ECE 445 Final Report

By

Ritvik Manda

Pranav Nair

Shivam Patel

Final Report for ECE 445, Senior Design, Fall 2024

TA: Dongming Liu

11 December 2024

Project No. 22

Abstract

The Smart Stick System (TripleS) is an innovative lacrosse performance tracking device that leverages advanced sensor technology and cloud integration to provide real time feedback on shot speed, accuracy, and stick form. The system includes three subsystems: LaxSense, a lightweight sensor module mounted on the stick; LaxHub, a central processing unit; and a React-based mobile application. TripleS achieved shot speed accuracy within ±10 mph, trajectory prediction within ±10 feet, and data transmission to the app in just 1.2 seconds. The modular design significantly helped integrate components and assisted during development, while AWS architecture opened up possibilities for scalability and reliability. This end to end solution successfully solves a relatively unaddressed gap in lacrosse training by allowing players to monitor and improve their performance with actionable insights, meeting all project objectives and requirements.

Contents

1. Introduction

The sport of Lacrosse has been missing the sophisticated performance analysis tools available in other athletic sports. Traditional training methods rely heavily on subjective observation, which is not very consistent. No tools such as those available for other sports like baseball, golf, soccer, etc are available to monitor and improve lacrosse form and accuracy, especially when a player is training alone. The lack of real-time, actual data focused feedback on metrics such as shot speed, accuracy, and stick form has meant that both novice and experienced players do not have a reliable method to track their progress and refine their skills.

Our Smart Stick System (TripleS) addresses this critical need in the lacrosse community. By leveraging sensor technology and data analytics, TripleS provides players and coaches with insights into lacrosse performance. The system gives immediate feedback and analysis, meaning users can now make data informed choices to enhance their gameplay. The end product is very user friendly and fits a completely exclusive niche in the world of sports performance tracking [1], [4], [5].

2. Outline

2.1 Introduction

Our TripleS solution integrates three critical subsystems—LaxSense, LaxHub, and the TripleS Application—to deliver a comprehensive and robust technological ecosystem, as shown in Figure 1. This integrated approach ensures seamless data collection, centralized management, and user-friendly interaction across our product.

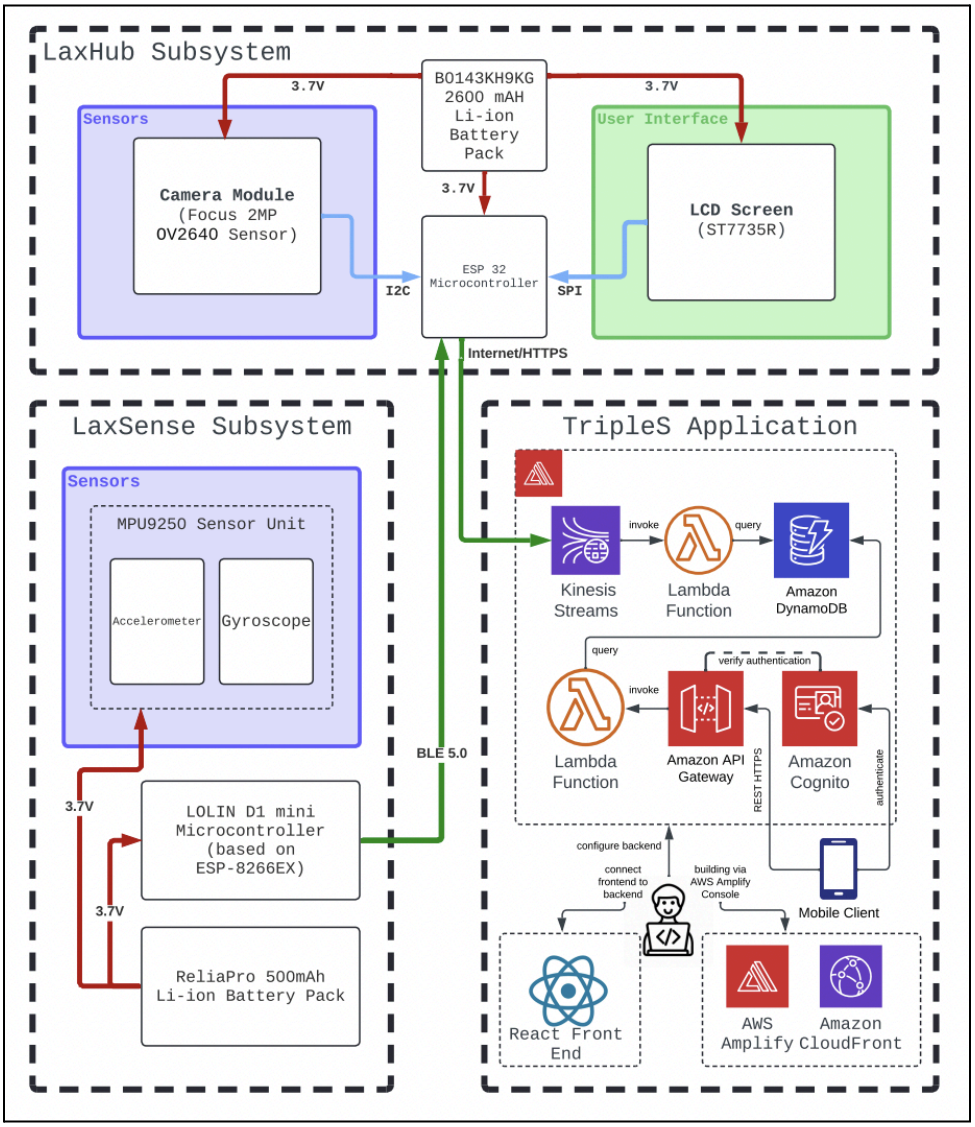


Figure 1: Initial Smart Stick System (Triple S) Block Diagram

Our initial TripleS solution emerged as an innovative proof-of-concept to transform lacrosse performance tracking through an integrated hardware-software ecosystem. The initial LaxHub prototype serves as our central processing unit, featuring a custom PCB, microcontroller, LCD screen, and camera with a

