Mechanical Engineering Capstone Final Report

ComSole

**Tyler Gabriel, Will Grubbs, Ian Lansdowne,
Mark McCulloch, Matthew Monson, Benito Tagle Ochoa**

Sponsor: Dr. Ya Wang

Studio Instructor: Dr. Ni Wang

MEEN-402 Section 504
Texas A&M University
December 8, 2025

TEXAS A&M UNIVERSITY
J. Mike Walker '66 Department of
Mechanical Engineering

Contents

Executive Summary

Parkinson's Disease is a progressive neurodegenerative disorder that affects critical biological functionality and includes motor loss issues like freezing of gait. The mission of the ComSole capstone project, sponsored by Dr.Ya Wang, is to create an insole-based product that helps fight against these debilitating symptoms through real-time detection and mitigation. Research into the science behind freezing of gait and similar market prototypes helped the team understand the most effective hardware choices and layout to incorporate high-quality data collection as well as reliable and precise stimulation. Key findings from the aforementioned research, as well as in-depth customer interviews, showed that a relatively low-cost IMU located in the heel of the foot with a haptic motor placed on the first metatarsal joint would optimize both input and output quality of an insole-based gait monitoring device with integrated feedback. Paired with a supervised and quasi-parametric machine learning model onboard an MCU, this system would be able to process gait data input and make accurate decisions for feedback output through the aforementioned modes. In addition, both customer response and clinical trials have shown that the most effective mode of stimulation is vibrational feedback pulses, with a need for FoG interventions being as often as multiple times a day. From these findings, the team was able to formulate design requirements, proceed through stages of design ideation to refinement, and eventually come to a final concept selection, which has been named ComSole. This product utilizes a single IMU, MCU, and haptic motor encased in a TPU outer insole along with the necessary electrical connections and a 3.3-volt, 1200 mAh battery supply. As shown in Figure 1 below, the controller unit and battery supply were chosen to exist in a case outside of the insole, while the IMU was placed at the heel, and the haptic motor was placed at the first metatarsal joint. This design was found to have the best overall ratings across all the major functional requirements as determined from the aforementioned research.

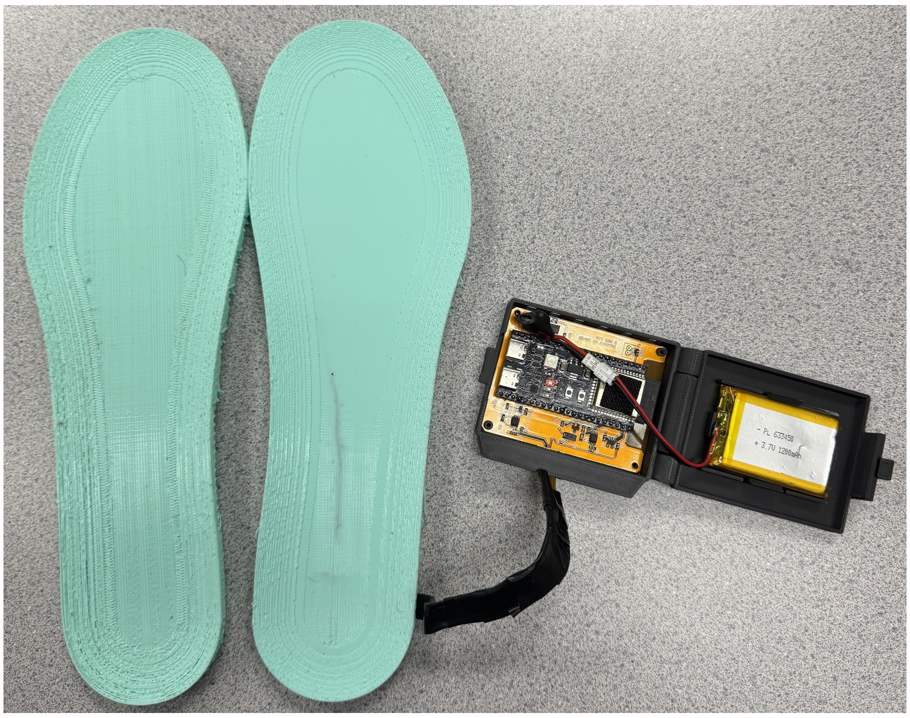


Figure 1: ComSole Final Prototype

Coupled with the ComSole product, the team also developed a companion mobile application. This app serves to track user gait data and display pertinent analytics for both the patient and a medical professional to use and process for a better understanding of each user's unique lifestyle. All firmware for the app was developed using a VS Code extension called Flutter with an SQLite database for local data storage.

Through the development, production, and rapid manufacturing of both product architectures, the team was able to successfully accomplish the creation of 5 working insole sets, one of which had the full model embedded haptic feedback system, three different firmware packages for data collection and processing, a fully refined mobile application for gait analytics, and an experimentally tested gait collection procedure. Figure 1 above shows the final system prototype with the embedded machine learning model and haptic feedback system, and Figure 2 shows all sets of functional insole products.

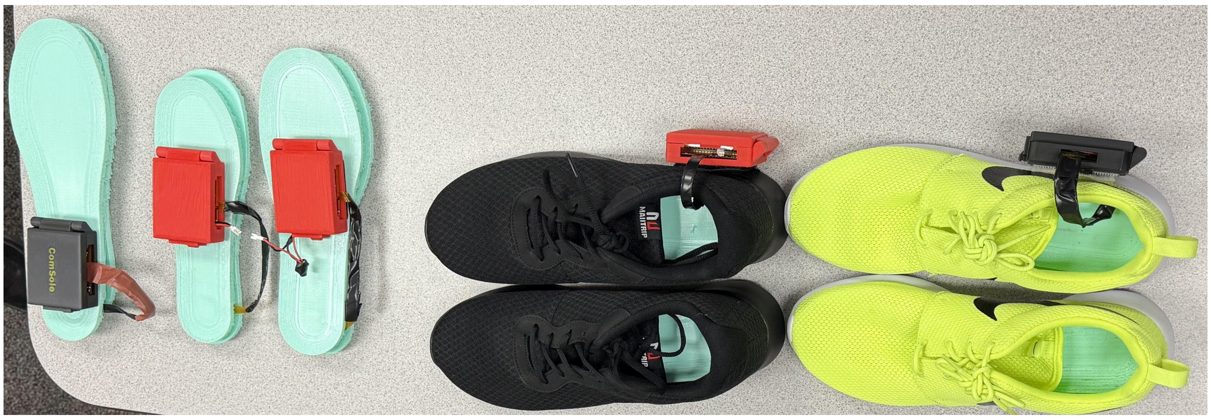


Figure 2: Finalized 5 Working Insole Sets

These insoles serve largely as a proof of concept for a future product that needs continued development. Future work should be largely focused on further electronics integration, increased machine learning recognition capabilities, and software optimization efforts. In addition, more patient data will need to be collected, labeled, and used to train the machine learning model.

Glossary

Term	**Definition**
Accelerometer	Device that measures the acceleration of an object relative to a free-fall observer.
BLE	Bluetooth low energy.
CAD	Computer-aided design.
ComSole	Command Insole.
CSV	Comma Seperated Values
DBS	Deep Brain Stimulation.
EVA	Ethylene Vinyl Acetate.
FEA	Finite element analysis.
FMECA	Failure modes, effects, and criticality analysis.
FoG	Freezing of Gait.
FTA	Fault tree analysis.
Gait	A person's kinetic movement patterns.
G-code	Geometric Code, used to control a 3D printer.
HoQ	House of Quality.
IMU	Inertial Measurement Unit.
IP	Intellectual Property.
MCU	Microcontroller Unit.
ML	Machine Learning.
MONI	Monitoring Insole.
MSE	Mean Square Error.
PCB	Printed circuit board.
PD	Parkinson's Disease.
PLA	Polyactic Acid or polyactide, a plastic material used in 3D printing.
Random Forest Classifier	Machine Learning ensemble classifier using multi-layered decision trees that can better predict and aggregate data-driven outcomes.
Sensitivity	Accuracy in detecting an occurrence.
SNPS	Solution Neutral Problem Statement.
Somatosensory	Physical perception of external stimuli like touch, vibration, pain, or temperature.
Specificity	Accuracy to detect the lack of an occurrence (false positives).
STL File	Stereolithography file, a mesh 3D model format.
TAMU	Texas A&M University
TPU	Thermoplastic Polyurethane, a flexible material used in 3D printing.
WWS	Windy Webb Schoenewald (Parkinson's Treatment)

1 Introduction and Problem Definition

Parkinson's disease is a neurodegenerative disorder that affects about 1% of all people over the age of 60 [1]. It causes several debilitating symptoms, from hand and foot tremors to freezing limbs for extended periods. The cause of Parkinson's remains unknown, and while it is a chronic disease, various treatments can help manage symptoms. One of the most challenging symptoms is Freezing of Gait (FoG), which affects 38–65% of Parkinson's patients [2]. FoG episodes occur unpredictably while walking, preventing the patient from lifting their foot mid-stride. These episodes can often be alleviated by applying a visual, auditory, or tactile cue, which helps the patient refocus and regain mobility. Although existing devices attempt to address FoG, they lack comfort, portability, and consistency in identifying and mitigating these events.

This senior capstone team, consisting of Tyler Gabriel, Will Grubbs, Ian Lansdowne, Mark McCulloch, Matt Monson, and Benito Tagle Ochoa, is developing an advanced insole-based solution to detect and mitigate FoG. The project is sponsored by Dr. Ya Wang, a researcher in human sensing and activity tracking under the Department of Mechanical Engineering at Texas A&M University. Dr. Wang's lab previously developed MONI (Monitoring Insole) [3], an early iteration of an insole designed to monitor Parkinson's symptoms. Building on this research, the team aims to create a next-generation insole with improved detection accuracy and real-time intervention capabilities.

1.1 Needs Analysis

Being a product designed around people who suffer from Parkinson's, a full-scale needs analysis was done in the design portion of the project, which showed several key metrics and needs from the analysis. A variety of different customer interviews were conducted to assess the needs and relative priorities for the product, from which the following was obtained. The three most important metrics identified were on-demand cueing, low cost, and a preference towards vibrational feedback. Polls from the interviews done can be seen below (Figures 3-5), which highlight these trends.

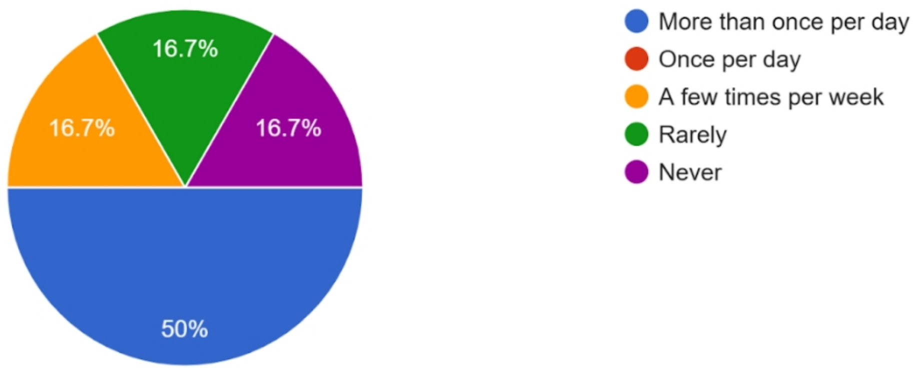


Figure 3: Freezing of Gait Frequency

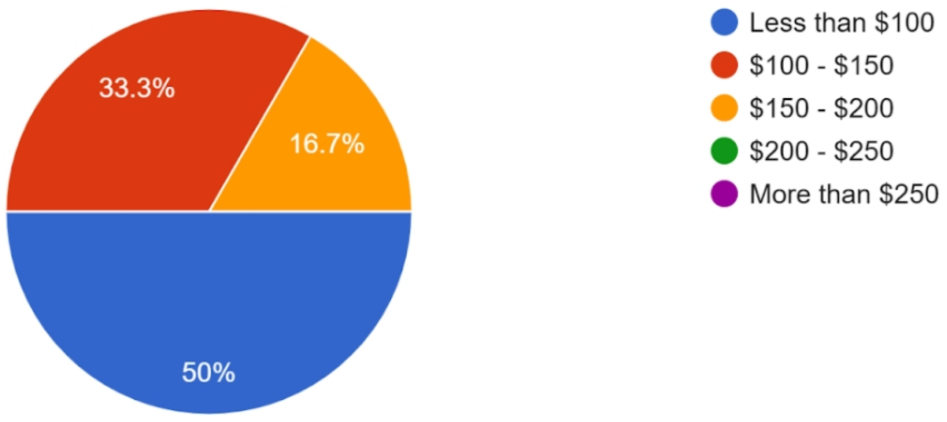


Figure 4: Freezing of Gait Assistance Device Pricing

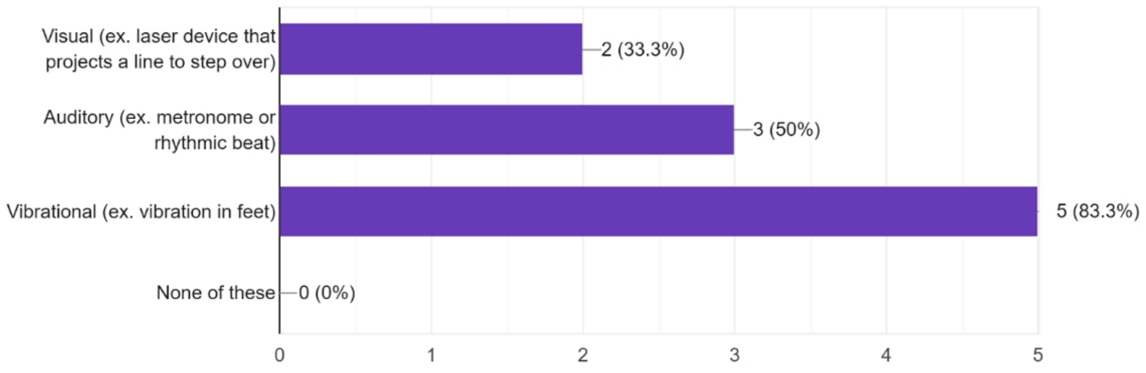


Figure 5: Patient Cue Preference

These needs were critical in establishing a purpose and direction for the project, which is summarized by the solution-neutral problem statement as “an insole-based prototype with an integrated mobile application to improve the quality of life for Parkinson’s Disease patients who deal with Freezing of Gait through real-time detection and intervention.”

The creation of an insole-based FoG detection and mitigation system was the next step in Parkinson’s disease symptom management. In order for this device to be considered successful, it needed to fulfill many customer requirements. During customer interviews, the team found that none of the patients interviewed currently used any sensory cueing or a data collection device similar to ComSole. Of the many reasons given for this, weight, ease of use, and accuracy were the most important contributors listed. Upon further consideration, these needs are intuitive because of the nature of the disease being treated. Because Parkinson’s, and more specifically FoG, greatly affects motor function, designing a product that does not make motor function more difficult is essential. For example, if the product was heavy, then customers who were already struggling with mobility would have an additional weight to carry with them, which would likely cause more issues than the product aims to remedy. Furthermore, if the device is difficult to use, then problems such as insertion, removal, and charging will cause more strife than the product will remove from everyday life. Using these observations, the team determined that in order for this product to be successful, it needed to be comfortable, which is best measured by its weight, ease of use (insertion, removal, and charging), and effectiveness at real-time detection and cueing of FoG events.

The usage environment for the device is everyday life for Parkinson's patients. This means that patients should be able to operate and maintain the device without assistance from any professionals. Patients should be able to complete all tasks necessary for usage and management. These tasks include the insertion and removal of the insole, charging, and data management and visualization. For these tasks to be accomplished, the insole needed to be lightweight, fit properly into the user's shoes, be easy to charge, and have a simple-to-use method of accessing collected data.

1.2 Design Parameters

The target values for the design requirements were determined in a variety of ways, such as customer interviews, market research, and discussions with sponsors. Based on the team's discussions and previous analysis, it was determined that the accuracy of the insole in detecting freezing of gait was the most important requirement of the insole. Through discussion with sponsors and reading through research papers they have published, 90% was seen as an ambitious but reasonable target for the insole based on the timeline of the project. This means that the machine learning model will have a 90% accuracy in determining when freezing is occurring, and feedback is being provided. The team didn't want the insole to be vibrating constantly when it isn't needed, but also wanted to ensure that it is vibrating when required. The next most important was the ability for the insole to be able to provide real-time feedback to mitigate freezing in a timely manner. The team chose a target of less than 5 seconds between detection and feedback based on a recommendation from Dr. Wang, along with research paper data. The next requirement ranked by importance was the cost of the insole. The cost of less than $200 was determined through customer surveys and interviews, as everyone the team talked to wanted it to be less than that amount. The goal is to get the price as low as possible, but this is a good benchmark for the team to strive for. Fourth was that the insole could extract meaningful gait data, and the target value was a 100 Hz sampling frequency. These values are more specific targets that are related to the data that will be collected and displayed by the product. The team determined these mostly through previous experience with similar types of data collection, as well as discussions with the sponsors.

The next design requirement was that the insole fit in the user's shoes. It is important to make sure that the insoles will be available to as many people as possible, which means creating multiple sizes that fit different shoe sizes. The early prototypes will fit men's sizes 8-11 shoes, and women's size 7 for data collection, as this is the size for the majority of the ComSole team, as well as patient foot sizes. In addition to length, the team needed to ensure that the insole is the correct width for standard shoes. The sixth requirement is the comfort of the insole, which had a target of 7 on a scale of 1-10. This was subjective based on surveying people who tested out the prototype, but the team decided that 7/10 was a good minimum comfort level, as this would likely mean that the insole would be comfortable enough to wear for multiple hours at a time. Next is that the battery life of the insole lasts at least one week under normal usage on one charge. This target comes from other market research, as NUSHU (the only market prototype that provides feedback) has a battery life of 8 hours, and the team is hoping to improve on that. The team wants to minimize the frequency of recharges for patients who may be forgetful or struggle with plugging the system in to charge. Finally, the insole requires an onboard thermal management system. Because this is a product worn directly on the human body, it is essential that the system detects any thermal runaway events and terminates all activity if an event occurs. These eight design requirements compose the key factors that the team has guaranteed to the sponsors that the product will have.

1.3 Functional Structure

The functional modeling process for ComSole helped identify key operations the system must perform to meet user needs and design requirements. By breaking ComSole down into its core functions, inputs, and outputs, the team was able to better understand the system's behavior and interactions.

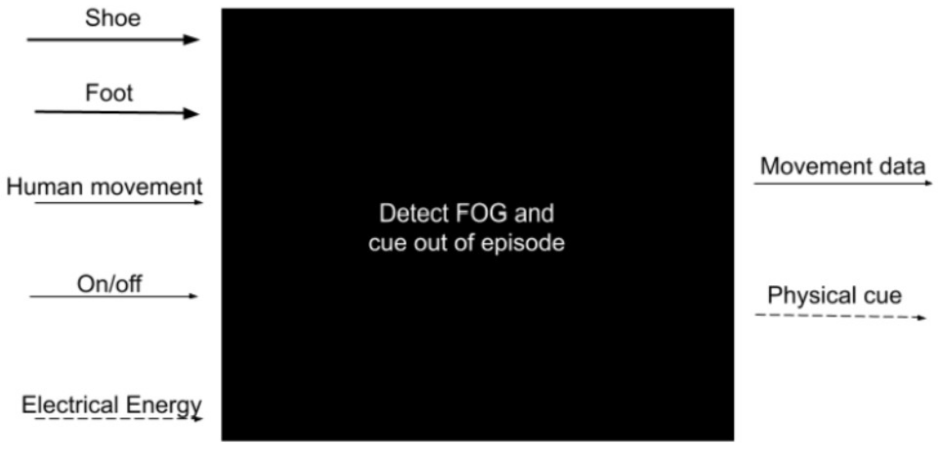


Figure 6: Black Box Model

Figure 6 shows the black box model for ComSole, which was used to map the relationship between the system's inputs and outputs at a high level. The key functions were identified to be power, movement, force, the foot, and the shoe. These inputs interact with the internal functions of the system to produce specific outputs: a haptic cue when FoG is detected, data for the app interface, heat generated by electrical components, etc.

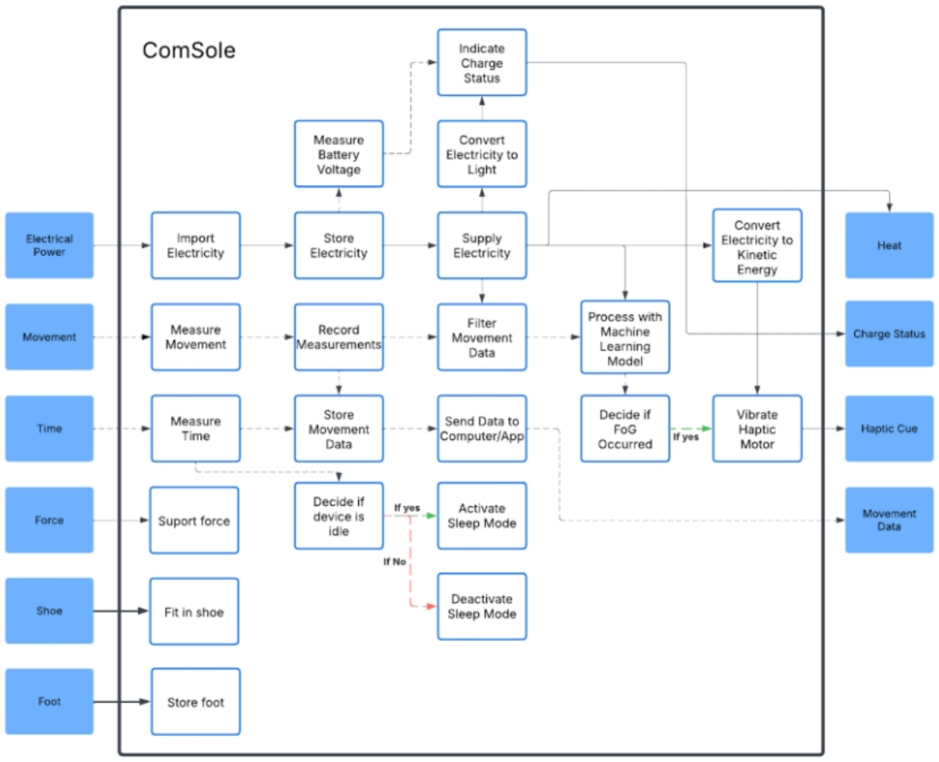


Figure 7: Functional Flow Diagram

A more detailed breakdown of these operations is provided in the functional flow diagram seen in Figure 7, which shows how data flows through the system: from raw sensor measurement through processing and classification using a machine learning model, to delivery of feedback. Alongside movement and force sensing, ComSole manages power delivery, charge state monitoring, and power optimization features like idle detection and sleep mode activation. The primary internal functions of the device include:

- Data Collection: Movement and force inputs are detected and recorded through inboard IMUs and other sensors.
- Determine Whether FoG is Occurring: The data is filtered and processed through a machine learning model trained to detect characteristic FoG patterns.
- Implement Symptom Mitigation (Cue): When FoG is detected, the system activates a haptic motor to deliver vibrational feedback to the user.
- Store Electrical Energy: The battery stores power and supplies it to all electrical components.
- Support Foot: The device must physically support the user's foot without causing discomfort

The structured functional model approach was critical when ensuring all design requirements are tied to a system behavior. For example, the ability to support real-time feedback is reflected in the transition between data acquisition to cue delivery. Moreover, the modular power-saving logic contributes to the overall device longevity - an important factor identified in both the HOQ and customer interviews.

2 System Description

2.1 Overview

The ComSole system consists of a shoe-mounted, wearable device to detect and prevent Freezing of Gait (FOG) in Parkinson's patients, shown below in Figure 8. It integrates hardware and software components into a small enclosure that can be mounted onto shoes without significantly impacting gait or comfort. Hardware incorporates a microcontroller board, an inertial measurement unit (IMU) for monitoring motion, and a rechargeable battery for implementation in portable systems. These electronics are housed within a specially adapted PLA casing providing mechanical protection and user access. A clip or Velcro attachment system secures the case to the shoe, with thermal vents to avoid overheating. Internal features such as screw holes, recesses, and snap-fit retainers provide correct alignment and strain relief of the wiring and PCB. The software component, visualized as a flowchart in Figure 9, analyzes IMU signals in real-time to detect aberrant gait signatures that relate to freezing episodes. Once detected, the microcontroller is able to stimulate piezoelectric vibrators incorporated in the device to deliver rhythmic sensory input and enable the patient to continue walking. Together, the mechanics, software, and electronics form a system that can both sense and act on FOG in real time. The product is lightweight, durable, and ergonomic, ideal for long-term wear during routine activities.

Table 1 below shows an interface breakdown of the primary systems with their associated functional requirements.

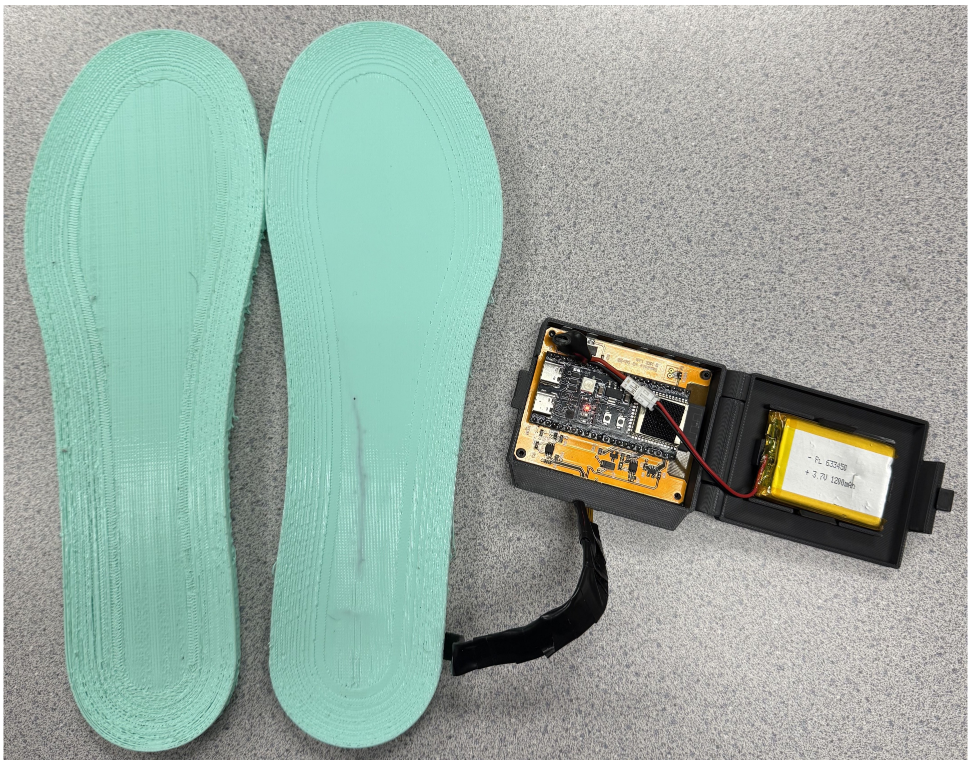


Figure 8: Assembly Picture with all Components

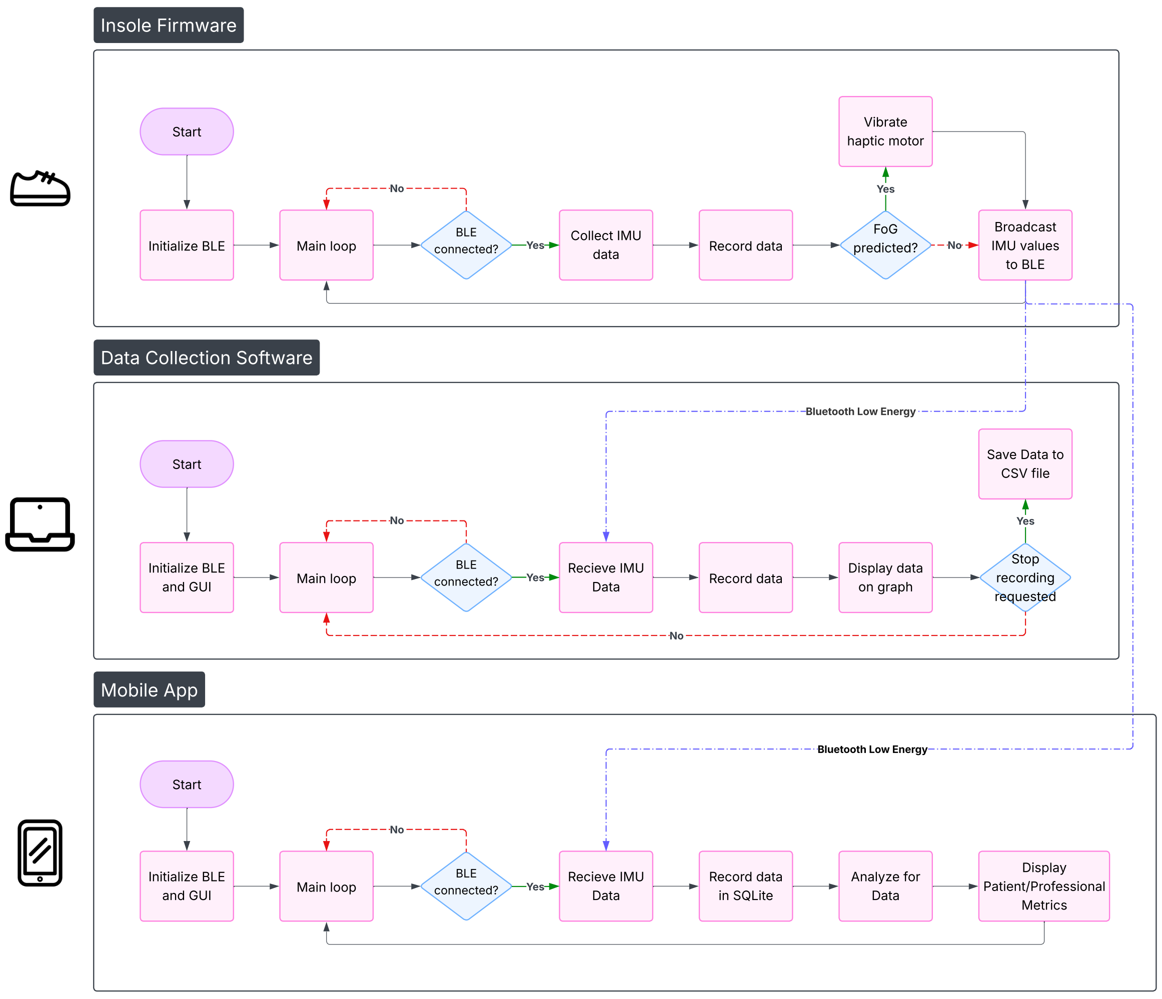


Figure 9: Insole Firmware, Data Collection Interface, and Mobile Application Flowchart

Table 1: ComSole Interface Breakdown with System Requirements

Subsystem	**Interface**	**System Requirements**
Electronics	Haptic Motor	• Receive 5V signal high and vibrate at 10K RPM.
Electronics	IMU	• Record and transmit 9-DOF absolute orientation data.
Electronics	PCB Microcontroller	• Record and upload IMU data through I/O ports • Control haptic motor and driver • Run embedded software
Electronics	Li-ion Battery	• Provide 3.7V at 1200 mAh
Mechanical Design	TPU Insole	• Sustain 10^7 cycles of usage • Withstand up to 300 lbs of human weight in normal conditions without electronic deformation
Digital/Software	Digital GUI	• Accurately display gait data parameters from IMU data transmission.
Mechanical Design	PCB Casing	• Protect PCB and battery hardware • Keep electronics contained within range of insole • Dissipate normal levels of heat from PCB to keep within operating conditions
Mechanical Design	IMU Casing	• Protect IMU from mechanical deformation.

2.2 Electronic PCB

The insole PCB connects the electronic components of the ComSole device listed in Table 1, including the microcontroller, IMU, haptic vibration motor, and battery. The team's PCB was designed by Dr. Ya Wang's research lab to meet the team's specifications for functionality of the device. A schematic of the PCB, included in Appendix A.1, shows how each of the electronic components used in the design is connected by the PCB.

Components of the PCB are labeled in Figure 10. The IMU and haptic motor are connected by a flexible bridge, which is designed to pass between the inside and outside of the shoe, routed over the side of the shoe. This flexible portion of the PCB allows the board to be printed as a single component and unifying the electronics inside and outside the insole.

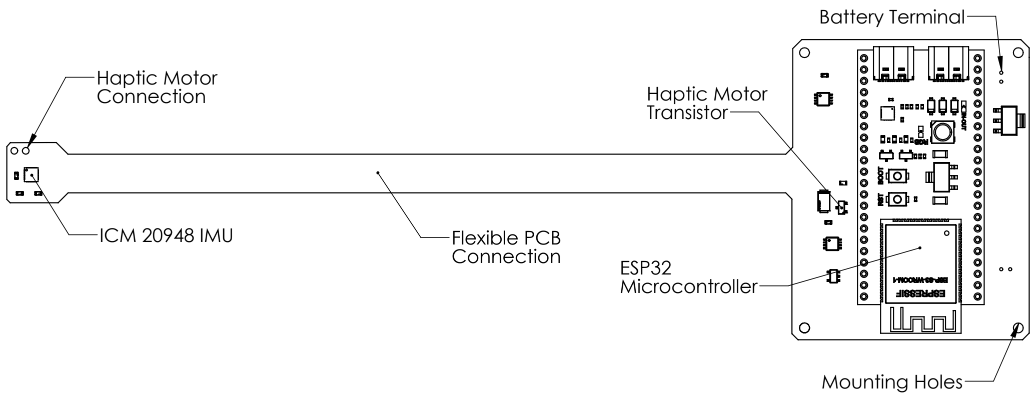


Figure 10: Insole PCB Drawing

2.3 Electronic Hardware Case Design

The hardware enclosure for the ComSole went through several design stages as the team developed a better understanding of the final components and implementation challenges. From the start, the team's main goals were to protect the electronics, keep all components secure during walking/freezing events, and ensure that the housing did not irritate the user when making contact with their foot. Additionally, the enclosure needed to provide easy access to microcontrollers for USB/battery charging and internal maintenance while still supporting rapid iteration. Additive manufacturing using PLA was selected early on because it offered tight tolerances, rigid mechanical behavior ideal for living hinges and snap-fits, low warping, and fast turnaround time for prototypes.

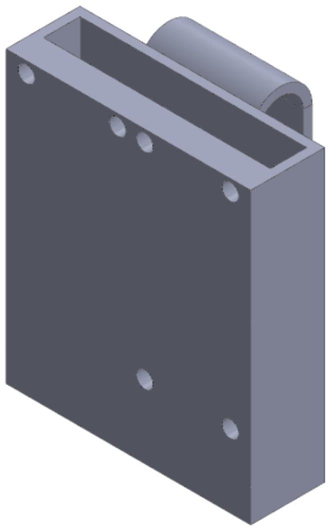


Figure 11: Original PLA Hardware Case with Externally Mounted Electronics

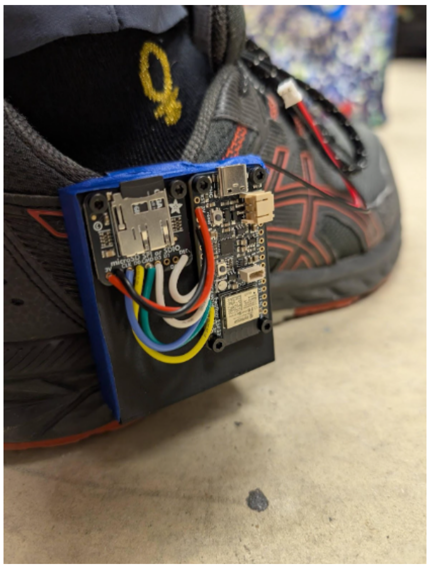


Figure 12: Original PLA hardware Case Inserted in Shoe

The first version of the case, shown in Figure 11, was very simple. All electronics, excluding the battery, were mounted externally on the back of the case and secured to the shoe using a clip that hooked into the shoe, as shown in Figure 12. The battery slid into a shallow slot while the entire assembly was clipped onto the shoe. At this point, the flexible PCB had not been fully developed, so all components were placed on the outside of the enclosure as a proof-of-concept and for ease of use. While functional, the design had easily-identifiable drawbacks, such as that the electronics were exposed and the layout made maintenance difficult, and provided inconsistent tolerancing for securing the battery. These limitations motivated the need for a fully enclosed and more refined design.

