Knee Injury Rehabilitation with Real-Time Feedback: The iKneed Smart Knee Brace Add-On

By Elizabeth Finley, Maximilian Miesen, and Sophia Chen

University of Southern California, 2025

A Final Project for BME 405: Senior Projects in the USC Viterbi Alfred E. Mann Department of Biomedical Engineering

Elizabeth Finley	Biomedical Engineering (Mechanical Emphasis) Bachelor of Science & Mechanical Engineering Master of Science
Maximilian Miesen	Biomedical Engineering Bachelor of Science & Product Development Engineering Master of Science
Sophia Chen	Biomedical Engineering (Molecular & Cellular Emphasis) Bachelor of Science

TITLE: Knee Injury Rehabilitation with Real-Time Feedback: The iKneed Smart Knee Brace Add-On

AUTHORS: Elizabeth Finley, Maximilian Miesen, and Sophia Chen

SUPERVISOR: Dr. John Mai

Abstract

This project presents the development of a smart knee brace add-on designed to monitor knee injury rehabilitation by tracking knee motion during exercises. The final model of the device utilizes two inertial measurement units (IMUs), a BNO08X IMU and a BNO055 IMU, positioned on the thigh and shin to capture orientation data. The knee joint angle is then calculated by finding the relative difference from the original calibration at 130 degrees. The system is built around an Adafruit Feather M0 Wifi microcontroller, which gathers data and outputs it using the Serial Monitor in the Arduino IDE. The final prototype will involve transmission of data using Wifi to a mobile app. The app will then provide live knee angle feedback, graphical data visualization, and session logging for patients. The product would then be attached to the knee brace using velcro. This low-cost, portable knee brace add-on has the potential to improve remote physical therapy monitoring and patient engagement during strenuous rehabilitation.

Introduction

Clinical Problem/Unmet Need

Currently, in knee injury rehabilitation, there is a gap between current treatment and recovery options. There are high rates of reinjury, so a personalized and data-driven rehabilitation plan to reduce reinjury risk is necessary. Rehabilitation progress is often tracked subjectively or through basic strength/mobility and patients infrequently visit clinics to monitor progress. Wearable smart knee brace technology that objectively measures joint angles and rehabilitation exercises would address the issue of subjective tracking. There is a need for a smart, lightweight, wearable system that provides real-time feedback for patients and clinicians during rehabilitation exercises to make sure that they are being performed correctly and consistently throughout their rehabilitation journey.

The primary stakeholders are patients recovering from a knee injury or surgery, physical therapists, and orthopedic surgeons. The secondary stakeholders are insurance providers, healthcare systems, and regulatory bodies (e.g. the FDA). The patients would receive real-time feedback on their rehabilitation exercises and the physical therapists can keep track of progress. The orthopedic surgeon helps oversee recovery and would appreciate the objective metrics provided by the device, which can complement clinical evaluations. The healthcare systems will be interested because this product may reduce costs by minimizing in-clinic visits. Furthermore, insurance providers may cover the device as a cost-effective solution to enhance recovery and reduce long-term expenses. The FDA will be focused on seeing if this device passes regulatory approval and sets a standard for future smart knee brace add-ons.

The typical user is a patient aged 18-70 recovering from a knee injury, the most common being an ACL injury. The user's needs include immediate corrective feedback, easy setup without professional assistance, a lightweight device, accurate data tracking and simple reporting for clinicians, and an affordable or insurance-reimbursable device. There are ~400,000 ACL reconstructions annually in the US and ~700,000 total knee replacements annually in the US [1][2]. In addition to just ACL injuries, approximately 10 million patients in the US visit doctors annually for knee injuries [3].

In addition to the statistics on how many people suffer knee injuries, there is a large market out there for this product. One area this device fits in is the wearable medical device market. In 2023, the wearable medical device market was valued at around $105.1 billion. This market is projected to grow at a compound annual growth rate of 15.9%, theoretically reaching a market value of $391.3 billion in 2032 [4]. This product also fits in the knee brace market. In 2024, the knee brace market was valued at around $1.19 billion. This market is projected to grow at a compound annual growth rate of 7.9% for a market value of $1.85 billion in 2030 [5]. There is a ton of opportunity in these markets and the smart knee brace add-on fits right in.

The smart knee brace add-on would likely be a class II medical device under the US FDA system because it is designed to monitor patient motion and provide real-time feedback during rehabilitation, which has a direct impact on treatment decisions. This is non-invasive with low to moderate risk. As for the regulatory pathway, it would require an FDA 510(k) Premarket Notification, with substantial equivalence to existing cleared devices. Standards to meet include Quality Management System for medical devices (ISO 13485), Application of Risk Management of Medical Devices (ISO 14971), safety and performance of medical electrical equipment (IEC 60601-1), and HIPAA compliance if patient data is transmitted or stored [6][7][8]. Key regulatory constraints are the following: electrical safety and EMI shielding to prevent interference with other devices, accuracy and reliability validation for sensors and feedback, and human factors/usability testing to ensure patients can understand and use feedback properly [9].

A human trial for the iKneed smart knee brace add-on would start with regulatory approval. The following would have to be submitted to the Institutional Review Board (IRB): study protocol with objectives, procedures, and risks, participant consent form, data privacy measures, and risk analysis. Most student trials fall under “non-significant risk (NSR)” devices, which would streamline IRB review. FDA approval would not be required unless the device was specifically intended for therapeutic, diagnostic, or commercial use [10]. FDA Design Controls (21 CFR Part 820.30) can be referenced for structure, but regardless, formal FDA submission is not applicable at the prototyping stage [11]. The next step would be participant engagement. In the consent form distributed to volunteers, there would be the following: purpose for engineering research (not medical diagnosis or treatment), the duration of the trial (e.g. 30-60 minutes), risks (minimal in this case: skin irritation and discomfort), voluntary participation and withdrawal rights, an explanation that there will be no direct benefits or compensation, and a disclaimer that data would be anonymized and securely stored [12]. The trial protocol would start with screening, then a brace fit check, and tasks would finally be performed such as sitting, standing, stair climbing, and lunges. Data would be collected from the device and then there would be a debrief with the volunteer about comfort and general feedback using the iKneed knee brace. Next is the data handling plan, using anonymized IDs and secure storage of data, with data use only for academic analysis and project reports (not shared publicly) [13]. Unanticipated events would have to be managed such as participants dropping out, device malfunction, and sensor error or data loss.

Predicate Devices/Competitive Products

Table 1. There are a few notable predicate and competitive devices on the market. They all have different advantages and disadvantages, but their respective FDA approvals demonstrate that there is justification for the project idea presented in this report.

Predicate Device	Pros	Cons	Relevance to iKneed
CyMedica Orthopedics' Intellihab System (K210604) [14]	• Neuromuscular electrical stimulation to strengthen the quadriceps and provide relief for knee pain • Mobile app connectivity via RF	• No statistically significant difference found in pain relief (VAS Nominated Activity pain score) • Limited data on motion tracking	Brace is similar to a wrap and uses RF communications to connect to a mobile app
CyMedica Orthopedics' Motive Knee Wrap (K220738) [15]	• Neuromuscular electrical stimulation • Connects to a mobile app to provide personalized training program and progress tracking	• Lacks a digital goniometer so it cannot measure range of motion	Brace that connects to a smartphone app for personalized training programs and progress tracking
Zimmer Biomet's Persona IQ [16]	• "Smart" knee implant that collects data on patient gait metrics for rehabilitation	• Invasive (knee implant)	Data collection on gait metrics

The team's proposed solution differs from existing predicate devices due to its specific focus on rehabilitation for knee injuries, such as ACL tears, by providing real-time feedback on knee angles during prescribed exercises. Unlike CyMedica Orthopedics' IntelliHab System and Motive Knee Wrap, which use neuromuscular electrical stimulation for quadriceps strengthening and pain relief, this report's device employs IMUs to monitor and guide proper movement patterns, ensuring patients perform exercises correctly and avoid overexertion. Additionally, while Zimmer Biomet's Persona IQ collects gait metrics for rehabilitation, this knee brace add-on is designed as an external product rather than an implant, making it non-invasive and accessible for at-home use. This distinct focus on real-time movement monitoring and exercise guidance sets this team's product apart from existing predicate devices while leveraging their approved technologies, such as wireless connectivity and mobile app integration, to support regulatory approval.

Based on our primary and secondary research, market research, and predicate/competitive device exploration, the knee brace add-on prototype will have these features:

1) Assist in rehabilitation of knee injuries
2) Monitor biomechanical measurements and collect data
3) Provide advice on form correction
4) Allow remote feedback and monitoring from clinicians

The purpose of iKneed is to address the general need for gait analysis of rehabilitation exercises. With these four features, the subject can adjust their rehabilitation progress in real-time and receive feedback from clinicians.

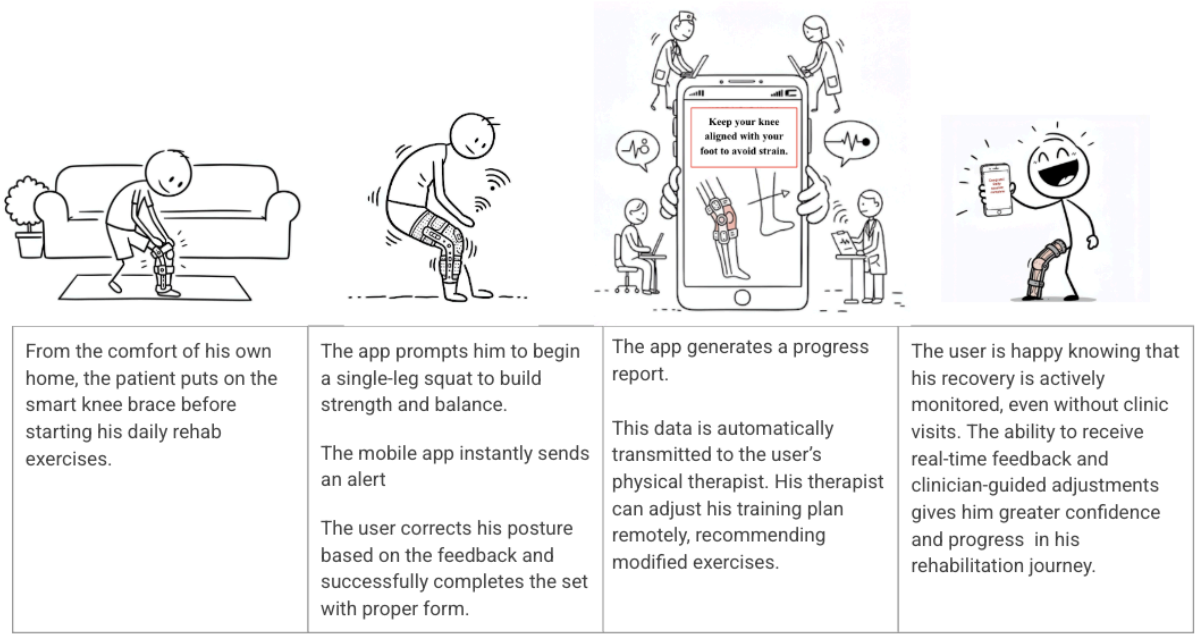

Figure 1. Example use case of a patient using the device.

Materials and Methods

The problem is that patients tend not to continue rehabilitation exercises at home, they don't frequently go in-person to physical therapy, or they aren't properly doing the exercises if they do them at home. There is a need for a device that provides real-time feedback on doing a rehabilitation exercise correctly and that sends alerts to a primary care provider or physical therapist that the patient is staying on track with their rehabilitation plan.

iKneed has a few notable functional requirements derived from user needs. These requirements are necessary for our device to successfully work. iKneed also has non-functional requirements that will make the device easier to use and improve overall functionality.

Functional requirements:

1) Device is light (add-on is significantly lighter than the knee brace): <10 lbs
2) Allows normal flexion and extension: 0-180º range of motion
3) Low profile casing: 34 x 17 x 18 mm (Box for BNO08X IMU in Figure 5), 35.50 x 22 x 13 mm (Box for BNO055 IMU in Figure 4), and 50 x 50 x 30 mm (Circle Center for PCB Board and LiPo Battery in Figure 6)

Here is the ideal functional decomposition for how the solution would work. Human movement and IMU sensor data would be the inputs to the microcontroller, which would then use signal processing to output the knee angle in real time.

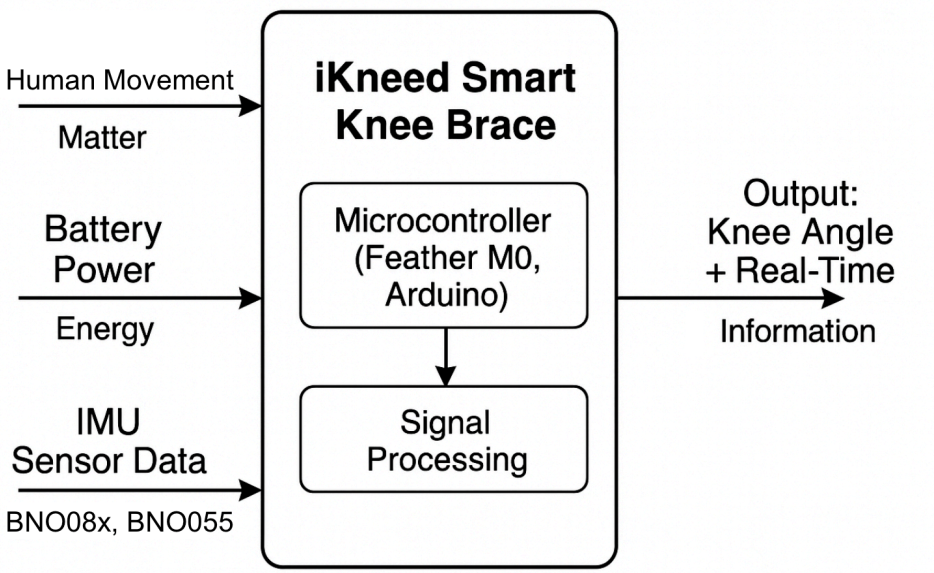

Figure 2. Functional decomposition: it senses movement, acquires data, processes signals, filters said data, and outputs results.

Concept Selection and Screening Process

Three concepts were developed based on the criteria of sensing accuracy, integration effort, power efficiency, communication, ease of use, and form factor. Sensing accuracy was given the most weight as it is the most important characteristic of the sensor for relevant knee angle readings.

The below figures help show why the team went with the 2 IMU model. The 2 IMU model had the highest sensing accuracy, being the best option to calculate the knee angle correctly. While it didn't rate as high in the other areas, it was the best path to accomplishing the goal of this project.

