Review the assignment due date

final-project-skeleton

Team Number: 23

Team Name: CJK

Team Member Name	Email Address
Christian Durante	cgd2@seas.upenn.edu
Jingyi Li	jingyii@seas.upenn.edu
Kendrick Xie	kenxie@seas.upenn.edu

GitHub Repository URL: https://github.com/upenn-embedded/final-project-f25-f25_final_project_t23_cjk

GitHub Pages Website URL: [for final submission]*

Final Project Proposal

1. Abstract

This project aims to design a wearable activity recognition system that detects and displays several distinct human movements such as running, jumping jacks, and sitting. Using data from IMU sensors mounted on the wrists, the system processes motion data in real time on an ATmega328PB Xplained Mini board and classifies the activity through a lightweight embedded model. Individual repetition tracking and overall session progress will both be communicated through an LCD screen as well as a speaker.

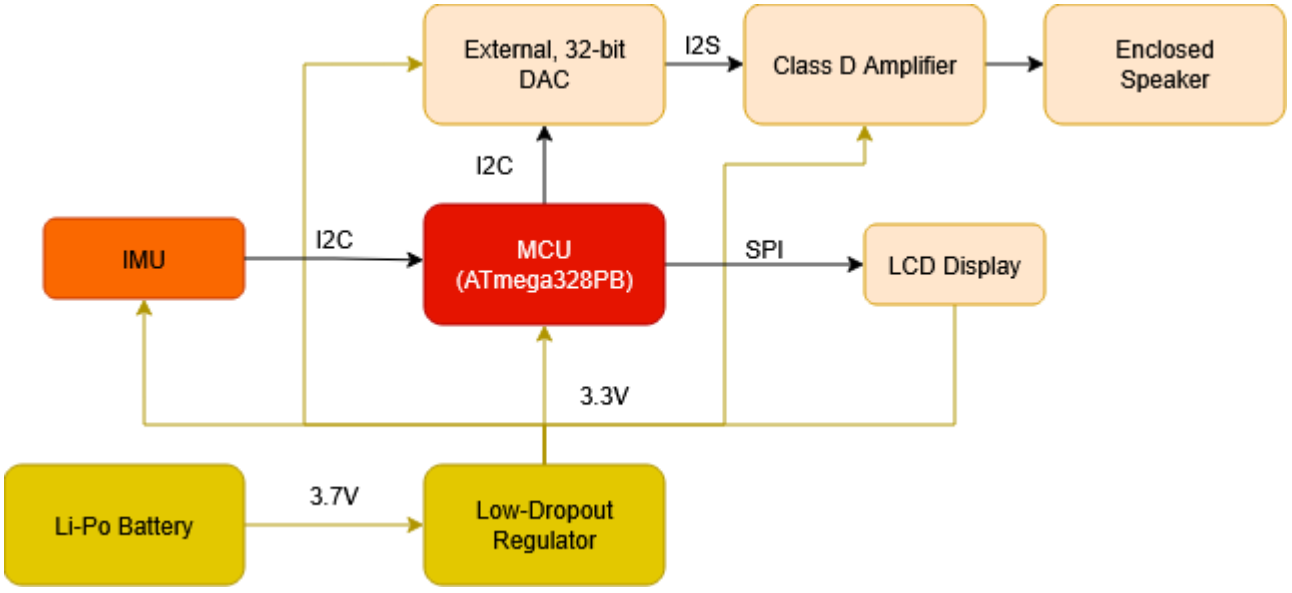
2. Motivation

Team CJK aims to create an energy-efficient, low-cost platform that recognizes basic human activities locally without needing a smartphone or cloud connection. This system will demonstrate how machine learning concepts can be applied in constrained embedded environments for health tracking, rehab monitoring, or gesture-based interfaces.

Many wearable fitness devices rely on large, pre-trained models or external cloud processing. This project explores an embedded, bare-metal approach to motion classification that runs entirely on a low-power microcontroller.

