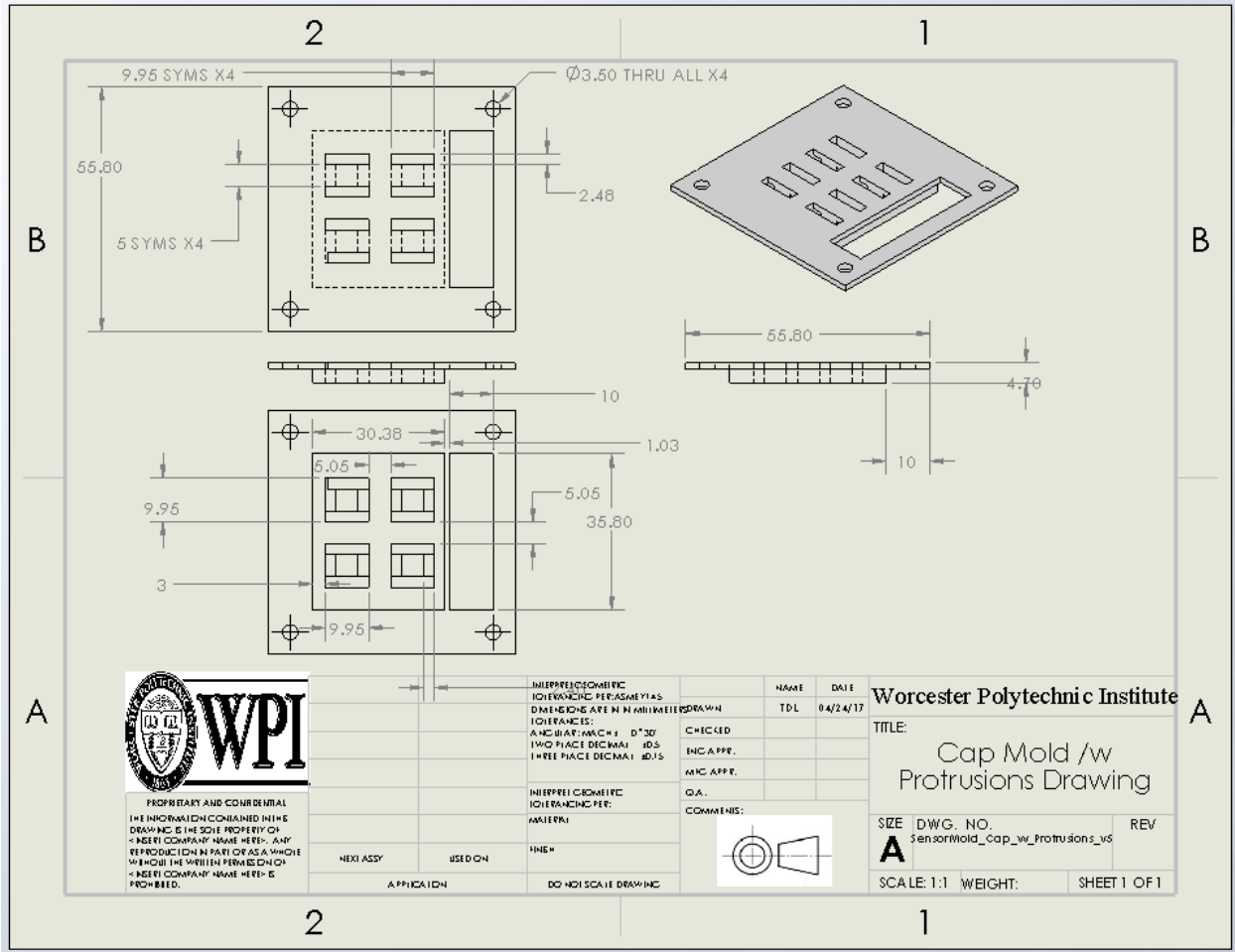
PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS
 DRAWING IS THE SOLE PROPERTY OF
 <INSERT COMPANY NAME HERE>. ANY
 REPRODUCTION IN PART OR AS A WHOLE
 WITHOUT THE WRITTEN PERMISSION OF
 <INSERT COMPANY NAME HERE> IS
 PROHIBITED.

NEXT ASSY APPLICATION	USED ON DO NOT SCALE DRAWING	INTERFERE GEOMETRIC DIMENSIONS PER: ASME Y14.5 DIMENSIONS ARE IN MILLIMETERS UNLESS INDICATED ANGLE: 1/4" = 1" 0°/30° TWO PLACE DECIMAL S.D.S. THREE PLACE DECIMAL S.D.S.	NAME TDL	DATE 04/24/17
		INTERFERE GEOMETRIC DIMENSIONS PER: MATERIAL:	DRAWN CHECKED ENCL APPR. MFC APPR. O.A. COMMENTS:	

NAME TDL	DATE 04/24/17
DRAWN CHECKED ENCL APPR. MFC APPR. O.A. COMMENTS:	

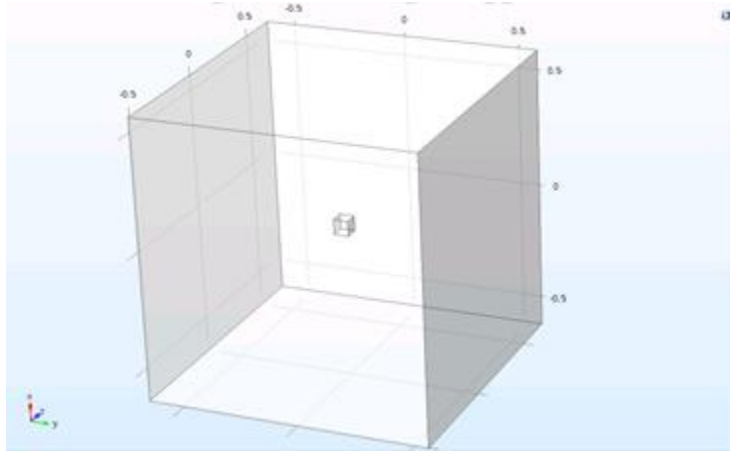
Worcester Polytechnic Institute
 TITLE:
 Mold Cap No Protrusion
 Model
 SIZE DWG. NO. REV
A SensorMold_Cap_No_Protrusions_v5
 SCALE: 1:1 WEIGHT: SHEET 1 OF 1



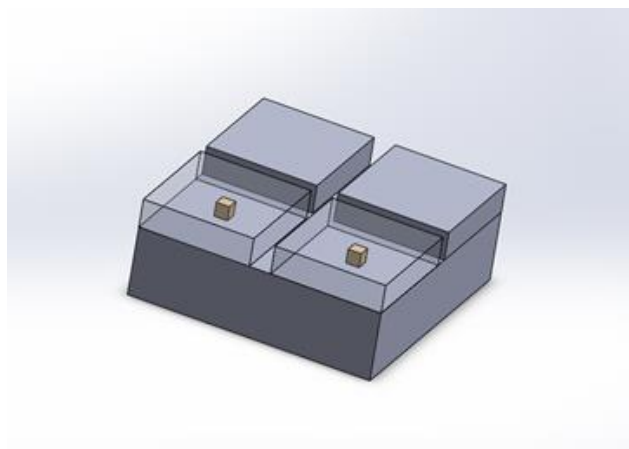
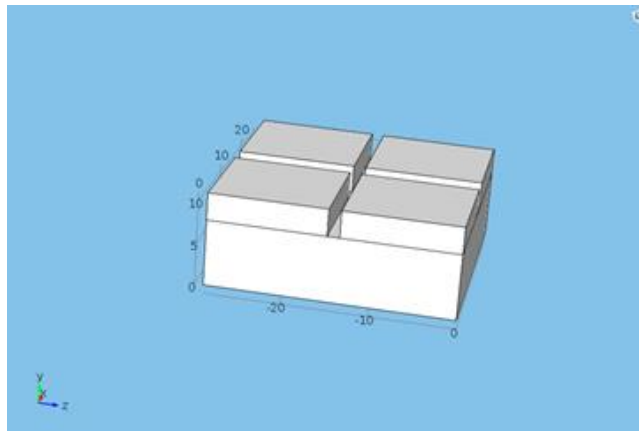


Appendix E: COMSOL Iterations

Iteration 1: Simple 1-Magnet COMSOL Model



Iteration 2: Complicated 2-by-2 COMSOL Model



Appendix F: Current COMSOL Model Parameters

Global Parameters

Parameters			
Name	Expression	Value	Description
Mu1	5.6E5[Pa]	5.6E5 Pa	Og-strain 1
Mu2	1562[Pa]	1562 Pa	Og-strain 2
Mu3	-5.2E5[Pa]	-5.2E5 Pa	Og-strain 3
Alp1	1.26	1.26	Og-par1
Alp2	5.1	5.1	Og-par2
Alp3	0.64	0.64	Og-par3
k	15E9[Pa]	1.5E10 Pa	Bulk Modulus Silicone
load	0[N]	0 N	
flux	1.48[T]	1.48 T	
theta	pi/3	1.0472	

