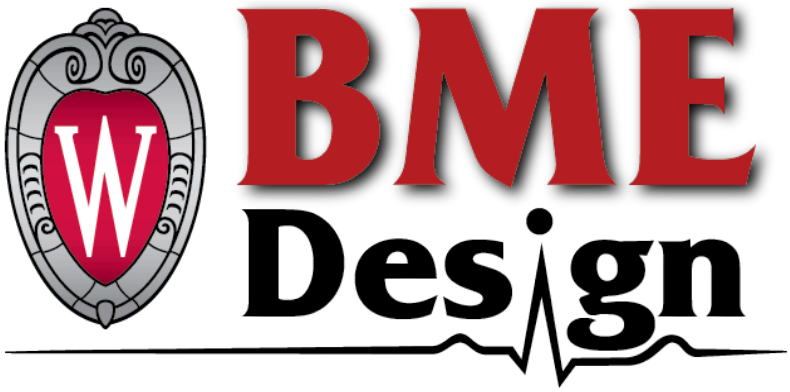

Prevention of Weightlifting Injuries via Barbell Modifications

Final Report

Dec 11, 2024

Biomedical Engineering 200/300

Client: Mr. Robert Gold

Advisor: Dr. William Murphy

Team Members:

Jackson Jarrett jrjarrett2@wisc.edu (Leader and BWIG)

Kai McClellan kamcclellan@wisc.edu (Communicator)

Gavin Gruber gtgruber@wisc.edu (BPAG)

Luke Schmeling laschmeling@wisc.edu (BSAC)

Table of contents

Abstract

Every year 27,000 people injure themselves while weightlifting [1]. Users have a much higher likelihood of injury with inadequate form [2]. The team’s solution to prevent weightlifting injuries is to isolate the root issue of injury for users and create a device that will help the user fix improper form while performing a complex lift such as the bench press. Competing devices attempted to fabricate devices with similar function, but focus primarily on force metrics and for a much higher cost per unit than what the team's device will cost. The design team’s biomedical device prevents injury by targeting, identifying, and correcting improper form.

Introduction

Problem Statement

Up to 27,000 injuries occur while weightlifting every year. Injuries are often caused by an uneven distribution of load on the barbell, leading to the weight lifter favoring one arm over the other [1]. The team has been tasked with designing a biomedical device that can prevent weight lifting injuries by targeting, identifying, and correcting improper form. This biomedical device must track the barbell path across all 3 axes during kinetic motion, allowing the user to note and make necessary changes to their form in order to prevent injury.

Background/Motivation

There are usually four types of major weightlifting injuries possible that can occur regardless of the lift which are strains, sprains, impingements and tears [1]. For our purposes the lift the team has been tasked with focusing on is bench press. Bench press is performed lying on your back whilst holding the barbell above one's head, you then lower the barbell to below the nipple line and then press the bar back above one's head. The two most common injuries while performing bench press are impingements and tears [2]. The reason for this is due to a space called the subacromial space. The subacromial space lies between the coracoacromial arch above and the humeral head below [3]. It contains the rotator cuff tendons, the long head of the biceps tendon, the shoulder joint capsule, and more important ligaments [3]. When performing bench press with improper form (the angle between the side and the elbow being between 70-90 degrees [2] the acromion bone or the coracoid process impedes upon the subacromial space and can snag onto the tendon causing it to become inflamed. After this occurs any repetitive use of

the shoulder can cause the now inflamed tendon to snag further on the bone and with enough repetition can cause a tear in the tendon to occur. The team's solution to this occurrence is to be proactive in preventing injuries instead of reactive by teaching and giving users feedback on proper form so that the possibility of injury while performing bench press is as low as possible.

Considerations and Specifications

The primary objective of this device is to monitor and assess the user's risk of injury during bench press exercises by identifying improper form. Specifically, the system is designed to detect deviations in barbell alignment, such as when the barbell is not parallel to the shoulders or is not level. Additionally, the device must track the path of the barbell and compare it to an ideal trajectory. Based on the analysis, the system will provide feedback to the user in an intelligible format, with guidance on how to correct any identified form issues to reduce the risk of injury. Beyond its functional requirements, the device must be suitable for use in commercial gym environments, allowing for easy transportation and setup by a single individual. Moreover, the product should not interfere with the original functionality of essential workout equipment, such as barbell clips, which must continue to prevent weight slippage, and wrist straps, which must maintain wrist support as intended. This ensures the device integrates seamlessly into existing gym practices while offering enhanced injury prevention capabilities.

Competing Designs

Injuries associated with weightlifting have been a persistent issue for many years. This has led to the development of various devices and applications aimed at tracking the barbell and enhancing the lifting experience. One of the most widely used mobile applications for tracking

barbell movement is the WL Analysis - Bar Path Tracker. This app utilizes the phone's camera to monitor the barbell and collect relevant data. Specifically, it tracks metrics such as bar path, velocity, horizontal displacement, and force [4], [5]. While this app is free to use, it has limitations. It only collects data from one side of the barbell, which means it cannot assess whether the barbell is parallel to the shoulders or if it is level. Consequently, it is unable to provide a complete analysis of form that is crucial for injury prevention.

Several competing designs that attach directly to the barbell are available, offering features similar to our proposed design. Two such products are Bar Sensei and Flex by GymAware. Bar Sensei is a sleeve that fits onto the center of the barbell, using an accelerometer to measure bar speed, acceleration, and both vertical and horizontal displacement. However, it does not provide data on the barbell's path or whether it is level. This device is primarily focused on tracking strength and power metrics rather than injury prevention, with a price of $499 [6].

Flex by GymAware operates in a similar manner to Bar Sensei but utilizes a laser optic system instead of an accelerometer for data collection. Additionally, it offers a visual representation of the barbell’s path. Nevertheless, it still lacks the capability to determine whether the barbell is parallel to the shoulders or level [7]. The Flex device is priced at $599.

Preliminary Designs and Evaluation

Design 1: Motion System Sensor

Figure 1. Motion System Design. A camera will be set up to track direct movement of the user, and this data will be recorded.

Design 2: Modified Barbell Weightlifting Clip Design

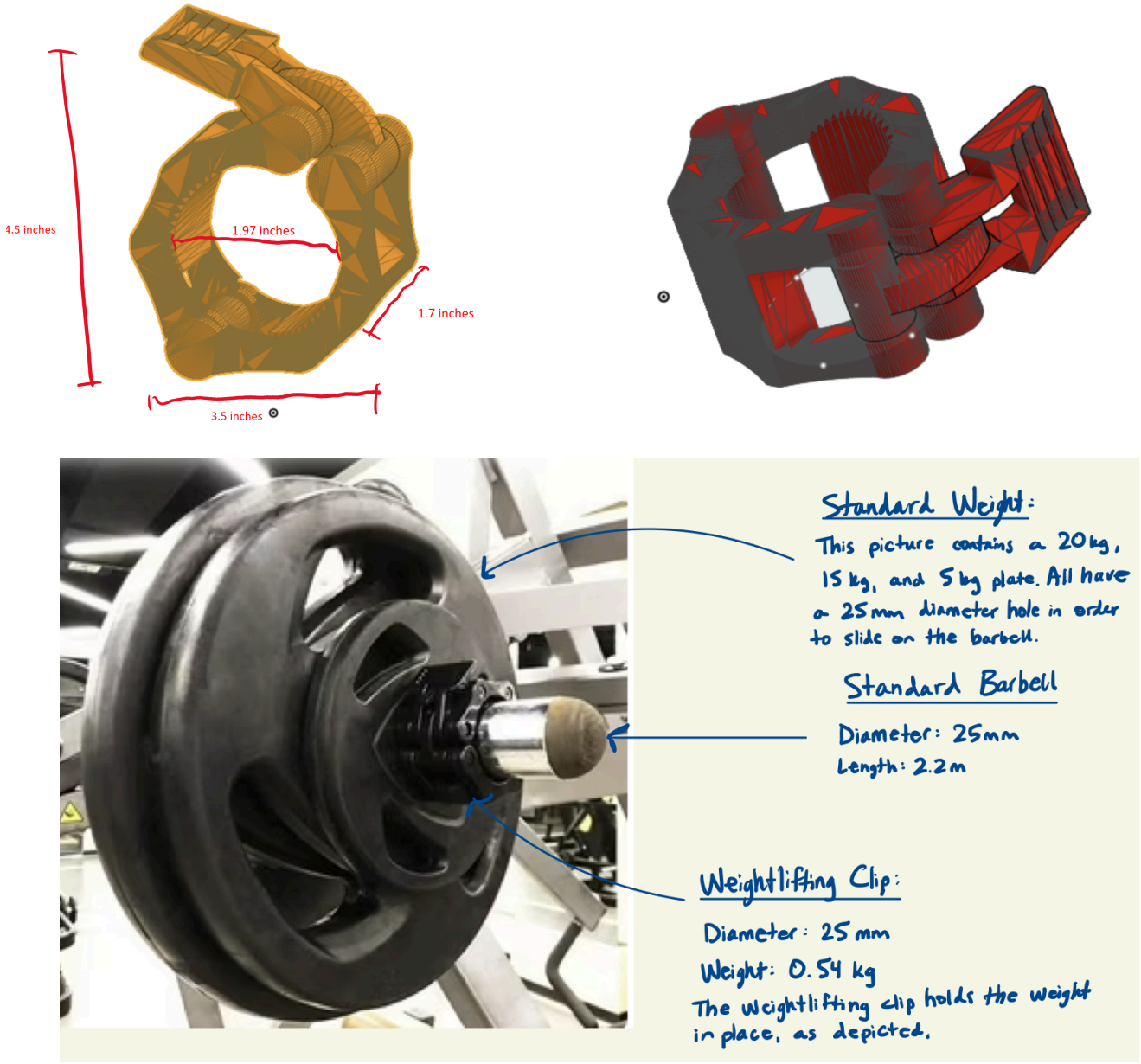

Figure 2. Barbell Weight Clips Design. An accelerometer will be housed within a modified but functional weight lifting clip that is used to clasp weight in place on the barbell.

Design 3: Wrist Strap Design

Figure 3. Wrist Straps Design. An accelerometer will be housed within a modified set of functional wrist straps that are used for wrist support when performing complex lifts such as the barbell bench press.

Design Matrix

Table 1. Design matrix for the Barbell Modifications to prevent injury.

Design Categories (Weight/100)	**Motion System**		**Barbell Weight Clips**		**Wrist Straps**	
Precision (30)	5/5	30	4/5	24	4/5	24
User Comfort (25)	5/5	25	5/5	25	3/5	15
Ease of Use (20)	2/5	8	5/5	20	4/5	16
Maintenance (10)	3/5	6	5/5	10	4/5	8
Ease of Fabrication (10)	2/5	4	4/5	8	3/5	6
Cost (5)	1/5	1	3/5	3	4/5	5
Total Points:	74		90		74	

Criteria Descriptions and Evaluation

Precision:

The precision category depicts how well the device tracks the movement of the barbell. Using existing literature and technical data, an exponential relationship will be expressed as the line of best fit. This line of best fit will represent the path of the barbell that will result in the best form and lowest injury rate when performing a complex lift such as the barbell bench press. Depending on the design model, this data will be collected in different ways. The precision of the injury prevention device will be quantified in inches, with an acceptable range of ±0.5 inch from the determined line of best fit. Precision is the category that holds the most weight due to its crucial impact on the project. A narrow range is required to ensure the success of the product, as well as the wellbeing and safety of the user.

The Motion System was scored a 5/5 and assessed a 30 for its weighted score in the precision category on the design matrix. The Motion system as shown is a proven technology that can diagnose and assess data at the highest level. Quantitatively, this technology would succeed our requirements of precision. Both the barbell weight clips and wrist strap designs are comparable in the precision category of the design matrix. The Barbell weight clips design as well as the wrist strap design would both utilize Arduino microcontroller and accelerometer technology in order to assess and track barbell movement. This technology will be successful and will be tested to be within the acceptable range of ±0.5 inch, thus we scored both designs with a ⅘ and a weighted score of 24. Solely looking from a precision standpoint, the motion system would be the ideal design to move forward with.

User Comfort:

The user comfort category represents the degree to which the device is noticeable to the user while performing the lift. Comfort is a universal necessity when designing a product to be used on or by the human body. This classification is especially important because of the environment in which the device will be used. When undergoing a complex movement such as the barbell bench press, the user will be putting their body under great stress by pushing their physical limits. With the prevention of injury as our primary goal, the comfort of the user while undergoing these actions have been highlighted as a very important class within our design matrix.

The Motion system was scored a 5/5 and assessed a 25 for its weighted score in the user comfort category on the design matrix. The Motion system involves no contact with the user itself, thus giving it no way to discomfort the user in any way from a physical standpoint. The same goes for the barbell weight clips design, which was also scored a 5/5 and assessed a 25 for its weighted score. The weight clips design would be attached to the barbell itself, and would not have any contact with the user while performing a repetition. The wrist straps design was scored a ⅗ and assessed a 15 for its weighted score in this category. This difference stems from the direct contact between the user arm and the wrist strap. Wrist straps, when functional, can make a positive impact on the user's wrist stability when performing a lift such as the barbell bench press. However, it is material dependent in terms of user comfort, and with the addition of the sensor technology being added to the wrist strap, we have assessed the design lower when compared to the motion sensor and barbell weight clips design. Solely looking from a user comfort standpoint, either the barbell weight clamps or the motion system would be the ideal designs to continue forward with.

