Review the assignment due date

final-project-skeleton

Team Number: T7

Team Name: VeloGuard

Team Member Name	Email Address
Jiaan Zhang	jiaanz7@seas.upenn.edu
Tiancheng Pu	ptc1018@seas.upenn.edu
Zibo Zhao	zhao226@seas.upenn.edu

GitHub Repository URL: https://github.com/upenn-embedded/final-project-f25-f25_final_project_t7_veloguard

GitHub Pages Website URL: [for final submission]*

Final Project Proposal

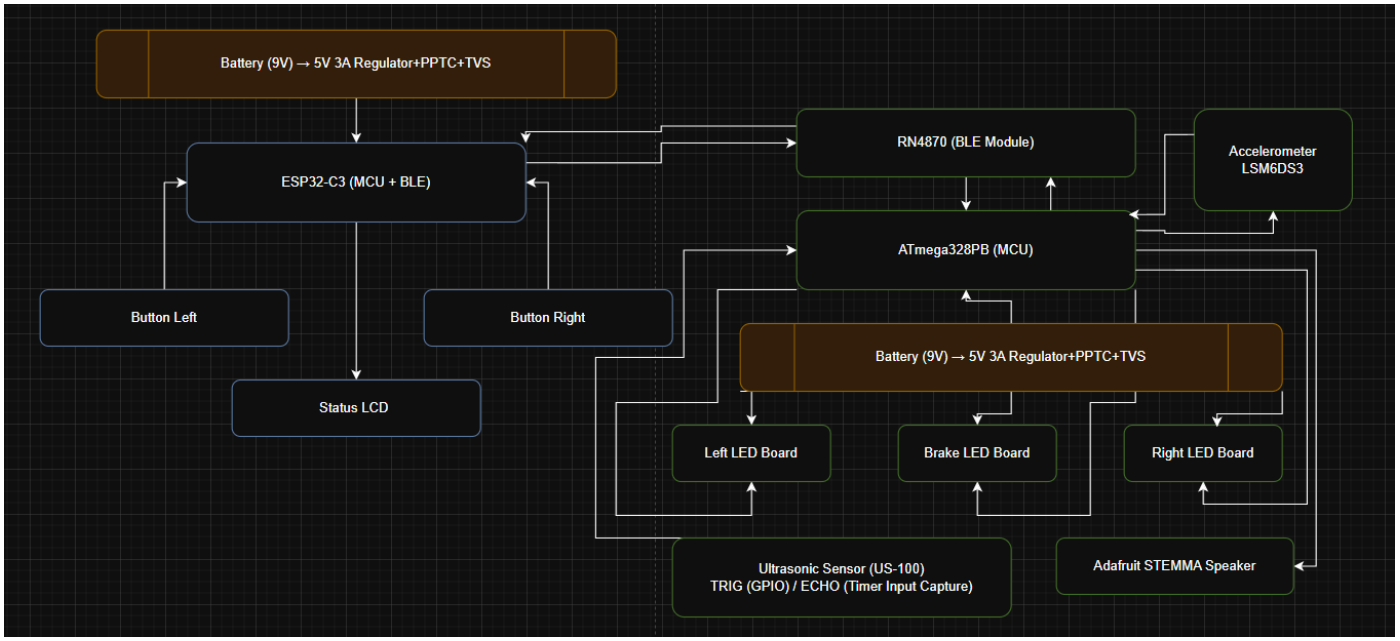
1. Abstract

Our project, **Smart Bike Light**, is a two-part wireless lighting and safety system for cyclists. The **front handlebar module** includes two buttons that transmit left and right turn signals via Bluetooth to the **rear signal module**. The rear module controls LED light boards for turn indicators, a brake light, and a buzzer alarm. It also integrates an ultrasonic distance sensor to detect vehicles approaching from behind and automatically triggers flashing and sound alerts when the distance becomes unsafe. This system demonstrates real-time embedded control, wireless communication, and sensor-based safety feedback to improve cycling visibility and road safety.