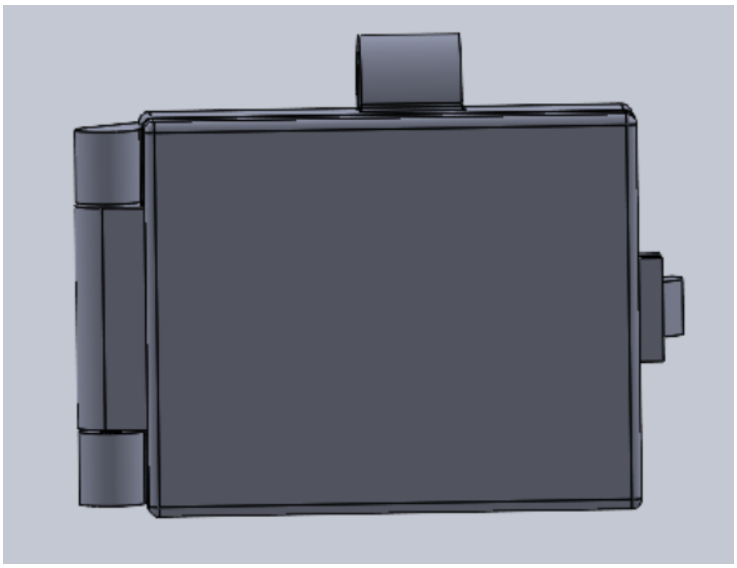


Figure 13: CAD Graphic for the Front Case Iteration 1

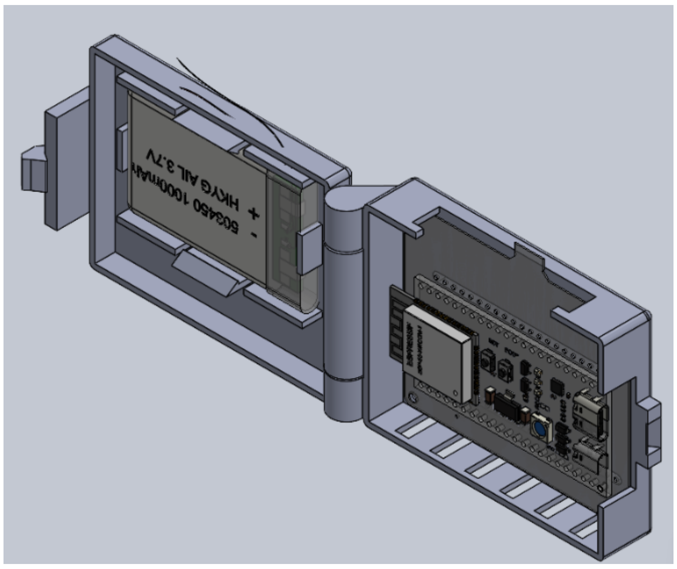


Figure 14: CAD Open Position Graphic for the Front Case Iteration 1

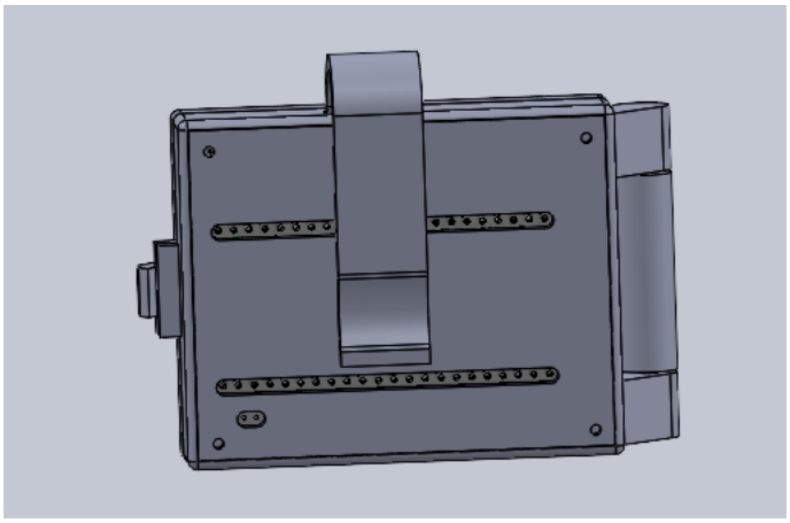


Figure 15: CAD Backside Graphic for the Front Case Iteration 1

The next major iteration, shown in Figure 13 - 15, was the first integrated enclosure that fully protected the electronics. This version introduced several key mechanical features, including a living hinge on the left side that was printed with the entire enclosure as a single piece, a snap-fit battery mount inside the case, and a snap-fit latch on the right side for closure without the use of external fasteners. A dedicated PCB mount was implemented on the right side within the case, along with an external clip that guided the PCB wiring into the shoe. This design printed successfully on the first attempt, with the living hinge and snap-fit latch functioning exactly as intended. However, testing revealed several issues. The rear routing clip created friction on the ankle. The PCB and the attached IMU were difficult to insert and remove because they had to be slid under the top slot of the case, USB charging access was partially obstructed, and tolerancing on the battery mount needed refinement.

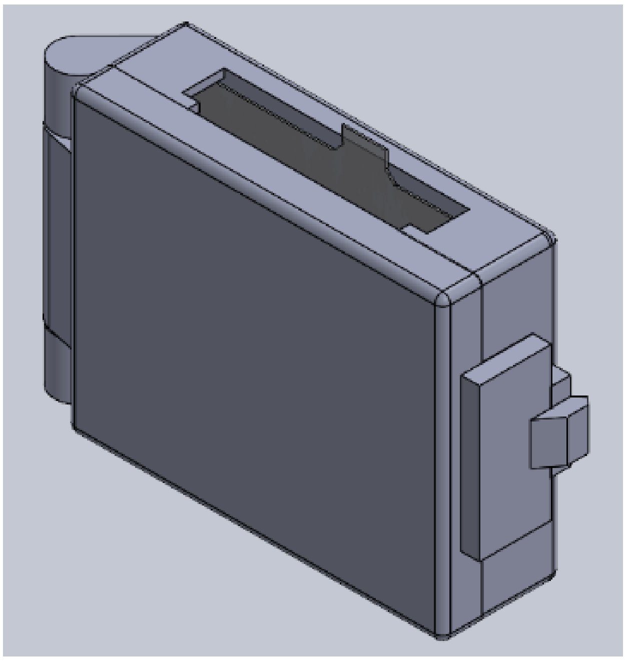


Figure 16: Final Case CAD Design

These findings guided the development of the final enclosure design, shown in Figure 16. The rear routing clip was removed entirely and replaced with a Velcro attachment on the backside of the case, which significantly improved comfort. The top opening was enlarged to make assembly and disassembly of the PCB and haptic motor substantially easier. Internally, the battery pocket was shifted to one side to create additional space for wiring, the voltage regulator, and any flexible PCB routing. The snap-fit latch was deepened to prevent accidental opening during gait cycles or shuffling, ensuring reliable operation during data collection. Vertical ventilation slots were added to improve heat dissipation from the microcontroller and battery during prolonged machine learning and sensor usage.

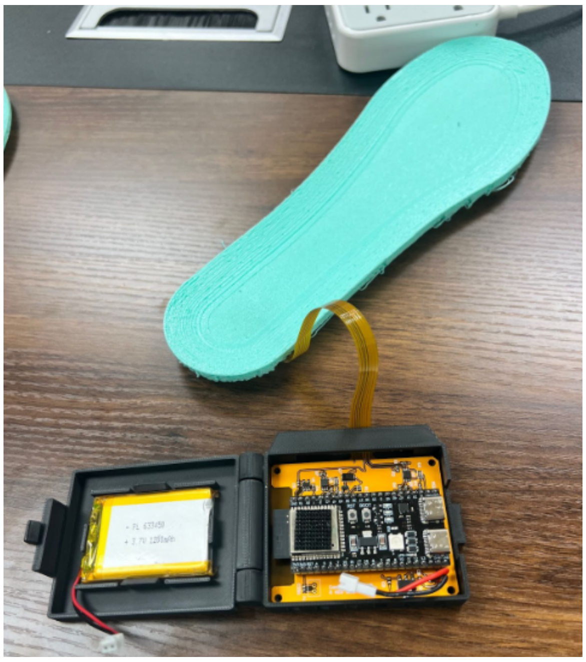


Figure 17: Final Design Integration Image

A CAD rendering of this updated design is shown in Figure 16, and a functional prototype from the previous iteration is shown above in Figure 17. The engineering drawings used for manufacturing are provided below in Figures 18 - 20.

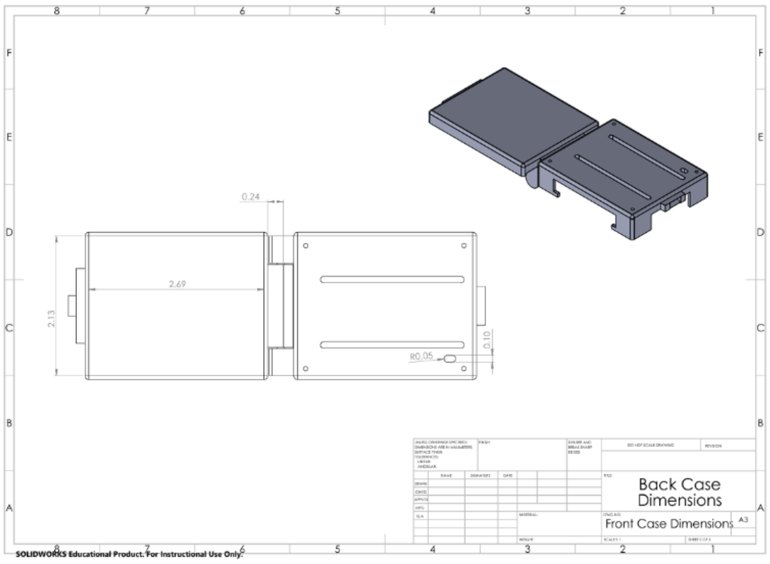


Figure 18: Final Case Backside Engineering Drawing

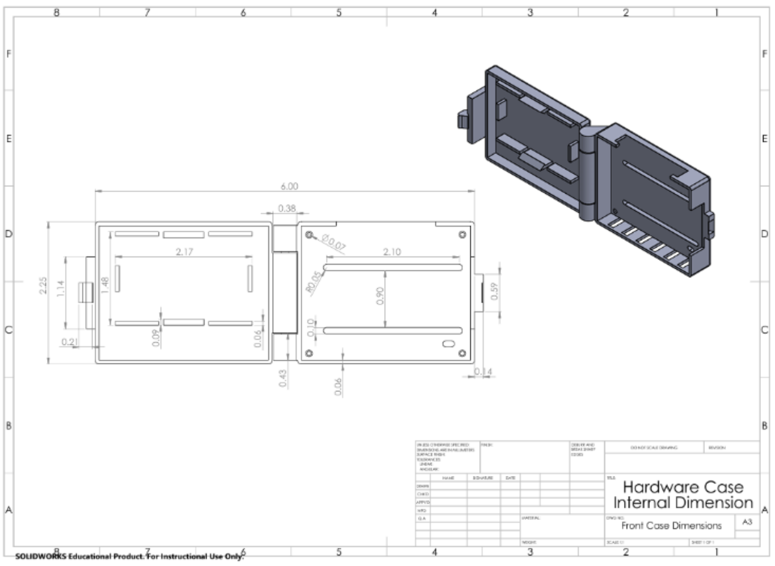


Figure 19: Final Case Open Engineering Drawing

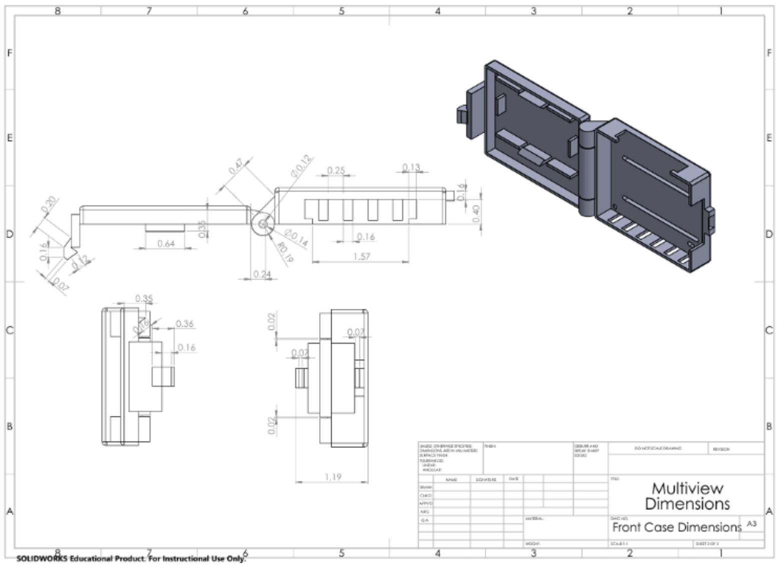


Figure 20: Final Case Side-View Engineering Drawing

Several materials were evaluated during the design process. PLA was ultimately selected because it provides excellent dimensional accuracy, rigidity for snap-fit and hinge features, minimal warping, and an overall clean surface finish. PETG offers more flexibility and higher heat resistance but introduces issues such as stringing, slower printing, and poorer tolerancing for precision features. Resin printing produces extremely detailed parts but is too brittle for living hinges and snap-fits, and the post-processing is labor-intensive. A failed resin print used in testing is shown below in Figure 21. The final PLA cases, shown in Figure 22, were printed in time for patient testing and performed reliably throughout all data-collection sessions.

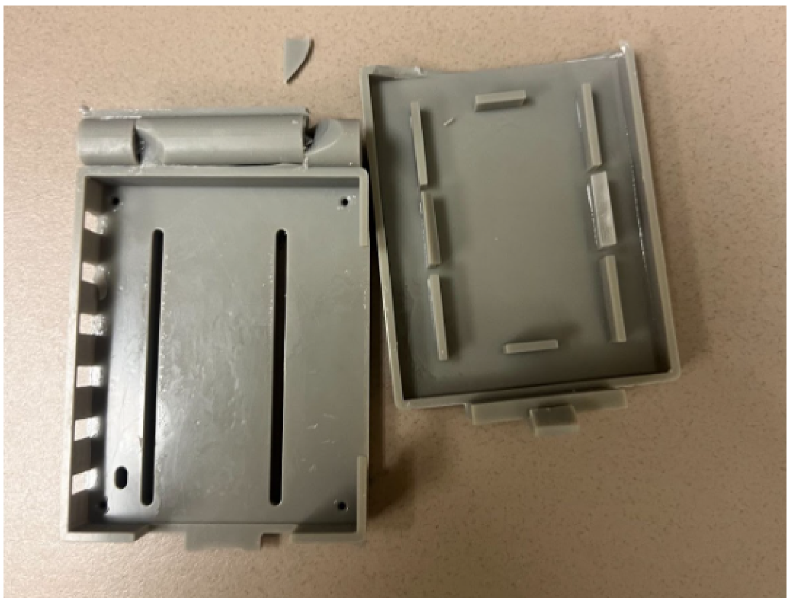


Figure 21: Resin Print Case Fail

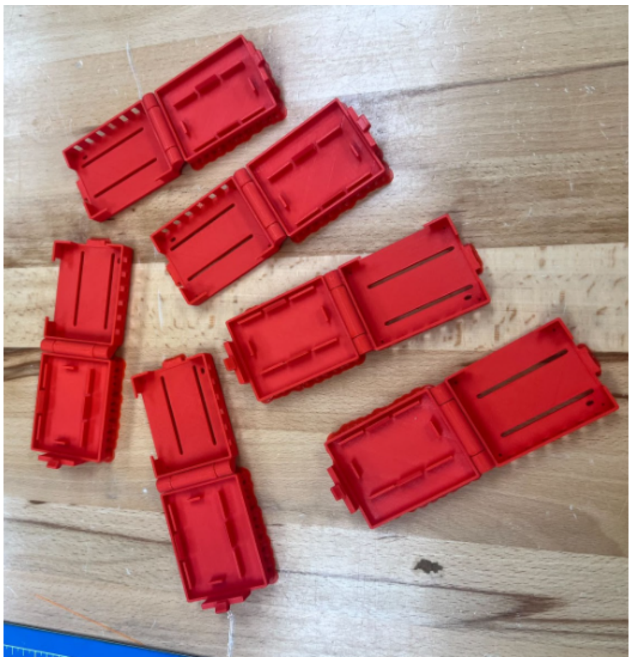


Figure 22: Final Case Design Prints

2.4 Insole Design

When looking at the physical insole subsystem, the team has made many iterations and gone through different prototypes to get to the current design. The function of the insole is to provide support and comfort for the user's foot while housing some of the electrical components, such as the IMU and the haptic disk. There were some major design choices that had to be made and scored against each other to determine how best to design the insole. When looking at the comfort of the insole, the team had to balance the thickness, curvature, shape, and hardness.

The stiffness of the insole was mostly how the team came to select a material and the infill pattern that would be used to achieve a firmer insole. The team wanted to make the insole soft enough that it would be comfortable to wear for long periods of time, but firm enough that Parkinson's patients would be able to feel it well. This firmness is necessary because Parkinson's patients have less sense of touch in their feet due to nerve degradation and need a firmer insole to be able to walk easily.

The insole was fabricated using 85A TPU with a 30% rectilinear infill pattern, selected after multiple iterations and subjective evaluations that identified this configuration as a good balance between stiffness, support, and strength. This selection process is described in more detail later in this section of the report.

The shape of the insole was made to be a pretty standard shape when looking at other insoles online, and that came in the team members' shoes. The team wanted the insole to be wearable by any standard person in any normal tennis shoe and didn't try to prioritize any specific or unique foot shape. The standard shape was created, and then this shape was kept identical for each different shoe size, using scaling techniques in SolidWorks to maintain it.

The other two aspects had to be balanced and weighed against each other to determine which criteria to prioritize and which were less important. When looking at thickness, this factor was going to impact whether or not the electrical components could be felt under the surface of the insole, and this was also going to impact the curvature. The thicker the insole, the better and more realistic the curvature could be. The team ultimately decided that minimizing thickness was the least important factor because, as long as the user's foot would fit in the shoe with the insole, the thickness would not be that significant. Also, having a thicker insole allowed for both more curvature and also allowed for more cushion above the electrical components to make sure the user couldn't feel them when walking.

Although the criteria above helped the team end up with the final design, there were some design iterations previously that the team decided to move away from for certain reasons. The first design and prototype can be seen in Figure 23 below. This design was a completely flat insole in the sense that there was no curvature outside of the shape of the insole. The top and bottom were completely flat. This decision was made because of the ease of printing and manufacturing, as there were more electrical components inside the insole at this time, which made it a more complicated design. The team left a channel for the components and wiring, so that they could be placed in after printing. An epoxy resin was poured into this channel to hold the components and wiring in place and also protect them. The main issues that the team found with this insole were the lack of curvature, hardness, and longevity. Once the resin hardened, it was very stiff and not comfortable to walk on. The contrast between the soft 3D printed material and the hard resin made walking on the insole a very uncomfortable experience for the user. The insole also acted as a mold for the resin, meaning the resin and components could be easily pulled out of the insole. Overall, this design lacked the comfort and stiffness requirements that the team was looking for, leading to changes for the next prototype.

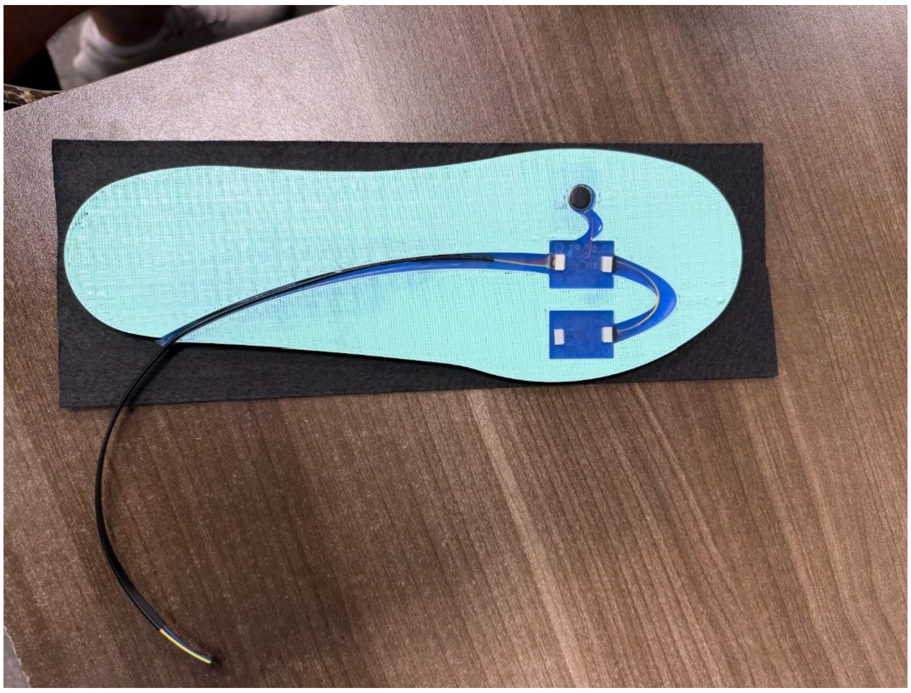


Figure 23: MEEN 401 Final Prototype

When the team came back from the summer, there were some major design changes that had to be made. The electronics were improved (discussed in detail later) to get as much as possible outside of the shoe. The flex PCB system was created, now requiring only the IMU and the haptic to be in the insole. Initially, the IMU was embedded in the upper section of the insole, but due to the redesigned PCB layout, along with a recommendation from the sponsor to move it, it was relocated to the heel. This change created new design challenges since the heel experiences higher contact pressures during walking. This led to a need for an IMU case to protect the IMU (discussed in detail in Section 2.4), and it also required some space to be included in the heel so that the IMU case could not be felt while walking. FEA was performed to get an estimate of how much space needed to be included (discussed in detail in Section 3.4), and then prototyping helped confirm that a space of .5 mm was adequate. The haptic cueing is still in its original position at the first metatarsal joint, as this was the place the team found to give the most effective feedback.

Along with moving the components around inside the insole, the team also addressed some of the other problems that were discovered from the prototype in Figure 23. The team created more curvature in the insole to improve the fit in the shoe and also improve the comfort. This was done by adding a filet to the top and bottom edges, which required the insole to be thicker to allow for a filet. Along with the curvature, the team wanted to cover up the components so that they were completely encased in TPU. This design change removed the need for the resin and would greatly increase the comfort. Originally, this was done by printing the insole in 2 parts: the bottom half that held the components and a cover to glue on top. This would make printing easier by giving each half a flat surface to print on. However, it was quickly determined that using glue would greatly reduce the lifespan of the insole, as glue would perform very poorly when it warmed up in the shoe. Glue also does not perform well in shear, which is what the insole would be experiencing when someone is walking. To fix this problem, the team decided to print the entire insole in one piece. To get the electronic components inside, the print would have to be stopped halfway through, and the components would be placed in before the print was resumed. The only problem with this was that supports were now needed during printing, making the bottom curvature slightly rough. However, the improved comfort for the user and protection for the components made this trade-off worth it in the eyes of the team. This entire process led to the final design that can be seen in Figure 24 below. Standardized engineering drawings can also be seen in Figure 25 and Figure 26 below, with Table 2 showing some of the dimensions based on the insole size.

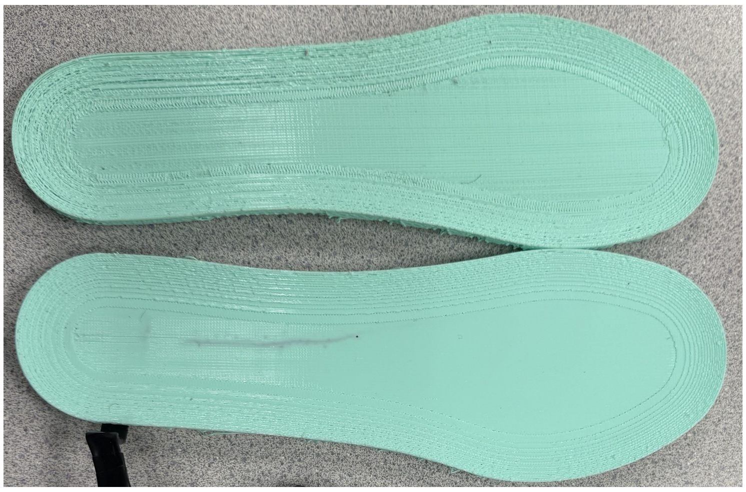


Figure 24: Final Insole Design

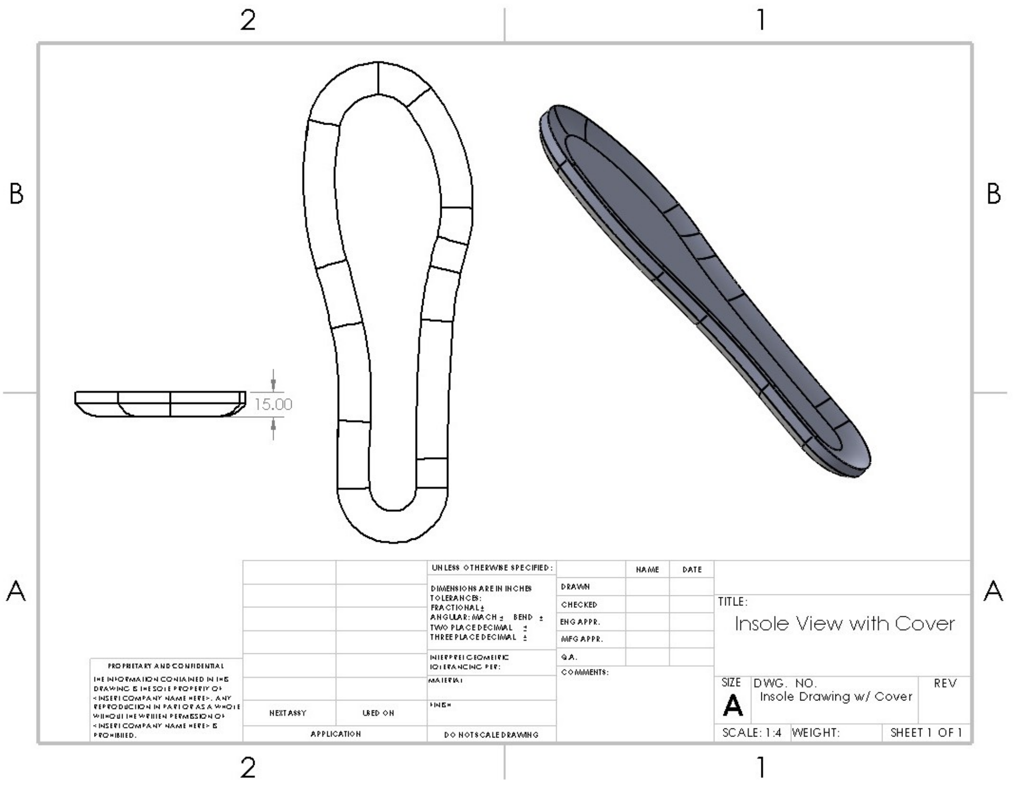


Figure 25: Insole Engineering Drawing with Cover

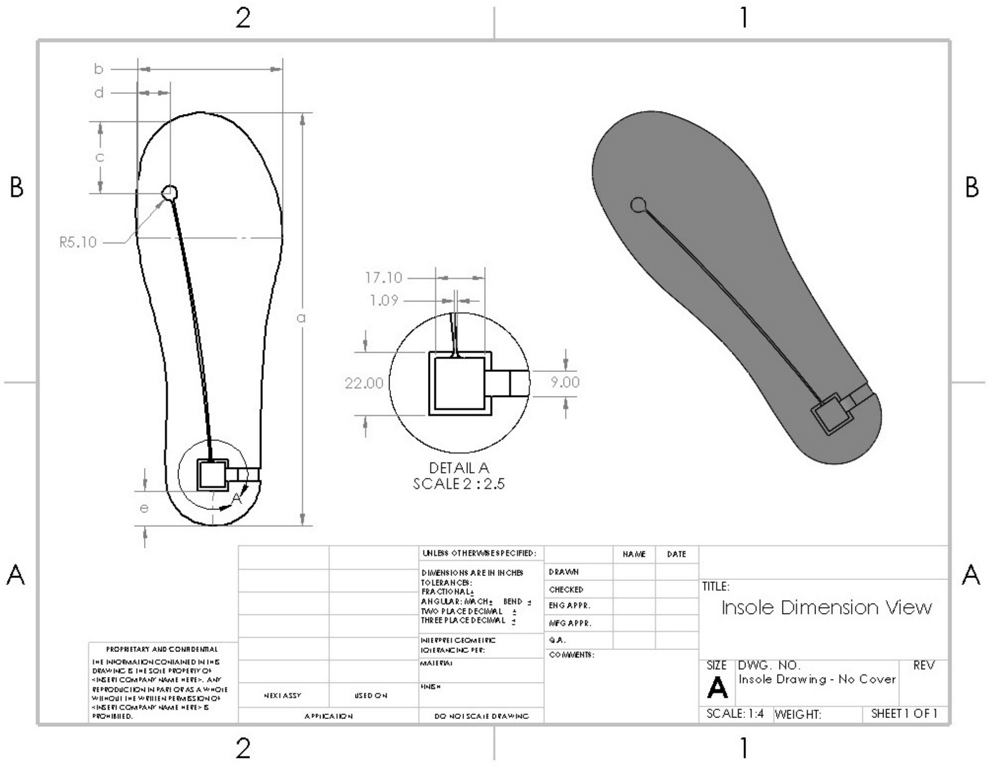


Figure 26: Insole Engineering Drawing with Dimensions (No Cover)

Table 2: Insole Dimensions Based on Shoe Size

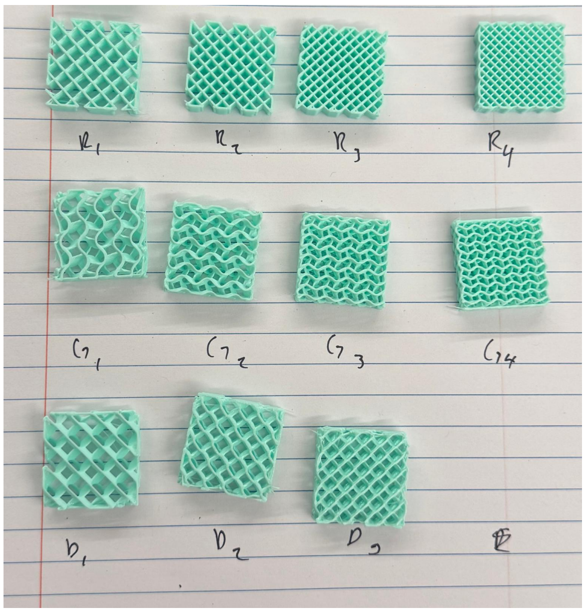
Size	**Length (a)**	**Toe box width (b)**	**Disk from toe (c)**	**Disk from side (d)**	**IMU from heel (e)**	**Cover Radius**
M5.5/W7	240.34 mm	84.9 mm	45 mm	20.5 mm	24 mm	34 mm
M6.5/W8	250.35 mm	88.45 mm	46 mm	21 mm	24.5 mm	35 mm
M8/W9	260.36 mm	92 mm	47 mm	21.5 mm	25 mm	36 mm
M9/W10	270.38 mm	95.53 mm	48 mm	22 mm	25.5 mm	37 mm
M10/W11	279.39 mm	98.71 mm	49 mm	22.5 mm	26 mm	38 mm
Mens 11	287.40 mm	101.54 mm	50 mm	23 mm	26.5 mm	39 mm
Mens 12	295.42 mm	104.38 mm	51 mm	23.5 mm	27 mm	40 mm

To evaluate the mechanical behavior of the insole material and select an appropriate stiffness for comfort and long-term use, a series of uniaxial compression tests was performed on 3D printed TPU samples. Both 85A and 95A Shore hardness TPU were tested using a universal Instron mechanical tester with a 5kN load cell at a constant displacement rate of 2.0 mm/min. All samples were printed at the same thickness as the final insole to ensure representative stiffness measurements.

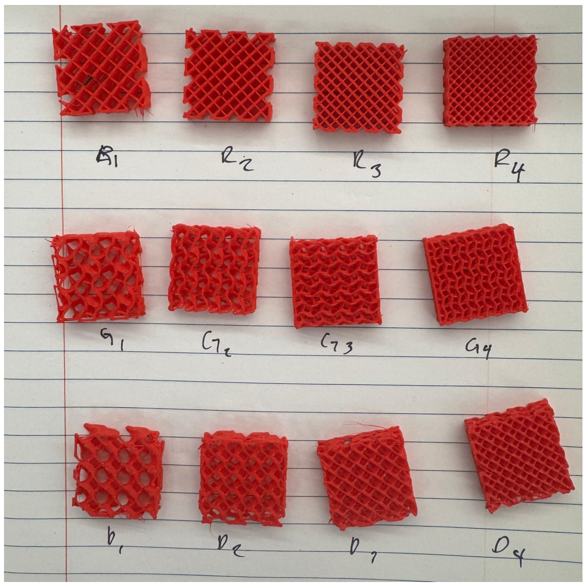
Three infill types were studied: rectilinear, gyroid, and 3D honeycomb, each printed at 15%, 20%, 25%, and 30% infill density. These patterns were selected because they represent a wide array of mechanical behaviors.

- Rectilinear provides directional stiffness.
- Gyroid has smooth stress distribution and energy absorption.
- 3D honeycomb has a high shear compliance.

For each printed sample, the compressive stress-strain response was calculated by dividing the load by the initial cross-sectional area and normalizing the displacement by the sample height. Figure 27 shows all 23 samples before they were tested.



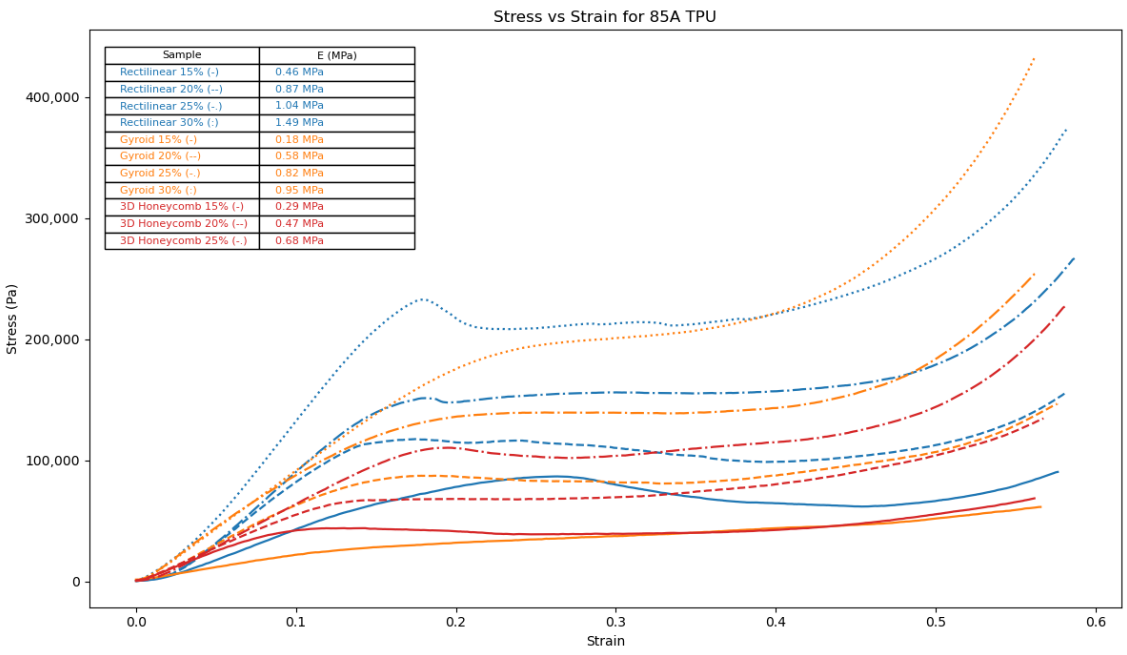
(a) 85A TPU Samples



(b) 95A TPU Samples

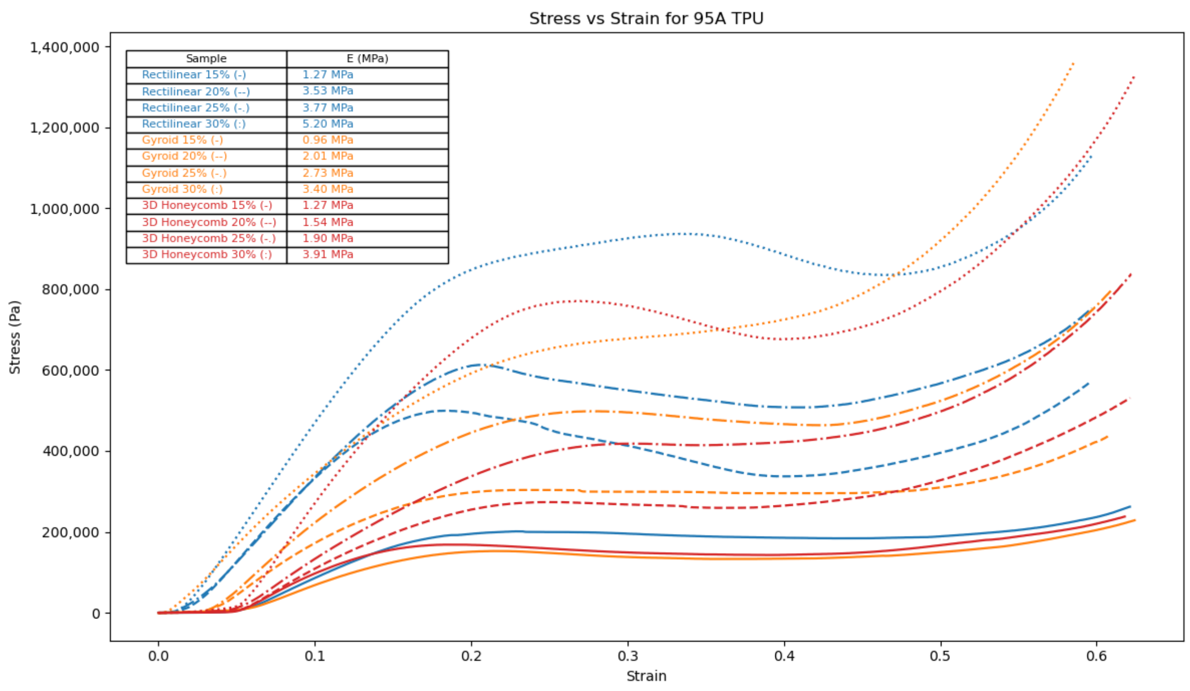
Figure 27: Printed TPU samples for Mechanical Testing: (a) 85A TPU and (b) 95A TPU with Rectilinear, Gyroid, and 3D Honeycomb Infill Patterns

To characterize stiffness, the modulus of elasticity E was calculated by fitting a linear regression to the initial linear region of the stress-strain curve, which corresponds to the sample's elastic region. The results showed that stiffness increased with infill percentage through all the patterns, which is the expected result. Rectilinear infill has the highest modulus relative to its density. Gyroid and honeycomb patterns had a more gradual stiffening and a larger strain range before densification. Figure 28 and Figure 29 show the stress-strain curves for all the tested samples.



