WEARABLE BASKETBALL JUMPSHOT MECHANICS ANALYZER
ECE 445 Design Document

Team #78
Tanmay Nair, Arjun Vyas, Aiden Zack

Professor: Joohyung Kim
TA: Mingrui Liu

Contents

1 Introduction

1.1 Problem

A basketball jumpshot involves a chain of body mechanics that requires coordination from your feet to your wrist to achieve a simple goal that is much more complicated than what the average person sees: Making the ball go in the hoop. So many players across the world have exhibited different mechanics in their jumpshot, so when they reach out to coaching for help, they tend to hear subjective advice that is often inconsistent, difficult to put into numbers, and, more importantly, harder to fit into the player's perspective. Existing resolutions utilize shot trajectory and do not tap into the biomechanics that reside in the shooter. In essence, this leads to players lacking reliable, repeatable data to identify points of improvement in their mechanics, address consistency issues, and record progress.

1.2 Solution

This project will implement a system dedicated to quantifying a user's basketball jumpshot by analyzing the consistency and timing of the "kinetic chain". It starts with node sensors that will be worn on the user's shooting wrist, shooting elbow, hip, and the knee of the user's shooting side. These sensors will hold an IMU, microcontroller, and wired communication. The knee sensor will focus on lower-body motion and take measures related to shot success, such as the timing of the jump and how much the knee flexes to determine the dip [5]. The wrist sensor will look at the upper-body mechanics that finish out the shot, like the angular velocity and release timing of the wrist, along with how high it sits for the follow-through [5]. The hip sensor will measure vertical displacement and horizontal tilt [5]. The elbow sensor will measure the angular velocity and push of the shot [5]. These 4 data nodes will be synchronized in our system, extracted for timing measures like jump-to-release, and then processed for evaluation and feedback. This will focus on the repeatability and timing of the user's body mechanics, providing user-oriented assistance that adjusts as the user progresses.

1.3 Visual Aid

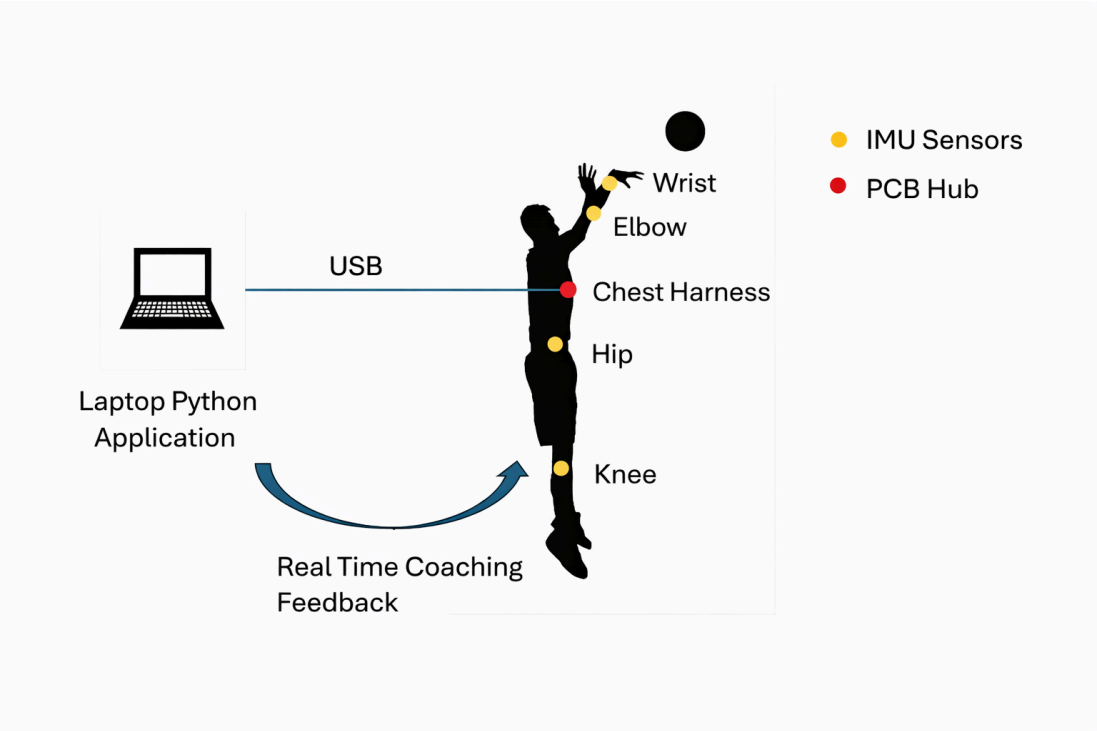

Figure 1: Visual Aid of Component Locations

1.4 High-Level Requirements

1.4.1 Temporal Synchronization and Resolution

This system will utilize a hardware FSYNC trigger to align time stamps of samples, along with a chip select to match incoming data to its corresponding IMU sensor. As a result, the system must meet (at a minimum) a sampling frequency of 200 Hz across all 4 IMU nodes. A temporal skew across sensors should be kept to less than 1 ms to match the FSYNC trigger. Meeting all of these specifications will enable the system to ensure that even high-velocity movements, like a wrist flick, will be sampled with enough data points and within a synchronized timeline to allow for latency calculations between movements.

1.4.2 Measurement Range and Stability

Each IMU subsystem must support a gyroscopic dynamic range of up to ±2000 degrees/second and an accelerometer range of at least ±16g due to explosive movements that may disrupt the signal and sensor performance. This is especially important at peak angular velocities of the wrist flick and high-G impacts like the shooter's landing. This will allow for efficient data collection and ensure its usability for the Python program throughout the entire jump shot.

1.4.3 Proper closure of the Feedback Loop

Our Python backend program must identify each data point to its corresponding sensor and time stamp, and interpret “kinetic milestones” such as maximum knee dip and peak jump acceleration. This will be done using peak detection and zero-crossing algorithms (precision of ±5 ms) to calculate temporal offsets. These offsets, along with the 6-axis IMU data, which we can use to quantify the magnitude of movements, will be stored according to the user’s inputter outcomes (make or miss). From here, the program will validate the shot quality based on these trials, producing interpretable feedback that encompasses timing, power, and form errors.

