Precision Dumbbell Assistant

By

Ellie Beck

Ronit Kumar

Cole Trautman

Final Report for ECE 445, Senior Design, Spring 2024

TA: Douglas Yu

1 May 2024

Project No. 40

Abstract

The Precision Dumbbell Assistant is a wearable fitness device that helps users perform bicep curls with proper form while also keeping track of their number of repetitions. This device uses two 6-axis Inertial Measurement Units (IMUs) to track the orientation of the user's upper and lower arm in the form of roll, pitch, and yaw. This data is processed by an ESP32 microcontroller as an input into form analysis that provides the user feedback through both a buzzer and a web-based application. Through extensive testing, the accuracy of the IMU sensors and real-time form analysis was verified. This project shows potential to be able to extend to other dumbbell exercises as well due to the dynamic nature of the sensor placement and form analysis. Users can significantly benefit by using this device at a home gym by maintaining proper form without the assistance of extensive gym equipment.

Contents

1. Introduction

1.1 Problem

Many gym goers struggle to maintain proper form during their workouts with dumbbells, which is why they rely heavily on exercise machines. Maintaining proper form is important for two reasons. Bad form can increase the risk of injuries, especially with heavier weight, and can reduce the efficiency of the exercise, making it less effective at building muscle and strength [1]. Many people want to have some sort of at-home gym so that they can work out in the comfort of their own home and maybe avoid paying a gym membership fee, but they will miss out on all the equipment that a full gym has to offer. If you are trying to construct an at-home gym, often all you will have, at least to start, is a set of dumbbells and a bench. Hence, there should be a relatively cost-effective way to help people maintain proper form even when they just use dumbbells so that they can get the maximum benefit from their exercise.

1.2 Solution

We designed a device that tracks the user's arm to ensure that their form is correct. Our design uses two 6-axis (accelerometer and gyroscope) IMU (inertial measurement unit) sensors to calculate the orientation of the user's arm. There is a small sensor board located on the lower arm, and a larger main board with another sensor and the main processor on the upper arm. This allows us to track the orientation of each part of the arm and determine whether the movement is correct or not. We have also developed an algorithm to detect the correct movement for an exercise, as determined by experts in the field. At the moment, the only supported exercise is dumbbell curls, but more exercises could be added in the future. When incorrect form is detected, the user is notified with a buzzer, and more detailed information is provided through a web-based app. The app connects to the processor via Bluetooth Low Energy, and allows the user to view their past set and see the number of reps and areas in which incorrect form was detected.

1.3 Visual Aid

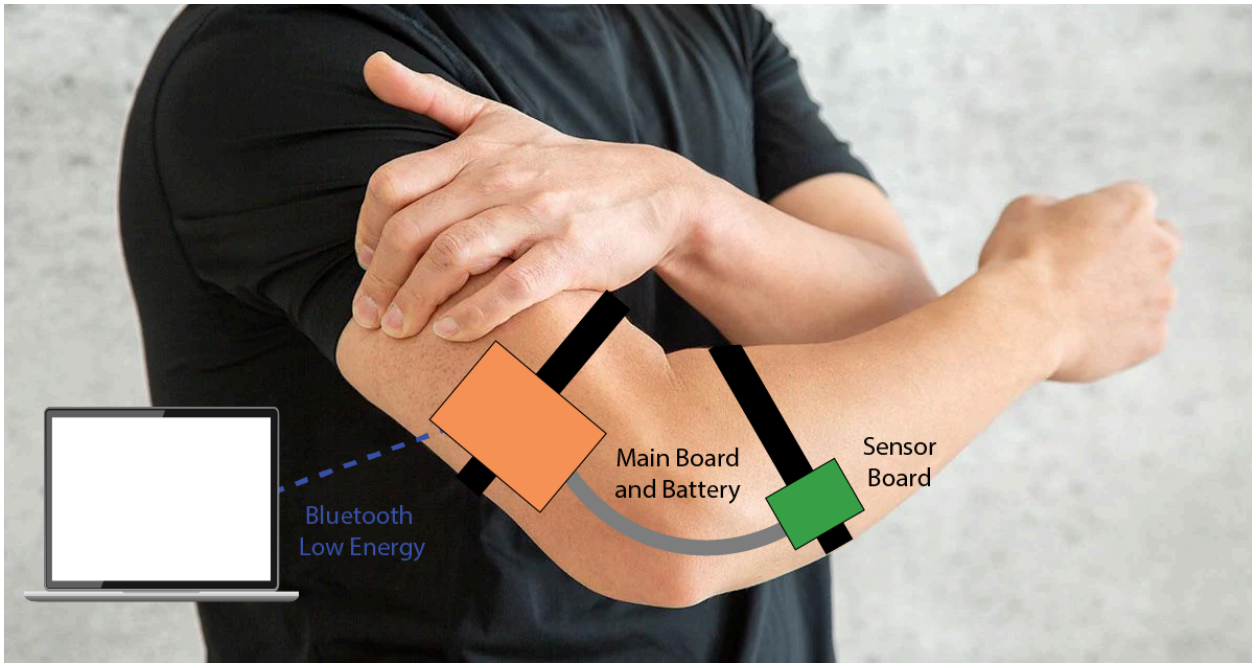

Figure 1: Visual aid of device positioning

As seen in Figure 1, our device has sensors attached to the user's arm in two locations: one on the back of the tricep, and one on the bottom of the forearm. If worn correctly, the cable going from the tricep to the forearm will go directly over the elbow. Each board is housed in a 3D printed enclosure so that it is protected from damage and can more easily and comfortably be attached to the user's arm. The board on the back of the tricep has all the main circuitry and its enclosure houses the battery, so it is bigger and heavier. Both enclosures can be attached to the user using elastic straps, and there is foam tape on the back of the enclosures to help them stay in place.

1.4 Block Diagram

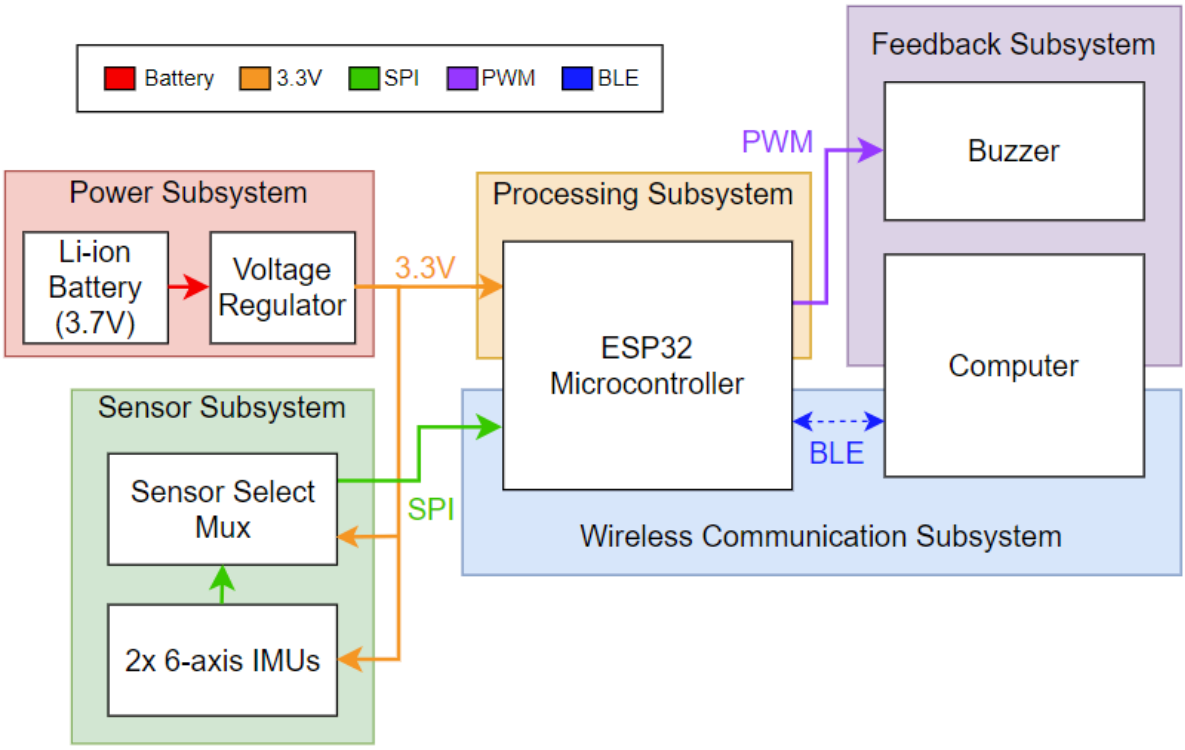

Figure 2: Precision Dumbbell Assistant Block Diagram

As seen in Figure 2, our device has 5 subsystems: Sensing, Processing, Wireless Communication, Feedback, and Power. The sensing subsystem is responsible for providing all the data necessary to calculate the position of the user's arm, namely acceleration and angular velocity data. This data is fed into the processing subsystem, which is responsible for converting acceleration and angular velocity into position and orientation. The processing subsystem is also responsible for calculating whether the position and orientation of the sensors are within the acceptable bounds for the exercise. The wireless communication subsystem is responsible for establishing and maintaining a wireless link between the device and a computer that is used to send data to a web app. The feedback subsystem is responsible for providing feedback to the user, both through a buzzer on the device and through the web app. Finally, the power subsystem is responsible for providing the other subsystems with the correct voltage.