Sample	E (MPa)
Rectilinear 15% (-)	0.46 MPa
Rectilinear 20% (--)	0.87 MPa
Rectilinear 25% (-.)	1.04 MPa
Rectilinear 30% (:)	1.49 MPa
Gyroid 15% (-)	0.18 MPa
Gyroid 20% (--)	0.58 MPa
Gyroid 25% (-.)	0.82 MPa
Gyroid 30% (:)	0.95 MPa
3D Honeycomb 15% (-)	0.29 MPa
3D Honeycomb 20% (--)	0.47 MPa
3D Honeycomb 25% (-.)	0.68 MPa

Figure 28: Measured Stress–Strain Curves for 85A TPU Across Infill Patterns



Sample	E (MPa)
Rectilinear 15% (-)	1.27 MPa
Rectilinear 20% (--)	3.53 MPa
Rectilinear 25% (-.)	3.77 MPa
Rectilinear 30% (:)	5.20 MPa
Gyroid 15% (-)	0.96 MPa
Gyroid 20% (--)	2.01 MPa
Gyroid 25% (-.)	2.73 MPa
Gyroid 30% (:)	3.40 MPa
3D Honeycomb 15% (-)	1.27 MPa
3D Honeycomb 20% (--)	1.54 MPa
3D Honeycomb 25% (-.)	1.90 MPa
3D Honeycomb 30% (:)	3.91 MPa

Figure 29: Measured Stress–Strain Curves for 95A TPU Across Infill Patterns

In Figure 29, it can be observed that 95A has a noticeable stiffness increase over 85A, but based on tested samples and comfort evaluations, the material chosen was 85A TPU with 30% rectilinear infill. It had a good balance between comfort, stiffness, and durability. This combination of pattern, infill, and material has a modulus of elasticity high enough to ensure noticeable haptic cues, while still being compliant enough for long-term usage without causing discomfort.

2.4.1 Insole Manufacturing

The manufacturing process for the insoles consists of 3D printing each insole from 85A TPU, inserting the PCB in the IMU case, and inserting the IMU into the insole. The PCB case is installed after the insole is printed. The insoles were manufactured on a BambuLab P1S 3D printer, shown in Figure 30, since it is capable of 3D printing 85A hardness TPU with minor modifications and has a print area larger than a size 12 insole. Printing 85A TPU requires a larger, non-standard 0.6 mm nozzle to extrude the filament, which was installed on the 3D printer. The Bambu Studio software was used to slice 3D design files and create Geometric Code (G-code) instructions for the 3D printer.

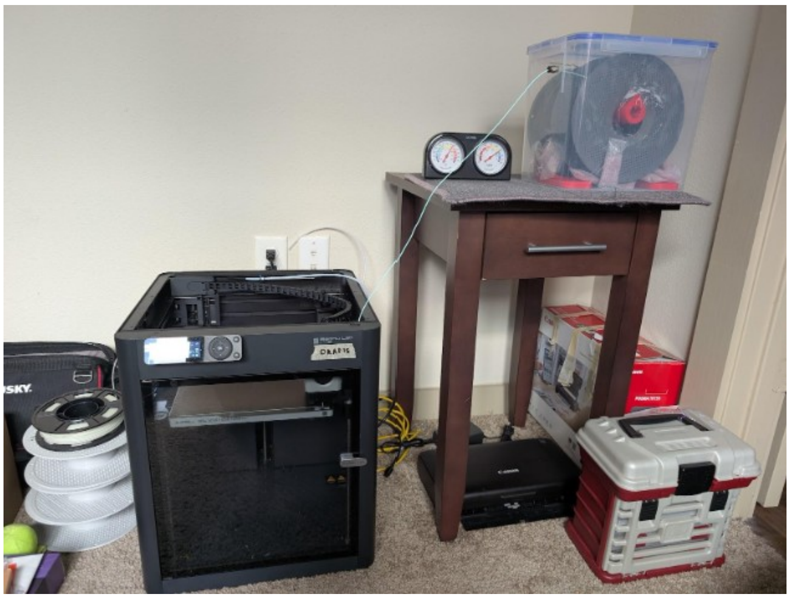
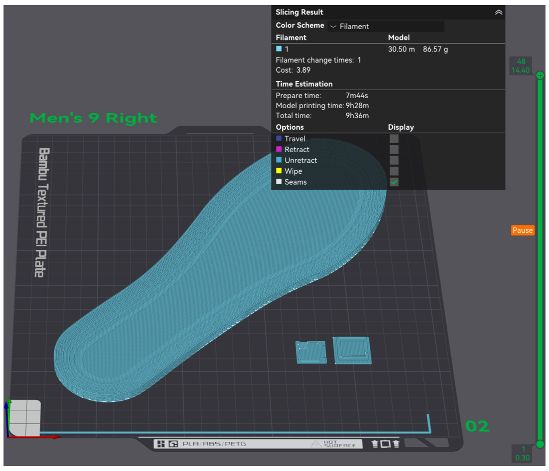


Figure 30: BambuLab P1S 3D printer configured for printing ComSole insoles

To prepare to print each insole, the design files for the insole and IMU case were exported from SolidWorks to a Stereolithography (STL) mesh file and imported into a 3D slicer, pictured in Figure 31. The infill settings were set to 30% Rectilinear, and the number of perimeters were reduced to 0 to avoid stiff spots inside the insole. The IMU case was printed with 100% infill to ensure stiffness and geometric integrity. After the file was sliced, a pause point was placed on the last layer above the IMU cavity, so the IMU and haptic motor can be placed in the insole. This print-in-place technique allows the insole to be manufactured without adhesives, since the electronic components can be placed in the insole before resuming the print as shown in Figure 32. Finally, the insole G-code can be exported to the 3D printer to manufacture the insole. Each insole takes between 7.5 and 11 hours to print depending on the insole size. ‘Dummy’ insoles without electronics were printed for the left shoes for users to have a pair of the same insoles. The team manufactured 7 pairs of insoles, 6 of them with functional PCBs shown in Figure 33.


Slicing Result
Color Scheme Filament

Filament	Model
🟦 1	30.50 m 86.57 g

Filament change times: 1
Cost: 3.89

Time Estimation
```
Prepare time:        7m44s
Model printing time: 9h28m
Total time:          9h36m
```

Options	Display
🟦 Travel	
🟪 Retract	
🟦 Unretract	
🟨 Wipe	
⬜ Seams	✓

Figure 31: Insole design file prepared for 3D printing in Bambu Studio

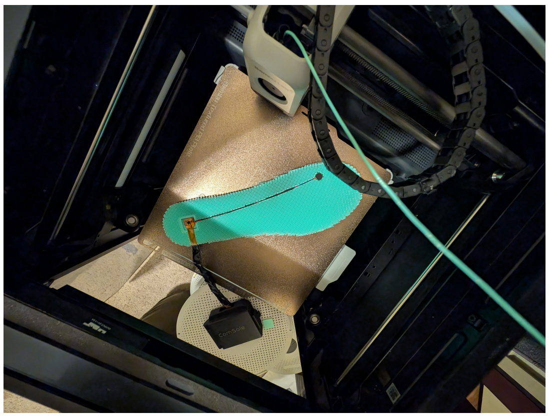


Figure 32: An insole with electronics inserted while the 3D printer code is paused

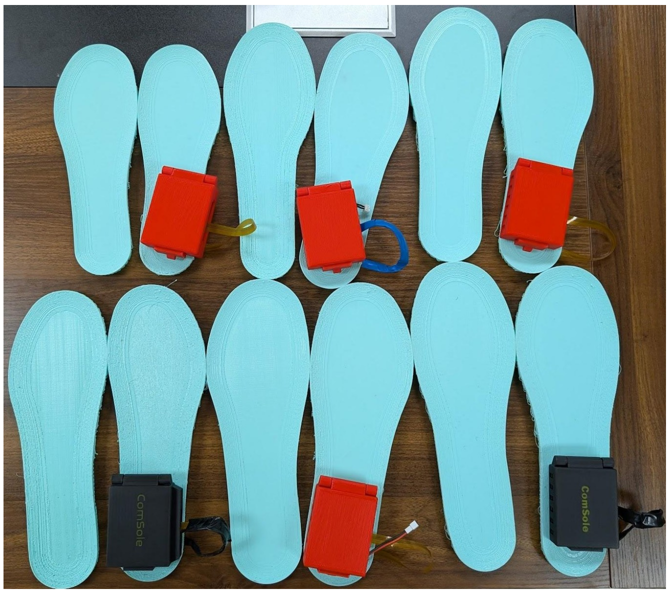


Figure 33: 6 assembled pairs of insoles

2.5 IMU Case Design

The IMU enclosure was modeled in SolidWorks and fabricated using 3D printed PLA to provide a lightweight, but rigid structure suitable for integration into the insole. The case consists of two parts bonded together with adhesive, which securely holds the PCB at the base while leaving a cutout for the IMU sensor. This configuration allows the sensor to remain flush with the insole surface without protrusion. The upper half of the case is fixed to the top layer of the insole using adhesive, ensuring that the entire assembly compresses uniformly with the surrounding insole so the user does not perceive any local hard spots. A small air gap was added beneath the case to prevent bottoming out under maximum load, which further maintains comfort during usage. Multiple iterations were created with variations in case geometry and materials to study their influence on transmissibility and vibration response. The final design prioritized comfort, reduced sensor noise, all while ensuring a 1:1 transmissibility ratio within the desired frequency range, to preserve accurate sensor readings during gait. The final case design can be seen below in Figure 34.

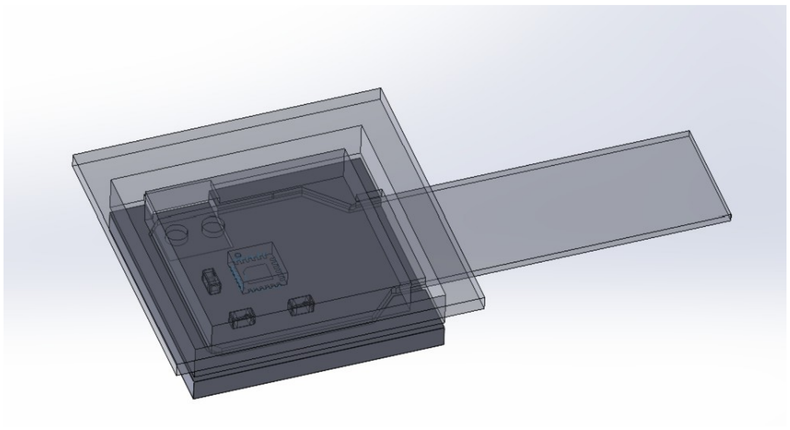


Figure 34: Current Insole Case Design

In order to ensure that the suspended IMU design would not create a noisy or dampened signal, the team constructed and evaluated a 1-DoF spring-damper model. Because the IMU is suspended in the air, the supporting structure of the PCB and case act as spring-dampers, creating a potentially oscillatory system. A visual representation of this system can be seen below in Figure 35

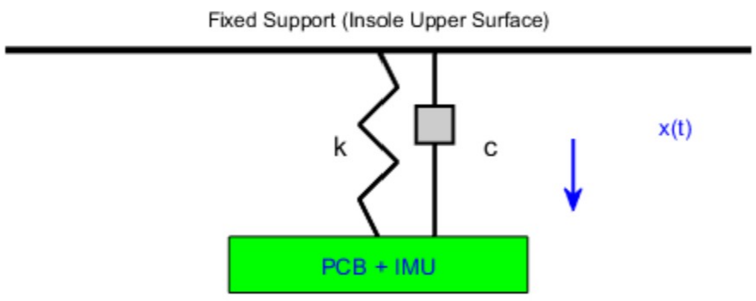


Figure 35: IMU 1-DoF Spring-Damper Model

Using this model and values determined from consulting online data sheets regarding both the PCB and TPU filament used, Figure 36 shows that at the natural walking frequency, the transmissibility of the gait is 1, and the transmissibility up to 100 HZ of noise is not greater than 1.

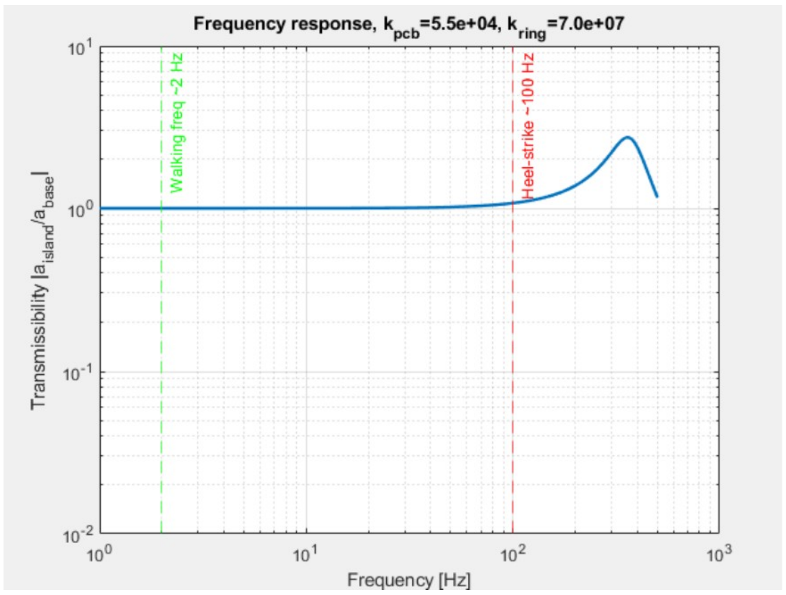


Figure 36: IMU Transmissibility Through Frequency Range

Engineering drawings for the IMU case can be found below in Figures 37 and 38.

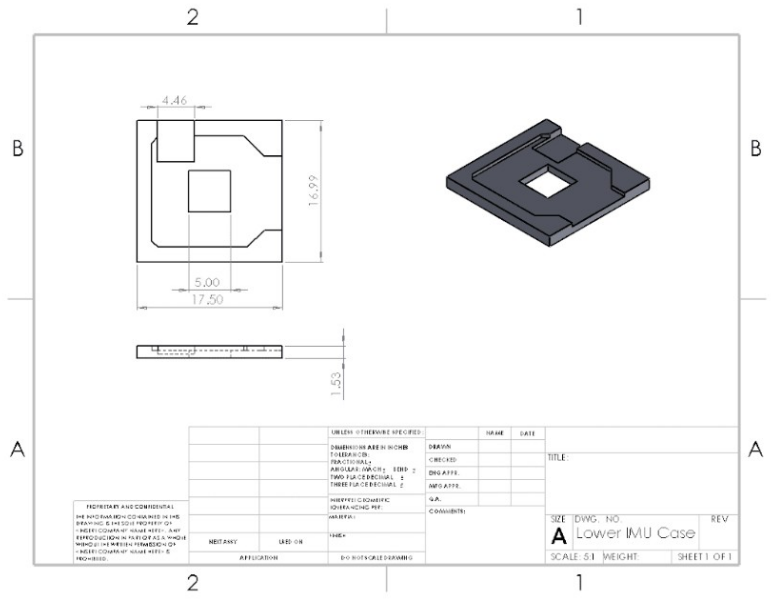

4.46
16.99
5.00
17.50
1.53
TITLE:
SIZE A DWG. NO. Lower IMU Case REV
SCALE: 5:1 WEIGHT: SHEET 1 OF 1

Figure 37: Lower IMU Case

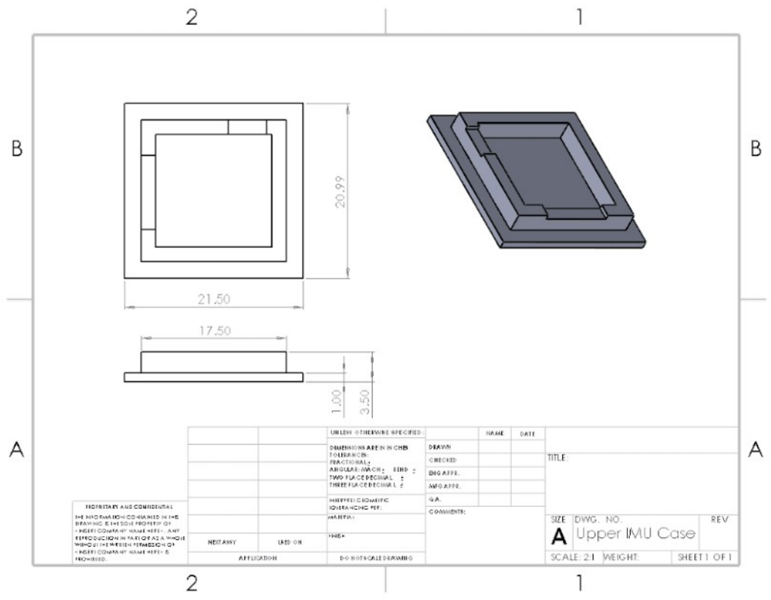

20.99
21.50
17.50
1.00
3.50
TITLE:
SIZE A DWG. NO. Upper IMU Case REV
SCALE: 2:1 WEIGHT: SHEET 1 OF 1

Figure 38: Upper IMU Case

Combining all of the previous aspects of the system that have been discussed (insole, PCB case, and IMU case), Figure 39 below shows a model of the physical mechanical system designed by the ComSole team.

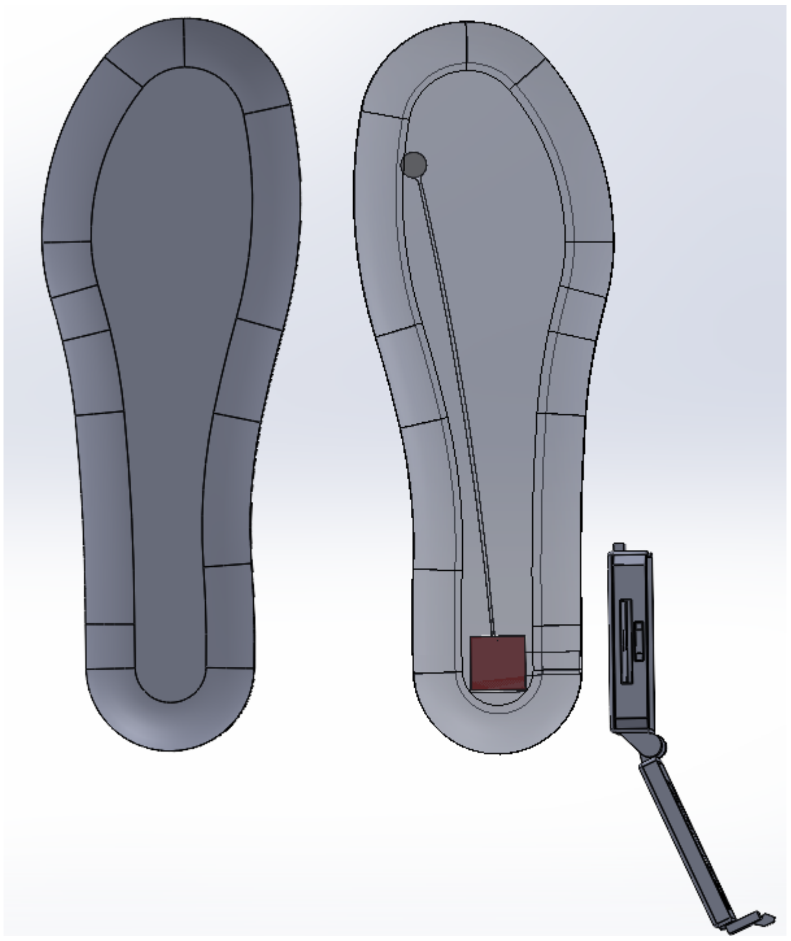


Figure 39: Entire Insole System Model

2.6 Insole Firmware

Firmware, used primarily on low-level embedded devices such as microcontrollers, was written to interface sensors and actuators in the insole with the data collection software and mobile application. The firmware for ComSole, included in Appendix C.2, runs on the ESP32 microcontroller and was written in C++ with PlatformIO. The firmware interfaces with the ICM 20948 IMU over the I^2C communication protocol using the vendor's library [4], and the haptic vibration motor with a digital output. The ESP32 microcontroller supports Bluetooth Low Energy (BLE), which the team chose to transfer data from the insole to peripheral devices. The BLE protocol was implemented using *NimBLE-Arduino* [5], a Bluetooth software package based on *NimBLE* for Arduino that runs on microcontrollers with PlatformIO. The machine learning model in Section 2.7 was implemented by exporting the model weights to a C language header file and re-implementing the Random Forest Classifier in C. The flowchart in Figure 40 shows the structure of the firmware code.

The ICM 20948 IMU is a 9-degree-of-freedom IMU, which measures each linear acceleration, angular velocity, and magnetic field intensity in three dimensions. The sensor measurements are read from the IMU using I^2C every 10 ms (100 Hz). The IMU data is processed and repackaged as a character string to send over BLE with *NimBLE-Arduino*. The data collection and Bluetooth communication are performed at the same frequency in the main loop of the firmware code. On startup, the firmware advertises itself with the name provided in the firmware configuration file in Appendix C.1. On each pass of the main loop, the firmware ensures there is a Bluetooth connection with a peripheral device. If no devices are connected, it does not send data over BLE to save energy.

The embedded machine learning model is also run in the main loop at 100 Hz. To use the machine learning model, the firmware first generates features to be input to the model, detailed in Section 2.7.2. The features are generated from a window of data, recorded from the IMU over time, and stored in an array. The generated features are input to a prediction function that produces a probability of whether FoG has occurred. If the probability is greater than a threshold, then the haptic vibration motor is activated.

The haptic motor is driven by battery power through a transistor, which is activated by a digital output from the ESP32 microcontroller. When the digital output is high, the transistor applies the battery voltage to the haptic vibration motor, and the disc vibrates. This cue can be felt by the user.

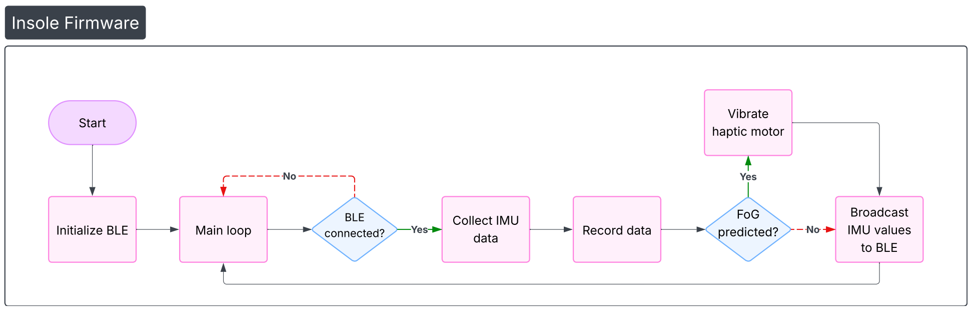


Figure 40: Insole Firmware Flowchart

2.7 Machine Learning

Machine learning is needed for the ComSole system because it allows the device to automatically recognize patterns in a patient's gait that can be classified as FoG events. Instead of hard-coding rules to classify these discrepancies, a model is trained to learn these patterns directly from labeled data. This section describes the full pipeline, from how the data are segmented and labeled to how the final random forest classifier was trained, tuned, evaluated, and deployed on the insole microcontroller.

2.7.1 Conceptual Background

Supervised learning and classification: In supervised learning, the developer provides the algorithm with pairs of inputs and outputs and asks it to learn a function that maps one to the other. In the case of ComSole:

- Inputs: Statistics of IMU signals (linear acceleration and angular velocity over a short time window).
- Output: A binary label that indicates whether that time window is an FoG event (1) or normal gait (0).

Because the output is a discrete class rather than a continuous number, this is a classification problem where the goal is to learn a decision rule that predicts the correct class for new and unseen walking data.

If the relationship between input and output were simple and/or known, such as a simple acceleration threshold, then a simple rule would be enough to detect FoG. However, FoG events depend on a combination of:

- Magnitudes of acceleration and angular velocity in multiple directions,
- Variability and rhythm of the gait signal,
- Irregular patterns.

These relationships are complex, non-linear, and distinct between patients. Machine learning allows the product to approximate this unknown mapping without having to write a threshold for every possible pattern.

2.7.2 Data Segmentation and Labeling

Sliding window segmentation: The raw IMU signals are sampled at 100 Hz. Instead of classifying each individual sample, short windows of data are analyzed. ComSole uses:

- Window length: 0.5 s (50 samples),
- Step size: 0.125 s (12 or 13 samples with 75% overlap).

Each window overlaps substantially with the previous one, which allows the model to precisely capture the events over time while still having enough samples to obtain important statistics. Figure 41 shows how consecutive overlapping windows slide over the signal.

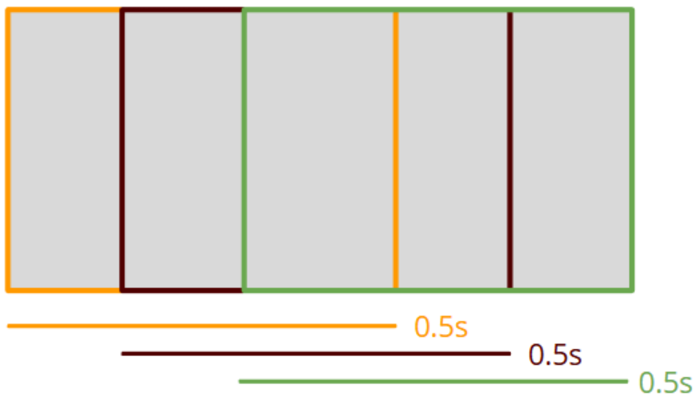


Figure 41: Illustration of sliding-window segmentation. Each colored bar represents a 0.5 s window. Windows are shifted by 0.125 s.

Window labeling: The raw dataset has an FoG label for each sample, which was annotated with the ComSole GUI. However, the model uses windows, so each window must be assigned a label. In the code, it was implemented as: `int(w["FoG"].mean() >= 0.3)`. This means that if 30% or more of the data points in the window are labeled as FoG, then the window is also labeled as FoG. This makes the model sensitive to windows that contain a significant amount of FoG activity while avoiding labeling windows as FoG do to an isolated mislabeled sample.

Feature engineering: Each 0.5 s window is converted into a vector of 48 features. These features characterize the signal and are computationally inexpensive so that they can also be computed in the insole microcontroller. For each of the six IMU channels {acc_x, acc_y, acc_z, gyr_x, gyr_y, gyr_z}, the following calculations are performed:

- **Mean**: average value over the window.
- **Standard deviation** (σ): how much the signal varies.
- **Minimum and maximum**: smallest and largest sample in the window.
- **Root-mean-square (RMS)**: which is RMS $= \sqrt{\frac{1}{N}\sum x_i^2}$
- **Energy**: $\frac{1}{N}\sum x_i^2$, the total 'strength' of the signal, proportional to RMS^2.
- **Zero-crossing rate (ZCR)**: Number of times the signal crosses its mean between consecutive samples.

To capture the correlation between axes, six correlation features are added. Since correlation measures how closely two signals follow one another over time, it shows whether motion on one axis tends to increase or decrease in sync with motion on another.

- corr(acc_x,acc_y), corr(acc_x,acc_z), corr(acc_z,acc_y)
- corr(gyr_x,gyr_y), corr(gyr_x,gyr_z), corr(gyr_z,gyr_y)

This is a total of 48 features per window. Before training, each feature is standardized using a `StandardScaler`, which subtracts the mean and divides by the standard deviation, to make them comparable in scale.

2.7.3 Model Choice: Random Forest Classifier

Decision trees: A decision tree classifier predicts a class by asking a sequence of yes/no questions about the feature values, where each internal node checks a condition such as gyr_x_zcr ≤ 0.12. Depending on the answer, the sample follows the right or left branch until reaching a leaf node, which stores the predicted class (FoG or no FoG) or a class probability. Figure 42 shows a simple decision tree with 2 conditions.

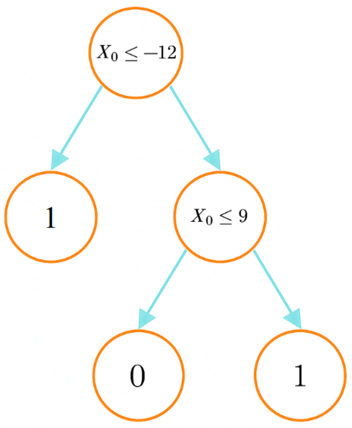


Figure 42: Decision Tree Example with Two Conditions

Decision trees are easy to interpret; however, a single tree is susceptible to overfitting training data.

Random Forest: A random forest is a collection of many decision trees, where each tree is trained on a slightly different subset of data and features. At prediction time:

1. Each tree outputs a probability that a window is FoG.
2. The forest averages these probabilities of all trees.
3. The final probability goes through a threshold to get a binary prediction.

Some of the advantages of this approach are:

- It can model complex, non-linear relationships between IMU features and FoG.
- It is robust to noise and outliers because many trees are averaged.
- It can have a low computational footprint.

2.7.4 Training and Hyperparameter Tuning

The data collected was divided into 3 sets:

- 60% for training
- 20% for validation (Used to tune the probability threshold and hyperparameters)
- 20% for testing (used only for the final performance report).

Data was split on a per-subject basis, to ensure that there was no data leakage between data sets to so that the model would generalize well between different users and not overfit to unique patterns.

Handling class imbalance FoG windows are much rarer than non-FoG windows. By default, a classifier could achieve a high "accuracy" by predicting no FoG almost all the time. To counteract this, the random forest was trained with `class_weight="balanced"`, which re-weights miss-classification penalties so that errors on FoG windows have a stronger contribution to the loss than errors on the more common non-FoG windows.

Grid search with cross-validation (GridSearchCV): The random forest has several hyperparameters that affect its performance:

- `n_estimators`: number of trees in the forest. More trees usually improve performance but increase computational effort.
- `max_depth`: maximum depth of each tree. Having a set limit can help prevent overfitting.
- `min_samples_split`: minimum number of training samples required to split a node.
- `min_samples_leaf`: minimum number of samples in a leaf node.

`GridSearchCV` was used to iterate through a combination of these parameters. For each combination, the training set was split internally into several folds using `StratifiedShuffleSplit` (which preserves the ratio of FoG to non-FoG windows in each fold). The model was trained on part of the data and evaluated on the remaining part. This was repeated across folds, and the combination with the best mean F1 score was selected. The F1 score represents model performance by combining precision and recall into a single metric. It rewards models that correctly detect FoG while minimizing both false positives and false negatives.

The final model used 1200 trees, with no maximum tree depth, a minimum number of training samples of 2, and a minimum number of samples in a leaf of 1.

2.7.5 Threshold Selection and Evaluation Metrics

The random forest outputs a probability p that a window contains FoG. To get a binary decision, a threshold T is selected. By default $T = 0.5$, but that is not necessarily optimal, especially because the classes are imbalanced. To fix this, the threshold T was swept from 0.05 to 0.95 on the validation set, and the F1 score was computed for each value. Figure 43 shows this curve. The F1 score peaks at around $T = 0.40$, so that was selected as the operating threshold for both the test set and the embedded model.

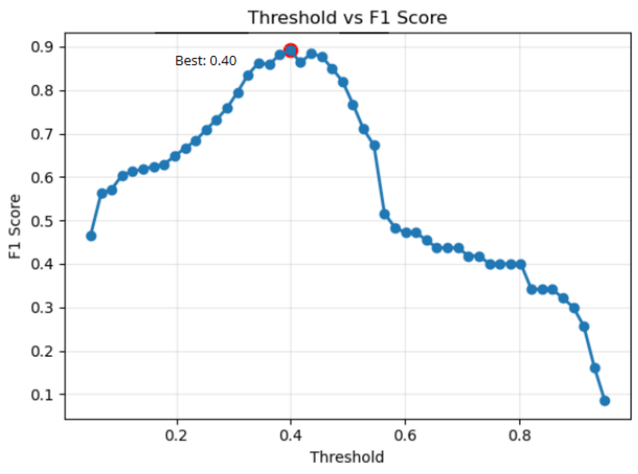


Figure 43: F1 Score as a Function of Classification Threshold

Confusion matrix and performance metrics: Predictions on the test set are summarized with a confusion matrix seen in Figure 44, where:

- True negatives (Upper-left) [TN]: correctly predicted non-FoG windows.
- False positives (Upper-right) [FP]: predicted FoG but actual is non-FoG.
- False negatives (Lower-left) [FN]: predicted non-FoG but actual is FoG.
- True positives (Lower-right) [TP]: correctly predicted FoG windows.

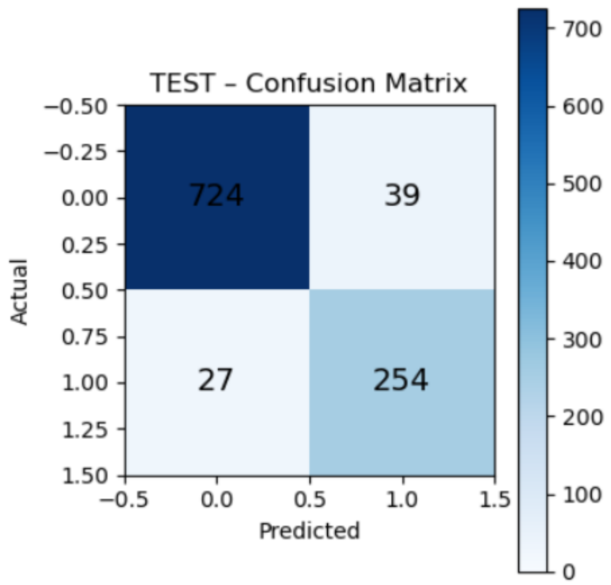


Figure 44: Confusion Matrix on the Test Set

From these, other metrics can be calculated to evaluate model performance:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Figure 45 shows the overall test results.

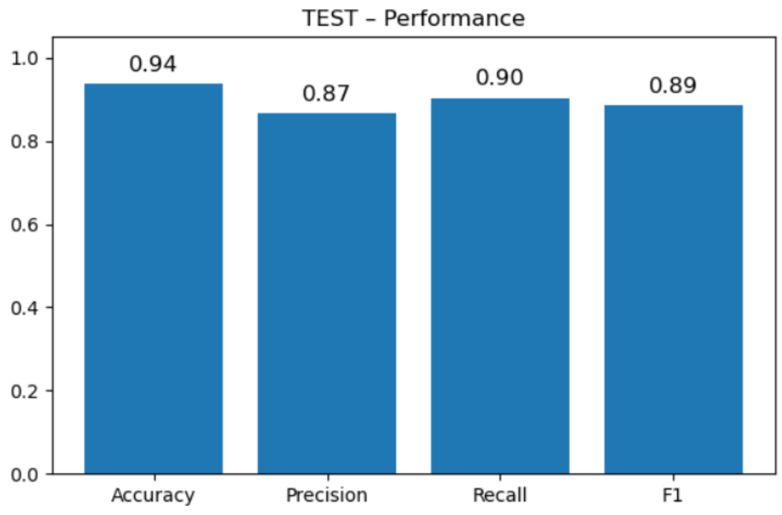


Figure 45: Overall Test Set Performance of the Model

To verify that the model behaves correctly over time, predictions were plotted on top of the labeled data for a test walking trial. Figure 46 shows an example timeline where the predicted FoG episodes (orange dashed line) align closely with the true FoG labels (blue solid line). Minor timing offsets are expected due to windowing and the 0.5s duration of each window.

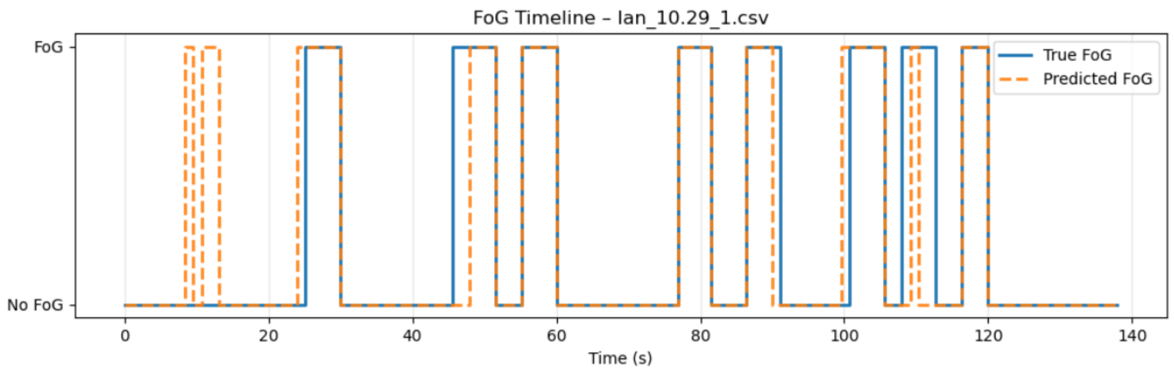


Figure 46: Example FoG Timeline for One Test Recording

2.7.6 Model Interpretability and Feature Importance:

One benefit of random forests is that it outputs how important each feature is for the final decision. Figure 47 shows the top features ranked by importance.

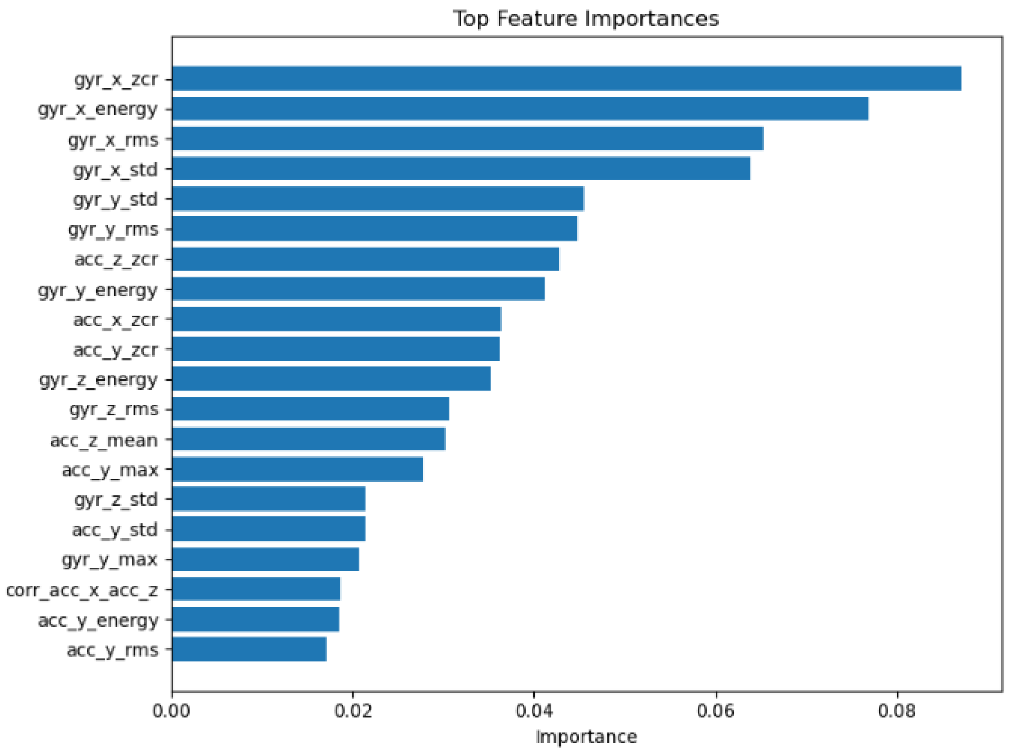


Figure 47: Top Feature Importance from the Trained Random Forest Model

Several observations align with the original prediction of linear and angular irregularities in shuffling:

- Gyroscope features, especially `gyr_x_zcr`, `gyr_x_energy`, and `gyr_x_rms`, dominate the top of the list. These capture the quick oscillatory motion in foot rotation, which is characteristic of a shuffling event.