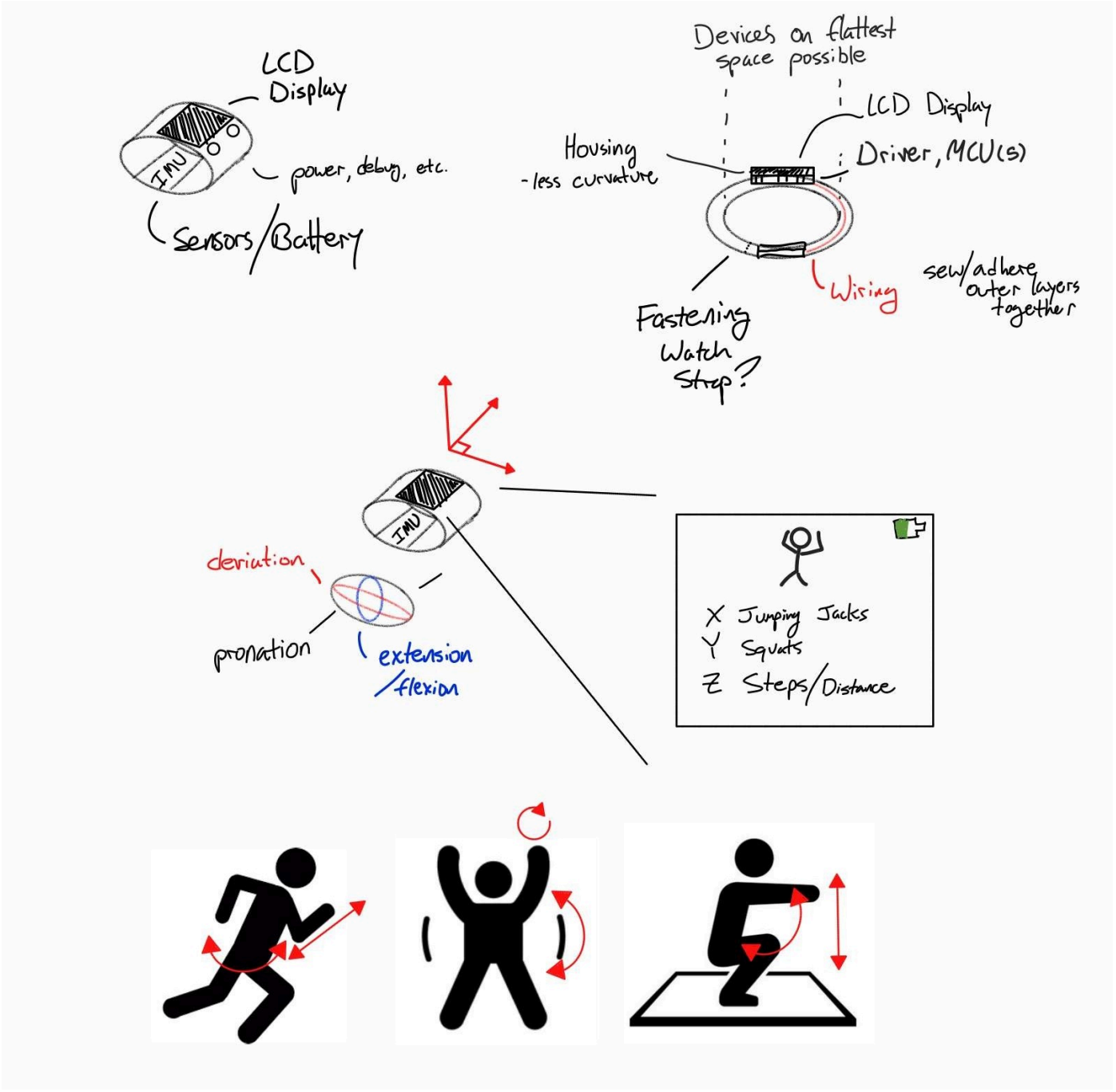
This problem was chosen to explore a specialized topic (Machine Learning) in the context of the courses established hardware tools. While this goal is drawn from a team member's specific background, the team's shared goal of creating a wearable device imposes a set of novel hardware challenges on top of what is otherwise a software-driven motivation. To this end, an additional audio feedback system integrates several hardware modules to provide additional feedback and ease of use.

3. System Block Diagram



4. Design Sketches

Little special manufacturing is required beyond soldering and 3D printing a simple enclosure for comfort and sensor stability. Some sewing or adhering may be needed to keep wiring protected between outer layers of the device.



5. Software Requirements Specification (SRS)

Formulate key software requirements here. Think deeply on the design: What must your device do? How will you measure this during validation testing? Create 4 to 8 critical system requirements.

