Smart Stick System (TripleS)

ECE 445 Design Document

Project 22

Ritvik Manda, Shivam Patel, Pranav Nair

Professor: Cunjiang Yu

TA: Dongming Liu

1 Introduction

1.1 Problem

Lacrosse players and coaches currently lack real-time, detailed performance metrics to help improve their gameplay. 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. Since lacrosse is not a well known sport, it becomes difficult for beginners and enthusiasts to start learning the mechanics of the stick and being proficient with it.

1.1 Solution

This project aims to address the need for a smart, data-driven tool that can measure shot speed, accuracy, and stick form, providing lacrosse players with accurate and instant feedback to enhance their training and technique. We seek to help experienced players to obtain performance data while also aiding beginners in strengthening their form and tactics. Our proposed system consists of a hub unit, a monitoring device fitted to the lacrosse stick, and a mobile app. The base or hub unit (known as the LaxHub) is the main processing unit, and will receive data from the remote unit as well as use a built in camera and lcd screen. We aim to use computer vision based processing to monitor a player's form, suggest changes, and track progress. A secondary subsystem is meant to be securely fit onto the end of a lacrosse stick (known as the LaxSense). This unit contains sensors to transmit information like swing speed and angle back to the LaxHub for processing and estimating metrics like ball speed and trajectory. Finally, the TripleS application provides detailed feedback based on LaxHub processing, and tracks the user's history making use of AWS storage and processing.

The end product is very user friendly and fits a completely exclusive niche in the world of sports performance tracking. It can help new and experienced users alike maintain, keep track of, and improve their lacrosse game. Because the full package is rather portable, with only a small box and minimal stick unit, it can be taken and used almost everywhere. We also plan to have rechargeable

batteries to add to this aspect, allowing players to take the system for use outside. Finally, with both an instantly updating built-in LCD screen as well as more detailed metrics on an app, we hope to give essential feedback quickly and clearly to assist in a training session.

1.2 Visual Aid

Figure 1: A standard lacrosse stick

A standard commercially available lacrosse stick is made of metal alloy and carbon fiber, and holds up the head (containing the net pocket to hold lacrosse balls). They are typically swung in a catapulting motion. We plan to use a standard lacrosse stick with the only main alteration being attaching a small module to the bottom to monitor information.

More specifically, lacrosse sticks are shaped in an octagon and typically are mostly hollow as seen in label A of Figure 2. They have a rubber stop at the end which we plan to remove and replace with our securely fitted monitor.