1.4.4 Demo Procedure

Demo will include a video demonstration of a “bad” shot as well as a “good” shot to match the data we collect. This provides significance and ease of inference to the telemetry data, which will help justify the feedback our Python program will suggest. For consistency and safety during the demonstration, all shots will be performed as free throws.

2 Design

2.1 Block Diagram

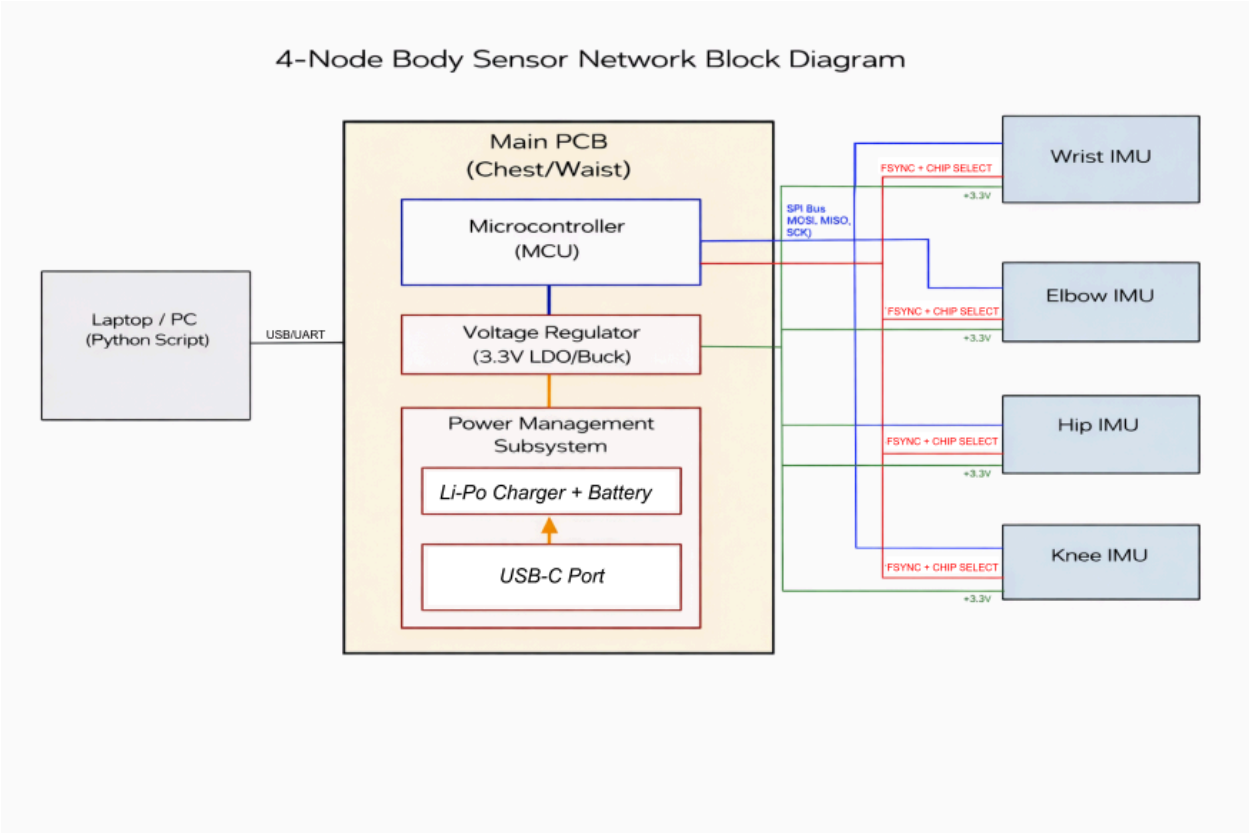

Figure 2: Wearable Basketball Jumpshot Mechanics Analyzer Block Diagram

2.2 Physical Design

The system is implemented as a wearable, body-mounted sensing platform consisting of a central PCB hub and multiple distributed IMU sensor nodes. The PCB hub is mounted on the user's torso using a chest harness and houses the microcontroller, power management circuitry, and USB interface for data transmission to a laptop. Four IMU sensor nodes are placed on the shooting-side wrist, elbow, hip, and knee to capture motion along the kinetic chain of a basketball jumpshot. Each sensor node is secured using adjustable straps or elastic bands to ensure consistent placement while allowing natural joint motion. The knee IMU is positioned along the femur on the lateral side of the quadriceps to capture lower-body motion associated with jump timing and knee dip. The hip IMU is mounted directly over the hip to measure vertical body movement and horizontal tilt. The elbow IMU is placed on the lateral side of the upper arm, just above the elbow joint, to capture elbow rotation and push during the shooting motion. The wrist IMU, responsible for capturing the wrist flick, is positioned near the base of the thumb to minimize interference with natural wrist movement during shot release and follow-through.

All electronics are enclosed within compact housings with smooth edges to prevent discomfort or interference during movement. Wired connections are smoothly routed along the body and strain-relieved to reduce cable movement and accidental disconnection. The system is designed to be lightweight and unobtrusive, allowing the user to perform natural shooting motions safely while collecting synchronized motion data.