1.5 High Level Requirements

In order for our project to be considered successful, we had to meet three requirements. The first was that the device needs to be accurate and consistent in motion and form analysis. Each sensor should be able to correctly calculate its position and orientation relative to a starting point within a tolerance of ±5% of the true values. We were able to partially complete this requirement, as we were able to

accurately calculate the orientation, but not position. The second requirement is that the device must be able to give the user feedback quick enough and loud enough to diagnose incorrect form. The entire system should read sensor data, analyze it, and provide feedback in no more than 50 ms, meaning that feedback should be provided at a minimum of 20 Hz. We were able to meet this requirement, because our code reads in and analyzes data at 100 Hz. The final requirement is that our device should not restrict the movement of the user in any significant way. Its weight must be negligible (< 200g). All the connections must be flexible enough and of appropriate length so that the device fits most arms without getting in the way. We were also able to meet this requirement because the device weighs under the threshold, and there is no noticeable resistance from the wire harness.

2. Design

2.1 Power Subsystem

For the battery, we picked the Tiny Circuits ASR00012 [2]. This battery supplies 3.7V and is small enough and light enough to meet our high level requirement of making the device comfortable to wear, while also being readily available at an affordable price. It is also rechargeable, which is important for a device that will be used regularly. The maximum discharge current is 1000mA and the minimum current needed by the ESP32 (see section 2.2 Processing Subsystem) is 0.5A so it filled that requirement as well [3]. We also needed to include the battery mating connector (JST SM02B-SRSSTB). We chose to use a LP2950CZ voltage regulator that was available through the ECE Service Shop because it supplies 3.3V, the voltage needed by all of our components, with an accuracy of ±1% which is well within our high level requirement of ±5%. It also offers battery protection with thermal shutdown and short circuit protection which protects the user from possible battery overheat without us having to add more battery protection circuitry [4]. We chose a standard USB A connector (87583-2010RPLF) because it allowed us to supply power and also use the USB programmer built into the ESP32. Our first iteration of the PCB had the USB 5V routed into the main components instead of the voltage regulator, but once we learned that USB provided 5V, we routed it into the regulator so that it could be dropped down to the correct 3.3V. The up-to-date design can be found below in Figure 3.

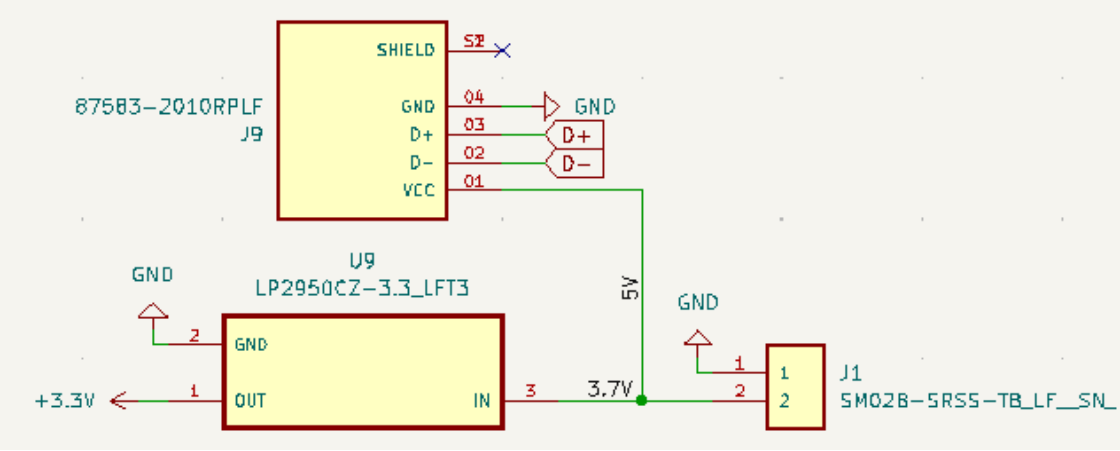

Figure 3: Power Subsystem Schematic

2.2 Processing Subsystem

We decided to use an ESP32-S3 as the microcontroller for this project because of its Bluetooth capability and processing power, as well as the ability to easily get it from the ECE shop. When selecting a microcontroller, we also had to make sure that it had the necessary SPI and PWM functionality [3]. The subsystem contains just the microcontroller, so it is relatively simple from the hardware point of view, as shown in the schematic, Figure 4. Almost all of the complexity is on the software side, because that is where the data is processed and analyzed. This software runs in a loop after the sensor initialization process has been completed, which will be discussed in the Sensing section. The main software loop (shown in Figure 5 below) retrieves both accelerometer and gyroscope data, processes it into usable orientation data, and then analyzes it using a form analysis algorithm we developed.