- Zero-crossing rate and energy features are important across axes, which shows that irregular, high-frequency components in the signal are strong indicators of FoG.

This alignment between the model's features and observed characteristics of shuffling in FoG increases confidence that the classifier is learning meaningful patterns instead of just noise.

2.8 Data Collection Software

Data collection software was developed by the team to receive, record, label, and display data from the insole during testing. The ComSole data collection software in Appendix C.3 was written in Python and collects data from the insole using BLE, records the data, and displays it in a live graph on a Graphical User Interface (GUI). The data can be labeled in real time with buttons on the GUI for marking timestamps for the start and stop of features, including FoG and walking direction. These labeling buttons were used to label data while recording data on patients at Rock Steady Boxing.

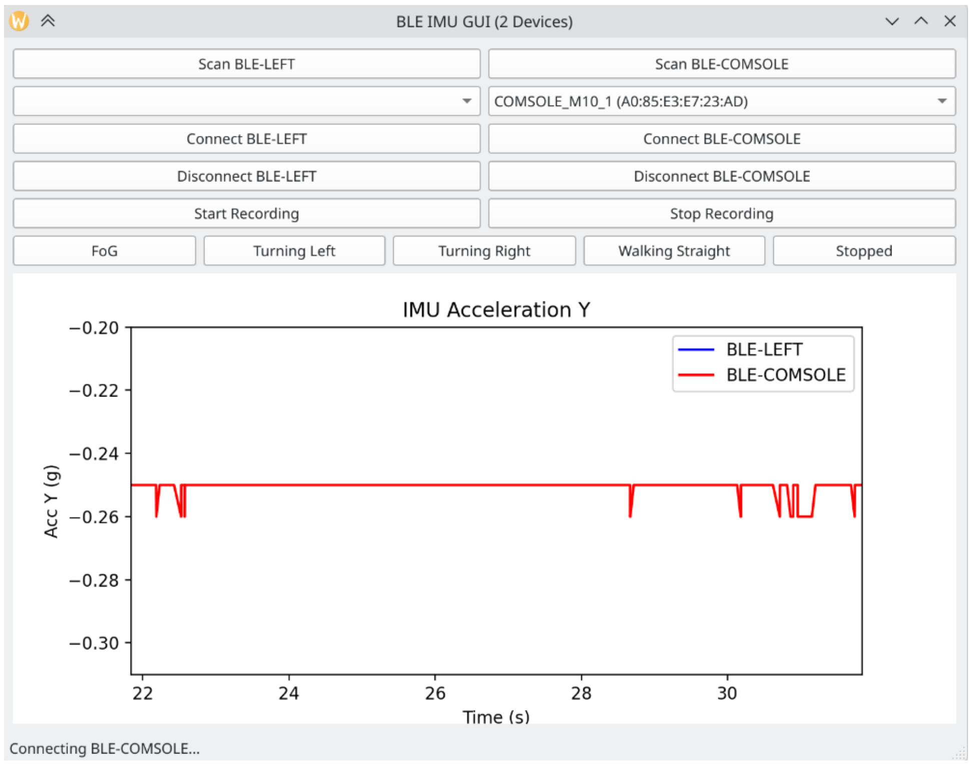


Figure 48: Graphical User Interface for Data Collection

The GUI, pictured in Figure 48, is split into sections. The left and right button sets are for connecting to a left and right insole. Since ComSole has electronics in only the right insole, the right side interface is used. The interface has a scan button, a device selection drop-down, and connection/disconnection buttons for interacting with the Bluetooth device. The user must first scan for the device to populate the device selection menu, select the correct device, and connect to it. The recording button can be clicked to begin and end recording timestamped IMU data, saving the data as a CSV when the recording is finished. The bottom set of buttons is used for labeling data, and the labels are reflected in the data as separate features for each timestamp. A graph is shown with live acceleration data from the IMU, displaying the Y-axis acceleration, which measures the forward and backward motion of the foot. At the bottom is a status bar to indicate the Bluetooth status.

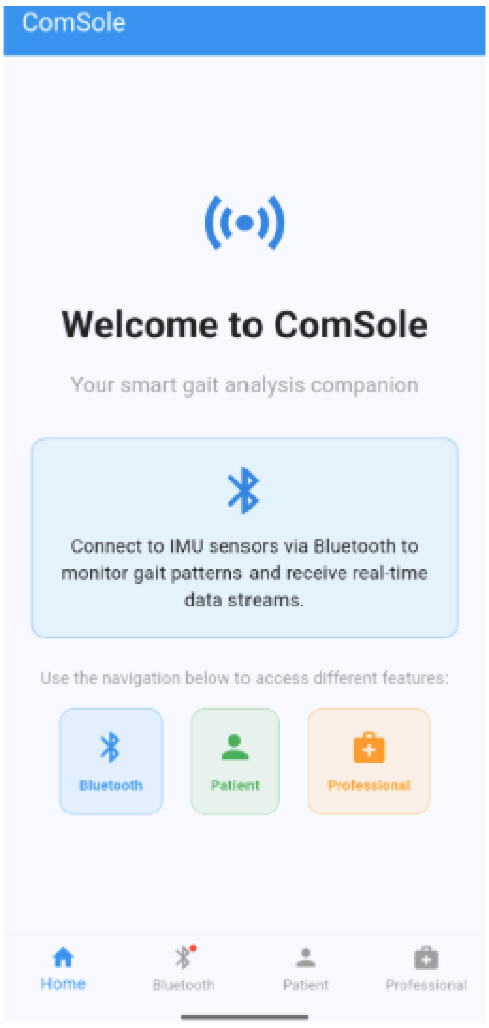
2.9 App Design

In addition to the physical insole product, the team determined that a companion mobile application would enhance the overall usability of the product for elderly patients and their caregivers.

The mobile app was designed in VS Code using an extension called Flutter. Flutter is an app development extension that wraps numerous packages together in order to simplify the app development process. By using Flutter, the team has been able to design an app that is ready for deployment on both iOS and Android mobile devices, that is capable of connecting to the insole via Bluetooth, receiving live gait data, processing and displaying said data, and storing it for future reference. In order to accomplish these functionalities, various software packages were used. Primarily, flutter_ble_plus was used to create and maintain the Bluetooth-low-energy connection used for data transfer, and SQLite was used to create the locally hosted database used for long-term data storage.

Because the app is intended for a primarily elderly audience, the design was focused on keeping the app as simple and intuitive to use as possible, while still adding value and usability to the overall ComSole product. With this motivation in mind, the app was designed with 4 tabs meant to separate key information as intuitively as possible into Home, Bluetooth, Patient, and Professional pages.

Figure 49 below shows the Home page, which is the page that opens when the app is first opened.


ComSole

Welcome to ComSole

Your smart gait analysis companion

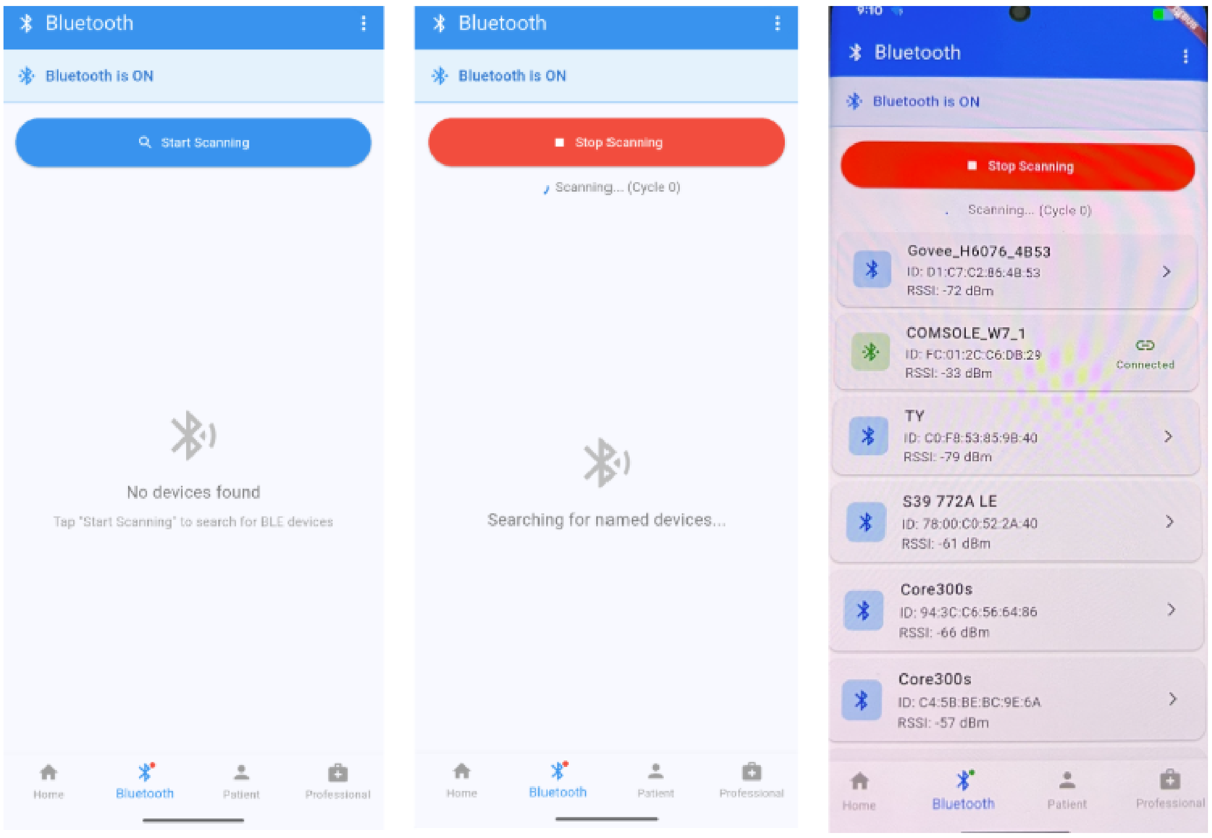
Connect to IMU sensors via Bluetooth to monitor gait patterns and receive real-time data streams.

Use the navigation below to access different features:

Figure 49: App Home Page

The home page contains a welcome statement, a brief explanation of its use, and instructions to use any of the navigation buttons below to utilize the different features. At the bottom of the screen, a main navigation bar is always accessible to change between the various screens.

Figure 50 below shows the Bluetooth page.



Govee_H6076_4B53
ID: D1:C7:C2:85:4B:53
RSSI: -72 dBm

COMSOLE_W7_1
ID: FC:01:2C:C6:DB:29
RSSI: -33 dBm
Connected

TY
ID: C0:F8:53:85:9B:40
RSSI: -79 dBm

S39 772A LE
ID: 78:00:C0:52:2A:40
RSSI: -61 dBm

Core300s
ID: 94:3C:C6:56:64:86
RSSI: -66 dBm

Core300s
ID: C4:5B:BE:BC:9E:6A
RSSI: -57 dBm

Figure 50: App Bluetooth Page

When first opened, the Bluetooth page has only one inter-actable: the start scanning button. Additionally, there is an icon with instructions in the middle of the screen to further suggest to users to use the start scanning feature. Once clicked, the app autonomously finds all named Bluetooth devices in the area and displays them in a list. Once the user's named device appears, all they need to do is click on it, and the connection is established. This process happens very quickly, typically locating the device within a second, and shows the device towards the top of the list since they appear in order located. Once connected, the icon turns green and states connected, providing further reinforcement that a positive connection has been established.

Figure 51 below shows the patient page.

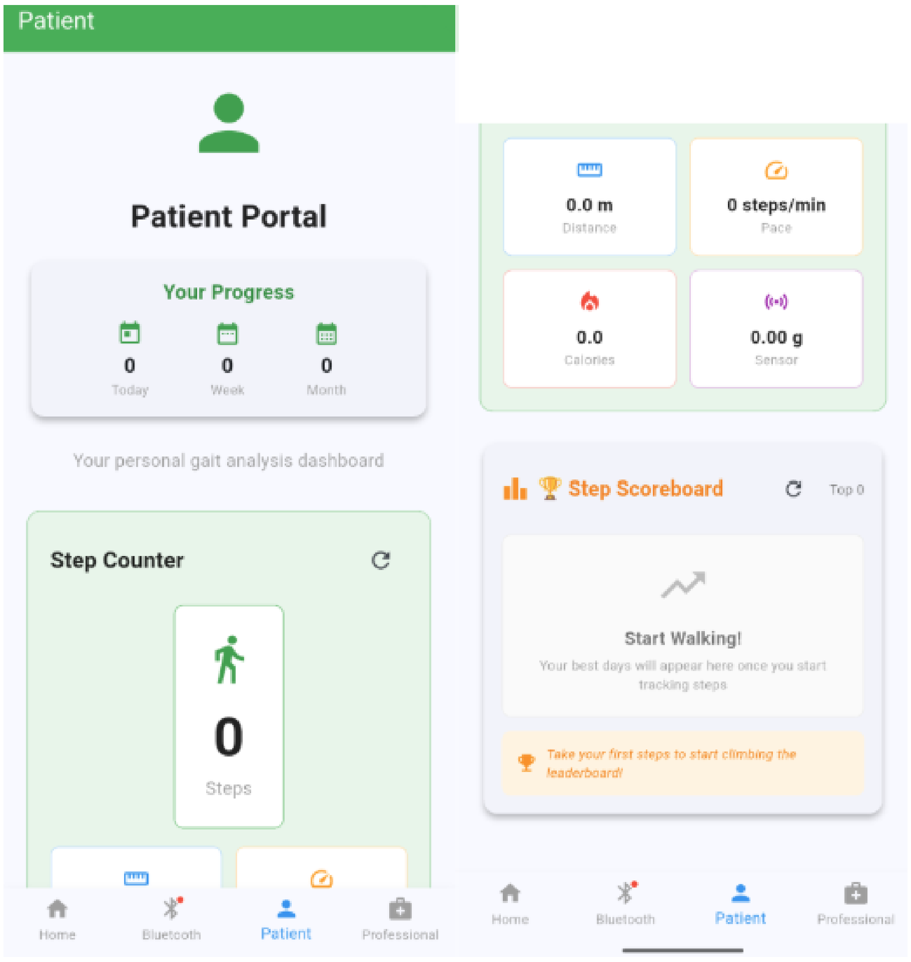


Figure 51: App Patient Page

Once connected to the insole, patients should move into the patient portal. This page is meant to show some simple gait metrics to give patients a general overview of their gait health, with some simple to to-understand metrics. First, the "Your Progress" widget. It shows users daily, weekly, and monthly step totals, so it's helpful in keeping track of short-to-medium term walking health. These numbers update as you walk around, and can only be reset if the app is deleted from the device. Next, the Step Counter card. This card shows the current connection, not the daily total, steps, distance, pace, calories burnt, and g-force. The specifics of how each of these metrics is calculated will be discussed shortly. The final part of the Patient page is the Step Scoreboard. This shows the user's top 5 highest step days, with both the date and the step count. The hope is that this card will motivate users to continue to move even on difficult days as the disease progresses.

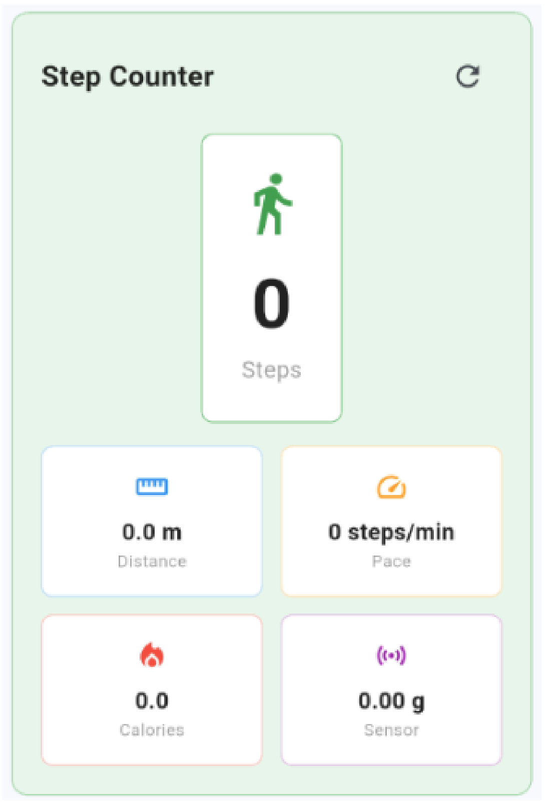


Figure 52: Step Counter Card

In the step counter card, seen in Figure 52, the main feature is counting the number of steps a person has taken. Using the data transferred from the insole, the app determines if a step has occurred if and only if two conditions are met: the magnitude of the insole's acceleration has reached 1.1 g, and at least 0.3 seconds have passed since the last step was recorded. Using a minimum acceleration value allows the app to determine when enough movement happens fast enough to likely be a step. The value of 1.1 g was determined experimentally, and it works as a filter for low acceleration movements such as shuffling feet, and is low enough to register slow steps from elderly patients. Furthermore, the 1.1 g acceleration is found using a 5-point moving average to eliminate any erroneous values that may be detected by the IMU. The 0.3-second time difference between steps is also a means to prevent non-steps from being registered from actions like rapid heel tapping. The time difference is, however, small enough that even very fast walking is fully captured by the app.

Distance is calculated using double acceleration integration, meaning that the acceleration data captured by the insole is multiplied by the time steps in between data points to find velocity, and that process is repeated to find distance traveled.

Pace is calculated by dividing the total number of steps during the connection by the time since the first recorded step occurred. This means that the pace is not instantaneous or over a set time frame, but the average since the first recorded step. Practically, this means that if the connection is established while stationary, say sitting at a desk, the user could continue to sit, and the pace would not begin being calculated until the wearer got up and began to walk around.

Calories are calculated by dividing the number of steps taken by 20. This is a conversion factor that was determined from investigating various online sources, and it is approximately the average value referenced online. Calories burned are a very difficult parameter to calculate since they depend on so many factors, such as age, height, weight, and gender, among other factors, but this provides a rough estimate for users to have some idea of how their movement directly affects their energy usage.

G-force is the most simply calculated metric, found by taking the magnitude of the accelerometer data. This parameter doesn't directly provide information about the patient's gait health, but by monitoring it over time, it can provide information on how their gait health changes. Since the IMU measures acceleration in all directions, the magnitude tells users how quickly their foot is moving in total. If their gait pace and distance covered stay the same, but the g-force magnitude decreases, this can inform users that they are no longer lifting their foot as high when walking. This is because an unchanging frequency and distance mean that their forward motion has not changed, but the decrease in magnitude means their motion in another direction has changed.

Figure 53 below shows the professional page. The professional page is meant to house data useful to users' doctors. Because the ComSole device is able to collect and transmit patient gait data throughout their everyday lives, the app has the capability to be a tool for doctors to gain a better understanding of their patient's walking health.

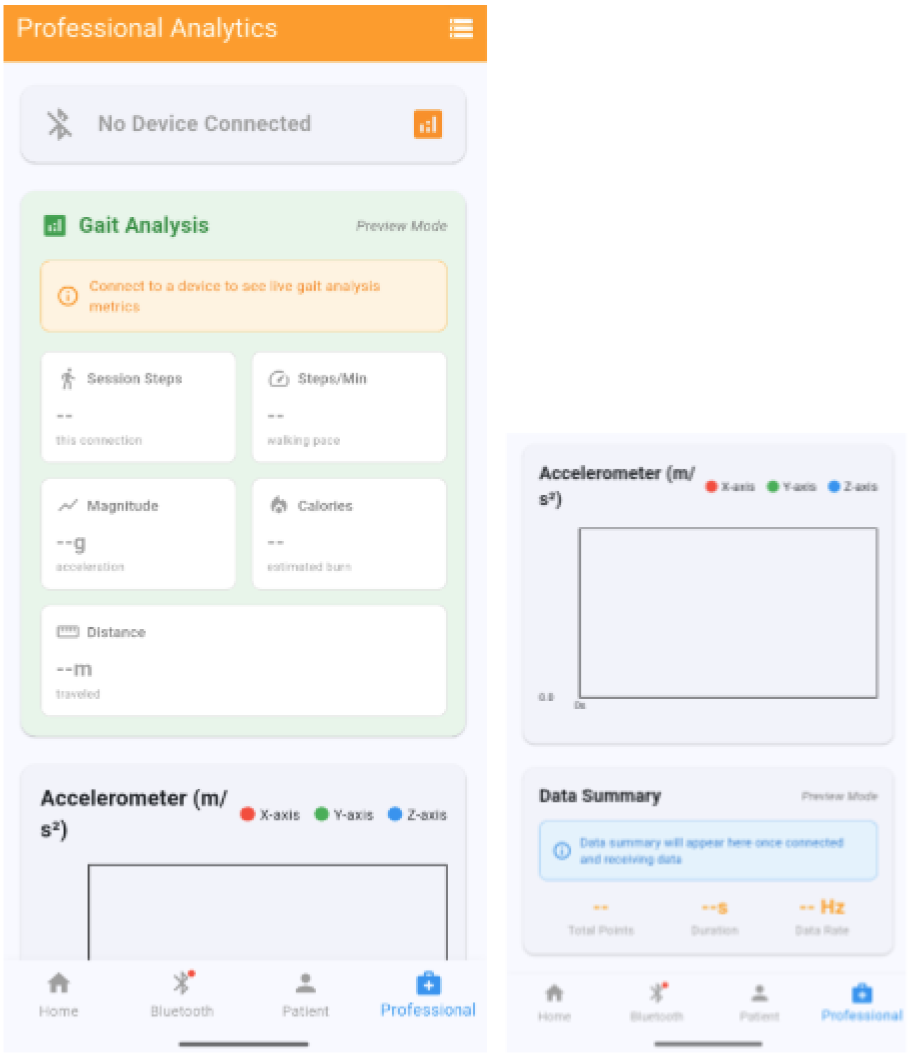


Figure 53: App Professional Page

Due to time constraints, the professional page shows all of the same information as the patient page in the Gait Analysis card. This is the primary point of future development suggestions in the app to be discussed later in this report. The professional page also shows a live data chart of the accelerometer data and a summary of the data stream. These items are useful in ensuring that the app is properly connected to the insole and that all data is being received.

3 Analysis for Design

3.1 Embodiment Design Checklist, Risk Analysis, and Standards

3.1.1 Embodiment Design Checklist

All customer needs identified through customer surveys, needs assessments, and brainstorming sessions are satisfied with the current design. Using the target metrics developed in conjunction with the project sponsor, the current design meets all targeted values.

The current architecture is capable of collecting accurate and robust enough information for the analysis required to be successful, and the moat design implemented minimizes any vibrational noise that may exist. The flexibility of the insole material will allow the haptic motor to vibrate without creating noise in the IMU's measurements as well.

The chosen 3D printing filament provides the ideal combination of material properties for this application. The soft 85A filament provides flexibility to bend and maintain optimal performance in various walking scenarios. The chosen print structure provides high strength, hardness, and stiffness in the primary loading direction, providing the support for patients and the protection of electronics required. Because the product is made of plastic, it is lightweight and has a sufficiently long fatigue life. The uniform print structure minimizes stress concentrations and allows the insole to wear uniformly throughout its life cycle. Analysis of the insole has been conducted to show that no significant resonance will occur to disrupt normal function, and the airy design aids in optimal heat dissipation.

The chosen design fits securely in shoes, with different sizes being made to fit different patients. The IMU is secured within the insole, so it accurately tracks all foot motion.

The hardware housing fulfills its primary function of protecting electronics and securing them onto footwear, with secondary functions being cable organization, cable accessibility, and thermal cooling. Snap-fit latch, hinge, and recessed element provide low-number-of-parts, efficient solutions for securing electronics and wiring, and vent cutouts serve as passive air-flow passages. The geometry of the enclosure has recessed pockets, snap-fit elements, and vents, all dimensioned for strength, and the PLA material was chosen on the basis of durability and rapid iteration.

Case kinematics are limited to the hinge and latch motion, which play well with printed tolerances, and energy considerations are limited to heat dissipation, which is controlled by vent geometry. Safety was considered in terms of rounded edges and smooth corners to avoid snagging or injury, and venting to avoid overheating risk. Case design also accommodates ergonomics while still being compact and light in weight, so it does not hinder walking movement, but with convenient accessibility to charging ports. From a production perspective, the case is easily made in inexpensive 3D printing, with the only off-the-shelf piece of hardware required being the small screws used to secure the PCB board to the case. Quality control has been maintained through consistent PLA print tolerances, and adhesive bonding of the clip in its recessed pocket ensures reliability, but long-term wear testing is yet to be performed. Assembly is simplified through a single-piece open print, with PCB and battery attached straight into recesses, and the clip being glued into a designed pocket, leaving ports open. As used, the latch enables frequent opening for servicing, venting prevents heat buildup, and the solutions involving a clip and Velcro provide a secure shoe connection.

The case life cycle was also considered, as PLA is recyclable, and modular clip and battery technology allows them to be replaced without reprinting the whole case. Maintenance is also made easy by the snap-fit latch that allows battery replacement or wiring inspection without difficulty, and the use of Velcro further improves serviceability. The cost remains low because a commercial clip and PLA filament are required, and design and production were on time, with fewer redesigns needed for future improvement.

All safety considerations have been made. The electronics inside the insole have been reinforced with additional protection; all electronics that could be removed from inside the shoe have been, and all parts have been shown to work together safely.

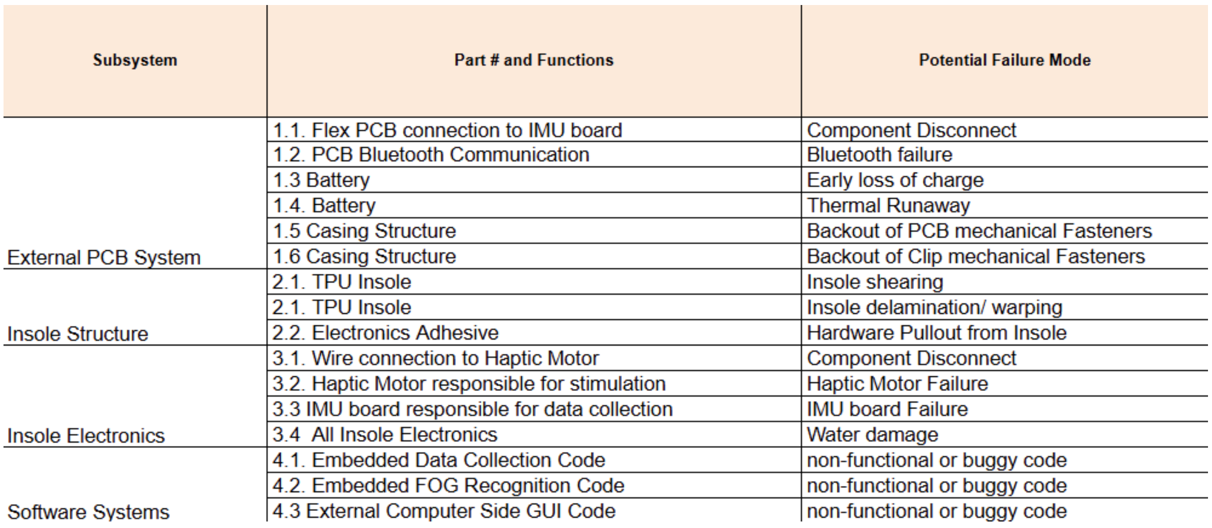
The insole harness has been specifically designed to match the desired insole feel for the target demographic. The insole stiffness has been optimized to best support the customer and provide the sturdy feel that they desire. The insole's shape has been optimized to provide the greatest foot support possible, with additional efforts still being made to further improve the design through customer feedback. The 3D printing process has been improved multiple times to create the best intrinsic "feel" for the product possible, and additional means to visually improve the product are being considered, such as using a fabric cover.

The production process has been performed multiple times. The printing process is lengthy, but easy to accomplish. Assembly of the product requires the ability to solder, but parts have been combined where practical to minimize manufacturing efforts. All materials are easily acquired. Tolerances have been shown to be compatible with minimal extra space. The insole fits inside the intended shoe size, with multiple sizes being made for varying customers.

Assembly has been shown to be simple, repeatable, and without ambiguity. The integration of electronics into the PCB has greatly simplified the production process. All operational conditions have been considered. The insole has been designed to be resilient, supportive, flexible, and stiff to satisfy all customer needs as well as ensure no electronics are damaged. Vibration-damping methods have been employed to better acquire gait data.

3.1.2 FMECA

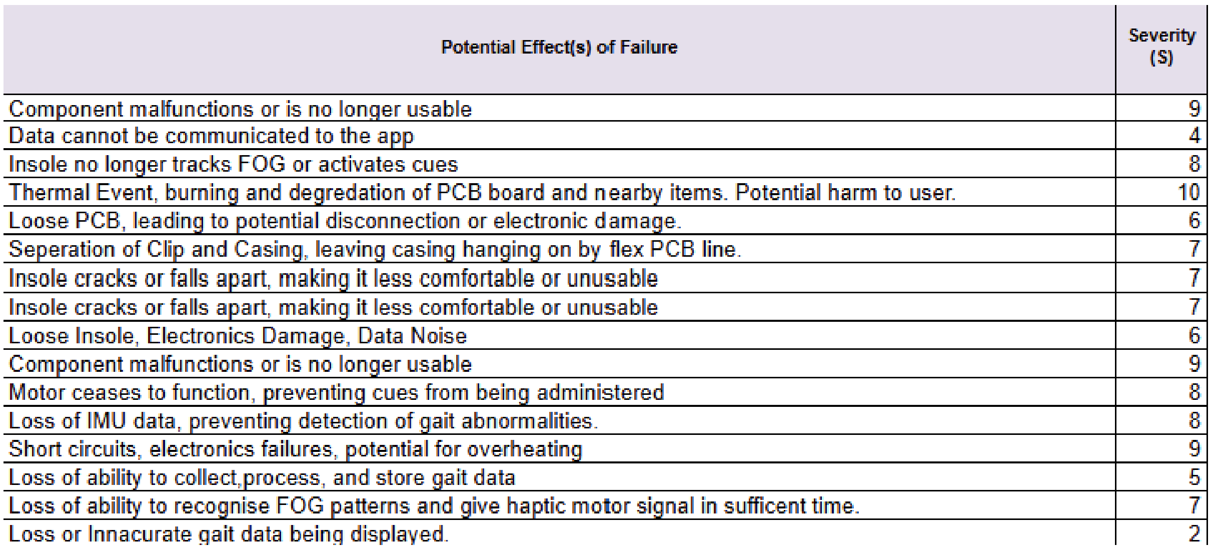
In order to properly assess, plan for, and mitigate risk, the team developed a failure modes, effects, and criticality analysis tool (FMECA) to assess the major failure modes of all design systems and subsystems within the ComSole product architecture. As seen in Figures 54 through 58 below, this was broken down into 4 main subsystems: the external PCB system, the insole structure, the insole electronics, and the software systems. Within each subsystem, all major components with probable failure modes were identified as seen in Figure 54.



Subsystem	Part # and Functions	Potential Failure Mode
	1.1. Flex PCB connection to IMU board	Component Disconnect
	1.2. PCB Bluetooth Communication	Bluetooth failure
	1.3 Battery	Early loss of charge
	1.4. Battery	Thermal Runaway
	1.5 Casing Structure	Backout of PCB mechanical Fasteners
External PCB System	1.6 Casing Structure	Backout of Clip mechanical Fasteners
	2.1. TPU Insole	Insole shearing
	2.1. TPU Insole	Insole delamination/ warping
Insole Structure	2.2. Electronics Adhesive	Hardware Pullout from Insole
	3.1. Wire connection to Haptic Motor	Component Disconnect
	3.2. Haptic Motor responsible for stimulation	Haptic Motor Failure
	3.3 IMU board responsible for data collection	IMU board Failure
Insole Electronics	3.4 All Insole Electronics	Water damage
	4.1. Embedded Data Collection Code	non-functional or buggy code
	4.2. Embedded FOG Recognition Code	non-functional or buggy code
Software Systems	4.3 External Computer Side GUI Code	non-functional or buggy code

Figure 54: DFMECA Overview

From these failure modes, potential effects and causes were identified along with the planned control method to address each mode. These can be seen in Figures 55 and 56 along with their associated severity, occurrence, and detection scores, respectively. These three scores were scaled 1-10 to give a summative idea of the risk associated with each failure mode, given their multiplication. Some of the highest risk modes identified were buggy detection software, water damage to electronics, and disconnection of the wiring to the haptic motor.



Potential Effect(s) of Failure	Severity (S)
Component malfunctions or is no longer usable	9
Data cannot be communicated to the app	4
Insole no longer tracks FOG or activates cues	8
Thermal Event, burning and degredation of PCB board and nearby items. Potential harm to user.	10
Loose PCB, leading to potential disconnection or electronic damage.	6
Seperation of Clip and Casing, leaving casing hanging on by flex PCB line.	7
Insole cracks or falls apart, making it less comfortable or unusable	7
Insole cracks or falls apart, making it less comfortable or unusable	7
Loose Insole, Electronics Damage, Data Noise	6
Component malfunctions or is no longer usable	9
Motor ceases to function, preventing cues from being administered	8
Loss of IMU data, preventing detection of gait abnormalities.	8
Short circuits, electronics failures, potential for overheating	9
Loss of ability to collect,process, and store gait data	5
Loss of ability to recognise FOG patterns and give haptic motor signal in sufficent time.	7
Loss or Innacurate gait data being displayed.	2

Figure 55: DFMECA Effects

Potential Causes and Mechanisms of Failure	Occurence (O)	Design Control	Detection (D)	RPN
Too much strain/friction	2	User Wear	1	18
Faulty signal	3	Antenna signal test	2	24
Over-tracking of gait parameters	3	Battery life test	2	48
Battery Puncture or Defect.	2	Adequate Casing Design.	2	40
Improper Installation of Fasteners	3	CAD design and Installation Procedures	1	18
Improper Installation of Fasteners	3	CAD design and Installation Procedures	1	21
Faulty Insole Print.	2	Instron tensile strength test	1	14
Faulty Insole Print/ abuse of insole	2	Instron tensile strength test and linear fatigue test	1	14
Inadequite Strength of adhesive.	4	Early Prototyping and User Wear.	2	48
Too much strain/friction	4	Instron bend test	2	72
Cable becoming unplugged	2	Instron linear fatigue test	1	16
Crushing of IMU.	3	FEA Simulations	1	24
Water/sweat ingress	4	Data collection tests	2	72
Lack of debugging code/ hidden edge cases	2	Patient Trials	4	40
Lack of debugging code/ hidden edge cases	2	ML Accuracy Test and System Latency Test	3	42
Lack of debugging code/ hidden edge cases	2	Patient Trials	4	16

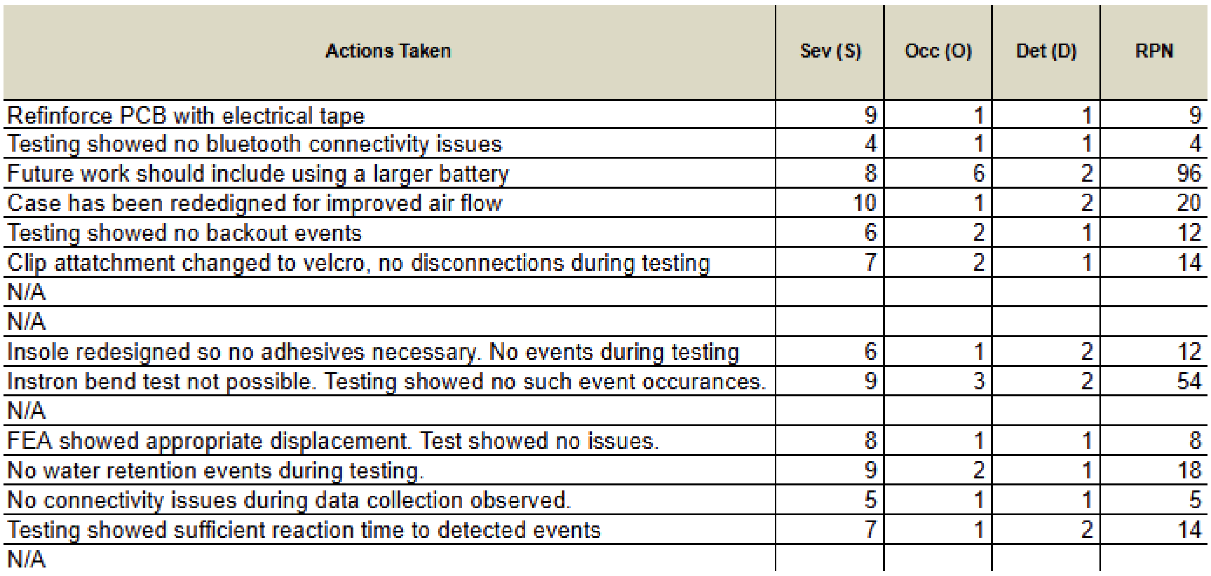
Figure 56: DFMECA Causes and Controls

To address these potential risks, controls along with the recommended actions were listed in Figures 56 and 57, giving a specific test or method of risk mitigation along with a description of specific action items. The last column in Figure 57 lists the person responsible for each control action according to the subteam structures the team has been using.