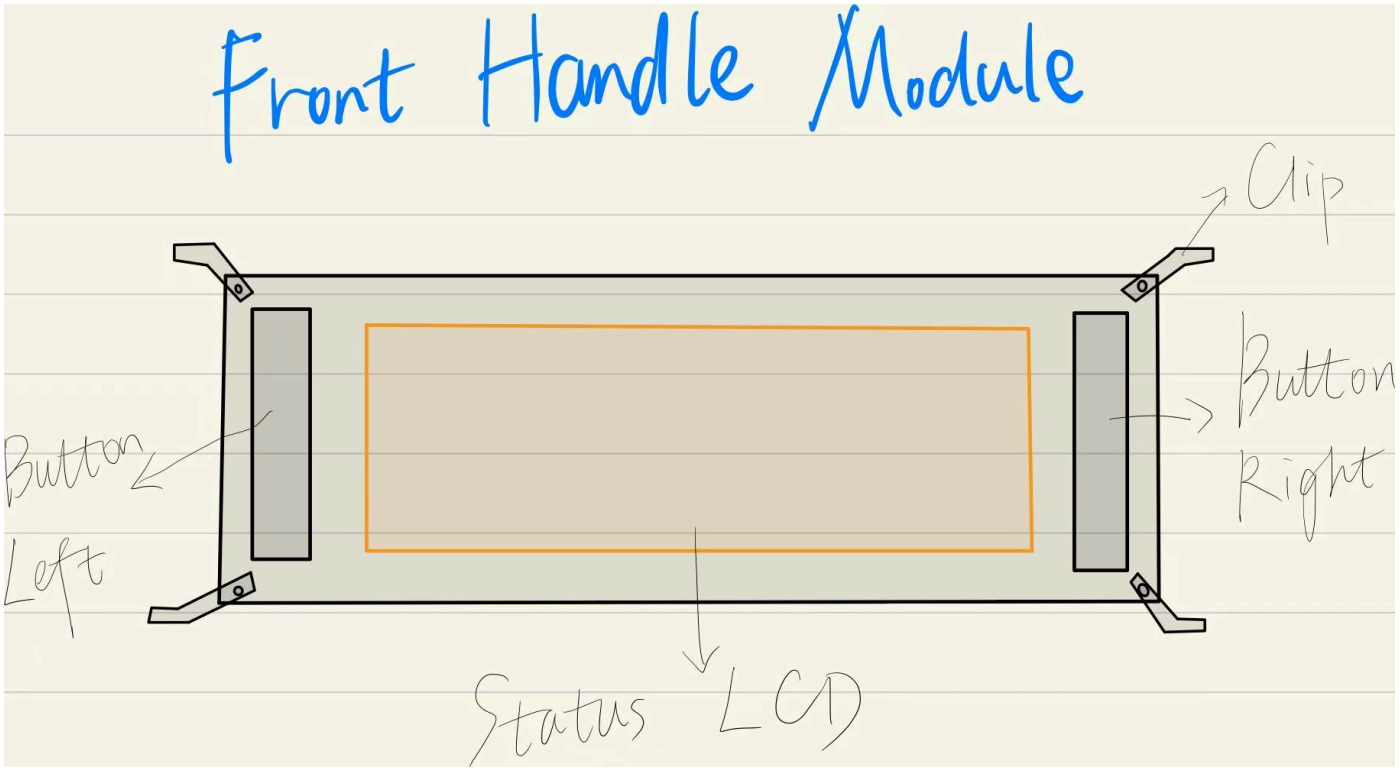
2. Motivation

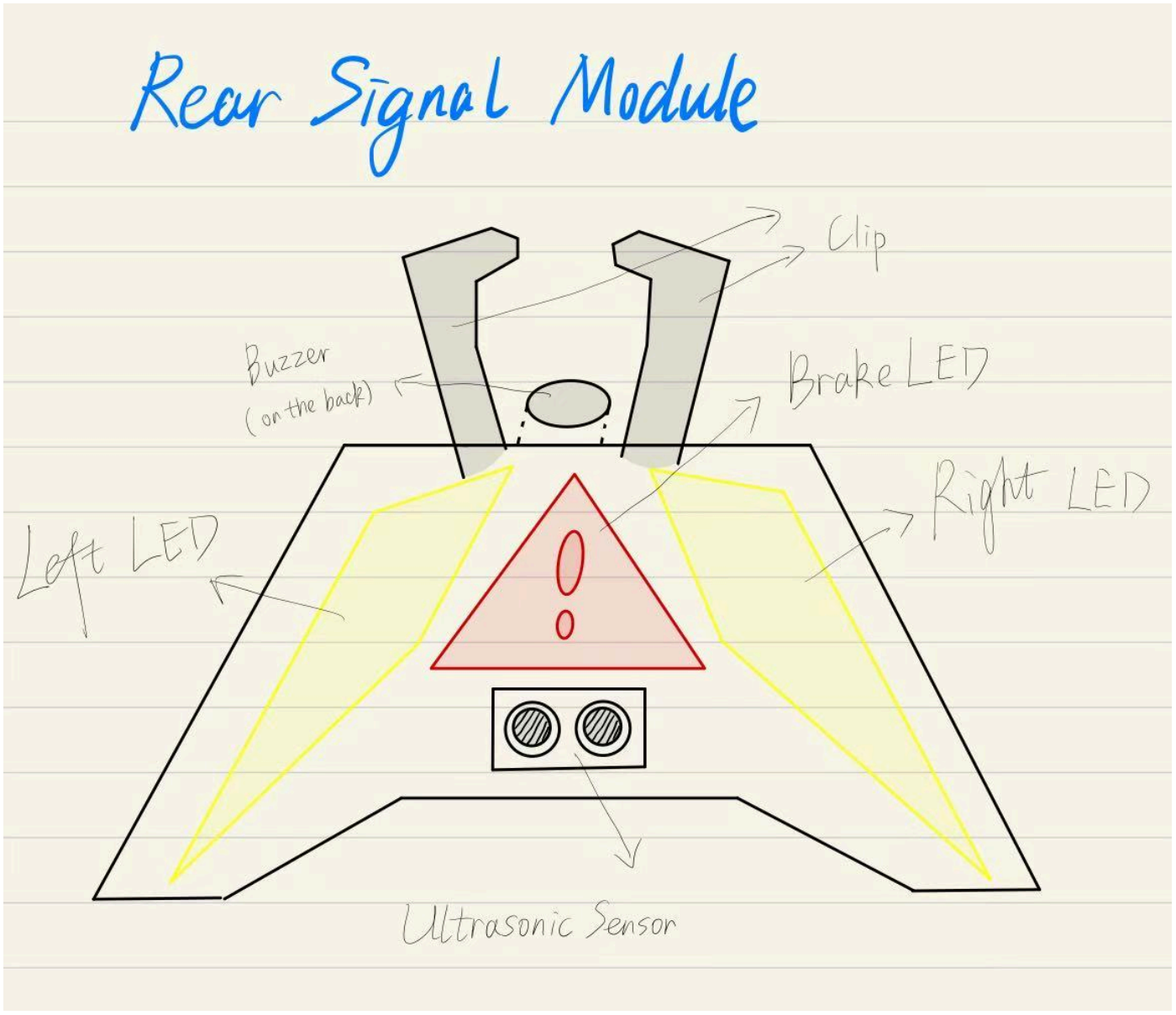
Cyclists often ride in low-visibility environments where hand signals and basic tail lights are not enough to ensure safety. Our project aims to solve this problem by creating an intelligent, responsive lighting system that improves communication between cyclists and surrounding vehicles. By combining Bluetooth wireless control, distance sensing, and PWM-based light modulation, the Smart Bike Light allows riders to signal turns more clearly and receive automatic warnings when another vehicle gets too close. The purpose of this project is to enhance night-riding safety, demonstrate real-time control using embedded systems, and provide a low-cost, easily installable upgrade for everyday bicycles.

3. System Block Diagram



4. Design Sketches





This project needs 3D printing to achieve the vision looks like the sketches and some function, such as being connected and fixed to the bicycle frame or seat.

5. Software Requirements Specification (SRS)

5.1 Definitions, Abbreviations

Here, you will define any special terms, acronyms, or abbreviations you plan to use for hardware

Term	Meaning
Left/Right mode	Only the corresponding side LED panel blinks.
Both-sides mode	Left and right panels blink together (hazard).
Brake steady	Center brake LED stays ON continuously.
Warning flash	Brake LED flashes at**4–6 Hz** (attention).
Burst strobe	Brake LED flashes at**8–12 Hz** (high urgency).
D_warn	Proximity warning threshold =**2.5 m**.
D_alert	Proximity alert threshold =**1.5 m**.
a_warn / a_alert	Deceleration thresholds:**0.8 m/s²** (warn), **1.5 m/s²** (alert).
Link loss	No valid BLE packet for**⩾ 1.0 s**.

5.2 Functionality

ID	Requirement Description	Validation Method
SRS-01	The front buttons**shall** command light modes: pressing **Left** or **Right shall** make only that side blink at **4–6 Hz**; pressing both **shall** make both sides blink at **4–6 Hz**. LEDs **shall** run at **maximum brightness** by default.	Record ⩾60 fps video while pressing buttons; count frames to verify 4–6 Hz and correct side(s); confirm command receipt in rear logs.

ID	Requirement Description	Validation Method
SRS-02	With a speed sensor installed, when speed is**0** or longitudinal deceleration is ⩾ **0.5 m/s²**, the brake LED **shall** turn **steady ON** and remain ON while the condition holds.	Use roller/hand-push; log speed/accel and brake state; verify steady ON during `v = 0` or `a ≤ -0.5`.
SRS-03	If filtered distance is within**(1.5 m, 2.5 m]***or* deceleration is ⩾ **0.8 m/s²**, the brake LED **shall** enter **warning flash** at **4–6 Hz** and **shall** exit only when distance is **> 2.5 m** and the condition clears for ⩾ **0.5 s**.	Mark 1–5 m; move target; record distance log and video; check frequency and hysteresis timing.
SRS-04	If distance is**⩽ 1.5 m***or* deceleration is ⩾ **1.5 m/s²**, the brake LED **shall** enter **burst strobe** at **8–12 Hz** and the buzzer **shall** beep in **1:1 sync** with the flashes.	Approach to ⩽1.5 m and perform a hard brake; capture audio + video; confirm 8–12 Hz and 1:1 beep/flash sync.
SRS-05	Ultrasonic ranging**shall** operate over **0.2–5.0 m** at ⩾ **10 Hz** update rate with error ⩽ ±**10 %** in the **0.5–3.0 m** region.	Place reflectors at known distances; log at ⩾10 Hz; compute error statistics vs. ground truth.

ID	Requirement Description	Validation Method
SRS-06	On BLE link loss for**⩾ 1.0 s**, the system**shall** hold **warning flash** (4–6 Hz) on the brake LED and **shall** ignore turn commands until the link recovers.	Power-cycle or shield BLE; verify warning flash persists and commands are ignored; confirm automatic recovery.
SRS-07	The LCD**shall** display **current speed** updated **⩾ 5 Hz** and **shall** show the active mode text (`Left/Right/Both/Brake/Warning/Burst/Link-Loss`).	Feed a known speed profile (roller or simulated pulses); measure update interval; verify mode text matches actual state.
SRS-08	The buzzer**shall** be active **only** during **burst strobe** and **shall not** sound in other modes unless a configuration flag enables it.	Cycle through all modes; confirm buzzer sounds only in burst; toggle the config flag and re-test.