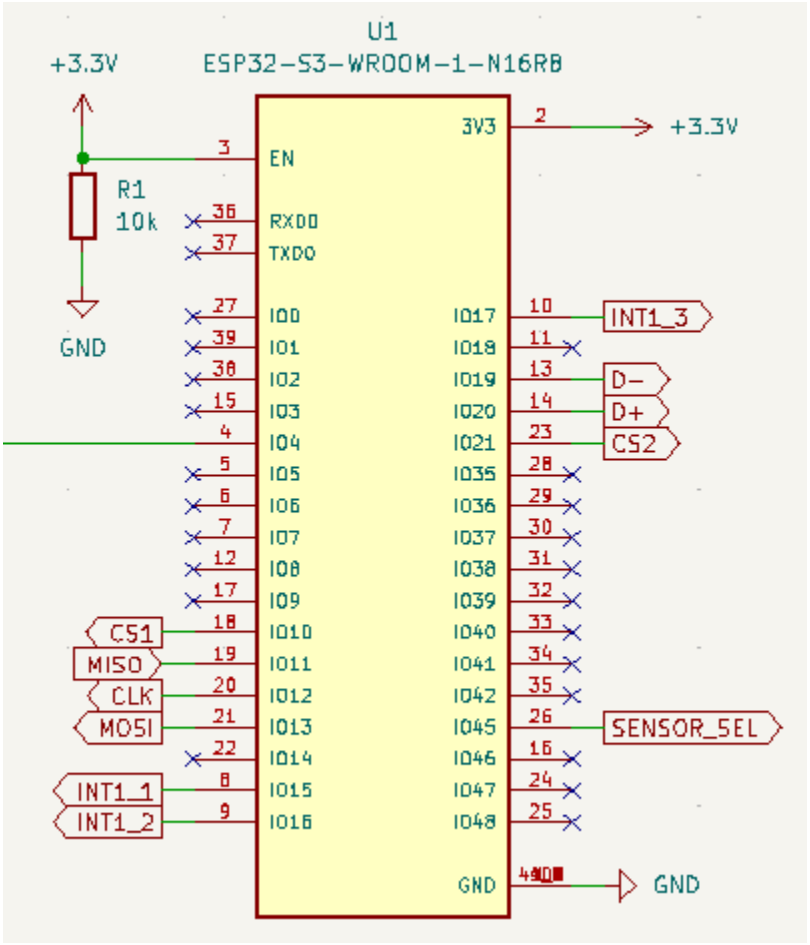

Figure 4: Processing Subsystem Schematic

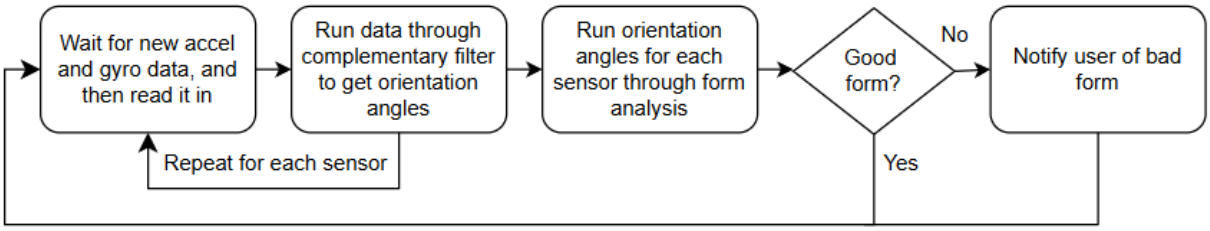

Figure 5: Processing Subsystem Software Flowchart

We had originally planned on processing the acceleration data into position data so that we would have more than just orientation to base our form analysis on, but we realized that it would be impossible to get any kind of good data. We learned that any sensor will have some inherent bias, so if we need to integrate our measured data to get data that we can actually use, data from at least two different sensors needs to be put through a filter to get an accurate value. This unfortunately ruled out position, because we had no other way to track the position besides the accelerometer. In real-world applications, the most common way to solve this problem is to supplement the acceleration data with GPS on large scales, and some kind of optical sensor on small scales [5]. Fortunately, we could still get accurate orientation data, because the IMU provides two ways to calculate the orientation: acceleration caused by gravity and angular velocity. We can calculate the orientation of the sensor at rest using just the accelerometer, because it measures the acceleration due to gravity. By putting different acceleration vectors into the inverse tangent function, we can calculate each of the orientation angles. This does not hold up when the sensor is in motion, however, because other acceleration forces could be acting on it. For this reason, we also need to use the angular velocity data from the gyroscope. By multiplying the most recent angular velocity by the time step (0.01 seconds), we can get the angular displacement of the sensor, and then add it to the last calculated orientation. This value can be affected by bias because it is being integrated, so in order to get an accurate value, we must use a filter of some kind. We chose to use a complementary filter because it is relatively easy to implement and gives us the accuracy we need. It takes 3% of the accelerometer orientation, and 97% of the gyroscope orientation, and adds them together to form the new calculated orientation. A diagram of the filter can also be found as Figure 6 below. This gives us the long-term stability of the accelerometer calculation (it doesn't drift), as well as the short-term accuracy of the gyroscope (the calculation is not affected by external forces)[6].

The form analysis algorithm is also included in the processing subsystem, and is much more simple than we originally thought it would be. Because we moved to a purely orientation based approach, all we needed to do was check whether each sensor stayed within the bounds we defined, and otherwise, notify the user. These bounds were determined experimentally, but with supervision and consultation with online resources. For example, if the bicep sensor measured pitch is greater than 110 degrees, that means that the user is likely leaning back, so that information is relayed to the Feedback subsystem and on to the user. The form analysis algorithm also tracks reps based on whether the forearm sensor hits certain pitch thresholds while all sensors stay in bounds.

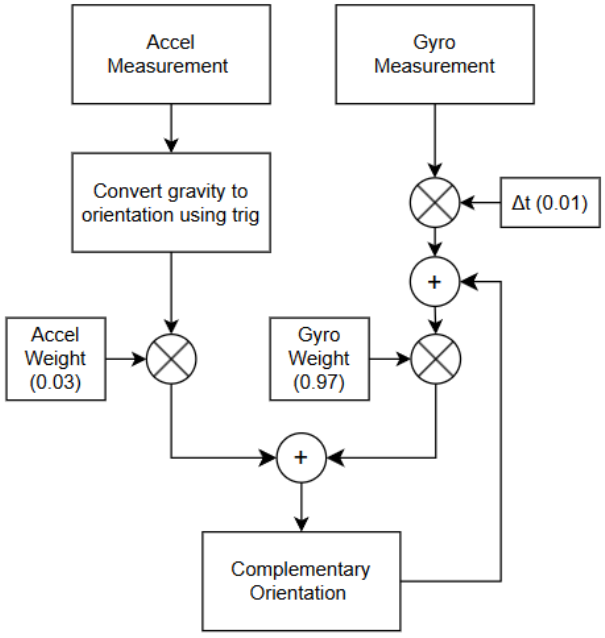

Figure 6: Orientation Complementary Filter Flowchart

2.3 Sensing Subsystem

The sensing subsystem is based around two LSM6DSM IMU sensors. These sensors were selected because they provide both acceleration and angular velocity data, because we originally wanted to calculate both the position and orientation of each sensor. Each sensor is connected to the ESP32 via SPI. We had originally selected the ESP32-S3 because it has 4 SPI buses, which would let us interface with each sensor concurrently. We found out two of the SPI buses are for internal use only, so we had to pivot to a round-robin system, where each sensor is read from individually [3]. In our original design, before our first PCB order, we had the CLK, MOSI, and MISO lines wired directly into each sensor, with only the CS pin being different. We realized, however, that the sensors are in I2C mode when CS is high and SPI mode when CS is low [7]. This meant that even if a sensor is not selected, it could possibly still send and receive data, thinking that the messages coming in are in I2C format. For that reason, we decided to isolate the SPI lines for each sensor using 4-bit muxes, meaning that if a sensor is not selected, all of its SPI lines are electrically disconnected from the ESP32. An example of this mux can be seen as U11 in the schematic, Figure 7 below. After our second PCB order, we realized that we had another problem with the muxes. If a sensor was not selected, its CS line would be floating, which would cause undefined behavior. To fix this, we added a 470Ω pull-up resistor to each CS line. This value was determined experimentally by trying different resistances and choosing the one that had the quickest rise time while also allowing the voltage to drop to zero when the sensor is selected. An example schematic can be seen as Figure 8 below.

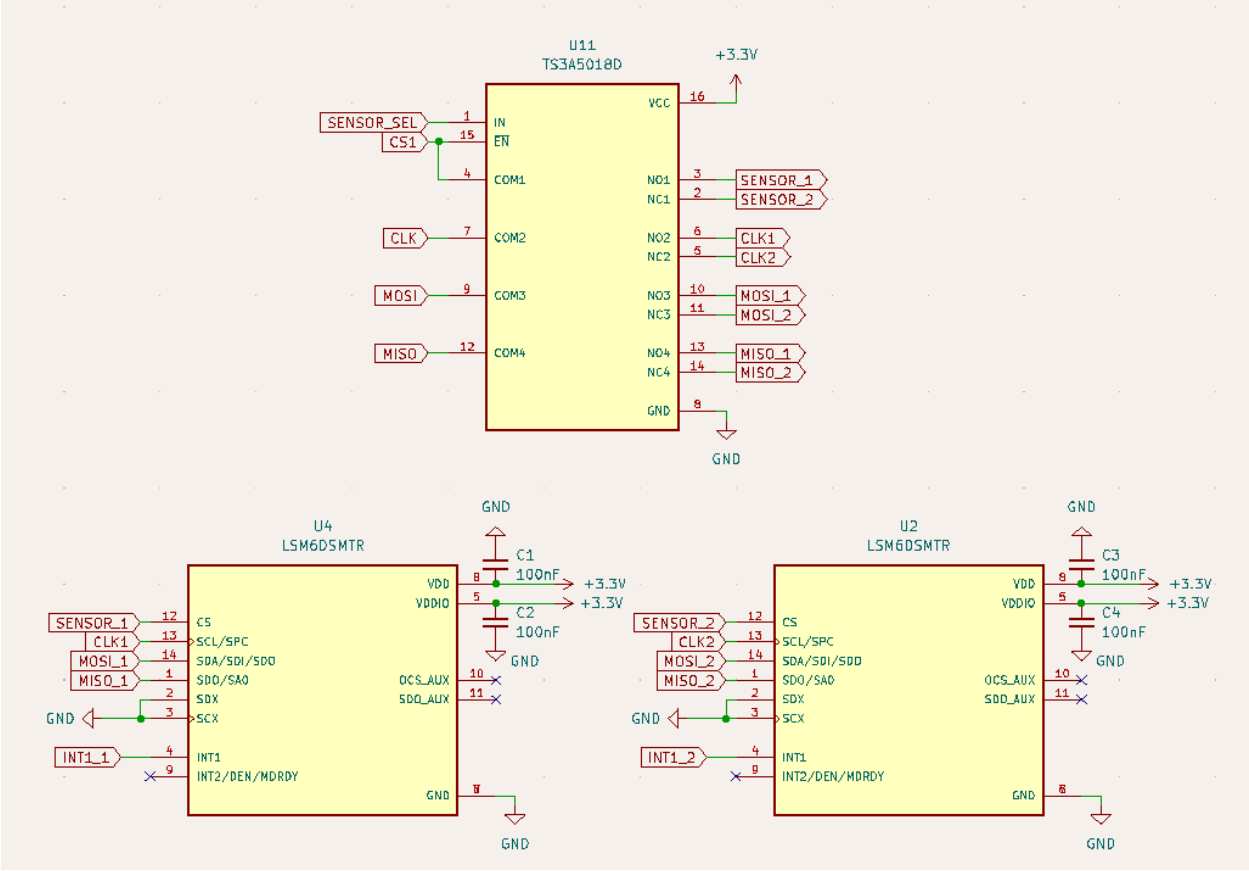

Figure 7: Sensing Subsystem Schematic

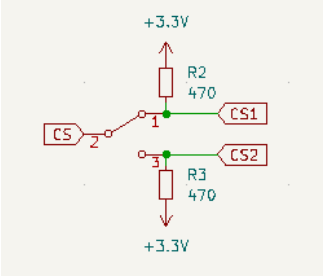

Figure 8: CS Pull-Up Example Schematic

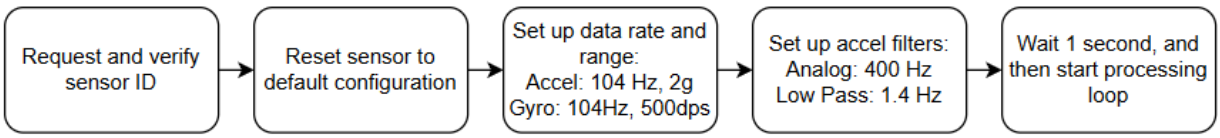

Figure 9: Sensing Subsystem Software Flowchart

The sensing subsystem also includes the software necessary to initialize and communicate with each sensor. To do this, we used the driver distributed by ST, the sensor manufacturer [8]. The driver is platform-independent, meaning that we only needed to define SPI read and SPI write functions, which are then called by the driver functions. On startup, the ESP32 sends out an ID request, and if the sensor