Bluetooth 5.0 module. As our first-generation design, this component was conceptualized to provide a compact, rechargeable central hub for data collection and intermediate processing, establishing a low-power Bluetooth connection with the LaxSense subsystem and facilitating cloud backend communication via a Kinesis client. The LaxSense subsystem represents our initial attempt to instrumentalize the lacrosse stick, integrating a microcontroller, accelerometer, and gyroscope to capture fundamental performance metrics. This early-stage design aimed to track critical parameters like shot speed, stick angle, and form through a compact, stick-mounted device powered by a modest battery system. Our TripleS application—the initial software interface—was developed as a React-based mobile platform leveraging AWS Amplify. This first-iteration design focuses on translating raw performance data from the microcontroller, stored in DynamoDB, into intuitive, actionable insights for athletes and coaches.

2.2 High-Level Requirements

1. **Real-Time Performance Tracking:** Deliver comprehensive swing feedback within 30 seconds, capturing shot speed, accuracy, and stick form with minimal latency from data collection to cloud transmission and application visualization.
2. **Precise Performance Metrics:** Ensure ball speed measurement accuracy within ±10 mph and trajectory within 10 feet, enabling athletes to track and understand their skill progression with data-driven insights.
3. **Minimally Invasive Design:** Engineer the LaxSense unit to weigh under 3 ounces, preserving stick mechanics and ensuring the device does not interfere with the player's natural movement and performance.

2.3 Changes to Block-Level Diagram

While our core system architecture remained fundamentally consistent, we strategically optimized our approach to meet our high-level requirements. We transitioned from Bluetooth to WiFi with MQTT clients, providing a more robust and scalable communication protocol between subsystems. The LaxSense and LaxHub modules remained structurally unchanged. However, we significantly enhanced the TripleS application's backend infrastructure by replacing Kinesis with AWS IoT Things and implementing certificate-based authentication for the ESP32 microcontroller. This shift to a pub/sub MQTT communication model streamlined our data transmission. The foundational technology stack—including DynamoDB, Lambda, React, Cognito, and Amplify—remained consistent, ensuring a seamless and secure application experience.

3. Design

3.1 Design Procedure

3.1.1 LaxHub Subsystem

The LaxHub is a communication hub unit with other hardware supplementing it with user accessibility. The ESP32-S3 microcontroller is used as the central processing unit [2]. Though alternatives like NodeMCU ESP32-S and STM32 were similarly used, the ESP32-S3 seemed like the best choice due to its low power performance, integrated WiFi, and processing power perfectly suited to our performance tracking requirement.

We considered many battery options, including smaller Li-ion batteries and USB-powered options, ultimately selecting the B0143KH9KG 3.7V-2600mAh Battery Pack. We looked for an option with extended battery life, a consistent 500mA power supply, and rechargeability as these are essential factors for a portable performance tracking device.

We also compared multiple camera sensors, weighing options with higher resolutions and different form factors. The 2MP OV2640 Sensor was the perfect compromise as it offered a compact form factor while being able to capture 1080p video at 30 ±12 fps [3]. This balanced approach ensures performance tracking capabilities without unnecessarily bulky hardware.

Rather than defaulting to I2C screens or larger displays, we chose the ST7735R LCD with an SPI interface. This decision prioritized faster data transfer and the ability to quickly display critical performance information, since we required that players receive near-instantaneous feedback.

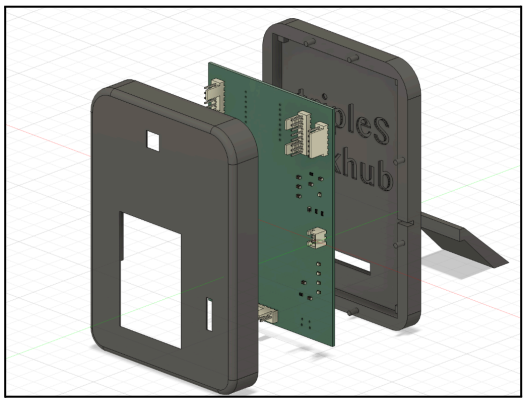


Figure 2: LaxHub Unit

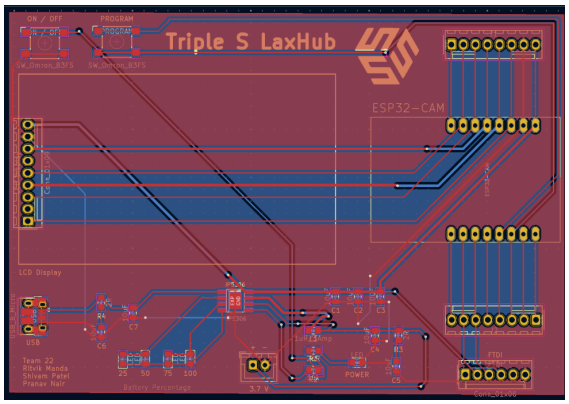


Figure 3: LaxHub Circuit Unit

Figure 2 above depicts the general design of our full LaxHub unit. The 3D printed shell encloses our PCB and components, leaving areas for user interaction. More specifically, the LCD screen, camera, and battery percentage LEDs show through, and the user can access power on/off buttons directly on the PCB. Figure 3 shows a much more detailed version of our circuit schematic on the PCB. The bottom area shows a power system centered around the IP5306 chip which is configured to step up voltage as well as read battery percentage. SMD LEDs in this section display battery percentage, and if the system is powered on or off. The bottom right bumper connection along with one of the top left buttons is used as a method to program our ESP32. Finally, we include ample space to wire up the LCD screen and ESP32 microcontroller itself.