Ease of Use:

The ease of use category represents how easy it is for the user to both setup and use while lifting. Making our product easy to use is important because we want our product to be readily available for all users. We also want our product to be able to be used in regular commercial gyms, so our product can't take too much time to set up or use, otherwise it would hinder the users' lifting experience.

The Motion system scored a ⅖ for this category and had an 8 for its weighted score. This is because in order to use the motion system, you would need to set up a camera in the gym and make sure it won't be disturbed and it can see you at the right angle when you are benching. It would be a struggle to find enough space to put the camera at a suitable distance away from the bench in many commercial gyms. This is not the case for the barbell weight clip design which scored a 5/5 and a 20 weighted score in this category. The barbell weight clip design wouldn't take any more work to use than using a regular bench clip. All you would have to do is bring the clips into the gym and slide them on the barbell. The one problem with this is many people don't like benching with clips on without a spotter because it can be more dangerous, however, you can just slide the clips on without clamping them onto the bar which would allow the weight to slide off if needed. The wrist strap design scored a ⅘ and a weighted score of 16. This design would like to be easy to use in any gym as it doesn't take up any space, you would just need to bring it into your gym. The reason it isn't a 5/5 is that you need to learn how to put on wrist straps and how to bench with wrist straps. This is not hard to learn or do, but it is one extra thing the user would have to learn before using the design. Looking only from a ease of use perspective, the barbell weight clamps would be the ideal design to continue with.

Maintenance:

The maintenance category represents how hard and how much work the design would be to maintain and keep working. Making sure our product doesn't require too much maintenance is important because it would deter a lot of people from using it, and if there was a lot of maintenance it would be much harder for our client to use for a long time. While maintenance isn't the most important category it is still essential to make sure our product isn't hard to maintain and will not break easily.

The motion system scored a ⅗ in this category. The motion system has some things that would regularly need to be maintained. The camera lenses need to be cleaned if they are ever dirty and it needs to be stored inside where the camera would not be broken. You would also have to check to make sure the camera software is working properly with the camera, and it would be very hard to fix anything if it breaks. The barbell weight clamp scored a 5/5 in this category because there is almost nothing you would need to do to maintain it. The only thing that would need to be replaced is the batteries whenever they run out of charge. The wrist strap design scored a ⅘ because the materials for the strap on the wrist strap need to be replaced whenever there is any damage to them or if there is to much wear and tear on them. The wrist straps would get worn out much quicker because there is tension on the straps whenever they are in use so the material would eventually deteriorate, and would need to be replaced. Solely looking from a maintenance standpoint, the barbell weight clamps would be the ideal design to continue with.

Ease of Fabrication:

The ease of fabrication is a necessary constraint to consider. If there is an easy, realistic, and valid design that does not take any shortcuts or lack taking any variables into consideration which works just as well if not better than an equally valid design, which takes high amounts of time and requires an abundance of trial and error to fabricate, the design to be selected will most assuredly be the former.

The motion system scored a ⅖ in this category. The idea centered around using code referred to us by our client that centered around cameras which was believed to be applicable to this project. Upon further review the code was determined to be beyond the scope of our knowledge and possibly not even able to be applied to our system and situation in a realistic manner. The barbell clamp received a ⅘ in the ease of fabrication category. With limited technology needed to determine a coordinate system with which barbell movement can be tracked and a fabrication process as simple as trying to find a way to attach a little chip in the proper orientation to a barbell clip, this option is a very realistic possibility. The wrist straps received a ⅗ in the ease of fabrication category. This in large part was due to the larger variability in being able to receive accurate measurements due to the possibility of unequal placement of the wrist straps on the wrist. The team would also need to find a way to attach the motion chips needed to track movement into the wrist straps without hindering mobility of the wrist or making them too bulky which is not as much of a concern regarding the barbell clamps. Solely looking from the ease of fabrication perspective the barbell clamps would be the ideal route to take.

Cost:

The cost category represents the expenses that will be incurred in the production of the design. Due to our allotted budget of $200, it is imperative that we do not exceed this amount in order to create a fully functional and thoroughly tested device. While this category may not be a pressing concern for some ideas brought forward in the design matrix, it remains nonetheless important to keep in mind currently in the decision making process, but also throughout the duration of prototyping, fabrication, testing, and final design.

The motion system scored a ⅕ in the cost category. This was in large part due to the costs that would be incurred buying two suitable cameras for the software as well as tripods on which they (the cameras) must stand. The estimated cost for the cameras alone would be somewhere in the range of $600 to $1000 which makes the motion system unrealistic for this aspect of the project. The barbell weight clips scored a ⅗ in the cost category and the wrist straps scored a ⅘ . The decision behind the rankings for these two housing forms was comparative. While neither option is incredibly cheap (hence why no housing form received a 5/5 ranking), purchasing wrist straps is cheaper than the purchase of two barbell clamps. From a solely cost effective perspective this leaves wrist straps as the ideal route to take.

Final Design

The design team decided to move forward with the Modified Barbell Weightlifting Clip design. This is due primarily to its high scoring in the user comfort, ease of use, maintenance, cost, and ease of fabrication categories. Its precision may have been comparatively less than the motion sensor design, however the cost and ease of use of the motion sensor design deemed the product unusable for the project at hand. The versatility and functionality of the modified barbell weightlifting clip design as shown in the design matrix led the team to move forward with this design. The design can be separated into external and internal sectors, with further details and specifications provided below on the specific components of the final prototype.

External Components

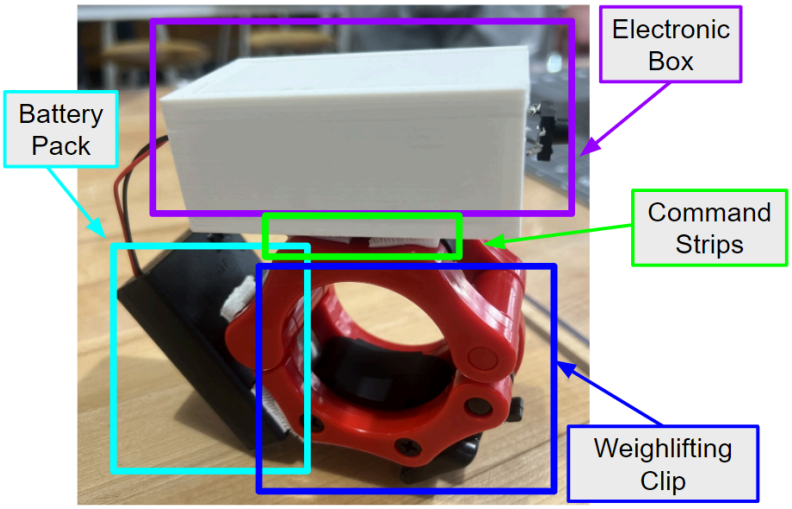

Figure 5. External View of Modified Weightlifting Clip, showing battery pack, electronic box, command strips, and weightlifting clip

Weightlifting Clip

The weightlifting clip was purchased after the Makerspace staff advised against the 3D printing of the weightlifting clip. This advice was based on the intent to protect the functionality of the weightlifting clip itself. The clip is designed to have a tight fit around a standard barbell with a diameter of 25 mm (1 inch) [8], [9].

Electronic Box

The Electronic box was 3D printed in the Makerspace. The function of this box was to house the necessary technology, such as the MPU-6050 and the Arduino Nano, in the smallest possible volume of space. It also had to be easily attachable to the existing weightlifting clip technology. For further information of the design and fabrication of the electronics box, please see the "*Electronic Box*" heading in the *Methods* section, as well as Appendix B.

Battery Pack

The battery pack component consists of a black box that can hold 3 AA batteries. These 3AA batteries power the breadboard which rests within the electronic box. The battery pack wires were soldered and extended so that they were able to reach and power the breadboard.

Command Strips

The command strips were cut and attached to the points on the weightlifting clip as depicted in Figure 5. The command strips served as connection points for both the battery pack and the electronics box.

Internal Components

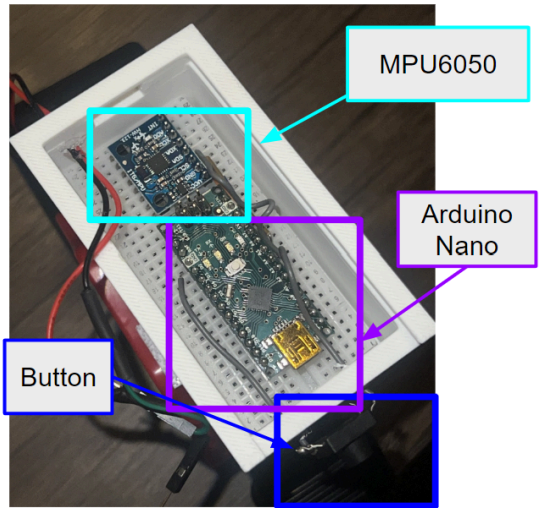

Figure 6. Internal View of Modified Weightlifting Clip, showing MPU-6050, Arduino Nano, and button

MPU-6050

The MPU-6050 sensor integrates a 3-axis accelerometer and 3-axis gyroscope, allowing measurement of angular velocity and acceleration. It communicates via the I2C protocol to the Arduino Nano[10], [11].

Arduino Nano

The Arduino Nano is the microcontroller hub of information that the design team decided to use for the collection and communication of information from the MPU-6050. Using the EEPROM data collection and storage system, acceleration data was collected from kinetic movement, stored on the Arduino Nano, and then read on MATLAB.

Button

The button in the prototype is incorporated into the EEPROM data collection through Arduino. When the button is pressed, the EEPROM data storage is cleared, and also begins collection of a new set of data. The incorporation of the button into the prototype was very important to prevent

the infinite collection of data, and allow us to differentiate data collection on set by set, or repetition by repetition basis.

Materials and Methods

Materials

The materials needed to construct the modified weightlifting clip are 2 barbell clips, 2 arduino nanos, 2 MPU-6050s, 2 battery packs that hold 3 batteries, 8 medium command strips, 2 bread boards, 2 half inch buttons, and at least 25 cm of wire. During the construction of this device you will also need hot glue and a hot glue gun, as well as a soldering iron and some solder. You will also need a drill with a 6 mm drill bit that will be used to drill a hole in the 3D printed box to the wires from the battery. Refer to Table 2 below and Appendix F for a detailed table of the materials needed for this project and their cost.

Table 2. Materials and their description used for the prototype

Material	**Item Description**
External Components	
Weightlifting Clip	
Nylon Resin	The frame of the clip is made of nylon resin with injection molded pressure pads
Electronics Box	
PLA Plastic[12]	The 3D printed box is made of PLA plastic
Hot Glue	The wires that are inserted into the box are sealed with hot glue to keep them in place and keep water out.
Battery Pack	
ABS Plastic	The housing of the batteries are made of ABS plastic
Wires	The battery pack has 2 aluminum wires that are used to connect the batteries to the breadboard.
Command Strips	

Synthetic Rubber	The synthetic rubber is used as the adhesive that holds the command strip to the barbell clip.
ABS Plastic	The velcro like surface of the command strips are made of ABS Plastic which hold the battery pack and the electronics box to the barbell clip
Internal Components	
MPU-6050[11], [13], [14]	This sensor module is used to collect acceleration and angular acceleration data.
Arduino Nano	This arduino is used to control, collect, and store data from the MPU-6050.
Breadboard	The breadboard is used to connect all the electrical components together and keep everything organized and secure.
Button	½ inch button used to turn data collection on in the MPU-6050
Wires (copper)	Copper wires used to connect the arduino to the MPU-6050.

Methods

Fabrication of Non-Electronics Component

Electronics Box

The box that contains the electronics is 3D printed out of PLA plastic. The box has an interior that is 7.62 cm x 3.81 cm and is 3.048 cm tall. The walls are all .508 cm thick. The exterior dimensions of the box are 8.636 cm x 4.826 cm x 4.064 cm. There is also a sliding lid that fully covers the box and has a latching mechanism that locks the lid on. For a detailed description of how the box and lid were designed and fabricated using OnShape refer to Appendix B. After 3D printing the box drill a hole with a 6 mm drill bit 1.2 cm from the top right and 1.4 cm down from the corner on the side with the hole for the latching mechanism (Shown in figure 7 below).