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

Term	Meaning
MCU	ATmega328PB (Rear) / ESP32-C3 (Front)
BLE	Bluetooth Low Energy link via HM-10 (UART bridge)
US Sensor	Ultrasonic sensor (US-100 / HC-SR04, TRIG + ECHO)
LCD	Status display on front module

6.2 Functionality

ID	Hardware Requirement	Validation Method
HRS-01 (Power Integrity)	The system shall be powered by a regulated 5 V rail on both modules. With all loads active (LEDs + buzzer + sensors + MCU), the voltage shall not drop below 4.8 V .	Measure Vout using oscilloscope under full-load stress (turn signals 2 Hz + brake on + buzzer on).
HRS-02 (Ultrasonic Sensor Performance)	The US sensor shall detect obstacles from 0.2 m to ⩾ 3.0 m , and produce a clean5 V TTL ECHO pulsemeasurable by the MCU timer.	Place objects at known distances (0.2/1/2/3 m); confirm Echo pulse width and measurement error.
HRS-03 (Accelerometer Interface Reliability)	The LSM6DS3 shall provide stable acceleration data via I^2C at ⩾ 100k Hz , with signal noise not preventing braking detection.	Poll and log ACC output for 10 s; confirm ⩾ 100k samples/sec and stable noise band.
HRS-04 (BLE Link Budget & Range)	The BLE link via HM-10 shall maintain a stable UART connection over ⩾ 2 m indoor line-of-sight withpacket error rate < 1% .	Perform BLE distance walk test; compare Rx vs. Tx byte counts in UART log.

ID	Hardware Requirement	Validation Method
HRS-05 (Fail-Safe Hardware Behavior)	When BLE link is lost, or when the MCU resets/browns-out, the rear system hardware shall ensurethe Brake LED defaults ON(via pull-up or safe-bias), providing continuous visibility even before firmware recovers.	Power-cycle BLE / press reset / induce brown-out; visually confirm Brake LED stays on.
HRS-06 (Buzzer Output)	The buzzer drive stage shall deliver ⩾60 dB SPL at 30 cm when activated, and be fully switch-controllable by the MCU (no audible leakage in idle).	Measure SPL using phone app at 30 cm; verify silence in idle via oscilloscope.
HRS-07 (Front LCD Interface)	The LCD interface shall support ⩾ 5 Hz refresh without causing missed BLE packets or sensor sampling on the front MCU.	Run refresh test; confirm stable BLE log + LCD updates with stopwatch timing.

7. Bill of Materials (BOM)

Handlebar Transmitter (Front Unit)

1. Two momentary buttons with a hardware debouncer IC MAX6817 (Dual-channel): We use two independent tactile buttons for “Left” and “Right,” both routed into a single MAX6817 dual debouncer so the MCU sees clean, single transitions for opens and closes. The MAX6817 runs from +2.7 V to +5.5 V , needs no external components , draws about 6 µA supply current, and hardens the inputs with ±15 kV ESD protection . Its internal qualification window removes both opening/closing bounce; the specified debounce duration for MAX6817 is typically ~50 ms (range ≈20–80 ms , device-grade dependent), which is ideal for gloved operation and bumpy riding conditions.
2. MCU with BLE(Seeed Studio XIAO ESP32-C3): We use the MCU which reads buttons (through the debouncer), formats and transmits left/right commands to the rear unit over BLE; also receives speed data from the rear and forwards it to the LCD. The board exposes a regulated 3.3 V rail for the LCD logic.

3. Status LCD (SPI TFT, ST7789-class): We use a 2.0" color TFT driven over 4-wire SPI (SCK, MOSI, CS, D/C; RST on a GPIO, no MISO required). The ESP32-C3 dev board is powered from 5 V but provides a regulated 3.3 V rail; the TFT is run from that 3.3 V rail so all logic levels are native to the MCU. The display shows Left/Right/Both/Brake/Warning/Burst/Link-Loss status and current speed. Using SPI keeps wiring simple on the handlebar while preserving GPIOs for the buttons and debouncing. The backlight current is budgeted on the 3.3 V rail and can be PWM-controlled (or series-limited) for comfortable night riding.
4. Front-end power (9 V battery → buck to 5 V): A compact DFRobot DFR0379 buck module (LM2596-based) steps the 9 V battery down to a stable 5 V rail for the ESP32-C3 board. The module accepts 4–40 V input and provides an adjustable 1.25–37 V output rated up to 3 A.

Rear Signal Unit(Tail Module)

1. Main MCU(Microchip ATmega328PB): The**ATmega328PB** serves as the central controller for the rear unit. The MCU receives command frames from the RN4871 over UART, parses the turn/brake instructions, and drives the left/right/brake channels accordingly. It also controls the speaker, measures distance by timing the US-100 echo pulse with a timer capture, and reads the accelerometer to detect rapid deceleration for brake-light activation. Processed motion data are forwarded back to the front unit via the BLE link, ensuring coordinated speed display.
2. BLE link module (Microchip RN4871): TheRN4871 Bluetooth module establishes and maintains wireless communication with the handlebar unit, transmitting control and sensor data between the front and rear systems. Operating at 1.9–3.6 V , the RN4871 is connected to the ATmega328PB via a UART interface for simple and reliable serial communication. The module supports Bluetooth® 5.0 , ensuring stable pairing and low-latency data transfer within typical riding distances. Its compact form and low-power operation make it ideal for battery-powered embedded applications, providing dependable connectivity without significantly increasing energy consumption.
3. LED indicator boards (American Opto Plus LED Corp. L381L-LEPGB3DI6), MOSFET-switched: Each function light (Left, Right, Brake) is built from multiple small 5 V LED boards connected in parallel to increase the illuminated area. The ATmega328PB controls each function channel, and logic-level MOSFETs switch the 5 V line to provide on/off control and PWM-based blink patterns for turn and brake indications.

4. Ultrasonic distance sensor (US-100): The US-100 ultrasonic module measures the distance between the bicycle and vehicles behind it. It operates with a trigger and echo interface, and the MCU measures the echo pulse width using a timer input capture. When the measured distance falls below a preset safety threshold, the MCU activates the speaker to warn the rider of a vehicle approaching too closely.
5. Speaker(Adafruit STEMMA Speaker): The speaker is controlled by the MCU and serves as an alert mechanism. It is triggered automatically when the ultrasonic sensor detects an object within the danger range, producing an immediate audio warning to enhance rider safety.
6. Accelerometer sensor(LSM6DS3): The accelerometer continuously measures the rear module’ s acceleration data. When rapid deceleration is detected, the MCU interprets it as braking on the brake light accordingly. The same acceleration data are transmitted via Bluetooth to the front MCU, which calculates and displays the current speed on the LCD.
7. Power module(9 V battery → buck to 5 V): A 9 V battery supplies power to the system. The voltage is regulated by the same LM2596-based buck converter used in the front unit, which steps the voltage down to a stable 5 V. This regulated output powers the ATmega328PB MCU and the LED indicator boards (the BLE module and the sensors are then powered by MCU), ensuring consistent performance and sufficient current capacity for all rear components.

Google Sheet for BOM link:
https://docs.google.com/spreadsheets/d/1bHwMPB5nC805xI1gRusFQk9g4mlo7jpOiUuIshi9kqY/edit?usp=sharing

8. Final Demo Goals

Scope: A tabletop (or bike-mounted) live demo showing all core features under realistic conditions. Each goal has a clear pass/fail criterion tied to the SRS.

G1 — Turn Signal Modes at Maximum Brightness

- The system **shall** illuminate only the left (or right) LED panel when the corresponding button is pressed, and **shall** illuminate **both** panels when both

buttons are pressed.
- Blink frequency **shall** be **4–6 Hz** in all turn modes; LEDs **shall** operate at maximum brightness.
- **Acceptance:** A ⩾60 fps video and logic log **shall** confirm 4–6 Hz and correct side(s) per input.