Another essential component of LaxHub processing is to predict the trajectory of the ball that would be released from a similar pass. We were able to use the gyroscope and accelerometer to calculate the angular velocity and acceleration of the shot, mainly due to the pivoting nature of the lacrosse shot. Using linear release speed and angle measured using the gyro, we predicted the final trajectory with the trajectory equation. We use

$$x = (v_0\cos\theta)t \text{ and } y = (v_0\sin\theta)t - (1/2)gt^2 \quad (1)$$

to correspondingly calculate the predicted distance and height trajectory values, and record/display them on the LaxHub unit.

3.1.2 LaxSense Subsystem

The LaxSense subsystem is an engineered solution for capturing lacrosse performance metrics, designed to provide motion tracking without compromising stick dynamics. Our sensor selection process explored multiple tracking options before deciding on the MPU-9250 sensor unit [7]. Alternatives like the LSM9DS1 and BMI160 were considered, but the MPU-9250 was best as it integrated a 3-axis accelerometer, gyroscope, and magnetometer in one. The configurable acceleration and angular velocity ranges (±2g to ±16g and ±250° to ±2000°/s) gave exceptional flexibility in capturing nuanced stick throwing motions as every player may be different.

We also looked through several microcontroller options on the LaxSense.. While options like NodeMCU ESP32-S and Adafruit Feather Huzzah ESP8266 again had similar features, the LOLIN D1 Mini with ESP-8266EX was the optimal choice [6]. Its low power consumption, compact form factor, and WiFi capabilities aligned perfectly with our performance tracking objectives. The ability to maintain a stable 1 ± 0.5 Mbps data transmission rate within a 10-meter range ensures real-time performance feedback without system latency.

Power management is another critical design consideration. We chose the 500mAh Li-ion battery after evaluating various power solutions, offering a balance between operational duration and minimal weight. This power system ensures reliable performance by delivering a consistent 3.7 ± 0.75 V output and supporting five hours of continuous operation. Finally, by maintaining a total weight under 3 ounces and utilizing low-power components, the LaxSense integrates seamlessly with the lacrosse stick, keeping the player's natural mechanics while providing performance tracking.

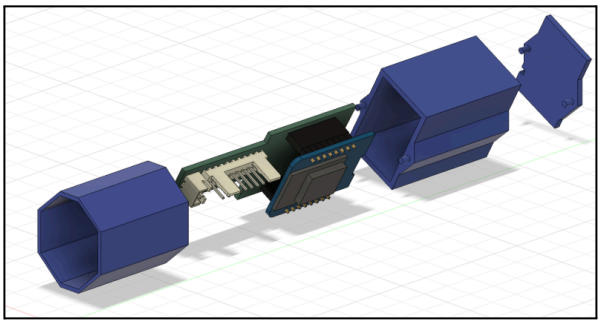


Figure 4: LaxSense Unit

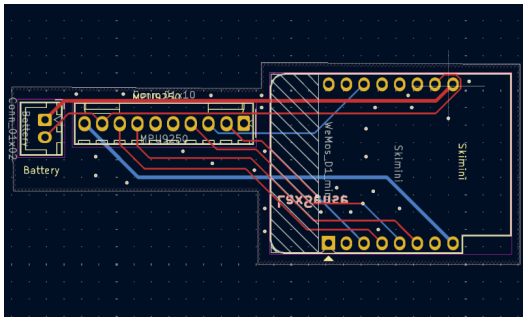


Figure 5: LaxSense Circuit Unit

Figure 4 above shows a model of the 3D printed enclosure and PCB for the LaxSense subsystem. The shape of the bottom of the enclosure is carefully measured to act as a substitute for the rubber stop on the ends of Lacrosse sticks, meaning this entire unit can slot into a Lacrosse stick perfectly. Figure 3 shows the overall circuit design for the LaxSense. This subsystem is very simple as it only connects the ESP8266 module with the MPU sensor array and a battery connection at the left. The PCB is shaped in a smaller rectangular prism to fit in the Lacrosse stick including the battery.

As part of development, we also analyzed accuracy of our sensor and LaxSense subsystem based on the following:

1. Gyroscope sensitivity: 131 LSB/(°/s) for ± 250 °/s full range scale
2. Gyroscope total root-mean-square noise 0.1 °/s
3. Typically a lacrosse shot takes approximately 0.2 seconds
4. Typically lacrosse shot speed ranges between 70-100 mph

Listed below are the steps and calculations to show the subsystem feasibility.

Step 1: Calculate the angular velocity of a typical shot, assuming a 90° rotation

$$\omega = 90° / 0.2s = 450°/s$$

Step 2: Calculate the gyroscope output for ω

$$450°/s * 131 \text{ LSB}/(°/s) = 58{,}950 \text{ LSB}$$

Step 3: Calculate the error due to gyroscope noise

$$\text{Error } \omega = 0.1°/s * 0.2s = 0.02°$$

Error in rotational measurement = 0.02° / 90° = 0.022%

Step 4.1: Translate rotational error to linear velocity error, assuming stick length of 1.016 m

$$v = \omega * r$$

$$v = \omega * r = (450°/s * \pi/180) * 1.016 \text{ m} = 80.1 \text{ m/s (179 mph)}$$