Description of Action	Responsibility & Target Completion Date
Ensure Firm Solder connections and monitor strain on flex line.	Benito/Ian/Will 10/15
Move antenna to a spot with less interference	Tyler 11/1
Program stricter sleep mode settings	Ian/Tyler/Mark 11/1
Design Casing to protect battery from damage,	Mark 10/10
Ensure all fasteners are installed at the adequate location and torque spec.	Benito/Ian/Will 10/30
Ensure all fasteners are installed at the adequate location and torque spec.	Benito/Ian/Will 10/30
Controlled epoxy measurment and mixing	Benito/Ian/Will 10/15
Controlled epoxy measurment and mixing	Benito/Ian/Will 10/15
Test proper adhesivee types and quanitity in intial prototypes and iterate as neccisary.	Matt.Benito,Will 10/30
Try different epoxies	Benito/Ian/Will 10/15
Adding stiff inserts near electronics	Benito/Ian/Will 10/15
Run FEA simulations and user wear to assess max deformations.	Will 10/6
Increase print infill	Benito/Ian/Will 10/15
Debug and improve/improvise code structure from feedback during usage in trials.	Ian/Tyler 11/1
Use test results to validate changes made in code and debugging.	Ian/Tyler 11/1
Debug and improve/improvise code structure from feedback during usage in trials.	Ian/Tyler 11/1

Figure 57: DFMECA Recommended Actions

Below in Figure 58 are the actions taken, as well as a re-evaluation of the RPN. Nearly all issues have been addressed, with only the battery life still being an ongoing issue. Future efforts should be made to improve battery life (in-depth discussion included in the Future Work section of this report.



Actions Taken	Sev (S)	Occ (O)	Det (D)	RPN
Refinforce PCB with electrical tape	9	1	1	9
Testing showed no bluetooth connectivity issues	4	1	1	4
Future work should include using a larger battery	8	6	2	96
Case has been rededigned for improved air flow	10	1	2	20
Testing showed no backout events	6	2	1	12
Clip attatchment changed to velcro, no disconnections during testing	7	2	1	14
N/A				
N/A				
Insole redesigned so no adhesives necessary. No events during testing	6	1	2	12
Instron bend test not possible. Testing showed no such event occurances.	9	3	2	54
N/A				
FEA showed appropriate displacement. Test showed no issues.	8	1	1	8
No water retention events during testing.	9	2	1	18
No connectivity issues during data collection observed.	5	1	1	5
Testing showed sufficient reaction time to detected events	7	1	2	14
N/A				

Figure 58: DFMECA Action Results

During the design process, the FMECA was used to guide both software and hardware design decisions. When designing the PCB case, vents were added early on in the design to mitigate the risk of overheating, preventing the most dangerous failure mode that the product could face. Similarly, the flex PCB was reinforced with an electrical tape wrapping in order to minimize the chance of a breakage occurring, a failure mode that would result in irreparable damage. When designing the app, software packages with proven reliability were chosen to minimize the chances of a disconnection.

3.1.3 Fault Tree Analysis

The other form of risk analysis done was a fault tree analysis (FTA), which helped break down the failure paths associated with the project's desired outcome of producing real-time feedback via a vibrational stimulus to help mitigate freezing of gait symptoms. As shown in Figure 59 below, the tree breaks down each failure path, utilizing Boolean logic gates as a means of showing the necessary inputs to cause a potential failure output as you work up the tree. The conclusions that were drawn from the FTA were that failure can be caused by three main categories: software, hardware, and the battery.

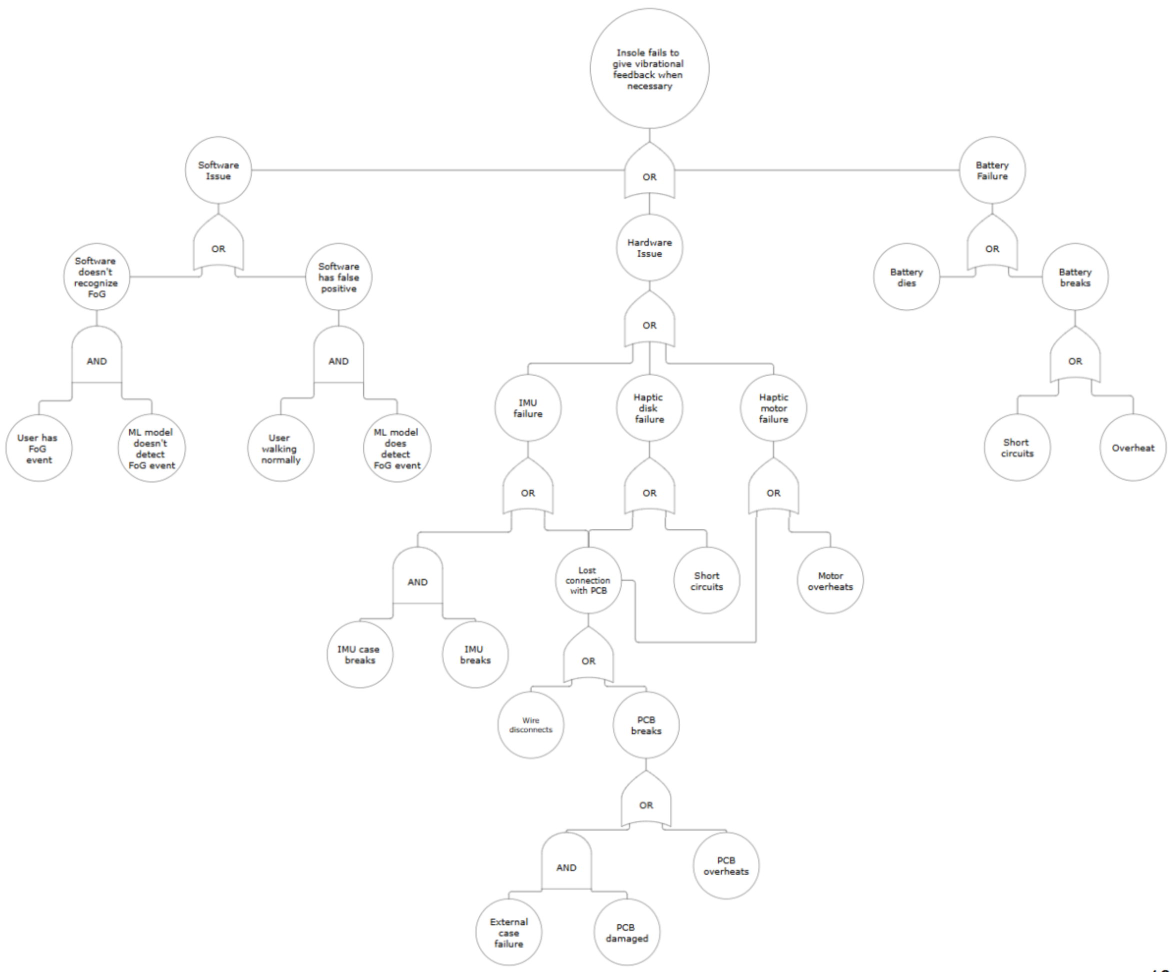


Figure 59: ComSole Fault Tree Analysis (FTA)

When looking at software, this is mostly focused on the ML model and making sure that it is able to detect FoG well and activate the vibrational stimulus effectively. Without a good ML model, the product will not provide a cue whenever it is needed, or it will provide too many cues, and the user will get frustrated by it. To mitigate this source of failure, the team spent a lot of time and effort to get a very strong ML model. There was a large emphasis on getting as much data as possible to train the model on and make it as good as possible. Moving on to the hardware, this is where more failure possibilities are present. The haptic disc, haptic driver, and IMU could all have issues that could result in the insole not functioning anymore. These issues could include wires coming unplugged, water or dirt damaging the electronics, heavy impacts breaking them, or things short-circuiting. To help prevent these issues, the team made sure to protect every component as best as possible with a case or cover. None of the wire connections are weak connections or in areas where large strains will occur to avoid wires coming unplugged. Lastly, the team consulted with multiple electrical engineering students to make sure everything was wired correctly and would be able to function well. The last area of concern was the battery itself, which could either die or fail via overheating. The team performed extensive validation tests on the battery to make sure there is as much data as possible about battery life to inform the users. Once the insole is in the hands of the user, it is up to them to keep the battery charged. In terms of overheating, the team made the decision very early in the process to move the battery out of the insole to avoid it overheating in someone's shoe. There are definitely more issues that could arise involving the vibrational cue of the insole, but these were some of the main ones that the team was able to identify and work to mitigate with the design.

3.1.4 Standards and Codes

Table 3: Project Codes and Standards.

Standard Name	**Governing Body**	**Code/Standard Number**
Failure Modes and Effects Analysis	SAE	J1739, 2021
GD&T Dimensioning and Tolerancing	ASME	Y14.5, 2018
Low Rate Wireless Networks	IEEE	802.15.4, 2024
Serial Port Profile 1.2: Bluetooth Specification	Bluetooth	SPP_SPEC V12
Comprehensive Guide to Tensile Testing of Plastics	ASTM	D638
Wi-Fi 4	IEEE	802.11n, 2009
Framework for AI Systems Using ML	ISO	74438, 2022
General Medical Electrical Equipment Safety	IEC	60601-1
Use of Human Subjects for Research	TAMU	15.999.01

Above in Table 3 is a list of the codes and standards the team consulted throughout the course of this project. These codes and standards guided design decisions and ensured all functionalities were accomplished safely and as successfully as possible.

3.2 Design for Manufacturing and Design for Assembly

In order to improve the manufacturability and assembly of the ComSole product, the team took numerous steps during the design process. First, the team determined that using 3D printing for all major mechanical components would be the fastest way to manufacture the product. Rather than using something like foam cutting and fabric stitching to create the insole, 3D printing allowed the team to quickly iterate through designs, as well as manufacture 5 final products of high quality. Next, the team's decision to embed the electronics inside the insole using a stop-and-go 3D print meant that there was essentially no assembly required. Placing the PCB inside the 3D print was simple and fast, and eliminated the need to fix the top and bottom halves of the insole together using some type of adhesive, so the assembly time was both simplified and sped up. Additionally, the decision to print the electronics case out of PLA was a design for manufacturing decision. Other 3D printing techniques, such as resin printing, were attempted, but the printing was more labor-intensive and created lower-quality products.

3.3 Experimental Data Collection

One additional component the team went through to have better design analysis on the insole models was to run data collection with real-life people who suffer from

Parkinson's. Using some connections the team had from a local non-contact gym for people who suffer from PD in the College Station area, two days of testing were set up to have people come out and try the insole and collect their gait, while the product could be tested for functionality and reliability. The following sections will go into detail about the key results.

3.3.1 Testing Procedure

In order to collect data that would have potential biomarkers of freezing symptoms, the team started by conducting research on previously conducted gait collection studies. From the results seen in [6], the team decided on a three-part, 20ft walking loop for each patient with a narrow walkway, sharp 180 degree turn, and verbal stop/start command being three potential FOG cues. Figure 60 below shows demonstrations of each of the three movements being conducted.

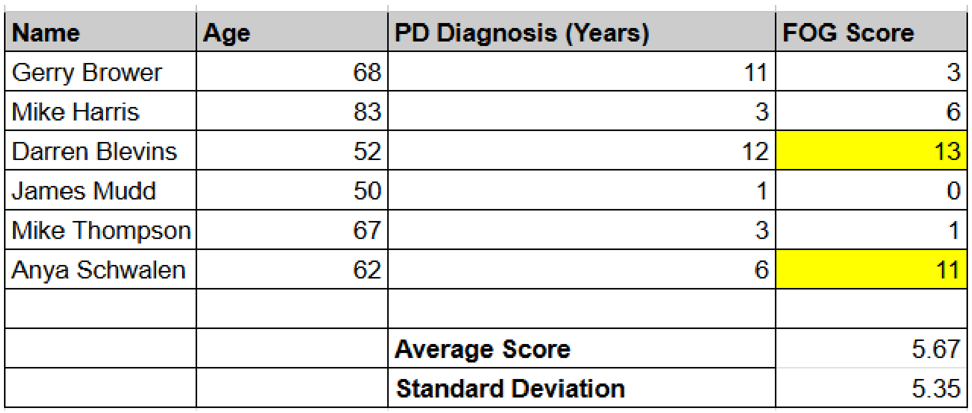
images/ExpProcedureExample.png

Figure 60: Experimental FOG motions being conducted (Removed for privacy)

As each patient went through the walking course, the data collection firmware, as mentioned in section 2.6, was used to collect CSV-formatted 6-axis motion data from the embedded insole IMU's which were labeled according to each motion type, using a GUI labeling button. Each trial was run 2 times with each respective patient for a total of 12 different walking loops over the course of 2 days and 6 different patients. Additional post-processing was done with the data, which was then given a specific classification based on the WWS FOG questionnaire that the team handed out to each patient, which can be seen in Appendix B.

3.3.2 Experimental Results

After going through collection trials, the team processed each data stream from which some key results were obtained. Of the six total patients, two were classified as potential freezers with an average freezing score of 5.67/24 as seen in Figure 61 below.



Name	Age	PD Diagnosis (Years)	FOG Score
Gerry Brower	68	11	3
Mike Harris	83	3	6
Darren Blevins	52	12	13
James Mudd	50	1	0
Mike Thompson	67	3	1
Anya Schwalen	62	6	11
		Average Score	5.67
		Standard Deviation	5.35

Figure 61: FOG Objective Results from Patient Questionnaires

With the results seen above, another conclusion was found with the linear trend between the length of FOG diagnosis and the magnitude of freezing plotted below in Figure 62.

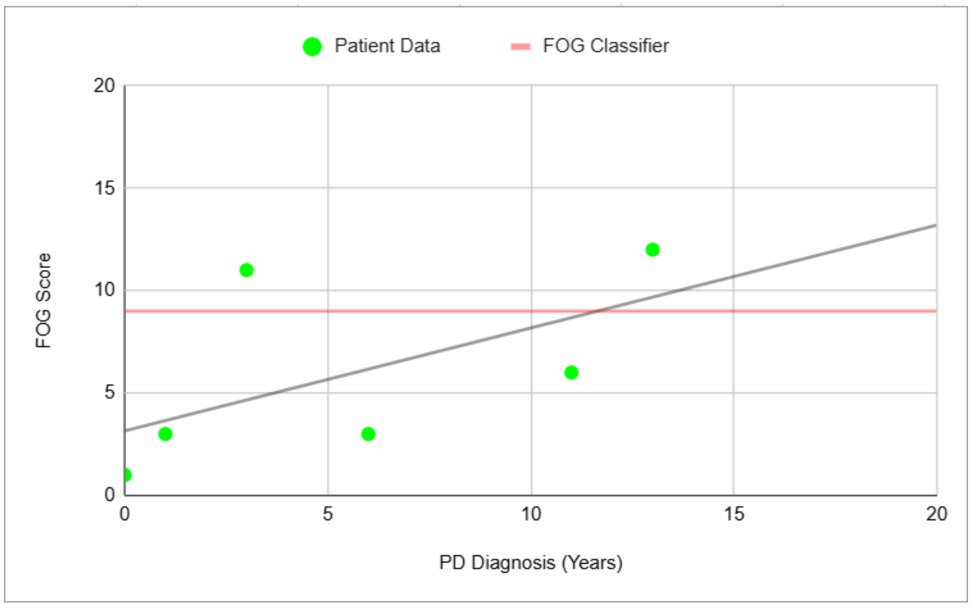


Figure 62: WWS FOG Classification as a function of PD diagnosis length.

After the trials were over, the team also conducted short patient interviews on user feedback, from which overall positive feedback was given on metrics such as comfort, fitment, and price. In all, the experimental trials helped validate product functionality and design choices, confirm gait metrics and theory behind freezing symptoms, and confirm user satisfaction.

3.4 Design Validation

3.4.1 Insole Design Validation

There were a variety of different validation techniques used during the design, manufacturing, and testing stages. Testing on prototypes was used as the main source of validation because the team prioritized prototyping early and had insoles to test with early enough in the semester, and the team figured this would give more accurate results. However, the team wanted to do some FEA tests to understand the forces acting on different components and the stresses and deformation that would result.

Starting off the validation, the team has performed multiple FEA studies on the insole. These studies were a good starting point for analyzing the design of the insole and whether there were going to be any major issues with comfort or longevity of components. These studies were done using SolidWorks.

The first FEA study conducted was a pressure test on the insole itself. Doing this allowed the group to see two main things necessary for the insole design: the deformation and the compressive stresses. To set this study up, the first step was determining how much pressure to apply to the insole and where to apply it. The team wanted to make sure the insole would be usable for all different body types, and so a boundary scenario of a 300-pound person was tested since this would be the absolute maximum weight of anyone using the ComSole product. When thinking about a step, most of a person's weight is distributed between their heel and the ball of their foot. A circle was used in each spot with a split line to apply the pressure to only these points, and the area of each circle was taken. Dividing the weight of the person by the combined surface area of these pressure points yields the value of 65.6 psi of pressure. It was assumed that when a person is walking, there are times when all of their weight is distributed on one foot since the other foot is in the air, but that it is evenly distributed between the heel and ball of their foot. The bottom of the insole was fixed because this would be supported by the shoe. In terms of materials applied to each of the parts in SolidWorks, some assumptions had to be made. Custom TPU materials were made and applied to both the insole and IMU case. The properties of the insole material were set based on the filament that was ordered and the infill used when printing (30%). This gave a compressive and tensile strength of 6 MPa and an elastic modulus of 20 MPa [7], which were the main properties necessary for this test. The IMU case had a compressive and tensile strength of 20 MPa and an elastic modulus of 60 MPa because this was printed with 100% infill [7]. Carbon steel was used for the haptic disk. The setup can be seen in Figure 63 below. An important note is that pressure is also being applied to the heel, but for some reason, the arrows aren't showing up in SolidWorks.

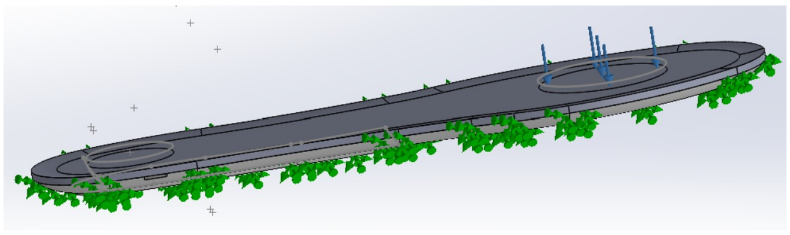


Figure 63: FEA Pressure Test Set-Up

The first goal of the pressure test was to see how much deformation the cover was experiencing from the pressure caused by a footstep. With the above setup, it was found that there would be a deformation of about 0.3 mm, meaning the heel and ball of the foot would sink into the cover by 0.3 mm. In order not to feel the components, the group originally thought about having an air gap above the IMU case and a haptic disk equal to this deformation amount, and that would stop the wearer from feeling the components. However, the original FEA studies that the group ran with this air gap above the components had higher deformation because there was less support with such a thin layer of TPU before an air gap. This was confirmed when the insole was printed and worn, as the components were very easy to feel. This led to a design change to have the air gap below the IMU case and have the overhang on the top part of the case, and resulted in the 0.3 mm value found in this study. The group had projected a deformation of somewhere around 0.5 mm, so this FEA result gave a better number than what was expected. The insole was still designed with a 0.5 mm gap, giving a factor of safety of close to 1.66. This was considered more than sufficient for any scenario because the FEA test already was a boundary test. The deformation study can be seen in Figure 64 below.

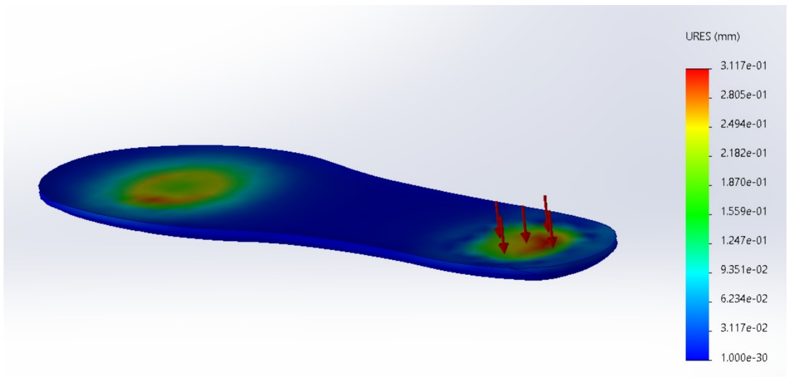


Figure 64: FEA Deformation Test Results

The other part of this FEA pressure test was to see how much compressive stress was being applied to the components inside the insole. Since the team had a finite number of functioning PCBs, the group wanted to confirm that there weren't going to be any weird, unexpected compressive stresses that might result in the breaking of an IMU. With the same pressure applied, the stress results were looked at to see where the highest stresses were going to be and what those values were. It is important to note that the SolidWorks model for the IMU case is slightly unrealistic, as there is no o-ring in the model to absorb the stress that will be applied. This test was mostly to see how much stress would be applied to the IMU case and to ensure that it wouldn't fracture and expose the IMU. It was found that about 12.5 MPa of compressive stress would be applied to the case, which is good analysis for deciding what is going to be possible for the design. The group had predicted that a maximum of 10 MPa would be applied to the IMU case from the pressure, so this exceeded expectations slightly. This number of about 12.5 MPa was not concerning, given that the lowest TPU materials have a compressive and tensile strength around 20 MPa at 100% infill, which is what was used for the IMU case. The results from this study can be seen in Figure 65 below.

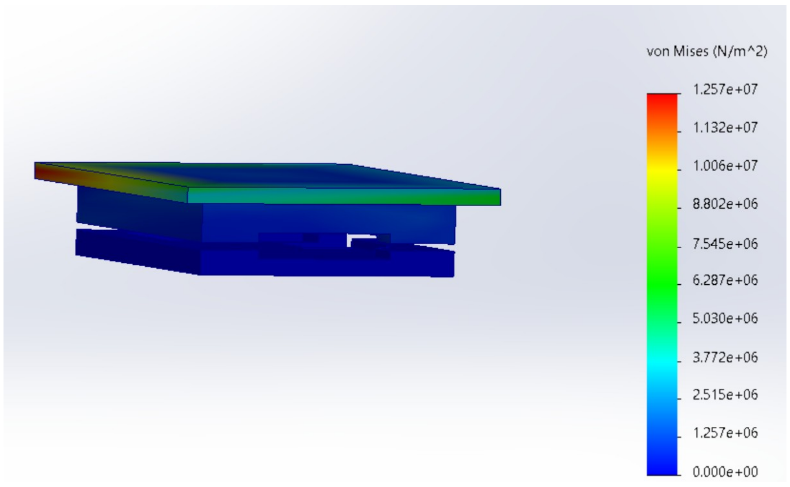


Figure 65: FEA Compression Test Results

During 401, the team also performed an FEA study that gave enough data with the original insole design, so it didn't need to be replicated with this design. This was a bending study to see how much strain different components would experience when a person took a step and the insole bent. To set up this study, the heel of the insole was fixed while the rest of it was free to move, and a force was applied to the ball of the foot to simulate the bending that occurs during a step. It was hard to estimate the forces required for this test, so the test was done based on the appearance of deformation in the insole, since the team had a good idea of how an insole looks when a person is stepping. The set-up can be seen in Figure 66 below.

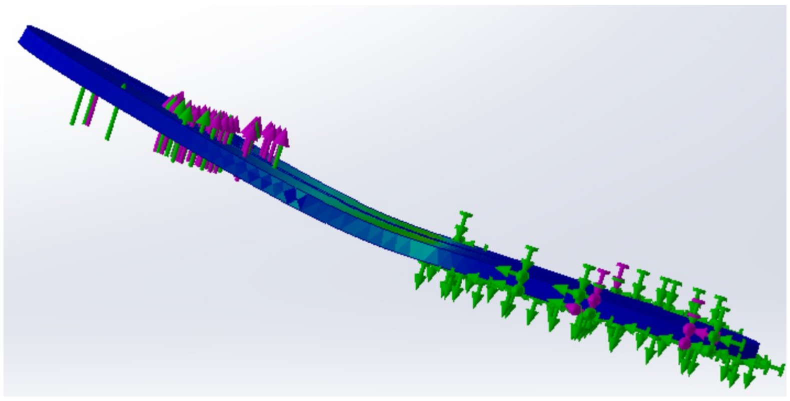


Figure 66: FEA Bending Test Set Up

Overall, this FEA study showed that the strain on the components was not going to cause an issue because they were only experiencing about 0.004%. The team was not expecting the strain to be a high percentage because no components are in the part of the insole where most of the bending is occurring. Two of the components being tested have been removed from the toe side of the insole and placed in the PCB (outside the shoe) or in the heel, so this is only applicable to the haptic disk now. The results can be seen in Figure 67 below.

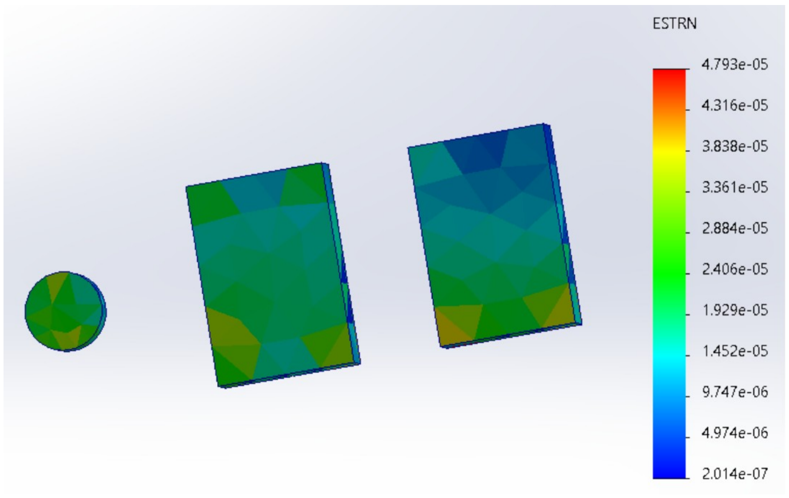


Figure 67: FEA Bending Test Results

The team looked into doing dynamic analysis on the insole design as well, but ultimately decided this wasn't going to be beneficial for a few reasons. One was the fact that the team was prototyping early; a lot of the design aspects that would be tested had already been validated using the physical prototype. Another reason was that the entire project had been done using SolidWorks, but SolidWorks doesn't allow dynamic analysis without a paid subscription. The team ultimately decided that the time and costs required to complete a dynamic analysis would not warrant the results it would provide, so the team decided to allocate that time elsewhere.

Next, throughout the process, the team created multiple prototypes in order to determine the 3D print style that would optimize the comfort and hardness of the insole, as well as see the limitations of the insole design and where it could be improved. The first prototype consisted of a solid infill 3D printed insole with cutouts that was filled with a high elasticity resin and was seen in Figure 23 when the insole design was discussed in-depth.

From this prototype, the team was able to demonstrate that all electronic selections made were viable. This insole also allowed for feedback from the project sponsors, giving the team more guidance on which design aspects should be emphasized more over others. Through conversations with them, it was determined that the curvature could be improved to cup around the bottom of the foot better. This prototype also showed that the epoxy resin wasn't going to be a viable long-term solution because of the comfort and longevity concerns described previously. Lastly, this prototype was made with 50% infill and was extremely stiff, allowing for adjustments to be made with future prototypes. All of this was discussed in more detail previously in the report, but it demonstrates the impact that prototyping had on the validation process. The team had multiple different design iterations throughout the two semesters, and each design was improved upon to help build the final design that the team landed on. Another example of a design that a team had during the MEEN 402 semester can be seen in Figure 68 and Figure 69 below. This design had more elaborate curvature to it throughout, but this actually became a negative when looking at how the insole would fit in different shoes and different user foot shapes. It also made the insole incredibly difficult to scale as more sizes were needed, so the team found a balance by taking into account universal principles of design [8].

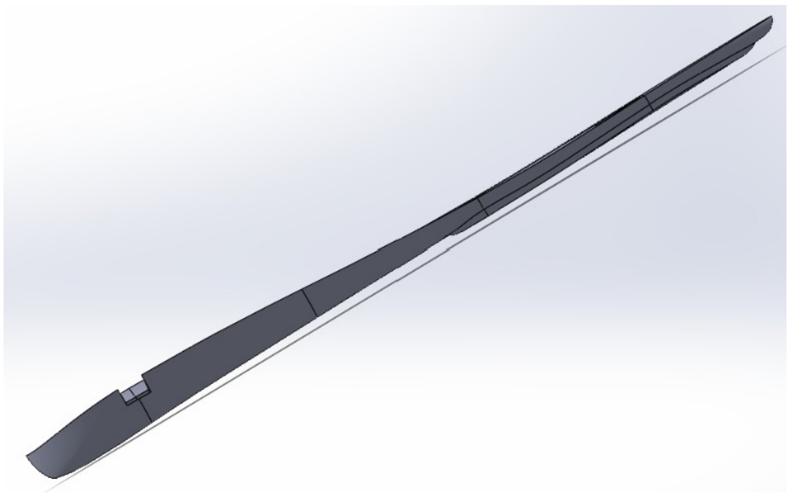


Figure 68: Previous Iteration Side View

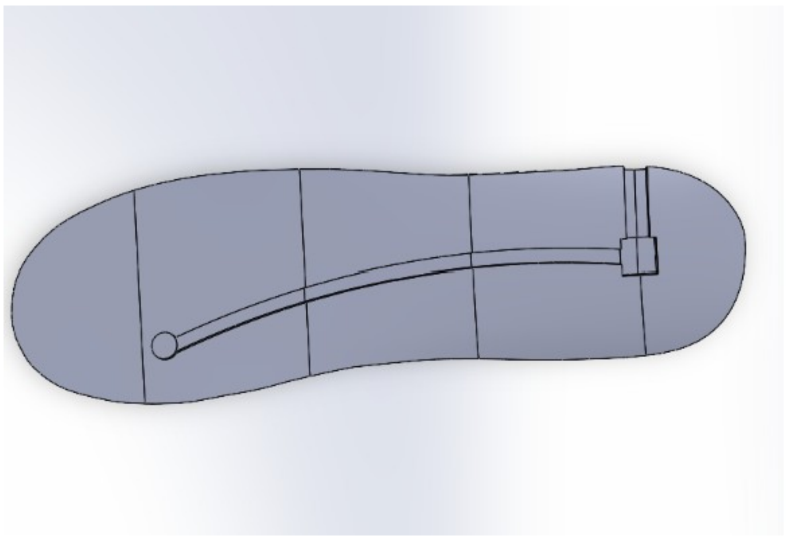


Figure 69: Previous Iteration Top View

Along with adjusting the insole shape, these different prototypes were originally used for testing out different infills to find what was going to be the best for the final product. However, the team realized that it would be much more efficient to design different samples and test the mechanical properties of each sample. This was discussed in detail in Section 2.3 in this report, but since this was also a form of validation, it is important to mention here as well.

3.4.2 Failure Weight Test

One of the validation tests that the team performed on the physical insole after the design was set was the failure weight test. The purpose of this test was to ensure that the insole is durable enough and strong enough to be usable by any realistic Parkinson's patient. The team used a maximum weight of 300 pounds as this is the weight of an American male in the 96th percentile for weight. When the team tested the insole during the test, Will performed the test at a body weight of 190 pounds. To perform it, Will went to the TAMU Rec Center and picked up different weights in 10 pound increments up to 230 pounds total and then loaded a 45 pound bar with the correct weight in 10 pound increments up to 300 pounds, picking the bar up on his back. At each weight increment, he took some steps around and made sure to put all of his weight on the insole with the electronics inside to make sure none of them would be damaged. With 300 pounds, he did some small jumps on the insole as well. Figure 70 depicts a snapshot of how this test was performed.

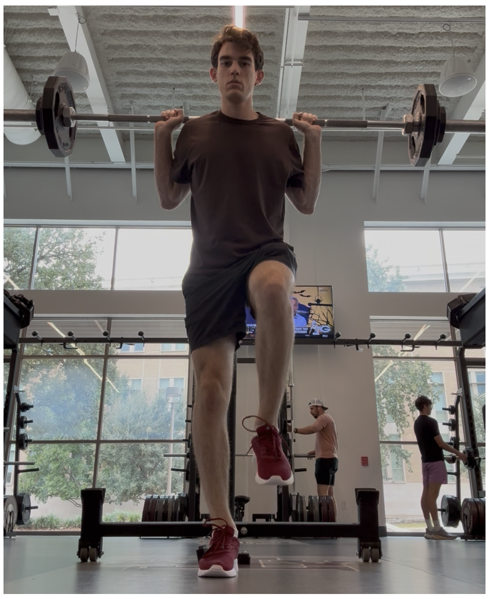


Figure 70: Performance of Weight Test

There are some conclusions that can be drawn from this test. The first is that the insole is durable enough to survive use by any user up to 300 pounds, given reasonable movements expected from an older user. Another is that even with a 300-pound person, the user cannot feel the IMU or the haptic disc inside the insole. Lastly, the entire product is functional and easy enough to use in public, based on the experience wearing it to the Rec Center.

3.4.3 Comfort Test

Another validation test on the physical insole that the team performed was the comfort test. The purpose of this test was to determine if the insole is comfortable for the average user at a level of at least 80% when compared to the user's normal insole comfort. To perform this test, the group had a collection of different people, including friends, acquaintances, and Parkinson's patients, put the insoles on and walk around in them for about a minute. After that, the user gave a score on a scale of 1-10, with a 1 being horrible comfort and a 10 being equivalent or better than their normal insole. The testee would also give feedback on why they gave it the score that they did, if they had any. Based on the 15 people who were tested, the insole had an average comfort score of 8.53 with a standard deviation of 1.09.

Some conclusions can be drawn about the comfort of the insole and some possible changes for future production. The insole itself is very comfortable, with an average score above what the group was aiming for. One piece of feedback was the insole gets a little sticky after being worn for any extended period of time, which could be fixed through the use of a fabric cover. Another point of feedback was if someone is wearing a shoe that isn't the same size as the insole, they can feel the edge of the insole some of the time. This is expected with any insole that is inside a shoe of a different size. It would definitely be beneficial to look into manufacturing half sizes to avoid this problem in the future. The most important conclusion was that it was very rare for anyone to feel any of the electrical components on the bottom of their foot, which was the main goal of the insole design. There was nobody that was tested that said anything about feeling the IMU when they were walking. The group can't definitively say that nobody will be able to feel it, but this is a very positive sign.

3.4.4 Fit Test

The next validation test that the team performed on the physical insole was the fit test. The purpose of this test was to determine if the insole would fit in different brands of shoes and that all of the different insole sizes that the team printed out would fit in the corresponding size shoe based on length, width, and shape. Based on the universal principles of design [8], in order for this to be a successful device that can hopefully be distributed to the population in the future, the insole needs to be able to be used by people with all different sized feet and with all different shoes.

The results of performing this test were that the insoles do fit well in all of the different shoes, including multiple different brands such as OnClouds, Nike, Reebok, and others. All of the different sizes fit in the respective shoe size as well, showing that the insoles are shaped correctly. There aren't many conclusions to be drawn from this test except for the fact that the insoles are designed well to fit into different shoes as required.

3.4.5 Battery Life Assessment

To evaluate the expected battery life of the ComSole device, the team first considered the primary components and their respective current draws. Table 4 summarizes the assumed active and standby currents for each component. Active current represents the estimated draw when the component is in use, while standby current reflects the draw when the device is in a sleep state. These currents form the basis for the runtime calculations using the standard relation $t = Q/I$, along with duty-cycled operation calculated as

$$I_{\text{avg}} = \frac{I_{\text{active}} \cdot t_{\text{active}} + I_{\text{sleep}} \cdot (T - t_{\text{active}})}{T},$$

and including boost regulator efficiency through

$$I_{\text{battery}} = \frac{V_{\text{out}} \cdot I_{\text{load}}}{V_{\text{battery}} \cdot \eta}.$$

Component	Active Current	Standby Current
ESP32	100 mA	100 µA
Haptic Driver	2.4 mA	4.1 µA
IMU	3.11 mA	8 µA
SD Card	150 mA	1 mA
Total	155.5 mA	3.51 mA

Table 4: Assumed component current loads for ComSole.

The team performed experimental battery life tests using three PL-633450 3.7 V, 1200 mAh Li-ion batteries under continuous Bluetooth load. The battery voltage decreased approximately linearly over time, and the boost regulator requires a minimum of 3.0-3.1 V to maintain reliable ESP32 operation. Figure 71 shows the measured battery voltage during continuous BLE operation. Comparing these results to the theoretical predictions, the team observed small differences. These discrepancies can be attributed to factors such as measurement error, boost regulator inefficiencies, and natural variations in component currents and actual battery capacities. Overall, the theoretical model provides a reasonably accurate approximation of battery behavior under typical usage.

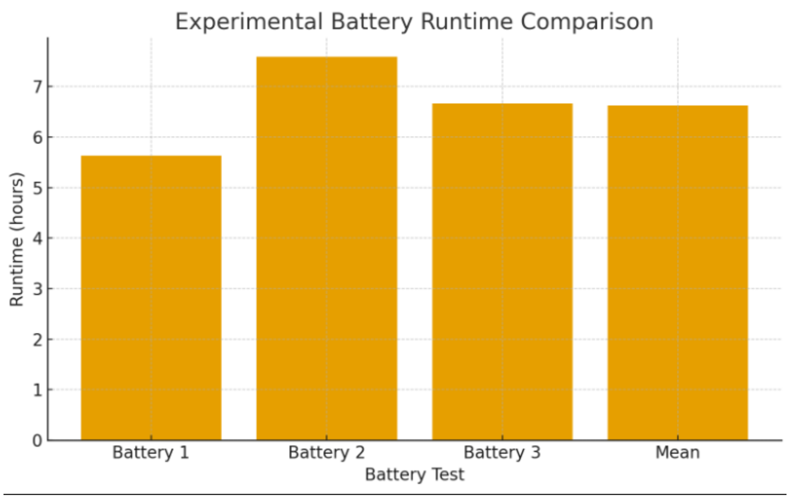


Figure 71: Experimental Battery Lifetime Chart Under Continuous BLE Transmission.