These must be testable! See the Final Project Manual Appendix for details. Refer to the table below; replace these examples with your own.

5.1 Definitions, Abbreviations

- IMU – Inertial Measurement Unit (accelerometer + gyroscope)
- LCD – Liquid Crystal Display (ST7735R)
- ISR – Interrupt Service Routine
- Classifier/Software – Embedded decision tree classifier running on MCU

5.2 Functionality

ID	Description
SRS-01	The IMU 3-axis accelerometer and gyroscope shall be sampled at at least 100 Hz ±5 Hz using an I2C interrupt-based routine.
SRS-02	The MCU shall process every 0.5-second window of IMU data to compute statistical and frequency features with no overlap or interference between windows
SRS-03	The embedded classifier shall correctly identify at least 3 activities (running, jumping jacks, sitting) with >85% accuracy.
SRS-04	The current detected activity shall update on the ST7735R LCD screen within 1 second of the motion window being processed.
SRS-05	The software shall output at least one unique audio signal per identifiable activity type.
SRS-06	The software shall responds to at least one session-wide parameter (i.e. excessive break time) with a warning on at least one of the output peripherals.
SRS-07	The software shall allow easy addition of new activities by retraining and re-exporting the model header file.
SRS-08	The system shall operate continuously for at least 2 hours without requiring a reset or power cycle.

6. Hardware Requirements Specification (HRS)

Formulate key hardware requirements here. Think deeply on the design: What must your device do? How will you measure this during validation testing? Create 4 to 8

critical system requirements.

These must be testable! See the Final Project Manual Appendix for details. Refer to the table below; replace these examples with your own.

6.1 Definitions, Abbreviations

- I2C – Inter-Integrated Circuit communication bus
- SPI – Serial Peripheral Interface
- LDO – Low Dropout Voltage Regulator
- IMU – Inertial Measurement Unit

6.2 Functionality

ID	Description
HRS-01	The system shall use two 6-axis IMUs (accelerometer + gyroscope) connected via I2C for motion data collection.
HRS-02	The LCD (ST7735R) shall communicate with the MCU via SPI and display activity text updates at least twice per second.
HRS-03	The speaker system will produce at least one identifiable and audible (30-70dB) noise per identifiable activity type.
HRS-04	The ATmega328PB Xplained Mini shall sample and process data at 100 Hz while maintaining <75% CPU utilization and maximum interrupt latency<1 ms.
HRS-05	The power subsystem shall provide 3.3V regulated output from a 3.7V Li-Po battery with at least 500 mAh capacity.
HRS-06	The total current draw of the entire system (MCU + sensors + LCD) during continuous operation shall remain ≤ 150 mA
HRS-07	The system shall operate continuously for at least 2 hours at 35 °C without reset, crash, or data loss.
HRS-08	All modules shall be mechanically integrated within an enclosure. The IMU orientation error relative to the wrist reference frame shall be ≤ 10°.

7. Bill of Materials (BOM)

ONGOING SHEET HERE

Initial List:

Component	Description	Reason for Use
ATmega328PB Xplained Mini	Main MCU and development board	Core processing and interfacing platform
2×IMU Sensor (e.g., MPU6050 or LSM6DSO)	6-axis motion sensing	Detect acceleration and rotation
ST7735R 1.8” LCD	Display, controlled via SPI	Show recognized activity text/icon in real time.``Familiarity, can build upon graphical library from Lab4: Pong
Li-Po Battery (3.7V 500mAh)	Power supply	Portable operation
LDO Regulator (3.3V)	Power conversion	Stable voltage for MCU + sensors
Class-D Audio Amplifier	Signal treatment for speaker	All speakers of interest operate at prohibitively high wattage for ATMega to drive directly.
External DAC	Convert MCU output to audio format	Higher audio quality than attainable with 10-bit DAC on ATMega
Speaker	Secondary output	Additional feedback for user that interferes less with operation (a workout)
(Optional) 3D-printed case	Mechanical support	Mounting and protection
(Optional) Feather ESP32 Wi-Fi Module	Wireless Communication	Potential extension of project: real-time training (purely data transfer similar to controller example).Wireless

Component	Description	Reason for Use
		communication between wrist sensor modules.