responds with the correct ID, we can assume that SPI communication is working. The ESP32 then sends out a command to reset the sensor to its default configuration, and waits until the sensor responds with confirmation that it has done so. Next, the microcontroller sends out messages to set the data output rate to 104 Hz for both the accelerometer and gyroscope, and to set the accelerometer scale to 2g and the gyroscope scale to 500 degrees per second. Finally, the ESP32 sends out a message to enable an analog filter to smooth the data and a low pass filter to filter out noise on the accelerometer. After this startup sequence, the code waits for 1 second for the sensor to warm up, and then starts reading and processing data from it. A flowchart explaining the startup process can also be found above in Figure 9.

2.4 Feedback Subsystem

The feedback system is important for real-time interaction with the user. The processed ESP32 data is sent to this subsystem through the wireless communication subsystem. The bicep curl form feedback from the data is displayed in both an auditory and visual form. Based on the form analysis that is done using various roll, pitch, and yaw thresholds on both the lower arm and upper arm orientation, incorrect form is detected. Incorrect form triggers a buzzer using PWM signals. The buzzer is programmed to run at a pitch of 200 Hz and at half the duty cycle, which means it will sound at half the maximum volume of the buzzer. The web-based application will then display the number of repetitions of bicep curls, which is normally tracked by making sure the user reaches a high and low pitch threshold value for the lower arm orientation, and descriptive feedback that tells the user to prevent moving his elbow away from his body, prevent bending his back, or prevent performing any other unadvised motion during the bicep curls. A flowchart for the software part of the subsystem can be found as Figure 10 below.

To alert the user to look at the app feedback, we chose the Piezo buzzer due to its compact nature that fit well with our PCB plan along with its energy efficiency. This buzzer had the ability to deliver a high pitch sound in our target range of 200-600 Hz. In addition, we used HTML and CSS for formatting and styling the web-based app respectively. Javascript was used for the processing of information and the functionality of buttons on the app [9]. One button was created to let the user connect to a BLE device directly from the app instead of using Bluetooth LE Explorer or any other medium. The app was run on Google Chrome, which required a secure HTTP server due to Chrome's security restrictions. A major challenge was faced when trying to develop an Android-based application instead since Android Studio did not let the BLE permission process through and the Android Emulator did not have the necessary hardware to support Bluetooth functionality. These were the main reasons we decided to change the design to include a web-based application.

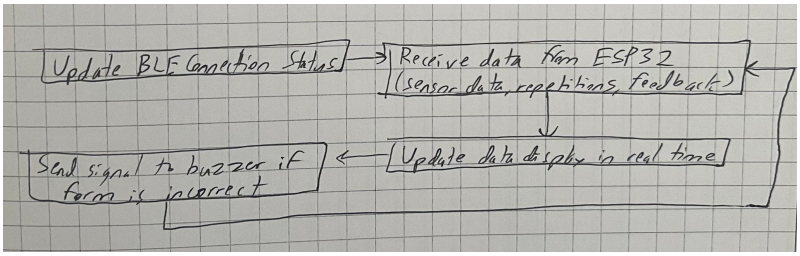

Figure 10: Feedback Subsystem Program on JavaScript Flowchart

2.5 Wireless Communication Subsystem

For the wireless subsystem, we chose to use Bluetooth Low Energy (BLE) instead of Bluetooth Serial due to BLE's power efficiency and ability to extend battery life. BLE allows for communication between the ESP32 microcontroller and the user's computer. After the ESP32 is powered on, BLE is initialized. Then, we declared the ESP32 as a BLE device and set up a BLE server on it. This server hosts a service that has characteristics that contain the data values for the roll, pitch, and yaw of both sensors, number of bicep curl repetitions, and also descriptive feedback all in one string value that is deserialized in the web-based application for display purposes. Advertising is also configured for the client, which is the computer, to be able to discover the ESP32 as a BLE device nearby. The software flowchart is shown as Figure 11 below.

Through the entire interaction between the server and client, this subsystem utilizes the standard BLE capabilities of the ESP32 microcontroller. BLE creates a successful connection between the server, which is the ESP32, and the client, which is the user's computer containing the web-based application [10]. After the connection is successful, the sensor data, number of repetitions, and bicep curl form feedback is sent via BLE from the ESP32 to the computer for real-time display updates on the application. If the BLE connection disappears, the application stops receiving real-time updates from the ESP32. Advertising is reset to give the user a chance to attempt a BLE reconnection. Along with the challenges faced with Android Studio, we were unable to add BLE functionality to the PCB design since there is an error in the ESP32-S3 library for BLE functionality with Arduino IDE framework. Hence, BLE only works with the breadboard prototype. Either using the old ESP32 version on the PCB design or programming the ESP32 using ESP-IDF would have most likely solved this problem.

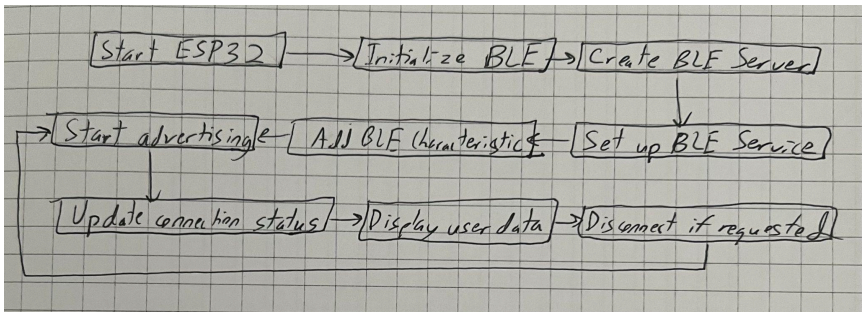

Figure 11: Wireless Communication Subsystem Software Flowchart

2.6 Physical Design

For the harness, we wanted to use connectors that were available in the ECE Service Shop. We initially thought we needed to carry 7 signals between the bicep and forearm PCBs (3.3V, GND, Sensor Select, CLK, MOSI, MISO, and Interrupt). We found that the interrupt signal was not needed. The connectors we chose (65039030LF and 69167-107HLF) had 7 pins but in future designs we would want to choose a 6 pin connector. Additionally, these connectors are vertical connectors but ideally we would have 90 degree connectors in order to minimize the bend radius of the harness. For the length of the wires on the harness, we used data from NC State University to find the length from the center of the bicep to the center of the forearm of a 99th percentile man [11]. We found that this was 14.551 inches. We added 2 extra inches to account for the bend radius required due to the connectors being vertical for a total of 16.551 inches of wire. We wanted the wires to be long enough that even the tallest person could use them without snapping a wire but not overly long which could cause the user to snag the wires on obstacles. The other component of the physical design was the PCB/battery enclosures. We were originally planning on just having a battery enclosure underneath the bicep PCB, but a student at our design review recommended we add enclosures on top of the PCBs as well, so we did so. The enclosures are 3D printed out of PLA, and are designed to snap together. Unfortunately, the printer we used was not able to print the parts to a tight enough tolerance, so we ended up just taping the enclosure parts together. The enclosures attach to the user's arm with elastic bands that slide through slots in the enclosures. This worked well enough, but other attachment options like Velcro might be easier to put on, albeit a little bit heavier. The sensor board enclosure also has foam tape on the bottom to protect the user from electrical components on the bottom of the PCB. CAD models of both enclosures can be seen in Figure 12 below

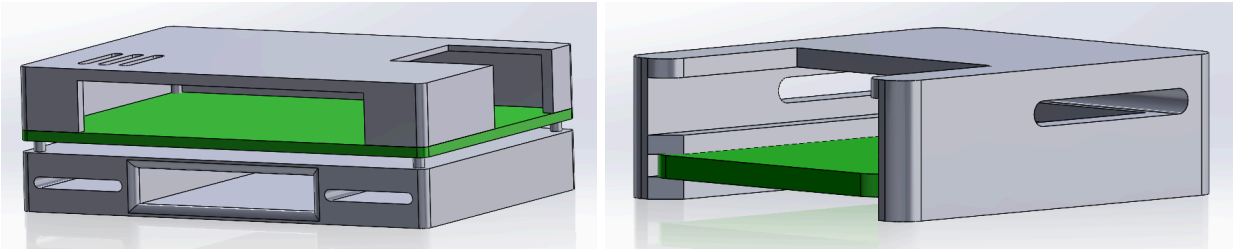

Figure 12: Bicep PCB and Battery Enclosure (left) and Sensor PCB Enclosure (right)

3. Verification

3.1 Power Subsystem

For the Power Subsystem, our requirement was for the voltage regulator to supply 3.3V± 5%. To test this, we used a multimeter to probe the power output and ground pin of the voltage regulator with the battery connected. We completed 10 trials to ensure consistent success. Refer to the Appendix A (section A.1.1) to see the requirements and verification table (Table A.1) as well as the verification data (Table A.2). The power subsystem passed all of the verification trials and therefore can be considered a success.