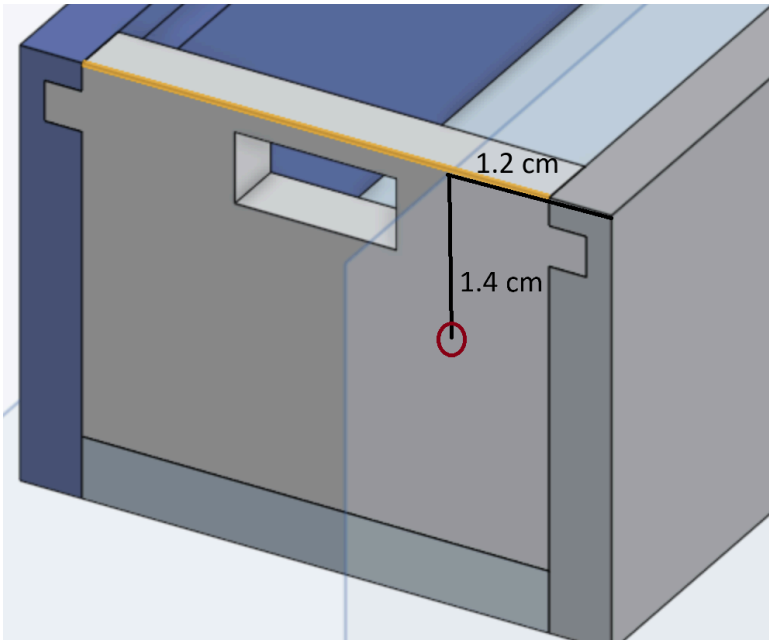

Figure 7. Shows where on the 3D printed box to drill the 6mm hole that the wire from the battery pack will go through.

Fabrication of Electronics Component

The electronics component consists of the breadboard, Arduino Nano, MPU-6050, and the button. These electronics are powered either by USB C - Micro USB connection through a computer, or through the power of the battery pack, which holds 3 AA batteries. The breadboard holds the Arduino Nano and MPU-6050, and allows for connection via battery pack as well as I2C communication. In order to connect the Arduino Nano to the power source, either the wires from the battery pack would have to be inserted into the VIN and GND ports, or the USB C - Micro USB connection would have to be active. The Arduino Nano communicates to the MPU-6050 via I2C communication as depicted in Figure 8.

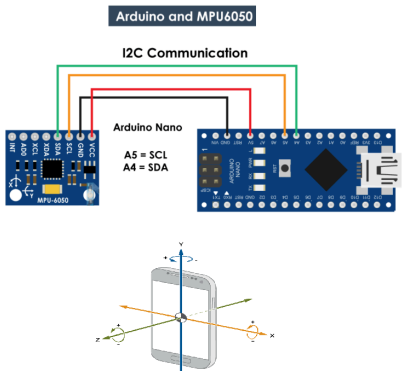

Figure 8. Depicts I2C Communication between the Arduino Nano and MPU-6050.

The button is connected to the Arduino Nano via wire. It is connected at the GND port and at the set A4 port. For more on the function of the button please see the *Button* segment in the *Internal Components* section under *Final Design*. The Arduino code that allows the button to start and clear the EEPROM data collection can be found in Appendix C.

Attaching Components to Barbell Clip

The command strips were cut in half and attached to the clip opposite of the lever that is used to open and close the clip (shown in figure 5). The command strips were then placed on the bottom of the 3D printed box and the battery pack. Then the 3D printed box and the battery pack were attached onto the barbell clip using the command strips. The wire from the battery pack was then inserted into the hole that was drilled into the 3D printed box. The wire should not have any extra wire outside of the box but it should not be pulled tight. Hot glue was then applied around the wire on the 3D printed box to secure the wire in place and prevent water from getting into the box. The breadboard with all the electronics wired up as described in the section above should then be stuck onto the bottom of the interior of the 3D printed box using the adhesive on the bottom of the breadboard.

Testing

Battery Life Testing Protocol

The battery pack that the design team moved forward with for the final prototype, and then into testing, contained 3 AA batteries and an on/off switch. To power 5 volts of power, 3AA batteries are assumed to last 3-5 hours[15]. The battery life of the prototype was deemed necessary for testing to ensure that the battery life would last for the entirety of the average workout. In order to test the battery life, we held three trials. For every trial, we would place a new set of batteries in the battery pack, and then would power on the device. Approximately

every 30 minutes for the first 3 hours, we would check that the device was still powered. After the 3 hour mark, we would check more frequently in order to get a more exact, quantitative result for battery life. For more on the battery testing protocol, please see the *Battery Testing Protocol* section in Appendix D. For the results of this section of testing and its implications, please read the *Battery Testing Results* section

Water Protection Testing Protocol

The protection of the prototype from water damage was deemed necessary for testing due to its importance. Because of the nature of our latch feature in the electronics box, as well as the holes drilled for the wires from the battery pack, the design team understood that the device was not going to be completely waterproof. However due to the possible implication or exposure to water or sweat in the gym, we wanted to ensure the success of the device if exposed to small amounts of liquid. To replicate the possibility of having sweaty hands when handling the device, hands were run under water for 3 to 5 seconds and then immediately grabbed the device after. In order to test the exposure to small amounts of water, a test was induced from approximately 60 cm (~2ft) away with the bottle set to deliver a wide spread of mist. For further information on the water protection testing protocol, please see the *Water Protection Testing Protocol* section in Appendix D. For the results of this section of testing and its implications, please read the *Water Protection Testing Results* section.

Accuracy Testing Protocol

Accuracy of the prototype was deemed the most important quantitative metric. In order to test this the design team met at the Nicholas Recreation Center on campus. We wanted to perform kinetic movement testing of the prototypes on each end of the barbell while performing bench press repetitions. Because of the importance of this data, we decided to perform three trials. For further information on the accuracy testing protocol, please see the *Accuracy Testing Protocol* section in Appendix D. For the results of this section of testing and its implications, please read the *Accuracy Testing Results* section.

Results

Battery Life Testing

The series of three AA batteries were turned on and left connected to the Arduino Nano for approximately 4.5hrs until it was no longer able to power the device. This was done three separate times to ensure consistency in the quality and performance of the batteries. During this time, no testing was performed to ensure that the voltage output had not changed. The results well exceeded the baseline value of 45 minutes, the average time it takes for a person to complete a workout.

Test 1	Test 2	Test 3
~4.42hrs	~4.49hrs	~4.59hrs

Table 3: Time-based performances of AA Batteries for Device Power Source.

Water Protection Testing Results

The device was subject to two water tests to simulate the accumulation of sweat a user might experience on their hands and to mimic the uses of cleaning spray commonly used in public gyms. For the sweat test, hands were run under water for 3 to 5 seconds and then immediately grabbed the device after. Results bore no noticeable accumulation of water inside the housing unit for the IMU. For the spray test, the device was sprayed with a standard spray bottle from roughly 60cm (~2ft) away with the bottle set to deliver a wide spread of mist. This test also showed no noticeable amounts of water inside the housing unit, determining that the device was safe from any traces of liquid that may be found in gym settings.

Accuracy Testing Results

After calibration, the device was taken to the Nicholas Recreational Center on UW campus to perform kinetic accuracy testing on a real barbell during bench press exercises. Three trials were performed, with two trials successfully outputting data. The second trial had issues with data collection and MATLAB's ability to read the data, so no graphical analysis was

performed for that trial. Both the first and third trials resulted in successful data retrieval, with the first test yielding the most accurate results.

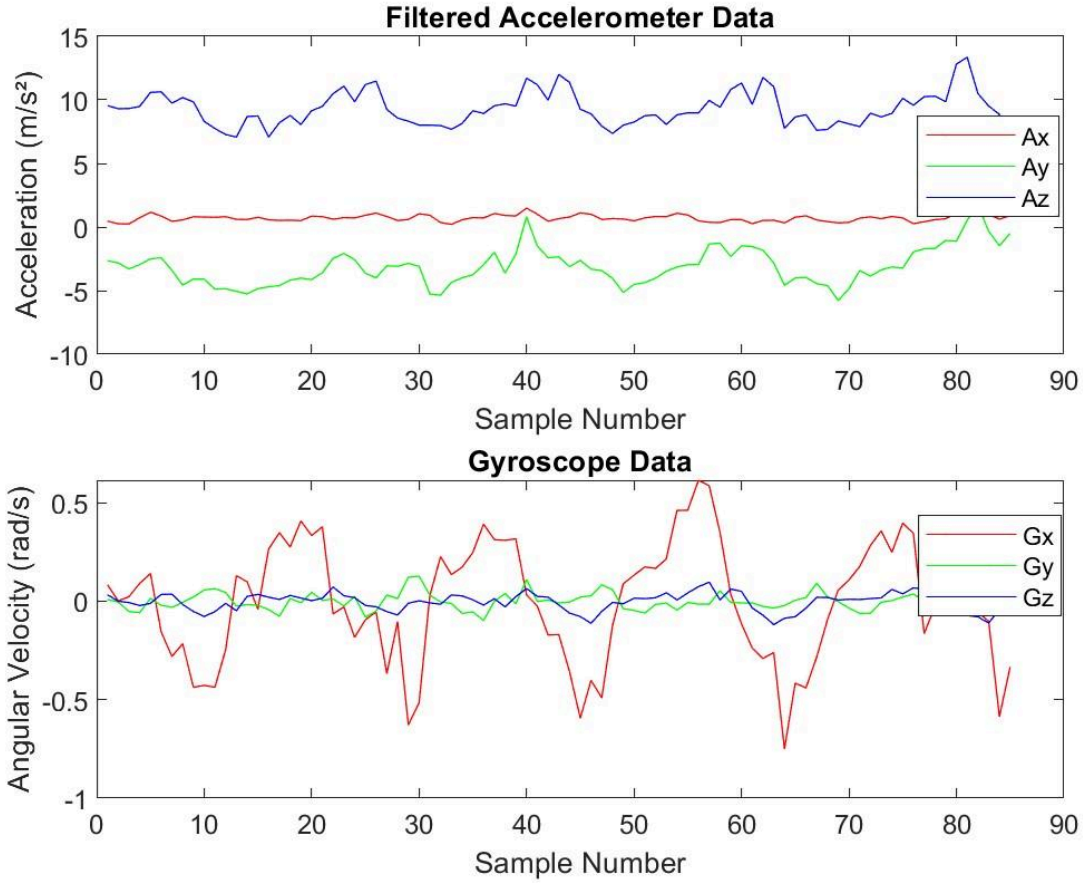

Figure 9: Subplot of Acceleration and Gyroscope Data during Testing Trial 1 - 5 repetitions.

The acceleration and gyroscope data supported our hypothesis by observation that only the Y and Z planes should have changes in direction synchronized with each other. For reference, the Z plane is perpendicular to the ground, the direction of gravity, which is also why the Z acceleration fluctuates around 9.8m/s^2. The Y plane is along the direction of the lifter from their head to their feet, with the positive direction being towards the lifter's head. For unknown reasons, the Y acceleration fluctuated around -4m/s^2 which may be a calibration error, offset error during calibration, or sensor quality error, but the behavior of the data matched our expectations. Likewise, the X plane, which crosses the lifter horizontally, stayed around zero, which matched our observations that the bar did not move in that direction at all.

For the gyroscope data, both the Y and Z planes exhibited no angular motion, supporting that the test lifter had very optimal form during this trial. The reasons that explain the changes in X angular data are due to wrist flexion-extension when performing each repetition. This was not considered initially, but after evaluation it makes sense this should occur during optimal-form lifting on bench press. The lifter during this trial also maintained a steady rate of ± 0.5 rad/s indicating that their pace between repetitions was controlled for the entire duration of the set.

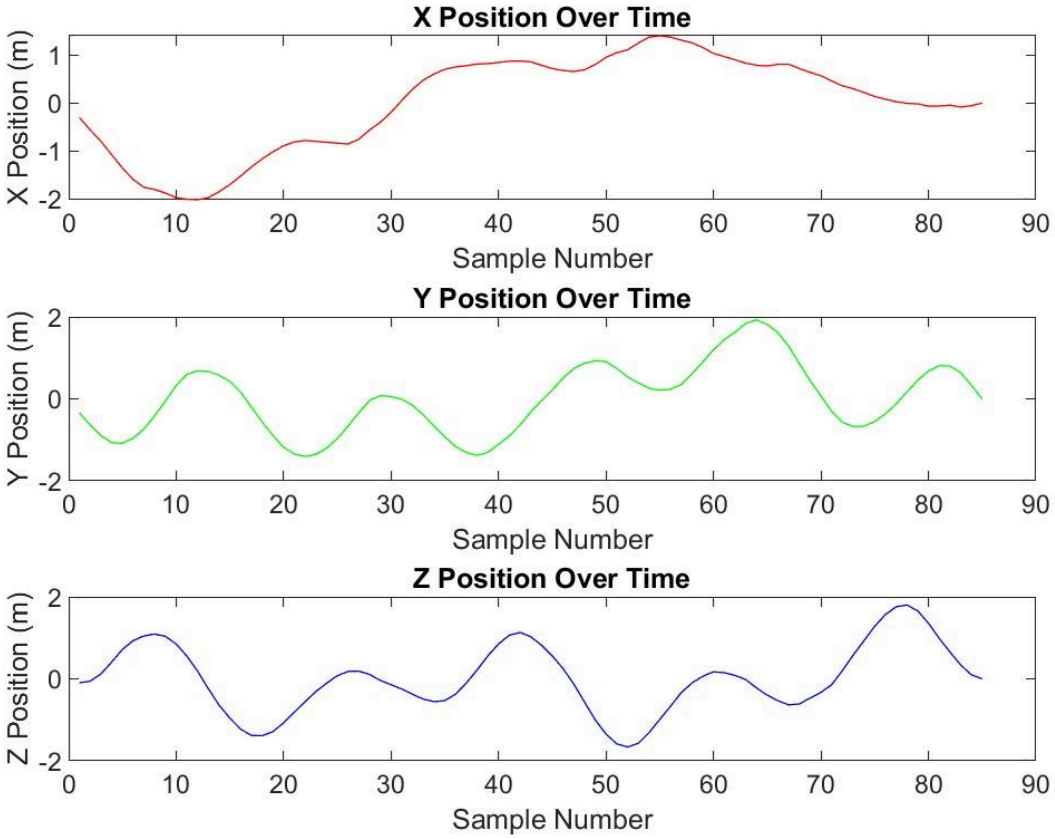

Figure 10: Subplot of Position over Time in all three Dimensions for Trial 1- 5 repetitions.