Neodymium Parameters

Property	Name	Value	Unit	Property group
<input checked="" type="checkbox"/> Relative permeability	mur	1.05	1	Basic
<input checked="" type="checkbox"/> Density	rho	rho(T[1/...]	kg/m ³	Basic
<input checked="" type="checkbox"/> Poisson's ratio	nu	0.24	1	Basic
<input checked="" type="checkbox"/> Young's modulus	E	1.6*10^...	Pa	Basic

Silicone Parameters

Property	Name	Value	Unit	Property group
<input checked="" type="checkbox"/> Density	rho	rho(T[1/...]	kg/m ³	Basic
<input checked="" type="checkbox"/> Relative permeability	mur	1	1	Basic
dL	dL	(dL(T[1/...]		Basic
CTE	CTE	CTE(T[1/...]	1/K	Basic
Thermal conductivity	k	k(T[1/K])...	W/(m·K)	Basic
Coefficient of thermal expansion	alpha	(alpha(T[...]	1/K	Basic
Heat capacity at constant pressu...	Cp	C(T[1/K]...	J/(kg·K)	Basic
TD	TD	TD(T[1/K]...	m ² /s	Basic
Young's modulus	E	125E6[Pa]	Pa	Young's mod
Poisson's ratio	nu	nu(T[1/K])	1	Young's mod
Bulk modulus	K	1.5E9[Pa]	N/m ²	Bulk modulus
Shear modulus	G	mu(T[1/K]...	N/m ²	Bulk modulus

Hyperelastic Model: Ogden Model

(Parameters refer to global parameters)

▼ Hyperelastic Material

Material model:

Ogden ▼

Nearly incompressible material

Ogden parameters

"	p	Shear modulus (Pa)	Alpha parameter
1		Mu1	Alp1
2		Mu2	Alp2
3		Mu3	Alp3

Appendix H: Foot Pressure Calculations

Variables/Helpful Information:

- Average US male height:
 - 5'9" - 5'10" [45]
- US shoe size for male height 5'9" - 5'10":
 - Size 9 - 10 [47]
 - Because a Size 10 shoe is readily available to us, this is what we chose
 - Approximate Area under US men's size 10 foot: $523.75\text{cm}^2 = 0.052375\text{m}^2$
- People with diabetic neuropathy are likely to be overweight/obese
 - As a design constraint, will cut off weight measurements just before obese classification
 - Max weight for this height before considered 'obese' = ~200lbs ~90kg ~ 883N [46]
- Max shear force = ~15% of body weight (normal, anterior-posterior)
- Safety factor = ~25% of body weight

Calculations:

Average Max Pressure on a foot (during stance phase):

$$\begin{aligned} \text{Average Max Pressure} &= \text{Max Weight} \div \text{Aea of Size 10 foot} \\ &= 883\text{N} \div 0.052375\text{m}^2 = 16.86\text{kPa} \end{aligned}$$

Size of one circuit module:

$$\begin{aligned} \text{Area} &= \text{Length} * \text{Width} \\ &= 29.8\text{mm} * 39\text{mm} = 0.0011622\text{m}^2 \end{aligned}$$

Max normal force to apply to one module during testing:

$$\begin{aligned} \text{Max Normal Forcee} &= \text{Average Max Pressure duing Stance Phase} * \text{Area of a Module} \\ &= 16859\text{Pa} * 0.0011622\text{m}^2 = 19.59\text{N} \end{aligned}$$

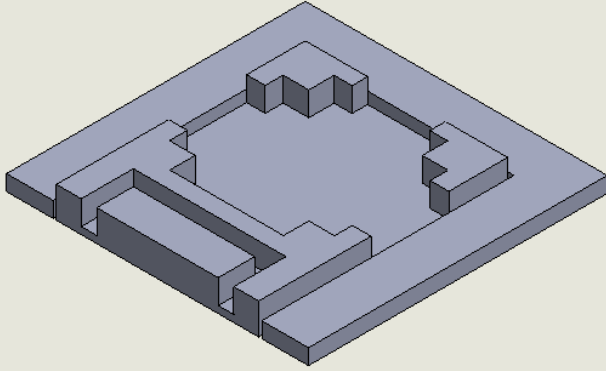
Average Shear Pressure (assuming 25% of sole is covered):



$$\begin{aligned} \text{Average Shear Force} &= \text{Safety Factor} * \text{Body Weight} \\ &= 0.25 * 883\text{N} = 220.75\text{N} \\ \text{Average Shear Pressure} &= \text{Average Shear Force} \div 25\% \text{ Area of a Size 10 Show} \\ &= 220.75\text{N} \div (0.25 * 0.052375\text{m}^2) = 16.86\text{kPa} \end{aligned}$$

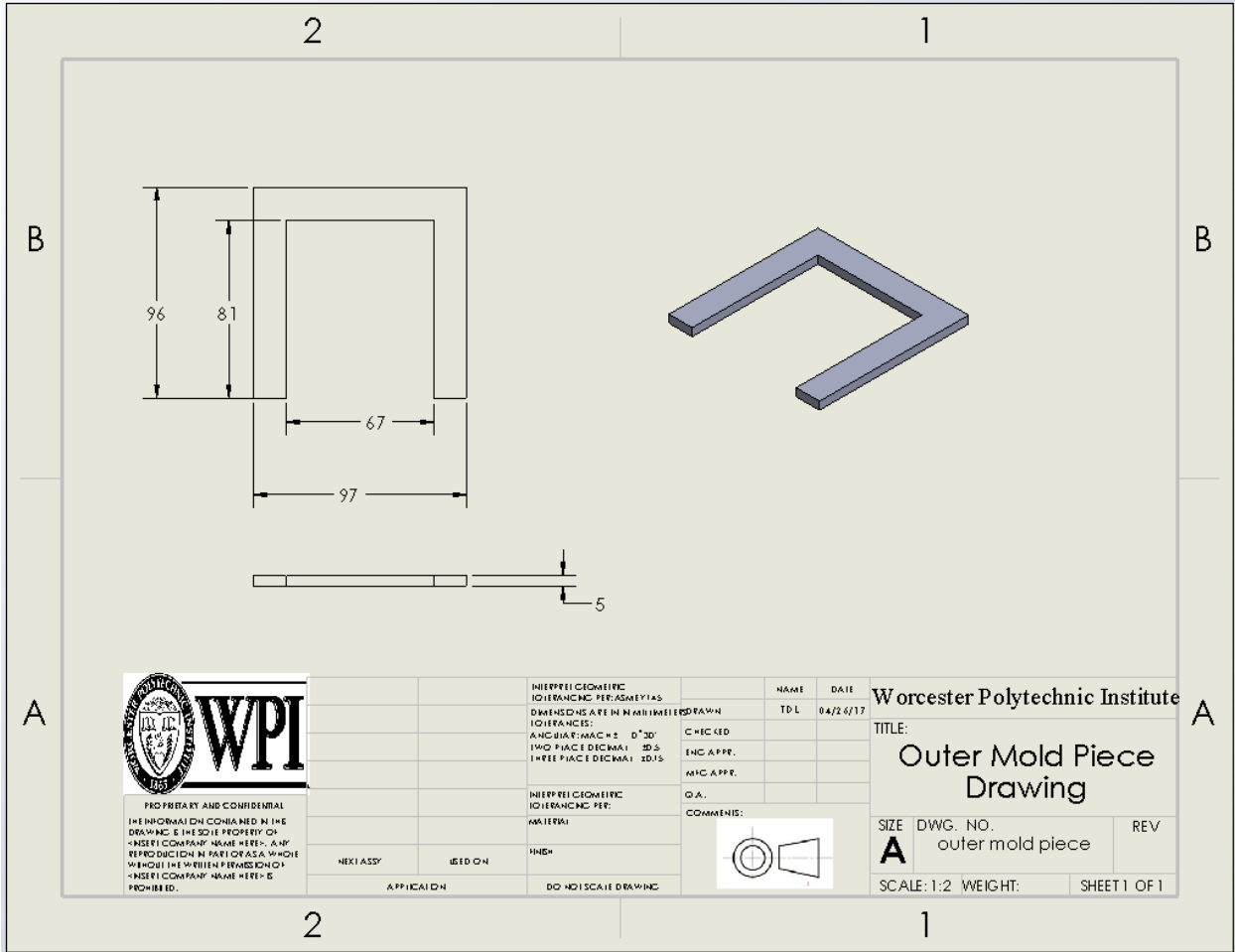
Max shear force to apply to one module:

$$\begin{aligned} \text{Max Shear Force} &= \text{Average Sear Pressure (with 25\% of sole covered)} * \text{Area of a Module} \\ &= 16859\text{Pa} * 0.0011622\text{m}^2 = 19.59\text{N} \end{aligned}$$