Step 4.2: Translate rotational error to linear velocity error in a realistic scenario

$$v = \omega * r * \text{transfer efficiency} = (450°/s * \pi/180) * (1.016 * 0.75) * 0.60 = 36.1 \text{ m/s (80.7 mph)}$$

Step 5. Calculate the error in linear velocity

$$80.7 \text{ mph} * 0.022\% = 0.0178 \text{ mph}$$

To make this calculation more robust, we can take additional errors into consideration:

Temperature drift: 0.75%

Calibration error: 1.25%

Step 6: Calculate the total error via the additional errors

$$\text{Total error} = \sqrt{(0.022^2+0.75^2+1.25^2)} = 1.46\%$$

$$80.7 \text{ mph} * 1.46\% = 1.18 \text{ mph error}$$

The calculated error of 1.18 mph is within our high-level requirement of ±10 mph accuracy for shot speed measurement and the 80.7 mph falls within the 70-100 mph for a typical lacrosse shot.

3.1.3 TripleS Subsystem

The TripleS application uses a modern cloud-native architecture with a React-based frontend deployed through AWS Amplify. While our initial design considered Amazon Kinesis for data streaming, we ultimately chose AWS IoT Core as a more suitable solution for our IoT device integration. This alternative approach uses MQTT protocol with pub/sub messaging, which provides several advantages over Kinesis streams. Instead of setting up Kinesis clients and managing stream shards, the ESP32 microcontroller connects directly to AWS IoT Core using X.509 certificates for secure authentication. This certification-based approach provides stronger security and simplified device management compared to managing Kinesis credentials. The MQTT pub/sub model allows for more efficient bi-directional communication between devices and the cloud, which is a good option for IoT applications. The AWS IoT Core implementation is largely better than Kinesis for our use case

because it has built-in device management, lower latency, and reduced complexity. While Kinesis would require managing shard limits and dealing with potential throttling issues, AWS IoT Core instead allows automatic scaling and dedicated IoT-optimized message routing. The pub/sub architecture eliminates the need to worry about streaming limits of 1MB per second or 1000 messages per second that would have constrained our Kinesis implementation.

Data from the ESP32 is published to specific MQTT topics, which trigger Lambda functions through IoT Rules. These rules can directly route messages to DynamoDB, simplifying our architecture by removing the need for intermediate stream processing. User authentication still occurs through Amazon Cognito, and API Gateway handles REST HTTPS communication, but the data ingestion path is more streamlined. The AWS IoT Core architecture maintains our requirement for real-time performance tracking and analysis, easily achieving sub-30-second latency. The MQTT protocol is specifically designed for IoT devices, offering lower overhead and better reliability for intermittent connections compared to Kinesis streams. Additionally, the serverless components (Lambda and DynamoDB) remain unchanged, allowing for efficient data processing. The new architecture also keeps data consistent by enforcing message formats through IoT Core's registry and rules engine, rather than relying on matching Lambda function implementations. This provides a more complete solution for ensuring data format consistency between device publishing and database storage. The overall design choice of AWS IoT Core over Kinesis streams results in a more efficient, secure, and scalable solution that better serves our IoT application needs while maintaining all core functionality requirements.

3.2 Design Details

The motion tracking system operates using an MPU9250 sensor on the LaxSense to calculate shot metrics. It begins with sensor initialization and calibration through our calibrate_imu() function, which takes 100 samples from the accelerometer and gyroscope, calculates their offset values by averaging these samples, and stores the offsets for compensating raw sensor data later. For motion detection, the system monitors the acceleration magnitude, calculated as shown below.

$$\text{acceleration} = \sqrt{accel_x^2 + accel_y^2 + accel_z^2} - 9.81$$

A shot is detected when this value exceeds the threshold of 15.0 m/s^2. When a shot is detected, the system calculates velocity using the following equations:

$$velocity_{final} = velocity_0 + acceleration \Delta t$$

$$distance = velocity \Delta t$$

Our calculate_trajectory() function then determines the parabolic path of the shot, using the launch angle, which is calculated below.

$$\Theta = tan^{-1}(\frac{accel_y}{accel_x})$$

The function additionally calculated the trajectory equation, shown below.

$$y = ax^2 + bx + c$$

In this equation, a, b, and c are derived from the shot's speed, distance, and gravity. In the main loop, the system continuously reads sensor data, applies calibration offsets, monitors shot events, and updates speed and distance in real-time. Once a shot is completed, the results are published via MQTT. The system uses interrupt-driven data acquisition through the isr_imu() function to ensure timely sensor readings and precise calculations. The LaxHub then receives these calculations and uses SPI, Serial Peripheral Interface, to display these metrics on the TFT display screen, as shown in Figure 6 [8]. This same data is then sent to AWS, which then processes the data, creates graphs, stores the images in S3 buckets, and the data itself in the DynamoDB table. The TripleS App makes calls to this table to display the images in a user-friendly manner as shown in Figure 6.

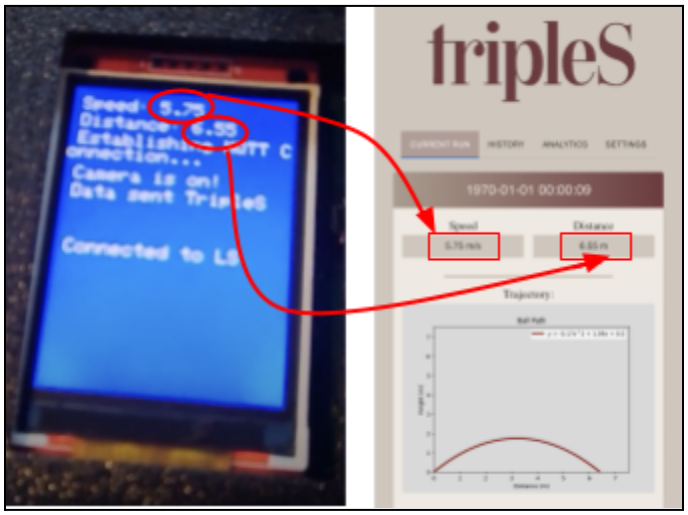


Figure 6: Correlation of Data Between TFT Display on LaxSense and TripleS Application

4. Verification

The verification process for our Smart Stick System focused on ensuring that all high-level requirements were met and that the system performed reliably under real-world conditions. Below, we summarize the testing and verification of each major requirement. A comprehensive Requirements and Verifications table for each subsystem is included in Appendix A for detailed reference.