The position data nicely complemented the acceleration data, showing that both the Y and Z still moved synchronized with each other. This is displayed through the peaks and troughs happening at nearly identical intervals with one another. One other aspect that this subplot highlights is the nuisance of drift and noise issues with IMU sensors, particularly one's at the quality of the one chosen for this project[16]. As previously mentioned, the X plane displayed no

motion by observation which was also supported by the acceleration subplot with the exception of minor noise fluctuations. However, when you integrate data twice to get position, these noise issues get amplified, and despite many attempts to normalize and detrend the data, the plot still displays a 3 meter range of position in the X plane, which is nowhere near where it actually was. This is also true for the Y and Z planes, but the trends in the data did prove to be accurate for those two subplots. For unknown reasons, this was not the case in the X plane.

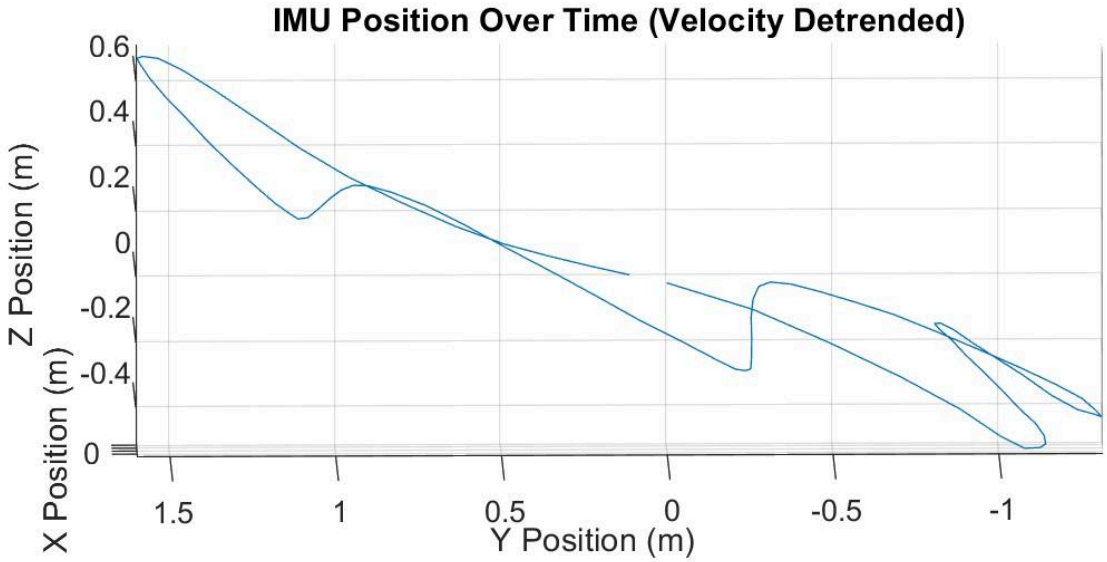

Figure 11: 3D Plot of Position over Time for Trial 1 - 1 repetition.

A 3D plot was generated from MATLAB to display the true path of the device in 3D space. The general shape is comparable to what was observed in the trial. However, the device was rotated from a neutral plane setup when the lifter brought the bar off of the rack, resulting in the data being rotated respectively. The sharp changes in direction are due to the noise as mentioned prior, and the units are still not accurate, but the general path taken and recorded does fit the predictions that were made. While these results were not perfect, they did demonstrate that the device can accurately track changes in motion in 3D space. This means that the potential for the device is well beyond our results; all errors that were encountered do have methods of mitigation, such as higher quality sensors and changes in the coding to fix offsets and units.

Discussion

Future Work

The initial expectations for the device were that it would have the capabilities to have "optimal form data" embedded in it so that any recording after that would be compared to analyze deviations from good form. Due to issues with code debugging and attempts to mitigate the noise/drift issues in the data received, the team's timeline was set back from completing this portion of the device. One additional limitation to this was the small storage space in the EEPROM of the Arduino Nano. Comparing for deviation is entirely possible, and the team did establish code to do this, but with limited time to do testing and evaluation on it there was no way to implement this aspect of the design within the given timeframe. A future design would focus on deviation and additionally incorporate an SD card or other storage device to remove data storage limitations.

Another key concept for this device would be to give post-repetition feedback via a mobile device app. Part of this would include switching out the current IMU for one with bluetooth capabilities and taking all of the MATLAB code and writing equivalent code in Arduino so that the board is programmed to do all of the data analysis without the need of a serial connection. This could also be done in the app software, and may even be easier so long as the app could effectively communicate with Arduino to access the dataset. Switching out the IMU would also mitigate the noise complications exhibited in the MPU-6050, and use of an app would also allow the device to start at the command of a digital touch with both sensors synchronized as opposed to pressing the two individual buttons simultaneously.

Lastly, making the device as small as possible is always a goal, so incorporating a smaller breadboard would optimize the space needed for the circuitry, thus minimizing the needed size for the housing unit. There are also alternatives to power sources, such as smaller batteries with equal performances as the three AA batteries that were used for this device.

Conclusion

Weightlifting can be a critical component of a healthy lifestyle. However, without proper form, the probability of injuries occurring during a workout can increase dramatically. Devices like the team's barbell weight clips which track motion of objects across 3 axes are on the market, however none are specifically geared to giving post feedback by comparing to an ideal barbell path for activities such as bench press.

While a lot of other devices have the means to be applied to prevent weightlifting injuries, no one has attempted to meet this demand the team has set out to fulfill. This project set out with the aim to combine in a novel way IMU's, coding, and 3D printing which is cost-effective, accurate, precise, and both tracks barbell path, and has the capacity to be stored and compared to a line of best fit for the interpretation of the user to fix errors in form. The device for this project includes a weight clipped with 3D printed compartments attached to it using command strips housing both breadboards and MPU 6050's. When a rep is to be performed a button is pressed on the side of the 3D printed compartment activating the IMU's to begin recording data. This data can then be read and processed by code written in C++ on Arduino by hooking up the IMU to the computer using a USB cord and then displayed using more code on MATLAB as a 3D graph for the user to see. This device explores the need to be user friendly, by the simple push of a button the process begins for everything one needs to fix one's form on bench press in comparison to competing products.

This project was a crucial step in the right direction addressing a whole new field and way of approaching fitness from a new perspective while also raising awareness for the dangers of lifting with improper form. By combining unique code and a modification to a common gym item, this device helps stop injuries by attacking the root of the issue -bad form- and doing it at a cheap and affordable price compared to the rest of the market. This may end up opening up new gateways and ideas in a field of technology which has barely scratched the service in terms of scientific innovation which could be used by physical therapists, doctors, hobbyist gym goers, bodybuilders, powerlifters, etc…. This demonstrates the uniqueness and potential impact this device could have on the ecosystem of the gym.

References

[1] “New National Study Examines Weight Training-Related Injuries.” Accessed: Oct. 09, 2024. [Online]. Available: https://www.nationwidechildrens.org/newsroom/news-releases/2010/03/new-national-study-examines-weight-training-related-injuries

[2] L. Noteboom, I. Belli, M. J. M. Hoozemans, A. Seth, H. E. J. Veeger, and F. C. T. Van Der Helm, “Effects of bench press technique variations on musculoskeletal shoulder loads and potential injury risk,” *Front. Physiol.*, vol. 15, p. 1393235, Jun. 2024, doi: 10.3389/fphys.2024.1393235.

[3] “Shoulder Impingement & Resistance Training: What Powerlifters, Weightlifters and Barbell Strength Athletes Need to Know to Understand Shoulder Injuries.” Accessed: Oct. 09, 2024. [Online]. Available: https://www.progressiverehabandstrength.com/articles/shoulder-impingement-resistance-training

[4] “WL Analysis - bar path tracker - Apps on Google Play.” Accessed: Oct. 03, 2024. [Online]. Available: https://play.google.com/store/apps/details?id=com.karolsmolak.wlanalysis&hl=en_US

[5] C. Ferri Marini *et al.*, “Barbell load distribution and lifting velocity affect bench press exercise volume and perceived exertion,” *PLOS ONE*, vol. 17, no. 12, p. e0278909, Dec. 2022, doi: 10.1371/journal.pone.0278909.

[6] “Bar Sensei.” Accessed: Sep. 19, 2024. [Online]. Available: http://files.assess2perform.com/barsensei.html

[7] flexadmin, “FLEXStronger | Velocity Based Training Made Simple,” FlexStronger. Accessed: Sep. 19, 2024. [Online]. Available: https://www.flexstronger.com/

[8] D. Jones, “Locking barbell collar,” USD780860S1, Mar. 07, 2017 Accessed: Dec. 08, 2024. [Online]. Available: https://patents.google.com/patent/USD780860S1/en

[9] A. D. O’Brien, “Weightlifting Barbell,” US20150038302A1, Feb. 05, 2015 Accessed: Dec. 08, 2024. [Online]. Available: https://patents.google.com/patent/US20150038302A1/en

[10] “MPU-6050,” TDK InvenSense. Accessed: Oct. 03, 2024. [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[11] RudraNarayanG, “MPU 6050 Gyro,Accelerometer Communication With Arduino (Atmega328p),” Instructables. Accessed: Oct. 03, 2024. [Online]. Available: https://www.instructables.com/Accelerometer-MPU-6050-Communication-With-AVR-MCU/

[12] “PLA vs Tough PLA.” Accessed: Oct. 09, 2024. [Online]. Available: https://support.bcn3d.com/knowledge/pla-vs-tough-pla-bcn3d

[13] Dejan, “Arduino and MPU6050 Accelerometer and Gyroscope Tutorial,” How To Mechatronics. Accessed: Oct. 03, 2024. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/

[14] “Arduino Guide for MPU-6050 Accelerometer and Gyroscope | Random Nerd Tutorials.” Accessed: Oct. 07, 2024. [Online]. Available: https://randomnerdtutorials.com/arduino-mpu-6050-accelerometer-gyroscope/

[15] “AA Battery Voltage Chart (Important Battery Metrics),” ShopSolar.com. Accessed: Nov. 25, 2024. [Online]. Available:

https://shopsolarkits.com/blogs/learning-center/aa-battery-voltage-chart
[16] L. Llamas, “Measure inclination with IMU, Arduino and complementary filter,” Luis Llamas. Accessed: Dec. 08, 2024. [Online]. Available: https://www.luisllamas.es/en/measure-imu-tilt-arduino-complementary-filter/
[17] “Weight training: Do’s and don’ts of proper technique - Mayo Clinic.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/weight-training/art-20045842
[18] M. Staniszewski, J. Tkaczyk, A. Kęska, P. Zybko, and A. Mróz, “Effect of rest duration between sets on fatigue and recovery after short intense plyometric exercise,” *Sci. Rep.*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-66146-2.
[19] C. for D. and R. Health, “Examples of Software Functions That Are NOT Medical Devices,” *FDA*, Sep. 2022, Accessed: Sep. 19, 2024. [Online]. Available: https://www.fda.gov/medical-devices/device-software-functions-including-mobile-medical-applications/examples-software-functions-are-not-medical-devices
[20] “CFR - Code of Federal Regulations Title 21.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?fr=890.1375
[21] C. for D. and R. Health, “General Wellness: Policy for Low Risk Devices.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-wellness-policy-low-risk-devices
[22] “InertiaCube® 4 | InterSense.” Accessed: Sep. 19, 2024. [Online]. Available: https://www.intersense.com/inertiacube4

Appendices

Appendix A: Product Design Specifications

Product Design Specifications: Preventing Weightlifting Injuries by Barbell Modifications

---

September 19th, 2024

Biomedical Engineering 200/300: Biomedical Engineering Fundamentals & Design

Client: Mr. Robert Gold

Advisor: Prof. William Murphy

Team Members:

Jackson Jarrett jrjarrett2@wisc.edu (Leader and BWIG)

Kai McClellan kamcclellan@wisc.edu (Communicator)

Gavin Gruber gtgruber@wisc.edu (BPAG)

Luke Schmeling lascmeling@wisc.edu (BSAC)

Function:

Thousands of weightlifting injuries occur every year. These injuries are often caused by an uneven distribution of load on the barbell, leading to the weight lifter favoring one arm over the other. Incorrect loading while weight training can lead to sprains, strains, fractures and other painful injuries [17]. The team has been tasked with designing a biomedical device that can prevent weight lifting injuries by targeting, identifying, and correcting improper form.

