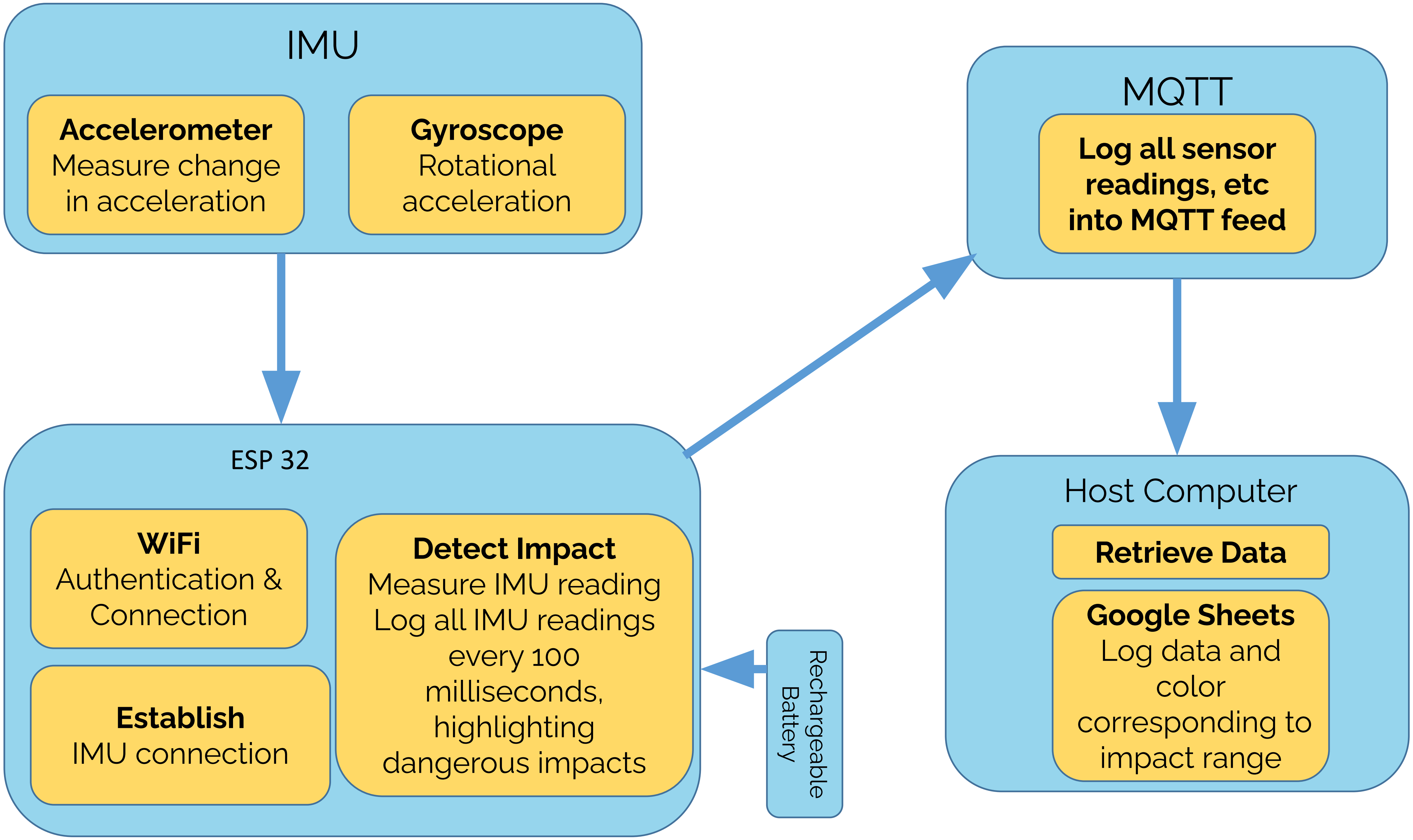
Impact Detecting Football Helmet

ME100 Final Project

Michael Salamy & Sean Hayes

Video Link: https://www.youtube.com/watch?v=mBKLO9NC9D0


IMU
Accelerometer
Measure change in acceleration
Gyroscope
Rotational acceleration
MQTT
Log all sensor readings, etc into MQTT feed
ESP 32
WiFi
Authentication & Connection
Establish
IMU connection
Detect Impact
Measure IMU reading
Log all IMU readings every 100 milliseconds, highlighting dangerous impacts
Rechargeable Battery
Host Computer
Retrieve Data
Google Sheets
Log data and color corresponding to impact range

Final Sensing Implementation

The sensors we used were the accelerometer and gyroscope as part of the MPU9250 provided in our kits.