G2 — Proximity-Based Warning and Alert

- When the measured distance is in **(1.5 m, 2.5 m)**, the brake LED **shall** enter **warning flash** at **4–6 Hz**.
- When the measured distance is ⩽ **1.5 m**, the brake LED **shall** enter **burst strobe** at **8–12 Hz**, and the buzzer **shall** beep in 1:1 sync with flashes.
- **Acceptance:** Floor marks at 3 m/2.5 m/1.5 m and synchronized audio/video **shall** prove threshold switching and sync.

G3 — Brake Steady from Speed/Deceleration

- With the speed sensor installed, when speed is **0** or longitudinal deceleration is ⩾ **0.5 m/s²**, the brake LED **shall** remain **steadily ON**.
- **Acceptance:** Logged speed/acceleration and LED state **shall** show steady ON during stop and ⩾0.5 m/s² decel.

G4 — Ultrasonic Performance Window

- The ultrasonic ranging **shall** operate over **0.2–5.0 m** at an update rate ⩾ **10 Hz** and error ⩽ ±**10 %** in the **0.5–3.0 m** region.
- **Acceptance:** Bench measurements at known targets **shall** meet update rate and error bounds.

G5 — Link-Loss Safety

- On BLE link loss of ⩾ **1.0 s**, the system **shall** hold the brake LED in **warning flash** and **shall** ignore turn commands until the link recovers.
- **Acceptance:** Induced link loss **shall** produce persistent warning flash and command ignore; normal control **shall** resume on reconnection.

G6 — LCD Telemetry

- The LCD **shall** display **current speed** updated ⩾ **5 Hz** and **shall** show the active mode text (`Left/Right/Both/Brake/Warning/Burst/Link-Loss`).
- **Acceptance:** Time-stamped screenshots/video **shall** verify update rate and correct mode text.

G7 — Configuration & Persistence

- Thresholds (`D_warn=2.5 m`, `D_alert=1.5 m`) and buzzer enable **shall** be configurable and **shall** persist across power cycles via non-volatile storage.
- **Acceptance:** After reconfiguration and reboot, proximity behavior and buzzer state **shall** match new settings.

G8 — Safety & Power Constraints

- The device **shall not** be powered by Li-ion/LiPo packs; **power banks or AA/AAA shall** be used.
- The assembled system **should** operate continuously for ⩾ **10 min** at demo with no resets or unsafe temperatures.
- **Acceptance:** Visual inspection of power source **shall** confirm compliance; a timed burn-in **shall** show stable operation.

9. Sprint Planning

Milestone	Functionality Achieved	Distribution of Work
Milestone	Functionality Achieved (Shall statements)	Distribution of Work & Acceptance
---	---	---
Sprint #1	• Rear ATmega328PB bring-up**shall** drive LED PWM at max brightness. • Ultrasonic driver **shall** stream distance at ⩾ **10 Hz** with basic filtering. • Front unit	**Tiancheng**: rear init & PWM; **Zibo**: ultrasonic + logging; **Jiaan**: buttons, BLE, LCD. **Acceptance:** scope traces for PWM; serial logs show ⩾10

Milestone	Functionality Achieved	Distribution of Work
	shall scan Left/Right/Both buttons and establish BLE UART. • LCD scaffold **shall** render mode text; speed input path **shall** be stubbed.	Hz distance; BLE echo works; LCD shows mode text.
Sprint #2	• Turn modes**shall** blink at **4–6 Hz**; state machine complete. • Proximity logic **shall** trigger warning at **2.5 m** and burst+buzzer at ⩽ **1.5 m** (1:1 sync). • Link-loss of ⩾ **1.0 s shall** hold warning flash. • Speed sensor integration **shall** set brake steady at stop or decel ⩾ **0.5 m/s²**. • Config/persistence and basic POST **shall** be implemented.	**Tiancheng**: state machine & timers; **Zibo**: thresholds/hysteresis & buzzer sync; **Jiaan**: speed interface & brake logic, config. **Acceptance:** videos/logs proving each trigger; POST banner on boot; config survives reboot.
MVP Demo	**End-to-end integration**: All above features **shall** operate on battery power with stable wiring; LCD **shall** show speed ⩾ 5 Hz; a ⩾ **10 min** burn-in **shall** pass without resets.	**Tiancheng**: integration & power; **Zibo**: test scripts, distance tape; **Jiaan**: demo script & README. **Acceptance:** live demo passes G1–G8 checks and burn-in.
Final Demo	• Calibration/tuning**shall** refine thresholds and frequencies; debounce **shall** be verified. • Enclosure & mounts **shall** be completed; thermal and basic splash checks **shall** pass. • Extended burn-in **should** run ⩾ **30–60 min** without resets. • Optional features (fall detection, flowing arrows) **should** be added if time permits.	**Tiancheng**: calibration & safety; **Zibo**: enclosure/mounts & thermal; **Jiaan**: final media/docs/tests. **Acceptance:** final demo passes all Section 8 goals; enclosure present; extended burn-in log available.

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

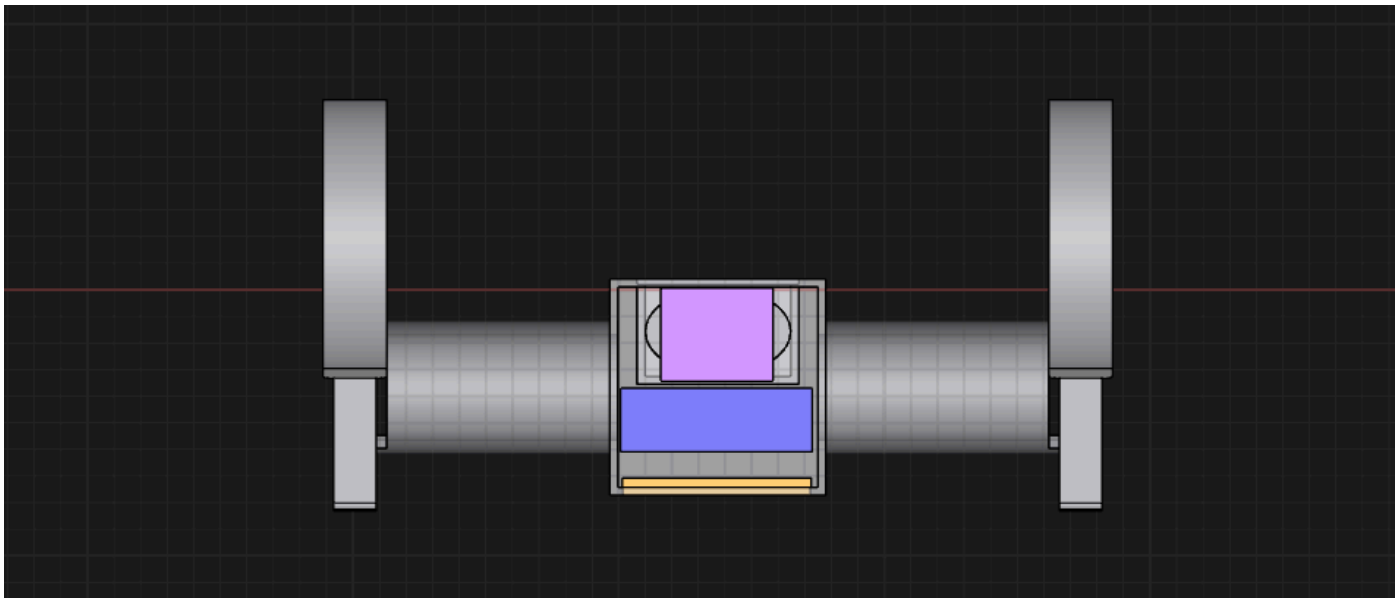
This week our parts still had not arrived, so on Friday November 7 we talked with our Account Manager and decided to adjust our BOM because of some supplier and shipping issues. After that meeting we also changed the way we split up the work. Since we could not start soldering yet, we decided to focus on the mechanical side and work on the 3D printed enclosures.