Table 5 summarizes calculated runtimes for various scenarios. Sleep mode operation results in minimal current draw, yielding roughly 150 hours, or more than six days of operation. Other Scenarios of varying ML, BLE, and duty-cycle operation result in various runtime variations. The worst-case scenario was obviously a continuous FoG ML and BLE connection that activates the haptic motor, which led to a runtime of about 3.77 hours. These results highlight how duty-cycled operation and intermittent BLE usage can significantly extend battery life, which is further presented in Figure 72.

Table 5: Battery runtime for various use cases.

Scenario	Description	Runtime (h)
Sleep Mode (Duty-Cycled, No ML/BLE)	Ultra-low current draw	150 (6+ days)
Moderate Load	Periodic sensing, no BLE	22.3
FoG-Triggered ML Only	Brief gait-detection bursts	16.7
FoG + BLE	Periodic BLE check-ins, occasional ML	21.5
BLE-Only (Experimental)	Continuous BLE transmission	6.25
Continuous FoG ML + BLE	Maximum load scenario	3.77

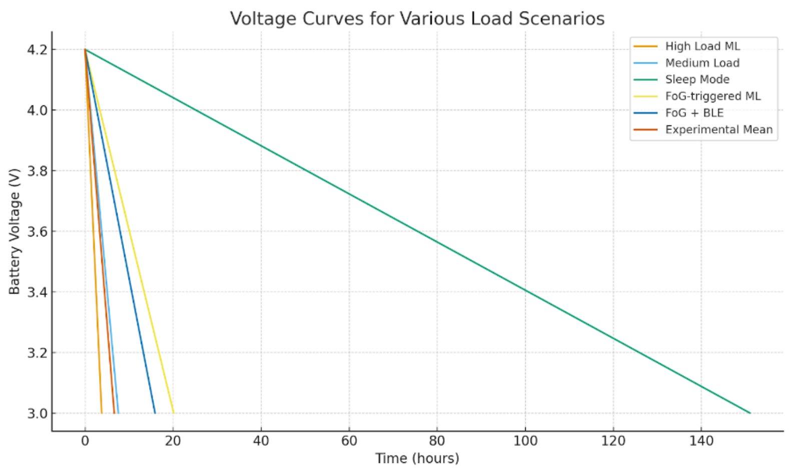


Figure 72: Battery voltage as a function of runtime for various use cases.

In practice, battery life can be extended in several ways. Increasing battery capacity directly increases runtime, while optimizing duty cycles reduces the proportion of time the device is active, thus lowering average current draw. Improvements in boost regulator efficiency can also reduce the effective current drawn from the battery. Finally, designing the system for intermittent BLE operation, rather than continuous transmission, is critical to achieving full-day or multi-day monitoring. Careful cycling of FoG ML bursts and BLE check-ins allows the device to conserve energy while still providing reliable functionality.

3.4.6 Latency Test

Next, the team conducted an event reaction latency test. In order to complete this, the team recorded the time between event occurrence and cue application. This was done by reviewing test data data sets with the machine learning model being fed the data in real time, and examining the time between the labeled event start and the model recognizing the event as occurring. This was done 5 times, and the results can be seen below in Table 6. The average response time was found to be 0.472 seconds, which is well below the team's goal of below 5 seconds.

Table 6: Latency Test Results

Sample Number	**Model Latency (s)**
1	0.25
2	1.1
3	0.39
4	0.27
5	0.35
Average	**0.472**

3.4.7 Accuracy test

Additionally, the team evaluated the accuracy of the machine learning model's predictions. The results can be seen below in Figures 73 and 74.

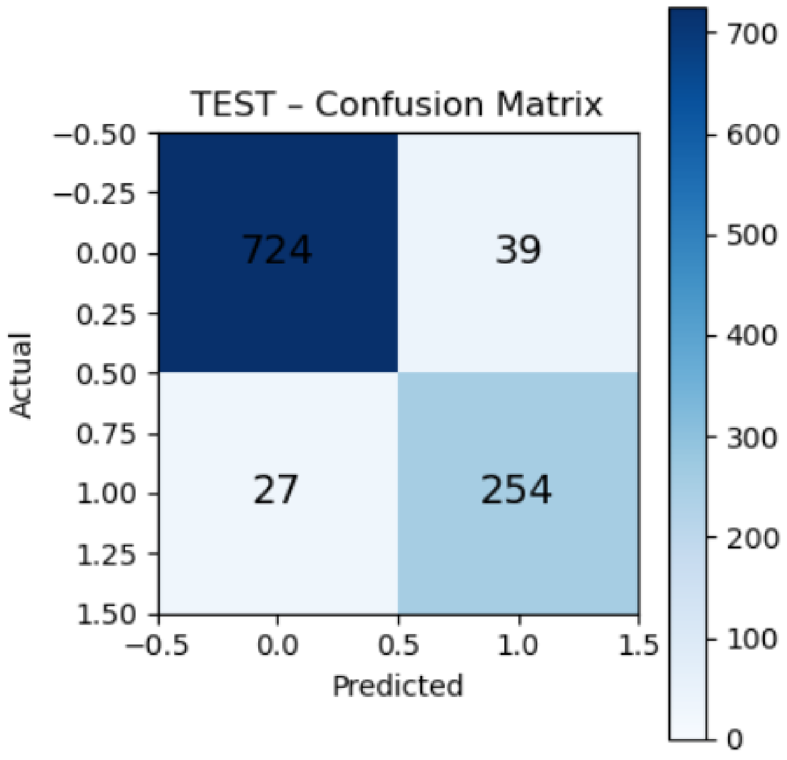


Figure 73: Machine Learning Model Confusion Matrix

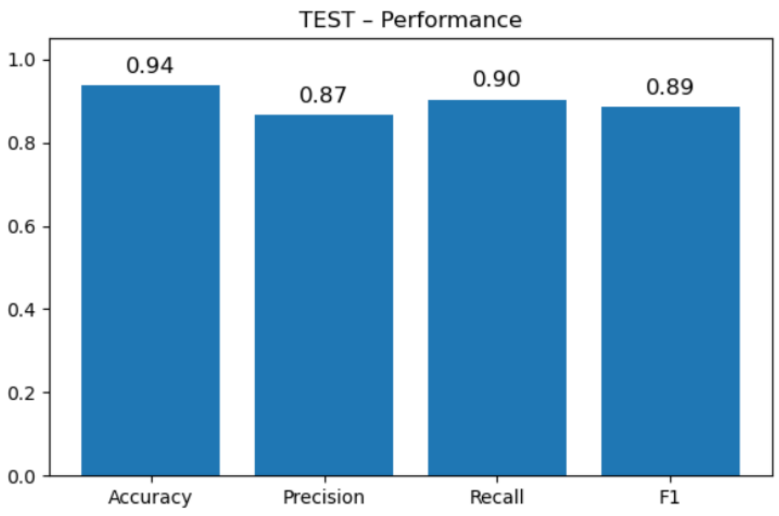


Figure 74: Machine Learning Model Test Performance

As seen above, the model performed exceptionally well. The model achieved a 94% accuracy, well above the goal of 90%, as well as great precision and recall. When considering all of these metrics together, the team can confidently say that the model both identifies when simulated freezing events are occurring, as well as does not flag normal walking patterns as freezing events.

3.4.8 Sampling Rate Validation

The final validation test performed was a sampling rate test. The team wanted to ensure that all necessary data was being collected and transmitted, so the data reception rate of the app was used to evaluate the sampling rate. As was expected, the 100 Hz data rate was received by the app as shown below in Figure 75.

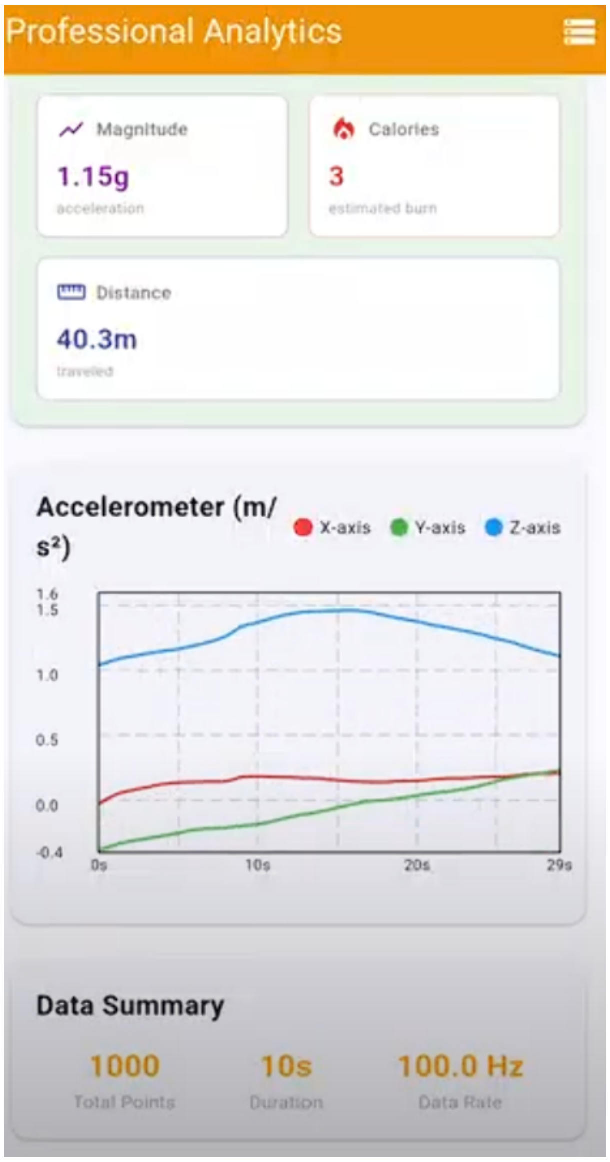


Figure 75: App Data Rate

3.5 Meeting Design Requirements

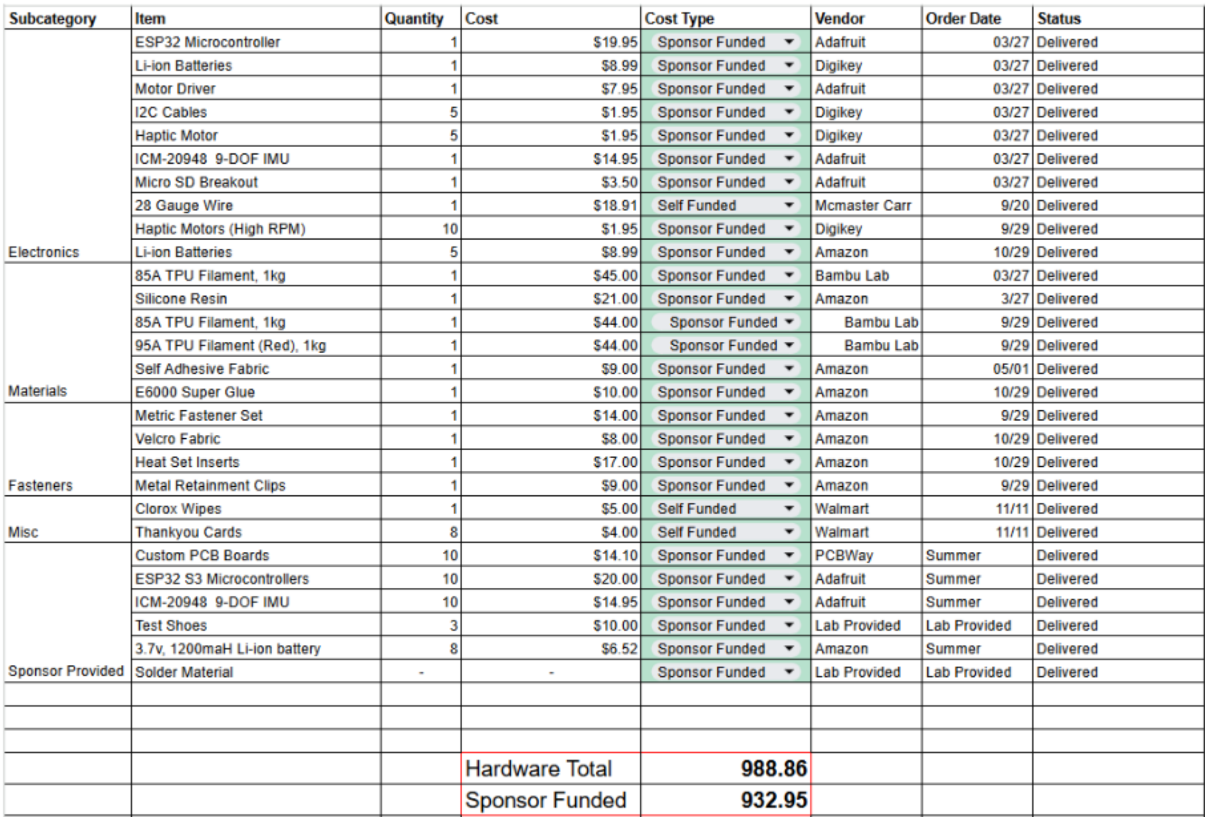
Table 7 shows the design requirements the team set out to satisfy with the ComSole product, and how each of those requirements was achieved. All values listed in this table were found using methods discussed elsewhere in this report. From this, it can be seen how the team was successful in achieving all critical design requirements in the final products, with the exception of the battery life target value, which is discussed in detail in section 3.4.5.

Table 7: Design Requirements Table

Design Requirement	**Target Values**	**Achieved Values**
Detect Freezing of Gait	$> 90\%$ accuracy	Machine learning model achieved 94% accuracy
Provide real-time feedback	Less than 5 s between event and feedback	Feedback time testing showed an average latency of 0.472 seconds
Low cost	Less than 200$ USD	Final cost per insole pair came out to approximately $199
Useful Data Extraction/Filtering	100 Hz Sampling Frequency	100 Hz Sampling Frequency achieved
Data Transmission Loss	5% IMU data transmission loss	No IMU data loss has been seen during any phase of testing or validation
Insole Meets Ergonomics Specifications	Prototype US size mens 8–11	Insoles for sizes men's 8–11 as well as women's 7 were created
Insole Meets Ergonomics Specifications	Perceived, 1–10 rating above 7	Average comfort rating was 8.53 with a standard deviation of 1.09
Insole Will Last a Normal Lifecycle of Usage	10^7 strains without critical failure	Unable to test insole to failure, no critical failures occurred during design, testing, or validation
Battery Life Under Normal Operating Conditions is Sufficient	$\sim$1 week of normal usage	Battery able to operate system at full power draw for an average of 6.5 hours
Thermal Management	Keep insole hardware within specified operating conditions via software monitored temperature conditions	No thermal events occurred during testing or validation

3.6 Cost Accounting and Cost Model

For the ComSole Project, the team received an approved budget of approximately $3500.00. During MEEN 401, a preliminary Bill of Materials (BOM) was developed assuming production of ten functional insole units, with some margins to account for prototyping, testing, and miscellaneous costs. Since then, over the course of the 402 semester, the team ended up with 5 working insoles from the costs of 7 initial electronics packages. These costs totaled out to be approximately $988.86 for all purchased components in this capstone, the majority of which came from sponsor funding. The entire breakdown is shown in Figure 76 below.



Subcategory	Item	Quantity	Cost	Cost Type	Vendor	Order Date	Status
	ESP32 Microcontroller	1	$19.95	Sponsor Funded	Adafruit	03/27	Delivered
	Li-ion Batteries	1	$8.99	Sponsor Funded	Digikey	03/27	Delivered
	Motor Driver	1	$7.95	Sponsor Funded	Adafruit	03/27	Delivered
	I2C Cables	5	$1.95	Sponsor Funded	Digikey	03/27	Delivered
	Haptic Motor	5	$1.95	Sponsor Funded	Digikey	03/27	Delivered
	ICM-20948 9-DOF IMU	1	$14.95	Sponsor Funded	Adafruit	03/27	Delivered
	Micro SD Breakout	1	$3.50	Sponsor Funded	Adafruit	03/27	Delivered
	28 Gauge Wire	1	$18.91	Self Funded	Mcmaster Carr	9/20	Delivered
	Haptic Motors (High RPM)	10	$1.95	Sponsor Funded	Digikey	9/29	Delivered
Electronics	Li-ion Batteries	5	$8.99	Sponsor Funded	Amazon	10/29	Delivered
	85A TPU Filament, 1kg	1	$45.00	Sponsor Funded	Bambu Lab	03/27	Delivered
	Silicone Resin	1	$21.00	Sponsor Funded	Amazon	3/27	Delivered
	85A TPU Filament, 1kg	1	$44.00	Sponsor Funded	Bambu Lab	9/29	Delivered
	95A TPU Filament (Red), 1kg	1	$44.00	Sponsor Funded	Bambu Lab	9/29	Delivered
	Self Adhesive Fabric	1	$9.00	Sponsor Funded	Amazon	05/01	Delivered
Materials	E6000 Super Glue	1	$10.00	Sponsor Funded	Amazon	10/29	Delivered
	Metric Fastener Set	1	$14.00	Sponsor Funded	Amazon	9/29	Delivered
	Velcro Fabric	1	$8.00	Sponsor Funded	Amazon	10/29	Delivered
	Heat Set Inserts	1	$17.00	Sponsor Funded	Amazon	10/29	Delivered
Fasteners	Metal Retainment Clips	1	$9.00	Sponsor Funded	Amazon	9/29	Delivered
	Clorox Wipes	1	$5.00	Self Funded	Walmart	11/11	Delivered
Misc	Thankyou Cards	8	$4.00	Self Funded	Walmart	11/11	Delivered
	Custom PCB Boards	10	$14.10	Sponsor Funded	PCBWay	Summer	Delivered
	ESP32 S3 Microcontrollers	10	$20.00	Sponsor Funded	Adafruit	Summer	Delivered
	ICM-20948 9-DOF IMU	10	$14.95	Sponsor Funded	Adafruit	Summer	Delivered
	Test Shoes	3	$10.00	Sponsor Funded	Lab Provided	Lab Provided	Delivered
	3.7v, 1200maH Li-ion battery	8	$6.52	Sponsor Funded	Amazon	Summer	Delivered
Sponsor Provided	Solder Material	-	-	Sponsor Funded	Lab Provided	Lab Provided	Delivered
			Hardware Total	988.86			
			Sponsor Funded	932.95			

Figure 76: Cumulative Capstone Final Budget

Table 8: Itemized BOM for a Singular Insole.

Category	**Items**	**Qty**	**Unit Price**	**Extended Price**
Electronics and Hardware	Lithium Ion Polymer Battery	1	$7.95	$7.95
	Haptic Motor	1	$1.95	$1.95
	28 Gauge Wire	1 ft	$3.08	$0.12
	Custom Flex PCB	1	$28.46	$28.46
	9 DOF IMU	1	$14.95	$14.95
	Metric Hardware*	4	$0.05	$0.20
Materials	PLA Filament	0.06 kg	$20.00	$1.47
	TPU 85A Filament	0.21 kg	$37.79	$9.23
Labor	Manufacturing / Assembly	5 hrs	$15.00/hr	$75.00
	Software Development and Upkeep**	1.5 hrs	$40.00/hr	$60.00
			Per Insole Set Cost	**$199.33**

*Hardware cost based on estimated fastener usage.
**Software development charge includes firmware debugging and periodic updates.

Table 8 is the finalized per-insole price breakdown based on the total budget spent plus the hypothetical labor rates given in lecture for the assembly and production costs incurred. A cost and percentage breakdown of all category-level costs for the combination of both semesters' worth of costs is shown below in Table 9.

Table 9: Holistic Cost Breakdown for ComSole Project.

Category	**Description**	**Estimated Share (%)**	**Total Projected Cost**
Materials	PLA and TPU filaments, adhesives, insole liners, casing attachment methods.	22	$221.00
Electronics & Components	Wiring, haptic motors, initial IMU's, MCU's, Custom Flex PCB's.	74	$731.00
Software Expenses	Licenses & data.	0	$0.00
Miscellaneous	Sum of the combined total margin.	4	$37.00

4 Broader Impacts of Design

4.1 Lifecycle of Design

As part of the design and manufacturing work, the team focused on multiple specifications to tailor the product to improve its design life-cycle. With the chosen insole layout, one decision that was made early on was to have the electronics embedded so that reuse in a new product would be available and easy, as the insole deteriorates under cyclical strain. With a simple print-in-place approach, all electronics could be completely swapped to a new insole, given that a material degradation or upgrade would deem such necessary. In terms of maintenance, the external PCB case was designed so that all of the electronics housed could be easily replaced or upgraded, given the need for something like a better battery or a different microcontroller upgrade. The environmental impact of the ComSole product is mainly related to the electronics and battery utilized, as the main power source is a 3.3-volt lithium-ion battery, which is known to be environmentally troublesome. Improvements for this include suggestions for manufacturers to give proper methods of disposal and alternative battery types with less toxic chemicals, like Sodium Ion devices.

4.2 Intellectual Property

Intellectual Property, or (IP) for short, refers to creations of the mind controlled by law that allow for exclusive rights to the usage for a period of time. For the ComSole product, the group itself does not have any IP rights, as it is a university research-funded initiative, but the concept of a smart insole device with an embedded two-part monitoring and feedback system is something that could have reason for the creation of something like a patent for the research lab's work and development. With patents, the main focus of concern would be showing sufficient uniqueness from other similar already existing devices, but given that to be true, the product could indeed become the intellectual property of Dr. Ya Wang and the mechanical engineering department. Given what the team has heard in the time working with the lab, it is very likely they will not be interested in such applications until the product has reached a more complete and ready-to-be-marketed state.

4.3 Liabilities

The ComSole product has the potential to be a serious liability concern. The insole system includes electronics that could create safety hazards such as shocks, pokes, or fires, so it was essential to consider how to mitigate these liabilities throughout the design process. First, the insole's compliant material and thick padding minimize the risk of any electronics poking the bottom of the user's feet. Next, all bulky and potentially dangerous electronics, such as the battery and microcontroller, were placed outside of the insole to minimize the risk of any breakages that could create a dangerous situation. Additionally, the case that holds the electronics was designed with ventilation cutouts to decrease the likelihood of any thermal runaway events that could result in sparking or fires. Finally, the product was used extensively by the team before introducing it to customers to ensure that they were at minimal risk for injuries. During both validation testing and design, the team interacted with the insole product extensively to verify its safety.

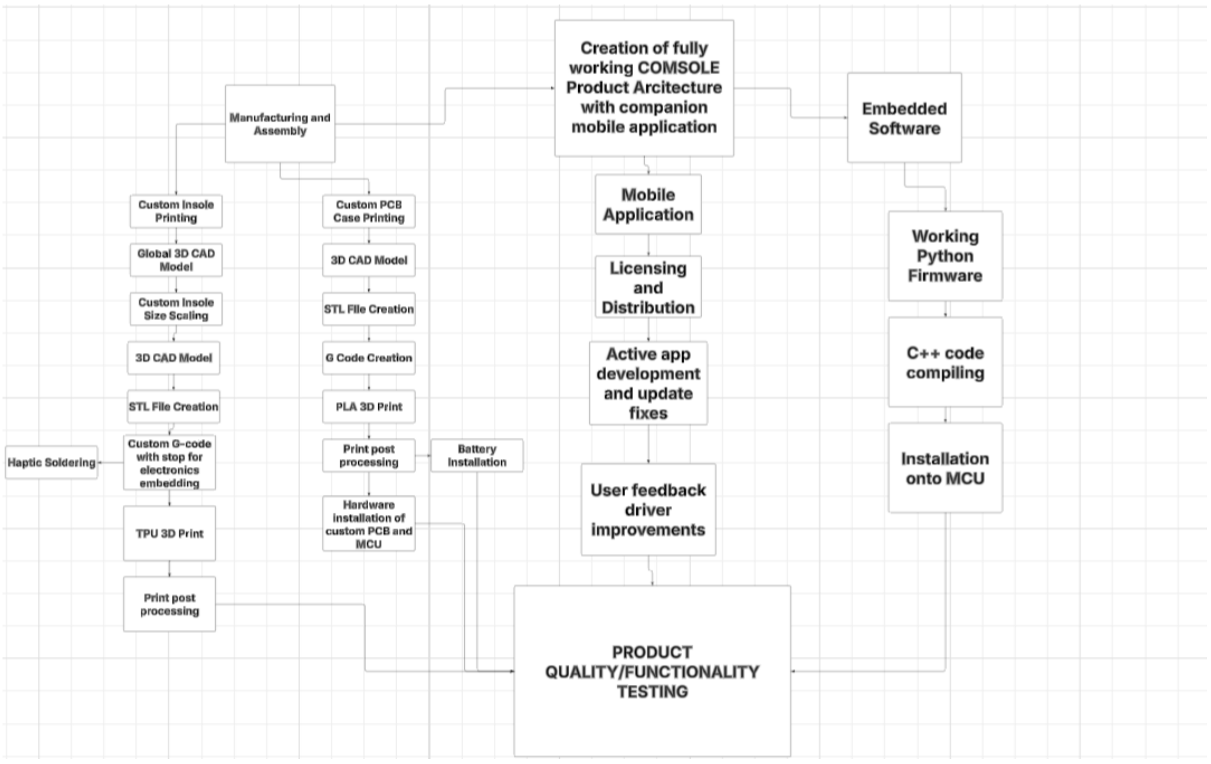
4.4 Ethical Considerations

With the key purpose of the project being the betterment of individuals lives, particularly those who suffer from late-stage PD symptoms, ethical implications were of high priority when designing and manufacturing the final product. One initial concern was the issue of safety when users would be given a light vibrational feedback, which could be intense for people with weaker lower limbs and appendages. The team looked into how these types of stimuli would effect the human system and came to the conclusion that the haptic disc being used wouldn't be potent enough to do anything more than give a light feedback sensation. Additionally, there was the other concern of making a product that would become so expensive that people in critical need of such a device wouldn't be able to readily afford it. This is especially important for those who would already be having to pay a lot in medical bills for their symptoms like most of ComSole's clientele. The team was able to improve this situation by utilizing the most cost effective setup with a rapid manufacturing approach to ultimately lower the per product cost to within a range acceptable to the standards of those who were interviewed. By being able to make a low cost, safe, and reliable product, the ethical issues that could arise from various use cases were heavily mitigated.

5 Summary

5.1 Work Breakdown Structure

With the ComSole product now being a working product system, a theoretical company would be able to implement the design using the flow chart as seen below in figure 77.



Creation of fully working COMSOLE Product Arcitecture with companion mobile application
Manufacturing and Assembly
Embedded Software
Custom Insole Printing
Global 3D CAD Model
Custom Insole Size Scaling
3D CAD Model
STL File Creation
Custom G-code with stop for electronics embedding
Haptic Soldering
TPU 3D Print
Print post processing
Custom PCB Case Printing
3D CAD Model
STL File Creation
G Code Creation
PLA 3D Print
Print post processing
Battery Installation
Hardware installation of custom PCB and MCU
Mobile Application
Licensing and Distribution
Active app development and update fixes
User feedback driver improvements
Working Python Firmware
C++ code compiling
Installation onto MCU
PRODUCT QUALITY/FUNCTIONALITY TESTING

Figure 77: Hypothetical Work Breakdown for Product Implementation

The primary categories for implementing the product are insole and case manufacturing, app implementation, and embedded firmware. With the product still primarily in a proof-of-concept phase, this workflow chart would be more refined as the lab continues to progress towards a more refined and industry-ready product.

5.2 Gantt Chart

The project was primarily organized using a Gantt chart, as seen in Figure 78 below. This tool allowed the team to break down all the primary deliverables into 5 main subgroups which were mechanical design, software development, data collection, validation testing, and lecture specific tasks. As the semester progressed, tasks would be completed and further progress would move along into a concurrent or dependent task on the timeline. Additionally, risk assessments were done on each task item as given in the third column color scheme, with the dark red indicating high risk tasks which were given special attention or consideration.

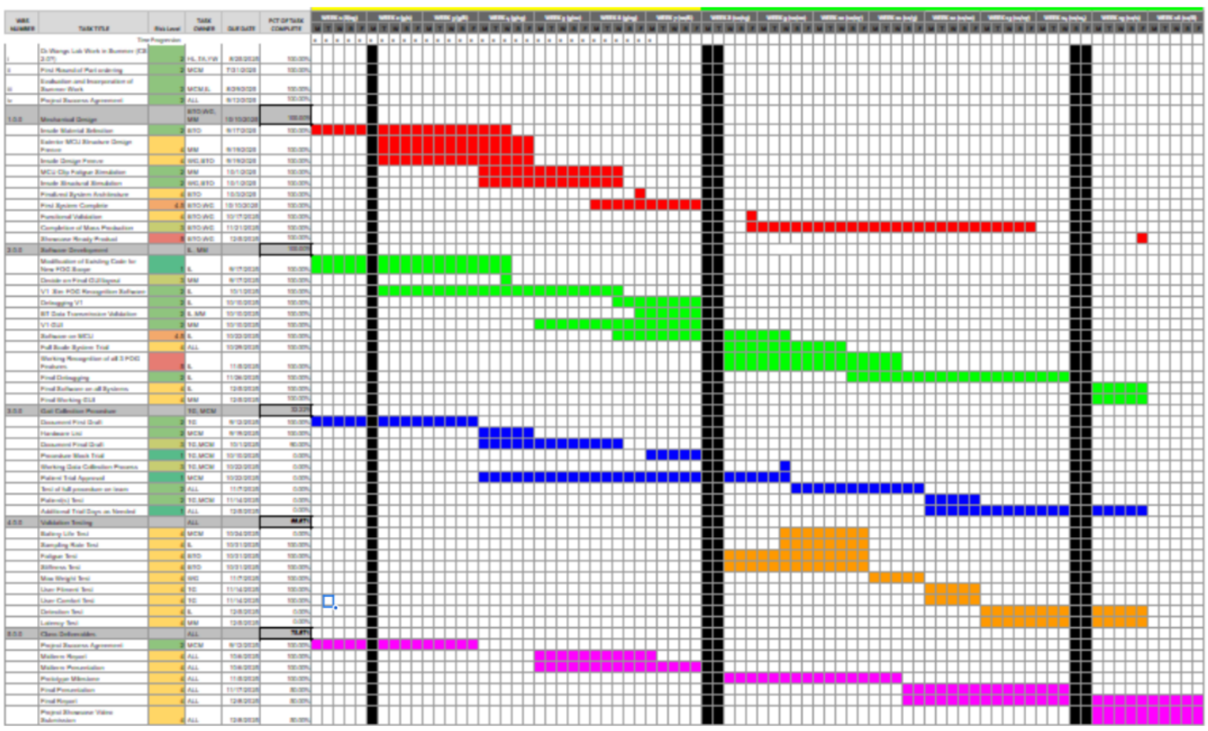


Figure 78: 402 Gantt Chart

5.3 Additional Steps to be Taken

While the current state of the design is fully functional and achieves the goals set out by the project group, there are still some additional resources that could improve the current functional framework. One key aspect is the identification of a wider scope of FOG cases. Currently, the product is able to detect shuffling and stuttering symptoms using the aforementioned machine learning model; however, there are several other varieties of freezing gait, including stutter steps, heel strike variations, and ankle rotations. With greater data collection and identification of the motion profiles that associate with such features, the product could, in theory, be able to help identify a much wider variety of freezing symptoms and, as a result, widen its impact scale. Other avenues for steps that could be taken could include more research into material optimization for the insole design and structure, and lower-cost electronics to improve the bottom-line costs for the customer without sacrificing functionality.

5.4 Limitations of Current Solution

Currently, as mentioned above, the insole successfully tracks 6-axis motion data in real time and is able to detect and mitigate basic freezing symptoms in an average delay of around 0.5 seconds. A companion mobile application is also able to successfully connect to the Insole via BLE and collect gait data in real time for processing and user analytics. Within these two product architectures, some of the key limitations primarily exist in the software side. While the machine learning model was able to be implemented with detection accuracies as high as 90%, that was only within a very specific subset of data training and only for a few basic motion profiles, so one limitation currently is the ability to detect other types of freezing symptoms and maintain similar accuracies on much larger scales of people. In addition, while the app is fully functional and collects a steady

100 Hz data stream, the user interface currently only has a basic display of analytics, including step counts, calories, and walking distance, but with further refinement, this could definitely be improved and expanded upon. Finally, on the hardware side, one current limitation is long-term reliability, as the current electromechanical integration has several points that were seen to develop stress concentrations and have inconsistent performance as structural housing. While these issues never fully resulted in a critical error, they create a major limitation on the product's ability for mass production until a much larger emphasis on quality is taken.

5.5 Future Work

At the conclusion of the team's time in the senior capstone program, the current future of the ComSole product rests in the hands of Dr. Wang's lab. Future work currently planned includes refinement of the mobile application, development of a insole with controller embedded architecture, and an internal battery supply as well. While these plans have yet to be fully developed, the current idea is to utilize smaller microcontrollers available through suppliers such as adafruit and a coin-type silver oxide battery supply to meet the thickness requirement to become fully embedded into the insole. The mobile application is looking to see upgrades on the professional analytics page to include more bio-feedback and information pertinent to someone like a medical professional. Additional lessons learned to be mitigated in future work include stress concentrations developed in the flexible PCB wiring from stiffening the connection, the need for greater data volume to train the machine learning model on, and more reliability and quality studies on the electronic development. For the machine learning model, more data from different patients will ensure that the model generalizes better. Additionally data collection on different FoG episodes, apart from shuffling, would be beneficial for having a more complete product.

In terms of insole design, there could be some slight improvements, including adding a fabric cover and possibly having extra size options (for thinner or wider feet). Once these aspects of the design are improved upon, the product would likely be able to start being produced and purchased by Parkinson's patients, which would require a plan of action for mass production. With the current insole design requiring a 3D print to get the necessary infill characteristics and insert the electronics, this would result in a challenge if there were heavy demand for the product. An injection mold with a different material could be used to get the same mechanical properties, and some kind of protective case that could withstand the molding process would be necessary to embed the components during manufacturing.

6 Acknowledgments

Over the course of both 401 and 402 semesters, the team has had the opportunity to work with several different individuals and groups who have made significant contributions towards the progression of this product. First and foremost, the team would like to thank Dr. Ya Wang and Dr. Haili Liu for being the sponsors and mentors for the project, providing both monetary and intellectual support the entire way. In addition, both of the team's capstone studio instructors, Dr. Astrid Layton and Dr. Ni Wang, helped guide the team and mold the project to become successful in the end. Cindy Conte and the staff at Rock Steady Boxing in College Station were also extremely helpful in setting up multiple days for customer interviews, data collection, and product feedback with real-life people who suffer from PD. While there are too many people to name in here, the team would also like to give acknowledgments to all the individuals with PD who came out and supported the team when working at RSB doing the aforementioned studies. The ComSole team is extremely thankful to everyone who had a hand in the project and hopes they know how important their impacts were.

References

[1] Bansal, S. K., Basumatary, B., Bansal, R., and Sahani, A. K., “Techniques for the detection and management of freezing of gait in Parkinson’s disease – A systematic review and future perspectives,” **10**, p. 102106.

[2] Forsaa, E., Larsen, J., Wentzel-Larsen, T., and Alves, G., “A 12-year population-based study of freezing of gait in Parkinson’s disease,” **21**(3), pp. 254–258.

[3] Hua, R. and Wang, Y., “Monitoring Insole (MONI): A Low Power Solution Toward Daily Gait Monitoring and Analysis,” **19**(15), pp. 6410–6420.

[4] SparkFun Electronics, “SparkFun ICM-20948 Arduino Library,” `https://github.com/sparkfun/SparkFun_ICM-20948_ArduinoLibrary`

[5] Ryan Powell, “NimBLE-Arduino,” `https://github.com/h2zero/NimBLE-Arduino`

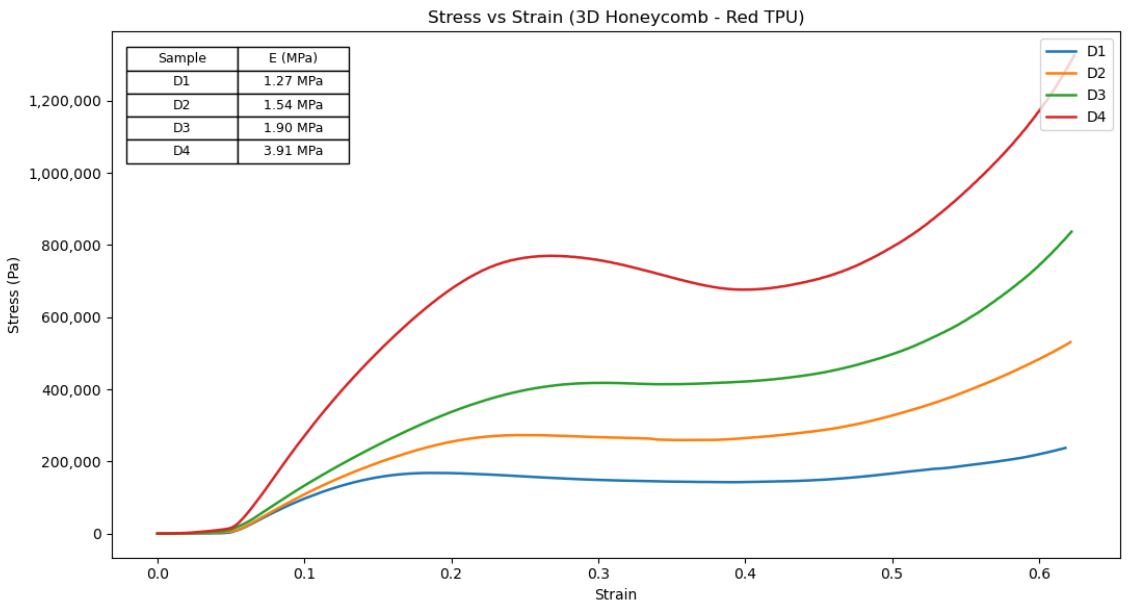
[6] John, A. R., Cao, Z., Chen, H.-T., Martens, K. E., Georgiades, M., Gilat, M., Nguyen, H. T., Lewis, S. J. G., and Lin, C., 2023, “Predicting the Onset of Freezing of Gait Using EEG Dynamics,” Applied Sciences, **13**(1), p. 302.