Appendix I: Shear Force Apparatus



 <p>PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.</p>			INTERPRET GEOMETRIC TOLERANCING PER: ASME Y14.5 DIMENSIONS ARE IN MILLIMETERS TOLERANCES: ANGULAR: $\pm 0.10^\circ$ TWO PLACE DECIMAL: ± 0.05 THREE PLACE DECIMAL: ± 0.025	NAME: TD L DATE: 04/24/17	Worcester Polytechnic Institute TITLE: Shear Force Testing Apparatus
	NEXT ASSY: APPLICATION:	USED ON: APPLICATION:	MATERIAL: PART NUMBER: DO NOT SCALE DRAWING	CHECKED: ENG. APPR.: MFG. APPR.: Q.A.: COMMENTS: 	SIZE: A DWG. NO.: Shear Force Testing Apparatus SCALE: 1:1 WEIGHT:



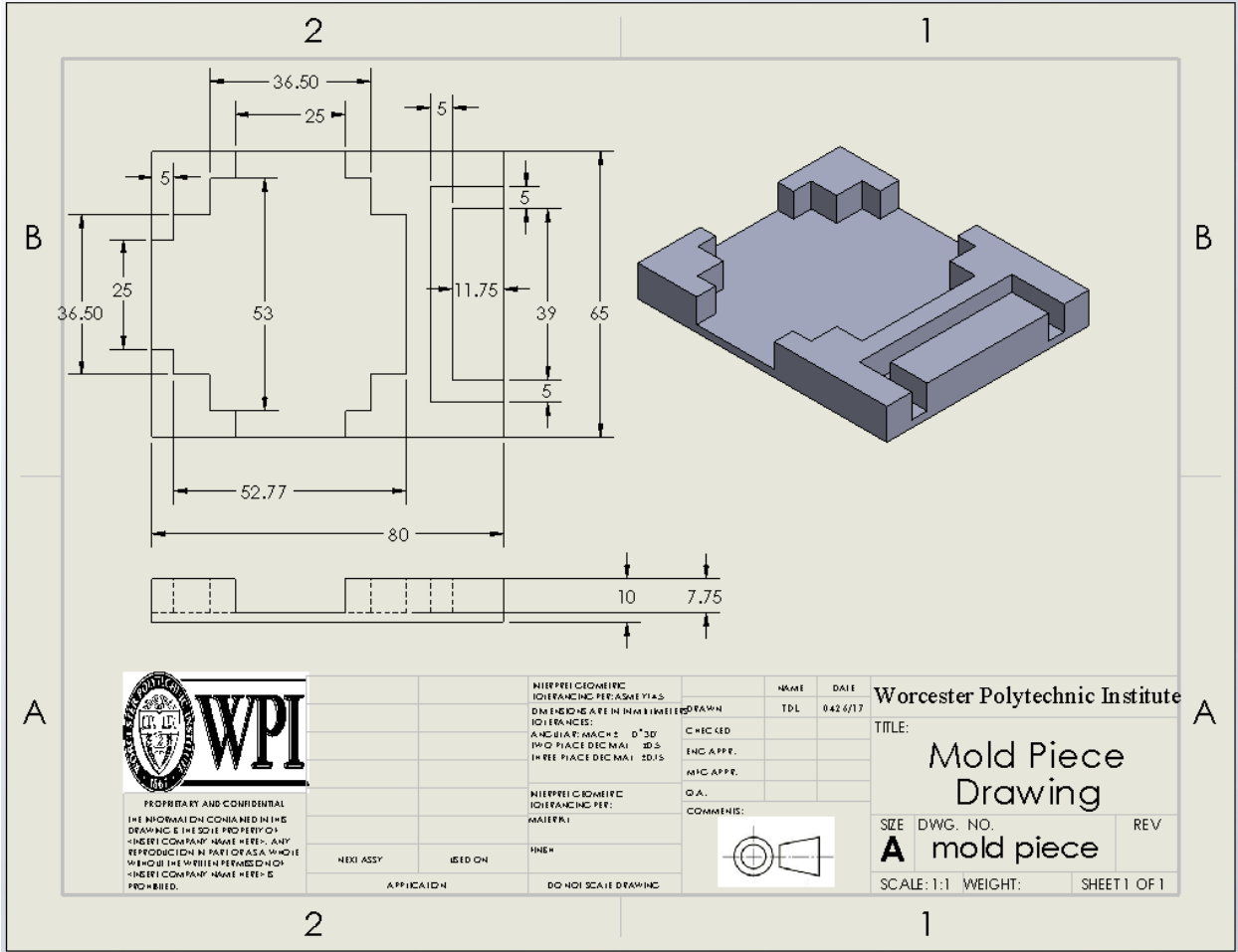
PROPRIETARY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THE
 DRAWING IS THE SOLE PROPERTY OF
 <INSERT COMPANY NAME HERE>. ANY
 REPRODUCTION IN PART OR AS A WHOLE
 WITHOUT THE WRITTEN PERMISSION OF
 <INSERT COMPANY NAME HERE> IS
 PROHIBITED.

INTERPRET GEOMETRIC TOLERANCES PER ASME Y14.5	NAME	DATE
DIMENSIONS ARE IN MILLIMETERS UNLESS OTHERWISE SPECIFIED	TD L	04/26/17
TOLERANCES:	CHECKED	
ANGLES: MAX ± 0°30'	ENG APPR.	
ANGLES: DECIMAL ±0.5	MHC APPR.	
ANGLES: DECIMAL ±0.5	D.A.	
INTERPRET GEOMETRIC TOLERANCES PER:	COMMENTS:	
MATERIAL:		
FINISH:		
APPLICATION		
DO NOT SCALE DRAWING		