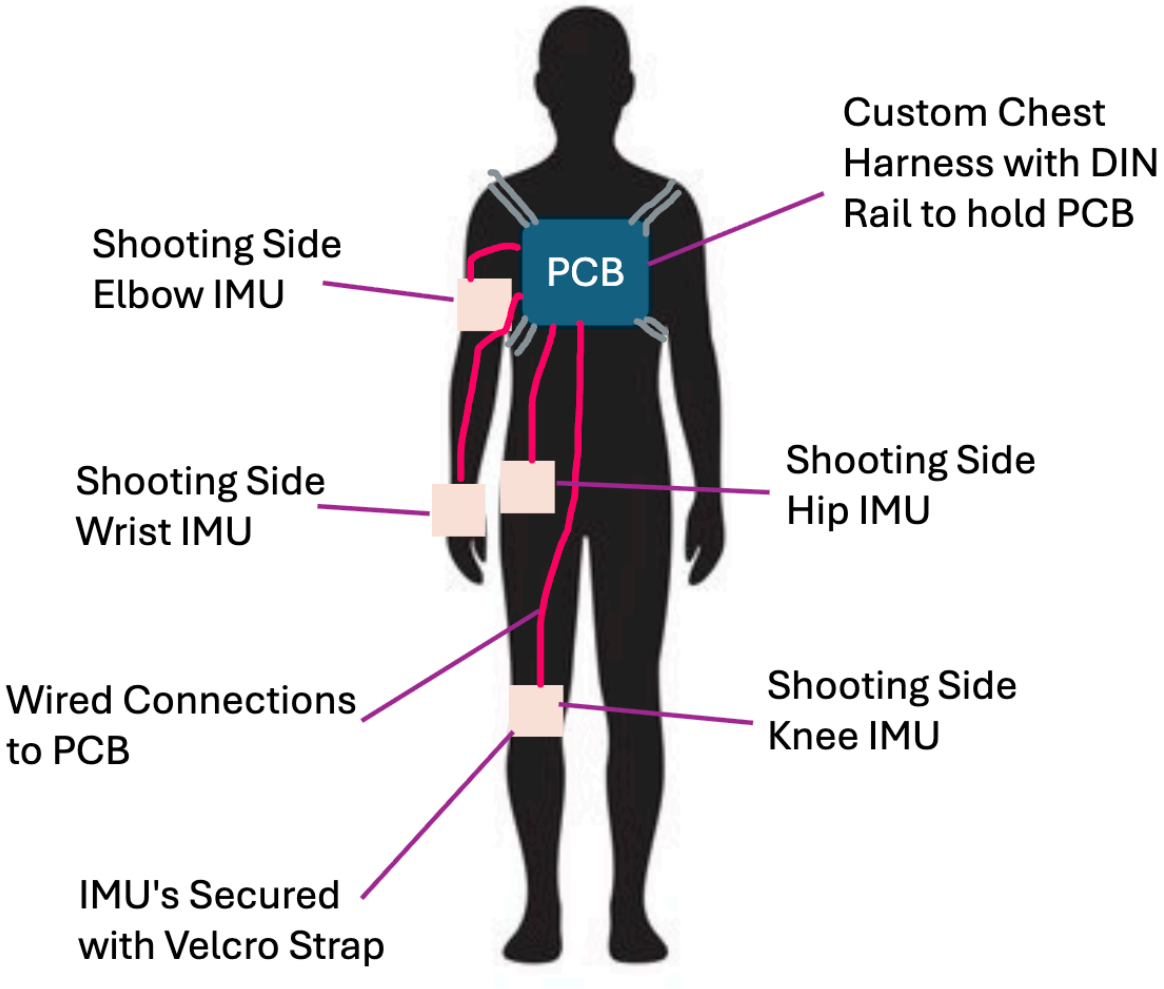

Figure 3: Physical Design Diagram

2.3 Subsystems

2.3.1 Microcontroller (Arduino Nano ESP32)

This subsystem consists of the Arduino Nano ESP32 microcontroller. The Arduino Nano ESP32 requires an input voltage of 5V which will be supplied by a variable voltage source through the

USB-C port. The purpose of the MCU is to be the centerpiece between IMU data and the data processing that takes place in the software. The MCU will read the data from IMU's live movements via SPI bus. The MCU will communicate with the computer via USB-C/UART and transfer the data. The system will maintain a UART baud rate of 115,200 to stream a 48 byte synchronized data packet to Python for data processing without any buffers.

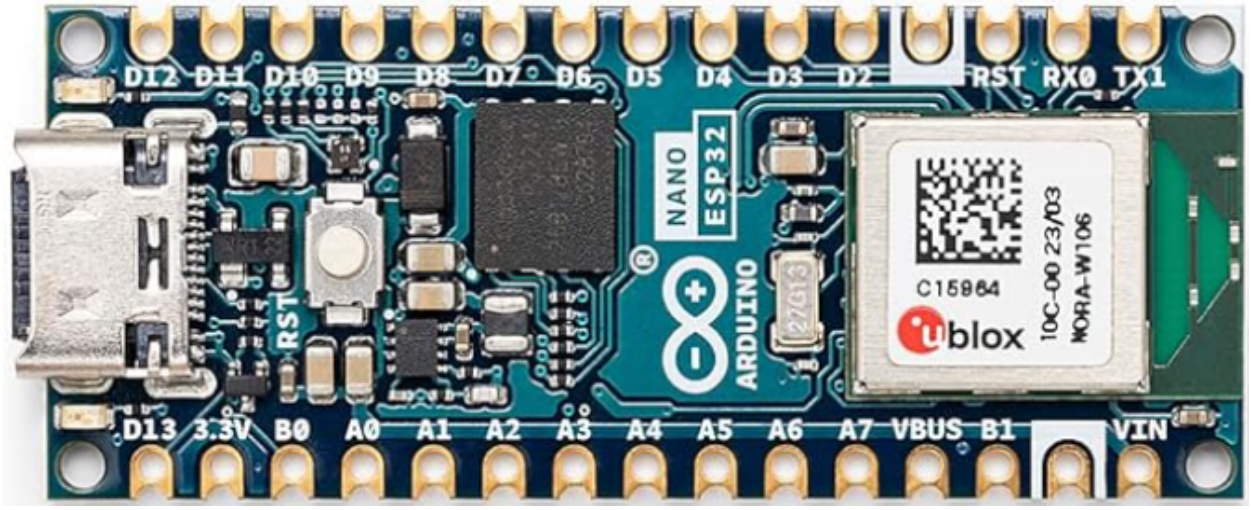

Figure 4: Arduino Nano ESP32 [8]