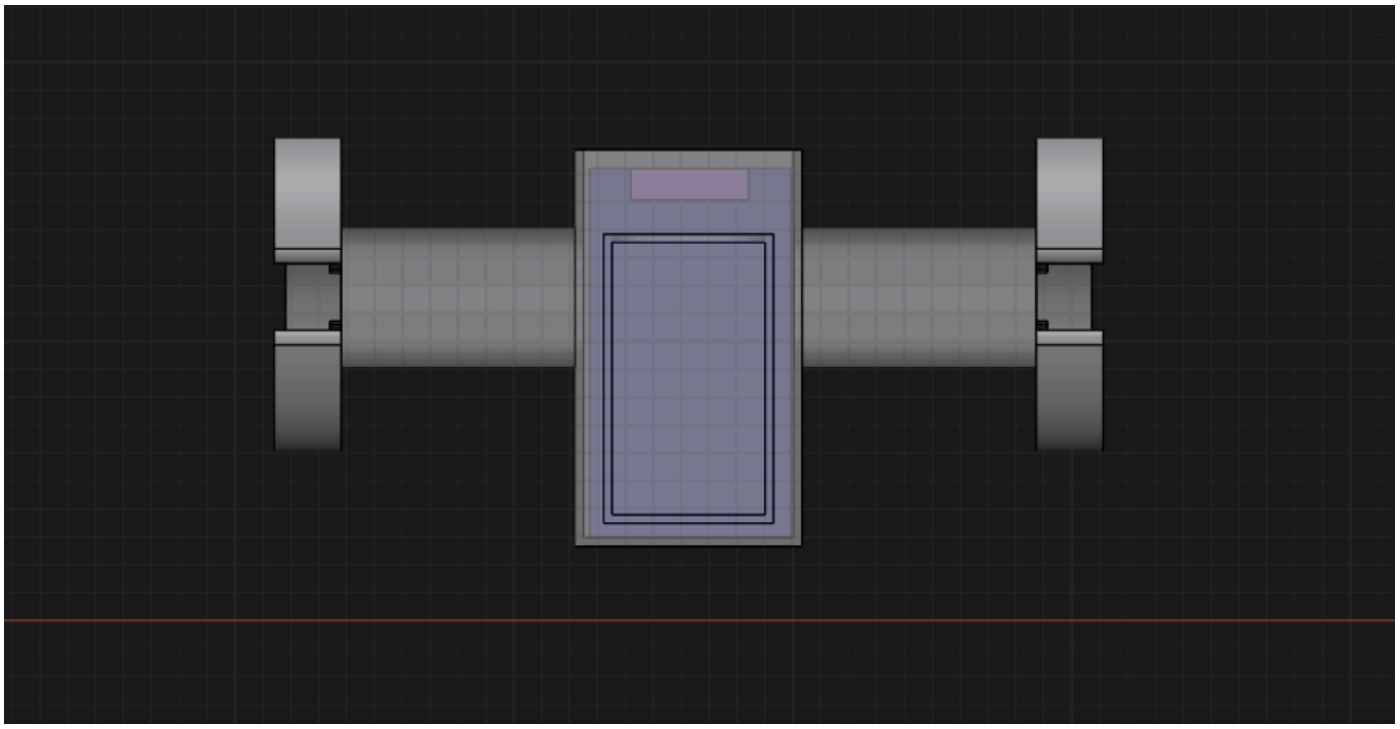
From Tuesday until now, Jiaan has been working on the front enclosure. This includes the housing for the LCD status screen and the joystick style input, a compartment for the battery, reserved space for the buck converter and the front MCU board, and two clamp pieces that will allow the box to be fixed to the bicycle. There are also openings for cable routing and basic wire management.

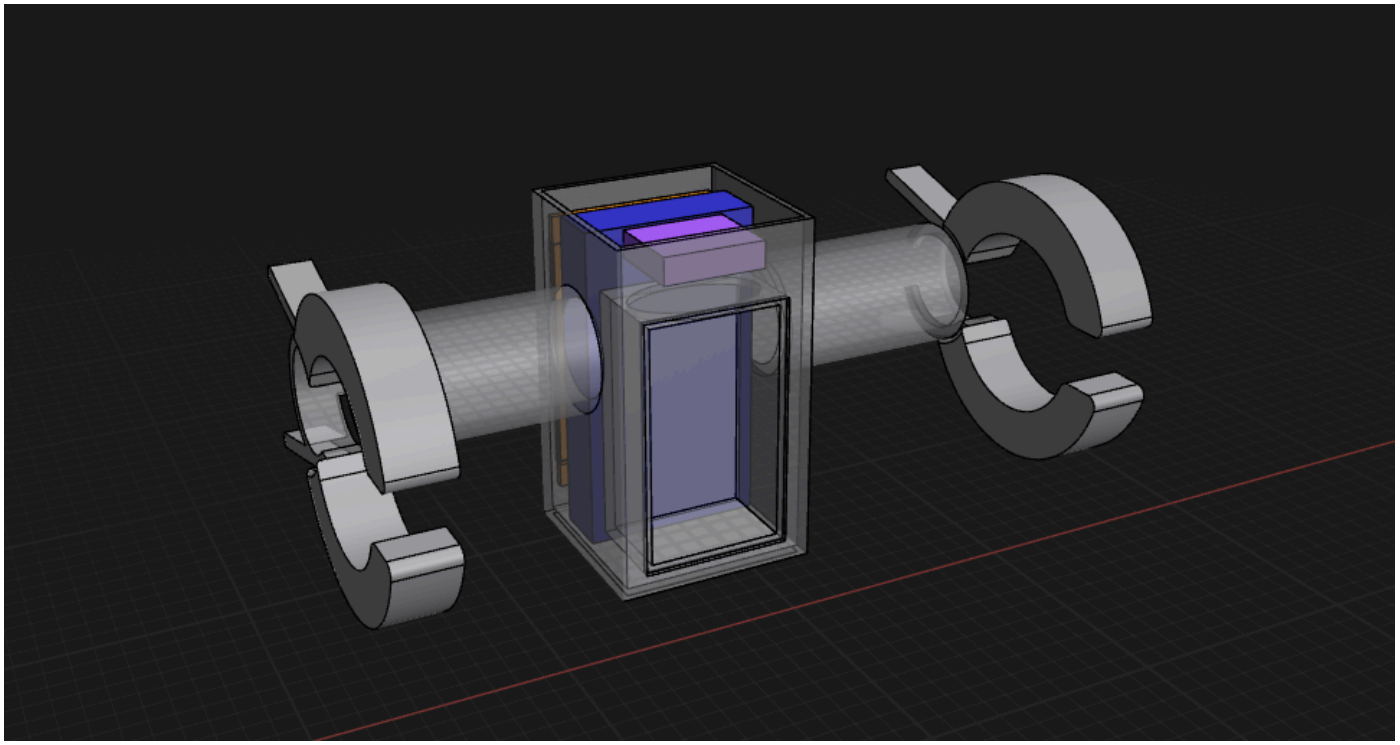
At the same time, Tiancheng and Zibo have been working together on the rear enclosure. They are laying out the 45 LEDs, deciding the overall size of the rear light housing, and arranging the inside space for the battery, MCU, accelerometer and buck converter. They are also planning where to place the side facing ultrasonic distance sensor and the speaker so that everything fits cleanly in one unit.

