Athletic Tracking Sensor Design Document

By

J.D. Armedilla

Ryan Horstman

Ethan Pizarro

Design Document for ECE 445, Senior Design Spring 2025

TA: Jiankun Yang

February 15, 2025

Project No. 38

Introduction

Problem

Weightlifting is a key method of staying active and maintaining fitness for many people. Currently, the main metric of progress in weightlifting for everyday weightlifters is varying the heaviness of the weight and the number of repetitions. For example, once could raise up 10 pounds (weight) 10 times in a row (repetitions). But, there is also value in (and workouts designed around) moving the weight at an appropriate speed, known as Velocity-Based Training (VBT) [1]. Continuing our example, instead of just counting how many times the weight is lifted, one tries to raise the weight slowly, never exceeding two meters per second. Effective velocity-based training emphasizes lifting heavy, while simultaneously ensuring one handles the weight well and with the speed desired.

However, velocity-based training is not accessible because available sensors for tracking velocity are costly and therefore infeasible for everyday weightlifters. If a lifter does buy an expensive sensor, those available lack some key features for an optimal workout. For example, some sensors assist with tracking velocity but do not assist with tracking form. Also, current sensors offer feedback that consists of the lifter doing their exercise and then checking their results on their phones. Since this feedback is not necessarily “real-time”, these sensors cannot indicate to a user to adjust their velocity between each repetition. This results in wasted repetitions; and, for form tracking sensors, users are not informed immediately if bad form is used during the workout, increasing the risk of injury [2-3].

Solution

We propose a compact wearable device that takes and transmits workout data to a phone via Bluetooth. It will utilize a 9-axis sensor (acceleration, gyroscope, and magnetometer). However, in addition to sending data to a phone, it will internally process data taken during the workout and provide immediate feedback to the user through haptic signaling and LED feedback. Before starting the workout, the user can indicate on his/her phone the workout they are performing and any desired constraints. This data will be sent to the device. The device will track the user's form and acceleration, alerting him/her if a desired constraint is not being met so that it can be immediately corrected mid-set. It would be small enough to velcro around a user’s wrist, hang on a necklace worn by the user, velcro-strap around a weight set, or attach it to a desired object. Using the acceleration data given by the tracking sensor, the app will display the collected velocity graphically and the lifting form so that the user can visually see his/her progress.

Visual Aid

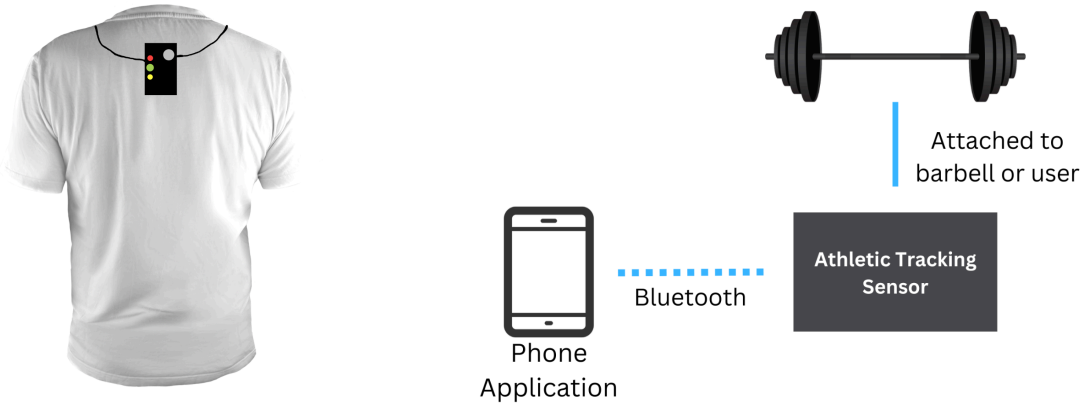

Figure 1: Usage of athletic tracking sensor.

High-Level Requirements

Our athletic tracking sensor will need to meet the following requirements:

- IMU readings on-chip will be able to determine velocity in the vertical direction with precision in the hundredths of a meter per second for bench press and back squat workouts. The device will alert the user immediately when a goal velocity is met/exceeded.
- The athletic sensor must be able to send and receive data via Bluetooth to/from our developed iPhone app, which will then use that data to graphically represent the athlete's velocity at a certain load.
- For form applications, our gyroscope will be able to detect degrees in a given direction at the precision of 3 degrees and alert the user within a half second of entering bad form.

Testing

To test our device, we will be simulating a weightlifting session in an enclose area. To ensure safety of the testers, we will not be using a barbell nor weighted plates. Instead, we will be using a PVC pipe that is similar in length to an actual barbell. We will test and verify operation of our device with the following two workouts:

Back Squat:

Back squat is the movement in which weight is placed across the user's shoulders, often a long metal bar called a barbell. The user then bends their legs at the knees and lowers their body until their thighs are parallel with the ground (squatting). Then, the user stands back up. Squat would require one sensor module on the upper back of the user. For the acceleration tracking aspect, the vertical velocity of the user would be gained (using the magnetometer data to isolate the vertical component of acceleration data). In addition to tracking and sending the data to the app, the user will input goals such as not wanting to pass a given velocity (in which case the haptics will actuate if the goal is not met), or wanting to exceed a certain velocity (in which case the haptics will actuate if the goal is met). For form tracking, the dangerous aspect of squat is back form. Often, lifters will lean too far forward at the bottom of their rep, putting themselves at risk. The microcontroller will analyze gyroscope readings to ensure the user's back stays within a safe degree from vertical.

Bench Press:

Bench Press is an exercise in which the user lays down on their back, and holds a barbell above their chest. They then lower the bar to tap their chest, and then push the weight back upwards. For bench press, we would have one sensor strapped to the center of the barbell or two on either end of the barbell. Acceleration tracking would be very similar to squat: isolating the vertical direction and then implementing speed goals. For form, we would additionally isolate lateral movement and haptically warn if dangerous amounts of movement occur (forward and backward, for example). Additionally, in the two-sensor mode, the levelness of the bar will be tracked to ensure one side is not higher/lower than the other.