Worcester Polytechnic Institute

TITLE:
Outer Mold Piece Drawing

SIZE	DWG. NO.	REV
A	outer mold piece	
SCALE: 1:2	WEIGHT:	SHEET 1 OF 1



Appendix J: Arduino Code for Modularity Test

```
/* THE XOR GATE MAKES HIGH ON AND LOW OFF FOR SENSORS
Code to get data from MLX90363 sensor.

This piece of code gets xyz magnetic flux magnitudes from the
sensor. It uses a function to compute CRCs.

Arduino Mega is used as the master device and a single MLX90363
sensor is used as the slave.

Written by Selim Ozel, 08.14.2015
Edited and Modified by David Laovoravit, 4.23.2017
*/

#include <SPI.h>           //Include SPI Library
#include <TimerOne.h>     //Include TimeOne Library

#define NUMBER 0

//Define and initialize CRC array, 256 bytes
char CRCArray[] = { //used to check correct data transfer
0x00, 0x2F, 0x5E, 0x71, 0xBC, 0x93, 0xE2, 0xCD, 0x57, 0x78, 0x09, 0x26,
0xEB, 0xC4, 0xB5, 0x9A, 0xAE, 0x81, 0xF0, 0xDF, 0x12, 0x3D, 0x4C, 0x63,
0xF9, 0xD6, 0xA7, 0x88, 0x45, 0x6A, 0x1B, 0x34, 0x73, 0x5C, 0x2D, 0x02,
0xCF, 0xE0, 0x91, 0xBE, 0x24, 0x0B, 0x7A, 0x55, 0x98, 0xB7, 0xC6, 0xE9,
0xDD, 0xF2, 0x83, 0xAC, 0x61, 0x4E, 0x3F, 0x10, 0x8A, 0xA5, 0xD4, 0xFB,
0x36, 0x19, 0x68, 0x47, 0xE6, 0xC9, 0xB8, 0x97, 0x5A, 0x75, 0x04, 0x2B,
0xB1, 0x9E, 0xEF, 0xC0, 0x0D, 0x22, 0x53, 0x7C, 0x48, 0x67, 0x16, 0x39,
0xF4, 0xDB, 0xAA, 0x85, 0x1F, 0x30, 0x41, 0x6E, 0xA3, 0x8C, 0xFD, 0xD2,
0x95, 0xBA, 0xCB, 0xE4, 0x29, 0x06, 0x77, 0x58, 0xC2, 0xED, 0x9C, 0xB3,
0x7E, 0x51, 0x20, 0x0F, 0x3B, 0x14, 0x65, 0x4A, 0x87, 0xA8, 0xD9, 0xF6,
0x6C, 0x43, 0x32, 0x1D, 0xD0, 0xFF, 0x8E, 0xA1, 0xE3, 0xCC, 0xBD, 0x92,
0x5F, 0x70, 0x01, 0x2E, 0xB4, 0x9B, 0xEA, 0xC5, 0x08, 0x27, 0x56, 0x79,
0x4D, 0x62, 0x13, 0x3C, 0xF1, 0xDE, 0xAF, 0x80, 0x1A, 0x35, 0x44, 0x6B,
0xA6, 0x89, 0xF8, 0xD7, 0x90, 0xBF, 0xCE, 0xE1, 0x2C, 0x03, 0x72, 0x5D,
0xC7, 0xE8, 0x99, 0xB6, 0x7B, 0x54, 0x25, 0x0A, 0x3E, 0x11, 0x60, 0x4F,
0x82, 0xAD, 0xDC, 0xF3, 0x69, 0x46, 0x37, 0x18, 0xD5, 0xFA, 0x8B, 0xA4,
0x05, 0x2A, 0x5B, 0x74, 0xB9, 0x96, 0xE7, 0xC8, 0x52, 0x7D, 0x0C, 0x23,
0xEE, 0xC1, 0xB0, 0x9F, 0xAB, 0x84, 0xF5, 0xDA, 0x17, 0x38, 0x49, 0x66,
0xFC, 0xD3, 0xA2, 0x8D, 0x40, 0x6F, 0x1E, 0x31, 0x76, 0x59, 0x28, 0x07,
0xCA, 0xE5, 0x94, 0xBB, 0x21, 0x0E, 0x7F, 0x50, 0x9D, 0xB2, 0xC3, 0xEC,
0xD8, 0xF7, 0x86, 0xA9, 0x64, 0x4B, 0x3A, 0x15, 0x8F, 0xA0, 0xD1, 0xFE,
0x33, 0x1C, 0x6D, 0x42
};

int PinSS; //Pin declaration for Arudino Mega 2500
int PinSS1 = 53; //Set Slave Select Pin at 53
int PinSS2 = 42; //check to change
int PinMOSI = 51; //Set Master Out, Slave In Pin at 51
int PinMISO = 50; //Set Master In, Slave Out Pin at 50
int PinSCK = 52; //Set Clock Pin at 52
int PinEN = 22; //Set Slave Select Pin at 8
int PinA0 = 24; //Set Slave Select Pin at 9
int PinA1 = 26; //Set Slave Select Pin at 10
int currentState = 1;

int switchNum = 1;
bool pass = true; // Buffers to read/write MLX90363
uint8_t readBuffer[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //declare Read Buffers, as an array comprising of 8 8 bit integers
0x00};
uint8_t writeBuffer[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //declare Write Buffers, as an array comprising of 8 8 bit integers
0x00};

// Bx,By,Bz variables
int16_t Bx = //declaring Bx
0;
variable as a 16 bit integer and set starting value at 0
int16_t By = //declaring By
0;
variable as a 16 bit integer and set starting value at 0
int16_t Bz = //declaring Bz
0;
variable as a 16 bit integer and set starting value at 0

// Error bits, CRC, virtual gain and rolling counter variables
uint8_t errorBits = //declaring error Bits
0;
as an 8 bit integer and set starting value at 0
uint8_t rollingCounter = //declaring Rolling Counter
0;
as an 8 bit integer and set starting value at 0
```

```

uint8_t CRC =
0; //declaring
Cyclic Redundancy Check as an 8 bit integer and set starting value at 0

// CRC function
uint8_t ComputeCRC(uint8_t Byte0, uint8_t Byte1, uint8_t Byte2, uint8_t Byte3, uint8_t Byte4, uint8_t Byte5, uint8_t Byte6) {
//Parse each byte into the CRC, pulling out of the CRC array
uint8_t CRC = 0xFF;
CRC = CRCArray[CRC ^
Byte0]; //Parsing Byte through an
XOR gate to crate a CRC
CRC = CRCArray[CRC ^ Byte1];
CRC = CRCArray[CRC ^ Byte2];
CRC = CRCArray[CRC ^ Byte3];
CRC = CRCArray[CRC ^ Byte4];
CRC = CRCArray[CRC ^ Byte5];
CRC = CRCArray[CRC ^ Byte6];
CRC =
~CRC; //Flipping
the Bits gets out the input that is wanted
return CRC;
}
int fullCircle=0;
// Sets the send message flag
int sendMessage =
0; //Set a flag for
message initialize to 0 (false)
void SendFlag() //A
helper method for setting messenger flag to true
{
sendMessage =
1; //Set a flag to
1(true)
}

void setup() //The setup before starting the looping program
{ // Mark comm pins as output or input
pinMode (PinMOSI, OUTPUT); //Set MOSI pin as an output
pinMode (PinMISO, INPUT); //Set MISO pin as an input
pinMode (PinSCK, OUTPUT); //Set Clock pin as output
pinMode (PinSS2,OUTPUT); // Make the MLX90363 sensor the active slave device
pinMode (PinSS1, OUTPUT); //Set Slave Select pin as an output
pinMode (PinEN, OUTPUT);
pinMode (PinA0, OUTPUT);
pinMode (PinA1, OUTPUT);
digitalWrite(PinSS1, LOW); //Turn off the Slave Select
digitalWrite(PinSS2,LOW);
digitalWrite(PinEN, LOW); //Turn off the Slave Select1
digitalWrite(PinA0, LOW); //Turn off the Slave Select2
digitalWrite(PinA1, LOW); //Turn off the Slave Select3
PinSS=PinSS1;

// Begin serial Comm //Set the baudrate to 9600
Serial.begin(9600);

// Required SPI configeration to communicate with MLX90363
// Details of SPI settings can be found in "Getting Started
// Guide" [GSG], under "SPI bus protocol".
SPI.begin();
//Initialize SPI and set SCK,MOSI,SS to outputs and SCK and MOSI asl low and SS as high
SPI.setBitOrder(MSBFIRST); //Set protocol to transmit Most significant bit first
SPI.setClockDivider(SPI_CLOCK_DIV32); //Set the clock to be 1/32 the frequency of the system clock
SPI.setDataMode(SPI_MODE1);
//SPI_MODE1 = Clock Polarity: 0 | Clock Phase: 1 | Output Edge: Rising | Data Capture: Falling

// Setup Timer for sending/receiving data
Timer1.initialize(500);//30000 //Initialize frequency of interrupt to 0.03 seconds
Timer1.attachInterrupt(SendFlag); //trip flag every 0.03 seconds
}
int counter = 0;
void loop()
{
if (sendMessage) { //If the timer interupt trips (0.03 seconds) collect data from sensor
//int ReadPin = 0;
switch (switchNum) { //switch case for each sensor in the module
case 1:
fullCircle=0;//if it has gone through all 4 sensors
currentState = 1;
digitalWrite(PinA0,LOW);// Sensor 00 (sensor 1)
digitalWrite(PinA1,LOW);
if (counter >= NUMBER+1) //used for adjusting how many times each sensor is run for each run through
pass=false;
switchNum++;
counter = 0;
}
else {

```

```

        counter++;
        pass=true;
    }
    break;
case 2:
    currentState = 2;
    digitalWrite(PinA0,LOW); // Sensor 01 (sensor 2)
    digitalWrite(PinA1,HIGH);
    if (counter >= NUMBER) { //used for adjusting how many times each sensor is run for each run through
        switchNum ++;
        counter = 0;
    }
    else {
        counter++;
    }
    break;
case 3:
    currentState = 3;
    digitalWrite(PinA0,HIGH); // Sensor 10 (sensor 3)
    digitalWrite(PinA1,LOW);
    if (counter >= NUMBER) { //used for adjusting how many times each sensor is run for each run through
        switchNum ++;
        counter = 0;
    }
    else {
        counter++;
    }
    break;
case 4:
    currentState = 4;
    digitalWrite(PinA0,HIGH); // Sensor 11 (sensor 4)
    digitalWrite(PinA1,HIGH);
    if (counter >= NUMBER+1) { //used for adjusting how many times each sensor is run for each run through
        switchNum = 1;
        counter = 0;
        pass=true;
        if(PinSS==PinSS1){ //used to switch between the two modules
            PinSS=PinSS2;
        }
        else if(PinSS==PinSS2){
            PinSS=PinSS1;
        }
        else{
            Serial.println("Error with Slave Toggle");
        }
    }
    else {
        pass=false;
        counter++;
    }
    fullCircle=1;
    break;
default:
    Serial.println("Error with switch case");
}

// Create a GET1 message. Format of messages are explained in both DataSheet
// [DS] and GSG.
writeBuffer[0] = 0x00;
//this is a set up step to have the sensor be a GET1 See DataSheet
writeBuffer[1] = 0x00;
writeBuffer[2] = 0xFF; // Timeout value is set as 65 ms
writeBuffer[3] = 0xFF; // Timeout value is set as 65 ms
writeBuffer[4] = 0x00;
writeBuffer[5] = 0x00;
writeBuffer[6] = 0x93; // Marker is set as 2 to get XYZ measurement. OP Code for GET1 message: 19 in Decimal.
writeBuffer[7] = ComputeCRC(0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x93); // CRC

// Transfer the content of writeBuffer to MLX90363.
noInterrupts();
digitalWrite(PinEN, HIGH);
digitalWrite(PinSS, HIGH);
//PinSS1,PinSS2, PinSS3 pull the Slave Select High turning it on
delay(10); //delay 1 microseconds as a propagation delay to prevent errors
digitalWrite(PinA0,LOW);
digitalWrite(PinA1,LOW);
for (int i = 0; i < 8; i++) { //send and receive through SPI to the sensors
    readBuffer[i] = SPI.transfer(writeBuffer[i]); //for loop to send out the array
}

digitalWrite(PinSS, LOW); //PinSS1,PinSS2, PinSS3 //pull the Slave Select Low turning it off
delay(10);
interrupts();

// Read most significant bits and add the least significant bits.
// Do this for Bx, By and Bz. Convert unsigned readBuffer data to
// signed data !Ghetto Style -if statements-!.
Bx = (readBuffer[1] & 0x3F) << 8;
Bx += readBuffer[0];
if (Bx >= 8192) {