Client Requirements:

I. Quantify strain on specific muscles when used in complex weightlifting movements such as the barbell bench press

II. Utilize motion technology and camera tracking to develop a service that can track proper weightlifting form

III. Explore emerging software opportunities while building upon previous years progression

Design Requirements:

1. Physical and Operational Characteristics:

a. Performance Requirements:

I. Modified weightlifting clips will house an Arduino Nano and MPU6050. This technology will be housed within the weightlifting clips, allowing for tracking of the barbell path on every repetition of the bench press.

II. Both devices will have an MPU6050 to collect data in the form of angular velocity and acceleration. MATLAB will derive this data into displacement in meters along the X, Y, and Z axis.

III. Collecting this data will provide the user with a barbell path. A line of best fit will be set that correlates with proper bench press form. Derivation from this line of best fit will be recorded and displayed on an application that the user can then analyze.

b. Safety:

I. If the barbell path is different from the line of best fit in the X axis, the user will be alerted on the application. This would be a result of an uneven distribution of weight on the barbell, or due to the user's body preferences. In both cases, the coordination system will alert the user to alter their form in order to prevent injury.

II. If the barbell path is different from the line of best fit along a combination of the Y and Z axis, the user will be alerted on the application. This would be the result of an improper

barbell path. In this case, the coordinate system will be displayed, allowing the user to correct their barbell bench form in order to prevent injury.

III. Electronic compartments will have no exposed electronic parts to prevent interaction with water or other fluids. Proper cover will ensure no malfunction to injury prevention technology

c. Accuracy and Reliability:

I. The weightlifting clips will be functional as clips, holding the weight on each end of the barbell, preventing sliding or movement of the weights.

II. The MPU6050 will be able to track angular velocity and acceleration of the barbell from the weightlifting clips.

   A. Stationary data will be collected. This data will showcase the accuracy of the accelerometers, even with no movement of the barbell.

   B. Kinetic data will be collected. This data will show the accuracy of the accelerometers corresponding to movement of the barbell.

III. All barbell modifications will be established on the basis of repeatability, in which actions can be performed upon every lift by the user

d. Life in Service

I. The barbell modifications will have ample power in order to be active for 45-60 minutes, the average time of complete workout [18].

II. The technology will be cased and able to travel via car, airplane, boat, etc.

e. Shelf Life:

I. This device should be stored inside in a climate-controlled environment, around 20-25℃. It should have minimal exposure to, and not be stored in outside conditions such as rain and snow.

II. The device should have a shelf life of at least 10 years while in storage and not being used.

III. If there are any batteries they should be replaced whenever they are dead or every 5 years.

f. Operating Environment:

I. This device will primarily be used inside at a weightlifting gym but can be used outside as long as there is no rain, snow, or extreme conditions (temperatures below 5℃ or above 30℃ or extreme winds)

II. The device should also be dust and dirt-resistant and be strong enough to withstand small impacts like being dropped from 1 meter.

g. Ergonomics:

I. This device should be usable by all people of all heights and weights given that they can use a barbell to do the weightlifting movement.

II. This device should be usable for benching with a 20kg barbell.

III. This device should be able to be transported by one person easily, and can be brought between different gyms and locations.

h. Size:

I. This product should be able to clasp on a standard barbell

II. The technology housing should be minimized for functionality and aesthetics

i. Weight:

I. This product should be able to be transported by one person so all total components should be less than 10kg.

II. Any barbell attachments should be less than 5 kg, and if the barbell attachment weighs more than 1 kg that weight should be recorded on the outside of the product so the user can accurately see how much weight they have on the bar with the added weight of the attachment.

k. Materials:

I. The materials should be durable enough so they can be dropped from 1 m and light enough to meet the requirements given above.

II. The materials should be safe to touch and should not react with common cleaning chemicals (bleach, alcohol, ammonia), sweat, or water.

l. Aesthetics, Appearance, and Finish:

I. This product shouldn’t have any exposed electronics.

II. The product should not have any excessively sharp edges

2. Product Characteristics:

a. Quantity:

I. *Two of these devices should be needed per barbell.*

II. *Can be scaled to the amount of barbells that are in the gym .*

b. Target Product Cost:

I. *The cost of competitors products are in the range of $700-$900. We believe we can undercut this cost by triple and sell it for ideally $150-200. However with a budget of $300 the product will even at maximum cost be well under competition price range.*

3. Miscellaneous:

a. Standards and Specifications:

I. *Under section 520(o)(1)(B) of the FD&C Act, software that is intended "for maintaining or encouraging a healthy lifestyle and is unrelated to the diagnosis, cure, mitigation, prevention, or treatment of a disease or condition" is not a device under section 201(h) of the FD&C Act. This also indicates that it is generally excluded form CPSC's authority over consumer products under the Consumer Product Safety Act.[19]*

II. *If we use EMG technology, the device is considered to be a diagnostic electromygraph (Definition: [A] device intended for medical purposes, such as to monitor and display the bioelectric signals produced by muscles, to stimulate peripheral nerves, and to monitor and display the electrical activity produced by nerves, for the diagnosis and prognosis of neuromuscular disease). This is classified as a Class II device and is exempt from the premarket notification procedures in subpart E of part 807 of this chapter subject to § 890.9. This device is considered noninvasive for testing and therefore does not need premarket approval and is exempt from Section 510(k).[20]*

III. *If the device does not use EMG technology, it is then classified as a "Low Risk Device" (Class I)and does not need premarket approval, clearance by the FDA, and is exempt from Section 510(k).[21]*

b. Customer:

I. There are not many competing designs on the market, but there are many systems that function in a similar fashion. Thus, the intended goal is to combine the technology used in other fields with the knowledge and devices in the fitness industry to make a one of one device that fine-tunes a user's form.

II. Cost effectiveness will be a major concern with this device, as current products with much less capabilities are ranged from $100 and up which is subpar given the population of people who could benefit from this technology.

III. It would be desired to have the device give a digital readout indicating their deviation from what will be considered their optimal range of motion. Other beneficial readouts would be which muscles were under the most strain during the lift and recommendations to improve their performance.

c. Patient-Related Concerns:

I. Since it is classified as a "Low Risk Device" there are minimal concerns for usage of it. There will be no need for sterilization since the device is noninvasive and strictly a biomechanical analysis tool.

II. Accuracy and precision will be crucial for the success of this type of device. The complexity and usability will have to be mitigated as much as possible as well.

III. The device will have a large amount of data to observe, analyze, and compute. This means that it will require robust software and computational resources.

IV. Integrating this technology with EMG tools may increase the complexity of its analysis and must be considered when designing.

d. Competition:

I. "FLEX" , a barbell velocity tracker, provides real-time display, giving immediate feedback on every rep and set, watches velocity and power, and scrutinizes technique and refines movement patterns. Accuracy of these values is unknown.[7]

II. " Bar Sensei" is another barbell velocity tracker that measures your bar speed, displacement, and power output while performing a lift. It is a device that attaches directly to the barbell and takes measurements based on the displacement of the device itself.[6]

III. "InertiaCube® 4" , a 3 DOF sensor, uses MEMS technology to sense angular rate of rotation, gravity and earth magnetic field along three perpendicular axes. The angular rates are integrated to obtain the orientation (yaw, pitch, and roll) of the sensor. Gravitometer and compass measurements are used to prevent the accumulation of gyroscopic drift through advanced sensor fusion algorithms. This technology offers very

low latency, unlimited range, precise factory calibration, smooth, jitter-free tracking, in situ static & dynamic magnetic compensation algorithms.[22]

References:

[6] "Bar Sensei." Accessed: Sep. 19, 2024. [Online]. Available: http://files.assess2perform.com/barsensei.html

[7] flexadmin, "FLEXStronger | Velocity Based Training Made Simple," FlexStronger. Accessed: Sep. 19, 2024. [Online]. Available: https://www.flexstronger.com/

[15] "Weight training: Do's and don'ts of proper technique - Mayo Clinic." Accessed: Sep. 19, 2024. [Online]. Available: https://www.mayoclinic.org/healthy-lifestyle/fitness/in-depth/weight-training/art-20045842[15]

[16] M. Staniszewski, J. Tkaczyk, A. Kęska, P. Zybko, and A. Mróz, "Effect of rest duration between sets on fatigue and recovery after short intense plyometric exercise," *Sci. Rep.*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-66146-2.

[17] C. for D. and R. Health, "Examples of Software Functions That Are NOT Medical Devices," *FDA*, Sep. 2022, Accessed: Sep. 19, 2024. [Online]. Available: https://www.fda.gov/medical-devices/device-software-functions-including-mobile-medical-applications/examples-software-functions-are-not-medical-devices

[18] "CFR - Code of Federal Regulations Title 21." Accessed: Sep. 19, 2024. [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?fr=890.1375

[19] C. for D. and R. Health, "General Wellness: Policy for Low Risk Devices." Accessed: Sep. 19, 2024. [Online]. Available: https://www.fda.gov/regulatory-information/search-fda-guidance-documents/general-wellness-policy-low-risk-devices

[20] "InertiaCube® 4 | InterSense." Accessed: Sep. 19, 2024. [Online]. Available: https://www.intersense.com/inertiacube4

Appendix B: Outer Box Fabrication Protocol

Creating Sliding lid on OnShape

1. Open a new sketch and draw a square with the dimensions of 0.508 cm x 8.128 cm
2. On one end of the box draw the latching mechanism with the measurements shown in figure 1

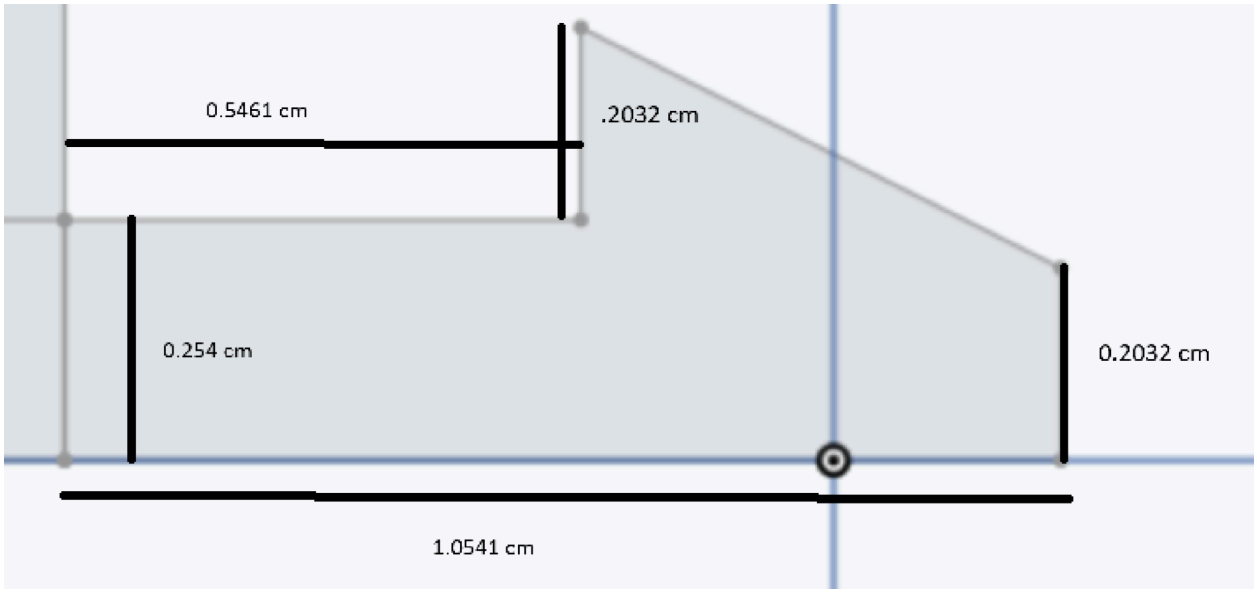

Figure 12. Sketch in Onshape for the latching mechanism on the sliding lid. Black lines show where measurements are.

3. Extrude the original square with the dimensions of 0.508 cm x 8.128 cm, 1.905 cm in each direction
4. Extrude the latching mechanism for the lid 0.635 in each direction.
5. Create another square in the sketch that covers the bottom half of the first square drawn, it should have the dimensions of 0.254 cm x 4.064 cm.
6. Extrude this new square 2.159 cm in each direction, it should now look like figure 2.