Design

Block Diagram

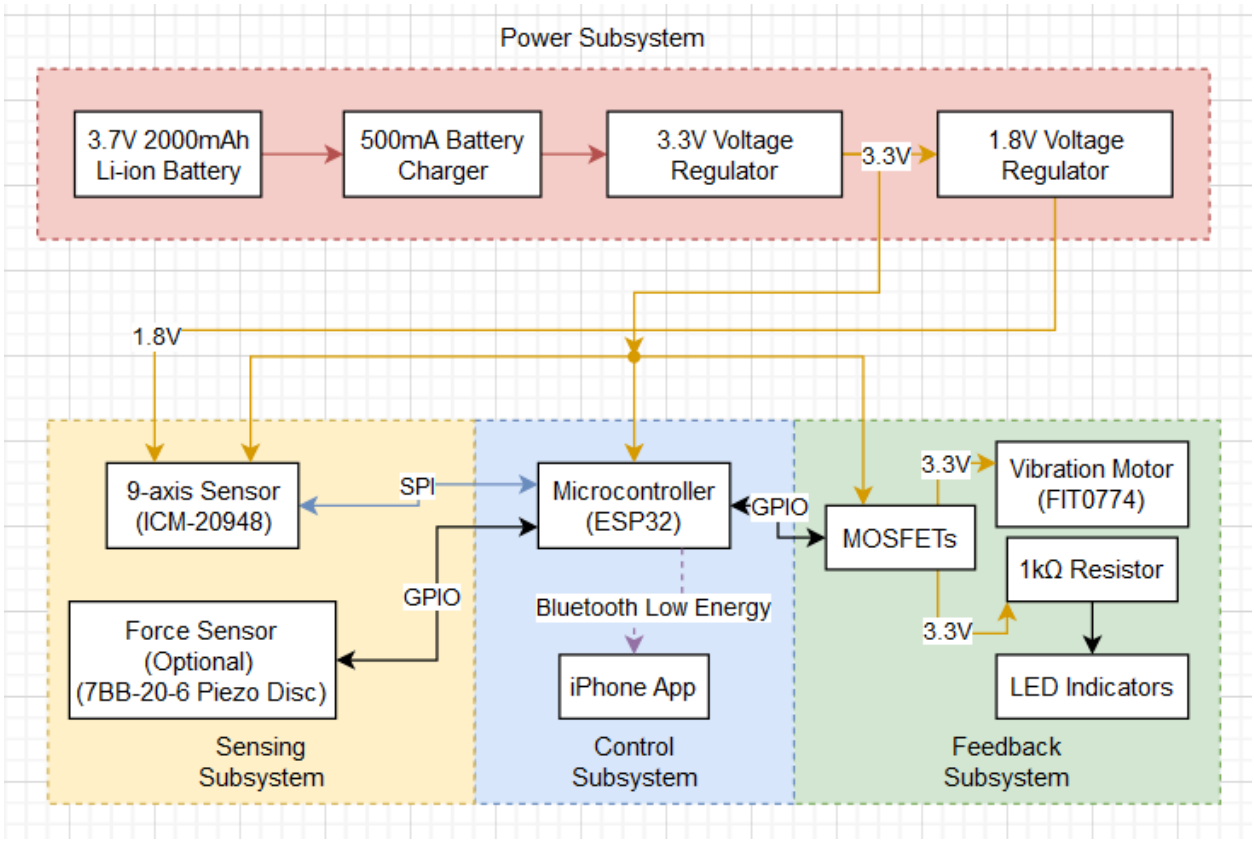

Figure 2: Block diagram for Athletic Tracking Sensor

Our block diagram consists of 4 different subsystems: power, sensing, control, and feedback. The power subsystem converts 3.7V battery power to 3.3V/1.8V regulated power and handles battery charging. The sensing subsystem consists of a 9-axis sensor and an optional force sensor to collect data on the position of the tracking sensor and how much force is being applied during an exercise. The control subsystem handles programming of the microcontroller and utilizes a mobile app to display collected data and set parameters for success and failure. The feedback subsystem provides haptic and visual feedback via a vibration motor and LEDs to alert the user of deviations from correct form as soon as possible.

Physical Design

The Athletic Tracking Sensor will be enclosed in a 3D-printed rectangular enclosure similar to the one below. The use of plastic ensures that signal disruptions are a lot less likely, especially since we are using Bluetooth connectivity. There will also be holes added to the enclosure to ensure proper airflow.