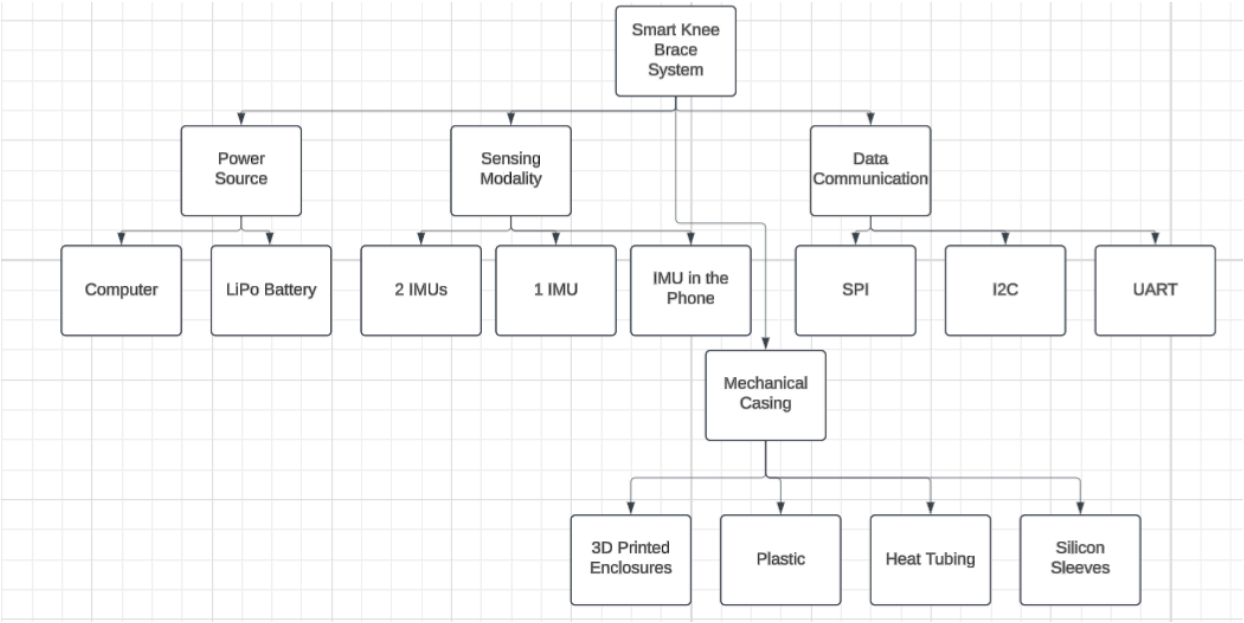

Figure 3. The concept classification tree breaks down the essential subfunctions of the smart knee brace system. It shows the solution options for each subproblem.

Following the concept classification tree, the team made a decision matrix focused on the subfunction of sensing modality. While data communication, power source, and mechanical casing are important, they all took a backseat to sensing modality. Without the correct structure for sensing modality, this project could not exist nor accomplish its goal.

Table 2. In this decision matrix, 1 is the worst and 5 is the best. The BNO08x IMU, Concept A, scored highest because of its robust sensor fusion (CEVA SH-2) and advanced 3D orientation outputs. It has a fast output rate and integrates with Arduino well.

Criteria	**Weight**	**Concept A (BNO08x + BNO055) [25][40]**	**Concept B (BNO085) [18]**	**Concept C (IMU in the Phone) [19]**
Sensing Accuracy	0.25	5	2	1
Integration Effort	0.2	2	3	3
Power Efficiency	0.15	3	3	4
Communication	0.1	4	4	4
Ease of Use	0.15	2	3	3
Form Factor	0.15	2	2	3
Weighted Total	-	3.1	2.7	2.75

A single IMU was not chosen because it alone cannot compute the relative angle between the thigh and the shin. Although a magnet and IMU could have been used in conjunction, magnet-based orientation is highly unstable. Magnetic fields can decay rapidly with distance and vary nonlinearly. Any movement can cause distortion due to rotation and environmental interference from other metals and electronics could potentially alter the readings. A magnetic solution would also require precise calibration and the IMUs already experience sensor drift problems [17].

Table 3. Concept A was selected because it's fully functional, testable, and modular. It is simple, accurate, and can be prototyped and debugged quickly. Concept B was close behind, but using a single IMU leads to a lack of complete understanding of the knee angle. In fact, it relies on guessing and has a lack of consistency in measurements. In addition, Concept C was not chosen as it relies on the patient having an updated phone. Many of the technologies similar to Concept C are also outdated and are constantly updated.

Subfunctions	**Concept A (2 IMUs) [25][40]**	**Concept B (Single IMU) [18]**	**Concept C (IMU in the Phone) [19]**
Sensing Modality	Dual	Inertial and Single	Dual
IMU Model	BNO08x/BNO055	BNO085	Onboard
Sensor Fusion	Onboard (SH-2)	Onboard (SH-2)	In Phone Software
Communication	I2C and SPI	I2C/SPI	BLE
Microcontroller	Arduino Feather M0	Arduino Feather M0	SoC (System on a Chip)
Mounting Method	3D Printed	3D Printed	Sleeve-integrated
Power Source	1000 mAh LiPo	1000 mAh LiPo	Powered by Phone
Output Interface	Serial Monitor/Wifi App	Serial Monitor/Wifi App	App GUI

In addition to the 2 IMU model (the Final Model), the team also built a test model, which consisted of 1 IMU plus a Flex Sensor. The rationale behind doing a test model is explained more in the Lab Notebook section. However, this model was chosen due to concerns that the 2 IMU model would not be finished in time. Using this model, one can find the knee angle, but it is not as accurate as the 2 IMU model. It relies heavily on educated guessing as the IMU and the flex sensor are supposed to act as backup for each other's result. With this model, the knee angle is not actually calculated, it is just predicted and this prediction can vary widely in accuracy.

Engineering Drawings

Throughout the design process, 3D-printed parts were used to act as stand-ins for the final material casing. These parts provided the mechanical hardware for the prototype throughout testing. Here are the Engineering Drawings for the three 3D-printed parts used in the prototype.

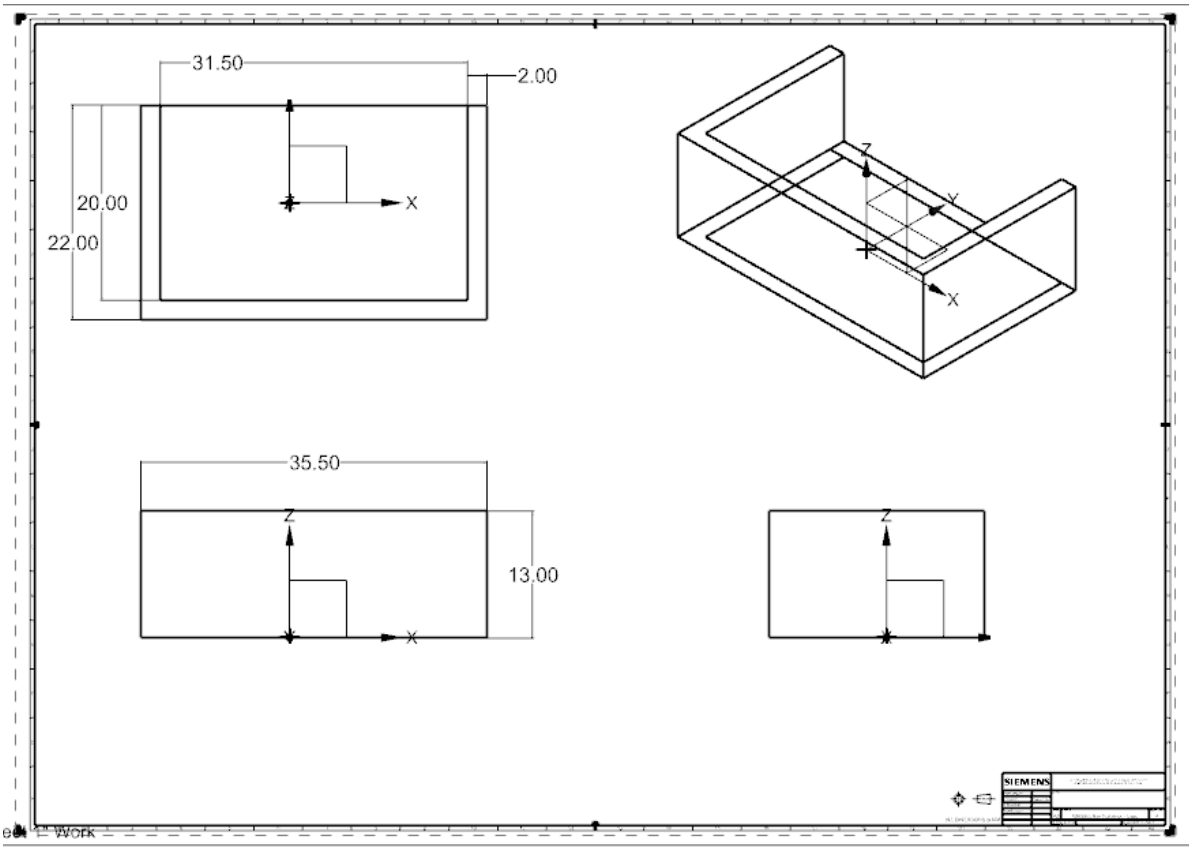

Figure 4. The Engineering Drawing for the Box to go over the BNO055 IMU. The dimensions are in the unit of millimeters.

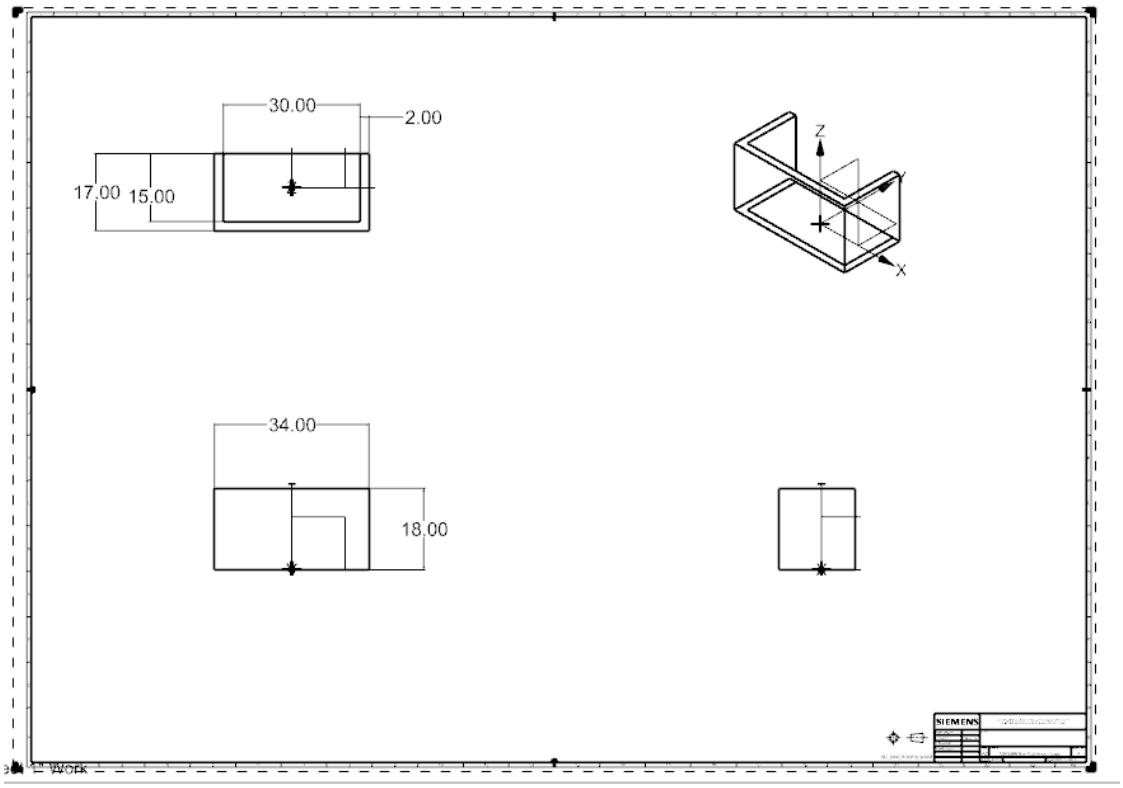

Figure 5. The Engineering Drawing for the Box to go over the BNO08X IMU. The dimensions are in the units of millimeters.

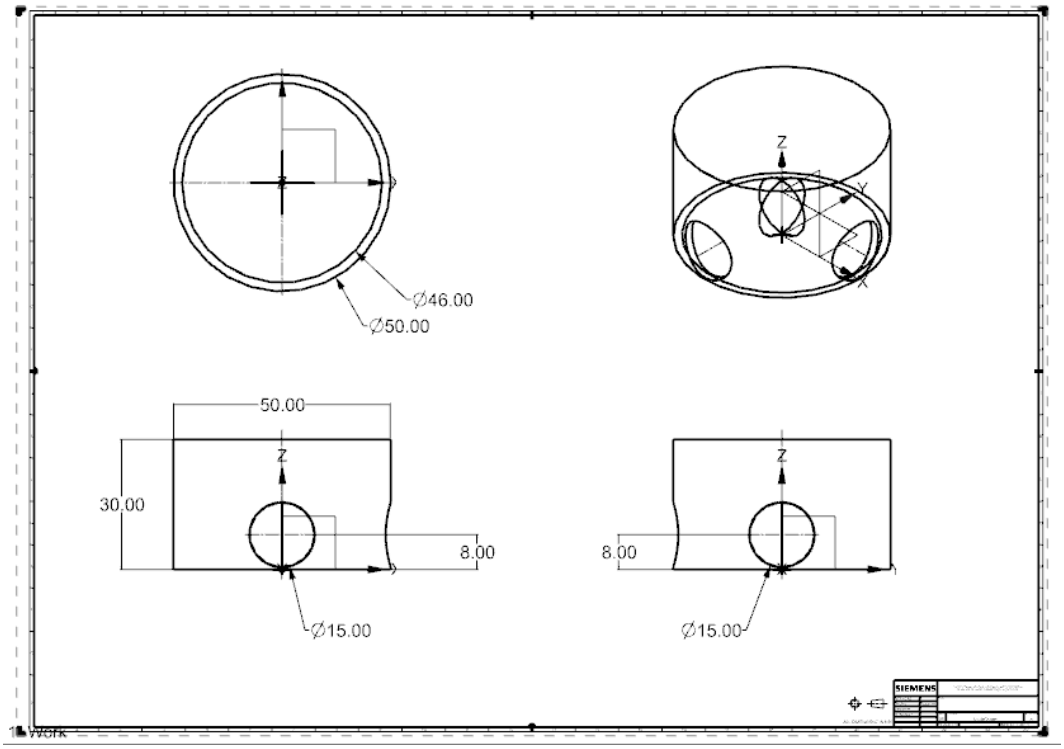

Figure 6. The Engineering Drawing for the Circle Center that goes at the point of the knee joint. It represents the circle center of the gyrometer. It would hold the PCB board and the battery in the final prototype. The dimensions are in the units of millimeters.

Wiring Diagrams

Here are the wiring diagrams for the two models: the test model (Flex Sensor + 1 IMU) and the final model (2 IMUs). This was the wiring used throughout the testing period.