The first high-level requirement for our system was to achieve real-time tracking by transmitting data from the sensor array to the cloud within 30 seconds. To test this, we used a stopwatch to measure the time from when a lacrosse swing was detected to when the data appeared in the TripleS application. The system consistently completed this process within 1.2 seconds, far exceeding our requirement. The significant speed improvement can be attributed to the use of AWS Kinesis for efficient data streaming and processing, as well as optimized communication protocols such as MQTT Pub/Sub over Bluetooth and Wi-Fi. These technologies allowed for minimal latency in data transfer and processing, even under varying network conditions.

The second high-level requirement was to predict ball distance with an accuracy of ±10 feet. We tested this manually by throwing a lacrosse ball across measured distances and comparing the actual distance with the predicted values displayed on the LCD screen. Over ten trials, the system achieved an average error of ±4.6 feet, which is well within our specified tolerance. This level of accuracy was achieved through careful calibration of the gyroscope and accelerometer in the LaxSense unit, as well as physics focused trajectory calculations using angular velocity and acceleration data.

The third high-level requirement specified that the LaxSense unit must weigh less than 3 ounces to avoid interfering with the natural balance of the lacrosse stick. Using a digital scale, we measured the total weight of the unit (including its 3D-printed housing, PCB, sensors, and battery) at 2.7 ounces, successfully meeting this requirement. This was especially crucial to us considering we did not want a difference in swing feel for players between practice and in game; ultimately we achieved this by carefully planning ahead and adding predicted weight values to keep our unit under the threshold.

The R&V tables from our design document provided a structured framework for testing all system components against their respective requirements. Each subsystem was verified separately with these verifications before integrating them into the final product. On the LaxHub, several of our requirements included printing log messages to the LCD screen when issues happened (disconnection with LaxSense, battery low) or when data was successfully passed to the cloud application. This was verified simply through observation and testing, and helped significantly with further development as well. Another requirement was for our

LaxSense to LaxHub connection to reconnect quickly after a failure, and this was also simple to implement using our Pub-Sub model. We added a heartbeat metric that simply made sure the other system was alive, and if not rebooted its connection to the relevant MQTT broker. Here we were able to verify by using a stopwatch to measure how long it takes for a reconnect to happen if one system is rebooted, and it falls within our threshold. All our requirements were verified successfully without any need for modifications or adjustments to tolerances. For further details on specific tests and their results, please refer to the R&V tables in Appendix A.

5. Costs

5.1 Parts

	Name	Description	Quantity	Cost	Total
1	ESP32-S3 Microcontroller	ESP32-CAM Camera WiFi + Bluetooth Module 4M PSRAM Dual-core 32-bit CPU Development Board with OV2640 2MP Camera Module Support Image WiFi Upload	1	$12.99	$12.99
2	2MP OV2640 Camera Sensor (Comes with ESP32)	OmniVision 2MP camera sensor for capturing images	1	$8.99	$8.99
3	ST7735R LCD Screen	HiLetgo 2.2 Inch ILI9341 SPI TFT LCD Display 240x320 ILI9341 LCD Screen with SD Card Slot for Arduino Raspberry Pi 51/AVR/STM32/ARM/PIC	1	$14.49	$14.49
4	B0143KH9KG Li-ion Battery Pack	Voice Amplifier Replacement Battery B0143KH9KG 3.7V 2600mAh Rechargeable Lithium-ion Battery, with XH2.54mm Connector	1	$12.69	$12.69
5	LOLIN D1 Mini Microcontroller	LOLIN mini Wi-Fi microcontroller	1	$14.99	$14.99
6	MPU-9250 Sensor Unit	MPU9250 GY-9250 9-Axis 9 DOF 16 Bit Gyroscope Acceleration Magnetic Sensor 9-Axis Attitude +Gyro+Accelerator+Magnetometer Sensor Module IIC/SPI MPU9250/6500	1	$14.29	$14.29
7	Li-ion Battery Charger	Adafruit charger for Li-ion batteries	1	$5.99	$5.99
8	FTDI 1232	3PCS FT232RL Mini USB to TTL FTDI Adapter Module, 3.3V 5.5V FT232R Breakout FT232RL USB to Serial Converter Adapter Board	1	$8.99	$8.99
9	EEMB 3.7V LiPo Battery 500mAh	EEMB 3.7V Lipo Battery 500mAh 403048 Lithium Polymer ion Battery Rechargeable Lithium ion Polymer Battery with JST Connector Make Sure Device Polarity Matches with Battery Before Purchase!!!	1	$7.90	$7.90