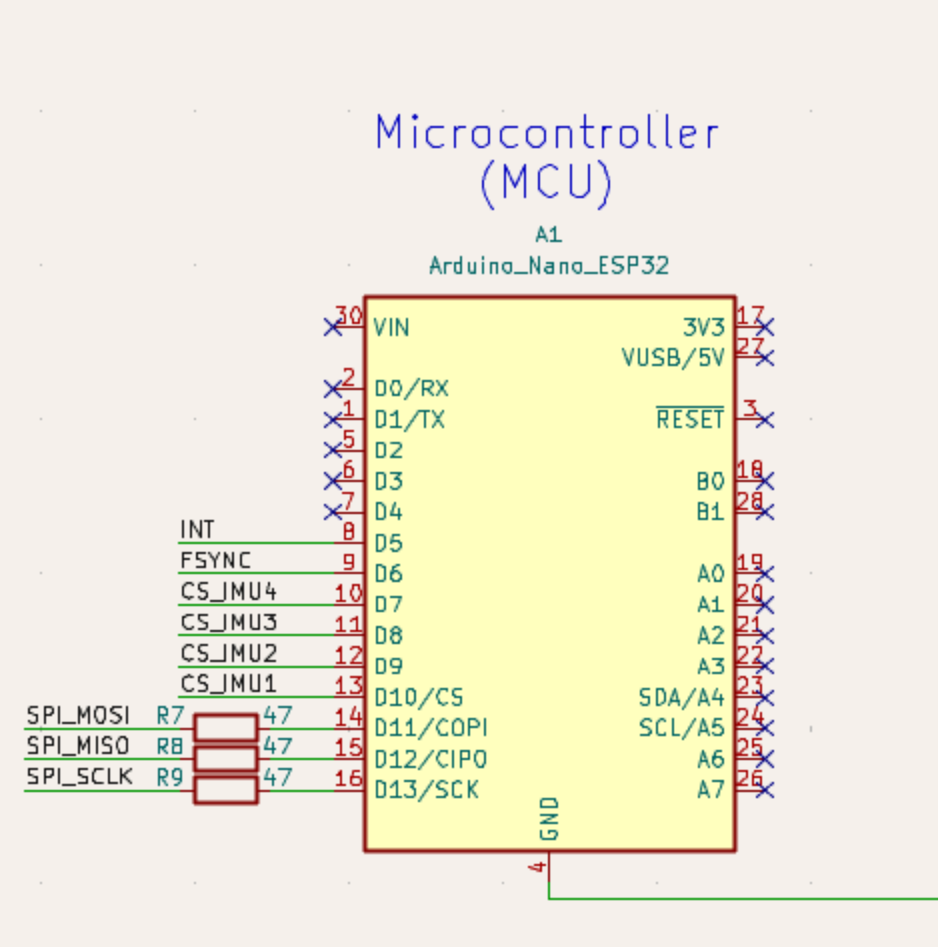

Figure 5: Microcontroller Subsystem Schematic

Requirements	**Verification**
• Arduino Nano ESP32 will receive an input voltage of 5V via USB-C connector. We will use a variable voltage source (i.e Scopy) to supply the ESP32 with required 5V.	• Verify the power LED on the ESP32 is on. Use a DC voltage multimeter to read the voltage between the VBUS and GND pins and verify the reading is ~5V.
• Arduino Nano ESP32 will communicate with the IMU sensors via SPI bus.	• Verify the SPI bus is intact by reading the IMU's WHO_AM_I register value. The returned value for the ICM-20948 IMU should be 0xEA.

• Arduino Nano ESP32 will read the recorded data from the IMU sensors.	• Verify the data is being read by waking the sensor and physically moving the IMU around. Read the acceleration numbers on the Arduino IDE and validate that the numbers being read are changing in real time.
• Arduino Nano ESP32 will transfer the read data to the PC for data processing and user feedback.	• Verify the data appears in the Arduino IDE serial monitor. When the IMU sensor is moving, validate that the numbers displayed in the serial monitor change relatively in real-time.

Table 1: Microcontroller – Requirements & Verification

2.3.2 IMU Connectors

This encompasses our 4 6-axis ICM-20948 IMU sensors that capture the full “kinetic chain” involved in a person’s basketball shot. These 4 sensors are wired and connected to the same SPI bus and common FSYNC line, but hold their own chip select connection from the MCU. Each sensor plays a different role in the chain and, as a result, will have its raw biomechanical data processed in Python based on the key movement they involve. The IMU on the wrist will measure angular rotation, angular velocity, vertical displacement, and horizontal displacement. The IMU on the elbow will measure angular rotation, horizontal displacement, and vertical displacement. The IMU on the hip will measure vertical displacement and horizontal tilt. Lastly, the IMU on the knee will measure angular rotation, angular velocity, and vertical displacement. These 4 IMU sensors will receive an input voltage of 3.3V from the battery management system.