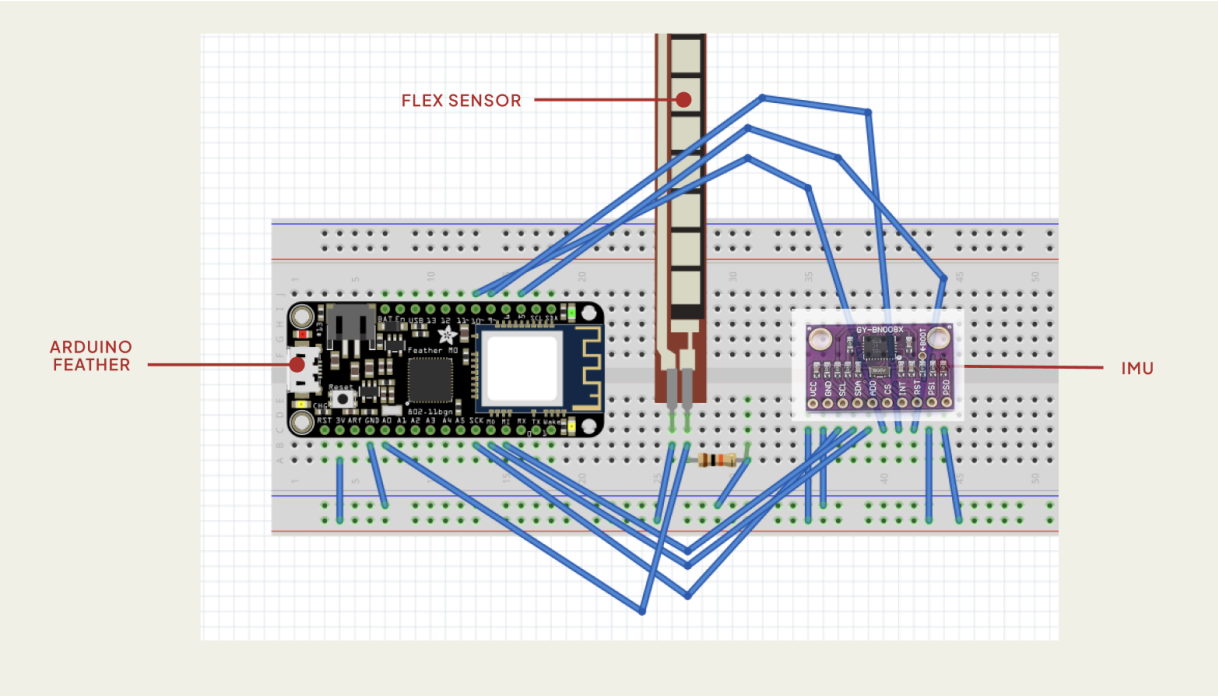

Figure 7. The test model had a microcontroller Arduino Feather M0 and IMU that both used soldered male header pins to allow easy placement on the breadboard. The flex sensor had wires connecting it to the analog A0 pin and power. The Adafruit Feather M0 Wifi utilizes an ATSAMD21G18 ARM Cortex M0 processor [20]. It was selected because it was a common product in the lab and worked with the breadboard. It was also recommended on the BNO085 datasheet for SPI and I2C communication with the IMU. The flex sensor was a great component for the test model because it has a direct response to knee bend and has a simple electrical interface.

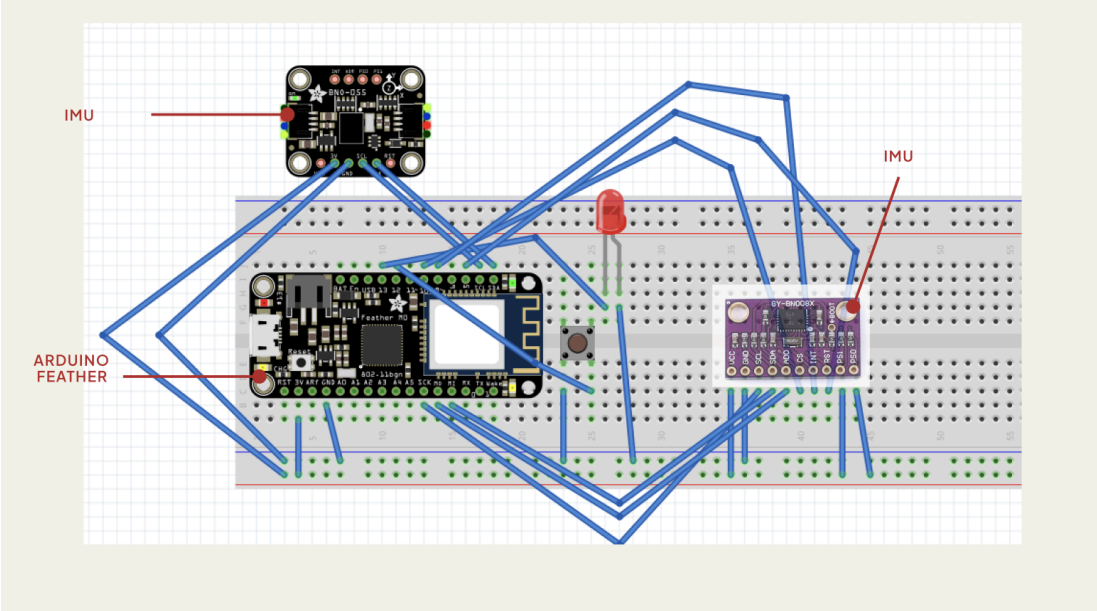

Figure 8. The final model had two IMUs and an Arduino Feather M0 microcontroller connected via a breadboard. A BNO055 IMU and a BNO08x IMU are used. One IMU is placed on the thigh and the other on the shin. The push button is used to reset for calibration. The LED is used to let the user know when calibration is complete after resetting the code. The Adafruit Feather M0 microcontroller acts as the central unit for communication and control. The microcontroller acquires data from both IMUs via I2C and SPI connections. These IMUs were selected because of their 9-axis fusion with built-in quaternion and Euler angle output, which is ideal for motion tracking.

Software Flowcharts

For both models, only software for the microcontroller was used. Thus, we had two different microcontroller code files: one for the test model and one for the final model. This is the breakdown of the software that was used in testing. For the test model, here is the flow chart:

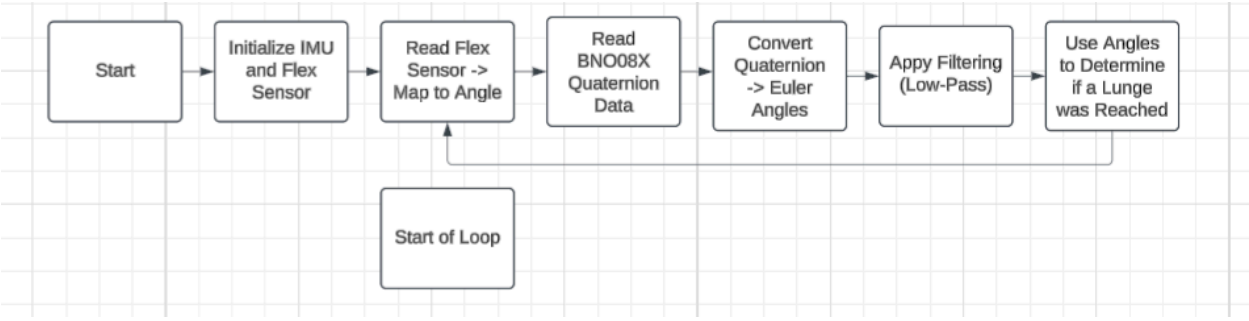

Figure 9. Software flow chart for the test model (1 IMU + Flex Sensor).

The digitized signals are from the flex sensor and the BNO08X IMU. The flex sensor uses the Analog A0 pin and is read using Arduino's ADC (analogRead() function). It converts the input voltage into an integer value between 0 and 1023 [21]. The BNO08X IMU uses SPI communication. The flex

sensor values are linearly mapped to a knee angle. An exponential moving average is also applied to the knee angle determined from the flex sensor values, acting as a low-pass filter. The quaternion data is then read digitally from the IMU. The code converts the BNO08X quaternion data to roll, pitch, and yaw using standard trigonometric equations. Throughout the code, the flex sensor and the IMU signals are digitized at different times throughout the loop. These two signals are not completely synchronized, but since they are sampled so frequently, real-time monitoring can be done.

In the test model code, the Adafruit_BNO08x and Wire libraries are used. These are all open-source libraries. The math function of quaternionToEuler utilizes common math equations. This code was created starting with the code to hook up a BNO08X IMU using SPI communication from the BNO085 Datasheet [25]. After this starting point, help from ChatGPT and personal code knowledge was used to finalize the code.

For the final model, here is the flow chart:

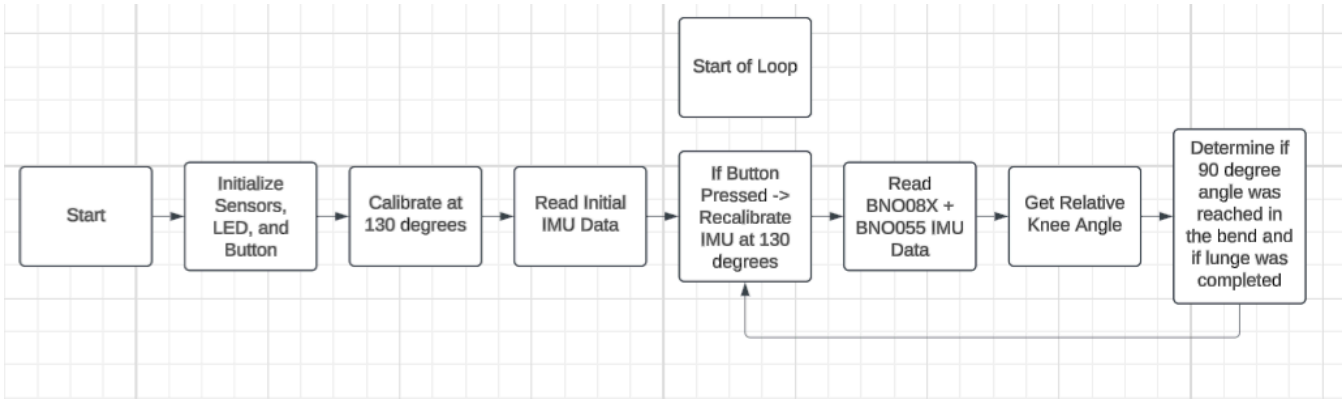

Figure 10. Software flow chart for the final model (2 IMUs).

The digitized signals are from the BNO055 IMU and the BNO08X IMU. The BNO055 IMU uses I2C communication to output Euler angles (pitch, roll, and yaw). These values are converted to quaternions during the code. The BNO08X IMU uses SPI communication to produce quaternion values (w, x, y, and z). The values from the two IMUs are read sequentially, but not at the same time. Each sensor is sampled every time the code loops. For processing, there is a lot of correction and calibration. After each time the code is initially run or the button is pressed, the current orientation values from the IMU are stored as the calibration. This is stored as the offsets for the 130-degree knee angle. The live readings from the IMUs are then adjusted by subtracting the offset values in order to get the relative difference in knee angle. This is carried out by utilizing the quaternion values to calculate the angular difference between the two IMUs.

In the final model code, the Adafruit_BNO08x, Adafruit_BNO055, math, SPI, and Wire libraries are used. These are all open-source libraries. The code utilizes common math equations, such as eulerToQuaternion, and writes a computeRelativeAngle equation using standard formulas. Finally, this code was created starting with the code to hook up a BNO08X IMU using SPI communication from the BNO085 Datasheet [25]. After this starting point, help from ChatGPT and personal code knowledge was used to finalize the code.

Organization of iKneed

The final product of iKneed uses the final model. For the final model, the Adafruit BNO08x IMU and Adafruit BNO055 IMU were connected to the Adafruit Feather M0 Wifi. The device used an inter-integrated circuit (I2C) connection for the BNO055 IMU and a serial peripheral interface (SPI) connection for the BNO08X IMU. I2C is a serial communication protocol that allows multiple chips and devices to communicate using just two wires: the serial clock line (SCL) and the serial data line (SDA) [22]. SPI is a high-speed full-duplex communication protocol used to connect microcontrollers to peripherals, in this case, an IMU. SPI wiring uses 4 main lines: MOSI, MISO, SCLK, and CS [23]. They both have their own advantages and disadvantages, but the team used both of them to ensure that the two IMUs were on different communication lines and they wouldn't interfere with each other.

In order to align with ISO 13485, the international standard for medical device quality management systems, there are a few key considerations. For the design and development process (clause 7.3), user needs must be clearly defined and verified. For this project, an example could be that the patient must be able to walk normally with the brace on for 8 hours without discomfort. The final product needs to meet user needs by testing real users walking and bending their knees. For risk management (clause 7.1), identified risks must be mitigated including risk of sensor data corruption and risk of battery overheating. Appropriate design controls will need to be selected and implemented. As for production and process controls (clause 7.5), there needs to be in-process inspections, such as where brace stiffness is measured and a functional test of the sensor is performed before shipping [24].

Signals and Processing

The inertial measurement units (IMUs) that the team used are 9-axis IMUs. The signals digitized are the 3-axis accelerometer (X, Y, Z) where acceleration is measured in g, a 3-axis gyroscope measuring angular velocity in °/s, and a 3-axis magnetometer measuring orientation in space. The sensor is digitized at the same time with a synchronous read via I2C connection or SPI connection. This is time-stamped in order to ensure alignment between axes. The signals are analyzed and processed using the internal signal processing of the Adafruit BNO08x IMU and the Adafruit BNO055 IMU [25][26]. The Adafruit BNO08X IMU also includes sensor fusion, bias correction, and drift compensation. Sensor fusion produced a quaternion-based orientation from acceleration, gyroscope, and magnetometer [25].

Results

Overall, the team saw amazing results. Both the test model and the final model worked as shown in the video turned in earlier. Both models were able to determine when a 90-degree angle was reached, which was the limit that was set to determine if a lunge was done or not. Both models also worked consistently and were easy to set up.

Below is some more information on what the final prototype looked like. Since the final model with 2 IMUs is the desired model, the rest of the prototype information only focuses on the final model and does not include the test model. First, this is the mockup of what the prototype would look like added on to a knee brace.

Figure 11. The MVP used our laptops' 5V DC power supply connected via USB. The Feather M0 Wifi has an onboard voltage regulator that converts 5V down to 3.3V and distributes 3.3V to its 3.3V pin that powers the external components (2 IMUs) [20]. In the final product, we would switch to a LiPo battery as an external power supply and use the Adafruit Feather M0 Wifi's integrated Wifi module and connect to a mobile application. The IMUs are placed in the boxes, with one IMU being in the top box and the other IMU being in the bottom box. Either IMU can be in either box and the system will work. The Circle Center holds the PCB Board and the LiPo battery. The tubing connecting the 3D parts would hold the wiring.

Here are the final mockups of the 3D-printed parts, which are used in the above diagram.