10	Ximimark SOIC8	Ximimark SOIC8 SOP8 to DIP8 IC Programmer Socket Converter Adapter Module 150mil 200mil For 25xx Eeprom Flash	1	$7.99	$7.99
11	CRCW08052R00DKEAH P	2 ohm resistors	10	$0.387	$3.87
12	LS M676-P2R1-1-Z	Red 633nm LED	10	$0.286	$2.86
13	ERJ-UP6D20R0V	200 ohm resistors	10	$0.261	$2.61
14	BDQQ00201210R33MPA	330 nH inductor	10	$0.172	$1.72
15	Breakaway pins	Exclusive! 2.54mm 1x42pin/2x42pin Gold-Plated Hand Breakaway Female Pin Header Strip (Single Row x 42 pin*10 pcs	1	$9.98	$9.98
16	IP5306 SOP8	Integrated 7 protocols for fast charging protocol ICs for USB ports.	1	$3.00	$3.00
17	JST XH Connectors	JST XH 2.54 mm Pitch 2-Pin JST Wiring Connecting IC Male Plugs, Female Sockets Housing and T-Shaped Crimp Terminal Connector Kit. 50 Sets/200 Pieces JST XH Connector Adapter Cable Assembly.	1	$5.99	$5.99
18	Amazon Kinesis Client SDK	Amazon's real-time streaming data service SDK	N/A	$0.00	$0.00
19	DynamoDB Database	Amazon NoSQL database service	N/A	$0.00	$0.00
20	AWS Lambda	Amazon's serverless compute service	N/A	$0.00	$0.00
Total					$139.34

5.2 Labor

The total cost for parts, as seen in the Bill of Materials above, is $139.34 before shipping. A 5% shipping cost adds an additional cost, bringing the total to $146.31. In Champaign County, a 9% sales tax on the parts cost adds to the final cost, resulting in a total of $158.85 for the components. For labor costs, calculated at $40/hr for 3 hours per day over 40 days, the total salary per team member comes to $4,800. Multiplying this by 3 team members results in a total labor cost of $14,400. Adding the labor cost to the total parts cost gives us a comprehensive total of $14558.85. Therefore, the overall total cost for this project, including materials, shipping, sales tax, and labor, amounts to $14558.85.

6. Conclusion

6.1 Accomplishments

The Smart Stick System (TripleS) project successfully achieved its primary objectives, meeting all high-level requirements and demonstrating its ability to provide accurate and real-time performance tracking for lacrosse players. The system was able to integrate hardware and software components well to deliver reliable data on player performance. The communication pipeline between the LaxSense unit, LaxHub, and the TripleS application performed particularly well, achieving results faster than anticipated. The system's ability to process and transmit data efficiently highlights the effectiveness of its design and implementation.

Additionally, the project showcased strong adaptability when taking on challenges during testing. The system's modular design allowed for easy troubleshooting and testing of individual subsystems and even components, ensuring reliability in the final product. Using AWS cloud services for data storage and analysis allowed for scalability and security and made the system practical for a larger deployment. These accomplishments highlight the project's success in addressing an ignored need in lacrosse training technology.

6.2 Uncertainties

Despite the project's successes, certain uncertainties remain that could impact TripleS in specific scenarios. Our reliance on stable network connectivity for data movement has a limitation, especially in environments with restricted or unreliable Wi-Fi networks, such as university facilities where some features like MQTT Pub-Sub were restricted. Additionally, the MPU-9250 sensor occasionally produced noisy data, which could affect accuracy under less controlled conditions. Even though this did not significantly impact overall results during testing, more refinement of data filtering techniques may be useful for more user value. Finally, due to delays in receiving hardware components, the LaxSense subsystem was tested and displayed using a breadboard instead of the custom PCB design. While this workaround allowed for successful testing, it introduced minor inefficiencies that could be resolved with the intended PCB implementation.

6.3 Ethical considerations

The development of TripleS closely followed ethical principles outlined in the IEEE Code of Ethics by prioritizing user safety, data privacy, and transparency. To ensure safety, the LaxSense unit was designed to be lightweight and securely attached to the lacrosse stick to minimize risks during use. We made sure that the unit was able to physically adhere to the lacrosse stick to make sure it could not fly out in any way. Electrical safety was also considered by always enclosing batteries in some kind of housing and using low-voltage components where possible. Regarding data privacy, only essential performance metrics were collected and securely stored on AWS servers with user authentication protocols in place. Users keep full control over their data with options for deletion of their account. Transparency was another focus for us. Users were clearly told about system capabilities and limitations within the application itself, including potential inaccuracies in trajectory predictions due to sensor noise. By always keeping these ethical concerns in mind through development, TripleS ensures it is both safe and respectful of user rights while keeping trust among its users.

6.4 Future work

In the future, we would like to enhance the functionality and user experience of TripleS using lessons we learned now. First, we would like to transition from Wi-Fi-based communication to Bluetooth which solves connectivity issues in restricted network environments while keeping relatively reliable data transmission. Additionally, implementing a Kalman filter or using a more advanced sensor could reduce noise in motion data and improve accuracy for trajectory predictions. Another option for improvement is adding real time physical feedback such as audio or haptics to notify users about swing quality or performance metrics during training. Finally, expanding the system's capabilities to support other sports or activities requiring motion tracking could open it up as a very useful tool for athletes and have a larger societal impact. These advancements would help solidify TripleS as an innovative tool for athletic performance tracking.

References

[1] "Biomechanics of the lacrosse shot," Biomechanics of the Lacrosse Shot, http://www.centerislandperformanceathletics.com/2016/03/biomechanics-of-lacrosse-shot.html (accessed Dec. 11, 2024).

[2] Esp32 series, https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (accessed Dec. 11, 2024).

[3] Utsource and Instructables, "Getting started with ESP32 CAM: Streaming video using ESP cam over WIFI: ESP32 security camera project," Instructables, http://www.instructables.com/Getting-Started-With-ESP32-CAM-Streaming-Video-Usi/. (accessed Dec. 11, 2024).