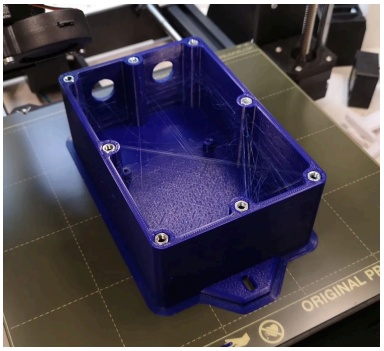

Figure 3: 3D-printed PCB enclosure example

Power Subsystem

Hardware Design Overview

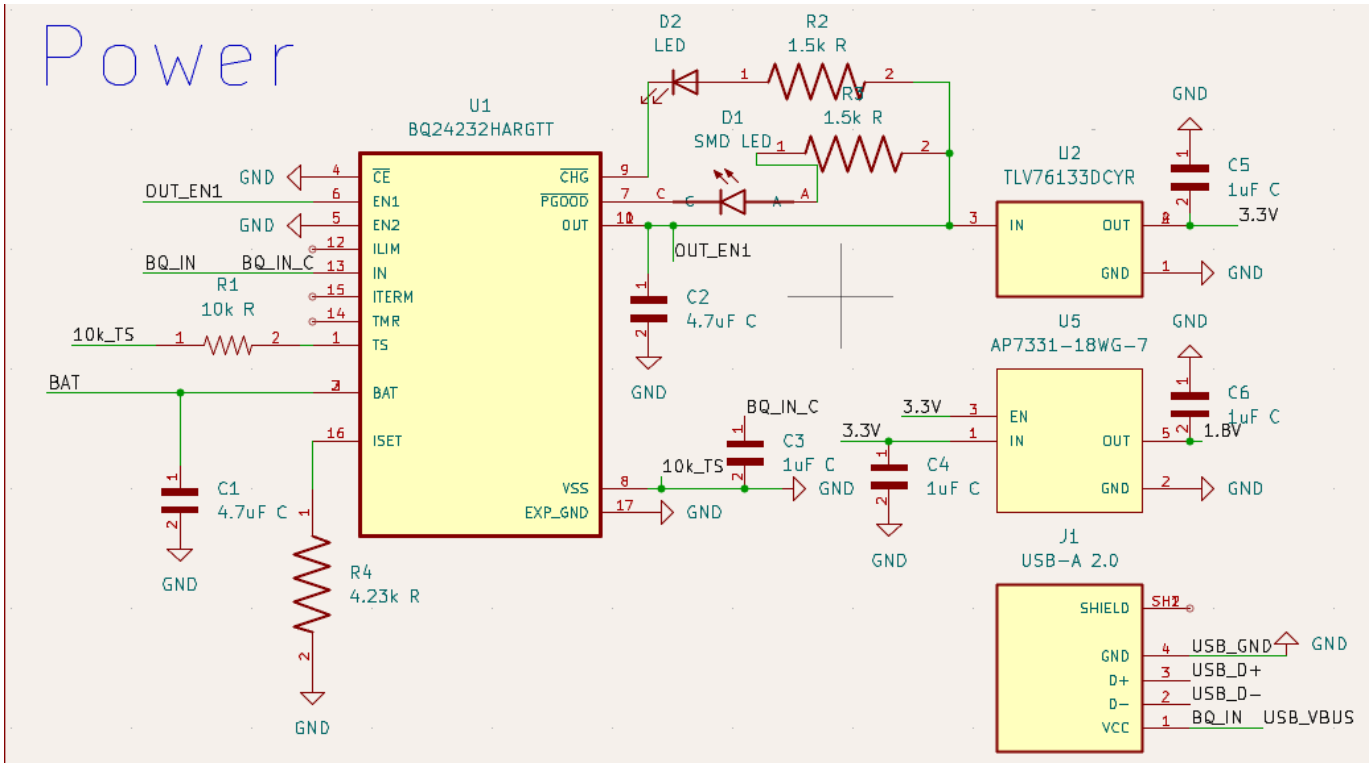

Figure 4: Power Subsystem Schematic Design

The power subsystem utilizes a 3.7V 2000mAh battery as its main power source. After being connected with a battery charging IC, the voltage is stepped down to 3.3V and 1.8V via two consecutive linear voltage regulators. This reduces the amount of heat created, mitigating any risks of injuring the user via heat or melting the plastic enclosure. The battery charger allows the user to recharge the device as much as needed. The power subsystem ensures stable power supply to all other components and ensures the device is capable of recharging.

Functionality & Contribution

The main goal of the power subsystem is to convert the Adafruit 3.7V 2000mAh battery's voltage to more stable supply voltages for our devices (3.3V and 1.8V) and enable recharging the device for its users.. The following components contribute to this goal:

- BQ24232HARGTT Li-ion charger: Enables charging, including an LED to indicate when said charging is occurring.
- TLV76133DCYR linear voltage regulator: Regulates the battery's 3.7V down to 3.3V for use on most of our devices
- AP7331-18WG-7 linear voltage regulator: Regulates the 3.3V output of the previous component down to 1.8V for the 9-axis sensor's I/O voltage.
- UP2-AH-1-TH USB-A 2.0 connector: Processes USB charging and allows data transfer to the control sensor.

The power subsystem converts the battery's higher voltage to lower voltages that our components and ICs can reliably use. The addition of a battery charger increases convenience for the user while allowing them to understand when their device is charging.

Interfaces

Inputs:

- Adafruit 3.7V 2000mAh Li-ion battery
    - Voltage range: 3.0 (discharged) to 4.2 (full charge)
    - Input to battery charger: 3.7V
- 3.3V output from TLV76133DCYR: input to AP7331-18WG-7 to regulate down to 1.8V

Outputs:

- 3.3V output from TLV76133DCYR: powers the ESP32-S3R8, USB-Serial converter, 9-axis sensor, and various throughhole components of the feedback subsystem
- 1.8V output from AP7331-18WG-7: powers the 9-axis sensor's I/O voltage

- Charging LED: indicates whether the battery is being charged or not at any given time

Requirements & Verification

Requirements	Verification
● The power subsystem must recharge the battery within 6 hours.	● Discharge the battery until its voltage is equivalent to 3.0V, or its discharged voltage ● Plug the device into a USB power source ● Check the voltage of the battery every hour until it reaches 4.2V, or its voltage when fully charged ● Repeat at least once to ensure consistency
● The power subsystem must supply continuous power to the rest of the system for at least 2 hours from a fully charged battery.	● Fully charge the battery and begin operating the athletic tracking sensor, including gathering measurements to be sent to the mobile app ● Utilize a stopwatch to track how long the device stays operational ● Ensure the operational time the device has under one charge exceeds 2 hours ● Repeat at least once to ensure consistency
● The power subsystem must not be overloaded by the device's components, failing to supply enough current for it to operate	● Fully charge the battery and begin operating the athletic tracking sensor ● Verify that the correct voltage is being outputted by both regulators in the subsystem ● Use a multimeter to measure total current consumption during both exercises used in testing ● Verify that current draw does not exceed 1 A at any point during operation

	• Repeat at least once to ensure consistency

Figure 5: Power Subsystem Requirements & Verifications

Design Decisions

We decided to use two linear voltage regulators chained together. By connecting our 3.3V regulator into the 1.8V regulator, it minimizes the amount of heat generated from said conversions. Because wearable devices and small form factors in general are susceptible to overheating (especially with a plastic enclosure), it is crucial we reduce our heat output as much as possible.

Battery charging is included so that the device can be reused regularly. While we do have to consider overheating and overcharging issues, it is important that users are able to charge the device overnight before they work out. Because some users may exercise up to 6 times a week, the athletic training sensor must be readily available on a daily basis.