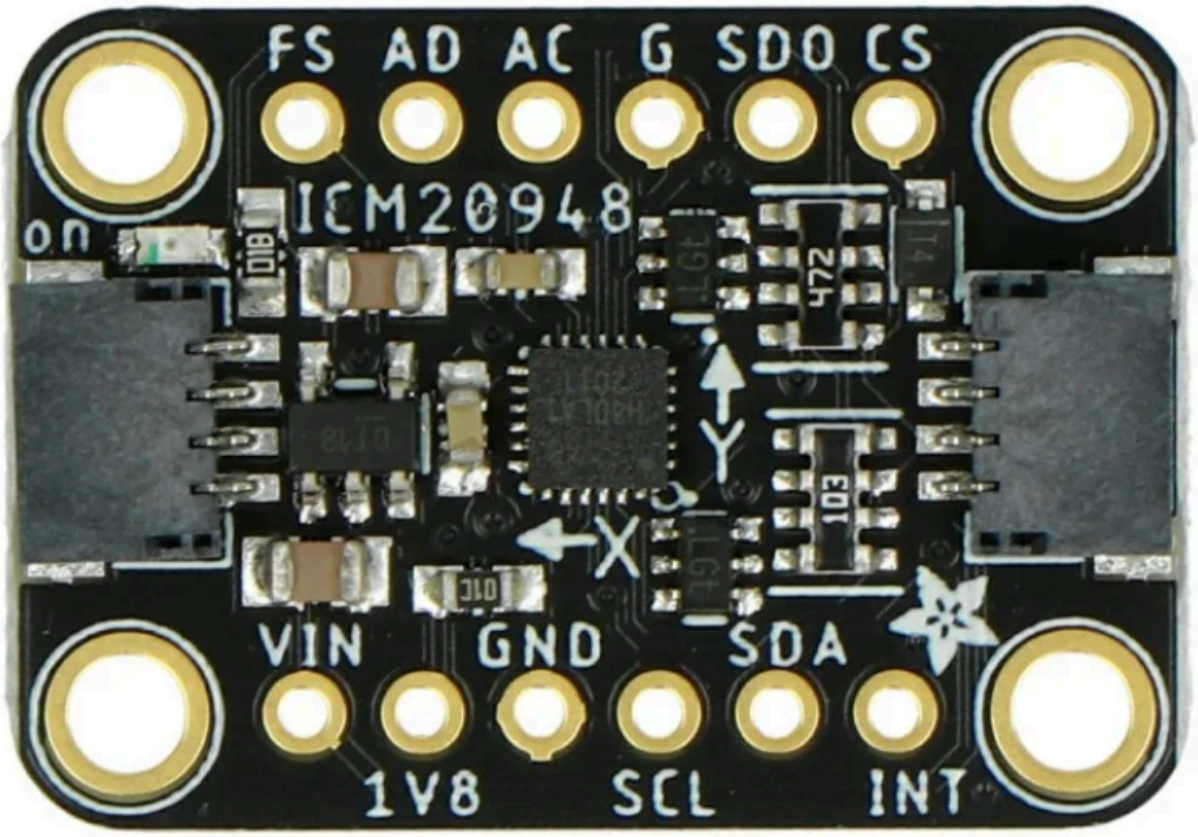

Figure 6: ICM-20948 IMU Sensor [7]

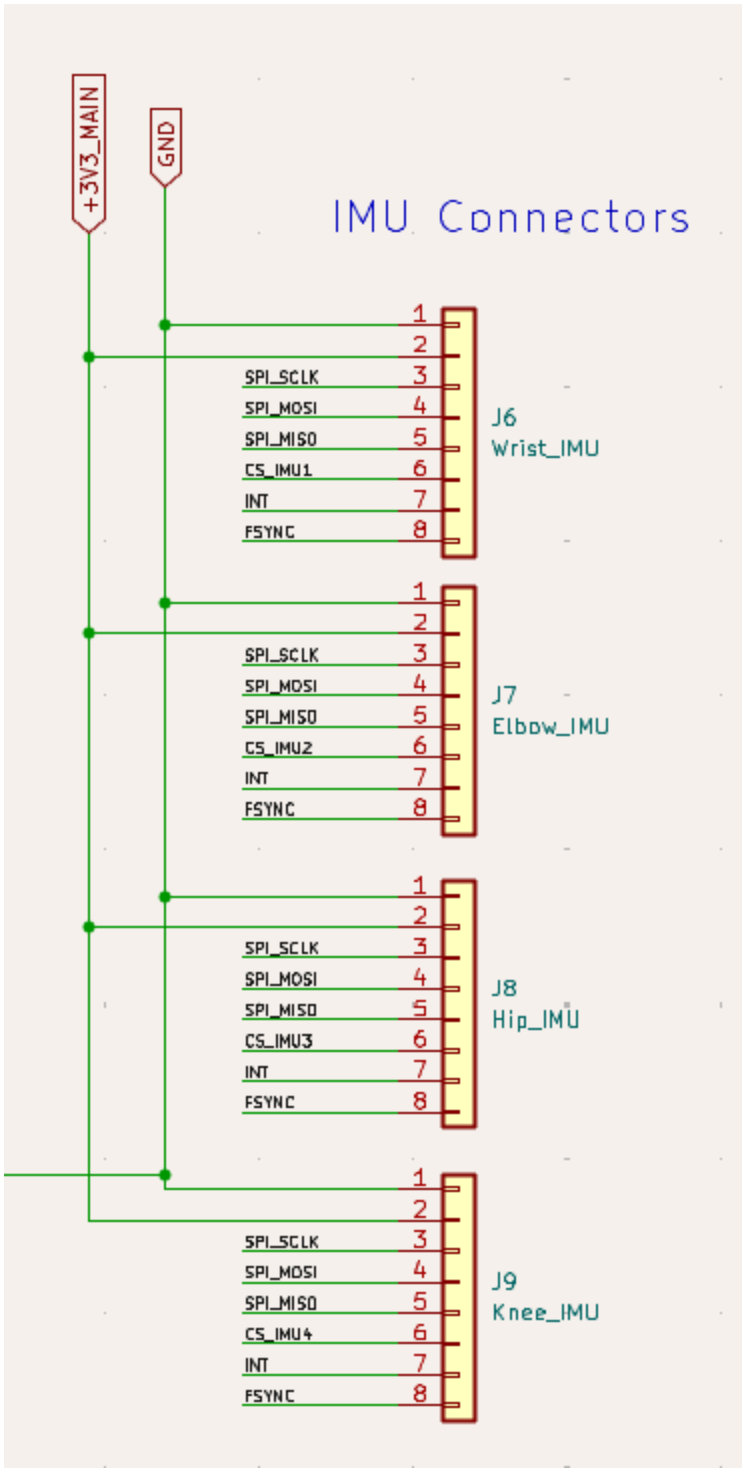

Figure 7: IMU Connectors Subsystem Schematic

Requirements	Verification
• The ICM-20948 IMU sensors will receive an input voltage of 3.3V from the battery management system.	• Verify the IMU sensors are receiving 3.3V by using a DC multimeter and measuring the voltage between the VIN and GND pins.
• The ICM-20948 IMU sensor will communicate with the Arduino Nano ESP32 via SPI bus.	• Verify the SPI bus is intact by reading the IMU's WHO_AM_I register value. The returned value for the ICM-20948 IMU should be 0xEA.
• The live ICM-20948 data will be read by the Arduino Nano ESP32.	• Verify the data is being read by waking the sensor and physically moving the IMU around. Read the acceleration numbers on the Arduino IDE and validate that the numbers being read are changing in real time.

Table 2: IMU Connectors – Requirements & Verification

2.3.3 Battery Management System

The battery management system consists of 4 different subcomponents: the USB-C Receptacle, the Charger & Power Path IC, the Li-Po Battery Connector, and a 3.3V LDO Regulator. The USB-C Receptacle will supply a 5V input voltage into the charger through a variable voltage source, similar to the Arduino Nano ESP32 Microcontroller. When plugged in, the charger will replenish the Li-Po Battery with 5V while simultaneously driving the charger's system output voltage (which internally stabilizes itself to ~4.4V) [6]. When no charging is apparent, and the 5V from the charger is gone, the power-path management of the charger system automatically switches its source voltage to the Li-Po Battery, which drives the system output voltage [6]. This system's output voltage is what feeds into the 4 ICM-20948 IMU sensors, that is, after being stepped down through the 3.3V LDO Regulator.