```

```

    Bx -= 16384;
}
By = (readBuffer[3] & 0x3F) << 8;
By += readBuffer[2];
if (By >= 8192) {
    By -= 16384;
}
Bz = (readBuffer[5] & 0x3F) << 8;
Bz += readBuffer[4];
if (Bz >= 8192) {
    Bz -= 16384;
}
// Extract error bits E0 and E1, CRC and rolling counter.
errorBits = readBuffer[0] >> 6;
CRC = readBuffer[7];
rollingCounter = readBuffer[6] & 0x3F;

// Print results to serial port. Only print them if previous
// data is read by the other end. ie: Matlab in my Laptop.
// if (counter == 0) { //counter == 3) {
    if(!pass){
        if(PinSS==PinSS1){
            Serial.print("PinSS1: ");
        }
        else if(PinSS==PinSS2){
            Serial.print("PinSS2: ");
        }
        Serial.print("N" + String(currentState) + "," + String(Bx) + "," + String(By) + "," + String(Bz)+",");

        if(fullCircle==1){
            Serial.println("");
        }
    }
    sendMessage = 0;
}
}

```

Appendix K: Code for JAVA GUI

Java Code GUI display (Back End)

//Original Code for Tiger Tank Serial Communication by Henry Poon 12.25.2010
//Edited and Repurposed by David Laovoravit 4.23.2017

```
package Communicator;;

import gnu.io.*;
import java.awt.Color;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.TooManyListenersException;
import java.lang.*;

public class Communicator implements SerialPortEventListener
{
    StringBuilder inBuff = new StringBuilder();//use this instead
    String buffer= "";
    int state=0;
    int start=0;
    //passed from main GUI
    GUI window = null;

    //for containing the ports that will be found
    private Enumeration ports = null;
    //map the port names to CommPortIdentifiers
    private HashMap portMap = new HashMap();

    //this is the object that contains the opened port
    private CommPortIdentifier selectedPortIdentifier = null;
    private SerialPort serialPort = null;

    //input and output streams for sending and receiving data
    private InputStream input = null;
    private OutputStream output = null;

    //enabling and disabling buttons
    private boolean bConnected = false;

    //the timeout value for connecting with the port
    final static int TIMEOUT = 2000;

    //some ascii values for for certain things
    final static int SPACE_ASCII = 32;
    final static int DASH_ASCII = 45;
    final static int NEW_LINE_ASCII = 10;
    final static int COMMA_ASCII = 44;

    //a string for recording what goes on in the program
    //this string is written to the GUI
    String logText = "";

    public Communicator(GUI window){
        this.window = window;
    }

    //search for all the serial ports
    //pre: none
    //post: adds all the found ports to a combo box on the GUI
    public void searchForPorts(){
        ports = CommPortIdentifier.getPortIdentifiers();

        while (ports.hasMoreElements()){
            CommPortIdentifier curPort = (CommPortIdentifier)ports.nextElement();

            //get only serial ports
            if (curPort.getPortType() == CommPortIdentifier.PORT_SERIAL){
                window.cboxPorts.addItem(curPort.getName());
                portMap.put(curPort.getName(), curPort);
            }
        }
    }
}
```

```

//connect to the selected port in the combo box
//pre: ports are already found by using the searchForPorts method
//post: the connected comm port is stored in commPort, otherwise,
//an exception is generated
public void connect(){
    //window.ChangeBarData1(1000);////testing
    String selectedPort = (String>window.cboPorts.getSelectedItem();
    selectedPortIdentifier = (CommPortIdentifier)portMap.get(selectedPort);

    CommPort commPort = null;

    try{
        //the method below returns an object of typ; e CommPort
        commPort = selectedPortIdentifier.open("MQP Demo", TIMEOUT);
        //the CommPort object can be casted to a SerialPort object
        serialPort = (SerialPort)commPort;

        //for controlling GUI elements
        setConnected(true);

        //logging
        logText = selectedPort + " opened successfully.";
        window.txtLog.setForeground(Color.black);
        window.txtLog.append(logText + "\n");

        //CODE ON SETTING BAUD RATE ETC OMITTED
        //XBEE PAIR ASSUMED TO HAVE SAME SETTINGS ALREADY

        //enables the controls on the GUI if a successful connection is made
        // window.keybindingController.toggleControls();
    }
    catch (PortInUseException e){
        logText = selectedPort + " is in use. (" + e.toString() + ")";

        window.txtLog.setForeground(Color.RED);
        window.txtLog.append(logText + "\n");
    }
    catch (Exception e){
        logText = "Failed to open " + selectedPort + "(" + e.toString() + ")";
        window.txtLog.append(logText + "\n");
        window.txtLog.setForeground(Color.RED);
    }
}

//open the input and output streams
//pre: an open port
//post: initialized input and output streams for use to communicate data
public boolean initIOStream()
{
    //return value for whether opening the streams is successful or not
    boolean successful = false;

    try {
        //
        input = serialPort.getInputStream();
        output = serialPort.getOutputStream();
        //writeData(0, 0);

        successful = true;
        return successful;
    }
    catch (IOException e) {
        logText = "I/O Streams failed to open. (" + e.toString() + ")";
        window.txtLog.setForeground(Color.red);
        window.txtLog.append(logText + "\n");
        return successful;
    }
}

//starts the event listener that knows whenever data is available to be read
//pre: an open serial port
//post: an event listener for the serial port that knows when data is recieved
public void initListener(){
    try{
        serialPort.addEventListener(this);
        serialPort.notifyOnDataAvailable(true);
    }
    catch (TooManyListenersException e){
        logText = "Too many listeners. (" + e.toString() + ")";
        window.txtLog.setForeground(Color.red);
        window.txtLog.append(logText + "\n");
    }
}
}

```



```

//disconnect the serial port
//pre: an open serial port
//post: closed serial port
public void disconnect(){
    //close the serial port
    try{
        //writeData(0, 0);

        serialPort.removeEventListener();
        serialPort.close();
        input.close();
        output.close();
        setConnected(false);
        // window.keybindingController.toggleControls();

        logText = "Disconnected.";
        window.txtLog.setForeground(Color.red);
        window.txtLog.append(logText + "\n");
    }
    catch (Exception e){
        logText = "Failed to close " + serialPort.getName() + "(" + e.toString() + ")";
        window.txtLog.setForeground(Color.red);
        window.txtLog.append(logText + "\n");
    }
}

final public boolean getConnected(){
    return bConnected;
}

public void setConnected(boolean bConnected){
    this.bConnected = bConnected;
}

//what happens when data is received
//pre: serial event is triggered
//post: processing on the data it reads
public void serialEvent(SerialPortEvent evt) {

    if (evt.getEventType() == SerialPortEvent.DATA_AVAILABLE){
        try{
            byte singleData = (byte)input.read();
            if (singleData == NEW_LINE_ASCII){
                start++;
            }
            if(start>3){//skip first 3 cycles for error clear
                if (singleData != NEW_LINE_ASCII){
                    logText = new String(new byte[] {singleData});
                    window.txtLog.append(logText);
                    if (singleData != COMMA_ASCII){//parse in data until delimiter
                        inBuff.append(logText); //then pushes data out and clear buffer
                        buffer=inBuff.toString();
                    }
                }
            }
            else{

                if(!buffer.isEmpty()&&!buffer.equals("")){
                    int tempOut=0;
                    buffer=inBuff.toString();
                    if(!(state==0||state==4||state==8||state==12||state==16)){
                        tempOut= Integer.parseInt(buffer);//convert the string to an int
                    }
                    switch (state){//0,4,8,12 --> is sensor numbers so screen out
                        case 1: window.ChangeBarData1(tempOut);
                            break;
                        case 2: window.ChangeBarData2(tempOut);
                            break;
                        case 3: window.ChangeBarData3(tempOut);
                            break;
                        case 5: window.ChangeBarData4(tempOut);
                            break;
                        case 6: window.ChangeBarData5(tempOut);
                            break;
                        case 7: window.ChangeBarData6(tempOut);
                            break;
                        case 9: window.ChangeBarData7(tempOut);
                            break;
                        case 10: window.ChangeBarData8(tempOut);
                            break;
                        case 11: window.ChangeBarData9(tempOut);
                            break;
                        case 13: window.ChangeBarData10(tempOut);
                            break;
                        case 14: window.ChangeBarData11(tempOut);
                            break;
                        case 15: window.ChangeBarData12(tempOut);
                            break;
                        default:
                            inBuff.delete(0,inBuff.length());
                    }
                }
            }
        }
    }
}