3.2 Processing Subsystem

The Processing Subsystem has a total of four requirements. The first is that the microcontroller should be able to calculate the position and orientation of each sensor with an accuracy of ±5% (Table A.3). This is met because we are able to calculate orientation within these requirements (Table A.4). We tested this by rotating the sensor along each axis (pitch, roll, and yaw), and measuring the results, which can be found in Appendix A. The second requirement is that the position and orientation of each sensor should be calculated and analyzed at at least 20 Hz (Table A.5). This requirement is partially met because we are able to calculate orientation at 100 Hz, but not the position (Table A.6). This is measured by probing a GPIO pin on the ESP32 that toggles every time a full set of data is analyzed, so the processing frequency is two times the measured frequency. The results of this test can also be found in Appendix A. The third requirement is that the calibration sequence must take less than 20 seconds and provide consistent results (Figure A.7). This requirement is met because no calibration is needed due to the accelerometer orientation calculation, which always has gravity as a reference (Table A.8). The device only has a startup sequence of about 1.25 seconds, well below the 20 second threshold. Finally, SPI communication must operate between each IMU sensor and the ESP32 microprocessor at a speed of at least 10 ± 1.0 MHz (Table A.9). This requirement is met because we are able to probe the CLK pin on the ESP32 and check its frequency, which is recorded in Appendix A (Table A.10).

3.3 Sensing Subsystem

The Sensing Subsystem has two requirements. The first requirement is that each IMU sensor should provide acceleration and angular velocity with an accuracy of ±5% (Table A.11). This requirement is met because the data received from the sensor is within the bounds, and is recorded in Appendix A (Table A.12 & A.13). The second requirement is that each IMU sensor should provide data at a rate of at least 20 Hz for a total combined rate of 60 Hz (Table A.14). This requirement is met because although we only

used two sensors, they are polled at 100 Hz each (Table A.15). This was tested with a GPIO pin in a very similar fashion to the processing frequency requirement above.

3.4 Feedback Subsystem

For the feedback subsystem, our requirement was for the buzzer's pitch to be between 200 and 600 Hz for at least a 1 second duration (Table A.16). The buzzer's pitch was specifically set to 200 Hz in the ESP32 code and was set to run for half of the maximum duty cycle, which means we met the above requirement (Table A.17 & A.18). The buzzer also needed to have a loudness value of at least 60 dB (Table A.19), and this was measured at approximately 65 dB (Table A.20), which means we met this requirement as well.

3.5 Wireless Communication Subsystem

For the wireless communication subsystem, we required that the BLE connection must work between the ESP32 and the user's device for at least 5 meters (Table A.21). We used the nRF Connect app, an application created by Nordic Semiconductor that is commonly used to debug BLE connections, to make sure that a smartphone can connect to the ESP32 at various distances up to 5 meters (Table A.22). This test passed for every distance. In addition, we required that the BLE packet loss must be 20% or less (Table A.23). We wrote a test case in the ESP32 code and displayed the results on the web-based application. A message was sent over BLE at a fixed interval of 1 second each. The real time for each message sent was tracked along with the message count. This test also passed since the message count incremented each time the real time incremented, meaning that there was no packet loss since a message was successfully being sent over BLE each second (Table A.24).

3.6 Physical Design

The physical design has two requirements. First, the bicep assembly (PCB, battery, and enclosure) must weigh less than 100g (Table A.25). It weighs 84g (Table A.26), so the requirement is met. The forearm assembly must also weigh less than 50g (Table A.27), and it does, at 24g (Table A.28).

4 Cost & Schedule

4.1 Cost Analysis

Tables 1 and 2 below show both parts and equipment costs for our project.

Table 1 Parts Costs

Description	Qty	Manufacturer	Mfg Part #	Value	Source	Unit Cost	Total Cost
Microcontroller	1	HiLetgo	ESP-WROOM-32	N/A	ECE Supply Center	$16.53	$16.53
Piezo Buzzer	1	Cylewet	CYT1008	5V	Amazon	$0.59	$0.59
Voltage Regulator	1	Texas Instruments	LP2950CZ	3.3V	ECE Services Shop	$1.09	$1.09
Conn Header SMD	1	JST Sales America Inc	SM02B-SRSS-TB	N/A	Digikey	$0.51	$0.51
Lithium Ion Battery Pack	1	TinyCircuits	ASR00012	3.7V	Mouser	$8.52	$8.52
7 Pin Connector Cable	2	Amphenol FCI	65039030LF	N/A	Electronics Service Shop	$0.92	$1.84
7 Pin Connector PCB	2	Amphenol FCI	69167-107HLF	N/A	Mouser	$1.44	$2.88
25 Feet 20 Gauge Wire	1	NTE Electronics Inc	WHS20-02-25	20 AWG	ECE Supply Center	$9.20	$9.20
040 60/40 ROSIN SOLDER	1	Kester	24-6040-0039	N/A	ECE Supply Center	$16.96	$16.96
1kg Spool PLA	1	AnyCubic	N/A	N/A	Amazon	$14.99	$14.99
6 Axis IMU	2	STMicroelectronics	LSM6DSMTR	N/A	Mouser	$4.17	$8.34
Resistor	1	Bourns	CR0805-JW-471ELF	470Ω	Mouser	$0.10	$0.10
Resistor	1	Stackpole Electronics Inc	RMCF0805JG10K0	10kΩ	Electronics Service Shop	$0.10	$0.10
Capacitor	4	Samsung Electro-Mechani	CL21F104ZAANNNC	0.1uF	Electronics Service	$0.10	$0.40

		cs			Shop		
Bicep PCB	1	JLC PCB	N/A	N/A	JLC PCB	$0.40	$0.40
Forearm PCB	1	JLC PCB	N/A	N/A	JLC PCB	$0.80	$0.80
						Total Cost:	**$83.25**

Table 2 Equipment Costs

Item	Make	Model	Retail Cost	Actual Cost
Soldering Iron	Uline	H-10799	$190	$0
Vise	Unknown	Unknown	~$20	$0
Oscilloscope	Aigilent	DSO7104B	$24,081	$0
Scale	Amazon Basics	Kitchen Scale with LCD	$12.59	$0
Measuring Tape	Amazon Basics	Self-Locking Tape Measure	$7.97	$0
Mutlimeter	Keysight	34461A	$1,457	$0
3D Printer	Prusa	MK3	$899	$0
Computer	PC	Unknown	~$1,000	$0
Waveform Generator	Agilent	33500B Series	$3,973	$0
		Total Cost:	$30,621	$0

Assuming that the average hourly rate for an electrical/computer engineer is $50/hr, and that we each worked on the project for 85 hours, we can use the following equation for total labor costs:

$$\$50 \times 85 \text{ hours} \times 2.5 \text{ overhead} \times 3 \text{ people} = \$31875$$

Adding the parts, equipment, and labor costs together, we can calculate the total cost as:

$$\$83.25 + \$0 + \$31875 = \$31958.25$$

This cost is incredibly labor heavy, but if our project were to be produced on a larger scale the costs would be much more proportional.

4.2 Schedule

The schedule we followed can be found in Table 3 below.