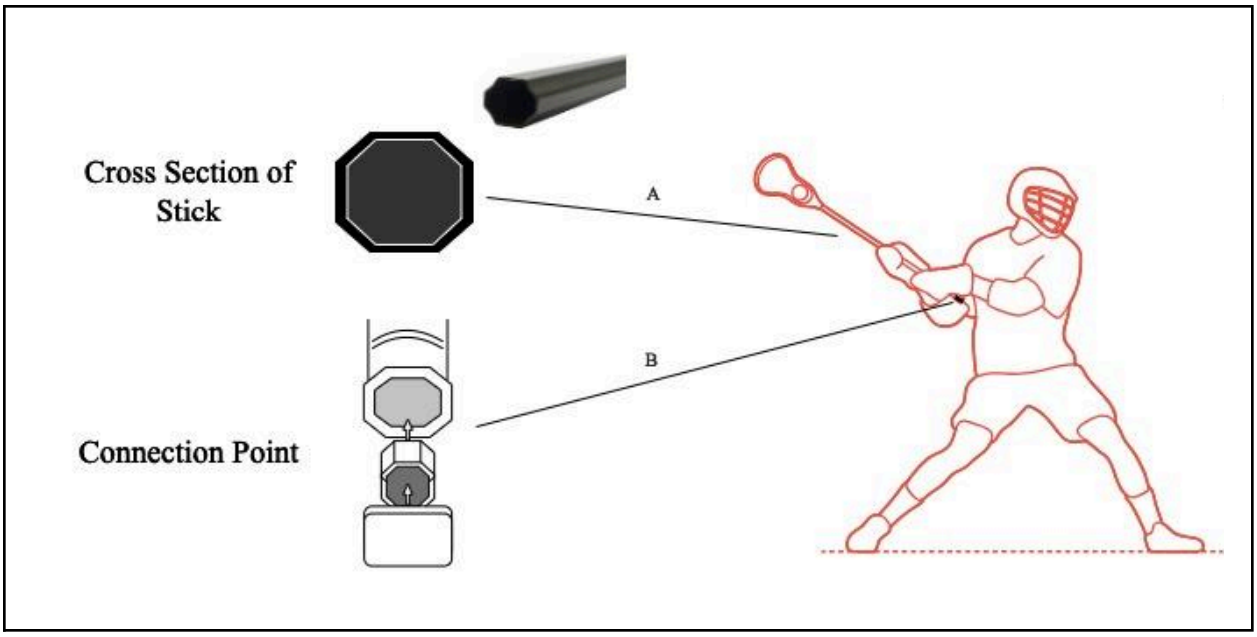

Figure 2: Visual Aid of the Stick System (LaxSense)

As shown more clearly in label B of Figure 2, our actual LaxSense unit (with a rectangular shaped 3D printed case) will be fixed with a hexagonal prism connector that can simply slot into the end of the lacrosse stick. Ideally we will not need any adhesive if this is a secure enough fit, meaning it can easily be swapped in and out for regular use or monitoring. Since the mounted components simply consist of sensors (gyroscope and accelerometer), a microcontroller, and a battery, we aim to house this in a small and lightweight container to avoid tampering with the player's feel for the lacrosse stick.

1.3 High Level Requirements

1. Our first requirement is **real-time performance tracking and analysis**, more specifically to provide feedback and improvements or failures of a swing within 30 seconds. The system must accurately measure and analyze metrics in near real-time, including shot speed, accuracy, and stick form. This delay also includes sending data to the cloud and outputting a more detailed response in the application.
2. Our second requirement is that the **accuracy of ball speed and trajectory** to be ±10 mph and within 10 feet respectively. It's important to measure this accuracy since players need to clearly see if their speeds and trajectories change over time. A user might want to keep track if

the speed of his throws become faster over time, while also maintaining a good trajectory.

3. Our third requirement is that the **LaxSense unit must weigh less than 3 ounces**. It's crucial to keep the unit light to avoid interfering with the mechanics of the lacrosse stick. If the LaxSense unit is too heavy, it could negatively affect shot speed and trajectory, which is not ideal. By keeping the unit under 3 ounces, we can ensure it doesn't disrupt the natural balance of the stick, allowing players to maintain optimal shot speed, accuracy, and overall performance.

2 Design

2.1 Physical Diagram

Below is a physical representation of our Smart Stick System. The modification of the lacrosse stick is present in the bottom portion, where the plug is located. The original plug will be replaced by our LaxSense system, which is a 3D printed plug with sensors.

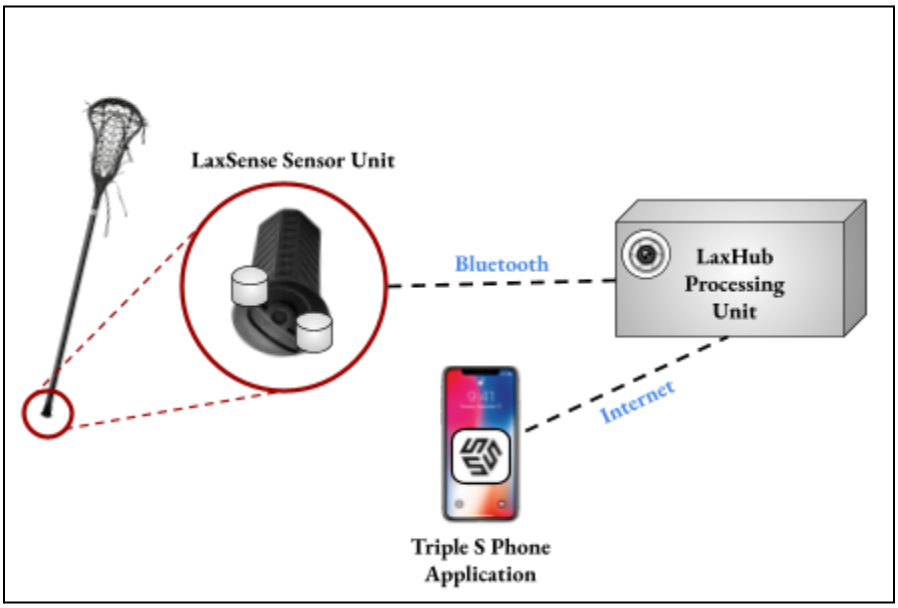

Figure 3: Smart Stick System (Triple S) Physical Diagram

The LaxHub processing unit will be placed on the ground when a player uses the lacrosse stick, tracking the motion and form of the player. The data measured from the LaxSense unit will be logged into the LaxHub. The data is then sent to our TripleS application via an AWS Kinesis agent.