```



```

jProgressBar2 = new javax.swing.JProgressBar();
jProgressBar3 = new javax.swing.JProgressBar();
jLabel15 = new javax.swing.JLabel();
jLabel16 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel17 = new javax.swing.JLabel();
jProgressBar4 = new javax.swing.JProgressBar();
jProgressBar5 = new javax.swing.JProgressBar();
jProgressBar6 = new javax.swing.JProgressBar();
jLabel18 = new javax.swing.JLabel();
jLabel19 = new javax.swing.JLabel();
jLabel20 = new javax.swing.JLabel();
jProgressBar7 = new javax.swing.JProgressBar();
jProgressBar8 = new javax.swing.JProgressBar();
jProgressBar9 = new javax.swing.JProgressBar();
jLabel22 = new javax.swing.JLabel();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
jProgressBar10 = new javax.swing.JProgressBar();
jProgressBar11 = new javax.swing.JProgressBar();
jProgressBar12 = new javax.swing.JProgressBar();
jLabel27 = new javax.swing.JLabel();
jLabel28 = new javax.swing.JLabel();
jLabel29 = new javax.swing.JLabel();
jProgressBar1 = new javax.swing.JProgressBar();
jLabel10 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
jLabel12 = new javax.swing.JLabel();

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane.setViewportView(jTextArea1);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("MQP- Demo");

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
jLabel1.setText("MQP-Demo");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel2.setText("Sensor 1");

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel5.setText("Select COM Port");

btnConnect.setText("Connect");
btnConnect.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnConnectActionPerformed(evt);
    }
});

btnDisconnect.setText("Disconnect");
btnDisconnect.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDisconnectActionPerformed(evt);
    }
});

jLabel6.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel6.setText("Sensor 3");

jLabel13.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel13.setText("Log");

txtLog.setEditable(false);
txtLog.setColumns(20);
txtLog.setLineWrap(true);
txtLog.setRows(5);
txtLog.setFocusable(false);
jScrollPane2.setViewportView(txtLog);

jLabel14.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel14.setText("X");

jProgressBar2.setMaximum(1500);
jProgressBar2.setMinimum(-500);

jProgressBar3.setMaximum(1500);
jProgressBar3.setMinimum(-500);

```

500

```
jLabel15.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel15.setText("Y");

jLabel16.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel16.setText("Z");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel3.setText("Sensor 2");

jLabel4.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel4.setText(" -

1500");

jLabel17.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel17.setText("Sensor 4");

jProgressBar4.setMaximum(1500);
jProgressBar4.setMinimum(-500);
jProgressBar4.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));

jProgressBar5.setMaximum(1500);
jProgressBar5.setMinimum(-500);

jProgressBar6.setMaximum(1500);
jProgressBar6.setMinimum(-500);

jLabel18.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel18.setText("X");

jLabel19.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel19.setText("Y");

jLabel20.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel20.setText("Z");

jProgressBar7.setMaximum(1500);
jProgressBar7.setMinimum(-500);

jProgressBar8.setMaximum(1500);
jProgressBar8.setMinimum(-500);

jProgressBar9.setMaximum(1500);
jProgressBar9.setMinimum(-500);

jLabel22.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel22.setText("X");

jLabel23.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel23.setText("Y");

jLabel24.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel24.setText("Z");

jProgressBar10.setMaximum(1500);
jProgressBar10.setMinimum(-500);

jProgressBar11.setMaximum(1500);
jProgressBar11.setMinimum(-500);

jProgressBar12.setMaximum(1500);
jProgressBar12.setMinimum(-500);

jLabel27.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel27.setText("X");

jLabel28.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
jLabel28.setText("Y");
```

0


```

        .addComponent(jLabel13)))
        .addComponent(jLabel3)
        .addComponent(jLabel2)
        .addComponent(jLabel5)
        .addGroup(layout.createSequentialGroup())
        .addComponent(cboxPorts, javax.swing.GroupLayout.PREFERRED_SIZE, 69,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnConnect)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(btnDisconnect)))
        .addGap(14, 14, 14)))
        .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jLabel17))
        .addGroup(layout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jLabel6))
        .addGroup(layout.createSequentialGroup())
        .addGap(25, 25, 25)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jLabel27, javax.swing.GroupLayout.PREFERRED_SIZE, 7,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel28, javax.swing.GroupLayout.PREFERRED_SIZE, 7,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel29))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
        .addComponent(jProgressBar11, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 972, Short.MAX_VALUE)
        .addComponent(jProgressBar10, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jProgressBar12, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
        .addGap(0, 0, Short.MAX_VALUE))
        .addComponent(jLabel10, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addComponent(jLabel11, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(jLabel12, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGap(38, 38, 38)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel5)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(cboxPorts, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(btnConnect)
        .addComponent(btnDisconnect))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel2)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel14)
        .addComponent(jProgressBar1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel15))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel16))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel4)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel3)
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jProgressBar4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel18))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar5, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel19, javax.swing.GroupLayout.Alignment.TRAILING))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addComponent(jProgressBar6, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel20, javax.swing.GroupLayout.Alignment.TRAILING))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel10)
        .addGap(1, 1, 1)
        .addComponent(jLabel6)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel22)
        .addComponent(jProgressBar7, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel23)
        .addComponent(jProgressBar8, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar9, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel24))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel11)
        .addGap(1, 1, 1)
        .addComponent(jLabel17)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel27, javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jProgressBar10, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jLabel28, javax.swing.GroupLayout.Alignment.TRAILING)
        .addComponent(jProgressBar11, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jProgressBar12, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel29, javax.swing.GroupLayout.Alignment.TRAILING))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jLabel12)
        .addGap(2, 2, 2)
        .addComponent(jLabel13)
        .addGap(5, 5, 5)
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 266, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(10, 10, 10))
);

getAccessibleContext().setAccessibleName("MQF Demo");

pack();
}

private void btnConnectActionPerformed(java.awt.event.ActionEvent evt) {
communicator.connect();
if (communicator.isConnected() == true)
{
if (communicator.initIOStream() == true)
{
communicator.initListener();
}
}
}

private void btnDisconnectActionPerformed(java.awt.event.ActionEvent evt) {
communicator.disconnect();
}
//output the data
public void ChangeBarData1 (int input){
jProgressBar1.setValue(input);
}
public void ChangeBarData2 (int input){
jProgressBar2.setValue(input);
}
public void ChangeBarData3 (int input){
jProgressBar3.setValue(input);
}
public void ChangeBarData4 (int input){
jProgressBar4.setValue(input);
}
public void ChangeBarData5 (int input){
jProgressBar5.setValue(input);
}
public void ChangeBarData6 (int input){
jProgressBar6.setValue(input);
}
public void ChangeBarData7 (int input){
jProgressBar7.setValue(input);
}
}

```

```

public void ChangeBarData8 (int input){
    jProgressBar8.setValue(input);
}
public void ChangeBarData9 (int input){
    jProgressBar9.setValue(input);
}
public void ChangeBarData10 (int input){
    jProgressBar10.setValue(input);
}
public void ChangeBarData11 (int input){
    jProgressBar11.setValue(input);
}
public void ChangeBarData12 (int input){
    jProgressBar12.setValue(input);
}
public static void main(String args[] ) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new GUI().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
public javax.swing.JButton btnConnect;
public javax.swing.JButton btnDisconnect;
public javax.swing.JComboBox cboPorts;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JProgressBar jProgressBar1;
private javax.swing.JProgressBar jProgressBar10;
private javax.swing.JProgressBar jProgressBar11;
private javax.swing.JProgressBar jProgressBar12;
private javax.swing.JProgressBar jProgressBar2;
private javax.swing.JProgressBar jProgressBar3;
private javax.swing.JProgressBar jProgressBar4;
private javax.swing.JProgressBar jProgressBar5;
private javax.swing.JProgressBar jProgressBar6;
private javax.swing.JProgressBar jProgressBar7;
private javax.swing.JProgressBar jProgressBar8;
private javax.swing.JProgressBar jProgressBar9;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JTextArea jTextArea1;
public javax.swing.JTextArea txtLog;
// End of variables declaration
}