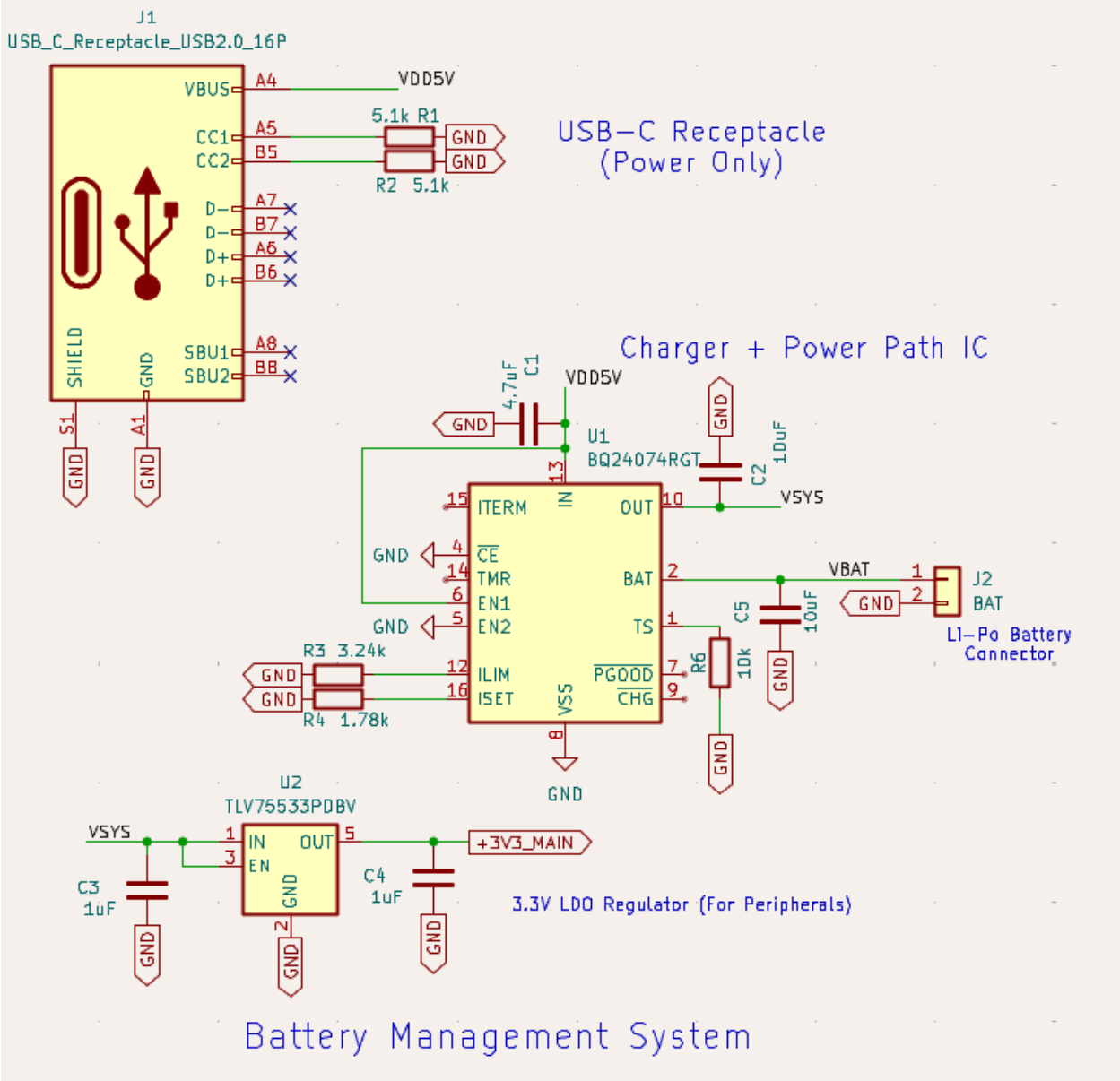

Figure 8: Battery Management Subsystem Schematic

Requirements	Verification
• The USB-C Receptacle will receive an input of 5V from the variable voltage source (i.e Scopy).	• Verify the receptacle receives 5V by using a DC multimeter. Measure the voltage between VBUS and GND and validate this value is ~5V.
• The Charger will receive an input of 5V from the USB-C Receptacle.	• Verify the charger receives 5V by using a DC multimeter. Measure the voltage between IN and GND and validate this value is ~5V.

• The Li-Po Battery will be charged by the charger.	• Validate that the Li-Po battery is actually being charged by plugging in a partially discharged battery. Validate that the battery voltage increases over time by using a DC multimeter.
• The system output voltage (VSYS) will receive 4.4V after internal stabilization when the USB-C is plugged in. When unplugged, VSYS will receive 3V - 4.2V.	• Verify the VSS receives 4.4V when the USB-C is plugged in by using a DC multimeter. Measure the voltage between VSYS and GND and validate this value is ~4.4V. When the USB-C is unplugged, repeat the same process and validate this value is between 3.0- 4.2V.
• The 3.3V LDO Regulator will step down the 4.4V or 5V (depending on if the charger is connected or not) to 3.3V.	• Verify the LDO regulator is stepping down the voltage to 3.3V by using a DC multimeter. Measure the voltage between OUT and GND and validate this value is ~3.3V. This is the voltage that is also being supplied to the 4 IMU sensors. For a sanity check also measure the voltage between VIN and GND on the IMU sensors to verify the proper voltage is being distributed.

Table 3: Battery Management System – Requirements & Verification

2.3.4 Software Data Processing

The Python/Arduino IDE-based backend subsystem runs on a PC and serves as the primary data processing and feedback engine for the system. It receives synchronized accelerometer and gyroscope data streamed from the microcontroller over USB-C, parses the incoming telemetry data, and organizes measurements by sensor location and time. To improve signal quality, the backend applies filtering and smoothing to the raw data before extracting biomechanical features such as movement magnitudes, peak events, and temporal offsets between joints along the kinetic chain. Using data collected from multiple user trials, the system constructs a personalized "Success Profile" that represents consistent motion patterns observed during successful shots. New trials are compared against this profile to evaluate timing consistency and repeatability, enabling the system to generate interpretable, real-time feedback without prescribing rigid mechanical corrections.