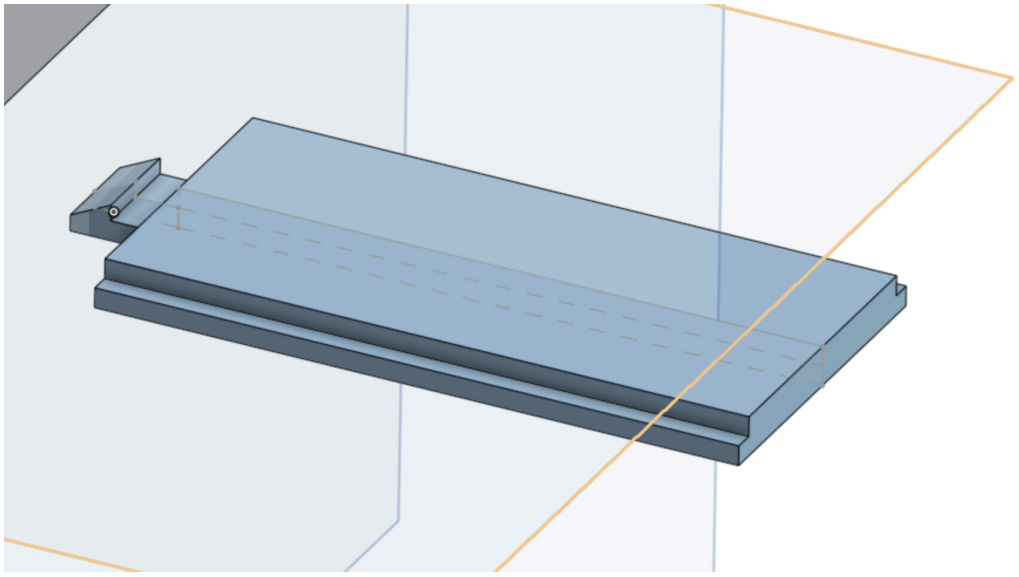

Figure 13. Shows how the lid should look like in Onshape.

Creating the Outer Box on Onshape.

1. Create a new sketch and draw a square with the dimensions of 4.826 cm x 3.556 cm.
2. Create the lines shown below in figure 3, inside of the square you just created. The measurements are the same on both sides of the square.

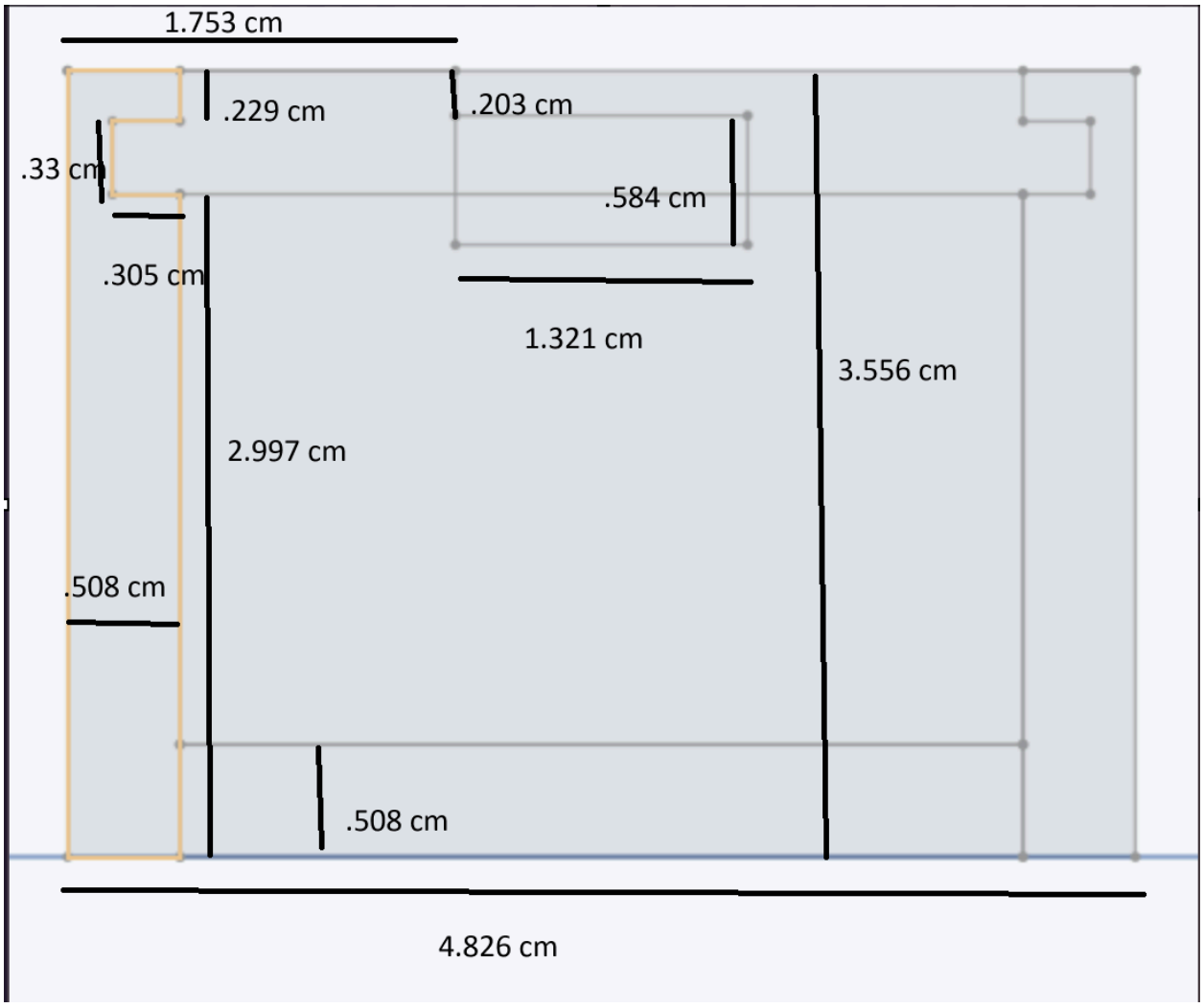

Figure 14. Sketch on onshape that shows all of the measurements needed to complete the 3D model for the outer box. The box is symmetrical down the middle so the measurements are the same on both sides.

3. Finish the sketch and select the shape that is highlighted in figure 3 along with the mirror shape on the other side of the square, also select the rectangle on the very bottom of the box.
4. Extrude these 3 shapes 8.636 cm in one direction.
5. Then select the very center shape and extrude that .508 cm but offset by 8.128 cm in the direction of your earlier extrudes.
6. Select the same center shape again along with the top shape that is along the very top of the square but not on the left and right sides.

7. Extrude these shapes .508 cm in the same direction as all earlier extrudes. This completes the 3D model of the box, it should now look like figure 4.

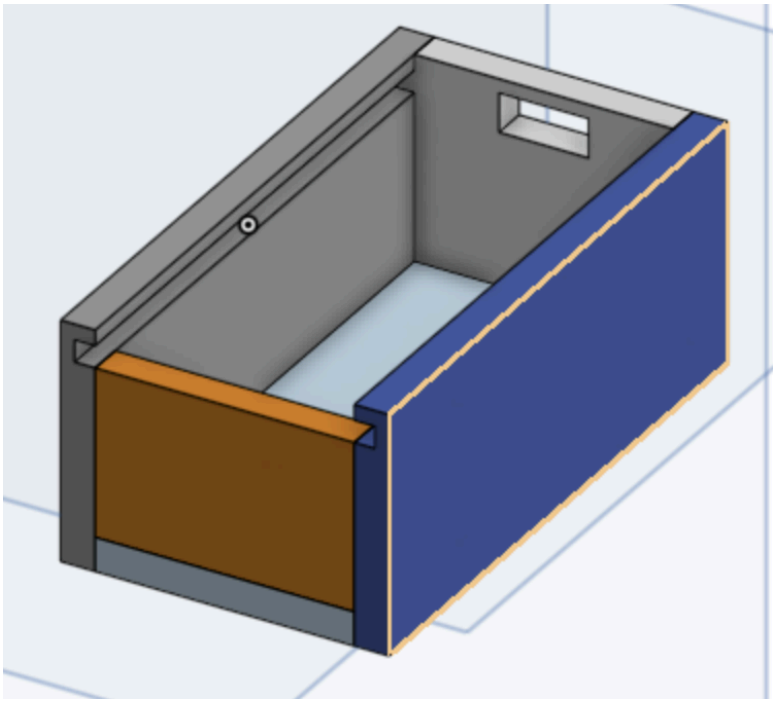

Figure 15. Completed 3D model of the outer box

8. 3D print all the parts using pla, then clean up any supports used and the box and lid are complete.