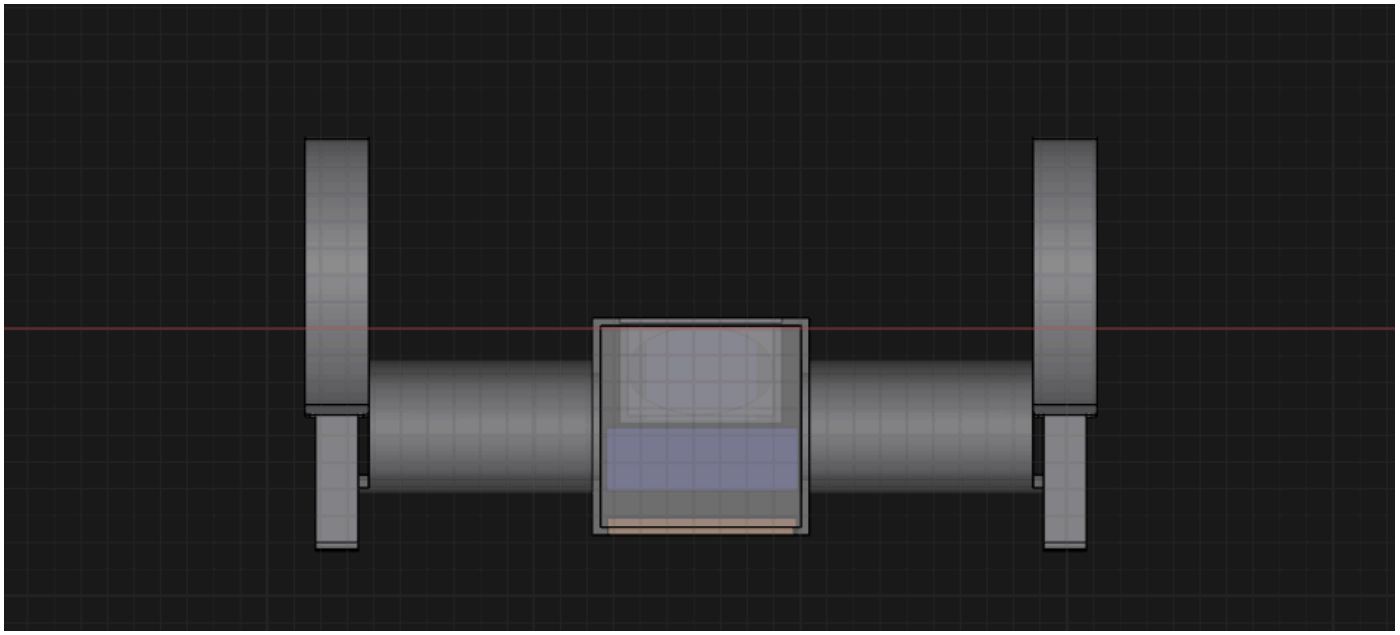
Even though most of the electronics are still missing, spending this week on the 3D design feels meaningful because it connects directly to how the final product will look and mount on a real bike. All three of us are using 3D printing for the first time, so we had to teach ourselves how to model the parts in Shapr3D and SolidWorks,

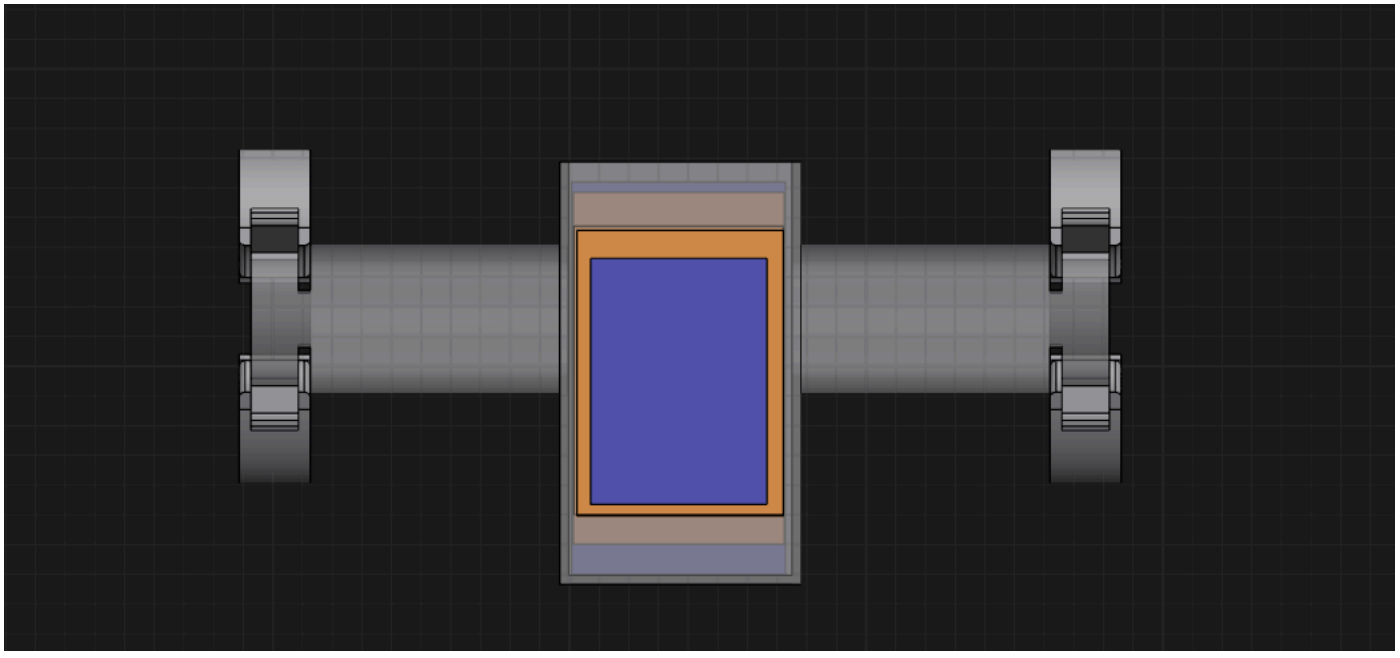
and we already learned a lot from trying things, making mistakes and fixing them.