2.2 Block Design

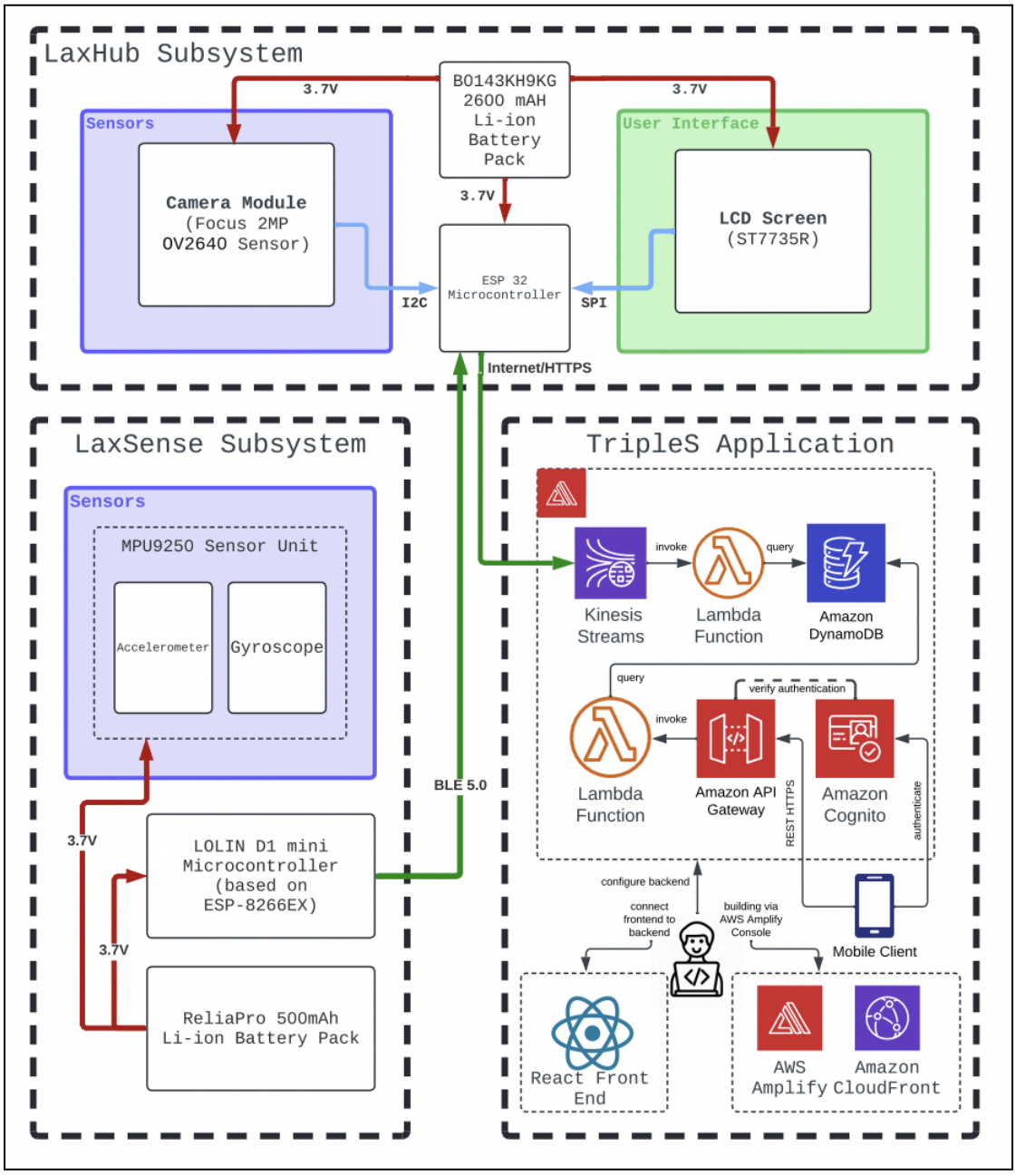

Figure 4: Smart Stick System (Triple S) Block Diagram

LaxHub is the main processing unit of this system and contains the custom PCB, microcontroller, LCD screen, and camera, as well as a bluetooth 5.0 module (as part of the microcontroller). The LaxHub will need to be powered by a rechargeable battery. The main peripheral connection of this subsystem is the low-power bluetooth connection with the LaxSense subsystem. It