Accelerometer

- We **measured linear acceleration in Gs** since that is the main unit used to quantify concussion impact.
- The Z-axis reading had to be **calibrated to start 1g lower than the initial reading**, since when still, gravity causes 1g to register as the Z-axis reading.
- The default from lab 7 was set to read ±2g, which we re-calibrated to read the maximum possible for the IMU9250, which is **±16g**.

Gyroscope

- We **measured rotational velocity in degrees per second**, since this a common unit in quantifying severity of a concussion (corresponds to a rotational acceleration as most hits take between 1 and 2 tenths of a second).
- The default from lab 7 was set to read ±250 degrees/second, which we re-calibrated so that the gyroscope would read the maximum of **±2000** degrees/second for the MPU9250.

⚠ While the typical concussion impact makes the head accelerate at least 60g linearly and over 15000 degrees/second, our model is simply a prototype limited by the range of detection in the IMU9250. However, **our project still exemplifies how data on unsafe hits can be detected and transmitted**, just simply scaled down due to the resources available to our team.

Final Sensing Implementation (cont.)

Our team found success in the IMU's performance and in validating its measurements.

Validation of Measurements

- For the accelerometer, one positive was seeing that the **initial reading of the z-axis was 1g** when upright, as it should be due to gravity.
- For a sanity check, we tested hitting the prototype with **increasing magnitudes** of force, making sure that the accelerometer and gyroscope **values rose accordingly**.
- We verified working parts by testing **similar impacts with the IMUs from both of our lab kits**, which yielded similar results for similar impacts.

Expected Performance

- Our team expected the IMU to be able to easily **detect varying impacts with a high degree of precision** after doing lab 7, which proved to be true.
- We expected to only be able to sense impacts every few seconds due to various sources of delays, but we found out we were actually **able to continually sense impacts every 100 milliseconds**, making for an effective model (see more later).
- Therefore, we were able to measure the x, y, and z linear accelerations and the x, y, and z rotational velocities.

Output/Actuation Implementation

The prototype has both a local and IoT output to notify display when a potentially risky impact has taken place.

Local Output

- **On board**, the script running on the ESP32 goes through if-statements in order to **register the range of impact** strength.
- Thus, linear acceleration and rotational velocity readings instantly cause the **LED corresponding to the correct range** to light
- If the accelerometer measures linear acceleration of one color and the gyroscope measures rotational velocity of a different color, both colored LEDs light up.

IoT Output

- MQTT sends the information to a Google sheet, which can be viewed from any device with internet access.
- **Cells** in the spreadsheet are **automatically highlighted with the same color** corresponding to the range of measurement and thus the LED that lit up locally on board.

In our model:

- The **green** light corresponds to a magnitude of **0-1000 degrees/second or 2-4g**.
- The **yellow** light corresponds to a magnitude of **>1000-1400 degrees/second or >4-6g**.
- The **red** light corresponds to a magnitude of **1600+ degrees/second or 6g+.**

Output/Actuation Implementation (cont.)

Our team found success in the IMU's performance and in validating its measurements.

Expected Performance

- Our model **works as we expected** for both the local and IoT outputs.
- The LEDs corresponding to the proper color light up seamlessly, with multiple colors lighting up if the gyroscope and accelerometer readings fall into different colored categories.
- The data transmission through the MQTT feed to the Google Sheet occurs even faster than we imagined – instantly after impact.

Application Needs

- The output is perfect for our need of detecting and saving impact information from football hits.
- The on-board LED signifies could signify that the hit was dangerous and to check the Google Sheet.
- The Google Sheet then gives the precise impact measurements to **determine if a player should be removed** from a game and/or **checked for a concussion.**

Final Communication Implementation

Our communication implementation consists of using MQTT to log data in a Google Sheet.

Protocol

Impact data is measured by the MPU9250 and logged to a Google Sheet through **using MQTT** as our protocol.

Initially, our team wanted to also use IFTTT in order to make use of the possible Applets; however, this turned out to be unnecessary for our project.

The code enables data to be logged with MQTT in the order of milliseconds, logging all impacts to the prototype.

MQTT was the right choice for us because it allowed us to seamlessly fulfill our project purpose without overcomplications.

Remote Data Log

In order to log data for tracking impacts for a given player, using the data for medical studies, etc., our team decided to **log data to a Google sheet**, which automatically color-codes hits in different ranges of linear acceleration and rotational velocity.