```


Appendix L: MATLAB Code

```
%% Graphing for MQP
clc; clear all; close all;

% Load data to graph
normData = xlsread('200N_normalforce_abstract.csv');
COMData = xlsread('COMSOL_60Nnormal_abstract.csv');
XshearData = xlsread('29.7N_-xshear_abstract.csv');
YshearData = xlsread('29.7N_-yshear_abstract.csv');

% Normal Force Data
NormxForce = normData(:,4);
NormxAxis = normData(:,1);
NormyAxis = normData(:,2);
NormzAxis = normData(:,3);

% COMSOL Normal Force Data
COMxForce = COMData(:,4);
COMxAxis = COMData(:,1);
COMyAxis = COMData(:,2);
COMzAxis = COMData(:,3);

% Shear in X Data
XshearTime = XshearData(:,4);
XshearxAxis = XshearData(:,1);
XshearyAxis = XshearData(:,2);
XshearzAxis = XshearData(:,3);

% Shear in Y Data
YshearTime = YshearData(:,4);
YshearxAxis = YshearData(:,1);
YshearyAxis = YshearData(:,2);
YshearzAxis = YshearData(:,3);

% Normal Force Graph
fig1 = figure;
plot(NormxForce, NormzAxis, 'k');
hold on
grid off
plot(NormxForce, NormxAxis, 'g');
hold on
plot(NormxForce, NormyAxis, 'b');
fit1 = fitlm(NormxForce, NormzAxis);
rsq1 = fit1.Rsquared.Ordinary;
title('Magnetic Flux, 200N Normal Force');
xlabel('Applied Normal Force (N)');
ylabel('Magnetic Flux (\muT)');
legend('Z-Axis', 'X-Axis', 'Y-Axis');
textstring1 = sprintf('R^2 = %0.3f', rsq1);
text(80, 1400, textstring1);
hold off

% COMSOL Graph
fig2 = figure;
plot(COMxForce, COMzAxis, 'k');
hold on
plot(COMxForce, COMxAxis, 'g');
```

```

hold on
plot(COMxForce,COMyAxis,'b');
fit2 = fitlm(COMxForce,COMzAxis);
rsq2 = fit2.Rsquared.Ordinary;
title('Simulated Magnetic Flux, 60N Normal Force');
xlabel('Applied Normal Force (N)');
ylabel('Magnetic Flux (\muT)');
legend('Z-Axis','X-Axis','Y-Axis');
textstring2 = sprintf('R^2 = %0.3f',rsq2);
text(20,1250,textstring2);
hold off

```

```

% Shear in x Graph
fig3 = figure;
plot(XshearTime,XshearzAxis,'k');
hold on
plot(XshearTime,XshearxAxis,'g');
hold on
plot(XshearTime,XshearyAxis,'b');
title('Shear in -X-Axis, 29.7N');
xlabel('Time (s)');
ylabel('Magnetic Flux (\muT)');
legend('Z-Axis','X-Axis','Y-Axis');
hold off

```

```

%Shear in y Graph
fig4 = figure;
plot(YshearTime,YshearzAxis,'k');
hold on
plot(YshearTime,YshearxAxis,'g');
hold on
plot(YshearTime,YshearyAxis,'b');
title('Shear in -Y-Axis, 29.7N');
xlabel('Time (s)');
ylabel('Magnetic Flux (\muT)');
legend('Z-Axis','X-Axis','Y-Axis');
hold off

```

Appendix M: Offset Filter Code

```
/* THE XOR GATE MAKES HIGH ON AND LOW OFF FOR SENSORS
Code to get data from MLX90363 sensor.
```

```
This piece of code gets xyz magnetic flux magnitudes from the
sensor. It uses a function to compute CRCs.
```

```
Arduino Uno is used as the master device and a single MLX90363
sensor is used as the slave.
```

```
Written by Selim Ozel, 08.14.2015
Edited and Modified by David Laovoravit, 4.23.2017
*/
```

```
#include <SPI.h>           //Include SPI Library
#include <TimerOne.h>      //Include TimeOne Library

#define NUMBER 0
#define MAXNUMTOFILTER 2000
#define MINNUMTOFILTER -2000

#define NUMOFSENSOR 4

#define LOWERBOUNDCHECK 150
int offsetData[NUMOFSENSOR]; // = 0;
int previousZ[NUMOFSENSOR];
int firstRunCount = 0;
//boolean firstFlag = true;
//Define and initialize CRC array, 256 bytes
char CRCArray[] = { //used to check correct data transfer
0x00, 0x2F, 0x5E, 0x71, 0xBC, 0x93, 0xE2, 0xCD, 0x57, 0x78, 0x09, 0x26,
0xEB, 0xC4, 0xB5, 0x9A, 0xAE, 0x81, 0xF0, 0xDF, 0x12, 0x3D, 0x4C, 0x63,
0xF9, 0xD6, 0xA7, 0x88, 0x45, 0x6A, 0x1B, 0x34, 0x73, 0x5C, 0x2D, 0x02,
0xCF, 0xE0, 0x91, 0xBE, 0x24, 0x0B, 0x7A, 0x55, 0x98, 0xB7, 0xC6, 0xE9,
0xDD, 0xF2, 0x83, 0xAC, 0x61, 0x4E, 0x3F, 0x10, 0x8A, 0xA5, 0xD4, 0xFB,
0x36, 0x19, 0x68, 0x47, 0xE6, 0xC9, 0xB8, 0x97, 0x5A, 0x75, 0x04, 0x2B,
0xB1, 0x9E, 0xEF, 0xC0, 0x0D, 0x22, 0x53, 0x7C, 0x48, 0x67, 0x16, 0x39,
0xF4, 0xDB, 0xAA, 0x85, 0x1F, 0x30, 0x41, 0x6E, 0xA3, 0x8C, 0xFD, 0xD2,
0x95, 0xBA, 0xCB, 0xE4, 0x29, 0x06, 0x77, 0x58, 0xC2, 0xED, 0x9C, 0xB3,
0x7E, 0x51, 0x20, 0x0F, 0x3B, 0x14, 0x65, 0x4A, 0x87, 0xA8, 0xD9, 0xF6,
0x6C, 0x43, 0x32, 0x1D, 0xD0, 0xFF, 0x8E, 0xA1, 0xE3, 0xCC, 0xBD, 0x92,
0x5F, 0x70, 0x01, 0x2E, 0xB4, 0x9B, 0xEA, 0xC5, 0x08, 0x27, 0x56, 0x79,
0x4D, 0x62, 0x13, 0x3C, 0xF1, 0xDE, 0xAF, 0x80, 0x1A, 0x35, 0x44, 0x6B,
0xA6, 0x89, 0xF8, 0xD7, 0x90, 0xBF, 0xCE, 0xE1, 0x2C, 0x03, 0x72, 0x5D,
0xC7, 0xE8, 0x99, 0xB6, 0x7B, 0x54, 0x25, 0x0A, 0x3E, 0x11, 0x60, 0x4F,
0x82, 0xAD, 0xDC, 0xF3, 0x69, 0x46, 0x37, 0x18, 0xD5, 0xFA, 0x8B, 0xA4,
0x05, 0x2A, 0x5B, 0x74, 0xB9, 0x96, 0xE7, 0xC8, 0x52, 0x7D, 0x0C, 0x23,
0xEE, 0xC1, 0xB0, 0x9F, 0xAB, 0x84, 0xF5, 0xDA, 0x17, 0x38, 0x49, 0x66,
0xFC, 0xD3, 0xA2, 0x8D, 0x40, 0x6F, 0x1E, 0x31, 0x76, 0x59, 0x28, 0x07,
0xCA, 0xE5, 0x94, 0xBB, 0x21, 0x0E, 0x7F, 0x50, 0x9D, 0xB2, 0xC3, 0xEC,
0xD8, 0xF7, 0x86, 0xA9, 0x64, 0x4B, 0x3A, 0x15, 0x8F, 0xA0, 0xD1, 0xFE,
0x33, 0x1C, 0x6D, 0x42
};

//Pin declaration for Arudino Mega 2500
int PinSS = 53; //Set Slave Select Pin at 53
int PinMOSI = 51; //Set Master Out, Slave In Pin at 51
int PinMISO = 50; //Set Master In, Slave Out Pin at 50
int PinSCK = 52; //Set Clock Pin at 52
int PinEN = 22; //Set Slave Select Pin at 8
int PinA0 = 26; //Set Slave Select Pin at 9
int PinA1 = 24; //Set Slave Select Pin at 10
int currentState = 1;

int switchNum = 1;

// Buffers to read/write MLX90363
uint8_t readBuffer[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
//declare Read Buffers, as an array comprising of 8 8 bit integers
uint8_t writeBuffer[8] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
//declare Write Buffers, as an array comprising of 8 8 bit integers

// Ex,By,Bz variables
int16_t Bx =
0; //declaring Bx
variable as a 16 bit integer and set starting value at 0
int16_t By =
0; //declaring By
variable as a 16 bit integer and set starting value at 0
```

```

int16_t Bz =
0; //declaring Bz
variable as a 16 bit integer and set starting value at 0

// Error bits, CRC, virtual gain and rolling counter variables
uint8_t errorBits = //declaring error Bits
0;
as an 8 bit integer and set starting value at 0
uint8_t rollingCounter = //declaring Rolling Counter
0;
as an 8 bit integer and set starting value at 0
uint8_t CRC = //declaring
0;
Cyclic Redundancy Check as an 8 bit integer and set starting value at 0

// CRC function
uint8_t ComputeCRC(uint8_t Byte0, uint8_t Byte1, uint8_t Byte2, uint8_t Byte3, uint8_t Byte4, uint8_t Byte5, uint8_t Byte6) {
//Parse each byte into the CRC, pulling out of the CRC array
uint8_t CRC = 0xFF;
CRC = CRCArray[CRC ^ Byte0]; //Parsing Byte through an XOR gate to crate a CRC
CRC = CRCArray[CRC ^ Byte1];
CRC = CRCArray[CRC ^ Byte2];
CRC = CRCArray[CRC ^ Byte3];
CRC = CRCArray[CRC ^ Byte4];
CRC = CRCArray[CRC ^ Byte5];
CRC = CRCArray[CRC ^ Byte6];
CRC = ~CRC; //Flipping the Bits gets out the input that is wanted
return CRC;
}
int fullCircle=0;
// Sets the send message flag
int sendMessage = 0; //Set a flag for message initialize to 0 (false)
void SendFlag()//A helper method for setting messenger flag to true
{
sendMessage = 1;//Set a flag to 1(true)
}

void setup() //The setup before starting the looping program
{
// Mark comm pins as output or input
pinMode (PinMOSI, OUTPUT);//Set MOSI pin as an output
pinMode (PinMISO, INPUT);//Set MISO pin as an input
pinMode (PinSCK, OUTPUT);//Set Clock pin as out put

// Make the MLX90363 sensor the active slave device
pinMode (PinSS, OUTPUT); //Set Slave Select pin as an output
pinMode (PinEN, OUTPUT);
pinMode (PinA0, OUTPUT);
pinMode (PinA1, OUTPUT);
digitalWrite(PinSS, LOW); //Turn off the Slave Select
digitalWrite(PinEN, HIGH); //Turn off the Slave Select1
digitalWrite(PinA0, LOW); //Turn off the Slave Select2
digitalWrite(PinA1, LOW); //Turn off the Slave Select3

// Begin serial Comm
Serial.begin(9600); //Set the baudrate to 9600

// Required SPI configuration to communicate with MLX90363
// Details of SPI settings can be found in "Getting Started
// Guide" [GSG], under "SPI bus protocol".
SPI.begin();
//Initialize SPI and set SCK,MOSI,SS to outputs and SCK and MOSI as low and SS as high
SPI.setBitOrder(MSBFIRST); //Set protocol to transmit Most significant bit first
SPI.setClockDivider(SPI_CLOCK_DIV32);
//Set the clock to be 1/32 the frequency of the system clock
SPI.setDataMode(SPI_MODE1);
//SPI_MODE1 = Clock Polarity: 0 | Clock Phase: 1 | Output Edge: Rising | Data Capture: Falling

// Setup Timer for sending/receiving data
Timer1.initialize(500);//30000 //1000 //change this to 3000 if you want to run in with JAVA/////Initialize frequency of
interrupt to 0.03 seconds
Timer1.attachInterrupt(SendFlag); //trip flag every 0.03 seconds
}
int counter = 0;
void loop() //

{
if (sendMessage) { //If the timer interrupt trips (0.03 seconds) collect data from sensor
//int ReadPin = 0;
switch (switchNum) //switching between each sensor on the module
case 1:
fullCircle=0;
currentState = 1;
}
}