also connects directly to the backend cloud component of our application via a Kinesis client. LaxSense is a subsystem that mounts on the back of the lacrosse stick, which will contain a microcontroller, accelerometer, and a gyroscope. These parts will work in conjunction to keep track of performance metrics such as shot speed, stick angle, and form. Because this is a standalone device, this will need to be powered by a small battery system. LaxSense directly sends data to the LaxHub via the built-in bluetooth module of the microcontroller. The TripleS application is our final subsystem, and handles data display and analysis. It receives data from the ESP 32 microcontroller through a Kinesis Client and places it in our database, which is DynamoDB in our backend. We will use React as our frontend and use AWS Amplify to help build the infrastructure. A mobile user will have access to this application and the data displayed will be from the very DynamoDB table that the microcontroller sends data to.

2.3 Functional Overview and Subsystems

2.3.1.1 LaxHub Description

The **LaxHub** consists of four main hardware components all connected through the PCB. Our main processing hub is the ESP 32 microcontroller. We specifically will use a ESP32-S3 which integrates a bluetooth 5.0 and wifi module. The microcontroller will handle communication with both other modules and requires this form of connectivity. More specifically, the microcontroller receives data from the LaxSense through low power bluetooth, and sends data to the backend cloud application by connecting to the internet. Receiving accelerometer and gyroscope data from the LaxSense will particularly be used in the LaxHub to predict data like throw speed and distance, and ultimately display on the local screen. Sending data to the application will be done by implementing Kinesis client code in the microcontroller. The next component is the battery unit. We chose a B0143KH9KG, 3.7V-2600mAh-9.62Wh Rechargeable Li-ion Battery Pack in order to have plenty of reserve battery and be able to charge it up when possible. The battery connects to the microcontroller, camera, and LCD screen, and needs to be able to supply a constant 500mA to all. The ESP32 microcontroller cannot function below 500mA. The camera module we will use is a 2MP OV2640

Sensor. It is a small camera option that is still able to capture 1080p at 30 ± 12 fps video for short frame by frame analysis. We need to account for some amount of frame drops limited at processing speed. The camera module is connected as an I2C device to the microcontroller and we will install the appropriate driver and devicetree to read and use its data. We will need a connection between these two components able to send 30 frames per second of data to the microcontroller. Similarly we have an LCD screen that will receive basic data to display from the microcontroller. We chose the ST7735R screen especially to have a SPI interface as opposed to I2C for faster data transfer and being able to display more critical information as quickly as possible. Most important connection points are simply MOSI (Master Out Slave In), SCLK (Serial Clock), and CS (Chip Select) on the SPI interface of the ESP32. Overall this is the most essential subsystem and as the hub of communication for all processes, it must use its connections with high rates of data movement to satisfy requirements of providing feedback as quickly as possible.

2.3.1.2 LaxHub Requirements and Verification Table

Requirements	**Verification**
• 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
• 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	• 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

automatically afterwards	• 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)
• 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 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
• 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	• Initially confirm the LaxHub and LaxSense subsystems are both active and connected ("connected" status) • Demonstrate a normal use of the system

user to a more detailed breakdown in the application	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