Table 3 Schedule

Week	Task	Person
2/25 - 3/2	Started ordering parts for prototyping	Everyone
	Finished PCB schematic and started layout	Ellie
	Ordered sensor eval kit	Cole
	Researched BLE and get successful communication between dev board and computer	Ronit
3/3 - 3/9	Finished layout and ordered PCBs	Ellie
	Established SPI communication between eval kit sensor and dev board	Cole
	Got successful communication between dev board and computer	Ronit
3/10 - 3/16	Spring Break	Everyone
3/17 - 3/23	Start weighing components to ensure that they meet the weight limit, tested standalone components like buzzer	Ellie
	Tested sensor accuracy	Cole
	Worked on BLE communication between ESP32 and smartphone	Ronit
3/24 - 3/30	Assembled PCB v1	Ellie
	Started work on position and orientation calculations, pivoted to just orientation	Cole
	Start developing app UI with dummy data, work out bugs with BLE	Ronit
3/31 - 4/6	Second PCB revision	Ellie
	Worked on complementary filter	Cole
	Pivoted to web based app	Ronit
4/7 - 4/13	Assembled second PCB	Ellie

	Tuned complementary filter, started on form analysis	Cole
	Worked on web app with dummy data	Ronit
4/14 - 4/20	Continued PCB assembly and debugging	Ellie
	Brought together sensor and bluetooth code, continued working on form analysis	Cole and Ronit
4/21 - 4/27	Demo, fix final bugs	Everyone
4/28 - 5/1	Presentation and Final Paper	Everyone

5 Ethics & Safety

5.1 Ethical & Safety Issues

Accuracy of Form: The most important ethical issue is obviously the accuracy of form enforced by our device as it could lead to injuries if inaccurate. To ensure this, we used data from a certified online training source that we have referenced below. The IEEE and ACM code of ethics mention to prioritize user safety and having a risk of injury goes directly against the code [12] [13].
Privacy of Data: The privacy of data is also a cause of ethical concern as many people who workout are sensitive about information in their fitness diaries and quality of form in exercises. We needed to ensure safe storage of our data. The IEEE and ACM code of ethics clearly state to prioritize privacy and confidentiality of user data [12] [13].
Wearable Device: We also had to make sure that the device is wearable for the safety of the user. The wires and elastic straps can't be too tight and should not prevent the user from moving their body parts naturally. When designing the enclosure we had to make sure that it was safe and comfortable for the user to wear.
Safe Materials: Lastly, the materials that we used had to be considered safe to contact human skin. Since voltage travels through the materials used for the connections, they can't get overheated to the extent that it causes skin irritation.
Battery Safety: Lastly, the 3.7V batteries must not get overheated since they are present in an enclosure that is in very close contact with the user's skin. Overheating could also damage other electronic components in the design.

6 Conclusion

To conclude, we will discuss both our accomplishments over the course of the semester as well as areas where our project came a little bit short. We learned a great deal over the course of the semester, and we are confident that with more work our device will be able to even more accurately analyze bicep curl form, as well as analyze other dumbbell exercises.

Our precision bicep assistant is able to accurately and consistently offer form analysis to the wearer. We are able to accurately determine the orientation of the sensors on the bicep and the forearm and use that data to detect the position of the user's arm and decide if they are using proper form that is both safe and maximizes their muscle growth. The buzzer is able to sound when the user is in an unsafe or non-optimal position at a volume that the user can easily hear, allowing them to quickly adjust their movement. The device is small enough that it does not adversely affect the user's form and the wire harness is able to accommodate the entire range of motion of the exercise. We have a working web app that gives users feedback about their form and counts the number of correct repetitions they perform.

However, there is some room for improvement. Future steps for this project would include updating the PCB with the correct footprint for the new buzzer, adding the 470Ω pull-up resistor to the analog switch, removing the redundant analog switch and connector, and shrinking the size of the board to make it even more comfortable for the user. We could also improve the harness by using 90 degree connectors as opposed to vertical connectors which would minimize the bend radius of the wires. We could also utilize connectors with screws or other attachment pieces in order to more securely hold the harness in place. Ideally, we would also replace the ESP32 with a model that is Bluetooth Low Energy compatible and can interface with the feedback app. We have also considered using a louder buzzer to make the feedback easier to hear in a loud gym environment. This technology also has a lot of potential to scale to other dumbbell or non-dumbbell exercises. For many exercises, it would be sufficient to simply add a new form analysis algorithm to track the new exercise. For other exercises, we would possibly need to add more sensors to more parts of the body by repeating our success with the forearm sensor board.

7 Citations

[1] HROSM, “Importance of Proper Weight Lifting Form - HROSM,” Hampton Roads Orthopaedics Spine and Sports Medicine, Apr. 01, 2023. https://www.hrosm.com/importance-of-proper-weight-lifting-form/

[2] “Tinycircuits - Overview,” GitHub, https://github.com/TinyCircuits (accessed May 1, 2024).

[3] ESP32-S3-WROOM-1 ESP32-S3-WROOM-1U, https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf (accessed May 2, 2024).

[4] LP295x-N Series of Adjustable Micropower Voltage Regulators, https://www.ti.com/lit/ds/symlink/lp2950-n.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1714579307138&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Flp2950-n.

[5] “Pure IMU-based positional tracking is a no-go,” YouTube, https://www.youtube.com/watch?v=_q_8d0E3tDk (accessed May 1, 2024).

[6] “Complementary filters¶,” Complementary_Filters, https://vanhunteradams.com/Pico/ReactionWheel/Complementary_Filters.html (accessed May 1, 2024).

[7] “iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope.” Accessed: Feb. 22, 2024. [Online]. Available: https://www.mouser.com/datasheet/2/389/lsm6dsm-1849628.pdf

[8] STMicroelectronics, “STMicroelectronics/lsm6dsm-PID: Lsm6dsm platform independent driver based on standard C language and compliant with Misra Standard,” GitHub, https://github.com/STMicroelectronics/lsm6dsm-pid (accessed May 1, 2024).

[9] “HTML CSS and Javascript website design tutorial - beginner project fully responsive,” YouTube, https://www.youtube.com/watch?v=FazgJVnrVuI (accessed May 1, 2024).

[10] “Bluetooth LE & Bluetooth — ESP-FAQ documentation,” espressif-docs.readthedocs-hosted.com. https://espressif-docs.readthedocs-hosted.com/projects/esp-faq/en/latest/software-framework/ble-bt.html#:~:text=The%20maximum%20throughput%20of%20Bluetooth (accessed Feb. 23, 2024).

[11] C. C. Gordon, “Anthropometric Data,” NCU State, Apr. 2006. Available: https://multisite.eos.ncsu.edu/www-ergocenter-ncsu-edu/wp-content/uploads/sites/18/2016/06/Anthropometric-Detailed-Data-Tables.pdf

[12] “IEEE Code of Ethics.” *IEEE*, www.ieee.org/about/corporate/governance/p7-8.html. Accessed 22 Feb. 2024.

[13] “Software Engineering Code - ACM Ethics.” *ACM Ethics - The Official Site of the Association for Computing Machinery’s Committee on Professional Ethics*, 8 June 2022, ethics.acm.org/code-of-ethics/software-engineering-code/.

Appendix A (Requirements and Verification Tables)

A.1 Power Subsystem

Table A.1: R&V for Power Subsystem

Requirement
The voltage regulator must regulate the voltage to 3.3 V ± 5% in order to ensure that there is enough power supply to ESP32 and sensors but not enough power to damage electronic components. The LP2950CZ has internal protection circuitry including short circuit protection and thermal protection that will protect against many accidental issues.
Verification
Equipment
Digital multimeter, 3.7 V lithium batteries, voltage regulator prototype
Test Procedure
1. Connect the 3.7 V lithium batteries to the voltage regulator prototype using the corresponding terminals. 2. Make sure the batteries are on and supplying voltage. 3. Record the final voltage after the regulator using a digital multimeter. 4. Vary the resistance in the voltage regulator. 5. Repeat steps 3 and 4 10 times and record the results. 6. If each recorded voltage value is in range 3.3 V ± 5%, this test is successful.
Presentation of Results
Include a data table recording the voltages from all 10 trials in the notebook and final report. Also, plot a line graph with input resistance on x axis and output voltage on y axis.

Table A.2: Verification Data for Power Subsystem

Trial #	Measured Voltage (V)	% Error	Pass/Fail
1	3.298	0.06%	Pass
2	3.207	2.81%	Pass
3	3.288	0.36%	Pass

4	3.294	0.18%	Pass
5	3.255	1.36%	Pass
6	3.283	0.52%	Pass
7	3.301	0.03%	Pass
8	3.296	0.12%	Pass
9	3.267	1%	Pass
10	3.277	0.7%	Pass

A.2 Processing Subsystem

Table A.3: R&V for ESP32 Orientation Data

Requirement
The microcontroller should be able to calculate the position and orientation of each sensor with an accuracy of ±5%
Verification
Equipment
LSM6DSMTR IMU sensor, ESP32 development board, Yardstick, Protractor
Test Procedure
Position: 1. Connect IMU sensor to ESP32 development board using manual. 2. Place the IMU sensor on a hard flat surface with the chip facing up, and a yardstick next to it. 3. Move the sensor along the yardstick 50cm. 4. Check to see if the calculated position has changed in the correct direction 50cm ±5%. 5. Repeat steps 3-4 5 times, and record the results 6. Repeat steps 2-5 for the Y and Z directions (stand the yardstick up for Z) 7. If all values are within the ±5% tolerance, the test passes. **Orientation:** 1. Connect IMU sensor to ESP32 development board using manual. 2. Place the IMU sensor on a hard flat surface with the chip facing up, and a protractor next to it. 3. Using the protractor, move the sensor until it is 45 degrees from flat.