Control Subsystem

Hardware Design Overview

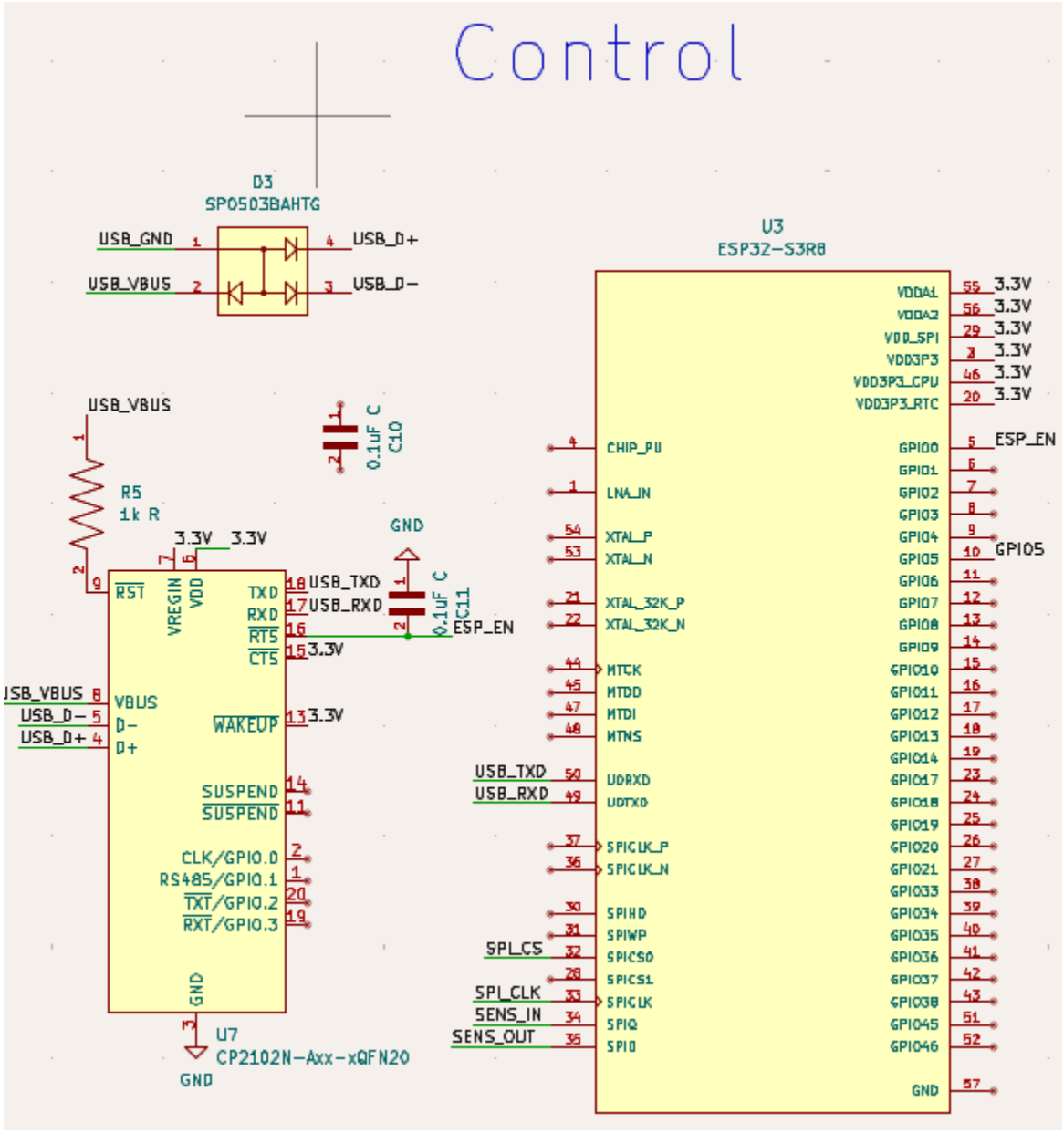

Figure 6: Control Subsystem Schematic Design

The control subsystem consists of our microcontroller (the ESP32-S3R8) and a USB to serial data converter in order to program it. As the main computational component of our device, the ESP32-S3R8 will be responsible for processing the data from our 9-axis sensor via SPI. Utilizing SPI allows for faster data transfer, which will allow our device to experience less latency when it

comes to alerting users of bad form. The embedded Bluetooth module will be used to communicate with our mobile phones so that we can display the data collected and set the parameters the ESP32-S3R8 should trigger at.

Functionality & Contribution

Operation of the athletic tracking sensor will be initiated via a “Record Data” button. This signals to the device that we are starting a workout and need to start measuring data. This also ensures a boost in efficiency, as we will not always have our device operating at full power when it does not need to do so.

Once operational, the control subsystem handles all of the data processing. The 9-axis sensor will send its data via SPI to the ESP32-S3R8. Once the sensor’s data has been sent over, velocity and acceleration will be calculated, which will be sent over Bluetooth to our mobile app for display. Computational power should not be an issue, as we are mostly just checking if measured values from the sensor exceed or fail to meet a certain threshold, as decided by the parameters set in the mobile app.

When those thresholds are exceeded, the control subsystem will also be responsible for initiating the feedback system. This includes turning on vibration motors or associated LEDs via the GPIO pins.