Requirements	Verification
• The Arduino IDE will receive the ICM-20948 IMU read data from the Arduino Nano ESP32 Microcontroller.	• Verify the data appears in the Arduino IDE serial monitor. When the IMU sensor is moving, validate that the numbers displayed in the serial monitor change relatively in real-time.
• The Arduino IDE will transfer the data to Python.	• Verify data from the Arduino IDE is being transferred to Python by sending sensor values using Serial.print() over USB-C. Then run a pyserial Python script over the same COM port. Validate the Python data is accurate by moving the IMU sensors and seeing that the changes recorded in the Arduino IDE are the same as the changes recorded in Python.
• Python program will filter out noise and signal discrepancies while maintaining records from fast-moving events.	• Verify the data is accurately filtered by plotting the frequency response and validating that all frequency-related discrepancies have been removed. This will involve a ~30Hz low-pass filter, such that high-frequency noise is silenced and signal integrity is maintained.
• Python program will capture samples that fully encompass the entire kinetic chain of the jumpshot.	• Verify that the "start of the jumpshot" event occurs when the gyroscope magnitude (particularly at the knee, which starts the movement) exceeds a set threshold. • Cross-check with the preceding sample of inactivity.
• Python program will create a personalized "Success Profile" based on trial shots for key timing offsets and motion magnitudes. The program will also produce feedback for the following shots, based on the discrepancies of the "Success Profile."	• Verify storage of "successful" shot attempts and their associated accelerometer/gyroscope data. • With a valid sample size built, verify program feedback aligns with "successful" data and cross-checks with current user attempts to deduce resolution. • Verify program adds additional "successful" shot attempts to the sample database to better generalize the user's shot profile.

Table 4: Software Data Processing– Requirements & Verification

2.4 Tolerance Analysis

One aspect that does pose a risk in our project is how we intend to synchronize data from our 4 sensor nodes and provide it to the microcontroller. Our effective solution involves using a

common FSYNC trigger. At the rising edge of each sampling cycle (time t_0), our microcontroller will send a FSYNC pulse to latch the IMUs' accelerometer and gyroscope instruments simultaneously. This prompts the IMUs to record and direct the data back to the MCU sequentially over SPI using the corresponding chip select lines. In essence:

- MCU will read from the Wrist IMU at $t_0 + \Delta t_1$
- MCU will read from the Elbow IMU at $t_0 + \Delta t_2$
- MCU will read from the Hip IMU at $t_0 + \Delta t_3$
- MCU will read from the Knee IMU at $t_0 + \Delta t_4$

In terms of a realistic situation, let's assume a basketball jumpshot (from the initiation of the knee dip to the ball leaving the player's hands) takes ~600-800 milliseconds, and we set our sampling rate to 200 Hz. This indicates a sample will be taken every 5 ms, which gives us ample data capture for even the fastest of movements:

- Wrist flick duration ~ 30 ms → N = $30\ ms\ /\ 5\ ms\ =\ 6\ samples$

Nyquist Check:
If we take the fastest event (wrist flick) and translate it into a frequency:

$$f_{event} \approx \frac{1}{0.03\ seconds} \approx\ 33\ Hz$$

Our sampling frequency of 200 Hz is more than double f_{event}, meeting Nyquist standards and preventing any form of aliasing from occurring.

The worst-case scenario to consider is if the FSYNC fails, in which case the sampling rate can be increased. This is viable, as the timing error between sensors $100\ \mu s$ compared to the time difference between samples (2 - $5\ ms$) is negligible.

The signal integrity significantly influences our SPI clock choice. Our SPI cable connections range from 0.3 m (PCB at chest → elbow) to 1 m (PCB at chest → knee). This latter length limits our SPI clock, as various factors such as ground bounce and skew between the serial clock and data lines decay the integrity of the signal. With the right measures taken, such as series resistors at the serial clock and data lines of the microcontroller, our SPI can comfortably run at **2 MHz**.
This achieves a minimal EMI, robust signal over 1-meter-long cables, and $224 \mu s$ transfer of 56 bytes, which occupies less than 5% of the sampling period of 5 ms:

$$4\ IMUs\ \times\ (12\ data\ bytes\ +\ 2\ address\ bytes)/\ IMU\ =\ 56\ bytes$$

3 Cost and Schedule

3.1 Costs Analysis

The total cost with all of the associated parts as seen with the figure below is $194.52. Our team is anticipating 15 hours per week of work for the 10 weeks of the project, totaling 150 hours. According to the University of Illinois, the average starting salary for entry level electrical engineers is $90,115 [4], giving an hourly rate of $43.32. For one team member it would be $43.32 x 2.5 x 150 hours, giving a total of $16,245. We have 3 team members in our group, so tripling that amount would give a total labor cost of $48,735. Summing both labor and materials, it would total to $48,929.52.

Description	Manufacturer	Qty	Price	Link
Arduino Nano ESP32 Microcontroller [8]	Arduino	1	$19.30	Link
IMU Sensor (6-Axis) [7]	Adafruit	4	$59.80	Link
Linear Voltage Regulator 3.3 V	ANMBEST	10	$6.99	Link
Li-Po Charger/Booster	SparkFun	1	$18.50	Link
Lithium Polymer Battery	HXJNLDC	1	$14	Link
Mounting Braces (Chest Harness, Velcro)	Custom	N/A	$35	N/A
DIN Rail	Custom	1	$10.93	Link
Miscellaneous (Wires, Cables, LEDs, Resistors, Capacitors)	ECE Shop	N/A	$30	ECE Shop

Figure 9: List of Components and Cost

3.2 Schedule

Week	Task	Person
March 1st - March 7th	**PCB ORDER MARCH 5TH (Thursday)**	Everyone
	Finalize PCB part selection for ordering and review with TA	
	Begin breadboard prototyping and validation of MCU–IMU communication	

March 8th - March 14th	**PCB ORDER MARCH 12TH (Thursday)**	Everyone
	Verify power regulation and voltage levels	
	Read and log accelerometer and gyroscope data	
	Assemble and solder second round PCB	
	Breadboard Demo - MCU communication with two IMUs	
March 15th - March 21st	Spring Break	Everyone
March 22nd - March 28th	**PCB ORDER MARCH 26TH (Thursday)**	Everyone
	Assemble and solder third-round PCB	
	Integrate IMUs with updated PCB hardware	
March 29th - April 4th	Assemble and solder final PCB revision	Everyone
	Full system integration (multiple IMUs + hub)	
	Initial data streaming to laptop	
April 5th - April 11th	**Progress Demo**	Everyone
	Validate system functionality during controlled motion	
	Debug communication and synchronization issues	
April 12th - April 18th	Refine data processing and timing analysis algorithms	Everyone
	Improve wearable mounting and cable routing	
	System reliability and repeatability testing	
April 19th - April 25th	**Mock Demo**	Everyone
	Fix any minor bugs	
April 26th -	**Final Demo**	Everyone

May 2nd		

Figure 10: Project Schedule

4 Ethics and Safety

4.1 Public Health

From a societal and global perspective, this solution promotes more objective and accessible feedback for athletic training by enabling athletes to better understand their own shooting mechanics and improve through self-guided practice. By focusing on quantifiable timing relationships and consistency rather than subjective judgment, the system supports safer and more informed improvement, potentially reducing the risk of injury from improper mechanics.