Appendix C: Arduino Code

Calibration Sketch

```
/*
  IMU Zero

  Calibrates the MPU6050 to compensate for measurement deviations or
errors that may
  occur due to factors such as gyroscope and accelerometer offset.

  When running this example, the code adjusts the MPU6050 sensor's
acceleration (X,Y,Z)
  and rotation (gyroscope) axis offset values to ensure that the readings
are as accurate as
  possible when the device is at rest.

  To get better results consider:
  1. The MPU6050 module is working fine.
  2. Put the MPU6050 on a flat and horizontal surface, and leave it
operating for
  5-10 minutes so its temperature gets stabilized.
  4. Is in a location where the pull of gravity is 1g.

  During the execution it will generate a dozen outputs, showing that for
each of the 6
  desired offsets, it is:
  - First, try to find two estimates, one too low and one too high.
  - Closing in until the bracket can't be made smaller.

  The line just above the "done" (it will take a few minutes to get there)
describes the
  optimum offsets for the X acceleration, Y acceleration, Z acceleration,
X gyro, Y gyro,
  and Z gyro, respectively.

  Find the full MPU6050 library documentation here:
  https://github.com/ElectronicCats/mpu6050/wiki
*/
```

```
#include "I2Cdev.h"
#include "MPU6050.h"

/* Create the class for MPU6050. Default I2C address is 0x68 */
MPU6050 mpu;
// MPU6050 mpu(0x69); // <-- use for AD0 high

const int usDelay = 3150; // Delay in ms to hold the sampling at 200Hz
const int NFast = 5000; // Number of quick readings for averaging, the
higher the better
const int NSlow = 20000; // Number of slow readings for averaging, the
higher the better
const int LinesBetweenHeaders = 5;

const int iAx = 0;
const int iAy = 1;
const int iAz = 2;
const int iGx = 3;
const int iGy = 4;
const int iGz = 5;

int LowValue[6];
int HighValue[6];
int Smoothed[6];
int LowOffset[6];
int HighOffset[6];
int Target[6];
int LinesOut;
int N;
int i;

void setup() {
  Initialize(); //Initializate and calibrate the sensor

  for (i = iAx; i <= iGz; i++) {
    Target[i] = 0; // Fix for ZAccel
    HighOffset[i] = 0;
    LowOffset[i] = 0;
  }
```

```
  Target[iAz] = 16384; // Set the taget for Z axes
  SetAveraging(NFast); // Fast averaging
  PullBracketsOut();
  PullBracketsIn();
  Serial.println("-------------- DONE --------------");
}

void loop() {
  //Write your code here
}

/*Initializate function*/
void Initialize() {
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif
  Serial.begin(115200);
  // Init the module
  Serial.println("Initializing MPU...");
  mpu.initialize();
  Serial.println("MPU initializated");
  // Check module connection
  Serial.println("Testing device connections...");
  if(mpu.testConnection() ==  false){
    Serial.println("MPU6050 connection failed");
  while(true);
  }
  else{
    Serial.println("MPU6050 connection successful");
  }

  Serial.println("\nPID tuning Each Dot = 100 readings");
  /*
  PID tuning (actually PI) works like this: changing the offset in the
MPU6050 gives instant results,
  allowing us to use the Proportional and Integral parts of the PID to
find the ideal offsets.
```

```
  The Integral uses the error from the set point (which is zero) and adds
a fraction of this error to
  the integral value. Each reading reduces the error towards the desired
offset. The greater
  the error, the more we adjust the integral value.

  The Proportional part helps by filtering out noise from the integral
calculation. The Derivative part is
  not used due to noise and the sensor being stationary. With the noise
removed, the integral value stabilizes
  after about 600 readings. At the end of each set of 100 readings, the
integral value is used for the actual
  offsets, and the last proportional reading is ignored because it reacts
to any noise.
  */

Serial.println("\nXAccel\t\tYAccel\t\tZAccel\t\tXGyro\t\tYGyro\t\tZGyro");

  manualCalibration();


  mpu.CalibrateGyro(6);
  Serial.println("\n600 Readings");
  mpu.PrintActiveOffsets();

  mpu.CalibrateGyro(1);
  Serial.println("700 Total Readings");
  mpu.PrintActiveOffsets();

  mpu.CalibrateGyro(1);
  Serial.println("800 Total Readings");
  mpu.PrintActiveOffsets();

  mpu.CalibrateGyro(1);
  Serial.println("900 Total Readings");
  mpu.PrintActiveOffsets();

  mpu.CalibrateGyro(1);
  Serial.println("1000 Total Readings");
  mpu.PrintActiveOffsets();
```

```
  Serial.println("\nAny of the above offsets will work nicely \n\nProving
the PID with other method:");
}

void SetAveraging(int NewN) {
  N = NewN;
  Serial.print("\nAveraging ");
  Serial.print(N);
  Serial.println(" readings each time");
}

void PullBracketsOut() {
  boolean Done = false;
  int NextLowOffset[6];
  int NextHighOffset[6];

  Serial.println("Expanding:");
  ForceHeader();

  while (!Done) {
    Done = true;
    SetOffsets(LowOffset); //Set low offsets
    GetSmoothed();
    for (i = 0; i <= 5; i++) { // Get low values
      LowValue[i] = Smoothed[i];
      if (LowValue[i] >= Target[i]) {
        Done = false;
        NextLowOffset[i] = LowOffset[i] - 1000;
      }
      else {
        NextLowOffset[i] = LowOffset[i];
      }
    }
    SetOffsets(HighOffset);
    GetSmoothed();
    for (i = 0; i <= 5; i++) { // Get high values
      HighValue[i] = Smoothed[i];
      if (HighValue[i] <= Target[i]) {
        Done = false;
        NextHighOffset[i] = HighOffset[i] + 1000;
```

```
      }
      else {
        NextHighOffset[i] = HighOffset[i];
      }
    }
    ShowProgress();
    for (int i = 0; i <= 5; i++) {
      LowOffset[i] = NextLowOffset[i];
      HighOffset[i] = NextHighOffset[i];
    }
  }
}

void PullBracketsIn() {
  boolean AllBracketsNarrow;
  boolean StillWorking;
  int NewOffset[6];

  Serial.println("\nClosing in:");
  AllBracketsNarrow = false;
  ForceHeader();
  StillWorking = true;
  while (StillWorking) {
    StillWorking = false;
    if (AllBracketsNarrow && (N == NFast)) {
      SetAveraging(NSlow);
    }
    else {
      AllBracketsNarrow = true;
    }
    for (int i = 0; i <= 5; i++) {
      if (HighOffset[i] <= (LowOffset[i] + 1)) {
        NewOffset[i] = LowOffset[i];
      }
      else { // Binary search
        StillWorking = true;
        NewOffset[i] = (LowOffset[i] + HighOffset[i]) / 2;
        if (HighOffset[i] > (LowOffset[i] + 10)) {
          AllBracketsNarrow = false;
        }
```

```
      }
    }
    SetOffsets(NewOffset);
    GetSmoothed();
    for (i = 0; i <= 5; i++) { // Closing in
      if (Smoothed[i] > Target[i]) { // Use lower half
        HighOffset[i] = NewOffset[i];
        HighValue[i] = Smoothed[i];
      }
      else { // Use upper half
        LowOffset[i] = NewOffset[i];
        LowValue[i] = Smoothed[i];
      }
    }
    ShowProgress();
  }
}

void ForceHeader() {
  LinesOut = 99;
}

/*Function to smooth the read values*/
void GetSmoothed() {
  int16_t RawValue[6];
  long Sums[6];
  for (i = 0; i <= 5; i++) {
    Sums[i] = 0;
  }

/* Get Sums*/
  for (i = 1; i <= N; i++) {
    mpu.getMotion6( & RawValue[iAx], & RawValue[iAy], & RawValue[iAz], &
RawValue[iGx], & RawValue[iGy], & RawValue[iGz]);
    delayMicroseconds(usDelay);
    for (int j = 0; j <= 5; j++){
      Sums[j] = Sums[j] + RawValue[j];
    }
  }
  for (i = 0; i <= 5; i++) {
```

```
    Smoothed[i] = (Sums[i] + N / 2) / N;
  }
}

/*Function for configure the oba=tained offsets*/
void SetOffsets(int TheOffsets[6]) {
  mpu.setXAccelOffset(TheOffsets[iAx]);
  mpu.setYAccelOffset(TheOffsets[iAy]);
  mpu.setZAccelOffset(TheOffsets[iAz]);
  mpu.setXGyroOffset(TheOffsets[iGx]);
  mpu.setYGyroOffset(TheOffsets[iGy]);
  mpu.setZGyroOffset(TheOffsets[iGz]);
}

/*Print the progress of the reading averages, add formatting for better
visualization*/
void ShowProgress() {
 /*Header*/
  if (LinesOut >= LinesBetweenHeaders) {

Serial.println("\t\tXAccel\t\t\tYAccel\t\t\t\tZAccel\t\t\tXGyro\t\t\tYGyro
\t\t\tZGyro");
    LinesOut = 0;
  }
  Serial.print(' ');
  for (i = 0; i <= 5; i++) {
    Serial.print('[');
    Serial.print(LowOffset[i]),
    Serial.print(',');
    Serial.print(HighOffset[i]);
    Serial.print("] --> [");
    Serial.print(LowValue[i]);
    Serial.print(',');
    Serial.print(HighValue[i]);
    if (i == 5) {
      Serial.println("]");
    }
    else {
      Serial.print("]\t");
    }
```

```
  }
  LinesOut++;
}

void manualCalibration() {
  const int numReadings = 100; // Number of readings for averaging
  int16_t ax, ay, az;
  long sumAx = 0, sumAy = 0, sumAz = 0;

  Serial.println("Starting manual calibration...");

  for (int i = 0; i < numReadings; i++) {
    mpu.getAcceleration(&ax, &ay, &az);
    sumAx += ax;
    sumAy += ay;
    sumAz += az;
    delay(50); // Delay between readings
  }

  // Calculate averages
  int16_t avgAx = sumAx / numReadings;
  int16_t avgAy = sumAy / numReadings;
  int16_t avgAz = sumAz / numReadings;

  Serial.print("Average Accel: ");
  Serial.print(avgAx); Serial.print("\t");
  Serial.print(avgAy); Serial.print("\t");
  Serial.println(avgAz);
}
```

Data Collection with Calibrated Offsets Sketch

```
#include <EEPROM.h>
#include "I2Cdev.h"
#include "MPU6050.h"

MPU6050 mpu;

/* OUTPUT FORMAT DEFINITION */
#define OUTPUT_READABLE_ACCELGYRO

// Pin definitions
#define BUTTON_PIN A3  // Button pin

int16_t ax, ay, az;
int16_t gx, gy, gz;
int16_t prev_ax = 0, prev_ay = 0, prev_az = 0;  // Previous accelerometer
readings
int16_t prev_gx = 0, prev_gy = 0, prev_gz = 0;  // Previous gyroscope
readings
bool blinkState;
int eepromAddress = 2;  // Start storing data from address 2 to avoid
overwriting flags
bool eepromFull = false;  // Flag to indicate EEPROM is full
#define FLAG_ADDRESS 0  // EEPROM address used to store the cleared flag
#define EEPROM_FULL_FLAG_ADDRESS 1  // EEPROM address used to store the
full flag
#define EEPROM_ADDRESS_POINTER 2  // EEPROM address used to store the
current address pointer

bool buttonPressed = false; // Button press detection
bool lastButtonState = HIGH; // Previous button state

// Offset values for accelerometer and gyroscope
int16_t XAccel_Offset = -1446;
int16_t YAccel_Offset = 91;
int16_t ZAccel_Offset = 1456;
int16_t XGyro_Offset = 43;
int16_t YGyro_Offset = 22;
int16_t ZGyro_Offset = 32;
```

```
// Threshold for logging significant changes
int16_t accelThresholdX = 150;  // Adjust as necessary for accelerometer
sensitivity on x-axis
int16_t accelThresholdY = 150;  // Adjust as necessary for accelerometer
sensitivity on y-axis
int16_t accelThresholdZ = 200;  // Adjust as necessary for accelerometer
sensitivity on z-axis

int16_t gyroThresholdX = 30;    // Adjust as necessary for gyroscope
sensitivity on x-axis
int16_t gyroThresholdY = 30;    // Adjust as necessary for gyroscope
sensitivity on y-axis
int16_t gyroThresholdZ = 30;    // Adjust as necessary for gyroscope
sensitivity on z-axis

void setup() {
    Wire.begin();
    Serial.begin(115200);

    pinMode(BUTTON_PIN, INPUT_PULLUP); // Configure button pin as input
with pull-up resistor

    Serial.println("Initializing MPU...");
    mpu.initialize();
    Serial.println("Testing MPU6050 connection...");
    if (!mpu.testConnection()) {
        Serial.println("MPU6050 connection failed");
        while (true);  // Halt the program if connection fails
    } else {
        Serial.println("MPU6050 connection successful");
    }

    // Set sensor offsets
    mpu.setXAccelOffset(XAccel_Offset);
    mpu.setYAccelOffset(YAccel_Offset);
    mpu.setZAccelOffset(ZAccel_Offset);
    mpu.setXGyroOffset(XGyro_Offset);
    mpu.setYGyroOffset(YGyro_Offset);
    mpu.setZGyroOffset(ZGyro_Offset);
```

```
    pinMode(LED_BUILTIN, OUTPUT);

    // Retrieve the saved EEPROM address pointer
    EEPROM.get(EEPROM_ADDRESS_POINTER, eepromAddress);

    // Check if EEPROM was previously full
    if (EEPROM.read(EEPROM_FULL_FLAG_ADDRESS) == 1) {
        eepromFull = true;  // Set the flag to indicate EEPROM was already
full
        Serial.println("EEPROM is full. No new data collection will
occur.");
    } else if (EEPROM.read(FLAG_ADDRESS) != 1) {
        // Only clear EEPROM if it's the very first time
        clearEEPROM();
        EEPROM.write(FLAG_ADDRESS, 1);  // Set the flag to indicate EEPROM
has been cleared
    } else {
        Serial.println("EEPROM already cleared, skipping.");
        Serial.print("Starting new data collection at address: ");
        Serial.println(eepromAddress);
    }
}

void clearEEPROM() {
    for (int i = 3; i < EEPROM.length(); i++) {
        EEPROM.write(i, 0);
    }
    eepromAddress = 3;  // Reset the storage address for the new test
    eepromFull = false; // Reset the full flag to start new collection
    EEPROM.write(EEPROM_FULL_FLAG_ADDRESS, 0);  // Clear the full flag in
EEPROM
    EEPROM.put(EEPROM_ADDRESS_POINTER, eepromAddress); // Store the new
starting address
    Serial.println("EEPROM cleared. Ready for new data collection.");
}

void storeDataInEEPROM(int16_t ax, int16_t ay, int16_t az, int16_t gx,
int16_t gy, int16_t gz) {
    if (eepromFull) {
```

```
        return;  // Stop function execution if EEPROM is full
    }

    // Check if the change in accelerometer or gyroscope readings exceeds
the threshold
    bool accelChanged = (abs(ax - prev_ax) > accelThresholdX) ||
                        (abs(ay - prev_ay) > accelThresholdY) ||
                        (abs(az - prev_az) > accelThresholdZ);

    bool gyroChanged = (abs(gx - prev_gx) > gyroThresholdX) ||
                       (abs(gy - prev_gy) > gyroThresholdY) ||
                       (abs(gz - prev_gz) > gyroThresholdZ);

    // Only store data if there is a significant change
    if (accelChanged || gyroChanged) {
        if (eepromAddress + 12 <= EEPROM.length()) {
            EEPROM.put(eepromAddress, ax); eepromAddress += 2;
            EEPROM.put(eepromAddress, ay); eepromAddress += 2;
            EEPROM.put(eepromAddress, az); eepromAddress += 2;
            EEPROM.put(eepromAddress, gx); eepromAddress += 2;
            EEPROM.put(eepromAddress, gy); eepromAddress += 2;
            EEPROM.put(eepromAddress, gz); eepromAddress += 2;

            // Save the updated address in EEPROM
            EEPROM.put(EEPROM_ADDRESS_POINTER, eepromAddress);

            // Update previous readings to the current readings
            prev_ax = ax; prev_ay = ay; prev_az = az;
            prev_gx = gx; prev_gy = gy; prev_gz = gz;
        } else {
            Serial.println("EEPROM full, data storage stopped.");
            eepromFull = true;  // Signal that the EEPROM is full
            EEPROM.write(EEPROM_FULL_FLAG_ADDRESS, 1);  // Write full flag
to EEPROM

            // Immediately read and send EEPROM data after it's full for
verification/debugging
            Serial.println("Reading and sending stored EEPROM data for
verification...");
            sendEEPROMData();
```

```
        }
    }
}

void sendEEPROMData() {
    Serial.println("Sending EEPROM data...");
    for (int i = 3; i < eepromAddress; i += 12) {
        int16_t ax, ay, az, gx, gy, gz;
        EEPROM.get(i, ax);
        EEPROM.get(i + 2, ay);
        EEPROM.get(i + 4, az);
        EEPROM.get(i + 6, gx);
        EEPROM.get(i + 8, gy);
        EEPROM.get(i + 10, gz);

        Serial.print("ax: "); Serial.print(ax);
        Serial.print(", ay: "); Serial.print(ay);
        Serial.print(", az: "); Serial.print(az);
        Serial.print(", gx: "); Serial.print(gx);
        Serial.print(", gy: "); Serial.print(gy);
        Serial.print(", gz: "); Serial.println(gz);

        delay(50); // Small delay to allow Serial communication stability
    }
    Serial.println("End of EEPROM data");
}

void handleButtonPress() {
    bool currentButtonState = digitalRead(BUTTON_PIN);
    if (lastButtonState == HIGH && currentButtonState == LOW) {
        buttonPressed = true;
    }
    lastButtonState = currentButtonState;

    if (buttonPressed) {
        buttonPressed = false;
        Serial.println("Button pressed. Clearing EEPROM...");
        EEPROM.write(FLAG_ADDRESS, 0);  // Clear the flag before clearing
the EEPROM
        clearEEPROM();
```

```
    }
}

void loop() {
    // Data Collection Mode
    if (!eepromFull) {
        mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

        // Only store data in EEPROM if there is significant movement
        storeDataInEEPROM(ax, ay, az, gx, gy, gz);

        // Blink LED to indicate activity
        blinkState = !blinkState;
        digitalWrite(LED_BUILTIN, blinkState);

        delay(125); // Adjust delay to control data sampling rate
    }

    // Check for button press to clear EEPROM
    handleButtonPress();

    // Command Handling Mode: Wait for 'R' or 'C' command
    if (Serial.available() > 0) {
        char command = Serial.read();
        if (command == 'R' && eepromFull) {  // Only respond to 'R' if
EEPROM is full
            Serial.println("Command 'R' received. Preparing to send EEPROM
data...");
            sendEEPROMData();
        } else if (command == 'C') {
            Serial.println("Command 'C' received. Clearing EEPROM...");
            EEPROM.write(FLAG_ADDRESS, 0);  // Clear the flag before
clearing the EEPROM
            clearEEPROM();
        } else {
            Serial.print("Unknown command received: ");
            Serial.println(command);
        }
    }
}
```