2.3.2.1 LaxSense Description

The **LaxSense** subsystem is a critical component mounted on the bottom of the lacrosse stick, responsible for gathering and transmitting performance metrics such as shot speed, stick angle, and form. It utilizes the MPU-9250 sensor unit, which integrates a 3-axis accelerometer, 3-axis gyroscope, and a magnetometer, working together to capture real-time data about the stick’s movement. The accelerometer measures the stick’s linear acceleration with configurable ranges of ±2g, ±4g, ±8g, and ±16g, ensuring precise tracking of shot velocity and force. Meanwhile, the gyroscope measures angular velocity with full-scale ranges of ±250, ±500, ±1000, and ±2000°/s, allowing for detailed analysis of stick rotation and form. The subsystem is powered by a 500mAh Li-ion battery, which must provide a continuous 3.7 ± 0.75 V output and support up to 5 hours of operation per charge. The MPU-9250 operates with low power consumption, so the connection from the battery to the sensor system must be a stable 3.5 ± 0.75 mA when all axes are enabled. The LOLIN D1 Mini microcontroller, based on the ESP-8266EX, processes the data and transmits it to the LaxHub via Bluetooth Low Energy (BLE). The microcontroller connects to the sensor system via I2C and so we must properly install the driver and devicetree for these sensors. The BLE connection to the LaxHub must maintain a stable data rate of at least 1 ± 0.5 Mbps with a range of up to 10 meters, to allow for real-time performance feedback without noticeable delays. The total weight of the subsystem must remain under 3 ounces to avoid affecting the lacrosse stick's balance. Any failure in sensor accuracy, bluetooth range, or power efficiency would cause the subsystem to fail, leading to the loss of data.

2.3.1.2 LaxSense Requirements and Verification Table

Requirements	**Verification**

• 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
• 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
• 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
• The system must monitor the battery voltage and log a warning on the LaxHub if the voltage drops below 3.7 volts	• Simulate low battery conditions by using a variable power supply to reduce voltage. • Confirm that a warning is logged on the LaxHub when the voltage falls below 3.7 volts

To connect the MPU9250 to the Lolin D1 Mini using I2C communication, we need to first connect the VCC pin of the MPU9250 to the 3.3V pin on the D1 Mini, the GND pin of the MPU9250 to the GND pin on the D1 Mini, the SDA pin of the MPU9250 to the D2 pin on the D1 Mini, and the SCL pin of the MPU9250 to the D1 pin on the D1 Mini. For powering the Lolin D1 Mini, we are going to use the 500 mAh Li-ion battery pack, ensuring the voltage is appropriate.

Connect the positive terminal of the battery to the 5V pin of the D1 Mini and the negative terminal to the GND pin. It's important to ensure that your Li-ion battery pack provides a voltage between 3.3V and 5V, as the Lolin D1 Mini has an onboard voltage regulator that can handle this voltage range.

2.3.3 TripleS Application Description

The frontend of the **TripleS application** is created using React, and we will use AWS Amplify to develop and deploy the mobile application. The TripleS application contains a singular DynamoDB database. Once the ESP 32 microcontroller has the necessary data, then this data is sent to Amazon Kinesis streams through a Kinesis Client. This is done by first setting up the AWS software development kit on the microcontroller, initializing the client, creating a data stream, and then sending the data in a JSON format. All of this is done via the Internet. The stream then sends the data to DynamoDB using a Lambda function which processes it. Users will go through Amazon Cognito to authenticate, there will be REST HTTPS communication for the API Gateway, which in turn will verify the authentication. This gateway then invokes a Lambda function which queries and reads the data from the DynamoDB table. One of our high-level requirements is that we aim for real-time performance tracking and analysis. To achieve this, we are using Amazon Kinesis streams which are excellent for real-time data streaming and we are using serverless architecture, such as Lambda and DynamoDB. Because of our choice of architecture, we should be able to get analysis and data under 30 seconds. The list of requirements such that if any of them are removed the subsystem would fail, are as follows: Since Kinesis has shard limits, each shard can support up to 1000 messages per second or 1MB per second, so if these thresholds are broken, it may lead to throttling and in turn, failures. Sending the data from Streams to Lambda is 2 MB per second (shared) per shard for all consumers, so if these thresholds are broken, it may lead to throttling and in turn, failures. We also have to ensure that the Lambda functions to send the data to DynamoDB and retrieving the data from the database both follow the same format, or else there will be errors in consistency regarding data.

2.3.1.2 TripleS App Requirements and Verification Table

Requirements	Verification
• 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
• 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
• 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

2.4 Software Design

Our core software component is the threaded processing done by the microcontroller within the LaxHub, along with further processing completed in a cloud instance before being sent to the application. Ultimately, the ESP 32 must simultaneously receive video data at a high rate, receive sensor data over bluetooth from the LaxSense, process basic data for display on the LCD screen, and send data to the cloud using a Wifi connection to university internet.