Interfaces

Inputs:

- 9-axis sensor data via SPI
- The 3.3V power source from the power subsystem
- Programs being loaded to the ESP32-S3R8 via the USB to serial converter

Outputs:

- Bluetooth Low Energy protocol sends velocity and acceleration data to the device hosting the mobile app
- GPIO signals to activate the feedback subsystem when relevant

Requirements & Verification

Requirements	**Verification**

• When the “Record Data” button is pressed, the control subsystem should determine if the user is hitting the necessary threshold for the velocity the user inputted.	• Have the user input a desirable and reasonable velocity into the mobile app. Observe if the MCU stores this value into its memory. • Have the user be in a proper position and press the button on the device • The user should then try to go way below the threshold. Observe if the boolean value for hitting the threshold is False. • The user should then be within the proper threshold. Observe if the boolean value for hitting the threshold is True. • The user should then be way over the threshold. Observe if the boolean value for hitting the threshold is False. • Alternate between these three conditions, observing the boolean value is in the proper state.
• When the “Record Data” button is pressed, the control subsystem should determine if the user is using the proper form.	• Have the user input the desired exercise as the weighted squat into the mobile app. Observe if the MCU stores this information in its memory. • Have the user place the device in the proper location on the back and press the button on the device, indicating the start of the workout • Observe if the MCU records the initial position of the user, and that the thresholds for proper form at certain locations are implemented. • Have the user be within the threshold. Observe if the boolean indicating if the user is using the proper form is True. • Have the user be outside of the threshold. Observe if the boolean indicating if the using proper form is

	False. Have the user re-enter the thresholds. Observe if the boolean is True. ● Have the user press the button on the device, indicating the end of the workout. ● Have the user input the desired exercise as the bench press. Observe if the MCU stores this information in its memory. ● Have the user place the device on the center of the barbell and press the button on the device, indicating the start of the workout. ● Observe if the MCU records the initial position of the user, and that the thresholds for proper form at certain locations are implemented. ● Have the user be within the threshold. Observe if the boolean indicating if the user is using the proper form is True. ● Have the user be outside of the threshold. Observe if the boolean indicating if the using proper form is False. Have the user re-enter the thresholds. Observe if the boolean is True. ● Have the user press the button on the device, indicating the end of the workout.

Figure 7: Control Subsystem Requirements & Verification

Design Decisions

Utilizing a button to initiate and then end workouts allows us to make our device more efficient. It is not necessary that data is transmitted when no exercises are currently being done. By utilizing the button to start an exercise, it allows the device to have certain components enter sleep mode. By minimizing the amount of time that components are running at full power, it ensures the device's battery lasts longer.

Additionally, utilizing the USB-A 2.0 connector for both charging and data purposes allows the PCB to be smaller. Having two connectors would complicate the design unnecessarily, as there would be two separate USB-A ports, with one handling charging the battery and the other handling microcontroller programming. By combining both purposes into one connector, it simplifies how the port should be used by users and prevents a waste of resources and space by having two of the same connector.

Sensing Subsystem

Hardware Design Overview

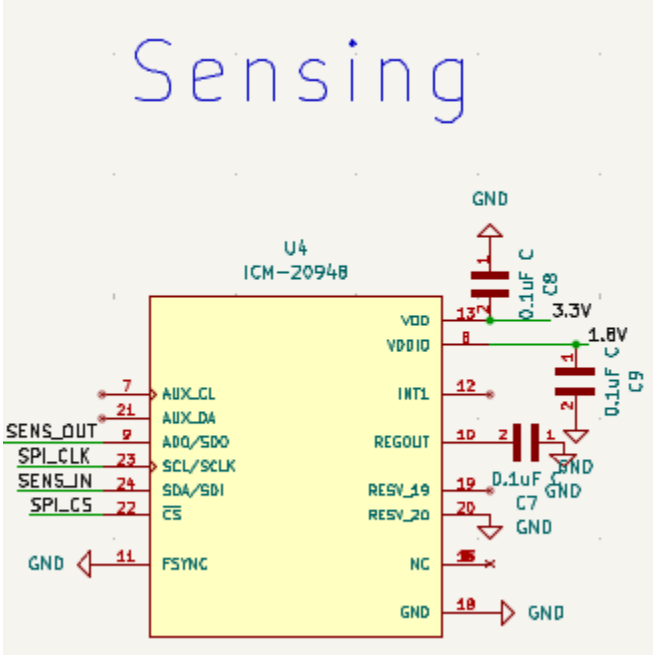

Figure 8: Control Subsystem Schematic Design

Our sensing subsystem currently consists of the ICM-20948, a 9-axis sensor composed of an accelerometer, a magnetometer, and a gyroscope. The sensing subsystem must send gyroscope data to the ESP32 often enough to meet our 0.5-second output responsiveness requirement for form detection, and spend the rest of its operation time collecting acceleration data. We can validate the sensor data by observing it once it is transmitted to the iPhone app. We can compare with video to ensure the data is representative of what is happening.

Functionality & Contribution

The main goal of the sensing subsystem is to provide real-time data of the device's acceleration and position. The following components contribute to this goal:

- ICM-20948 9-axis Sensor: The 9-axis sensor used to track the position/velocity of the user.

Upon powering on, the microcontroller will interface with the ICM through SPI to initialize the registers required for operation, ensuring proper data delivery speed and formats. Once the workout has begun (indicated by the user actuating the push button) The microcontroller continually takes data readings from the sensor. Due to the nature of the ICM, only one sensor can be read from at a time. This issue is talked through in depth in the 'Tolerance Analysis' section below. For a high level overview, the microcontroller will switch between the three sensors at a frequency appropriate to achieving our goals.

The cycle of data readings will meet the following requirements:

- Ten readings per second from the gyroscope
- Three readings from the magnetometer per second
- Rest of readings from accelerometer

Once we have our data streaming in, the data analysis will go as follows. First the magnetometer will analyze the three magnetic data vectors to determine which direction is vertical. Then acceleration readings will be made. The acceleration data will be pre-processed by removing the roughly 9.8m/s^2 due to gravity, and combining the remaining 3 vectors to achieve a total acceleration vector, shown in equation 1, where each component is an acceleration vector.

$$\mathbf{A\_total = A\_x + A\_y + A\_z - A\_gravity} \qquad \textbf{(2}$$

After preprocessing, the acceleration data will be continually integrated to determine velocity. Acceleration data will continually be collected until a gyroscope reading is requested or it is time to recalibrate the vertical reading from the magnetometer.