[7] BASF 3D Printing Solutions BV, 2022, “Ultrafuse TPU 85A Technical Data Sheet,” Version 2.5, accessed 2025-12-08, `https://move.forward-am.com/hubfs/AES%20Documentation/Flexible%20Filaments/TPU%2085A/TDS/Ultrafuse_TPU_85A_TDS_EN_v2.5.pdf`

[8] “About Universal Design,” accessed 2025-03-16, `https://universaldesign.ie/about-universal-design`

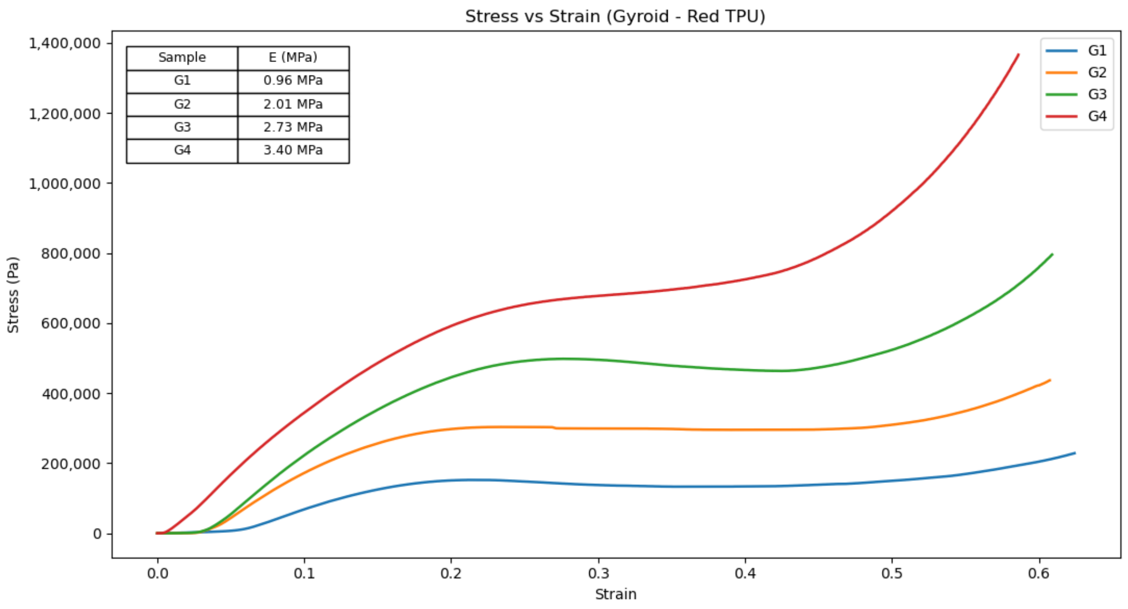
Appendices

A Figures



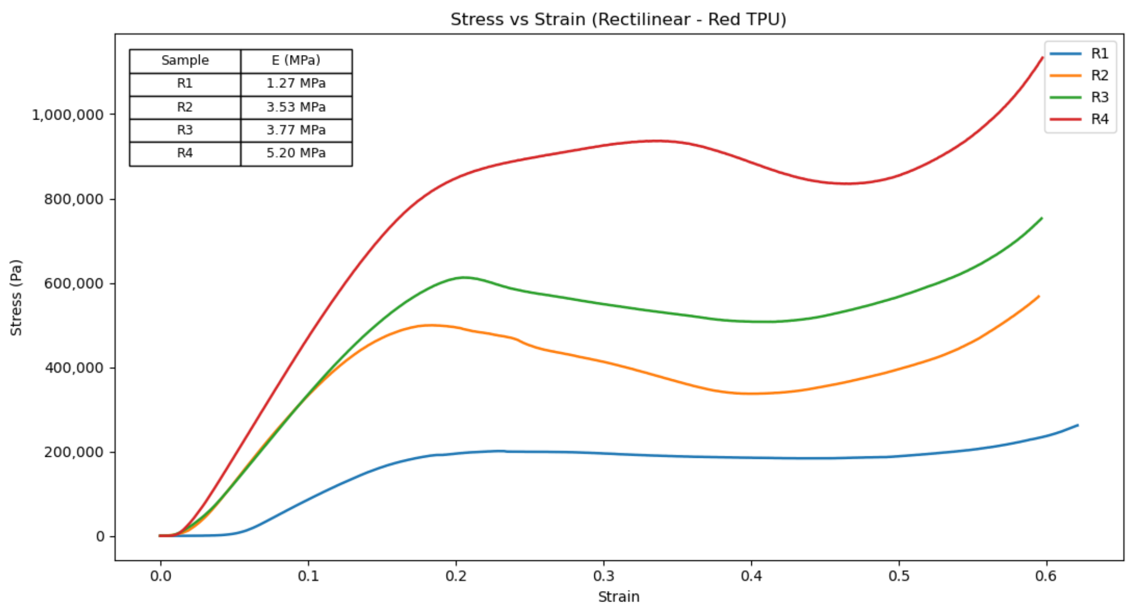
Sample	E (MPa)
D1	1.27 MPa
D2	1.54 MPa
D3	1.90 MPa
D4	3.91 MPa

Figure 79: Stress–strain response for 3D honeycomb infill patterns (95A red TPU)



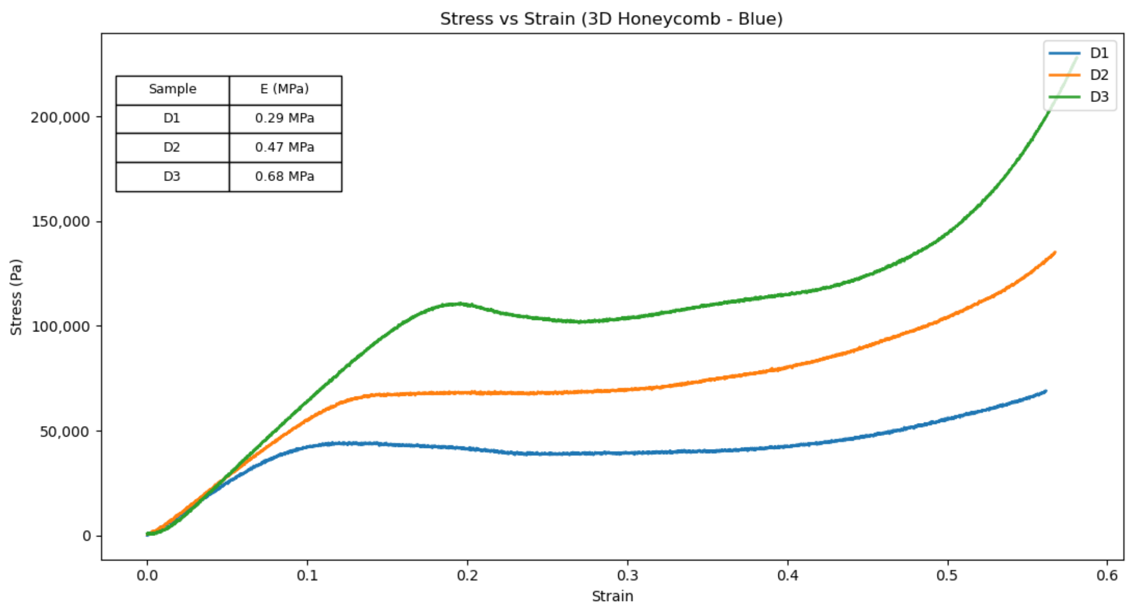
Sample	E (MPa)
G1	0.96 MPa
G2	2.01 MPa
G3	2.73 MPa
G4	3.40 MPa

Figure 80: Stress–strain response for gyroid infill patterns (95A red TPU)



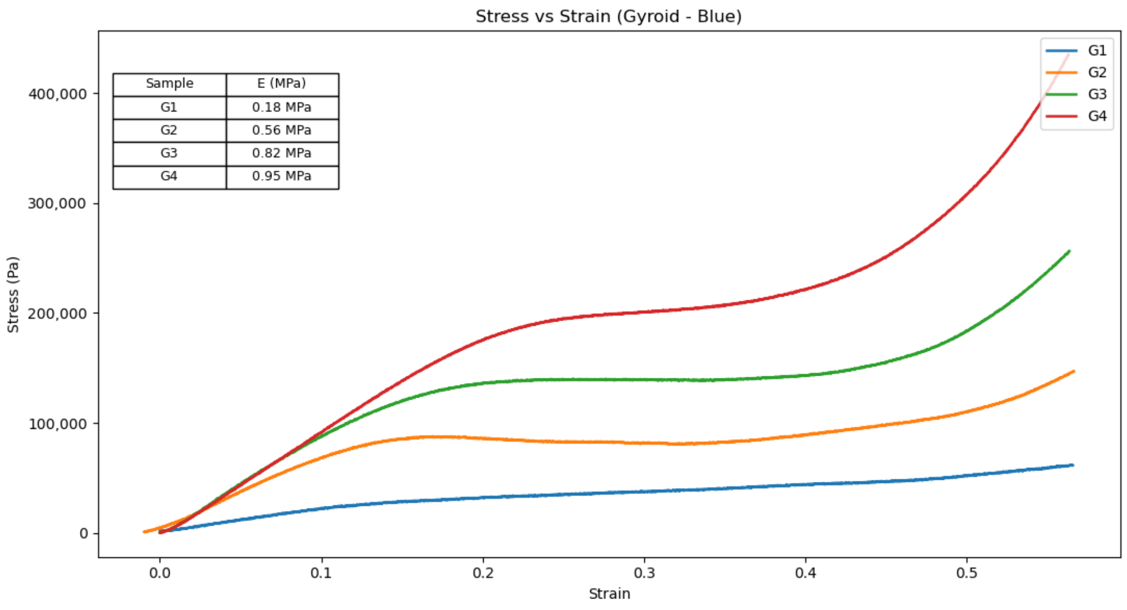
Sample	E (MPa)
R1	1.27 MPa
R2	3.53 MPa
R3	3.77 MPa
R4	5.20 MPa

Figure 81: Stress–strain response for rectilinear infill patterns (95A red TPU)



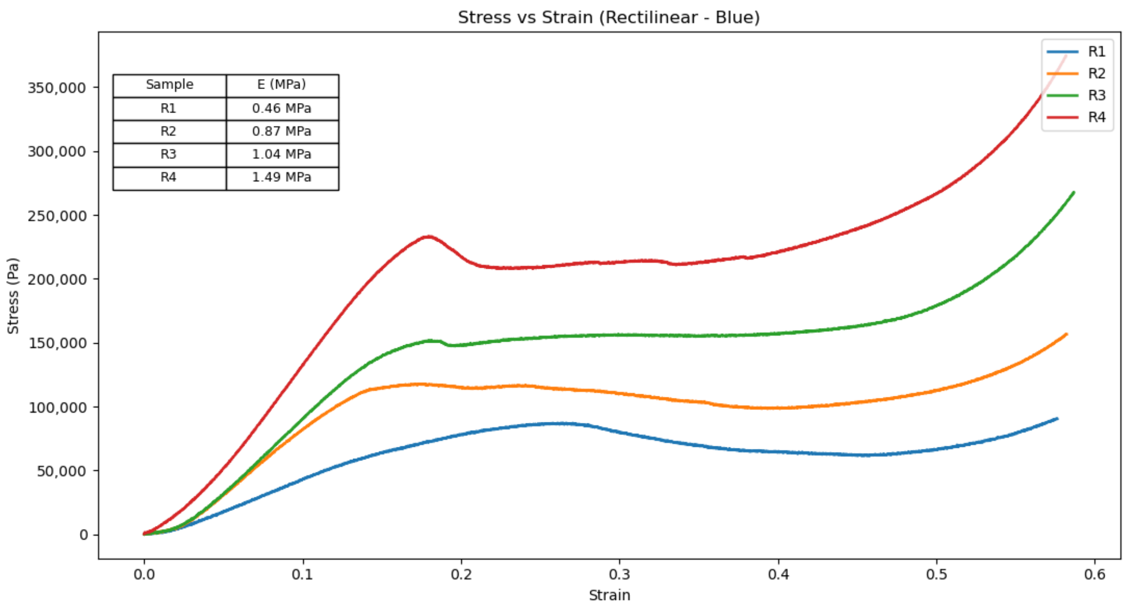
Sample	E (MPa)
D1	0.29 MPa
D2	0.47 MPa
D3	0.68 MPa

Figure 82: Stress–strain response for 3D honeycomb infill patterns (85A blue TPU)



Sample	E (MPa)
G1	0.18 MPa
G2	0.56 MPa
G3	0.82 MPa
G4	0.95 MPa

Figure 83: Stress–strain response for gyroid infill patterns (85A blue TPU)



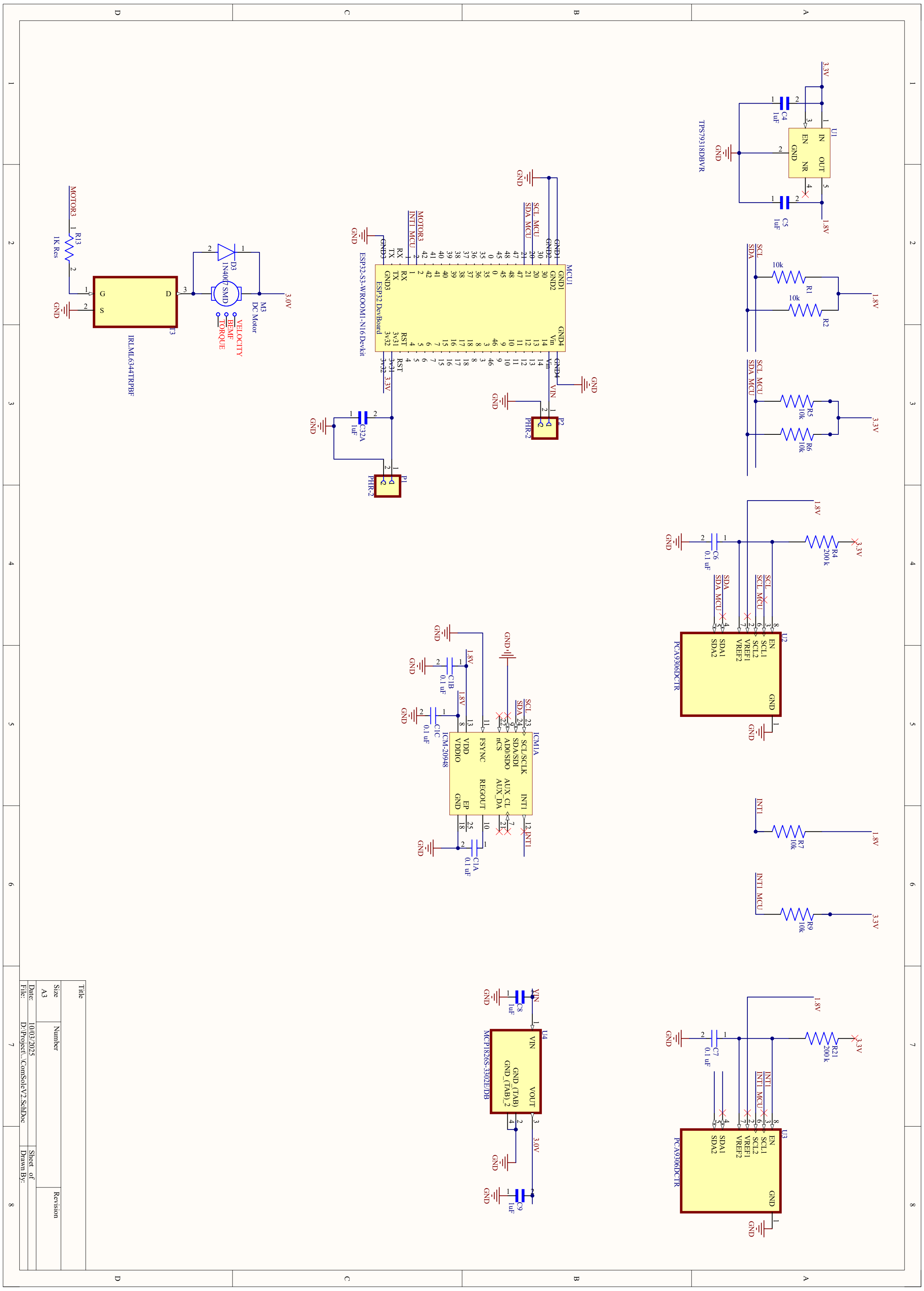
Sample	E (MPa)
R1	0.46 MPa
R2	0.87 MPa
R3	1.04 MPa
R4	1.49 MPa

Figure 84: Stress–strain response for rectilinear infill patterns (85A blue TPU)

Name	Role/Relation	Mode of Interview	Contact Info	Gender	Stage of PD	Summary of Key Findings
Myrtle Huddleston	Wife and Caregiver	In-person	-	Male (Don)	4	- Don has episodes multiple times in the day, no specific timing but will occur most often when he is near countertops and doorways - Takes Cabidopa and Levodopa for PD symptoms but has had issues with hallucenations when given too much - Has tried using canes w/ visual cues proved too be ineffective and potentially dangerous, too deaf for audio cues - deals with gestating gait (smaller and faster shuffling-like walk) - Price would matter alot to them along with ease of use for Myrtles sake - Don wouldnt use it but Mrytle would be interested in an app
Linda Lavine	Wife and Caregiver	Phone Call	979-639-1341	Male (Larry)	4	- Larry has FoG multiple times a day - Also is on various PD medicine, has been moderately helpful - Has used Canes,Walkers,and PT methods to help with symptoms - Price of product and potential for a warranty would be of high importance to them - Linda would be very intersted in an app to see Larrys walk habits - Vibrational Cueing seems like it would be the only method that would work for Larry
Harold Schroeder	Patient	Online Questionaire	2546519577 H.schroeder58@yahoo.com	Male	2	- Frequent FoG episodes, most common in evening - Takes medication to help with symptoms - Auditory and Vibrational stimulus could help for him - Mobile App would be of interest - lightweight, easy to insert, and long battery life were of highest importance
Delise Jung	Patient	Online Questionaire	2816848519 delisega@msn.com	Female	5	- Has used or is using Canes, PT, and Medication for symptoms - Visual and Vibrational cueing would be the most effective for her - interested in toe-tapping/heel tapping exercises for FoG study - uniterested in app - price is of high importance (<100USD) - wants a soft cushioning insole
Janice Crockett	Patient	Online Questionaire	9795741446 crockett1018@gmail.com	Female	3	- Has experienced FoG but occurs very rarely with no noticeable patterns - Has used mobility aids to help with symptoms - Audio and Vibrational Feedback of highest priority - high priority for low cost (<100USD) - app would ve very helpful with stride length,variablity, and step count being important metrics - wants minimal parts visable and easy to insert insole
Anonymous	Patient	Online Questionaire	-	Male	1	- Has experienced FoG but occurs very rarely with no noticeable patterns - using PT and mobility aids - no preference for cueing mode - interested in a mobile app for data collection - placed high priority in almost all customer needs, so would likely end up being an exensive device to meet requirnments

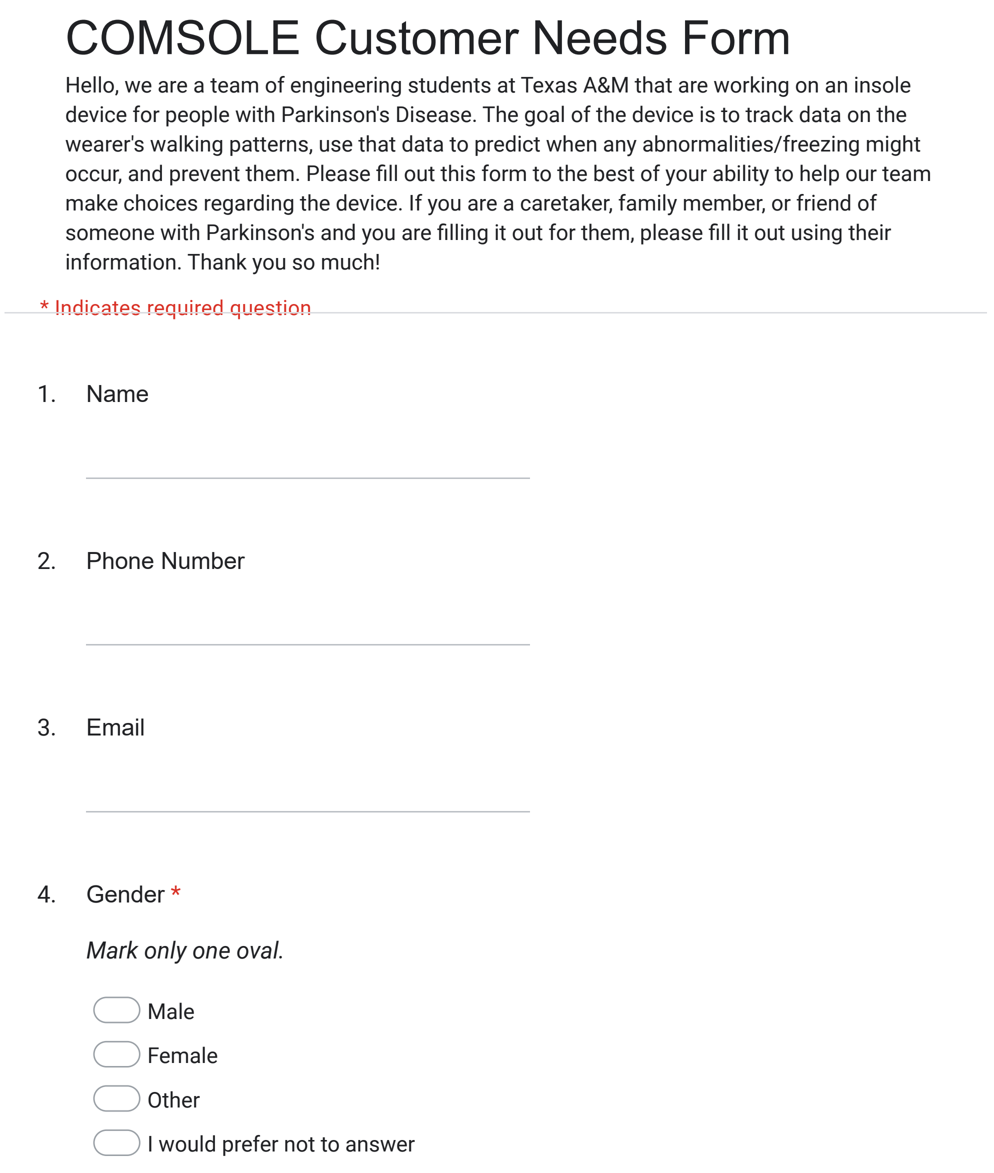
Figure 85: Parkinson's Patient Interviews

A.1 PCB Schematic



B Forms

B.1 Customer Needs Form



COMSOLE Customer Needs Form

Hello, we are a team of engineering students at Texas A&M that are working on an insole device for people with Parkinson's Disease. The goal of the device is to track data on the wearer's walking patterns, use that data to predict when any abnormalities/freezing might occur, and prevent them. Please fill out this form to the best of your ability to help our team make choices regarding the device. If you are a caretaker, family member, or friend of someone with Parkinson's and you are filling it out for them, please fill it out using their information. Thank you so much!

* Indicates required question

1. Name

2. Phone Number

3. Email

4. Gender *

   Mark only one oval.

   - Male
   - Female
   - Other
   - I would prefer not to answer

5. Which stage of Parkinson's are you currently in? *

Mark only one oval.

- 1 (Early)
- 2 (Moderate)
- 3 (Moderate to Severe)
- 4 (Severe)
- 5 (End-Stage)

6. Have you ever experienced Freezing of Gait episodes (a sudden inability to move forward while walking, common symptom of Parkinson's)? *

Mark only one oval.

- Yes
- No

7. If so, how frequently do they occur?

Mark only one oval.

- More than once per day
- Once per day
- A few times per week
- Rarely
- Never

8. What times of day are any mobility symptoms most frequent?

Mark only one oval.

- ◯ Morning
- ◯ Afternoon
- ◯ Evening
- ◯ No noticeable pattern

9. Do you currently use any of the following methods to help with mobility issues? Select all that apply. *

Check all that apply.

- ☐ Physical Therapy
- ☐ Mobility aids (canes, walkers, etc.)
- ☐ Medication
- ☐ Sensory cueing techniques (rhythmic stepping, metronome)
- ☐ Any data collection devices (insoles, belts, etc.)
- ☐ None of the above

10. What types of stimulation cues do you think would help with any symptoms? Select all that apply. *

Check all that apply.

- ☐ Visual (ex. laser device that projects a line to step over)
- ☐ Auditory (ex. metronome or rhythmic beat)
- ☐ Vibrational (ex. vibration in feet)
- ☐ None of these
- ☐ Other: ______________________

11. Would you be willing to receive gentle vibrations from an insole device to help prevent freezing of gait episodes? *

Mark only one oval.

- ⬭ Yes
- ⬭ No
- ⬭ Maybe, if the level of vibration could be customized

12. Which features would you find the most valuable in an insole device? Select up to 3. *

Check all that apply.

- ☐ Data collection on your gait
- ☐ Mobile application to display the above data to you and provide insights
- ☐ Artificial Intelligence (AI) algorithm to detect FOG episodes
- ☐ Cueing technique to help prevent FOG episodes
- ☐ Toe tapping or heel tapping exercises

13. How much would you be willing to spend on an insole that has the above features? *

Mark only one oval.

- ⬭ Less than $100
- ⬭ $100 - $150
- ⬭ $150 - $200
- ⬭ $200 - $250
- ⬭ More than $250

14. Do you think being able to see your own mobility data and track progress in an application on your phone would be helpful? *

Mark only one oval.

- ◯ Very helpful
- ◯ Somewhat helpful
- ◯ Not helpful

15. Select which data you would like the device to be able to track. Select as many as you want. *

Check all that apply.

- ☐ Stride length
- ☐ Stride velocity
- ☐ Average number of steps
- ☐ Walking speed
- ☐ Stride variabilty

Customer Needs

Please rate the following aspects of a potential insole device from 1-5 (1 being low importance and 5 being high importance) based on your opinion.

16. The insole needs to be very lightweight *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

17. Soft and have a lot of cushion *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

18. Firm and provide a lot of support *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

19. Minimal devices/wires/parts visible outside the shoe *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

20. Easy to insert or remove *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

21. Long battery life *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

22. On/Off switch to stop tracking data *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

23. Quick recharge time *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

24. The insole should be able to use some sort of cueing/stimulation technique to help with Freezing of Gait *

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

25. Affordable

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

26. Wear/Water Resistant

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

27. Long-Lasting

Mark only one oval.

	1	2	3	4	5	
Low	◯	◯	◯	◯	◯	High Importance

B.2 Participant Consent Form

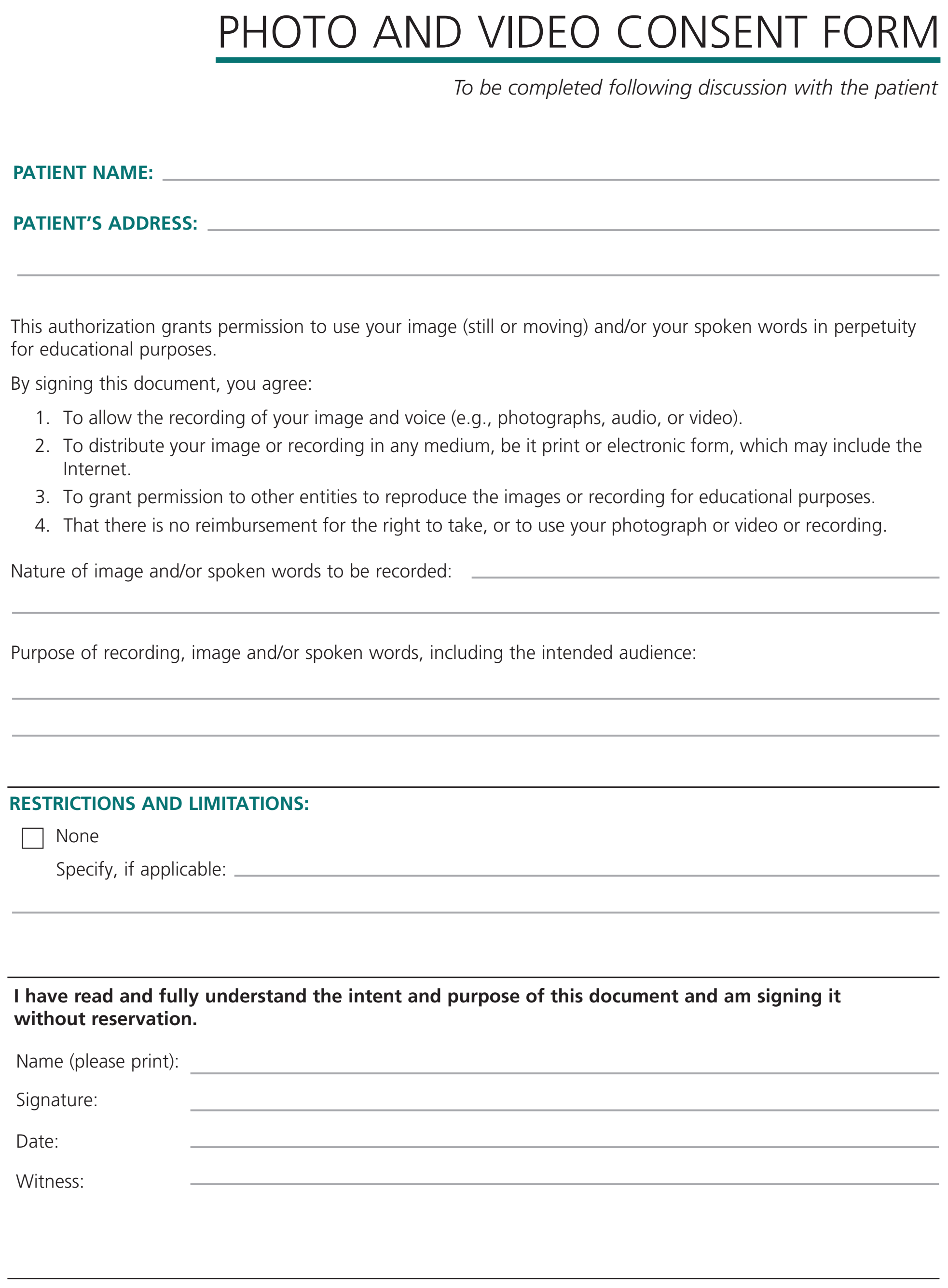


PHOTO AND VIDEO CONSENT FORM

To be completed following discussion with the patient

PATIENT NAME: ______________________________

PATIENT'S ADDRESS: ______________________________

This authorization grants permission to use your image (still or moving) and/or your spoken words in perpetuity for educational purposes.

By signing this document, you agree:

1. To allow the recording of your image and voice (e.g., photographs, audio, or video).
2. To distribute your image or recording in any medium, be it print or electronic form, which may include the Internet.
3. To grant permission to other entities to reproduce the images or recording for educational purposes.
4. That there is no reimbursement for the right to take, or to use your photograph or video or recording.

Nature of image and/or spoken words to be recorded: ______________________________

Purpose of recording, image and/or spoken words, including the intended audience:

RESTRICTIONS AND LIMITATIONS:

☐ None

Specify, if applicable: ______________________________

I have read and fully understand the intent and purpose of this document and am signing it without reservation.

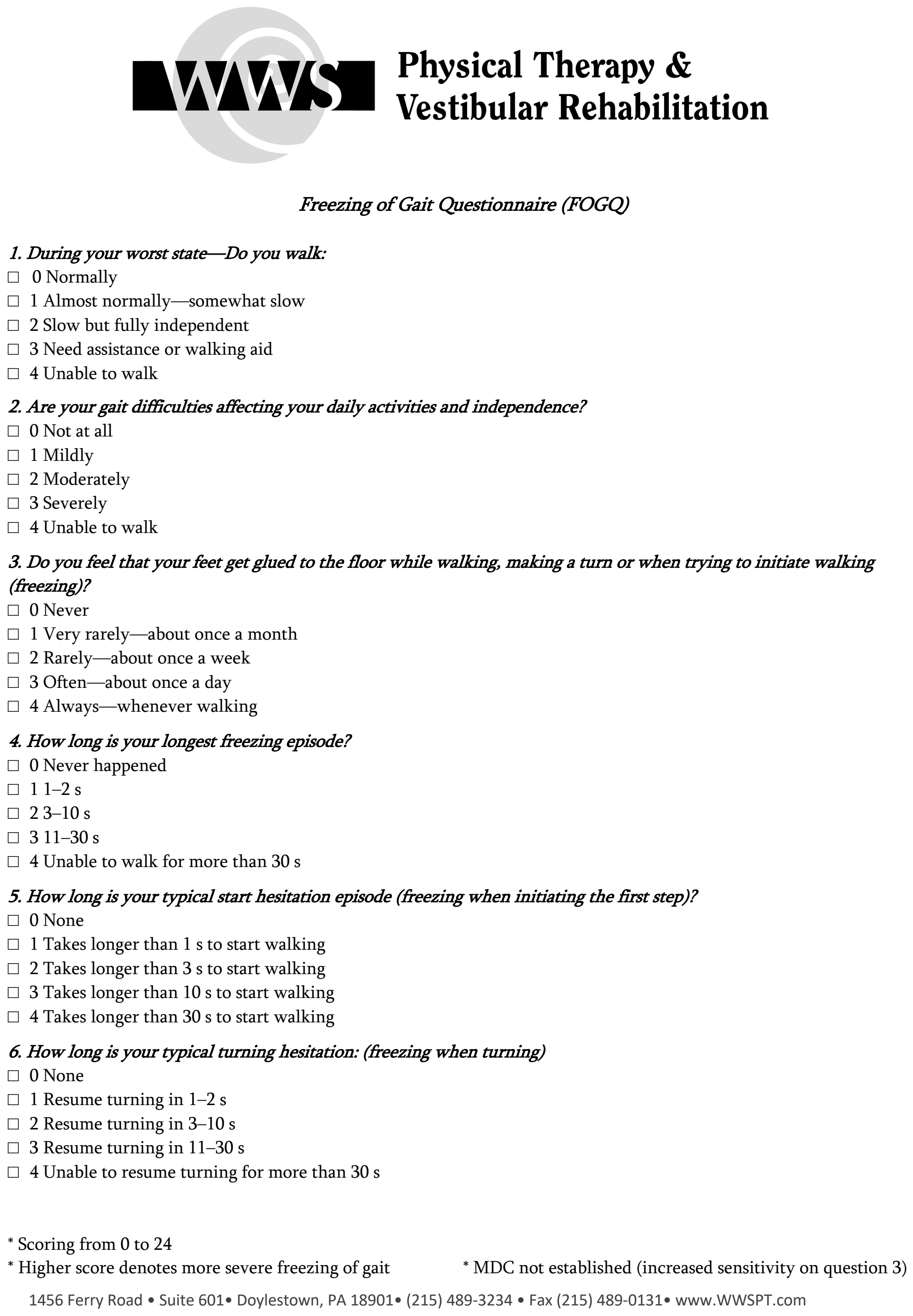
Name (please print): ______________________________

Signature: ______________________________

Date: ______________________________

Witness: ______________________________

B.3 New FoG Questionnaire



Physical Therapy & Vestibular Rehabilitation

Freezing of Gait Questionnaire (FOGQ)

1. During your worst state—Do you walk:
- ☐ 0 Normally
- ☐ 1 Almost normally—somewhat slow
- ☐ 2 Slow but fully independent
- ☐ 3 Need assistance or walking aid
- ☐ 4 Unable to walk

2. Are your gait difficulties affecting your daily activities and independence?
- ☐ 0 Not at all
- ☐ 1 Mildly
- ☐ 2 Moderately
- ☐ 3 Severely
- ☐ 4 Unable to walk

3. Do you feel that your feet get glued to the floor while walking, making a turn or when trying to initiate walking (freezing)?
- ☐ 0 Never
- ☐ 1 Very rarely—about once a month
- ☐ 2 Rarely—about once a week
- ☐ 3 Often—about once a day
- ☐ 4 Always—whenever walking

4. How long is your longest freezing episode?
- ☐ 0 Never happened
- ☐ 1 1–2 s
- ☐ 2 3–10 s
- ☐ 3 11–30 s
- ☐ 4 Unable to walk for more than 30 s

5. How long is your typical start hesitation episode (freezing when initiating the first step)?
- ☐ 0 None
- ☐ 1 Takes longer than 1 s to start walking
- ☐ 2 Takes longer than 3 s to start walking
- ☐ 3 Takes longer than 10 s to start walking
- ☐ 4 Takes longer than 30 s to start walking

6. How long is your typical turning hesitation: (freezing when turning)
- ☐ 0 None
- ☐ 1 Resume turning in 1–2 s
- ☐ 2 Resume turning in 3–10 s
- ☐ 3 Resume turning in 11–30 s
- ☐ 4 Unable to resume turning for more than 30 s

* Scoring from 0 to 24
* Higher score denotes more severe freezing of gait * MDC not established (increased sensitivity on question 3)

1456 Ferry Road • Suite 601• Doylestown, PA 18901• (215) 489-3234 • Fax (215) 489-0131• www.WWSPT.com

C Code

C.1 Firmware Configuration (platformio.ini)

```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:esp32-s3-devkitm-1]
platform = espressif32
board = esp32-s3-devkitm-1
framework = arduino
lib_deps =
        wollewald/ICM20948_WE@^1.2.4
        h2zero/NimBLE-Arduino@^2.2.3
build_flags = -D BT_NAME='"COMSOLE_M10_1"' -D DUTY_CYCLE=10
;monitor_port = COM9
monitor_speed = 115200
```