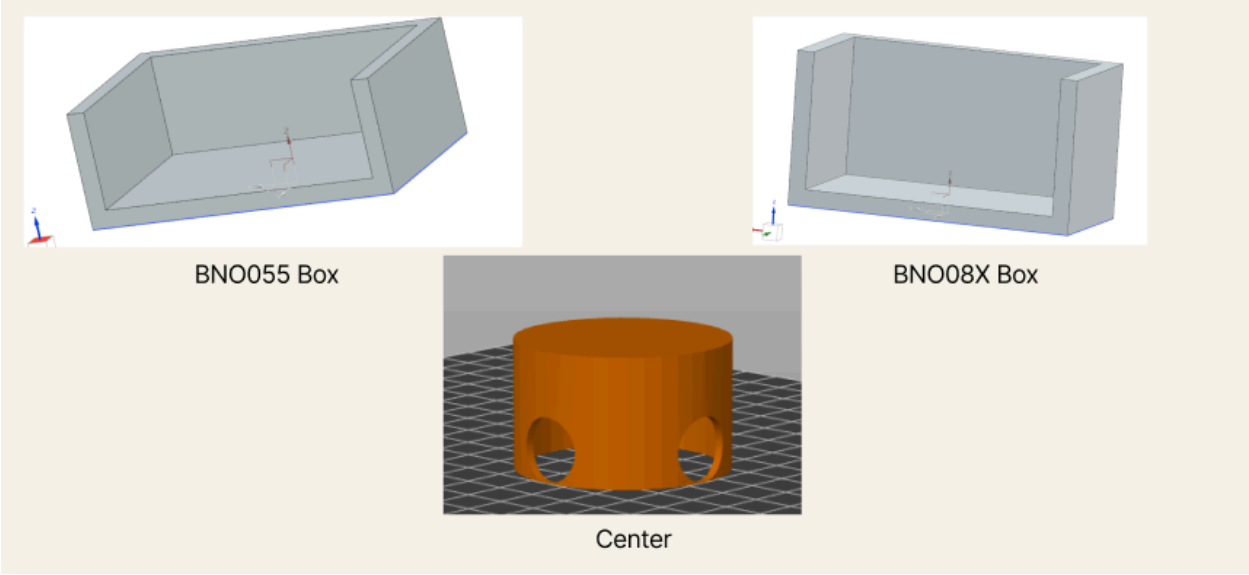

Figure 12. 3D printed hardware casing for prototype (MVP). The dimensions are in millimeters and can be found in the above drawings in the report (Figures 4, 5, and 6).

To create the mechanical prototype, the Engineering Drawings in Figures 4, 5, and 6 were used to create 3D-printed parts. The 3D-printed parts hold the IMUs (the two boxes), and the PCB Board and the LiPo Battery (the Circle Center). The 3D-printed parts were then connected using heat tubing melted over plastic tubing. This was done to create the design shown in Figure 11. 3D prints and heat tubing were used now for a prototype. However, for the final product, the heat tubing would be replaced by silicone sleeves in order to protect the electrical wiring. The 3D printed parts would be replaced with plastic parts. The prototype would be attached to the knee brace using velcro.

Next, these are the average current values for the components from the final model. As mentioned in Figure 11, a LiPo battery would be utilized in the final prototype. Based on these current values, a 1000 mAh LiPo battery could last for around 26 hours, letting one use the knee brace add-on all day or for multiple days before needing to charge.

Table 4. Total current draw is around 37.8 mA. With a 1000 mAh LiPo battery, the estimated runtime with a 37.8 mA draw is ~26 hours.

Component	Quantity	Current (Typical) Draw
BNO08x	1	~3.5 mA (acting at 100 Hz) [27]
BNO055	1	~12.3 mA [28]
Arduino Feather M0 Wifi	1	~22 mA [29]

Here is a more detailed flow structure of the final model software code file. Both the test model and the final model software code files are attached externally to the document. To access the test model code, go here:

https://drive.google.com/file/d/1pptOvdNYZbrE6lBBzmBnyugvVdOnV49l/view?usp=sharing. To access the final model code, go here: https://drive.google.com/file/d/1QIYWQTuLyIdbKV81BSGw2asRaY5zAe-X/view?usp=sharing. To watch the YouTube video on how to run the documents, go here: https://www.youtube.com/watch?v=ClarH7gNEgE. For the YouTube video, the team suggests using headphones at full volume as the recorder has a computer with a messed-up mic. Next up, here is also a diagram that shows the detailed operation of the microcontroller throughout the running of the final model. This diagram lists out the functions used in the code by the microcontroller.

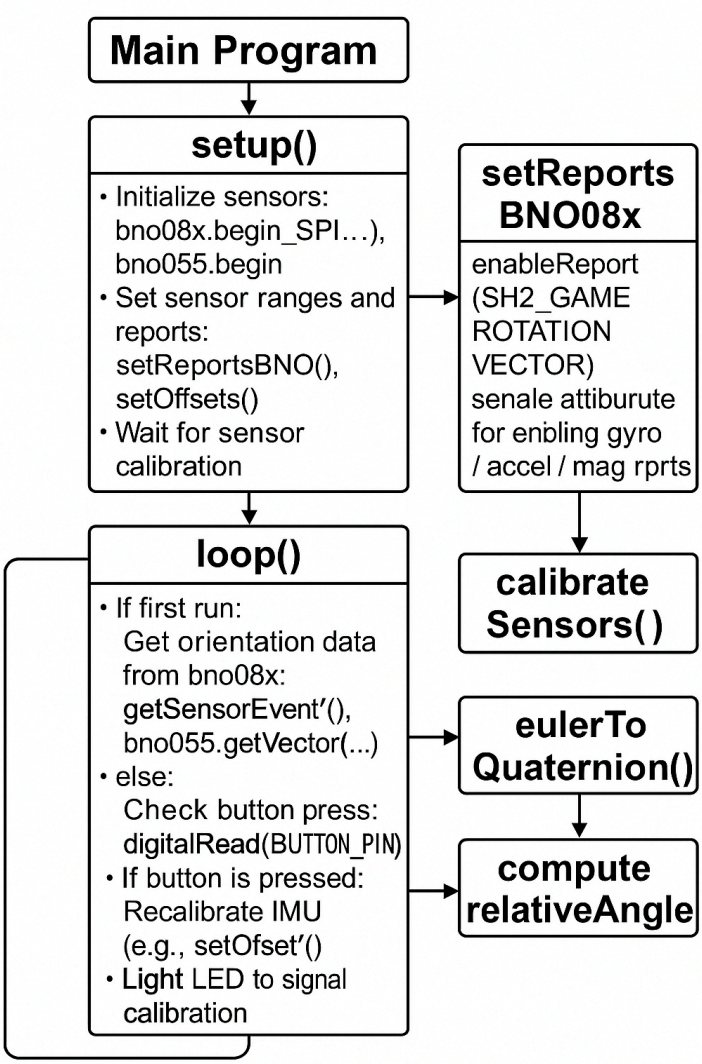

Figure 13. Setup initializes the sensors, loops checks if the button is pressed and calibrates IMU, and setReports enables gyroscope, accelerometer, and magnetometer data.

Continuing on, below is a diagram that breaks down how the microcontroller flows during the software program. This diagram is a more general description of how the code works. There is also a table

that lists the max sampling rates of the BNO08X IMU, providing information on the sampling rate of all signals.

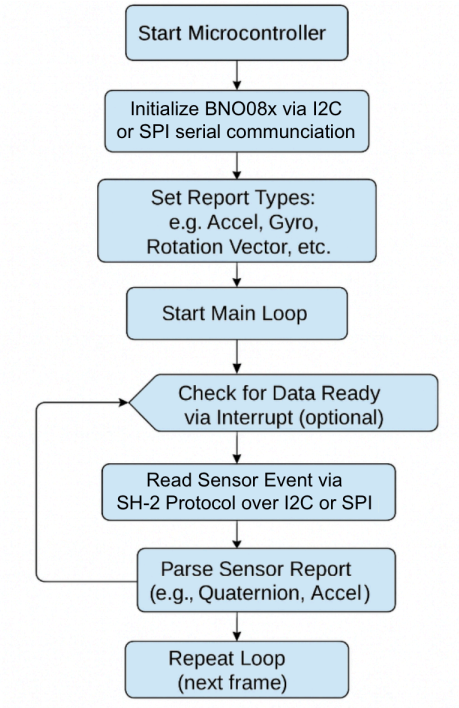

Figure 14. Flow chart of the microcontroller. It acts as a host that interfaces with the BNO08X IMU and the BNO055 IMU via I2C/SPI using the Adafruit BNO08x or the Adafruit BNO055 library. The code initializes the sensor, requests specific sensor data, reads the output data stream, and sends it to a computer or logging system. The internal signal for the accelerometer is analog and the output to the microcontroller is digital. For the gyroscope, the input is analog and the output is digital, and for the magnetometer, the input is analog and the output is digital. For data exchange with the software (SW program), the sensor communicates with the microcontroller via I2C (by default) or SPI. Sensor fusion data is sent using Hillcrest SH-2 protocol packets. The microcontroller communicates with host software and the BNO08x provides timestamped sensor events, which enable synchronization across devices [18].

Table 5. The BNO08x allows for per-report control of Output Data Rates (ODR) [27].

Sensor Report	Max Sampling (Data) Rate (Hz)
Accelerometer	500 Hz
Gyroscope	400 Hz
Magnetometer	100 Hz
Rotation Vector (Quaternion)	400 Hz

Gravity Vector	90 Hz
Linear Acceleration	400 Hz

In the table above, the accelerometer & gyroscope require high sampling rates for accurate motion tracking. The magnetometer doesn't have to be as high, as the magnetic field of Earth doesn't change much.

Strengths and Weaknesses Analysis

Strengths: It has real-time feedback where iKneed provides immediate performance correction, which is crucial for rehabilitation exercises to prevent bad habits and even reinjury. The device is compact and lightweight. It uses low-cost hardware. For future applications and a fully-fledged product, it can collect motion data to be used by clinicians to assess progress. It has a user-friendly interface that can be utilized by elderly or tech-inexperienced users.

Weaknesses: There is limited contextual understanding. The IMU alone cannot detect all types of improper form. Currently, the IMU is inconsistent due to issues with the prototype's breadboard. There are also battery constraints with the Li-Po battery. Despite it having a relatively long battery life (a day), it is still an inconvenience if it depletes.

Priority Improvements and Recommended Approaches

To improve the functionality and user experience of a joint monitoring brace, four essential enhancements are suggested:

1. Broaden sensing capabilities beyond inertial measurement units (IMUs): The current dependence on an IMU limits the system's capacity to monitor joint load or pressure. This issue can be resolved by integrating multi-modal sensing technologies, such as force-sensitive resistors (FSRs) or pressure pads, within the brace. These sensors would facilitate real-time joint load monitoring, providing a more precise assessment of joint activity. The FSRs should be strategically placed near the patella or lateral support areas to gather significant pressure data.

2. Enhance mobile application accessibility: The existing mobile application interface may be excessively complicated, particularly for older adults or individuals with physical disabilities. Implementing a more straightforward user interface, along with haptic and auditory feedback, would greatly enhance usability. This can be accomplished by incorporating a vibration motor and a small speaker. Furthermore, introducing an 'Accessibility Mode' that includes larger text, voice prompts, and simplified navigation will render the application considerably more inclusive.

3. Sensor accuracy can be compromised by minor misalignments or brace slipping during movement. To guarantee reliable data, an auto-calibration system must be instituted. The system would

perform baseline recalibration periodically and also monitor for chaotic acceleration patterns that signal the brace is not properly positioned. When this happens, the system can let the user know via the connected mobile app.

4. In order to enhance battery longevity and minimize the disruption caused by frequent recharging, it is essential to implement a more energy-efficient architecture. Transitioning to a low-power Bluetooth Low Energy (BLE) chip, such as the Nordic nRF52, alongside an increase in battery capacity, will facilitate prolonged usage [30]. This would make the product more accessible to patients.

Environmental Impact

The smart knee brace add-on has electronics in a 3D-printed housing. The electronics should be marked with e-waste recycling symbols and sent to e-waste recyclers such as e-Stewards [31]. Li-Polymer batteries should be disposed of at a hazardous waste drop-off site. The housing for the electronics can be recycled at a plastic recycling center that accepts 3D-printed waste.

Bill of Materials

Here is everything that the team bought for the project. Prices may have been updated, but the prices listed here is what was paid by the team at the time. The total cost of materials was $176.09.

- Teyleten Robot GY- BNO085 AR VR IMU High Accuracy Nine-Axis 9DOF AHRS Sensor [32]
    - 3x units
    - $18.99 each
    - Website: https://www.amazon.com/Teyleten-Robot-BNO085-Accuracy-Nine-Axis/dp/B0CL26J81F
- KBT 3.7V 1000mAh Li-Polymer Battery: 523450 Lipo Rechargeable Lithium-ion Replacement Batteries with PH 2.54 JST Connector, PH1.25/2.0 JST Connector for Replacement - 2Pack [33]
    - 1x unit
    - $15.98
    - Website: https://www.amazon.com/KBT-3-7V-1000mAh-Li-Polymer-Battery/dp/B0BJPFR756/ref=asc_df_B0BJPFR756?tag=bingshoppinga-20&linkCode=df0&hvadid=80814294283644&hvnetw=o&hvqmt=e&hvbmt=be&hvdev=c&hvlocint=&hvlocphy=&hvtargid=pla-458441376285506 2&msclkid=355bea20ccb91263a6468d8a94949d0a&th=1
- Baseline 12-1025 Absolute Axis 360 Degree Plastic Goniometer, 12" Length [34]
    - 1x unit
    - $18.69
    - Website: https://www.amazon.com/Baseline-12-1025-Absolute-Plastic-Goniometer/dp/B008N3SQJ8
- MIIRR Human Knee Joint Model with Knee Ligament, Life Size Anatomical Knee Joint Flexible Skeleton Model, Perfect for Medical Learning and Teaching [35]

    - 1x unit
    - $29.99
    - Website: https://www.amazon.com/MIIRR-Human-Joint-Model-All/dp/B09VTJ4M9G?th=1
- OIIKI Goniometer Set 6” / 360 Degree [36]
    - 1x unit
    - $5.58
    - Website: https://www.amazon.com/Goniometer-inch-360-Degree-Plastic/dp/B0DHRST8M8?th=1
- Long Flex Sensor Adafruit Industries [37]
    - 2x units
    - $12.95 each
    - Website: https://www.adafruit.com/product/182
- 40 PCS 20 CM (8 inch) Breadboard Jumper Wires Length Optional Dupont Wire Assorted Kit Female to Male Multicolored Ribbon Cables [38]
    - 1x unit
    - $3.99 each
    - Website: https://www.amazon.com/California-JOS-Breadboard-Optional-Multicolored/dp/B0BRTHR2RL/ref=sr_1_5?crid=20QG6W8D58JAD&dib=eyJ2IjoiMSJ9.tjHxIQLJsk16_0YVtUGN6Tqnr8euWNsWVjpSaq5RQkb9i7FP8hWDcWd7nUWsAPlVs-09xUf804joVmKfvzdMsLVIcM6m57tHYW8Lq9FNganYmcFeVxjreqJG92xjFxaArS2EfqYwb7F0pik0dmzDlotklo6MnaTDQFFvPnECsLf4ZXDW4TlY6YxMGHQGvWIDRNRlRFWl1Z5JBRbMG7_rLyS1sfp8g6f23-oeKFOWooQ.opP1FcCMTIIlz28hZTuHSkC3g_OjelcKc4aVFHq3Xxo&dib_tag=se&keywords=male%2Bfemale%2Bwires&qid=1744161654&sprefix=male%2Bfemale%2Bwires%2Caps%2C209&sr=8-5&th=1
- Teyleten Robot GY- BNO085 AR VR IMU High Accuracy Nine-Axis 9DOF AHRS Sensor Module [32]
    - 1x unit
    - $18.99 each
    - Website: https://www.amazon.com/Teyleten-Robot-BNO085-Accuracy-Nine-Axis/dp/B0CL26J81F

Technically Difficult Problems

The team ran into many problems throughout the semester. Here is a breakdown of the main problems and the solution that ended up working. For more detailed information, check out the Lab Notebook section.