```

```

digitalWrite(PinA0,LOW);
digitalWrite(PinA1,LOW);
//ReadPin = PinSS;
if (counter >= NUMBER) (//used for adjusting how many times each sensor is run for each run through
    switchNum ++;
    counter = 0;
}
else {
    counter++;
}
break;
case 2:
currentState = 2;
digitalWrite(PinA0,LOW);
digitalWrite(PinA1,HIGH);
//ReadPin = PinSS1;
if (counter >= NUMBER) (//used for adjusting how many times each sensor is run for each run through
    switchNum ++;
    counter = 0;
}
else {
    counter++;
}
break;
case 3:
currentState = 3;
digitalWrite(PinA0,HIGH);
digitalWrite(PinA1,LOW);
//ReadPin = PinSS3;
if (counter >= NUMBER) (//used for adjusting how many times each sensor is run for each run through
    switchNum ++;
    counter = 0;
}
else {
    counter++;
}
break;
case 4:
currentState = 4;
digitalWrite(PinA0,HIGH);
digitalWrite(PinA1,HIGH);
//ReadPin = PinSS2;
if (counter >= NUMBER) (//used for adjusting how many times each sensor is run for each run through
    switchNum = 1;
    counter = 0;
}
else {
    counter++;
}
fullCircle=1;
break;
default:
    Serial.println("Error");
}

// Create a GET1 message. Format of messages are explained in both DataSheet
// [DS] and GSG.
writeBuffer[0] = 0x00; //this is a set up step to have the sensor be a GET1 See DataSheet
writeBuffer[1] = 0x00;
writeBuffer[2] = 0xFF; // Timeout value is set as 65 ms
writeBuffer[3] = 0xFF; // Timeout value is set as 65 ms
writeBuffer[4] = 0x00;
writeBuffer[5] = 0x00;
writeBuffer[6] = 0x93; // Marker is set as 2 to get XYZ measurement. OP Code for GET1 message: 19 in Decimal.
writeBuffer[7] = ComputeCRC(0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x93); // CRC

// Transfer the content of writeBuffer to MLX90363.
noInterrupts();
digitalWrite(PinEN, HIGH);
digitalWrite(PinSS, HIGH); //PinSS1,PinSS2, PinSS3 pull the Slave Select Low turning it on
delay(10);
//delay 1 microseconds as a propagation delay to prevent errors
digitalWrite(PinA0,LOW);
digitalWrite(PinA1,LOW);
for (int i = 0; i < 8; i++) { //send and received through SPI to the sensors
    readBuffer[i] = SPI.transfer(writeBuffer[i]); //for loop to send out the array
}
//delay 15 microseconds as a propagation delay to
prevent errors
digitalWrite(PinSS, LOW); //PinSS1,PinSS2, PinSS3 pull the Slave Select High turning it off
// digitalWrite(PinEN, LOW);
delay(10);
interrupts();

// Read most significant bits and add the least significant bits.
// Do this for Bx, By and Bz. Convert unsigned readBuffer data to
// signed data !Ghetto Style -if statements-!.
String StringBx = "";

```

```

String StringBy = "";
String StringBz = "";

Bx = (readBuffer[1] & 0x3F) << 8;
Bx += readBuffer[0];
if (Bx >= 8192) {
    Bx -= 16384;
}
if(Bx<=MAXNUMTOFILTER && Bx >= MINNUMTOFILTER){
    StringBx= (String)Bx;
}
By = (readBuffer[3] & 0x3F) << 8;
By += readBuffer[2];
if (By >= 8192) {
    By -= 16384;
}
if(By<=MAXNUMTOFILTER && By >= MINNUMTOFILTER){//band pass filter
    StringBy= (String)By;
}
Bz = (readBuffer[5] & 0x3F) << 8;
Bz += readBuffer[4];
if (Bz >= 8192) {
    Bz -= 16384;
}
if(Bz<=MAXNUMTOFILTER && Bz >= MINNUMTOFILTER){//band pass filter
    if(firstRunCount<NUMOFSENSOR*4){//offset filter
        previousZ[currentState-1]=Bz;
        offsetData[currentState-1]=0;
        firstRunCount++;
    }
    if(abs(previousZ[currentState -1]-Bz)>LOWERBOUNDCHECK){
        offsetData[currentState-1] = offsetData[currentState-1]+(previousZ[currentState -1]-Bz);
    }
    previousZ[currentState -1]=Bz;
    Bz = Bz+offsetData[currentState-1];//offset data is made for general need. x[3] y[3] z[3]. if two modules x[3][3]

    StringBz= (String)Bz;
}

// Extract error bits E0 and E1, CRC and rolling counter.
errorBits = readBuffer[0] >> 6;
CRC = readBuffer[7];
rollingCounter = readBuffer[6] & 0x3F;

// Print results to serial port. Only print them if previous
// data is read by the other end. ie: Matlab in my Laptop.
Serial.print("N" + String(currentState) + "," + StringBx + "," + StringBy + "," + StringBz+",");

    if(fullCircle==1){
        Serial.println("");
    }
sendMessage = 0;
}
}

```

Appendix N: 3rd Iteration COMSOL Data Table

Magnetic Flux Values vs. Simulated Force Applied

X (mT)	Y (mT)	Z (mT)	Force (N)
0.159	0.113	0.764	0
0.183	0.144	0.85	10
0.213	0.18	0.943	20
0.248	0.222	1.044	30
0.292	0.272	1.154	40
0.349	0.334	1.273	50
0.435	0.414	1.402	60