C.2 Firmware Main Code (main.cpp)

```
/* BLE + 1 IMU left 20250513
 * new: add 3-axis compass data
 * This example demonstrates how to use one ICM20948 IMU with I C
 * and send its data over BLE notifications.
 *
 * The LED on pin 2 indicates the connection status:
 * - Solid ON: Connected
 * - Blinking: Not connected
 */

#include <Wire.h>
#include <NimBLEDevice.h>
#include "ICM20948_WE.h"
#include "fog_inference.h"
#include "features.h"

// Pin definitions for I C bus and motor
#define SDA_PIN 21
#define SCL_PIN 20
#define MOTOR_PIN 2

// BLE UART Service UUID and Characteristics
#define UART_SERVICE_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define UART_TX_UUID "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

#ifndef BT_NAME
#define BT_NAME "BLE-RIGHT"
#endif

#ifndef DUTY_CYCLE
#define DUTY_CYCLE 10 // in milliseconds, corresponds to 100 Hz
#endif

// I C bus instance
TwoWire I2C1 = TwoWire(0); // I C bus

// IMU instance
ICM20948_WE imu1(&I2C1, 0x68); // IMU1 on I C

// BLE variables
NimBLECharacteristic *txChar = nullptr;
bool connected = false;

// IMU initialization status
bool imu1_ok = false;
```

```
// Data processing variables
#define WINDOW_SIZE_MS 2000
#define N_WINDOW_SAMPLES (WINDOW_SIZE_MS / DUTY_CYCLE)
float ax[N_WINDOW_SAMPLES];
float ay[N_WINDOW_SAMPLES];
float az[N_WINDOW_SAMPLES];
float gx[N_WINDOW_SAMPLES];
float gy[N_WINDOW_SAMPLES];
float gz[N_WINDOW_SAMPLES];
float features[48];

int fog_label = 0; // Predicted label

// Function to initialize the IMU
bool setupIMU(ICM20948_WE &imu, const char *label)
{
  if (!imu.init())
  {
    Serial.printf("   ␣Failed␣to␣initialize␣%s\n", label);
    return false;
  }
  imu.setAccRange(ICM20948_ACC_RANGE_2G);
  imu.setGyrRange(ICM20948_GYRO_RANGE_250);
  imu.setAccDLPF(ICM20948_DLPF_6);
  imu.setGyrDLPF(ICM20948_DLPF_6);
  Serial.printf("   ␣%s␣initialized\n", label);
  return true;
}

void setup()
{
  Serial.begin(115200);
  Serial.println("   ␣BLE␣IMU␣notify␣(single␣I C␣bus)");

  // Initialize I C bus
  I2C1.begin(SDA_PIN, SCL_PIN, 100000); // I C with 100kHz
     clock

  // Initialize IMU
  imu1_ok = setupIMU(imu1, "IMU1");

  // Initialize BLE
  NimBLEDevice::init(BT_NAME); // Set BLE device name
  NimBLEDevice::setSecurityAuth(false, false, false);

  // Create BLE server and service
  NimBLEServer *pServer = NimBLEDevice::createServer();
```

```
NimBLEService *service = pServer->createService(UART_SERVICE_UUID);

// You are missing this line:
txChar = service->createCharacteristic(
     UART_TX_UUID,
     NIMBLE_PROPERTY::NOTIFY);

// Manually add CCCD (Client Characteristic Configuration Descriptor) for notifications
txChar->addDescriptor(new NimBLEDescriptor(
     "2902",                                            // CCCD UUID
     NIMBLE_PROPERTY::READ | NIMBLE_PROPERTY::WRITE,  // permissions
     2,                                                 // max length
     txChar                                             // associated characteristic
     ));

// Create RX write characteristic (not used in this example)
service->createCharacteristic(
     "6E400002-B5A3-F393-E0A9-E50E24DCCA9E",
     NIMBLE_PROPERTY::WRITE);

service->start();

// Start BLE advertising
NimBLEAdvertising *adv = NimBLEDevice::getAdvertising();
adv->addServiceUUID(UART_SERVICE_UUID);
adv->setName(BT_NAME); // Set advertised name

NimBLEAdvertisementData ad;
ad.setFlags(0x06);
ad.setName(BT_NAME); // Set advertisement data name
adv->setAdvertisementData(ad);

adv->start();
Serial.println("      BLE advertising started");
Serial.printf("      Device Name: %s\n", BT_NAME);

pinMode(BUILTIN_LED, OUTPUT);
digitalWrite(BUILTIN_LED, HIGH);

// Setup motor for indication
pinMode(MOTOR_PIN, OUTPUT);
digitalWrite(MOTOR_PIN, LOW);
```

```
}

void loop()
{
  static unsigned long lastTime = 0;
  static unsigned long lastBlinkTime = 0;
  static bool lastConnectionStatus = false;
  static bool ledState = false;

  // Check BLE connection status
  connected = NimBLEDevice::getServer()->getConnectedCount() >
     0;

  if (connected)
  {
    if (!lastConnectionStatus)
    {
      Serial.println("      BLE client connected");
      ledState = true;
      digitalWrite(BUILTIN_LED, HIGH);
    }
    // Change frequency to 10 Hz (every 100 ms)
    if (connected && imu1_ok && millis() - lastTime >=
       DUTY_CYCLE)
    {
      lastTime = millis();

      imu1.readSensor();
      xyzFloat acc1 = imu1.getGValues();
      xyzFloat gyr1 = imu1.getGyrValues();
      xyzFloat mag1 = imu1.getMagValues();

      for (int i = N_WINDOW_SAMPLES - 1; i > 0; i--)
      {
        ax[i] = ax[i - 1];
        ay[i] = ay[i - 1];
        az[i] = az[i - 1];
        gx[i] = gx[i - 1];
        gy[i] = gy[i - 1];
        gz[i] = gz[i - 1];
      }

      ax[0] = acc1.x;
      ay[0] = acc1.y;
      az[0] = acc1.z;
      gx[0] = gyr1.x;
      gy[0] = gyr1.y;
      gz[0] = gyr1.z;
      compute_features(ax, ay, az, gx, gy, gz, N_WINDOW_SAMPLES, features);
      fog_label = fog_predict_label(features);

      char payload[128] = {0};
      snprintf(payload, sizeof(payload),
               "A1:%.2f,%.2f,%.2f G1:%.2f,%.2f,%.2f M1:%.2f,%.2f,%.2f",
               acc1.x, acc1.y, acc1.z,
               gyr1.x, gyr1.y, gyr1.z,
               mag1.x, mag1.y, mag1.z);

      txChar->setValue(payload);
      txChar->notify();
      digitalWrite(BUILTIN_LED, HIGH);

      // Vibrate motor if fog_label is 1
      digitalWrite(MOTOR_PIN, fog_label == 1 ? HIGH : LOW);
    }
  }
  else
  {
    if (lastConnectionStatus)
    {
      Serial.println("     BLE client disconnected, restarting advertising");
      delay(200);
      NimBLEDevice::getAdvertising()->start();
      ledState = false;
      digitalWrite(BUILTIN_LED, LOW);
      lastBlinkTime = millis();
    }
    // Blink the LED to indicate waiting for connection
    if (millis() - lastBlinkTime >= 500)
    {
      lastBlinkTime = millis();
      ledState = !ledState;
      digitalWrite(BUILTIN_LED, ledState ? HIGH : LOW);
    }
  }

  lastConnectionStatus = connected;
}
```

C.3 Data Collection Console (Python)

```python
# BLE IMU GUI with Real-Time Multi-Label Markers + Start/Stop Recording
# Supports 5 simultaneous states: FoG, Turning Left, Turning Right, Walking Straight, Stopped

import sys
import asyncio
import threading
from bleak import BleakScanner, BleakClient
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QPushButton, QComboBox, QVBoxLayout, QWidget, QHBoxLayout, QLabel
)
from PyQt5.QtCore import pyqtSignal, QObject, QThread, QTimer
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
import os
import datetime


UART_TX_UUID = "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"


class BLEWorker(QObject):
    imu_data_received = pyqtSignal(str, float, float, float, float, float, float)
    scan_finished = pyqtSignal(str, list)
    connected = pyqtSignal(str)
    disconnected = pyqtSignal(str)
    error = pyqtSignal(str, str)

    def __init__(self, device_id):
        super().__init__()
        self.device_id = device_id
        self.client = None
        self.keep_running = False
        self.loop = asyncio.new_event_loop()
        self.thread = threading.Thread(target=self.loop.run_forever, daemon=True)
        self.thread.start()

    def run_async_task(self, coro):
        asyncio.run_coroutine_threadsafe(coro, self.loop)

    def scan_devices(self):
        self.run_async_task(self._scan_devices())

    def connect_device(self, address):
        self.run_async_task(self._connect_device(address))

    def disconnect_device(self):
        self.run_async_task(self._disconnect_device())

    async def _scan_devices(self):
        try:
            devices = await BleakScanner.discover(timeout=5.0)
            self.scan_finished.emit(self.device_id, devices)
        except Exception as e:
            self.error.emit(self.device_id, f"Scan error: {e}")

    async def _connect_device(self, address):
        try:
            self.client = BleakClient(address)
            await self.client.connect()
            self.connected.emit(self.device_id)
            await self.client.start_notify(UART_TX_UUID, self.handle_notify)
            self.keep_running = True
            while self.client.is_connected and self.keep_running:
                await asyncio.sleep(0.1)
            await self.client.disconnect()
            self.disconnected.emit(self.device_id)
        except Exception as e:
            self.error.emit(self.device_id, f"Connect error: {e}")
            self.disconnected.emit(self.device_id)

    async def _disconnect_device(self):
        self.keep_running = False
        if self.client and self.client.is_connected:
            await self.client.disconnect()
        self.disconnected.emit(self.device_id)

    def handle_notify(self, _, data: bytearray):
        try:
            text = data.decode(errors="ignore").strip()
            acc = gyr = None
            parts = text.split()
            for part in parts:
                if part.startswith("A1:"):
                    acc = [float(x.strip().replace("\x00", "")) for x in part[3:].split(",")]
                elif part.startswith("G1:"):
                    gyr = [float(x.strip().replace("\x00", ""))
                       for x in part[3:].split(",")]
            if acc and gyr:
                self.imu_data_received.emit(self.device_id, *acc
                   , *gyr)
        except Exception:
            pass


class IMUPlotCanvas(FigureCanvas):
    def __init__(self, parent=None):
        self.fig = Figure(figsize=(5, 3))
        self.ax = self.fig.add_subplot(111)
        super().__init__(self.fig)
        self.setParent(parent)
        self.update_counter = 0
        self.update_interval = 10  # plot every 10 samples

        self.y_data = {"BLE-LEFT": [], "BLE-COMSOLE": []}
        self.t_data = {"BLE-LEFT": [], "BLE-COMSOLE": []}
        self.max_seconds = 10
        self.ax.set_title("IMU Acceleration Y")
        self.ax.set_xlabel("Time (s)")
        self.ax.set_ylabel("Acc Y (g)")

        self.line_left, = self.ax.plot([], [], "b-", label="BLE-
           LEFT")
        self.line_right, = self.ax.plot([], [], "r-", label="BLE
           -COMSOLE")
        self.ax.legend()
        self.ax.set_xlim(0, self.max_seconds)
        self.ax.set_ylim(0, 0.6)
        self.start_time = None
        self.recording = False  # <- new flag

        self.data_file = {"BLE-LEFT": None, "BLE-COMSOLE": None}
        self.data_file_path = {"BLE-LEFT": None, "BLE-COMSOLE":
           None}
        self.activity_states = {
            "fog": False,
            "turning_left": False,
            "turning_right": False,
            "walking_straight": False,
            "stopped": False,
        }

    def create_new_files(self):
        """Create new timestamped CSV files for recording."""
        now = datetime.datetime.now()
        dt_str = now.strftime("%Y%m%d_%H%M%S")
        for dev in ["BLE-LEFT", "BLE-COMSOLE"]:
            filename = f"{dev}_{dt_str}.csv"
            self.data_file_path[dev] = os.path.join(os.path.dirname(os.path.abspath(__file__)), filename)
            self.data_file[dev] = open(self.data_file_path[dev], "w", encoding="utf-8")
            self.data_file[dev].write(
                "timestamp,acc_x,acc_y,acc_z,gyr_x,gyr_y,gyr_z,fog,turning_left,turning_right,walking_straight,stopped\n"
            )
            print(f"Recording to: {self.data_file_path[dev]}")

    def set_state(self, label, active):
        """Set boolean state and mark transitions."""
        if label in self.activity_states:
            prev = self.activity_states[label]
            self.activity_states[label] = active
            if prev != active and self.start_time is not None:
                import time
                t = time.time() - self.start_time
                color = "r" if label == "fog" and active else ("g" if active else "m")
                self.ax.axvline(t, linestyle="--", color=color, linewidth=1)
                self.ax.text(
                    t,
                    self.ax.get_ylim()[1],
                    f"{label} {'ON' if active else 'OFF'}",
                    fontsize=8,
                    rotation=90,
                    va="top",
                    ha="left",
                )
                self.draw()

    def update_plot(self, device_id, acc_x, acc_y, acc_z, gyr_x, gyr_y, gyr_z):
        import time
        now = time.time()
        if self.start_time is None:
            self.start_time = now

        t = now - self.start_time
        self.t_data[device_id].append(t)
        self.y_data[device_id].append(acc_y)

        # write only when recording is active
        if self.recording and self.data_file[device_id]:
            row = f"{t:.3f},{acc_x:.6f},{acc_y:.6f},{acc_z:.6f},{gyr_x:.6f},{gyr_y:.6f},{gyr_z:.6f}," + ",".join(
                "1" if self.activity_states[k] else "0" for k in self.activity_states
            ) + "\n"
            self.data_file[device_id].write(row)
            self.data_file[device_id].flush()

        # trim old data
        for dev in ["BLE-LEFT", "BLE-COMSOLE"]:
            while self.t_data[dev] and (self.t_data[dev][-1] - self.t_data[dev][0]) > self.max_seconds:
                self.t_data[dev].pop(0)
                self.y_data[dev].pop(0)

        self.update_counter += 1
        if self.update_counter % self.update_interval != 0:
            return  # skip plotting until enough samples collected

        self.line_left.set_data(self.t_data["BLE-LEFT"], self.y_data["BLE-LEFT"])
        self.line_right.set_data(self.t_data["BLE-COMSOLE"], self.y_data["BLE-COMSOLE"])

        all_t = self.t_data["BLE-LEFT"] + self.t_data["BLE-COMSOLE"]
        if all_t:
            left = max(0, max(all_t) - self.max_seconds)
            right = max(all_t)
            if left == right:
                right += 0.1
            self.ax.set_xlim(left, right)

        all_y = self.y_data["BLE-LEFT"] + self.y_data["BLE-COMSOLE"]
        if all_y:
            y_min, y_max = min(all_y), max(all_y)
            self.ax.set_ylim(y_min - 0.05, y_max + 0.05)

        self.draw()

    def clear_plot(self):
        for dev in ["BLE-LEFT", "BLE-COMSOLE"]:
            if self.data_file[dev]:
                self.data_file[dev].close()
                self.data_file[dev] = None
        self.y_data = {"BLE-LEFT": [], "BLE-COMSOLE": []}
        self.t_data = {"BLE-LEFT": [], "BLE-COMSOLE": []}
        self.ax.clear()
        self.ax.set_title("IMU Acceleration Y")
        self.ax.set_xlabel("Time (s)")
        self.ax.set_ylabel("Acc Y (g)")
        self.ax.legend(["BLE-LEFT", "BLE-COMSOLE"])
        self.draw()


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("BLE IMU GUI (2 Devices)")
        self.resize(800, 600)

        # BLE Controls
        self.scan_btn_left = QPushButton("Scan BLE-LEFT")
        self.device_combo_left = QComboBox()
        self.connect_btn_left = QPushButton("Connect BLE-LEFT")
        self.disconnect_btn_left = QPushButton("Disconnect BLE-
            LEFT")

        self.scan_btn_right = QPushButton("Scan BLE-COMSOLE")
        self.device_combo_right = QComboBox()
        self.connect_btn_right = QPushButton("Connect BLE-
            COMSOLE")
        self.disconnect_btn_right = QPushButton("Disconnect BLE-
            COMSOLE")

        self.plot_canvas = IMUPlotCanvas(self)

        # Recording buttons
        self.start_rec_btn = QPushButton("Start Recording")
        self.stop_rec_btn = QPushButton("Stop Recording")
        rec_layout = QHBoxLayout()
        rec_layout.addWidget(self.start_rec_btn)
        rec_layout.addWidget(self.stop_rec_btn)

        # Activity buttons
        self.buttons = {}
        labels = ["FoG", "Turning Left", "Turning Right", "
            Walking Straight", "Stopped"]
        btn_layout = QHBoxLayout()
        for label in labels:
            btn = QPushButton(label)
            btn.setCheckable(True)
            btn.setMinimumWidth(120)
            btn.clicked.connect(lambda checked, l=label: self.toggle_label(l, checked))
            self.buttons[label] = btn
            btn_layout.addWidget(btn)

        # Layouts
        left_layout = QVBoxLayout()
        left_layout.addWidget(self.scan_btn_left)
        left_layout.addWidget(self.device_combo_left)
        left_layout.addWidget(self.connect_btn_left)
        left_layout.addWidget(self.disconnect_btn_left)

        right_layout = QVBoxLayout()
        right_layout.addWidget(self.scan_btn_right)
        right_layout.addWidget(self.device_combo_right)
        right_layout.addWidget(self.connect_btn_right)
        right_layout.addWidget(self.disconnect_btn_right)

        controls_layout = QHBoxLayout()
        controls_layout.addLayout(left_layout)
        controls_layout.addLayout(right_layout)

        main_layout = QVBoxLayout()
        main_layout.addLayout(controls_layout)
        main_layout.addLayout(rec_layout)
        main_layout.addLayout(btn_layout)
        main_layout.addWidget(self.plot_canvas)

        container = QWidget()
        container.setLayout(main_layout)
        self.setCentralWidget(container)

        self.status_bar = self.statusBar()
        self.status_bar.showMessage("Status: Idle")

        # BLE Workers
        self.ble_worker_left = BLEWorker("BLE-LEFT")
        self.ble_worker_right = BLEWorker("BLE-COMSOLE")
        self.ble_thread_left = QThread()
        self.ble_thread_right = QThread()
        self.ble_worker_left.moveToThread(self.ble_thread_left)
        self.ble_worker_right.moveToThread(self.ble_thread_right)
        self.ble_thread_left.start()
        self.ble_thread_right.start()

        # Signals
        self.start_rec_btn.clicked.connect(self.start_recording)
        self.stop_rec_btn.clicked.connect(self.stop_recording)

        self.scan_btn_left.clicked.connect(lambda: QTimer.
           singleShot(0, self.ble_worker_left.scan_devices))
        self.connect_btn_left.clicked.connect(self.
           start_connect_left)
        self.disconnect_btn_left.clicked.connect(lambda: QTimer.
           singleShot(0, self.ble_worker_left.disconnect_device)
           )
        self.scan_btn_right.clicked.connect(lambda: QTimer.
           singleShot(0, self.ble_worker_right.scan_devices))
        self.connect_btn_right.clicked.connect(self.
           start_connect_right)
        self.disconnect_btn_right.clicked.connect(lambda: QTimer
           .singleShot(0, self.ble_worker_right.
           disconnect_device))

        self.ble_worker_left.scan_finished.connect(self.
           on_scan_finished_left)
        self.ble_worker_right.scan_finished.connect(self.
           on_scan_finished_right)
        self.ble_worker_left.imu_data_received.connect(self.
           plot_canvas.update_plot)
        self.ble_worker_right.imu_data_received.connect(self.
           plot_canvas.update_plot)

        self.devices_left, self.devices_right = [], []

    # —— Recording Control ——
    def start_recording(self):
        self.plot_canvas.create_new_files()
        self.plot_canvas.recording = True
        self.status_bar.showMessage("Recording started.")

    def stop_recording(self):
        self.plot_canvas.recording = False
        for dev in ["BLE-LEFT", "BLE-COMSOLE"]:
            if self.plot_canvas.data_file[dev]:
                self.plot_canvas.data_file[dev].close()
                self.plot_canvas.data_file[dev] = None
        self.status_bar.showMessage("Recording stopped.")

    # —— Label Control ——
    def toggle_label(self, label, active):
        key = label.replace(" ", "_").lower()
        movement_keys = ["turning_left", "turning_right", "walking_straight", "stopped"]

        if key in movement_keys:
            if active:
                for other_label in self.buttons:
                    other_key = other_label.replace(" ", "_").lower()
                    if other_key in movement_keys and other_label != label:
                        self.buttons[other_label].setChecked(False)
                        self.plot_canvas.set_state(other_key, False)
            self.plot_canvas.set_state(key, active)

        elif key == "fog":
            self.plot_canvas.set_state(key, active)

        active_labels = [l for l, b in self.buttons.items() if b.isChecked()]
        state_text = ", ".join(active_labels) if active_labels else "None"
        self.status_bar.showMessage(f"Active: {state_text}")

    # --- BLE Connection Logic ---
    def on_scan_finished_left(self, device_id, devices):
        self.devices_left = sorted(devices, key=lambda d: d.name or "")
        self.device_combo_left.clear()
        for d in self.devices_left:
            self.device_combo_left.addItem(f"{d.name or 'Unknown'} ({d.address})", d.address)
        self.status_bar.showMessage("Status: BLE-LEFT Scan complete")

    def on_scan_finished_right(self, device_id, devices):
        self.devices_right = sorted(devices, key=lambda d: d.name or "")
        self.device_combo_right.clear()
        for d in self.devices_right:
            self.device_combo_right.addItem(f"{d.name or 'Unknown'} ({d.address})", d.address)
        self.status_bar.showMessage("Status: BLE-COMSOLE Scan complete")

    def start_connect_left(self):
        idx = self.device_combo_left.currentIndex()
        if idx < 0:
            return
        address = self.device_combo_left.currentData()
        self.status_bar.showMessage("Connecting BLE-LEFT...")
        QTimer.singleShot(0, lambda: self.ble_worker_left.
           connect_device(address))

    def start_connect_right(self):
        idx = self.device_combo_right.currentIndex()
        if idx < 0:
            return
        address = self.device_combo_right.currentData()
        self.status_bar.showMessage("Connecting BLE-COMSOLE...")
        QTimer.singleShot(0, lambda: self.ble_worker_right.
           connect_device(address))

    def closeEvent(self, event):
        QTimer.singleShot(0, self.ble_worker_left.
           disconnect_device)
        QTimer.singleShot(0, self.ble_worker_right.
           disconnect_device)
        self.ble_worker_left.loop.call_soon_threadsafe(self.
           ble_worker_left.loop.stop)
        self.ble_worker_right.loop.call_soon_threadsafe(self.
           ble_worker_right.loop.stop)
        self.ble_thread_left.quit()
        self.ble_thread_right.quit()
        self.ble_thread_left.wait()
        self.ble_thread_right.wait()
        self.plot_canvas.clear_plot()
        event.accept()


if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

C.4 Machine Learning Training Script (Python)

```python
# FoG detection model (manual train/val/test split)
# run with:
#   python FOG_ML_1.py --mode train
#   python FOG_ML_1.py --mode plot

import os
import glob
import argparse
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import StratifiedShuffleSplit,
    GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    confusion_matrix, classification_report
)

from matplotlib.colors import Normalize
from joblib import dump, load

# Google Colab compatibility
# from google.colab import drive
# drive.mount('/content/drive')

# %cd "/path/to/source"

# folders
TRAIN_DIR = r"Train"
VAL_DIR   = r"Validation"
TEST_DIR  = r"Test"
PREDICTION_DIR = r"Prediction"
os.makedirs(PREDICTION_DIR, exist_ok=True)

ARTIFACT_DIR = "artifacts"
os.makedirs(ARTIFACT_DIR, exist_ok=True)

# window settings
WINDOW_SEC = 2.0
STEP_SEC   = 0.5
FORCE_DT = 0.01
SEED = 2025

```

```
# csv columns I expect
REQUIRED_COLS = ["timestamp","acc_x","acc_y","acc_z","gyr_x","
    gyr_y","gyr_z","FoG"]

# load all csvs in a folder
def load_all_csvs(folder):
    paths = sorted(glob.glob(os.path.join(folder, "*.csv")))
    if not paths:
        raise FileNotFoundError(f"No CSV files found in {folder}
            ")

    dfs = []
    for p in paths:
        df = pd.read_csv(p)
        missing = [c for c in REQUIRED_COLS if c not in df.
            columns]
        if missing:
            raise ValueError(f"{os.path.basename(p)} missing
                columns: {missing}")

        df = df[REQUIRED_COLS].copy()
        df["source_file"] = os.path.basename(p)

        for c in REQUIRED_COLS:
            df[c] = pd.to_numeric(df[c], errors="coerce")

        df.dropna(inplace=True)
        df["FoG"] = (df["FoG"] > 0).astype(int)
        dfs.append(df)

    data = pd.concat(dfs, ignore_index=True)
    return data.sort_values(["source_file", "timestamp"]).
        reset_index(drop=True)


# figure out dt but keep it fixed for consistency
def infer_dt_seconds(df):
    t = df["timestamp"].values
    diffs = np.diff(t)
    if len(diffs) == 0:
        return 0.02
    dt = float(np.median(np.abs(diffs)))
    return dt/1000.0 if dt > 5 else max(dt, 0.01)


# get sliding window indices
def window_indices(n, win, step):
    i = 0
    while i + win <= n:
        yield (i, i + win)
        i += step


# simple FoG label for a window
def window_label(fog_window):
    return 1 if fog_window.mean() >= 0.3 else 0


# basic stats for one axis
def features_from_window(arr):
    feats = {}
    feats["mean"] = float(np.mean(arr))
    feats["std"] = float(np.std(arr))
    feats["min"] = float(np.min(arr))
    feats["max"] = float(np.max(arr))
    feats["rms"] = float(np.sqrt(np.mean(arr**2)))
    feats["energy"] = float(np.sum(arr**2) / len(arr))
    s = np.sign(arr - np.mean(arr))
    feats["zcr"] = float(np.mean(s[:-1] != s[1:])) if len(s) > 1
        else 0.0
    return feats


# build row of features for one window
def build_feature_row(win_df):
    row = {}
    axes = ["acc_x","acc_y","acc_z","gyr_x","gyr_y","gyr_z"]

    for ax in axes:
        f = features_from_window(win_df[ax].values)
        for k, v in f.items():
            row[f"{ax}_{k}"] = v

    pairs = [
        ("acc_x","acc_y"),("acc_x","acc_z"),("acc_y","acc_z"),
        ("gyr_x","gyr_y"),("gyr_x","gyr_z"),("gyr_y","gyr_z")
    ]

    for a, b in pairs:
        if win_df[a].std() > 1e-6 and win_df[b].std() > 1e-6:
            row[f"corr_{a}_{b}"] = float(np.corrcoef(win_df[a],
               win_df[b])[0,1])
        else:
            row[f"corr_{a}_{b}"] = 0.0

    return row
```

```

# generate feature dataset using sliding windows
def make_windowed_dataset(df):
    dt = FORCE_DT
    win = max(1, int(WINDOW_SEC / dt))
    step = max(1, int(STEP_SEC / dt))

    X_rows, y_rows, meta = [], [], []

    for fname, sub in df.groupby("source_file"):
        sub = sub.reset_index(drop=True)
        t0 = sub["timestamp"].iloc[0]

        for s, e in window_indices(len(sub), win, step):
            win_df = sub.iloc[s:e]
            if len(win_df) < win:
                continue

            X_rows.append(build_feature_row(win_df))
            y_rows.append(window_label(win_df["FoG"]))

            meta.append({
                "source_file": fname,
                "t_start": win_df["timestamp"].iloc[0] - t0,
                "t_end": win_df["timestamp"].iloc[-1] - t0
            })

    X = pd.DataFrame(X_rows)
    y = np.array(y_rows)
    meta = pd.DataFrame(meta)

    return X, y, meta


# fit RF with grid search
def fit_rf_with_grid(X_train_s, y_train):
    rf = RandomForestClassifier(
        random_state=42,
        n_jobs=-1,
        class_weight="balanced"
    )

    param_grid = {
        "n_estimators": [400,800,1200],
        "max_depth": [None, 20, 40],
        "min_samples_split": [2, 3, 5],
        "min_samples_leaf": [1, 2]
    }

    cv = StratifiedShuffleSplit(n_splits=5, test_size=0.2,
        random_state=SEED)
    gs = GridSearchCV(rf, param_grid, cv=cv, scoring="f1",
        n_jobs=-1)
    gs.fit(X_train_s, y_train)
    return gs


# pick threshold that gives best F1 on validation
def pick_threshold(model, X_val_s, y_val):
    proba = model.predict_proba(X_val_s)[:,1]
    thresholds = np.linspace(0.05, 0.95, 50)

    scores = [
        f1_score(y_val, (proba >= t).astype(int), zero_division
            =0)
        for t in thresholds
    ]

    return thresholds[int(np.argmax(scores))]


# final metrics
def evaluate(model, X_s, y, threshold):
    proba = model.predict_proba(X_s)[:,1]
    pred = (proba >= threshold).astype(int)

    acc = accuracy_score(y, pred)
    prec = precision_score(y, pred, zero_division=0)
    rec = recall_score(y, pred, zero_division=0)
    f1 = f1_score(y, pred, zero_division=0)

    return acc, prec, rec, f1, pred


# draw confusion matrix
def plot_confusion_matrix(cm, title="Confusion Matrix"):
    fig, ax = plt.subplots(figsize=(4,4))
    im = ax.imshow(cm, cmap="Blues", norm=Normalize(vmin=0, vmax
        =np.max(cm)+1))

    for (i, j), val in np.ndenumerate(cm):
        ax.text(j, i, f"{val}", ha='center', va='center',
            fontsize=14)

    ax.set_xlabel("Predicted")
    ax.set_ylabel("Actual")
    ax.set_title(title)
    fig.colorbar(im)
    plt.tight_layout()
    return fig


# bar chart for accuracy, precision, recall, f1
def plot_metrics_bar(acc, prec, recall, f1, title="Performance")
    :
    fig, ax = plt.subplots(figsize=(6,4))
    metrics = ["Accuracy", "Precision", "Recall", "F1"]
    values = [acc, prec, recall, f1]

    ax.bar(metrics, values)
    ax.set_ylim(0,1.05)

    for i, v in enumerate(values):
        ax.text(i, v + 0.03, f"{v:.2f}", ha='center', fontsize
           =12)

    ax.set_title(title)
    plt.tight_layout()
    return fig


# top feature importance
def plot_feature_importance(model, feature_names, top_n=20):
    importances = model.feature_importances_
    idx = np.argsort(importances)[-top_n:]

    fig, ax = plt.subplots(figsize=(8,6))
    ax.barh(np.array(feature_names)[idx], importances[idx])
    ax.set_title("Top Feature Importances")
    ax.set_xlabel("Importance")
    plt.tight_layout()
    return fig


# timeline plot for FoG predictions
def plot_fog_timeline(meta, y_true, y_pred):
    figures = []

    for file, group in meta.groupby("source_file"):
        fig, ax = plt.subplots(figsize=(10,3))

        t = group["t_start"].values
        yt = y_true[group.index]
```

```
            yp = y_pred[group.index]

            ax.step(t, yt, where="post", label="True FoG", linewidth
               =2)
            ax.step(t, yp, where="post", label="Predicted FoG",
                    linestyle="--", linewidth=2, alpha=0.9)

            ax.set_yticks([0,1])
            ax.set_yticklabels(["No FoG", "FoG"])
            ax.set_xticks(np.arange(min(t), max(t)+0.1, 0.1))
            ax.set_title(f"FoG Timeline  {file}")
            ax.set_xlabel("Time (s)")
            ax.legend()
            ax.grid(True, axis='x', alpha=0.3)
            plt.tight_layout()

            figures.append(fig)

        return figures


    # full training process
    def run_train_mode():
        print("\nLoading data...")

        train_df = load_all_csvs(TRAIN_DIR)
        val_df   = load_all_csvs(VAL_DIR)
        test_df  = load_all_csvs(TEST_DIR)

        X_train, y_train, meta_train = make_windowed_dataset(
           train_df)
        X_val, y_val, meta_val       = make_windowed_dataset(val_df)
        X_test, y_test, meta_test    = make_windowed_dataset(test_df
           )

        feature_path = os.path.join(ARTIFACT_DIR, "feature_names.csv
           ")

        if os.path.exists(feature_path):
            feature_names = pd.read_csv(feature_path, header=None)
               [0].tolist()
            X_train = X_train.reindex(columns=feature_names)
            X_val   = X_val.reindex(columns=feature_names)
            X_test  = X_test.reindex(columns=feature_names)
        else:
            feature_names = list(X_train.columns)

        # clean
        X_train.replace([np.inf, -np.inf], np.nan, inplace=True)
        X_val.replace([np.inf, -np.inf], np.nan, inplace=True)
        X_test.replace([np.inf, -np.inf], np.nan, inplace=True)
        X_train.dropna(inplace=True)
        X_val.dropna(inplace=True)
        X_test.dropna(inplace=True)

        y_train = y_train[:len(X_train)]
        y_val   = y_val[:len(X_val)]
        y_test  = y_test[:len(X_test)]

        meta_train = meta_train.iloc[:len(X_train)]
        meta_val   = meta_val.iloc[:len(X_val)]
        meta_test  = meta_test.iloc[:len(X_test)]

        # scale
        scaler = StandardScaler()
        X_train_s = scaler.fit_transform(X_train)
        X_val_s   = scaler.transform(X_val)
        X_test_s  = scaler.transform(X_test)

        print("Training␣model...")
        gs = fit_rf_with_grid(X_train_s, y_train)
        model = gs.best_estimator_

        threshold = pick_threshold(model, X_val_s, y_val)

        # metrics
        train_acc, train_prec, train_rec, train_f1, train_pred =
           evaluate(model, X_train_s, y_train, threshold)
        val_acc, val_prec, val_rec, val_f1, val_pred = evaluate(
           model, X_val_s, y_val, threshold)
        test_acc, test_prec, test_rec, test_f1, test_pred = evaluate
           (model, X_test_s, y_test, threshold)

        cm_test = confusion_matrix(y_test, test_pred)

        # plots
        plot_confusion_matrix(cm_test, title="TEST␣    ␣Confusion␣
           Matrix")
        plot_metrics_bar(test_acc, test_prec, test_rec, test_f1,
           title="TEST␣    ␣Performance")
        plot_feature_importance(model, list(X_train.columns))
        plot_fog_timeline(meta_test, y_test, test_pred)

        # save per-file predictions
        print("Saving␣prediction␣CSVs...")
        for file_name, group in meta_test.groupby("source_file"):

            original_path = os.path.join(TEST_DIR, file_name)
            raw_df = pd.read_csv(original_path)

            raw_df["timestamp"] = pd.to_numeric(raw_df["timestamp"],
                errors="coerce")
            t0 = raw_df["timestamp"].iloc[0]
            raw_df["timestamp"] = raw_df["timestamp"] - t0

            raw_df["predicted_fog"] = 0
            raw_df["predicted_proba"] = 0.0

            idxs = group.index
            file_meta = meta_test.loc[idxs]
            file_pred = test_pred[idxs]
            file_proba = model.predict_proba(X_test_s[idxs])[:,1]

            for (_, row_meta), yp, pr in zip(file_meta.iterrows(),
               file_pred, file_proba):
                t_start = row_meta["t_start"]
                t_end = row_meta["t_end"]
                mask = (raw_df["timestamp"] >= t_start) & (raw_df["
                   timestamp"] <= t_end)
                raw_df.loc[mask, "predicted_fog"] = int(yp)
                raw_df.loc[mask, "predicted_proba"] = float(pr)

            save_name = file_name.replace(".csv", "_predictions.csv"
               )
            save_path = os.path.join(PREDICTION_DIR, save_name)
            raw_df.to_csv(save_path, index=False)

        # save model + artifacts
        dump(model, os.path.join(ARTIFACT_DIR, "best_fog_model.
           joblib"))
        dump(scaler, os.path.join(ARTIFACT_DIR, "scaler.joblib"))
        pd.Series(X_train.columns).to_csv(os.path.join(ARTIFACT_DIR,
            "feature_names.csv"), index=False, header=False)

        np.save(os.path.join(ARTIFACT_DIR, "y_val.npy"), y_val)
        np.save(os.path.join(ARTIFACT_DIR, "y_test.npy"), y_test)
        np.save(os.path.join(ARTIFACT_DIR, "proba_val.npy"), model.
           predict_proba(X_val_s)[:,1])
        np.save(os.path.join(ARTIFACT_DIR, "proba_test.npy"), model.
           predict_proba(X_test_s)[:,1])

        with open(os.path.join(ARTIFACT_DIR, "threshold.txt"), "w")
           as f:
            f.write(str(threshold))

        print("Training complete.")
        plt.show()


    # quick plot mode
    def run_plot_mode():
        # meta_val = pd.read_csv(os.path.join(ARTIFACT_DIR, "
            meta_val.csv"))
        # meta_test = pd.read_csv(os.path.join(ARTIFACT_DIR, "
            meta_test.csv"))

        y_val = np.load(os.path.join(ARTIFACT_DIR, "y_val.npy"))
        y_test = np.load(os.path.join(ARTIFACT_DIR, "y_test.npy"))
        proba_val = np.load(os.path.join(ARTIFACT_DIR, "proba_val.
            npy"))
        proba_test = np.load(os.path.join(ARTIFACT_DIR, "proba_test.
            npy"))

        with open(os.path.join(ARTIFACT_DIR, "threshold.txt"), "r")
            as f:
            thr = float(f.read().strip())

        val_pred = (proba_val >= thr).astype(int)
        test_pred = (proba_test >= thr).astype(int)

        print(classification_report(y_val, val_pred, zero_division
            =0))
        print(classification_report(y_test, test_pred, zero_division
            =0))

        threshold = pick_threshold(model, X_val_s, y_val)

        # metrics
        train_acc, train_prec, train_rec, train_f1, train_pred =
            evaluate(model, X_train_s, y_train, threshold)
        val_acc, val_prec, val_rec, val_f1, val_pred = evaluate(
            model, X_val_s, y_val, threshold)
        test_acc, test_prec, test_rec, test_f1, test_pred = evaluate
            (model, X_test_s, y_test, threshold)

        cm_test = confusion_matrix(y_test, test_pred)

        plot_confusion_matrix(cm_test, title="TEST    Confusion 
            Matrix")
        plot_metrics_bar(test_acc, test_prec, test_rec, test_f1,
            title="TEST    Performance")
        plot_feature_importance(model, list(X_train.columns))
        plot_fog_timeline(meta_test, y_test, test_pred)

        plt.show()


    parser = argparse.ArgumentParser()
    parser.add_argument("--mode", choices=["train", "plot"], default
        ="train")
    args = parser.parse_args()

    if args.mode == "train":
        run_train_mode()
    else:
        run_plot_mode()
```