Table 6. A breakdown of the technical difficult problems that the team ran into and the solutions that they ended up using.

Problem	**Solution**

Difficult to get two IMUs to communicate with the Adafruit Feather M0 microprocessor. Most IMUs have only one selectable I2C address.	Use SPI for one IMU and I2C for the other.
Two of the same IMUs could not work at the same time using an Adafruit Feather M0 microprocessor, even when one IMU was using SPI and the other IMU was using I2C.	Use two different IMUs (BNO055 and BNO08x), allowing for both to use slightly different communication IDs. They are then able to work together.
No team member with electrical knowledge besides a basic class.	Spend additional time working on circuits and code. Trying two different options to make sure we would have something to present. Ended up with both options working.
Accurate sensor readings and knee angle output.	Use sensor fusion to get orientation. Convert raw accelerometer and gyroscope data into Euler angles (pitch, roll, yaw). Also, use a simple low-pass filter in the Arduino script to smooth knee angle in the Test Model.
Make it usable in real life. The solution doesn't matter if it can't actually be applied to a knee brace.	Make it into an add-on. It's a product that you can add to any knee brace and still be used. This is because it is made of simple materials (plastic and silicone sleeves) and designed in a way where one just attaches it using velcro.

Computer Code

Here are our two code files. The first code file is for the Test Model and the second code file is for the Final Model.

```ino
#include <Adafruit_BNO08x.h> // Install this Library
#include <Wire.h>

// Flex Sensor Setup
const int FLEX_PIN = A0;
int flexRaw;
float kneeFlexAngle = 0.0;
float smoothedKneeFlexAngle = 0.0;

// Calibrate these values based on your raw readings
const int FLEX_MIN = 455;  // knee at 0°
const int FLEX_MAX = 475;  // knee at 90°

// BNO08x Setup (SPI Mode)
#define BNO08X_CS    10
#define BNO08X_INT    9
#define BNO08X_RESET  5

Adafruit_BNO08x bno08x(BNO08X_RESET);
sh2_SensorValue_t sensorValue;

// Quaternion to Euler conversion
void quaternionToEuler(float qx, float qy, float qz, float qw, float &pitch, float &roll, float &yaw) {
  float sinr_cosp = 2.0f * (qw * qx + qy * qz);
  float cosr_cosp = 1.0f - 2.0f * (qx * qx + qy * qy);
  roll = atan2(sinr_cosp, cosr_cosp) * 180.0 / PI;

  float sinp = 2.0f * (qw * qy - qz * qx);
  if (abs(sinp) >= 1)
    pitch = copysign(90.0, sinp);
  else
    pitch = asin(sinp) * 180.0 / PI;

  float siny_cosp = 2.0f * (qw * qz + qx * qy);
  float cosy_cosp = 1.0f - 2.0f * (qy * qy + qz * qz);
```

```
  yaw = atan2(siny_cosp, cosy_cosp) * 180.0 / PI;
}

void setup() {
  Serial.begin(115200); //Baud rate that the Serial Monitor is using
  while (!Serial);  // Wait for Serial Monitor
  Serial.println("Booting...");

  pinMode(FLEX_PIN, INPUT); //Sets up the Flex Sensor as an Input

  // Makes sure the BNO08X IMU is working over SPI
  Serial.println("Initializing BNO08x over SPI...");
  if (!bno08x.begin_SPI(BNO08X_CS, BNO08X_INT)) {
    Serial.println("❌ Failed to find BNO08x over SPI.");
    while (1) delay(10);
  }
  Serial.println("✅ BNO08x connected!");

  // Confirms which IMU is being used and what Firmware is on the IMU
  for (int n = 0; n < bno08x.prodIds.numEntries; n++) {
    Serial.print("Part ");
    Serial.print(bno08x.prodIds.entry[n].swPartNumber);
    Serial.print(": Version ");
    Serial.print(bno08x.prodIds.entry[n].swVersionMajor);
    Serial.print(".");
    Serial.print(bno08x.prodIds.entry[n].swVersionMinor);
    Serial.print(".");
    Serial.print(bno08x.prodIds.entry[n].swVersionPatch);
    Serial.print(" Build ");
    Serial.println(bno08x.prodIds.entry[n].swBuildNumber);
  }

  // Checks that the software is able to get the Game Rotation Vector from the IMU
  if (!bno08x.enableReport(SH2_GAME_ROTATION_VECTOR)) {
    Serial.println("❌ Failed to enable game rotation vector.");
```

```
  } else {
    Serial.println("✅ Game rotation vector enabled.");
  }

  delay(100); // Delay
}

void loop() {
  delay(500);  // Delay for readability

  // Read Flex Sensor
  flexRaw = analogRead(FLEX_PIN);
  kneeFlexAngle = map(flexRaw, FLEX_MIN, FLEX_MAX, 0, 90);
  kneeFlexAngle = constrain(kneeFlexAngle, 0, 135);
  smoothedKneeFlexAngle = 0.9 * smoothedKneeFlexAngle + 0.1 * kneeFlexAngle;

  // Prints out the Knee Angle calculated using the Flex Sensor
  Serial.print("Smoothed Knee Angle (Flex): ");
  Serial.print(smoothedKneeFlexAngle);
  Serial.print("°\t");

  // Handle IMU reset
  if (bno08x.wasReset()) {
    Serial.println("IMU was reset. Re-enabling reports...");
    bno08x.enableReport(SH2_GAME_ROTATION_VECTOR);
  }

  // Read IMU Quaternion Data
  if (bno08x.getSensorEvent(&sensorValue)) {
    // Read Game Rotation Vector from IMU
    if (sensorValue.sensorId == SH2_GAME_ROTATION_VECTOR) {
      float i = sensorValue.un.gameRotationVector.i;
      float j = sensorValue.un.gameRotationVector.j;
      float k = sensorValue.un.gameRotationVector.k;
      float r = sensorValue.un.gameRotationVector.real;
```

```
      float pitch, roll, yaw;
      quaternionToEuler(i, j, k, r, pitch, roll, yaw);

      pitch = -pitch;  // Flip if needed based on mounting orientation

      // Calibrate IMU Pitch (34 raw → 70° real)
      float calibratedPitch = map(pitch, 0, 34, 0, 70);
      calibratedPitch = constrain(calibratedPitch, 0, 135);

      // Print out the Calibrated Pitch from the IMU
      Serial.print("Calibrated IMU Pitch: ");
      Serial.print(calibratedPitch);
      Serial.println("°");

      // Determines if Lunge was Done Correctly Using the IMU Pitch Values
      if (calibratedPitch >= 50.0 && calibratedPitch <= 90.0) {
        Serial.println("Is the lunge correct? → Yes ✅");
      } else {
        Serial.println("Is the lunge correct? → No ❌");
      }
    }
  }
}
```

Figure 15. This is the code for the test model connecting the flex sensor to the IMU. To create the knee angle output, the script reads the flex sensor, mapping the raw value to a 0-90º range. It applies a simple low-pass filter and prints the smoothed knee angle. For orientation, it converts quaternion values into Euler angles and prints pitch, roll, and yaw in degrees.

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BNO08x.h> // Install this Library
#include <Adafruit_BNO055.h> // Install this Library
#include <math.h>

// BNO08x (SPI) Setup
#define BNO08X_CS 10
#define BNO08X_INT 9
#define BNO08X_RESET 5

// Pins for Button and LED
#define BUTTON_PIN 12
#define LED_PIN 13

// Resets the BNO08x IMU and sets the BNO055 IMU at its I2C address
Adafruit_BNO08x  bno08x(BNO08X_RESET);
sh2_SensorValue_t sensorValue;
Adafruit_BNO055 bno055 = Adafruit_BNO055(55, 0x28);

// Quaternion structure
struct Quaternion {
  float w, x, y, z;
};

//Defines the Offset Quaternion variables for the two IMUs
Quaternion qOffsetBNO055;
Quaternion qOffsetBNO08x;
bool isCalibrated = false;
bool resetInProgress = false;

// Convert Euler angles (BNO055) to quaternion
Quaternion eulerToQuaternion(float roll, float pitch, float yaw) {
  roll  *= DEG_TO_RAD;
  pitch *= DEG_TO_RAD;
```

```
  yaw   *= DEG_TO_RAD;

  float cy = cos(yaw * 0.5);
  float sy = sin(yaw * 0.5);
  float cp = cos(pitch * 0.5);
  float sp = sin(pitch * 0.5);
  float cr = cos(roll * 0.5);
  float sr = sin(roll * 0.5);

  Quaternion q;
  q.w = cr * cp * cy + sr * sp * sy;
  q.x = sr * cp * cy - cr * sp * sy;
  q.y = cr * sp * cy + sr * cp * sy;
  q.z = cr * cp * sy - sr * sp * cy;
  return q;
}

// Compute angle (in degrees) between two quaternions
float computeRelativeAngle(Quaternion q1, Quaternion q2) {
  float dot = q1.w*q2.w + q1.x*q2.x + q1.y*q2.y + q1.z*q2.z;
  dot = constrain(dot, -1.0, 1.0);
  float angle = 2 * acos(dot) * RAD_TO_DEG;
  return angle;
}

// Makes sure BNO08x is able to Deliver Input Values (Game Rotation Vector)
void setReports(void) {
  Serial.println("Setting desired reports for BNO08x");
  if (!bno08x.enableReport(SH2_GAME_ROTATION_VECTOR)) {
    Serial.println("Could not enable game vector");
  }
  delay(500);
}

// Run Calibration at a Knee Angle of 130 Degrees
```

```
71  void calibrateSensors() {
72    setReports();
73    Serial.println("Hold knee at 130° position for calibration...");
74    delay(3000);  // Wait for you to position knee
75
76    // Capture BNO055 calibration orientation
77    imu::Vector<3> euler = bno055.getVector(Adafruit_BNO055::VECTOR_EULER);
78    qOffsetBNO055 = eulerToQuaternion(euler.x(), euler.y(), euler.z());
79
80    // Capture BNO08x calibration orientation
81    unsigned long startTime = millis();
82    bool gotBNO08x = false;
83    while (millis() - startTime < 5000) {
84      if (bno08x.getSensorEvent(&sensorValue)) {
85        if (sensorValue.sensorId == SH2_GAME_ROTATION_VECTOR) {
86          qOffsetBNO08x.w = sensorValue.un.gameRotationVector.real;
87          qOffsetBNO08x.x = sensorValue.un.gameRotationVector.i;
88          qOffsetBNO08x.y = sensorValue.un.gameRotationVector.j;
89          qOffsetBNO08x.z = sensorValue.un.gameRotationVector.k;
90          gotBNO08x = true;
91          break;
92        }
93      }
94      delay(10);
95    }
96
97    // If able to get a BNO08x value, Calibration has been completed
98    if (gotBNO08x) {
99      isCalibrated = true;
100     Serial.println("Calibration complete!");
101   } else {
102     Serial.println("Error reading BNO08x for calibration");
103   }
104
105   digitalWrite(LED_PIN, HIGH);  // Turn on LED when calibration is done
```

```
  digitalWrite(LED_PIN, HIGH);  // Turn on LED when calibration is done
}

void setup(void) {
  Serial.begin(115200); // Baud Rate that the Serial Monitor is running on
  while (!Serial) delay(10);

  pinMode(BUTTON_PIN, INPUT_PULLUP);  // Button with pull-up resistor
  pinMode(LED_PIN, OUTPUT); // Defines LED as an OUTPUT
  digitalWrite(LED_PIN, LOW); // LED is turned off

  Serial.println("Adafruit BNO08x & BNO055 Knee Angle Measurement"); // Prints out this Starting Line

  // Initialize BNO08x
  if (!bno08x.begin_SPI(BNO08X_CS, BNO08X_INT)) {
    Serial.println("Failed to find BNO08x chip");
    while (1) { delay(10); }
  }
  Serial.println("BNO08x Found!");
  delay(500);

  // Initialize BNO055
  if (!bno055.begin()) {
    Serial.println("Failed to find BNO055 chip");
    while (1) { delay(10); }
  }
  Serial.println("BNO055 Found!");
  delay(500);

  // Prints out Information regarding the BNO055 Status, the Self-Test ID, and the System ID
  uint8_t status, selfTest, systemId;
  bno055.getSystemStatus(&status, &selfTest, &systemId);
  Serial.print("BNO055 Status: "); Serial.println(status);
  Serial.print("Self-Test: "); Serial.println(selfTest);
```

```
  Serial.print("System ID: "); Serial.println(systemId);

  calibrateSensors();
}

void loop() {
  // Resets the BNO08x Sensor
  if (bno08x.wasReset()) {
    Serial.println("BNO08x sensor was reset");
    setReports();
    delay(500);
  }

  // Loop if Button is Pressed and Reset of Code is Set In Progress
  if (digitalRead(BUTTON_PIN) == LOW && !resetInProgress) {
    resetInProgress = true;
    digitalWrite(LED_PIN, LOW);  // Turn off LED during reset
    isCalibrated = false;

    // Prints out Lines on the Status of the Reset
    Serial.println("Reset button pressed. Resetting system...");
    Serial.println("Please reset equipment! Waiting 10 seconds:");

    // Prints a Countdown from 10 to 0
    for (int i = 10; i > 0; i--) {
      Serial.print(i);
      Serial.println("...");
      delay(1000);
    }

    // Reruns the Calibration of the Sensors
    calibrateSensors();
    resetInProgress = false;
  }

```

```
// Continues the code if isCalibrated Variable Lets the Code Know that Calibration Has Been Completed
if (!isCalibrated) return;

delay(100); // Delay to Not Overwhelm Servers

// Get current BNO08x quaternion
Quaternion qBNO08x;
if (bno08x.getSensorEvent(&sensorValue)) {
  if (sensorValue.sensorId == SH2_GAME_ROTATION_VECTOR) {
    qBNO08x.w = sensorValue.un.gameRotationVector.real;
    qBNO08x.x = sensorValue.un.gameRotationVector.i;
    qBNO08x.y = sensorValue.un.gameRotationVector.j;
    qBNO08x.z = sensorValue.un.gameRotationVector.k;
  }
}

// Get current BNO055 quaternion
imu::Vector<3> euler = bno055.getVector(Adafruit_BNO055::VECTOR_EULER);
Quaternion qBNO055 = eulerToQuaternion(euler.x(), euler.y(), euler.z());