Interfaces

Inputs:

- The microcontroller will set up the sensor through SPI protocol, involving sending register initialization data, and sensor select
- During sensing, the ESP32 will indicate when it is ready for the next data acquisition

Outputs:

- Readings from the magnetometer, accelerometer, and gyroscope

Requirements & Verification

Requirements	Verification
● The sensing subsystem must start recording once the ESP32 sends the proper signal.	● Have the ESP32 send the necessary signal that initializes the sensor and tells it to start recording. ● Observe if the sensing subsystem starts sending data to the MCU.
● The sensing subsystem must collect gyroscope data often enough to alert the user of bad form usage within 0.5±0.2 seconds.	● Have the sensor record data. ● Ensure that the data includes a gyroscope reading at least three times per second.

Figure 9: Sensing Subsystem Requirements & Verifications

Design Decisions

We chose the IMC sensor, which combines the three sensors we require into one, instead of three separate sensors in order to save space. Due to the nature of our device as a wearable, we want it to be as small as possible to enhance user experience. The optional addition of a force sensor attachment allows our device to become more versatile and applicable to a larger variety of workouts, though a focus on acceleration tracking allows prioritization of our main goals in a time-constrained project, and that will be our focus initially.

Feedback Subsystem

Hardware Design Overview

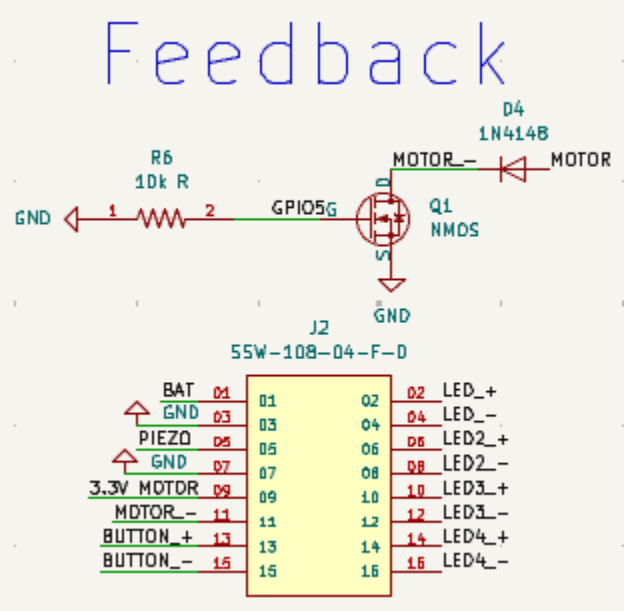

Figure 10: Feedback Subsystem Schematic Design

This handles haptic and visual feedback when exercise requirements are not being met, as well as basic status feedback of the device in general. The vibration motor will receive power through a MOSFET. Thus, when the GPIO is high, it will raise voltage at the gate of the MOSFET and allow power to go to the haptic motor. This output GPIO pin will also be directly routed to the yellow LED to indicate visually as well as haptically.

Additionally, we include three other LEDs to indicate to the user the state of the device.

The feedback subsystem must be able to provide haptic and visual feedback while an exercise is occurring, contributing to our real-time feedback requirement. In the demo, we will intentionally not meet our input goals and demonstrate bad form (without heavyweight) to feedback system works.

Functionality & Contribution

The main goal of the feedback subsystem is to provide real-time feedback to the user, alerting the user if the user is not hitting certain constraints. The following components contribute to this goal:

- FIT0774 Vibration Motor: The motor should activate once the user enters bad form/not hitting a certain velocity requirement, and will deactivate once the user corrects themselves.
- LEDs: These LEDS present the current state of the environment. There are three different colored LEDs on the device:
    - Red LED: conveys alerts to the user, either of bad form (through blinks) or of a goal being met/broken (through a sustained 'ON' period)
    - Yellow LED 1: indicates that the battery is low
    - Yellow LED 2: indicates the device is waiting for the user to start the workout
    - Green LED: indicates that the device is turned on

Interfaces

Inputs:

- Indication from the microcontroller that the user needs to be alerted, either through a pulsing signal (repeated on and offs) indicating dangerous form measurements, or a constant 'ON' for a set amount of time, indicating a goal is met/not met depending on the exercise.
- Logical '1' or '0' indicating if the device is on
- Logical '1' or '0' indicating if the device's battery is charged
- Blinking signal to indicate if the device is waiting for the user to start his/her workout, or solid logical '1' if workout is in progress

Outputs:

- Visual alerts through a red LED
- Haptic alerts through a vibration motor
- Visual indications through the yellow and green LEDs conveying the current state of the device as a whole.

Requirements & Verification

Requirements	Verification
● The feedback subsystem's LEDs should be able to indicate the proper state of the device.	● Turn on the device. Observe if the green LED turns on. Turn off the device. Observe if the green LED turns off. ● Place the MCU in a state where it

	waits for user input. Observe if the respective yellow LED is blinking. ● Press the button on the device. Observe if the yellow LED is solid. ● Place the MCU into data recording mode. Have it record an initial position and develop the thresholds for good form. Have the user stay within these thresholds. Observe if the red LED is off. ● Have the user go outside of the thresholds. Observe if the red LED turns on within (a certain time). ● Have the user re-enter the thresholds. Observe if the red LED is off within (a certain time).
● The feedback system should activate/deactivate the vibration motors	● Place the MCU into data recording mode. Have it record an initial position and develop the thresholds for good form. Have the user stay within these thresholds. Observe if the motor is off. ● Have the user go outside the thresholds. Observe if the motor turns on within (a certain time). ● Have the user re-enter the thresholds. Observe if the red LED is off within (a certain time).

Figure 11: Feedback Subsystem Requirements & Verifications

Design Decisions

Our decision to run the haptic motor's power through a MOSFET is due to the fact that the ESP output voltage is not sufficient to actuate the motor. Therefore the MOSFET receives the signal from the ESP to actuate, but the physical motor is ran by the 3.3V. Additionally, we had to include a pull down resistor so that if the GPIO pin connected to the MOSFET is not actuated, the gate of the MOSFET sees a logic '0' instead of indeterminate.

The rest of the LEDs are included to give a more user-friendly experience. The user can clearly see if the device is on, if the device is waiting to start, and battery status.