[4] Lacrosse Monkey, "Lacrosse passing guide: How to throw a lacrosse ball," LacrosseMonkey.com, https://www.lacrossemonkey.com/learn/how-to-pass-lacrosse-ball (accessed Dec. 11, 2024).

[5] A.-G. Sports, "Lacrosse shooting: A guide to increased shot speed and a quicker release - A-game sports," A, https://agamesports.net/2017/09/04/lacrosse-shooting-guide-increased-shot-speed-quicker-release/ (accessed Dec. 11, 2024).

[6] "D1 boards," D1 Boards - WEMOS documentation, https://www.wemos.cc/en/latest/d1/index.html (accessed Dec. 11, 2024).

[7] "MPU-9250," TDK InvenSense, https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/ (accessed Dec. 11, 2024).

[8] Adafruit, https://cdn-shop.adafruit.com/datasheets/ST7735R_V0.2.pdf (accessed Dec. 11, 2024).

Appendix A: Requirement and Verification Tables

1. LaxHub Requirements and Verification Table

Requirements	Verification	Verification Status
• When the hub system is active and detects that a swing has happened, it should complete internal processing and display basic output on the LCD screen within 10 seconds of receiving data	• First confirm the LaxHub and LaxSense subsystems are both active and connected (look for a "connected" status on screen) • Once in range and in frame of the LaxHub, do a test swing • Wait to see a processing message on the screen • Confirm that predictions are outputted within 10 seconds	Yes
• If any communication failure occurs between the LaxHub and LaxSense device subsystems when they are active in a waiting state, the LaxHub should let the user know through the LCD screen within 5 seconds of losing the connection, and attempt to reconnect automatically afterwards	• Initially confirm the LaxHub and LaxSense subsystems are both active and connected (look for a "connected" status on screen) • Temporarily turn off the LaxSense subsystem,possibly by disconnecting its battery • Wait for 5 seconds and check to see a disconnected message on the LaxHub LCD screen • Now reconnect the LaxSense battery and make sure it turns on • Wait and check for a reestablished connection (back to a "connected" status on the screen)	Yes
• If the battery percentage of the LaxHub system falls below 15%, the LaxHub should detect this and display a low battery warning on the LCD screen	• First connect a new fully charged battery to the system and turn it on for standard use • Calculate approximately how long the battery should take to get to 15% by looking at overall current draw • Wait for this amount of time, and expect to see a battery warning message on the	Yes

	screen • Calculate approximately how long the battery should take to run out with 15% left • Wait for this amount of time, and make sure the system dies from battery loss	
• When the system is alive and ready to use the camera, it should display a “Camera On” indicator on the LCD screen; In addition if the Camera is obscured or not reading frames it expects to, it should display a camera error to the screen within 5 seconds	• Initially confirm the LaxHub and LaxSense subsystems are both active and connected (look for a “connected” status on screen) • Ensure the LaxHub unit is turned on and the LCD display initially shows a “Camera On” status • Cover the camera temporarily with a screen or cover so it is unable to see the lacrosse player • Wait 5 seconds and make sure we see a “Camera error” message on the screen	Yes
• When the LaxHub has received and transmitted data to the cloud, display a message to show the data has been sent for processing, ultimately routing the user to a more detailed breakdown in the application	• Initially confirm the LaxHub and LaxSense subsystems are both active and connected (“connected” status) • Demonstrate a normal use of the system by doing a practice swing • Expect to see a “Data sent to cloud” message on the LCD screen • Later check the TripleS application has updated with more data	Yes

2. LaxSense Requirements and Verification Table

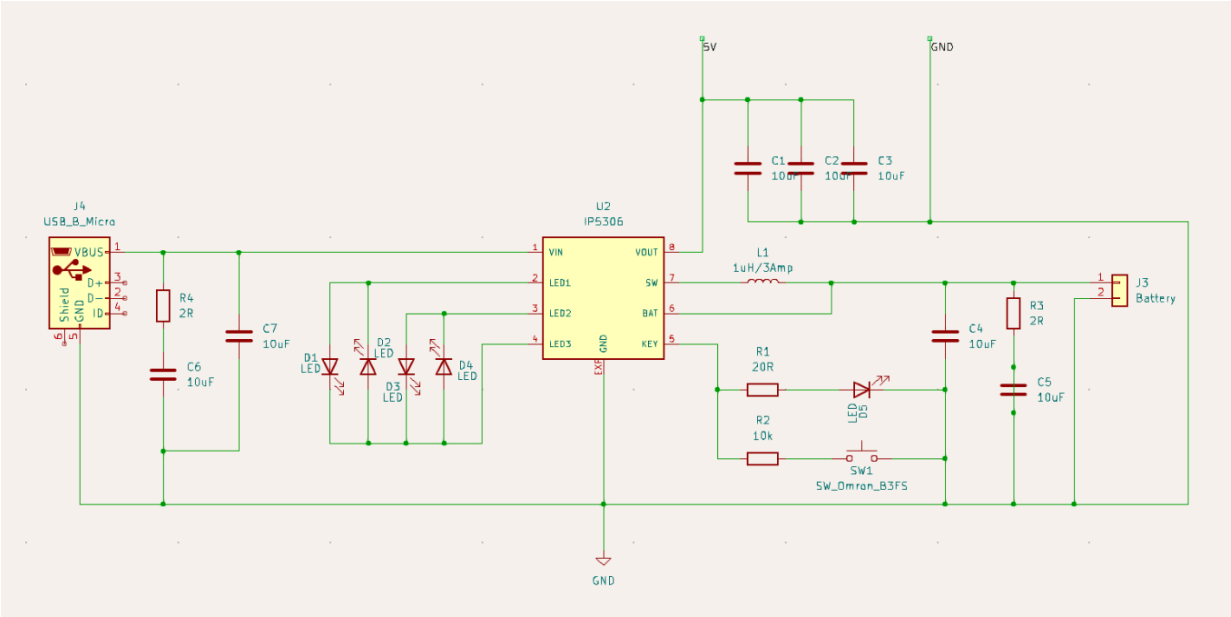
Requirements	Verification	Verification Status
• If the bluetooth BLE connection is lost when the subsystems are in use, the system should attempt to reconnect within 5 seconds	• Establish a BLE connection between LaxSense and LaxHub. • Turn off the LaxHub momentarily, turn it back on, and verify that the system reconnects automatically	Yes
• Only when the LaxSense detects a shot, denoted by rapid acceleration, 15 ± 3 m/s^2, it must log and transmit the data to the LaxHub within 10 seconds. This is to prevent any unnecessary logging.	• Simulate a shot by moving the stick in a fast, forward motion • Confirm that LaxHub received the data transmitted by LaxSense within 10 seconds (timer) by observing the logs on LaxHub. • Additionally, move the stick lightly and verify that no log has been sent to the LaxHub unit	Yes
• If there is a critical miscommunication between the LOLIN D1 Mini and the MPU-9250 sensor, the subsystem must stop transmitting data and log an error on the LaxHub	• To simulate a failure, we can interrupt the I2C connection by disconnecting the sensor unit from the microcontroller. • Make sure that the system stops transmitting data once the communication error is detected - this is needed so that LaxHub does not receive false readings • Verify that an error is logged on LaxHub to maintain the integrity of data	Yes
• The system must log a warning on the LaxHub if the voltage drops below 3.7 volts	• Simulate low battery conditions by using a variable power supply • Confirm that a warning is logged on the LaxHub when the voltage falls below 3.7 volts	Yes