4. Check to see if the calculated orientation has changed in the correct direction 45° ±5%. 5. Repeat steps 3-4 5 times, and record the results 6. Repeat steps 2-5 for the other two axes (lay the protractor down for yaw) 7. If all values are within the ±5% tolerance, the test passes.
Presentation of Results
Record all readings in separate data tables in the notebook for the position and orientation, and take note whether each test passes or not.

Table A.4: Data for ESP32 Orientation

Trial #	Measured Angle (°)	% Error	Pass/Fail
1 (Pitch)	43.48	3.50%	Pass
2 (Pitch)	43.29	3.95%	Pass
3 (Pitch)	43.83	2.67%	Pass
4 (Pitch)	43.97	2.34%	Pass
5 (Pitch)	42.95	4.77%	Pass
6 (Roll)	43.69	2.99%	Pass
7 (Roll)	43.50	3.45%	Pass
8 (Roll)	43.47	3.52%	Pass
9 (Roll)	43.68	3.02%	Pass
10 (Roll)	42.88	4.94%	Pass
11 (Yaw)	43.65	3.09%	Pass
12 (Yaw)	42.95	4.77%	Pass
13 (Yaw)	43.38	3.73%	Pass
14 (Yaw)	43.78	2.79%	Pass
15 (Yaw)	43.55	3.33%	Pass

Table A.5: R&V for Sensor Data Frequency

Requirement
The orientation and position of each sensor should be calculated and analyzed at at least 20 Hz
Verification
Equipment
Precision Dumbbell Assistant, Oscilloscope
Test Procedure
1. Connect two sensor boards to the main board as shown in the visual aid 2. Connect the oscilloscope to a GPIO pin on the ESP32 3. Write a test program that toggles the value of the GPIO pin after every processing cycle (position and orientation update / analysis) 4. Check the frequency of the GPIO pin. If it is higher than 10 Hz, then the test is successful. 10 Hz is the target value because the pin is toggled on every cycle, so there will be two cycles for every period of the GPIO signal. 5. Run the test 5 times, and ensure that it meets the 20 Hz benchmark for every test.
Presentation of Results
Record the results of each test in the notebook and mark whether the test passed or not.

Table A.6: Data for Sensor Data Frequency

Trial #	Measured Frequency (Hz)	Actual Frequency (Hz)	Pass/Fail
1	49.00	98.00	Pass
2	49.30	98.60	Pass
3	48.57	97.14	Pass
4	49.41	98.82	Pass
5	48.62	97.24	Pass

Table A.7: R&V for Sensor Calibration

Requirement

The calibration sequence must take less than 20 seconds and provide consistent results.
Verification
Equipment
Precision Dumbbell Assistant, Stopwatch
Test Procedure
Original Procedure: 1. Put on the device, start the calibration sequence, and start the stopwatch 2. If the user is able to complete the calibration in 20 seconds or less, this part of the test passes 3. The user should perform 10 of the same correct movements to the best of their ability, and if they receive positive feedback on 9/10, the second test passes 4. Repeat steps 2-3 5 times, and if all 10 tests pass, the calibration sequence is verified. New Procedure: 1. No calibration is needed because the accelerometer calculates orientation from gravity 2. The startup sequence takes 1.25 seconds in code, well below the threshold, so this does not have to be rigorously tested
Presentation of Results
Record the calibration time and number of correct movements for each trial in the notebook, and mark whether each test passed or not.

Table A.8: Data for Sensor Calibration

Trial #	Startup Time (sec)	Pass/Fail
1	1.25	Pass

Table A.9: R&V for SPI Communication

Requirement
SPI communication must operate between each IMU sensor and the ESP32 microprocessor at a speed of at least 10 ± 1.0 MHz.
Verification
Equipment
Oscilloscope, frequency counter, LSM6DSMTR IMU sensor, ESP32 development board

Test Procedure
1. Connect the oscilloscope to the SPI clock pin of both the IMU sensor and ESP32. 2. Initialize IMU sensor to transmit data to ESP32. 3. Use the frequency counter to measure the frequency of the SPI clock pin. 4. If the frequency value is between 9 and 11 MHz, this test is successful. 5. Repeat steps 2-4 10 times to generate enough data.
Presentation of Results
Include a data table for the frequency readings in the notebook and final report that includes all 10 trials.

Table A.10: Data for SPI Communication

Trial #	Measured Frequency (MHz)	Pass/Fail
1	10.00	Pass
2	10.02	Pass
3	10.01	Pass
4	10.03	Pass
5	9.98	Pass
6	10.02	Pass
7	9.97	Pass
8	9.98	Pass
9	10.01	Pass
10	10.03	Pass

A.3 Sensing Subsystem

Table A.11: R&V for Sensor Accuracy

Requirement

Each IMU sensor should provide acceleration and angular velocity with an accuracy of ±5%
Verification
Equipment
LSM6DSMTR IMU sensor, ESP32 development board, flat surface
Test Procedure
Accelerometer: 1. Connect IMU sensor to ESP32 development board using manual. 2. Place the IMU sensor on a hard flat surface with the chip facing up. 3. Request accelerometer readings using the ESP32 microcontroller. 4. Record the provided accelerometer data. 5. Repeat steps 3 and 4 5 times to gather enough data. 6. Repeat steps 2 through 5 for each of the 2 remaining axes (see datasheet pg. 20 for acceleration axes relative to package) 7. Each value should be within ±5% of the desired value. The desired value should be 1.0 with the sign of the axis pointing straight down. For example, the correct value for testing in the +Z direction (chip facing up) is X = 0.0g, Y = 0.0g, Z = -1.0g. 8. If all readings are within ±5% of the desired value, the test is successful. **Gyroscope:** 1. Connect IMU sensor to ESP32 development board using manual. 2. Place the IMU sensor on the surface with the chip facing up. 3. Request gyroscope readings using the ESP32 microcontroller. 4. Record the provided gyroscope data. 5. Repeat steps 3 and 4 5 times to gather enough data. 6. Repeat steps 2 through 5 for each of the 2 remaining axes (see datasheet pg. 20 for gyroscope axes relative to package) 9. Each value should be within ±5% of the desired value, which should be 0 for all tests 10. If all readings are within ±5% of the desired value, the test is successful.
Presentation of Results
Record all readings in separate data tables in the notebook for the accelerometer and gyroscope, and take note whether each test passes or not.

Table A.12: Data for Sensor Acceleration Accuracy

Trial #	Measured Acceleration (Gs)	% Error	Pass/Fail

1 (X)	1.00	0%	Pass
2 (X)	0.99	1%	Pass
3 (X)	0.98	2%	Pass
4 (X)	1.00	0%	Pass
5 (X)	1.02	2%	Pass
6 (Y)	1.02	2%	Pass
7 (Y)	1.01	1%	Pass
8 (Y)	1.02	2%	Pass
9 (Y)	0.99	1%	Pass
10 (Y)	1.02	2%	Pass
11 (Z)	0.98	2%	Pass
12 (Z)	1.01	1%	Pass
13 (Z)	1.02	2%	Pass
14 (Z)	0.98	2%	Pass
15 (Z)	1.01	1%	Pass

Table A.13: Data for Sensor Angular Velocity Accuracy

Trial #	Measured Angular Velocity (dps)	% Error	Pass/Fail
1 (Pitch)	0.00	0%	Pass
2 (Pitch)	0.00	0%	Pass
3 (Pitch)	-0.02	0%	Pass
4 (Pitch)	0.00	0%	Pass
5 (Roll)	0.02	0%	Pass
6 (Roll)	-0.02	0%	Pass
7 (Roll)	0.02	0%	Pass

8 (Roll)	0.00	0%	Pass
9 (Roll)	0.00	0%	Pass
10 (Roll)	0.02	0%	Pass
11 (Yaw)	0.00	0%	Pass
12 (Yaw)	0.00	0%	Pass
13 (Yaw)	0.02	0%	Pass
14 (Yaw)	-0.02	0%	Pass
15 (Yaw)	0.00	0%	Pass

Table A.14: R&V for Sensor Frequency

Requirement
Each IMU sensor should provide data at a rate of at least 20 Hz for a total combined rate of 60 Hz
Verification
Equipment
Precision Dumbbell Assistant, Oscilloscope
Test Procedure
1. Connect two sensor boards to the main board as shown in the visual aid 2. Connect the oscilloscope to a GPIO pin on the ESP32 3. Write a test program that toggles the value of the GPIO pin after receiving data from each of the three sensors. 4. Check the frequency of the GPIO pin. If it is higher than 10 Hz, then the test is successful. 10 Hz is the target value because the pin is toggled on every read, so there will be two reads for every period of the GPIO signal.
Presentation of Results
Record the results of the test as a single value in the notebook and mark whether the test passed or not.