Software Design

The mobile app communicates with the control subsystem via its Bluetooth module. This app will be developed for Apple iOS. Mainly, the application will take velocity data and the user's weightlifting form data from the control subsystems and process that data to show the velocity at which the user is lifting a certain load. Other possible features that could be added include a simple rep counter, rest tracking, or other exercise-related quality-of-life enhancements.

The user will interact with the app's user interface to input the desired constraints for the workout. The app's screen will change depending on the state of the device.

The application will have the following states:

- ENTER_INFO: In this state, the user will be able to input the desired velocity and the desired workout. The app will enter the next state once the user hits a button on the screen, labeled "Send Data".
- DATA_SEND: In this state, the app will communicate with the MCU, sending over the user-inputted data. While in this state, the app will indicate that the MCU and the app are communicating with each other. It will wait for the MCU to send a signal back to the app confirming that it received the data. If that signal is received, the app will enter the CONFIRM state. If the signal is not received after a certain amount of time, the app will enter a TIME_OUT state.
- TIME_OUT: This state is reached if the MCU, after a certain amount of time, does not send a signal back to the app confirming that it received the data from the application. It will be able to return to DATA_SEND by pressing a button labeled "Send Again".
- CONFIRM: The app will indicate that the MCU has received the necessary data, and will prompt the user to start the workout, and press the button once they are ready to start recording data. Once the button is pressed, the app will enter the CONFIRM state.
- RECORD: The app will receive velocity data from the MCU. The app will enter the COMPILE state once the button on the device is pressed again.
- COMPILE: In this state, the data received from the MCU is compiled into a velocity graph, showing the velocity of the user over time. This graph will also present the areas where the user had good/bad velocity/form. The app will enter the ENTER_INFO state once the "Restart Workout" button on the app is pressed.
- CONNECTION_FAILED: If at any point, the app loses connection with the MCU outside of the COMPILE, TIMEOUT, and ENTER_INFO states, this screen will appear on the application, indicating that the app lost connection. The user can press a button labeled "Restart Workout", taking the app back to the ENTER_INFO state.

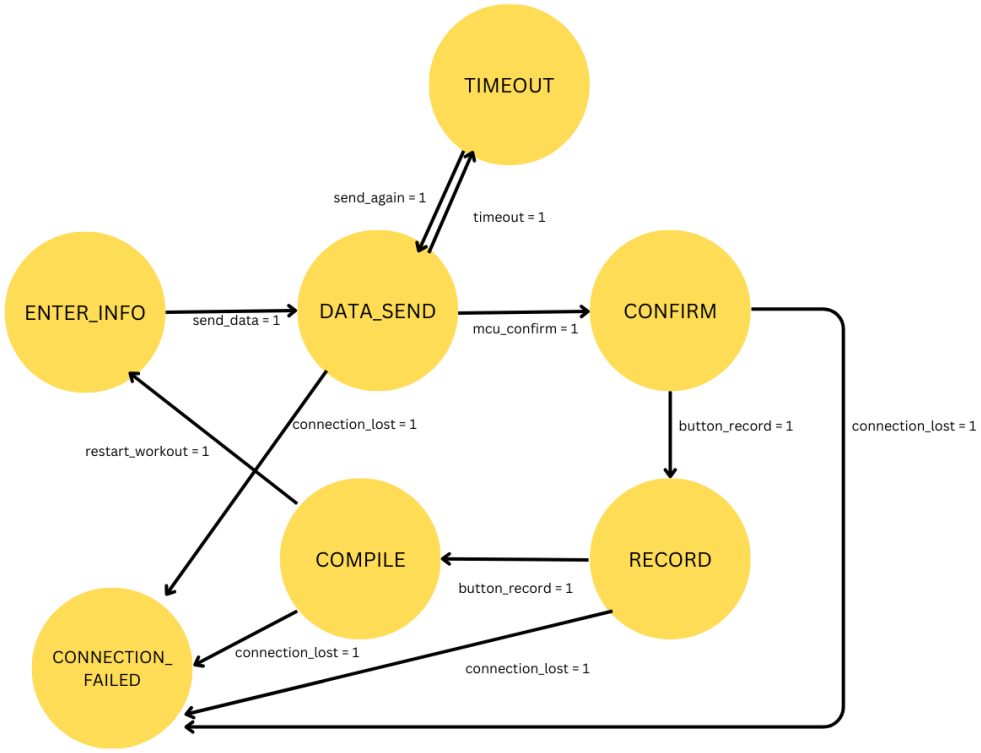

Figure 12: App state diagram for Athletic Tracking Sensor.

Tolerance Analysis

Successful completion of the athletic tracking sensor hinges on the data collection from the 9-axis sensor and the processing speed of the ESP32. We must be able to collect and process data quickly enough that the device can alert the user in real-time, otherwise, the user is not getting immediate feedback. Because of the nature of the ICM-20948, there are three sensors on board and only one can be read from at a time. The maximum low noise output frequencies are 1.125kHz for the gyroscope and accelerometer and 100 Hertz for the magnetometer. We will have to alternate between the on-chip sensors with priority to acceleration since that data is integrated. Our goal will be to maximize velocity measurements while getting at least 10 gyroscope readings per second. The magnetometer readings will just be used to recalibrate the vertical axis, so it will be much less frequently read, 3 times per second. With a goal of 100 samples per second for velocity readings we spend .118 of each second sampling data, calculated in equation 2. Switching between the sensors will also take time, 20 ms according to the datasheet for the accelerometer, the rest are not given so we will assume similar. Switching 23 times per second brings that number to .46 seconds of switching per second. Combined with our sample goal we are using .578 of each second, as seen in equation 3. This proves we will have a buffer for adjustments and non-idealities in our design. For the processor, it runs at a maximum

of 240MHz with a dual processor. This should be enough to maintain sensor readings, continuously integrate acceleration, maintain Bluetooth transmission, and internally monitor data.

$$(3*1/100 + 10*1/1{,}125 + 100/1{,}125) = .118 \text{ seconds} \quad (2$$

$$\text{Switching time} + \text{sampling time} = .118s + .46s = .578 \text{ seconds} \quad (3$$