// Compute angle from 130° reference
float angleCurrent = computeRelativeAngle(qBNO055, qBNO08x);
float angleOffset = computeRelativeAngle(qOffsetBNO055, qOffsetBNO08x);
float kneeAngle = angleCurrent - angleOffset;

// Prints out What the Knee Angle currently is Based on the Reference Angle of 130 Degrees
Serial.print("Knee Angle away from 130° reference: ");
Serial.print(kneeAngle);
Serial.println(" degrees");

// If the Current Knee Angle Is At A Difference of 25 degrees or more, Serial Monitor Prints that Lunge was Done Correctly
if (abs(kneeAngle) > 25.0) {
  Serial.println("Lunge completed correctly");
}

  delay(200); // Delay so Servers are Not Overwhelmed
}
```

Figure 16. This is the code for the final model, where it tracks the orientation of two IMUs using quaternions. This calibrates the system at a known rotational angle of 130 degrees. It computes the current knee angle based on the relative orientation difference between the two IMUs. It detects if the lunge was completed correctly.

Software Subroutines and External Routines for the Final Model

The Final Model reads orientation data from two IMUs (BNO055 via I2C communication and BNO08X via SPI communication), processes the data using quaternions, and determines whether the user's knee reaches the proper angle for a lunge. It also utilizes a button to trigger a reset, leading to a new calibration, and an LED to let the user know when calibration is finished after the reset.

The design pattern is procedural as the code is organized in functions and follows a top-down flow. The code is also event-driven as pressing a button can reset the calibration.

The code also utilizes many subroutines. There are three main categories for the subroutines: under the setup() section, under the loop() section, and the additional functions. Here is a hierarchy of the subroutines:

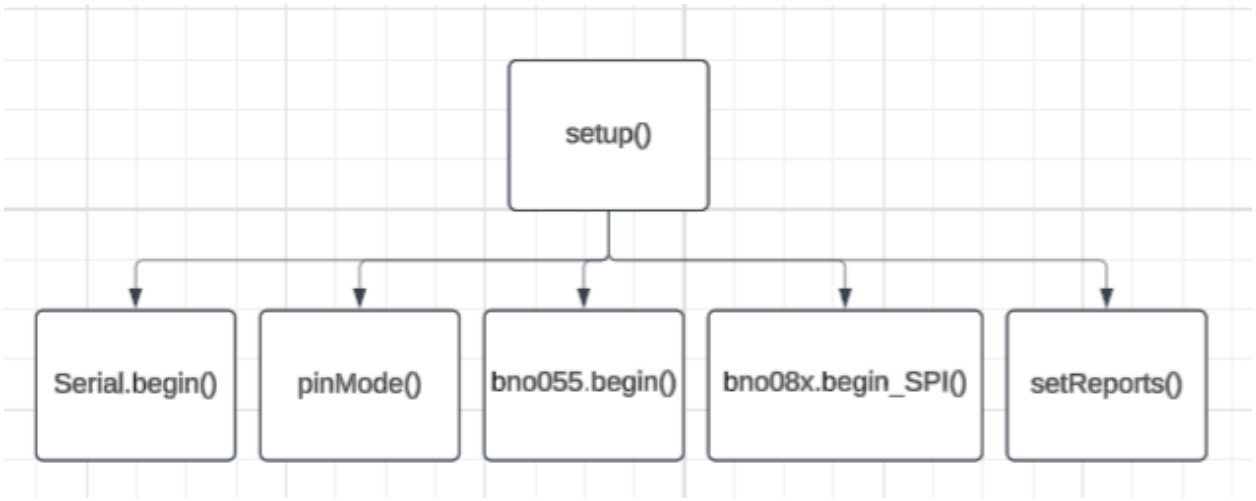

Figure 17. Diagram showing all of the subroutines under the setup() section in the Final Model Code.

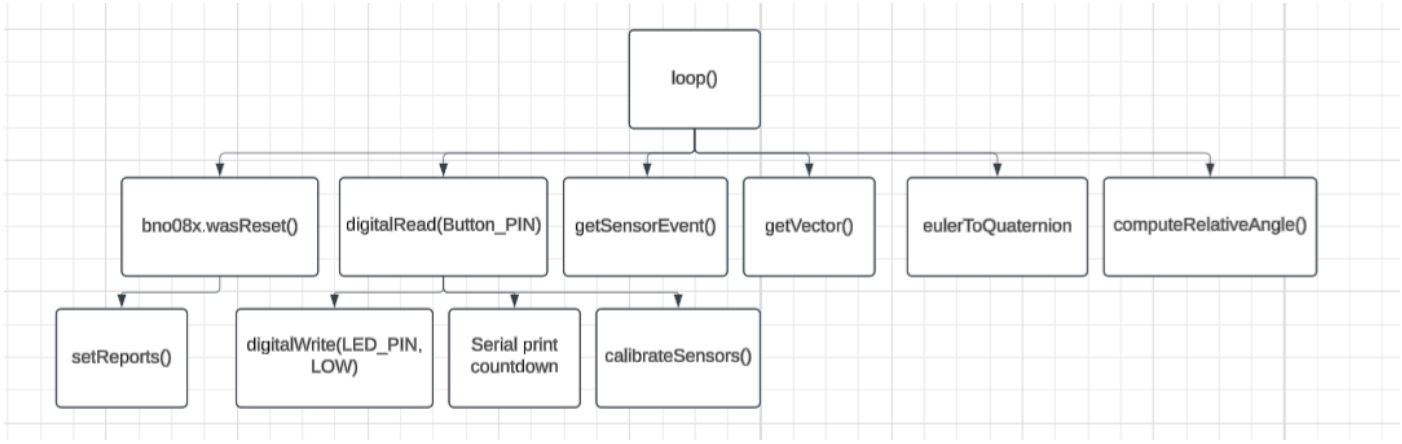

Figure 18. Diagram showing all of the subroutines under the loop() section in the Final Model Code.

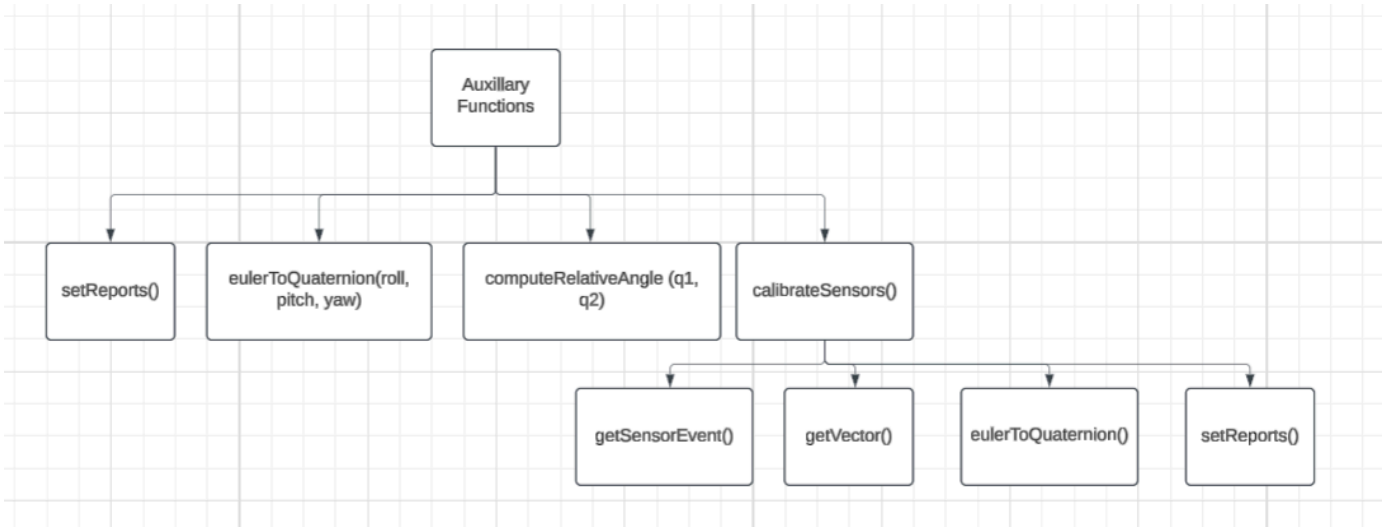

Figure 19. Diagram showing all of the subroutines that are not under the setup() or loop() sections. Instead, they are auxiliary functions in the Final Model Code.

All of these software subroutines play an important role in the Final Model Code. Here is a breakdown of them:

- Serial.begin()
    - Role: Initializes serial communication at the specified baud rate
    - Input: Baud Rate
    - Output: None
- pinMode()
    - Role: Sets the digital pin as an input or an output
    - Input: Digital pin number and the mode of input, output, or input_pullup
    - Output: None
- bno055.begin()
    - Role: Initializes the BNO055 IMU
    - Input: None
    - Output: Returns true (if the initialization was successful) or false (if the initialization was not successful)
- bno08x.begin_SPI()
    - Role: Initializes the BNO08X IMU using SPI communication
    - Input: Chip Select and Interrupt pins
    - Output: Returns true (if the initialization was successful) or false (if the initialization was not successful)
- setReports()
    - Role: Sets up the BNO08X IMU to send out game rotation vectors
    - Input: None
    - Output: None
- bno08x.wasReset()
    - Role: Checks if the BNO08X IMU underwent a reset
    - Input: None
    - Output: Returns true (if the sensor was reset) or false (if the sensor was not reset)
- digitalRead(Button_PIN)
    - Role: Reads if the button pin is High or Low
    - Input: Button_PIN
    - Output: High or Low
- getSensorEvent()
    - Role: Gets the latest sensor reading from the BNO08X IMU
    - Input: None
    - Output: Returns true (if new data is available) or false (if new data is not available)
- getVector()
    - Role: Gets the orientation data from the BNO055 in the Euler angles format
    - Input: the Vector type
    - Output: A vector containing the roll, pitch, and yaw angles
- eulerToQuaternion(roll, pitch, yaw)
    - Role: Converts the Euler angles to Quaternion values
    - Input: roll, pitch, and yaw values

    - Output: Quaternion values (w, x, y, and z)
- digitalWrite(LED_PIN, LOW)
    - Role: Sets the output level of the digital LED pin
    - Input: LED_PIN and what the desired output level of Low or High is
    - Output: None
- Serial print countdown
    - Role: Prints a 10 second countdown to the Serial Monitor
    - Input: None
    - Output: Sends output to serial monitor
- calibrateSensors()
    - Role: Captures and stores the current quaternion values from both IMUs for reference
    - Input: None
    - Output: Stores reference quaternions in qOffsetBNO08x and qOffsetBNO055
- computeRelativeAngle(q1, q2)
    - Role: Calculates the relative angular difference between the two quaternion values
    - Input: q1 and q2 as quaternion value structures
    - Output: Angle in degrees

GUI

To facilitate the sensor and firmware structure, the team designed a graphical user interface that educates, encourages, and guides patients with their rehabilitation goals. In the displayed prototype GUI, the main focus was to engage patients and maintain motivation by taking data and turning it into actionable insights. It enables users to effortlessly monitor progress against personalized goals, receive reminders on time to facilitate consistency, and visualize achievement to reinforce good behavior – all from the comfort and convenience of their home. It also allows clinicians to monitor adherence and performance remotely to make care data-driven without adding complexity to the end user.

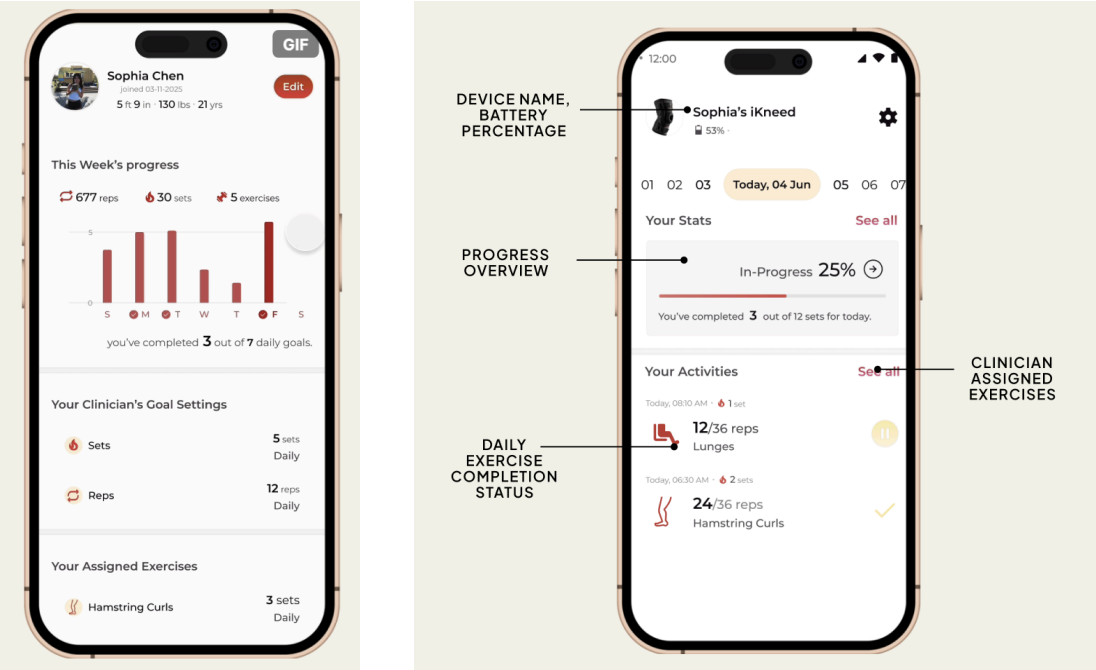

Figure 20. The iKneed home screen displays weekly progress, clinician goals, assigned exercises, awards, and metrics. This layout keeps patients informed and motivated on their fitness and rehabilitation journey. A horizontal date bar lets you flip between days, while the Progress Overview card tracks how many of your prescribed sets you've completed with live percentage and total-sets count. The activities list breaks down each exercise by time, reps/sets, and gives you pause or mark-complete controls for seamless session management.

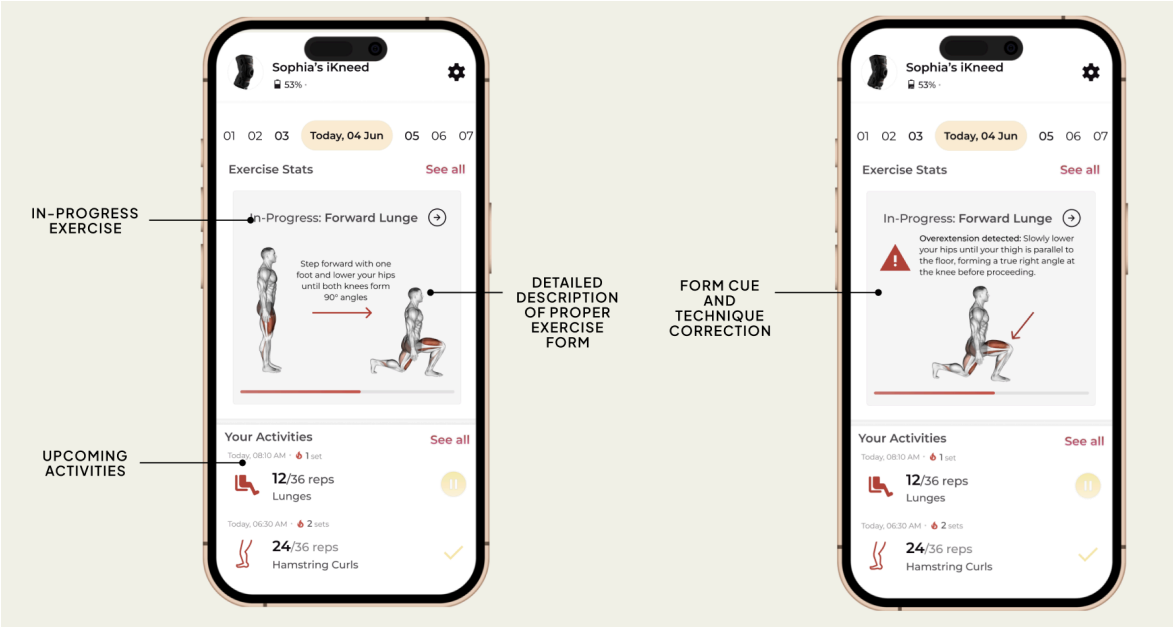

Figure 21. This screen is your pre-exercise tutorial: it presents a clean, animated 3D demonstration of the forward lunge—highlighting foot placement, hip alignment, and a 90° front-knee angle—while concise, step-by-step instructions walk you through each phase so you know exactly how to set up your body before you begin. Once you've reviewed the form cues, tapping "Start" will launch the live activity view, where the iKneed brace delivers real-time feedback on your motion to help you stay on track and perfect your technique throughout every rep. Real-time motion data from your iKneed guides you through every rep with visual and numerical feedback. This view tracks your knee-flex angle, then overlays an animated form cue to help you correct your technique on the spot.

Discussion

Lessons Learned

The impact is that there will be increased access to quality rehabilitation at a low cost. Healthcare spending will decrease overall and this will contribute to the growing multi-billion dollar industry of wearable health technology. For future students, we would recommend using only one IMU attached to the thigh and a strong magnet attached to the calf. This would bypass the need for a second IMU, which unnecessarily complicates things. A combination of a single IMU and a magnet would properly produce a knee angle.

As per ISO 13485, the project should be treated as a real medical device from the start of development. Everything should be documented: design decisions, user needs, improvements, and test protocols [6]. This would help because ISO 13485 emphasizes documentation and a repeatable process. These habits early on will reduce risk and improve team accountability. Because of the emphasis on user needs in ISO 13485, interviewing clinicians and users to use their feedback to adhere to design requirements would be beneficial. With regards to another standard, ISO 14971, a risk management file

should be made. This would be a live document including potential hazards (e.g., battery overheating and sensor failure), risk evaluations, and mitigation strategies, such as results [7]. This would help because it ensures that safety is built into the design rather than being haphazardly tested at the end.

Other key engineering standards include IEC 60601-1 for safety and performance of electrical medical equipment, IEEE 11073 for interoperability standards for medical devices, and FDA design controls (21 CFR part 820.30), which is particularly useful for the US-based regulatory context [8][11][39].

Continuing on, we used the Adafruit website to purchase the long flex sensor for the test model. While this is not required if future students plan on only developing the final model, it is a good place to start. The Adafruit BNO085 Absolute Orientation Sensor protocol on the Adafruit Learning System website created by Kevin Townsend is an excellent resource for wiring and Arduino code instructions.

For future BME Senior Project students, the team suggests that they try to avoid two simple mistakes: buying off-brand and not reading extensively the data sheet before buying. We bought the BNO08X IMU off-brand. At first, we thought it's cheaper and we can get it through Amazon so it will come quicker. While that is true, there was a lack of information surrounding the off-brand IMU. The pinout was completely different from the on-brand IMU with no documentation to explain these differences. Furthermore, the only datasheet that we could find was one for the on-brand IMU. While this datasheet did help us figure out how to connect the IMU, it took a lot of trial and error. It would have been a lot easier to just buy an on-brand IMU. Continuing on, we mainly glanced at the data sheets before buying the IMUs. It didn't hurt us terribly in any way, but if we had read through them more extensively, we would have been better prepared for any issues that came up. For example, it took two to three weeks to figure out why an ESP32 was so finicky for the I2C connection for the BNO08X. If we had just read the data sheet earlier, we would have seen that fact right there.

We do suggest many of the products that we ended up using however. The Adafruit Long Flex Sensor is a good choice for flex sensors because it's cheap and pretty simple to use. It integrates nicely into the tracking of knee flexion and is reliable with its measurements. The Adafruit Feather M0 Wifi was also very easy to work with. It does have a limitation though as it doesn't support Bluetooth. If your project involves Bluetooth capabilities, we would suggest the Adafruit Feather M0 Bluetooth or the ESP32. The LiPo Battery is also a good product. It's very easy to test with. It's also lightweight and compact. However, make sure to calculate your total current usage before buying the LiPo battery. You don't want to buy a LiPo battery that you constantly have to recharge.

The team also suggests using female-male wires over soldering. Soldering is great for very small connections. It works well to connect pins to the IMU. However, it sucks for connecting longer wires to try to move the sensors off the breadboard. The soldering often breaks. It's also just hard to work with when you need to move a sensor around, but have to do it really fragilely. This project ended up using female-male wires to connect the BNO08X IMU. There were still problems because sometimes the wires would break inside or the breadboard ports would just die. However, it was easier to fix because we didn't have to go into the lab to resolder before being able to test again. Instead, you just buy more wires.

In addition, the team suggests realizing earlier that the problem may just be your breadboard. Throughout the process, our IMU would stop working multiple times. We first thought it was the IMU's fault. We did so many tests and fixes, but nothing would change. The last thing we did was just move the IMU to a different breadboard port and it worked again like magic. The breadboards in the lab have been used by many teams. They are not top-of-the-line anymore and it is very likely some of the ports will just

burn out on you. Check that first. Don't let yourself waste time looking for a problem right in front of you.