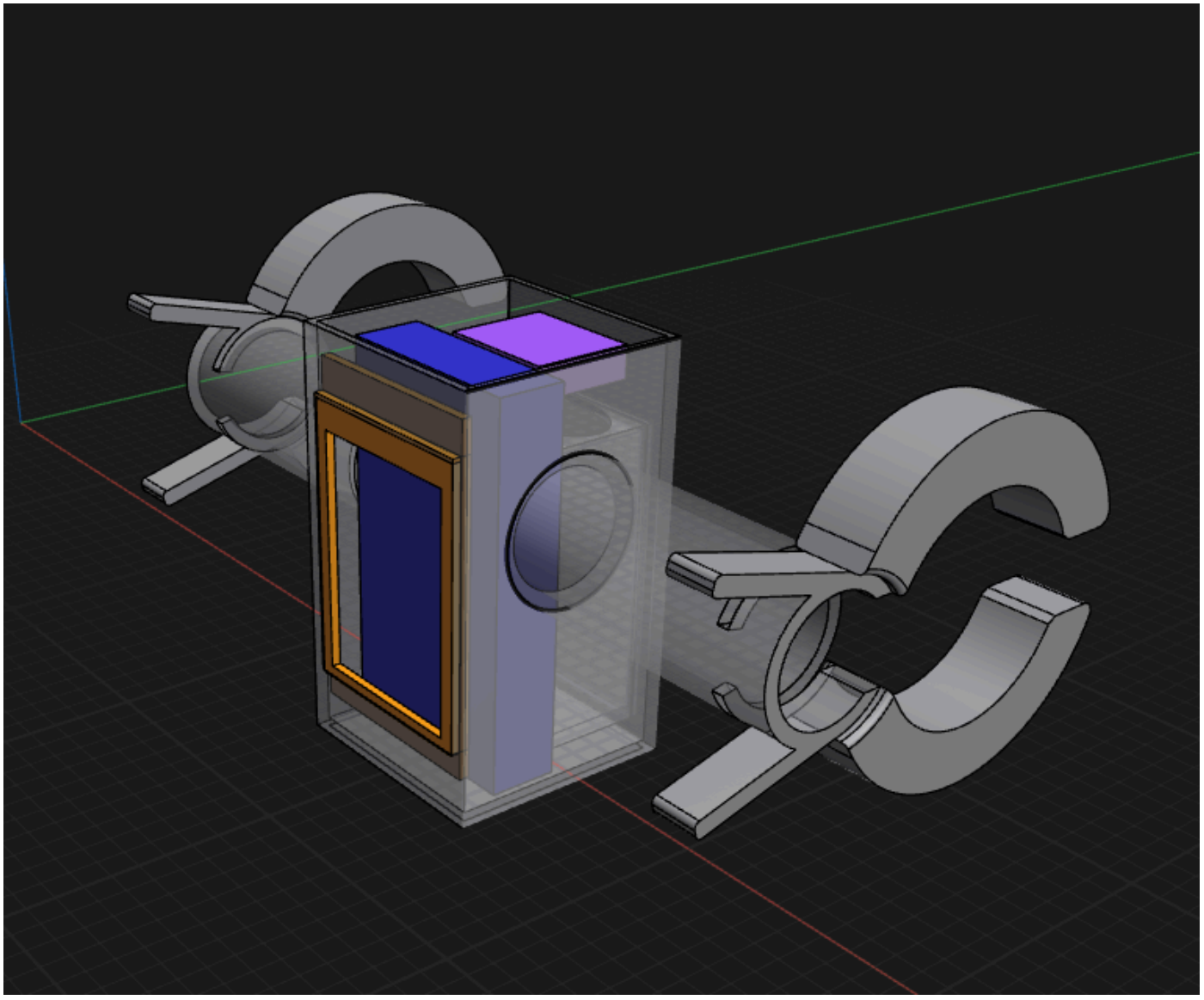


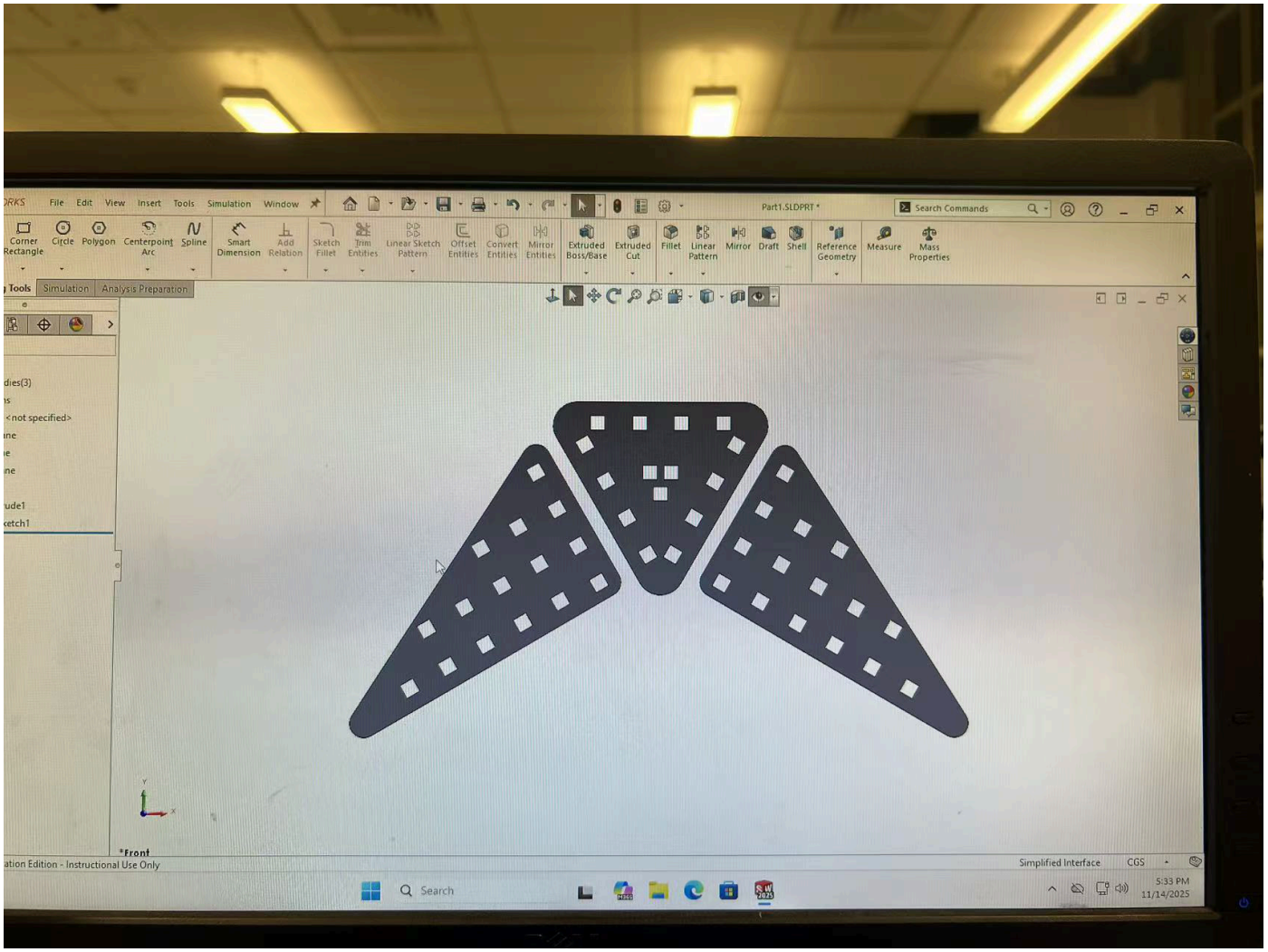


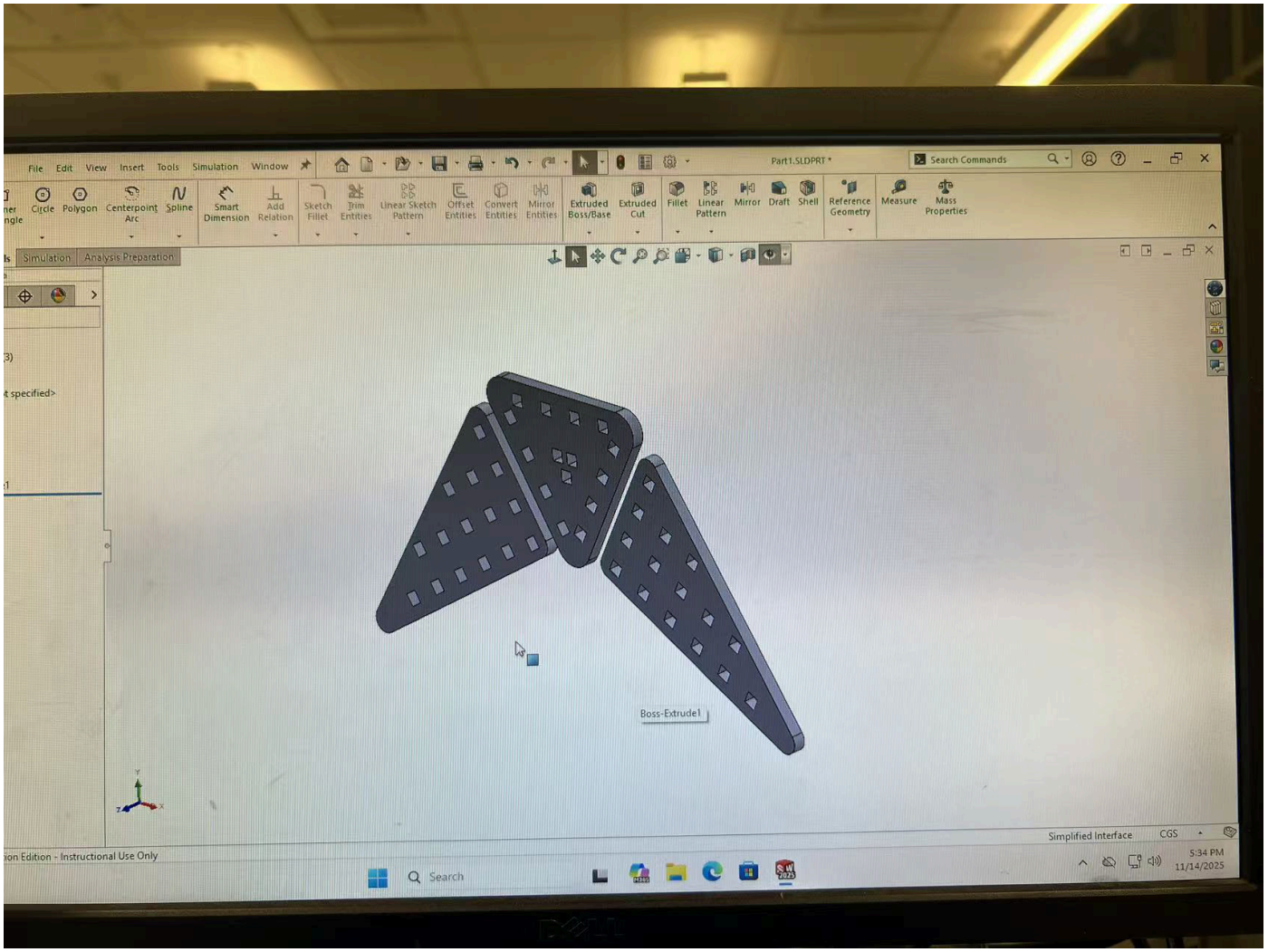









File Edit View Insert Tools Simulation Window
Part1.SLDPRT *
Search Commands
Corner Rectangle
Circle
Polygon
Centerpoint Arc
Spline
Smart Dimension
Add Relation
Sketch Fillet
Trim Entities
Linear Sketch Pattern
Offset Entities
Convert Entities
Mirror Entities
Extruded Boss/Base
Extruded Cut
Fillet
Linear Pattern
Mirror
Draft
Shell
Reference Geometry
Measure
Mass Properties
Simulation
Analysis Preparation
*Front
Simplified Interface
CGS
Search
5:33 PM
11/14/2025


File Edit View Insert Tools Simulation Window
Part1.SLDPRT *
Search Commands
Circle
Polygon
Centerpoint Arc
Spline
Smart Dimension
Add Relation
Sketch Fillet
Trim Entities
Linear Sketch Pattern
Offset Entities
Convert Entities
Mirror Entities
Extruded Boss/Base
Extruded Cut
Fillet
Linear Pattern
Mirror
Draft
Shell
Reference Geometry
Measure
Mass Properties
Simulation
Analysis Preparation
Boss-Extrude1
Simplified Interface
CGS
Search
5:34 PM
11/14/2025

Current state of project

Right now the electronics hardware has not arrived yet, which seems to be common for many teams in the class at this stage. Because of that, we do not have any real boards or sensors to power up and test, and there is no soldering or wiring done yet.