Appendix D: Testing Protocols

Battery Life Testing Protocol

The average battery life of 3 AA batteries powering 5 volts of power is in the 3-5 hour range. To test this, we will:

1. Put brand new batteries into the battery pack, and power the device
2. We will leave the battery pack on, and every 30 minutes we will check to make sure that the batteries are still powering the device for the first 3 hours
    a. After the first 3 hours, we will check the device more frequently to ensure that we can get a more accurate value of the battery life of the 3 AA batteries
3. Once we reach a point in which the device is no longer, powered, we will note the approximated battery life of the 3AA batteries

Water Protection Testing Protocol

Our prototype will not be completely waterproof. We understand that this is not ideal, but in order to ensure the success of our device, we will test the device with 50 mL of water being emitted from a spray bottle. We will also test the device with wet hands in order to replicate the effects of dealing with the device with sweaty hands during a workout.

Sweat Test

1. Test to ensure device is working and powered
2. Apply 50 mL of water to hands
3. Handle device
4. Dry off device
5. Test to ensure device is working and powered

Spray Test

1. Test to ensure device is working and powered
2. Fill spray bottle with 50 mL of water

3. Spray device evenly from approximately 2 feet away, using all 50 mL of water
4. Dry off device
5. Test to ensure device is working and powered

Accuracy Testing Protocol

Accuracy testing was always going to be the staple piece of our testing. Based on our goals and what the prototype looks to accomplish, it holds the most weight. We looked to test and record the displacement vs. time data of bench press repetitions over three trials:

1. Load the weight clips on to the barbell
2. With Luke on the bench, Kai and Jackson pressed the button to begin data collection.
3. Luke performed 5 reps of the bench press
4. Data Collection ends
5. MATLAB collects data from Arduino, and graphs position (displacement) over time

Appendix E: MatLab Data Code

```
clear;
clc;
% Setup Serial Connection
serialPort = 'COM6';  % Replace with your Arduino's COM port
baudRate = 115200;    % Ensure this matches the Arduino code
serialObj = serialport(serialPort, baudRate, "Timeout", 30); % Set a longer
timeout for reading
% Pause to allow Arduino to reset and be ready
pause(5);
% Clear initial buffer to get rid of any old messages
flush(serialObj);
% Request EEPROM Data
writeline(serialObj, 'R');  % Send 'R' command to Arduino
% Additional pause to give Arduino time to start sending the data
pause(2);
% Initialize arrays to store the data
ax = [];
ay = [];
az = [];
gx = [];
gy = [];
gz = [];
% Read and process EEPROM data
disp("Receiving EEPROM data...");
while true
   try
       % Attempt to read a line from the Arduino
       dataLine = readline(serialObj);

       % Display the line for debugging purposes
       disp(dataLine);

       % Check if the dataLine is not empty and contains the end marker
       if ~isempty(dataLine) && contains(dataLine, "End of EEPROM data")
           break;  % Stop reading when the end marker is found
       end
```

```
        % Extract the numeric data from the line
        tokens = regexp(dataLine, 'ax: ([\-\d]+), ay: ([\-\d]+), az: ([\-\d]+),
gx: ([\-\d]+), gy: ([\-\d]+), gz: ([\-\d]+)', 'tokens');
        if ~isempty(tokens)
            values = str2double(tokens{1});
            ax(end + 1) = values(1);
            ay(end + 1) = values(2);
            az(end + 1) = values(3);
            gx(end + 1) = values(4);
            gy(end + 1) = values(5);
            gz(end + 1) = values(6);
        end
    catch ME
        % If an error occurs (e.g., timeout), display a warning and break the
loop
        warning('An error occurred while reading from Serial: %s', ME.message);
        break;
    end
end
% Close the serial connection
clear serialObj;
% Check the number of readings
numReadings = length(ax);
if numReadings < 2
    error("Insufficient data points collected for accurate analysis. Make sure
data collection was successful.");
end
% Convert raw accelerometer data to m/s² (assuming sensitivity scale factor is
16384 for +/- 2g)
accelScaleFactor = 16384; % +/- 2g setting for MPU6050
ax = (ax / accelScaleFactor) * 9.81; % Convert to m/s²
ay = (ay / accelScaleFactor) * 9.81; % Convert to m/s²
az = (az / accelScaleFactor) * 9.81; % Convert to m/s²
% Convert raw gyroscope data to radians per second (assuming sensitivity scale
factor is 131 for +/- 250 deg/s)
gyroScaleFactor = 131; % +/- 250 deg/s setting for MPU6050
deg2rad_factor = pi / 180;
```

```
gx = (gx / gyroScaleFactor) * deg2rad_factor; % Convert to rad/s
gy = (gy / gyroScaleFactor) * deg2rad_factor; % Convert to rad/s
gz = (gz / gyroScaleFactor) * deg2rad_factor; % Convert to rad/s
% Filter the accelerometer data to reduce noise
cutoff = 1; % Cutoff frequency in Hz for low-pass filter (adjust as needed)
[b, a] = butter(4, cutoff / (2.5 / 2), 'low');
ax_filtered = filtfilt(b, a, ax);
ay_filtered = filtfilt(b, a, ay);
az_filtered = filtfilt(b, a, az);
% Plot accelerometer data
figure;
subplot(2,1,1);
plot(1:numReadings, ax_filtered, 'r', 1:numReadings, ay_filtered, 'g',
1:numReadings, az_filtered, 'b');
title('Filtered Accelerometer Data');
legend('Ax', 'Ay', 'Az');
xlabel('Sample Number');
ylabel('Acceleration (m/s²)');
% Plot gyroscope data
subplot(2,1,2);
plot(1:numReadings, gx, 'r', 1:numReadings, gy, 'g', 1:numReadings, gz, 'b');
title('Gyroscope Data');
legend('Gx', 'Gy', 'Gz');
xlabel('Sample Number');
ylabel('Angular Velocity (rad/s)');
%% Integration to Compute Velocity and Position
% Sampling rate information
Fs = 10; % Sampling frequency in Hz (based on Arduino code's delay of 100ms)
dt = 1 / Fs; % Time interval between samples in seconds
% Now, we will use a function to calculate and plot the position
calculateAndPlotPosition(ax_filtered, ay_filtered, az_filtered, gx, gy, gz,
dt);
% Function to calculate and plot position based on accelerometer and gyroscope
data
function calculateAndPlotPosition(ax, ay, az, gx, gy, gz, dt)
    numReadings = length(ax);

    % Initialize orientation (quaternion)
```

```
   orientation = quaternion(1, 0, 0, 0); % Assuming initial orientation is
identity quaternion

   % Initialize arrays for storing orientation, velocity, and position
   orientations = quaternion.zeros(numReadings, 1);
   orientations(1) = orientation;

   velocity = zeros(3, numReadings);
   position = zeros(3, numReadings);

   % Loop through the data to estimate orientation and integrate acceleration
   for i = 2:numReadings
       % Angular velocity vector (gyroscope readings)
       omega = [gx(i); gy(i); gz(i)]; % In radians per second

       % Update orientation using gyroscope data (integration)
       delta_theta = omega * dt; % Change in orientation angle in radians

       % Convert delta_theta to quaternion
       delta_q = quaternion(delta_theta', 'rotvec'); % delta_theta' is 1x3

       % Update orientation
       orientation = orientations(i - 1) * delta_q; % Quaternion multiplication

       % Normalize quaternion to prevent drift
       orientation = normalize(orientation);

       orientations(i) = orientation;

       % Rotate accelerometer data to global frame
       acc_body = [ax(i); ay(i); az(i)];
       acc_global = rotateframe(orientation, acc_body');
       acc_global = acc_global';

       % Compensate for gravity
       gravity = [0; 0; 9.81];
       linear_acc = acc_global - gravity;
```

```
        % Integrate acceleration to get velocity
        velocity(:, i) = velocity(:, i - 1) + linear_acc * dt;
    end

    % Apply detrend filter to velocity data to reduce drift
    velocity_detrended = detrend(velocity')';

    % Integrate detrended velocity to get position
    position = cumsum(velocity_detrended * dt, 2);

    % Plotting the Position
    figure;
    plot3(position(1, :), position(2, :), position(3, :));
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    zlabel('Z Position (m)');
    title('IMU Position Over Time (Velocity Detrended)');
    grid on;
    axis equal;
end
%%
figure;
% X Position
subplot(3, 1, 1); % 3 rows, 1 column, first plot
plot(position(1, :), 'r');
xlabel('Sample Number');
ylabel('X Position (m)');
title('X Position Over Time');
% Y Position
subplot(3, 1, 2); % 3 rows, 1 column, second plot
plot(position(2, :), 'g');
xlabel('Sample Number');
ylabel('Y Position (m)');
title('Y Position Over Time');
% Z Position
subplot(3, 1, 3); % 3 rows, 1 column, third plot
plot(position(3, :), 'b');
xlabel('Sample Number');
```

```
ylabel('Z Position (m)');
title('Z Position Over Time');
```

Appendix F: Materials and Expenses

Item	Description	Manufacturer	Mft Pt#	Vendor	Date	QTY	Cost Each	Total	Link
Non-Electronics									
8 AA Batteries	batteries	Amazon Basic	$840,083,632,039.00	Amazon	10/30/2024	11	$7.66	$7.66	Amazon Basics 8-Pack AA Alkaline Batteries, 1.5 Volt, Long-Lasting Power
Battery Cell Box	Holder for the batteries	Goweewon	GW-3AA	Amazon	10/30/2024	1	$10.60	$10.60	https://www.amazon.com/Goweewon-AA-Battery-Holder-Leads/dp/B0CWFCPNRT/ref=sr_1_5?crid=19TDHQHE6UIWC&dib=eyJ2IjoiMSJ9.kiveJc35vs9m
Tough PLA	Ultimaker Tough PLA	Makerspace				500 g	$0.08/g	$40.00	https://making.engr.wisc.edu/3d-printers/
Barbell Clamp	Barbell Clamp	GW tech	661757455941-	Amazon	10/17/2024	1	$15.77	$15.77	Amazon.com: GW Tech Barbell Clamps 2 inch, Heavy Duty Exercise Collars 2" Quick Release Pair of Locking Pro Olympic Weight Bar Plate Locks Collar Clips
Command Strip	attachment of our box to	3M	B073XR4X72	Amazon	10/30/2024	1	$16.90	$16.90	Command 15 lb Large Picture Hanging Strips 14 Pairs (28 Command

	the clip								Strips), Damage Free Hanging Picture Hangers, No Tools Wall Hanging Strips for Christmas Decorations
Electronics									
Arduino Nano	Arduino Nano	Arduino	A000005	Arduino	10/13/2024	2	$26.73	$53.46	https://store.arduino.cc/products/arduino-nano?srsltid=AfmBOopWjqK-rkYIleqGizCt6IPV-0Kpnh_kMw9wSPioA0jNm1Vpr0mn
MPU6050	IMU	Arduino	SEN-10937	SparkFun	10/13/2024	2	$8.05	$16.10	https://www.sparkfun.com/products/10937
Breadboard	Breadboard	Breadboard - Self-Adhesive (White)		SparkFun	10/13/2024	2	$10.39	$20.78	https://www.sparkfun.com/products/12002
MPU6050 Module	Holder for the MPU 6050	HiLetGo	B00LP25V1A	Amazon		1	$10.99	$10.99	HiLetgo GY-521 MPU-6050 MPU6050 3 Axis Accelerometer Gyroscope Module 6 DOF 6-axis Accelerometer
							Total	$146.11	