Lastly, work consistently on the project. It's easy to just push it aside because everything is due at the end of the semester, but this is a tough class. You have to make something completely new and actually make it work. You can't do that in one night. When we first started out in the semester, we mainly just worked for 2-3 hours whenever we had lab. We would only work once a week and to be honest, we were behind. We did end up doing well at the end and finishing a week before the video was due, but that was only because our team got stressed out and started cramming over spring break. Without that, it's hard to imagine that this project would be where it's at right now. Don't be us. Do the work every week and you won't be stressed. You just have to do it.

Helpful Courses

Several notable BME classes were useful for this project. BME 404: Computational Neuromechanics (formerly Orthopaedic Biomechanics) was useful for understanding degrees of freedom and interpretation of flexion/extension angles. BME 210: Biomedical Computer Simulation Methods, where programming in MATLAB is a focus, helped with writing firmware for sensor calibration with Arduino. Despite none of the team members having an electrical emphasis in biomedical engineering, having a basic understanding of wiring circuits on a breadboard because of EE 202: Linear Circuits was crucial for developing our MVP. We also utilized institutional knowledge by constantly talking to Professor Mai and Trent about our project. We also got helped from Ray Peck with soldering.

Contributions

The below table breaks down the contributions that each team member made to the project. Overall, the work was evenly distributed. Each team member had their respective strength. Max worked heavily on the report, Sophia worked heavily on the GUI and the presentation, and Elizabeth worked heavily on the wiring and coding.

Table 7. The contributions that each team member made to the project.

	Final Report	Videos	Outcome
Max	Worked on the whole report	Filmed the working test model (flex sensor and IMU)	Worked primarily on the test model (coding and wiring)
Sophia	Worked on the whole report	Voiceover	Worked on test model (coding and wiring) and designed GUI + presentation
Elizabeth	Worked on the whole report	Filmed and edited the working final model (2 IMUs)	Made the final model and 3D-printed casing (CAD)

Appendix

Lab Notebook

We spent 3-5 hours per week on the project. A total of $176.09 was spent. All three members of the team are biomedical engineering students with research experience. Elizabeth had significant experience in CAD and 3D printing. We were all involved with the Arduino coding of the device. IMUs, flex sensors, Li-Polymer battery, goniometer, and the knee model were readily available online. We ordered from Amazon and Adafruit.

Week 1 (Feb 25): We received the parts. We first started working on the IMU (BNO08X), wanting to test the initial measurement code on it. We originally tried to test the sensor by just plugging the pins into the breadboard without soldering the pins to the IMU. We soon realized we needed to solder the pin connections for it to work in the breadboard. After soldering, we tried out the initial Arduino code found online, but it didn't work. We made plans to read more into the BNO085 datasheet in order to find a solution. We also read academic journals to figure out how to get the knee angle from two IMUs.

Week 2 (Mar 4): We started off with reading more into the BNO08X datasheet in order to figure out how to wire up the IMU correctly. Eventually, we were able to figure out the right wiring. We basically needed to switch from trying I2C communication to SPI communication. This was because we were using an ESP32 at the time, and as the datasheet explained, I2C communication worked poorly on the IMU if an ESP32 microcontroller was being used. After switching to SPI, the BNO08X worked and we were able to write initial code. We then tried the Arduino Feather M0 with the BNO08X IMU and we're able to get the SPI communication to work there too. We made plans to start working on two IMUs at once for the next week.

Week 3 (Mar 11): This week was split into two parts: starting the GUI and trying to wire up two IMUs at once. For the GUI, we brainstormed what we wanted it to have. This included user-friendly feedback, a warning sign, and the measurement of a correct angle being ±10 degrees. We also decided that we didn't want it to be an actual functioning mockup, but mainly to display what the app would theoretically have. Sophia then started working on building the GUI. Elizabeth then worked on wiring up two IMUs (both BNO08X) at once. She originally used the feather M0, but could only get one IMU to work on SPI and one IMU to work on I2C at individual times. They would never work at the same time together. Individual codes only for SPI/I2C had to be run. After much trial and error, a plan was made to try two different options: using two Feather M0s or switching to using an ESP32 microcontroller. For next week, it was decided that the ESP32 would be tried first. The team also decided to reach out to Rachel Lin as she had a similar senior project last year. However, this message yielded very little help as the use of two IMUs for their project was extremely different.

Week 4 (Mar 25): Over spring break, Elizabeth worked on the two IMUs. She first tried the ESP32, but could only get it to connect to one IMU via I2C. She could not get the ESP32 to connect to one IMU using SPI communication. Thus, she wasn't able to get both IMUs to work at the same time

using an ESP32. Before trying two Feather M0s, we brainstormed some additional options, including trying two different IMUs at once (such as BNO08X and BNO055) and using a backup option. We wanted to try two different IMUs at once as they might have different communication lines. For example, with I2C, two BNO08X IMUs have the same I2C addresses. However, a BNO08X IMU and a BNO055 IMU would theoretically have different I2C addresses. Since the two IMUs working at once seemed like a very hard issue, we decided on a backup option of one IMU and one flex sensor. The flex sensor would measure the bending angle and the IMU would provide a 3D orientation to validate the flex sensor reading. The 2 IMU option is more accurate, but the flex sensor + 1 IMU option would provide a simpler test method to validate the knee angle measurement objective.

When we came back from spring break (the week of March 25th), we completed a multitude of tasks on the to-do list. Sophia worked on the GUI while she was out of town, and Max and Elizabeth split up the hardware tasks. Elizabeth first found the I2C address of the BNO055 IMU, which turned out to be 0x28 (different from the 0x4B I2C address of the BNO08X IMU). Elizabeth then got the BNO055 IMU to work by itself using I2C communication with the ESP32 Wroom and the Feather M0. Elizabeth then decided to use the Feather M0 to try I2C communication with the BNO055 IMU and SPI communication with the BNO08X IMU at the same time. After much testing and code editing, she was able to get both of the IMUs to work and send out data at the same time, accomplishing a big goal. At the same time, Elizabeth wired up the flex sensor and gave it to Max to work with. Max then calibrated the flex sensor so that we knew what flex sensor data matched with which knee bending angles. After completing this, Max started working on the CDR document, primarily the abstract and introduction.

For next week, the plan was to continue working on both models: 1 IMU + Flex Sensor (Max and Sophia) and 2 IMUs (Elizabeth).

Week 5 (Apr 1): To start this week off, Elizabeth soldered the pins to the third BNO08X IMU that we had. This was done so that Max and Sophia could test it with the flex sensor. Max and Sophia then worked on testing out the flex sensor + 1 IMU model. They were able to get some code to run, but still needed to further edit the code. At the same time, Elizabeth wrote some code to get the respective orientation values from the BNO055 IMU and the BNO08X IMU. Using the code, she was able to get the Game Rotation Vector from the BNO08X and the Euler Angles from the BNO055. The next step then became soldering these two sensors with longer wires to be able to test them out using the goniometer and write up code to take their orientation values and calculate the knee angle.

The Thursday of this week, Elizabeth soldered longer wires to the two IMUs for the preparation of attaching them to the goniometer.

Week 6 (Apr 8): Over the weekend, Elizabeth was able to test the BNO055 with the long wires, getting it to work. However, some of the long wires came off the BNO08X, so she couldn't test it. Then, on Tuesday, Professor Mai and Trent explained that soldering long wires straight to the pin out was not a good idea. Instead, they gave the team crimped female-male wires to use with the BNO08X. These were originally a lot easier to work with. Using the crimped wires, Elizabeth was able to test out the 2 IMU model.

To do so, she first had to attach the IMUs to the goniometer. She tried tape and velcro, but neither worked. However, Band-Aids did! Once the IMUs were attached, Elizabeth tested the code, making sure that both IMUs still worked together. She then worked on the code to find the knee angle, using the difference between the values from the two IMUs. However, as soon as she started testing it, the crimped

female-male wires broke. The IMU would no longer work when wires were connected to the breadboard. It would only work when directly plugged into the breadboard, which made it impossible to test out the knee angle.

After this incident, the team went to Ray Peck in the makerspace. He suggested buying more female-male wires or soldering. The team first tried soldering, using silicon wire as well since it is easier to move around. However, silicon wire is made of stranded wiring, which is nearly impossible to plug into the breadboard. The silicon wire was also soldered to additional pins, but no matter what was done, the IMU still failed to work with the soldering. Due to the lack of success, the team bought female-male wires and the IMU finally worked again. Using these new wires, a code that calibrated at 90 degrees was tested. When the goniometer was then moved to 60 degrees, a 30-degree difference was observed. Sometimes the measurement would be a little bit off (example: 28 degrees), leading the team to notice that a tolerance would be needed in the code for counting a lunge.

Max and Sophia did further bug fixing with the IMU + Flex Sensor. They made sure everything was properly wired to the corresponding pins and finally got it printing accurate knee angle measurements in the Arduino Serial Monitor.

Week 7 (Apr 15): This week, we were working on putting our video together. To start off, we wanted to record videos of both models, the Test Model (1 IMU + Flex Sensor) and the Final Model (2 IMUs), working. Unfortunately, we ran into an issue right away. One of the IMUs for the Final Model stopped working. We thought that it crashed out, but it turned out to be a breadboard issue. For some reason, certain ports in the breadboard would just stop working. Once we figured that out, we were able to plug the IMU into different ports on the breadboard and get it working again. For the video of the Final Model, the calibration was changed to 130 degrees and then the goniometer was moved to 90 degrees. This made it easier to get consistent values since the wires moved best in that position.

We then recorded working videos for both models and started working on the overall video. Elizabeth made the video, while Sophia provided the voiceover. All team members were involved in testing the device and recording it. This video was then submitted. During this week, the 3D-printed boxes were also worked on. The original design was printed, but the boxes were too big. Elizabeth then changed the box size to better fit the IMUs.

Week 8 (Apr 22): This week was mainly adding minor components to the project. First off, a button was added that would reset the code and rerun it. This would allow you to recalibrate the gyrometer and measure the angle again. An LED was also added that would turn off during the resetting process and then turn on when the calibration was complete. Secondly, the new boxes were printed. A circle center to represent the center of the gyrometer was also printed. This center will hold the PCB board and the battery in the final product. The final physical prototype was put together with the 3D printed parts being brought together by heat tubing melted over a plastic pipe. These materials will be replaced in the final product, but were used to show a model of what this product would likely look like.