3. TripleS App Requirements and Verification Table

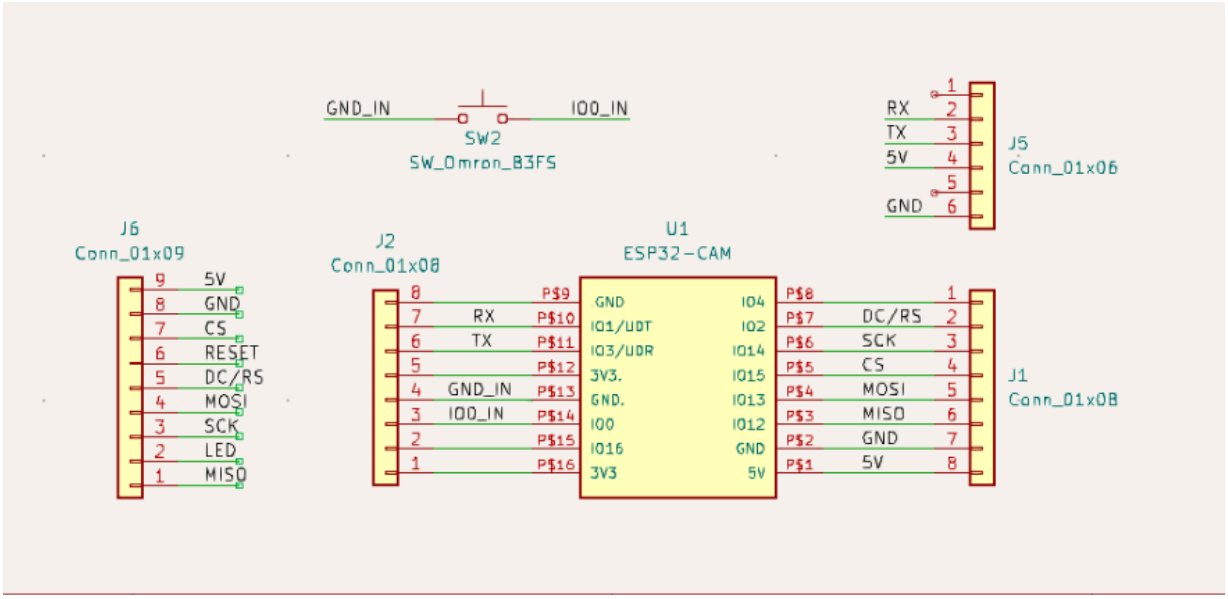
Requirements	Verification	Verification Status
• The end data should be sent to Amazon Kinesis Streams in JSON format within 30 seconds from creation time (readings from LaxSense)	• Simulate data from LaxSense, and track to see the time it takes for the data to travel from one subsystem to another • Confirm data transfer through logs sent to the microcontroller • Once the data reaches the cloud, verify it has been reached within 30 seconds since the shot start time	No
• Ensure that the Lambda function transmits data to DynamoDB that does not exceed 2 MB / per second per shard to avoid throttling.	• Deploy a temporary, test Lambda function that sends messages to DynamoDB • Configure AWS Cloudwatch on the Lambda function • Use the AWS Cloudwatch console to monitor the throughput and metrics and see if it is below the 2MB limit	Yes
• The format of the data sent from the Lambda functions to DynamoDB and retrieved from DynamoDB has to be consistent to prevent any data errors for calculations	• Conduct tests where data is sent to DynamoDB and also retrieved • This can be done by hard coding values via the LaxHub microprocessor to be sent to the Kinesis Streams • Validate that the data retrieved (client side mobile application) matches what is being sent	Yes

Appendix B: Subsystem Schematics

1.1. Schematic for LaxHub Subsystem: Main Skeleton



1.2. Schematic for LaxHub Subsystem: Connections



2. Schematic for LaxSense Subsystem