Cost Analysis

Labor

Assuming that an average ECE graduate makes around $37.50/hr [4], and each member will work around 80 hours for the course of this project, we estimate that labor costs for one person will be:

$$labor = 37.50 * 2.5 * 80 = 7500$$

Parts

Adafruit 3.7V 2000mAh Lithium Ion Battery	1	$13.99
BQ24232HARGTT Battery Charger	1	$2.77
TLV76133DCYR Voltage Regulator	1	$0.41
CPF0603F10KC1 10k ohm resistor	1	$0.10
597-3305-607F SMD LED	1	$0.81
ERA-6AED152V 1.5k ohm resistor	2	$0.26
UP2-AH-1-TH USB-A 2.0 Connector	1	$0.59
ERA-6AEB4321V 4.32k ohm resistor	1	$0.10
GRT188R61C475KE13J 4.7uF capacitor	2	$0.19
GRM033D70J105ME01D 1uF capacitor	4	$0.16
AP7331-18WG-7 1.8V 300mA regulator	1	$0.66
ESP32-S3R8	1	$2.28
1825910-6 Push Button	2	$0.12
CP2102N-A02-GQFN20R Usb-Serial Converter	2	$5.59
ICM-20948 9-axis Sensor	1	$7.11
GRM155R71A104KA01D 0.1uF capacitor	3	$0.10
CEB-20FD64	1	$1.24
FIT0774 Vibration Motor	1	$0.99
DMN1019USN-13 MOSFETs	1	$0.57
SSW-108-04-F-D 16-pin connector	1	$2.80

Figure 13: Cost of Parts Table

This brings parts costs before tax to $47.68. Sales tax is 6.25%. This brings parts cost to $50.66.

Total Cost

Overall, the total cost will be labor * 3 + cost of parts. This will bring total cost to $22,550.66.

Schedule

Week	Ryan	Ethan	JD
March 2nd - 8th	Sensor and Feedback Subsystem in depth, Part ordering	Block Diagram, Power, Control, Ethics, Finish Scematic	Cost Analysis, R&V, in depth app
March 9th - 15th	Breadboard Demo Prep (Circuitry)	Breadboard Demo Prep (Coding/Microcontroller)	PCB routing/finalization, PCB order sent
March 16th - 22nd	Spring Break		
March 23rd - 29th	Interface with sensor (getting data) from PCB	Microcontroller w/ Feedback System on PCB + Misc. Programming	Have program for generating velocity graph made (test with mock data)
March 30th - April 5th	Code written for data interpretation	Bluetooth Functionality Working	App built minus bluetooth
April 6th - 12th	Integration		
April 13th - 19th	finalizing design	finalize design	finalize design
April 20th - 26th	Mock Demo		
April 27th - May 3rd	Writing Final Paper/ Prepping Presentation		

Figure 14: Schedule for Athletic Tracking Sensor

Ethics and Safety

Due to our use of Bluetooth, we need to ensure that users have their exercise data secured, as per the IEEE Code of Ethics #1. Practically, this means implementing a form of encryption to protect against anyone who would want to intercept the data being sent. While the athletic tracking sensor does not explicitly require personal data, it is still imperative that we protect users' privacy in any way possible. The Bluetooth controller on the ESP32 supports LE Privacy 1.2. This feature ensures that the message authentication code (MAC) address associated with the ESP32 will be randomized at certain intervals [5]. Any malicious device that wishes to intercept the data sent by the tracker will be unable to determine the true MAC address of the tracker. Apple devices also support address randomization [6]. Thus, the two devices will be able to connect to each other while still being able to ensure data privacy from malicious devices. To alleviate user concerns about the app, we will inform users of the mobile app of the data that is being collected and being sent to the device.

Because our device is wearable and will be used while performing athletic exercises, safety precautions must be taken to ensure no one is injured during testing, as per IEEE Code of Ethics #9 [7]. Since we are testing with heavy lifts, including squats and the bench press, we need to allocate adequate space for those lifts to be performed. This is especially important as testing will occur in UIUC's recreation centers available to all students, so others will be around us constantly. Additionally, we need to make sure the wearable design does not inhibit movement or cause injury, especially due to heat. Ensuring that our design does not hurt the wearer is of utmost importance, as a device that enhances exercise should not prevent the user from doing so. We will address this by keeping the device compact and ensuring the location of the device will not interfere with workout movements. Generation of heat will also be minimized amongst our components, with airflow holes being implemented into the PCB's enclosure to reduce the risk of any injuries happening due to heat.

The project itself fits #1 and #2 in the IEEE Code of Ethics [7]. The tracking sensor aims to enhance athletic training for anyone, supporting their well-being and promoting exercise for all. It also introduces users to the idea that their timing while exercising is important. Different purposes of training offer different benefits, so gaining direct knowledge of how the user's workout is going is important for both beginners and advanced lifters.

Citations

[1] O. Walker, “Velocity-Based Training.” scienceforsport.com. https://www.scienceforsport.com/velocity-based-training/ (accessed Feb. 10, 2025).
[2] GymAware, “GymAware RS (LPT)”, gymaware.com. https://gymaware.com/gymaware-rs/ (accessed Feb. 11, 2025)
[3] GymAware, “FLEX - Barbell Tracker for Velocity Based Training (VBT)”, gymaware.com. https://gymaware.com/product/flex-barbell-tracker/ (accessed Feb. 11, 2025).
[4] Grainger, “Salary and Hiring Data Portal,” Illinois.edu, 2024. https://ecs.grainger.illinois.edu/salary-data/data-portal
[5] “Bluetooth TechnologyProtecting Your Privacy,” Bluetooth® Technology Website, Apr. 02, 2015. https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/
[6] “Bluetooth security,” Apple Support. https://support.apple.com/guide/security/bluetooth-security-sec82597d97e/web
[7] IEEE, “IEEE Code of Ethics,” ieee.org. https://www.ieee.org/about/corporate/governance/p7-8.html (accessed Feb. 11, 2025).