Figure 5: An OpenCV MPII Pose Tracking Overlay on an Example Lacrosse Pass

One process cycle begins with the devices simply set up to capture information (camera pointing at a person). An OpenCV MPII lightweight model will be running on the ESP 32 using captured frame data from the camera, essentially running “pose tracking”. This means the program is able to predict the “skeletal” position of people from video data, mainly keeping track of arms, legs, and joints. An example of how this pose tracking works is shown in Figure 5. Using a predicted motion here, as well as a significant change in threshold on the LaxSense accelerometer, the system triggers the start of a cycle. The frames for a full predicted throw motion are captured (~3 seconds or 90 frames) and recorded as raw position data through CV, ignoring all other data from the image like color, saturation, etc. The tracked position data for all of these frames along with sensor data from the LaxSense is immediately sent for cloud processing via an AWS Kinesis client. As soon as this is done, the ESP 32 sends a hard coded message “Data sent to cloud” on the LCD screen. For simplicity, to write to the LCD screen, the program will use Adafruit's ST7735 library, which is compatible with the ESP 32 microcontroller. Locally, we then begin processing the accelerometer and gyroscope data to predict speed, distance, and trajectory, explained further mathematically in the next section. With this information processed, we once again display it on the LCD screen. Ideally, we wait for a final pingback from the cloud indicating processing is finished, the database is updated, and the latest information can

be accessed from the app. From the perspective of the ESP 32 on the LaxHub, we now simply wait to restart the loop.

To predict the trajectory of the ball that would be released from a similar pass, we can first use an accelerometer threshold value to consider exactly when the shot begins and ends. Since we are using a threshold this will not be exact, but we will round lower so that it is more sensitive to any kind of shot. Now using the gyroscope and accelerometer, we can calculate the angular velocity and acceleration of the shot, mainly due to the pivoting nature of the lacrosse shot. More specifically, the angular velocity can be measured throughout the shot path simply by tracking over short periods of time. After recording a continuous function of velocity at the LaxSense microcontroller, acceleration is found by simply taking the mathematical derivative. Knowing the stick length and average angular velocity over the throw period, we use the equation $v = rw$ to estimate linear velocity. Now using linear release speed and angle measured using the gyro, the final trajectory can be predicted with the trajectory equation. We use

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

to correspondingly calculate the predicted distance and height trajectory values, and record/display them on the LaxHub unit. Ultimately this will have noise that might need to be filtered and these values will be estimates. Regardless, this estimated information being displayed almost immediately to a training player would be very helpful to see differences over time.

2.5 Component Selection

We selected each component primarily based on their functionality along with how compatible they are with other parts. Here we briefly discuss why we chose several of these components.
ESP32-S3 (ESP32-CAM) Microcontroller - We chose the ESP 32 microcontroller due to its accessibility, well documented uses, and size. The ESP32-S3 specifically matches our use cases of having to communicate through bluetooth, connect to a network through Wifi, and have compatibility with a camera unit. We are easily able to connect our camera sensor to this microcontroller with a ribbon cable, which will help significantly.

2MP OV2640 Camera Sensor - This camera sensor was chosen again due to its small size while being very capable. It is easily connected to the ESP 32, and is able to provide 30 fps video at 1080p resolution. While we likely will not need to use 30fps or a clear 1080p video, this ensures we are gathering more data rather than less which can later be compressed.

ST7735R LCD Screen - We chose this screen unit due to its small but readable size, and once again simple connectivity with the microcontroller. This screen uses an SPI interface, which as discussed earlier, is a great option for faster data transfer. This seemed crucial to us considering all the tasks the microcontroller must handle.

B0143KH9KG 3.7V-2600mAh-9.62Wh Rechargeable Li-ion Battery Pack - This battery pack is a great option we found for our system as it has enough of a capacity to power the LaxHub for several hours, and is rechargeable. From a product perspective, it is great to have a rechargeable battery pack instead of having to find and replace new batteries each time after a few uses of the system.

LOLIN D1 Mini Microcontroller - On the LaxSense side, we very much prioritized lightweight and small components to fit within our criterion of weight (to avoid changing the feel of the lacrosse stick). This microcontroller, which is adapted from the ESP8266, is very small but powerful enough to process and transmit sensor information. It includes a bluetooth module which we will use to send the information to the LaxHub.