8. Final Demo Goals

More details under Sprint Planning

On demo day, the prototype will:

- Be worn or held by a person performing **running, jumping jacks, and sitting.**
    - Library of movements subject to change as scope of 1 or 2 IMU modules is an open question at time of Proposal
- Provide Feedback:
    - Display the detected activity name (e.g., “RUNNING”, “SITTING”) on the ST7735R LCD in real time.
    - Play an audio signal between each movement repitiion, as well as addition, session-wide warnings or messages.
- Show smooth transitions between activities with stable recognition.
- Optionally, demonstrate the ability to add new activities (e.g., walking, standing) by uploading a retrained model.

9. Sprint Planning

You've got limited time to get this project done! How will you plan your sprint milestones? How will you distribute the work within your team? Review the schedule in the final project manual for exact dates.

Milestone	Functionality Achieved	Distribution of Work
Sprint #1	Software: Be able to read IMU data in some form Externally Train ML model, early	Kendrick has expressed a goal to focus on software.

Milestone	Functionality Achieved	Distribution of Work
	conversion to C Hardware: Power - still connected to XPlained/Computer Screen Operational Inventory Purchased Items, take action on any remaining purchases Drive Speakers	
Sprint #2	Software: Accurately recognize and count at least 1 gesture. Create low-fidelity audio messages Hardware: All MCUs and modules operational on one testing model (even if not wearable) Verify battery power operation, most debugging and testing still done on breadboard Bare minimum mounting - use purchased components to be able to wear a testing setup.So	Kendrick has expressed a goal to focus on software. Screen iteration and python/C interfacing are likely a unique role from enclosure/wearable design (moving beyond breadboarding to decrease hardware footprint).
MVP Demo	Software: Accurately recognize and count at least 2 gestures Maintain audio library to scale with recognizable gestures Hardware: Screen - show IMU data in real time for debugging. Safely wearable, some bulkier	Ditto Above

Milestone	Functionality Achieved	Distribution of Work
	components like boards/screen exposed / slightly inhibiting movement. Fully operational without external power (computer, lab bench)	
	Software Accurately demonstrate at least 3 gestures Debug Mode: show underlying IMU data, to better convey “moment of recognition” to verify processes of model. Audio signals for each movement type are relayed in real time. Hardware Comfortably Wearable: All major components contained throughout uncautious movement (teaching staff or other non-team personnel testing) Demonstrate/Test full battery life cycling Discuss/justify hardware sizing. Review Fully justify the decision to use 2nd wrist sensors or not.	
Final Demo		All team members should be able to discuss/justify the basic operation and choices of software used.

This is the end of the Project Proposal section. The remaining sections will be filled out based on the milestone schedule.

Sprint Review #1

Last week's progress

Sorted by functionality

- Implemented **IMU data logging** on the ATmega328PB
    - Reads accelerometer and gyroscope values from the LSM6DSO over I2C
    - Streams raw IMU measurements over UART at 9600 baud in clean CSV-like format
    - Verified stable and continuous data capture on the serial monitor
- Developed **Python-based data collection pipeline**
    - Script captures 10-second IMU sessions directly from serial
    - Automatically organizes data into folders by activity type
    - Produces consistent training data for machine learning
- Built an **initial activity classification model**
    - Extracts mean and standard deviation features from each IMU axis
    - Trains a lightweight Decision Tree classifier
    - Uses a 70/20/10 train/validation/test split
    - Demonstrates accurate separation of “sitting” vs “moving” activities
- Implemented **real-time Python activity detection**
    - Streams IMU data from serial
    - Uses a sliding 1-second window and same feature extraction as training
    - Continuously prints predicted activity in real time
- Ported **LCD Display Code**
    - Simplified Lab 4 code base to verify ability to drive LCD display
    - In the process, somehow damaged ATMega functionality
        - Progressed impaired by need to replace and re-solder another ATMega
    - Did not get to merge with Kendrick's IMU-reading scripts, code bases are pretty disogranized/seperated currently
- **Researched Audio Peripherals** - Connections and challenges for connecting audio peripherals

- **Parts have not arrived**, so preparing ahead of time to expedite development when they arrive.
- Identified repeated use of I2C, requiring use of select pins, etc. that previous labs have not quite required
- Detailed in Research Document (linked as part of current state)