Initially, our team wanted to transmit data to a MySQL compatible database. However, the **universality of Google sheets** and the ability to automatically color cells made it a better option. The data can always be easily imported to MySQL workbench if needed.

Google sheets was the appropriate choice in effectively tracking the data for the widest range of uses possible.

Link to Google Sheet Our Team Used in the Demo:
https://docs.google.com/spreadsheets/d/12Yc_oGTJ1AxxpA1hJ2iSmWmtyB_RIwZ2tkc1izAvtdw/edit#gid=0

Final Communication Implementation (cont.)

While minor challenges and issues exist, our team believes they are outweighed by the benefits of our communication model.

MQTT Challenges/Potential Issues

- The amount of data sent over MQTT could be potentially minimized for storage optimization.
- **Losses in Wi-Fi** on a football field would cause MQTT to stop working, potentially **missing unsafe hits** on the field

Google Sheets Challenges/Potential Issues

- A **Google developer account** is still **needed** to access the data (Berkeley provides)
- Google Sheets API has a request limit, which is 100 requests per 100 seconds.
- Wi-Fi/Service losses in the area the spreadsheet is being viewed from could cause live data to be delayed in appearing on the screen.

Overall Benefits

- While the ESP32 and obviously the prototype must be local, the use of data sheets allows this data to be viewed virtually anywhere from any device that has internet (phone, tablet, etc.).
- The **accessibility and universality promotes collaboration** between those who would use the data, meaning that the data can be used effectively for research purposes.

Final Computation Implementation

An overview of our code displays how our team registered and transmitted impact data.

Code Summary

- The code ran on the ESP32 was adapted from Lab 7 with the measurement ranges expanded, as previously discussed.
- The spreadsheet code runs on the computer in a loop, “listening” to the MQTT feed and formatting the data

Timer Usage

- Timer callbacks were utilized on the ESP32 to turn on the lights for **1000 milliseconds** to **avoid sleeping the entire device**.
- Had we not done this, there would likely be **times** when the IMU9250 was **not sensing**, leaving a player vulnerable to getting hit without knowing that the impact could have caused a concussion.
- The use of multiple timers allowed us to nearly cover all time so there is little risk when a player can get hit without a reading being sent.

Notable Functions

checkvals(Timer) : Sources data from the IMU and checks ranges to send to MQTT while also triggering the proper LED to light up.

data(c, u, message) : Formats data from MQTT into a row with date and time, uploads row to Google spreadsheets, highlights cells green, yellow, or red based on thresholds.

Final Computation Implementation (cont.)

Our team utilized Micropython to create our prototype's functionality and a rechargeable battery to power it.

Platform Used

- Our team used **Micropython** due to its versatility and our experience using it from this class
- Our team was also much more comfortable with the general Python coding language, as opposed to the C++ coding language.
- Our example script from Lab 7 used Micropython, so simply adjusting that for our purposes made the most sense.

Power Management

- To **optimize sensitivity**, we opted out of providing the device with a sleep mode.
- In a future development, a sleep mode may be implemented to **optimize power** if sensitivity levels can be maintained for player safety
- Our team decided to use a **rechargeable battery** with a battery life longer than the typical football game to power our prototype.

What We Learned

- Our team got practice using theoretical knowledge from lectures to implement functionality of a real-world, tangible device with Micropython scripts.
- Next time, we would try to minimize the amount of code on the ESP32 and run more processing on the external computer to optimize the speed of the device. A future implementation may utilize mechanical actuation to further protect the player's head in the case of extreme impact.

Conclusion

Overall, our team learned a lot from this project and is interested in extending and expanding on it in the future.

What Did/Did Not Work

Overall, what worked was...

- Our team was able to fulfill our purpose of detecting and transmitting unsafe impact hits.
- Readings are transmitted in real-time without fail.

What did not work was...

- Our team could not figure out how to have the script run and log impact data faster than every 100 milliseconds, which is potentially enough time for a hit to occur and be missed.

Next Steps

- Our team would attempt to acquire an **IMU that could measure the magnitudes** needed for **actual concussions** in order to create a more realistic and usable model.
- We would also look to use smaller parts and package the model together so that it could be installed on the **inside of a helmet** without sacrificing player comfort or adding significant weight.
- Later implementations may **protect the player in the case of extreme impact**. This may take the form of a speaker within the helmet to warn the player, or even mechanical actuation to protect their head.