MPU-9250 sensor unit - We chose this sensor unit because it comes with both sensors we need to use, a gyroscope and an accelerometer. It also includes a magnetometer, which we plan to not use in this project. We can very easily connect this unit to the LaxSense microcontroller as well through the SPI interface.

2.6 Tolerance Analysis

It is important to focus on the accuracy of the gyroscope in the LaxSense subsystem, which is crucial in measuring stick rotation, which directly impacts trajectory. Let's analyze the feasibility of meeting our requirement based on the given following information:

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

Now listed below are the steps and calculations to prove the feasibility of the MPU-9250 gyroscope sensor unit.

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

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

$$\text{Error in rotational measurement} = 0.02° / 90° = 0.022\%$$

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

$$v = \omega * r$$

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

It's good to note that the release point of the ball is not necessarily at the top of the stick and that not all of the stick's angular velocity is transferred to the ball. As we are placing our hands towards the bottom of the stick, the pivot point of the catapult-like action changes as well. To achieve a more realistic calculation, we can do the following:

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

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

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:

1. Temperature drift: 0.75%
2. Calibration error: 1.25%

Step 6: Calculate the total error via the additional errors

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

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

The calculated error of 1.18 mph is still well within our high-level requirement of ±10 mph accuracy for shot speed measurement. It's also important to note that the 80.7 mph falls well within the 70-100 mph for a typical lacrosse shot. This analysis shows that the gyroscope in the MPU-9250 is sufficiently accurate for our application, even when considering a more realistic shot speed and mechanics, and with slightly increased error factors for temperature drift and calibration.

3 Cost and Schedule

3.1 Cost Analysis

The total cost for parts, as seen in the Bill of Materials below, is $92.75 before shipping. A 5% shipping cost adds an additional $4.64, bringing the total to $97.39. In Champaign County, a 9% sales tax on the parts cost adds another $8.34, resulting in a total of $105.73 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 $14,505.73. Therefore, the overall total cost for this project, including materials, shipping, sales tax, and labor, amounts to $14,505.73.

	Name	Description	Quantity	Cost
1	ESP32-S3 Microcontroller	Espressif microcontroller for IoT applications	1	$12.99
2	2MP OV2640 Camera Sensor	OmniVision 2MP camera sensor for capturing images	1	$8.99
3	ST7735R LCD Screen	Adafruit 1.8" TFT LCD display	1	$9.95
4	B0143KH9KG Li-ion Battery Pack	Odec 3.7V-2600mAh rechargeable battery	1	$12.69
5	LOLIN D1 Mini Microcontroller	LOLIN mini Wi-Fi microcontroller	1	$14.99
6	MPU-9250 Sensor Unit	InvenSense 9-axis motion tracking sensor	1	$14.59
7	Li-ion Battery Charger	Adafruit charger for Li-ion batteries	1	$5.99
8	FTDI FT232HQ	USB to UART interface chip for serial communication	1	$4.57
9	EEMB 3.7V LiPo Battery 500mAh	403048 Lithium Polymer rechargeable battery.	1	$7.99
10	Amazon Kinesis Client SDK	Amazon's real-time streaming data service SDK	N/A	$0.00
11	DynamoDB Database	Amazon NoSQL database service	N/A	$0.00
12	AWS Lambda	Amazon's serverless compute service	N/A	$0.00

3.2 Schedule

Week	Task	Member
October 6th → October 12th	Order parts to start prototyping	Everyone
	Take measurements of the parts and start CAD design for LaxSense	Ritvik
	Research communication between microcontrollers and subsystems	Shivam, Pranav
	Start PCB Design and get it reviewed	Everyone
October 13th → October 19th	Print 3D prototype	Shivam
	Test 3D prototype to see if sensors fit in the module	Pranav, Ritvik
	Finish PCB Design	Everyone
	First Round PCB Design - Pass Audit	Everyone
October 20th → October 26th	Continue working on CAD Design	Shivam
	Start developing openCV model for the camera	Pranav
	Start building LaxSense and LaxHub	Shivam, Ritvik
	Finish CAD Design	Ritvik
	Second Round PCB Design - Pass Audit	Everyone
October 27th → November 2nd	Research best practices for building AWS based app	Pranav, Ritvik
	Start building the Triple S application via pipeline	Ritvik, Shivam
	Start connecting LaxSense and LaxHub	Shivam, Pranav
	Beta testing for Laxhub	Everyone
	Third Round PCB Design - Pass Audit	Everyone
November 3rd → November 9th	Beta testing for application and openCV model	Everyone
	Continue testing subsystems for faults	Everyone