Current state of project

(Embedded C – ATmega328PB)

- Initializes UART and I2C
- Configures the LSM6DSO IMU to stream accel/gyro data at 104 Hz
- Reads raw sensor values every 10 ms
- Outputs formatted IMU readings:`t_ms, ax, ay, az, gx, gy, gz`
- Serves as the primary data source for both training and live inference

`data_collection.py`

- Connects to the board over serial (`/dev/cu.usbmodem102`)
- Records **10-second data windows**
- Stores CSVs under: activities_data/`<activity>`/`<activity>`_N.csv
- Used to collect labeled datasets such as **sitting** and **moving**

`train_activity_classifier.py`

- Loads CSVs for all activities
- Extracts summary features:
- mean(ax…gz)
- std(ax…gz)
- Performs a **70/20/10** train/validation/test split
- Trains a `DecisionTreeClassifier(max_depth=3)`
- Saves model as `activity_tree.pkl`

`realtime_activity_detector.py`

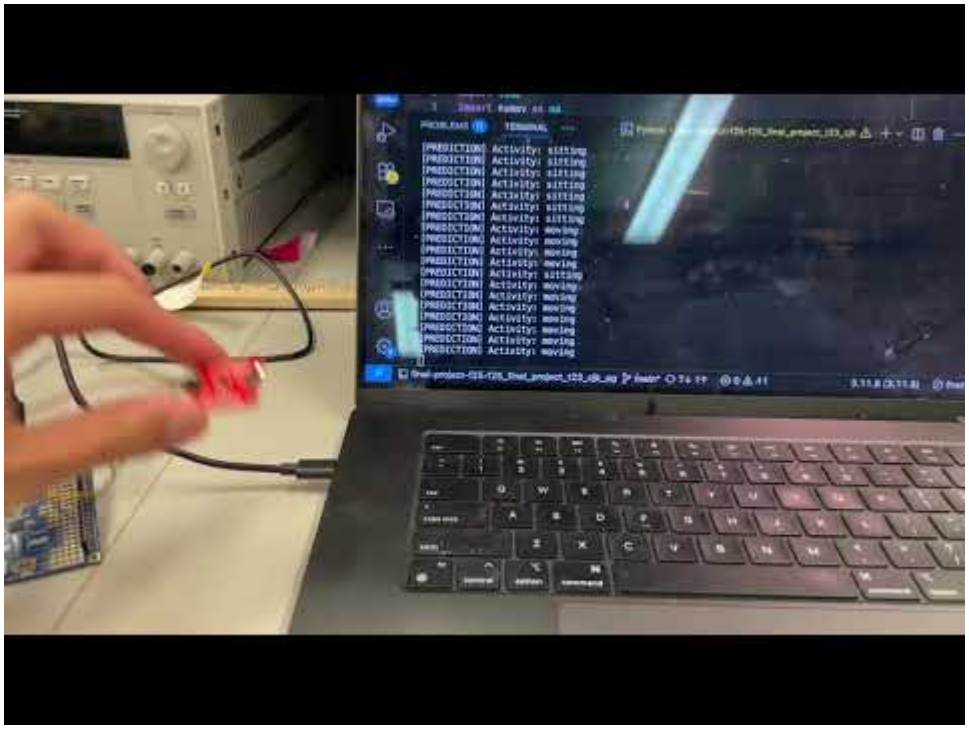
- Opens the IMU serial data stream
- Buffers a **100-sample (~1s)** sliding window
- Computes the same features used for training

- Loads the trained Decision Tree model
- Prints predicted activity every second, for example:
    - [PREDICTION] Activity: sitting
    - [PREDICTION] Activity: moving

Research Document/Reference for other systems - Audio, Wiring, etc.

Demo Video

Real-Time Activity Classification Demo



Specific Task Definitions

- Be able to read IMU data in some form
    - AHEAD, see next
- Externally Train ML model, early conversion to C
    - COMPLETE
    - essentially the "stretch" goal of the first, pipeline to python testing and early ML model implemented
- Screen Operational
    - COMPLETE

- Drive Speakers
    - AS COMPLETE AS POSSIBLE
    - Part has yet to arrive, performed some preliminary research.