Economically, the system provides a low-cost alternative to traditional motion-capture systems and private coaching, improving access to biomechanical feedback for a wider population. Culturally and globally, basketball is one of the most widely played sports across diverse communities, and this portable, wearable solution can be used in regions where training resources or professional coaching are limited. The system has minimal environmental impact, as it relies on low-power, reusable electronic components without requiring large physical installations when compared to other training and motion-analysis systems.

4.2 Engineering Standards

This project considers applicable engineering standards relevant to wearable, low-power electronic systems and data-driven software applications. IEEE standards and best practices guide the design of the embedded hardware and software with respect to electrical safety, system reliability, and responsible interaction with users [2]. General UL safety guidelines for low-voltage, battery-powered consumer electronics inform component selection, power management, and enclosure design to reduce electrical and mechanical hazards [3]. Additionally, ACM ethical guidelines influence the design of the data analysis and feedback software to ensure transparent interpretation of sensor data and to avoid misleading performance claims [1].

4.3 Ethics Guidelines

The ethical risk of injury resulting from changes in shooting mechanics relates directly to IEEE Code of Ethics Principles 1, 3, and 6 [2]. To avoid ethical breaches, the system is designed to prioritize user safety by limiting feedback to observational metrics focused on timing consistency and repeatability, rather than instructing users to make abrupt or corrective changes to their mechanics. In accordance with Principle 3, the system avoids absolute claims about a “proper” shooting form, as there is no “perfect” form, and clearly communicates its limitations. To further prevent harm under Principle 6, safeguards are implemented to discourage abrupt form changes, including warnings against overuse and guidance to discontinue use if discomfort or pain occurs. These design choices ensure that feedback supports user awareness without encouraging unsafe changes that exceed the physical limitations of the user’s body. These considerations align with course ethics guidelines emphasizing harm prevention, transparency, and responsible system use.

4.4 Ethical and Safety Concerns

This system involves multiple electronic sensors worn directly on the body during dynamic athletic movement, which introduces several potential safety concerns. Electrical risks include battery overheating, short circuits, and improper charging of batteries. Mechanical risks include sensor detachment, sharp enclosure edges, and interference with natural joint motion, which could cause discomfort during use. To mitigate these concerns, all electronics operate at low voltage, and sensors are enclosed and mounted to the body. The system is intended for short-duration, dry, indoor use to further reduce electrical and mechanical hazards. Another potential safety concern relates to the feedback provided by the system, which may lead users to modify their shooting mechanics. Changes in form that do not align with an individual's physical limitations may introduce abnormal joint movement or muscle strain, potentially resulting in injury. This system does not involve high voltage, hazardous materials, or unsafe current levels applied to the human body. While batteries are used, they are low-capacity and handled in accordance with established battery safety guidelines. These precautions are consistent with the course safety guidelines regarding wearable electronics, low-voltage systems, and battery-powered devices. No new ethical or safety concerns arose after the proposal phase.

4.5 Safety Concern Procedures

All batteries used in the system are low-capacity lithium-polymer (LiPo) cells. LiPo packs are charged only using manufacturer-recommended LiPo chargers or charging circuits with appropriate voltage/current limits. Batteries are never charged unattended, and are inspected before use for swelling, damage to the pouch, exposed wiring, or abnormal heating. If swelling, odor, or excessive heat is observed, the system is powered off immediately and the battery is removed from use. LiPo cells are stored and transported to prevent short circuits, and wiring is protected using insulation and strain relief. For human interface safety, all sensor nodes are enclosed with smooth edges and secured so that it does not restrict natural joint motion. The system is intended for short-duration use, and users are instructed to discontinue use immediately if discomfort or pain occurs. Sensor placement avoids areas prone to pinching or excessive compression during movement. During development and testing, all electronics operate at low voltage and are powered via regulated supplies. Exposed conductors are avoided, and devices are handled only when powered off during assembly or modification. Demonstrations are conducted in dry, indoor environments with adequate space to allow safe movement and are supervised to reduce hazards due to accidental misuse.

5 References

[1] Association for Computing Machinery, "ACM Code of Ethics and Professional Conduct," 2018. [Online]. Available: https://www.acm.org/code-of-ethics

[2] I*EEE Code of Ethics*, IEEE, 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[3] *UL Solutions*, "Lithium-ion battery safety concerns and precautions for brands," UL, [Online]. Available: https://www.ul.com/insights/lithium-ion-battery-safety-concerns-and-precautions-brands

[4] University of Illinois Urbana-Champaign, "Electrical Engineering undergraduate program" Grainger College of Engineering, [Online]. Available: https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/electrical-engineering

[5] *Physiopedia*, "Biomechanics of the Basketball Jump Shot", 2024 [Online]. Available: https://www.physio-pedia.com/Biomechanics_of_the_Basketball_Jump_Shot

[6] *Texas Instruments,* "BQ2407x Standalone 1-Cell 1.5-A Linear Battery Chargers with Power Path", 2021 [Online]. Available: https://www.ti.com/lit/ds/symlink/bq24074.pdf?ts=1772108676050&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FBQ24074

[7] *ICM-20948 9-DoF IMU Breakout Board*, Adafruit Industries LLC. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/4554/17039173

[8] *"USB C Power Adapter Cable,"* Amazon.com, 2025. [Online]. Available: https://www.amazon.com/dp/B0C947C9QS/ref=twister_B0FLBG6TH1?_encoding=UTF8&th=1