	Fourth Round PCB Design - Pass Audit	Everyone
November 10th → November 16th	Finish application creation	Ritvik, Pranav
	Finish openCV model	Pranav, Shivam
	Finalize Assembly	Everyone
	PCB Design Revisions	Everyone
	Fifth Round PCB Design - Pass Audit (finalize PCB design)	Everyone
November 17th → Mock Demo Day	Fix minor bugs	Everyone
	Demo	Everyone

4 Ethics and Safety

Our lacrosse performance tracking system raises several important ethical and safety considerations that we must carefully address throughout development and deployment. Our system collects and processes personal performance data of players, and in accordance with the IEEE Code of Ethics principle to "respect the privacy of others", we will implement data protection measures. This means allowing users full control over their data (including the ability to delete it), only collecting data necessary for the system's core functionality, and clearly communicating our data practices to users. Transparency is another key ethical principle we will adhere to, following the ACM Code of Ethics principle of honesty. We will clearly communicate the capabilities and limitations of our system, provide accurate information about the accuracy of our measurements, and be transparent about any data sharing practices to ensure “full disclosure of all pertinent system capabilities, limitations, and potential problems”. For example, we would be very clear about how distance and trajectory values are based on predictions, and have a high margin of error. We have to keep safety in mind as well for our product design. The LaxSense unit attached to the lacrosse stick must not have a chance of causing any physical danger to players. We will ensure the unit is securely attached and cannot come loose during play, use materials that are safe for skin contact and won't shatter if impacted, and design the unit to minimize risk of injury if a player falls on it. We plan to use a soft plastic 3D-printed shell, which will help it stay safe and lightweight.

Electrical safety is also a concern, as both the LaxHub and LaxSense contain electrical components. Specifically, the lithium-ion batteries we plan to use can pose a fire hazard. We will comply with IEC 60950-1 for IT equipment safety, ensure all batteries are properly enclosed and protected from impact, and use low-voltage components where possible to minimize electrical hazards. In terms of regulatory compliance, we will ensure our product complies with relevant regulations, including FCC regulations for wireless devices, and Consumer Product Safety Commission guidelines for sports equipment.

5 References

"Biomechanics of the Lacrosse Shot." *Biomechanics of the Lacrosse Shot*, Center Island Performance Athletics, www.centerislandperformanceathletics.com/2016/03/biomechanics-of-lacrosse-shot.html. Accessed 3 Oct. 2024.

Esp32 Series, Espressif, www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Accessed 3 Oct. 2024.

"Getting Started with ESP32 CAM: Streaming Video Using ESP Cam over WIFI: ESP32 Security Camera Project." Instructables, Instructables, 22 Feb. 2022, www.instructables.com/Getting-Started-With-ESP32-CAM-Streaming-Video-Usi/.

"Lacrosse Passing Guide: How to Throw a Lacrosse Ball." LacrosseMonkey.Com, Lacrosse Monkey, 15 July 2024, www.lacrossemonkey.com/learn/how-to-pass-lacrosse-ball.

Lacrosse Shooting: A Guide to Increased Shot Speed and a Quicker Release, A-Game Sports, 13 Aug. 2018, agamesports.net/2017/09/04/lacrosse-shooting-guide-increased-shot-speed-quicker-release/.

"Lolin D1 Mini." LOLIN D1 Mini - WEMOS Documentation, www.wemos.cc/en/latest/d1/d1_mini.html. Accessed 3 Oct. 2024.

"MPU-9250." TDK InvenSense, 3 Nov. 2023, invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/.

Rabil, Paul. Lacrosse Passing Guide: How to Throw a Lacrosse Ball, YouTube, 29 Nov. 2017, www.youtube.com/watch?v=TnsMrcQ0H-M.

ST7735R, Adafruit, cdn-shop.adafruit.com/datasheets/ST7735R_V0.2.pdf. Accessed 3 Oct. 2024.