Table A.15: Data for Sensor Frequency

Trial #	Measured Frequency (Hz)	Actual Frequency (Hz)	Pass/Fail
1	49.00	98.00	Pass
2	49.30	98.60	Pass
3	48.57	97.14	Pass
4	49.41	98.82	Pass
5	48.62	97.24	Pass

A.4 Feedback Subsystem

Table A.16: R&V for Buzzer Pitch

Requirement
The buzzer must have a pitch between 200 Hz and 600 Hz with a tolerance of ± 10% and the sound must last at least 1 second.
Verification
Equipment
Buzzer, oscilloscope, function generator
Test Procedure
1. Connect the buzzer to the function generator using corresponding terminals. 2. Use the function generator to generate a square function with a frequency of 200 Hz. 3. Use the oscilloscope to record the frequency of the resulting sound. 4. Increase the frequency of the function by 100 Hz. 5. Repeat steps 3 and 4 until you reach 600 Hz. 6. If each recorded frequency is in the range of the input frequency ± 10%, this test is successful.
Presentation of Results
Include a data table recording the input and output frequencies in the notebook and final report. Also, plot a line graph with input frequencies on x axis and output frequencies on y axis.

Table A.17: Data for Buzzer Pitch

Input Frequency (Hz)	Output Frequency (Hz)	Error (%)	Pass/Fail
200	186	7.00	Pass
300	278	7.33	Pass
400	410	2.50	Pass
500	476	6.00	Pass
600	635	5.83	Pass

Table A.18: Line Graph for Buzzer Pitch

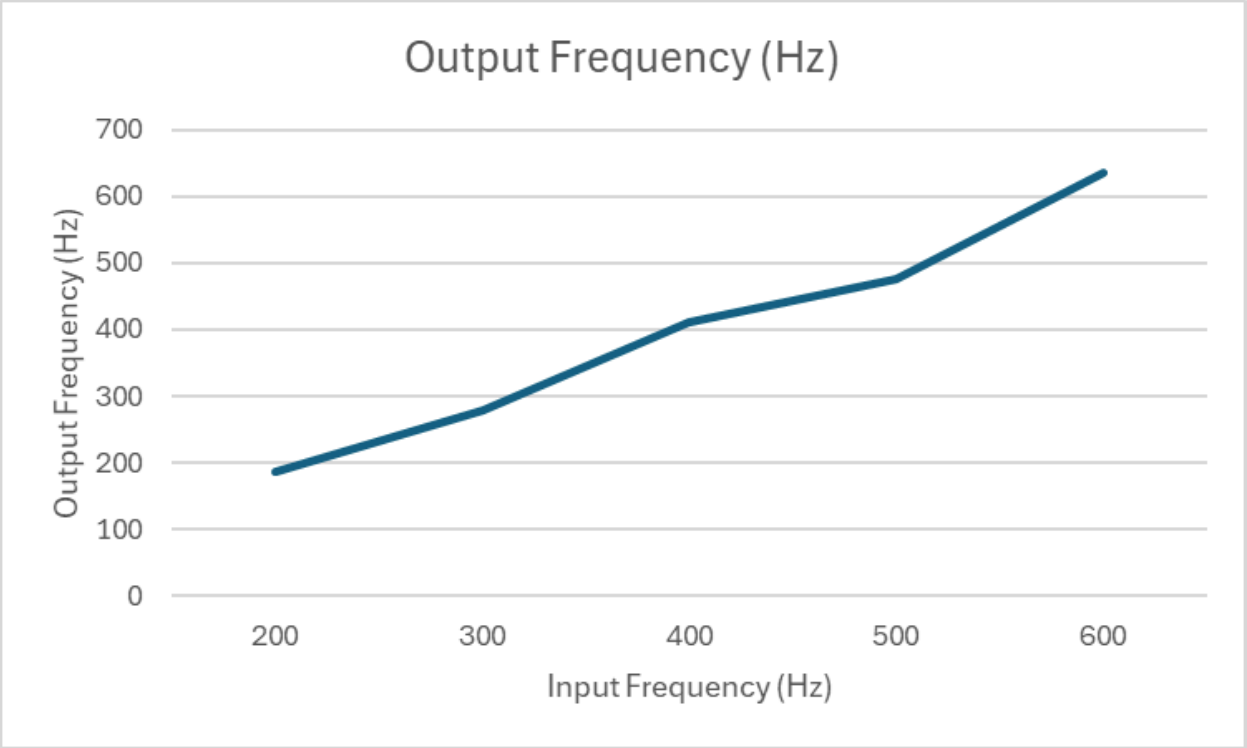

Table A.19: R&V for Buzzer Loudness

Requirement
The buzzer must have a loudness of at least 60 dB
Verification

Equipment
Buzzer, function generator, smartphone
Test Procedure
1. Connect the buzzer to the function generator using corresponding terminals. 2. Use the function generator to generate a square function with a frequency of 200 Hz. 3. Use a smartphone decibel measurement app to measure the loudness of the buzzer. 4. Increase the frequency of the function by 100 Hz. 5. Repeat steps 3 and 4 until you reach 600 Hz. 6. If each recorded frequency has a loudness above 60 dB, the test passes.
Presentation of Results
Include a data table recording the loudness for each frequency, and note whether the test passes for each in the notebook and final report.

Table A.20: Data for Buzzer Loudness

Frequency (Hz)	Loudness (dB)	Pass/Fail
200	64	Pass
300	62	Pass
400	68	Pass
500	65	Pass
600	67	Pass

A.5 Wireless Communication Subsystem

Table A.21: R&V for BLE Distance

Requirement
There must be a Bluetooth Low Energy connection between the ESP32 microcontroller and the user's device at a distance of at least 5 meters.
Verification

Equipment
ESP32 microcontroller, laptop, measuring tape
Test Procedure
1. Place the ESP32 microcontroller at a fixed location. 2. Initialize ESP32 with a program that sets up BLE.. 3. Use measuring tape to measure 1 meter distance from ESP32. 4. Use a laptop to connect to the ESP32 microcontroller through bluetooth. 5. Record if connection is successful. 6. Use measuring tape to move back 1 meter further. 7. Repeat steps 4-6 until connection is unsuccessful. 8. If the final distance is at least 5 meters, this test is successful.
Presentation of Results
Include a data table recording the distances and connection success or failure in the notebook and final report.

Table A.22: Data for BLE Distance

Trial #	Distance (m)	Pass/Fail
1	0.5	Pass
2	1.0	Pass
3	1.5	Pass
4	2.0	Pass
5	2.5	Pass
6	3.0	Pass
7	3.5	Pass
8	4.0	Pass
9	4.5	Pass
10	5.0	Pass

Table A.23: R&V for BLE Packet Loss

Requirement
The BLE connection must have packet loss of 20% or less at 5m.
Verification
Equipment
ESP32 microcontroller, smartphone, measuring tape
Test Procedure
1. Place the ESP32 microcontroller at a fixed location. 2. Initialize ESP32 with a program that sends BLE notifications to a smartphone located 5m away at a set interval for 5 minutes 3. Measure how many messages are received on the smartphone out of how many are sent, and calculate the percentage of messages that were received. 4. Repeat this test 4 times in varying conditions (inside, outside, out of sight)
Presentation of Results
Record the results of the test in a table and label which results go with which condition and mark whether the test passed or not. This table should be in both the notebook and final report

Table A.24: Data for BLE Packet Loss

Trial #	Sent	Received	Percent	Condition	Pass/Fail
1	300	300	100%	Inside	Pass
2	300	300	100%	Outside	Pass
3	300	300	100%	Out of Sight	Pass

A.6 Physical Design

Table A.25: R&V for Device Weight

Requirement
Each main board, battery, and enclosure should weigh less than 100g
Verification
Equipment
Main Board, Battery, Enclosure, Scale
Test Procedure
1. Place the assembled main board, battery and enclosure on the scale and record the value. 2. If the assembly weighs less than 100g, the test is successful.
Presentation of Results
Record the results of the test as a single value in the notebook and mark whether the test passed or not.

Table A.26: Data for Device Weight

Weight(g)
84

Table A.27: R&V for Sensor Board Weight

Requirement
Each sensor board and accompanying wire harness should weigh less than 50g
Verification
Equipment
Sensor Board and Wire Harness, Scale
Test Procedure
1. Place the sensor board and wire harness on the scale, and record the value. 2. If the board and harness weigh less than 50g, the test is successful.

Requirement
Each sensor board and accompanying wire harness should weigh less than 50g
Verification
Equipment
Presentation of Results
Record the results of the test as a single value in the notebook and mark whether the test passed or not.

Table A.28: Data for Sensor Board Weight

Weight(g)
24