Week 9 (Apr 29): This week was focused on the final presentation of our project. Elizabeth and Sophia worked on the presentation. It was an easy week until right before the presentation when the breadboard ports burnt out again. We had to find new breadboard ports that worked to make sure we could show a live demo. We were able to do so!

We then worked on the report. Max made additional edits to the abstract and introduction. He made most of the figures and tables, including the captions and descriptions. He also wrote the discussion section. Elizabeth made a lot of the software diagrams. She added the Engineering Drawings for the 3D-printed hardware casing she designed and created. She also made edits and suggestions for the whole report. Sophia finished up the GUI section. She also wrote the conclusion and made edits and suggestions.

Data Sheets

One data sheet the team used was the BNO085 Data Sheet. This Data Sheet provided the SPI code and wiring for the BNO08X IMU. It also provided the I2C code and writing for the BNO08X IMU. This Data Sheet also explained that one should not use the ESP32 for I2C connection with the BNO08X IMU because it was highly finicky. This Data Sheet also suggested using the Adafruit BNO08x Arduino Library to enable and read sensor reports. It also explained that this library only supports a single BNO08X sensor, a point that was helpful in leading the team to use a BNO055 IMU as the second IMU. It also explained how one will get quaternion values from this IMU and how to utilize them [18]. Here is the link to the data sheet: adafruit-9-dof-orientation-imu-fusion-breakout-bno085.pdf.

Another data sheet the team used was the BNO055 Data Sheet. This data sheet provided the I2C code and wiring for the BNO055 IMU [40]. Here is the link to the data sheet: adafruit-bno055-absolute-orientation-sensor.pdf.

A third data sheet that the team used with the Adafruit Feather M0 Wifi with ATWINC1500 Data Sheet. This data sheet provided information on the pinout for the Adafruit Feather M0 Wifi. This ensured that everything was wired up correctly [20]. Here is the link to the data sheet: adafruit-feather-m0-wifi-atwinc1500.pdf.

The data sheet for BNO08x was used for the table with maximum sampling rates [27]. Here is the link: https://www.digikey.in/en/htmldatasheets/production/3210483/0/0/1/bno080.

Conclusion

Knee injury rehabilitation requires constant monitoring and feedback due to issues with improper form, delayed recovery, and risk of reinjury. iKneed provides an elegant solution to this problem by delivering continuous, real-time biomechanical feedback during rehabilitation using IMU sensor technology and a microcontroller that communicates with a device.

In our final model, two IMUs (an Adafruit BNO055 IMU via I²C and a BNO08x IMU via SPI) provide complementary orientation readings, where the BNO055 is streaming Euler angles and the BNO08x is streaming quaternions. The associated code loops through bno08x.getSensorEvent() and bno055.getSensorEvent(), stores the calibration offsets at a 130° reference position, computes the live knee flexion by subtracting these from live quaternion data, and manages all sensor fusion and data parsing with setup(), loop(), and setReports() functions of the Adafruit_BNO08x, Adafruit_BNO055, Wire, and SPI libraries. A companion GUI then presents weekly progress, clinician goals, prescribed exercises, awards, and real-time performance metrics in order to maintain patient motivation and on-track.

By combining dual-IMU sensing, low-cost hardware, and a lightweight add-on structure, iKneed delivers accurate angle detection to ±10°, offering instant corrective feedback to patients and enabling remote clinician monitoring. The data-driven process can significantly improve exercise compliance, reduce the incidence of compensatory movement patterns, and reduce overall recovery times.

In the future, a controlled trial of standard home exercise programs versus iKneed-augmented rehabilitation would compare functional score improvement and reinjury rates. Other refinements through the inclusion of force-sensitive resistors to track loads, automated calibration protocols to detect brace misalignment, and AI-driven exercise instruction will continue to improve accuracy and user engagement. Expanding the mobile interface with adjustable accessibility options will further make the system inclusive for older or visually impaired users.

The iKneed is a promising innovation in home orthopedic rehabilitation. Its synergy of objective, real-time feedback and remote oversight can democratize high-quality rehabilitation, lower healthcare costs, and in the end, improve long-term patient outcomes for a variety of knee injuries.

Acknowledgements

Special thanks to Dr. John Mai, Trent Benedict, Ray Peck, and James Yoo for their support and advice. Thank you to the Alfred E. Mann Department of Biomedical Engineering for funding this project.

References

[1] J. Evans, A. Mabrouk, and J. I. Nielson, *Anterior Cruciate Ligament Knee Injury*. Treasure Island, FL: StatPearls Publishing, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK499848/

[2] "Total Knee Replacement." OrthoInfo. https://orthoinfo.aaos.org/en/treatment/total-knee-replacement (accessed May 1, 2025).

[3] S. R. Naqvi, R. C. Beavis, P. Mondal, R. Bryce, and D. A. Leswick, "Incidence Rates of Surgery After Knee MRI: Association According to Referring Physician Type and Patient's Age and Sex", *Orthopaedic Journal of Sports Medicine*, vol. 9, no. 11, Nov. 2021, doi: 10.1177/23259671211052560

[4] "Wearable Medical Devices Market - By Device, Application, End Use - Global Forecast 2024 - 2032." GMI. https://www.gminsights.com/industry-analysis/wearable-medical-devices-market (accessed May 5, 2025).

[5] "Knee Braces Market Size, Share & Trends Analysis Report By Product (Prophylactic, Functional, Rehabilitative, Unloader), By Application (Sports, Ligament), By End Use, By Region, And Segment Forecasts, 2025 - 2030." GII. https://www.giiresearch.com/report/grvi1631541-knee-braces-market-size-share-trends-analysis.html#:~:text=Knee%20Braces%20Market%20Size%2C%20Share%20%26%20Trends%20Analysis,are%20the%20key%20growth%20boosters%20for%20this%20market. (accessed May 5, 2025).

[6] "ISO 13485." ISO. https://www.iso.org/iso-13485-medical-devices.html (accessed May 1, 2025).

[7] "ISO 14971:2019." ISO. https://www.iso.org/standard/72704.html (accessed May 1, 2025).

[8] "IEC 60601-1-11:2015." ISO. https://www.iso.org/standard/65529.html (accessed May 1, 2025).

[9] "Electromagnetic Compatibility (EMC) of Medical Devices." FDA. https://www.fda.gov/media/94758/download (accessed May 1, 2025).

[10] "Significant Risk and Nonsignificant Risk Medical Device Studies." FDA.

https://www.fda.gov/regulatory-information/search-fda-guidance-documents/significant-risk-and-nonsignificant-risk-medical-device-studies (accessed May 1, 2025).
[11] “Part 820 - Quality System Regulation.” Code of Federal Regulations. https://www.ecfr.gov/current/title-21/chapter-I/subchapter-H/part-820 (accessed May 1, 2025).
[12] “Informed Consent FAQs.” U.S. Department of Health and Human Services. https://www.hhs.gov/ohrp/regulations-and-policy/guidance/faq/informed-consent/index.html (accessed May 1, 2025).
[13] “HIPAA Privacy Rule.” U.S. Department of Health and Human Services National Institutes of Health. https://privacyruleandresearch.nih.gov/ (accessed May 1, 2025).
[14] “K210604.” FDA. https://www.accessdata.fda.gov/cdrh_docs/pdf21/K210604.pdf (accessed February 5, 2025).
[15] “K220738.” FDA. https://www.accessdata.fda.gov/cdrh_docs/pdf22/K220738.pdf (accessed February 5, 2025).
[16] “Persona IQ.” Zimmer Biomet. https://www.zimmerbiomet.com/en/products-and-solutions/specialties/knee/persona-iq.html (accessed February 5, 2025).
[17] E. A. Belalcazar-Bolanos, D. Torricelli, and J. L. Pons, “Automatic Detection of Magnetic Disturbances in Magnetic Inertial Measurement Units Sensors Based on Recurrent Neural Networks”, *Sensors*, vol. 23, no. 24, Dec. 2023, doi: 10.3390/s23249683
[18] “Adafruit 9-DOF Orientation IMU Fusion Breakout - BNO085.” Adafruit. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-9-dof-orientation-imu-fusion-breakout-bno085.pdf (accessed February 25, 2025).
[19] “MPU-9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking Device.” TDK. https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/ (accessed May 1, 2025).
[20] “Adafruit Feather M0 Wifi with ATWINC1500.” Adafruit. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-feather-m0-wifi-atwinc1500.pdf (accessed February 25, 2025).
[21] “analogRead() - Arduino Reference.” Geeks for Geeks. https://www.geeksforgeeks.org/analogread-arduino-reference/ (accessed May 5, 2025).
[22] “The Differences Between I2C and SPI (I2C vs SPI).” Total Phase. https://www.totalphase.com/blog/2021/07/i2c-vs-spi-protocol-analyzers-differences-and-similarities/ (accessed May 5, 2025).
[23] H. Dewey-Hagborg. “Introduction to the Serial Peripheral Interface.” Arduino. https://docs.arduino.cc/tutorials/generic/introduction-to-the-serial-peripheral-interface/ (accessed May 1, 2025).
[24] “ISO 13485:2016(en).” ISO. https://www.iso.org/obp/ui/en/#iso:std:iso:13485:ed-3:v1:en. (accessed May 1, 2025).
[25] “Adafruit 9-DOF Orientation IMU Fusion Breakout - BNO085 (BNO080) - STEMMA QT / Qwiic.” Adafruit. https://www.adafruit.com/product/4754 (accessed February 25, 2025).
[26] “Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055.” Adafruit. https://www.adafruit.com/product/2472 (accessed April 1, 2025).
[27] “BNO080 Datasheet by SparkFun Electronics.” DigiKey. https://www.digikey.in/en/htmldatasheets/production/3210483/0/0/1/bno080 (accessed May 1, 2025).

[28] “BNO055.” BOSCH. https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf (accessed May 1, 2025).
[29] “Adafruit Feather M0 Wifi - ATSAMD21 + ATWINC1500.” Adafruit. https://www.adafruit.com/product/3010 (accessed March 2, 2025).
[30] “nRF52 DK.” Nordic. https://www.nordicsemi.com/Products/Development-hardware/nRF52-DK (accessed May 1, 2025).
[31] “Defining Excellence in Ethical Electronics Recycling and Reuse.” e-Stewards. https://e-stewards.org/ (accessed May 1, 2025).
[32] “Teyleten Robot GY - BNO085 AR VR IMU High Accuracy Nine-Axis 9DOF AHRS Sensor Module Better than BNO080 BNO055.” Amazon. https://www.amazon.com/Teyleten-Robot-BNO085-Accuracy-Nine-Axis/dp/B0CL26J81F (accessed February 10, 2025).
[33] “KBT 3.7V 1000mAh Li-Polymer Battery: 523450 Lipo Rechargeable Lithium-ion Replacement Batteries with PH 2.54 JST Connector, PH1.25/2.0 JST Connector for Replacement - 2Pack.” Amazon. https://www.amazon.com/KBT-3-7V-1000mAh-Li-Polymer-Battery/dp/B0BJPFR756/ref=asc_df_B0BJPFR756?tag=bingshoppinga-20&linkCode=df0&hvadid=80814294283644&hvnetw=o&hvqmt=e&hvbmt=be&hvdev=c&hvlocint=&hvlocphy=&hvtargid=pla-458441376285506 2&msclkid=355bea20ccb91263a6468d8a94949d0a&th=1 (accessed February 10, 2025).
[34] “Baseline 12-1025 Absolute Axis 360 Degree Plastic Goniometer, 12” Length.” Amazon. https://www.amazon.com/Baseline-12-1025-Absolute-Plastic-Goniometer/dp/B008N3SQJ8 (accessed February 11, 2025).
[35] “MIRR Human Knee Joint Model with Knee Ligament, Life Size Anatomical Knee Joint Flexible Skeleton Model, Perfect for Medical Learning and Teaching.” Amazon. https://www.amazon.com/MIIRR-Human-Joint-Model-All/dp/B09VTJ4M9G?th=1 (accessed February 11, 2025).
[36] “OIIKI Goniometer Set 6” / 360 Degree.” Amazon. https://www.amazon.com/Goniometer-inch-360-Degree-Plastic/dp/B0DHRST8M8?th=1 (accessed February 11, 2025).
[37] “Long Flex sensor.” Adafruit. https://www.adafruit.com/product/182 (accessed March 24, 2025).
[38] “40 PCS 20 CM (8 inch) Breadboard Jumper Wires Length Optional Dupont Wire Assorted Kit Female to Male Multicolored Ribbon Cables.” Amazon. https://www.amazon.com/California-JOS-Breadboard-Optional-Multicolored/dp/B0BRTHR2RL/ref=sr_1_5?crid=20QG6W8D58JAD&dib=eyJ2IjoiMSJ9.tjHxIQLJsk16_0YVtUGN6Tqnr8euWNsWVjpSaq5RQkb9i7FP8hWDcWd7nUWsAPlVs-09xUf804joVmKfvzdMsLVIcM6m57tHYW8Lq9FNganYmcFeVxjreqJG92xjFxaArS2EfqYwb7F0pik0dmzDlotklo6MnaTDQFFvPnECsLf4ZXDW4TlY6YxMGHQGvWIDRNRlRFWl1Z5JBRbMG7_rLyS1sfp8g6f23-oeKFOWooQ.opP1FcCMTIIlz28hZTuHSkC3g_OjelcKc4aVFHq3Xxo&dib_tag=se&keywords=male%2Bfemale%2Bwires&qid=1744161654&sprefix=male%2Bfemale%2Bwires%2Caps%2C209&sr=8-5&th=1 (accessed April 8, 2025).
[39] “ISO/IEEE 11073-20701:2020.” ISO. https://www.iso.org/standard/78227.html (accessed May 5, 2025).
[40] K. Townsend. “Adafruit BNO055 Absolute Orientation Sensor.” Adafruit.

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-bno055-absolute-orientation-sensor.pdf (accessed April 1, 2025).