Next Week's Plan

Addressing each point from the Proposal's plan for the week, identifying where we are ahead, behind

- Accurately recognize and count at least 1 gesture.
    - ON SCHEDULE
    - We have the pipeline of IMU data - python training established
    - Purely by the language of the requirement, we are determining one non-stationary state already, but clearly we can:
        - Have this functionality on-board
        - Convey it using our other periperhals
- Create low-fidelity audio messages
    - ON SCHEDULE* (we correctly predicted this having the largest ordering pipeline, so "actually drive the speaker" for week 2 if on schedule for now.)
    - Some risk this still stays in the realm of pseudocode
- All MCUs and modules operational on one testing model (even if not wearable)
    - AHEAD OF SCHEDULE: Simply need to route IMU data to Display as well as Python training.
- Verify battery power operation, most debugging and testing still done on breadboard
    - ON SCHEDULE - again, another "bare minimum" funcitonality for something that hasn't arrived yet appears like a major goal for week 2
- Bare minimum mounting - use purchased components to be able to wear a testing setup.
    - ON SCHEDULE - all on-hand peripherals funcitoning, low-fidelity glove arriving is all we need.

    - Even less risk of ordering slowing this task past week 2 since Amazon can all but guarantee timely shipping.

- Assess participation
    - This is mentioned here more to keep a paper trail rather than to attribute any blame / issue.
    - A team member has communicated a family matter that has prevented them from contributing, even to the point of keeping them out of town.
    - If this continues to be the case, teaching staff may need to step in to ensure the best outcome for the project/team.
        - The extent of time this has prevented full participation is concerning.

Sprint Review #2

Last week's progress

Current state of project

Next week's plan

MVP Demo

1. Show a system block diagram & explain the hardware implementation.
2. Explain your firmware implementation, including application logic and critical drivers you've written.
3. Demo your device.
4. Have you achieved some or all of your Software Requirements Specification (SRS)?
    1. Show how you collected data and the outcomes.
5. Have you achieved some or all of your Hardware Requirements Specification (HRS)?

    1. Show how you collected data and the outcomes.

6. Show off the remaining elements that will make your project whole: mechanical casework, supporting graphical user interface (GUI), web portal, etc.

7. What is the riskiest part remaining of your project?

    1. How do you plan to de-risk this?

8. What questions or help do you need from the teaching team?

Final Project Report

Don't forget to make the GitHub pages public website! If you've never made a GitHub pages website before, you can follow this webpage (though, substitute your final project repository for the GitHub username one in the quickstart guide): https://docs.github.com/en/pages/quickstart

1. Video

[Insert final project video here]

- The video must demonstrate your key functionality.
- The video must be 5 minutes or less.
- Ensure your video link is accessible to the teaching team. Unlisted YouTube videos or Google Drive uploads with SEAS account access work well.
- Points will be removed if the audio quality is poor - say, if you filmed your video in a noisy electrical engineering lab.

2. Images

[Insert final project images here]

Include photos of your device from a few angles. If you have a casework, show both the exterior and interior (where the good EE bits are!).

3. Results

What were your results? Namely, what was the final solution/design to your problem?

3.1 Software Requirements Specification (SRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
SRS-01	The IMU 3-axis acceleration will be measured with 16-bit depth every 100 milliseconds +/-10 milliseconds.	Confirmed, logged output from the MCU is saved to "validation" folder in GitHub repository.

3.2 Hardware Requirements Specification (HRS) Results

Based on your quantified system performance, comment on how you achieved or fell short of your expected requirements.

Did your requirements change? If so, why? Failing to meet a requirement is acceptable; understanding the reason why is critical!

Validate at least two requirements, showing how you tested and your proof of work (videos, images, logic analyzer/oscilloscope captures, etc.).

ID	Description	Validation Outcome
HRS-01	A distance sensor shall be used for obstacle detection. The sensor shall detect obstacles at a maximum distance of at least 10 cm.	Confirmed, sensed obstacles up to 15cm. Video in "validation" folder, shows tape measure and logged output to terminal.

4. Conclusion

Reflect on your project. Some questions to address:

- What did you learn from it?
- What went well?
- What accomplishments are you proud of?
- What did you learn/gain from this experience?
- Did you have to change your approach?
- What could have been done differently?
- Did you encounter obstacles that you didn’t anticipate?
- What could be a next step for this project?

References

Fill in your references here as you work on your final project. Describe any libraries used here.