On the other hand, the overall system architecture is clear and the main components are already chosen in the BOM. The front and rear module roles are fixed, and the enclosure design is starting to catch up with that plan. The front CAD model for the LCD and input box is close to a first version that can be printed, and the rear enclosure model is in progress with the LED arrangement and internal layout. Once the parts arrive, we expect to move quickly from this mechanical work into assembly and firmware bring up.

Next week's plan

Our goal is to finish all of the 3D modeling work by the end of this weekend and then send the designs to the PRL for printing. Next week we expect to spend around four days on this and on preparing for assembly. If the parts still have not arrived by the middle of next week, we plan to start writing the software anyway. We can begin with the rear unit logic and some basic structure for the front firmware, even though real debugging will have to wait until the hardware is on the table.

As soon as the boards and sensors arrive, we will start soldering and doing the first wiring of the system. The rear unit will probably be the first one we can test, since the front side depends more directly on its specific MCU board. After both BLE modules are ready, we will work on pairing them, and in parallel we will solder and test the LED boards inside the new 3D printed housings. The long term idea is to bring the electronics to a stable state, then mount everything in the enclosures and move into a full round of product testing and small refinements until the system feels like a complete and reliable bike light.

Sprint Review #2

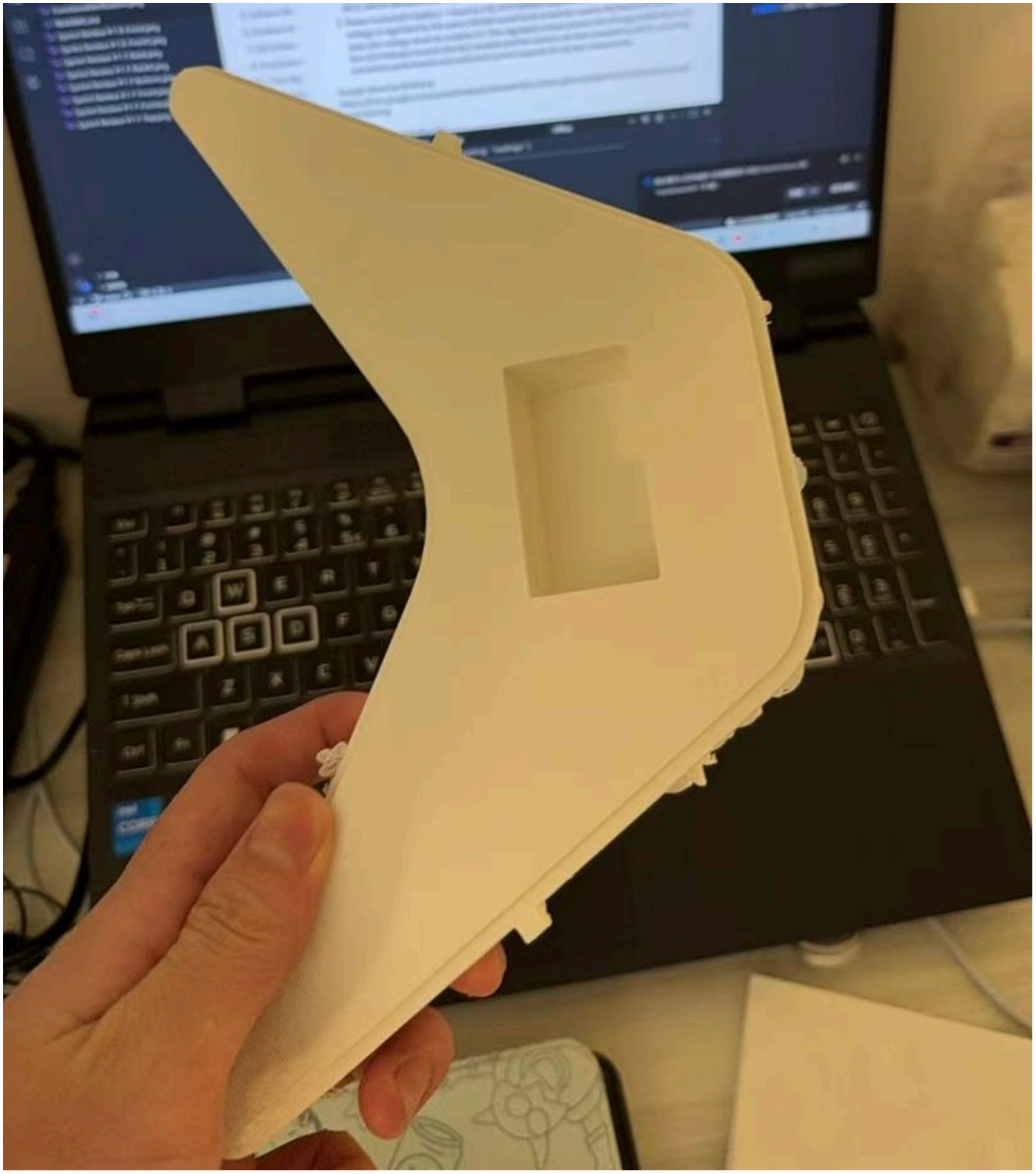
Last week's progress

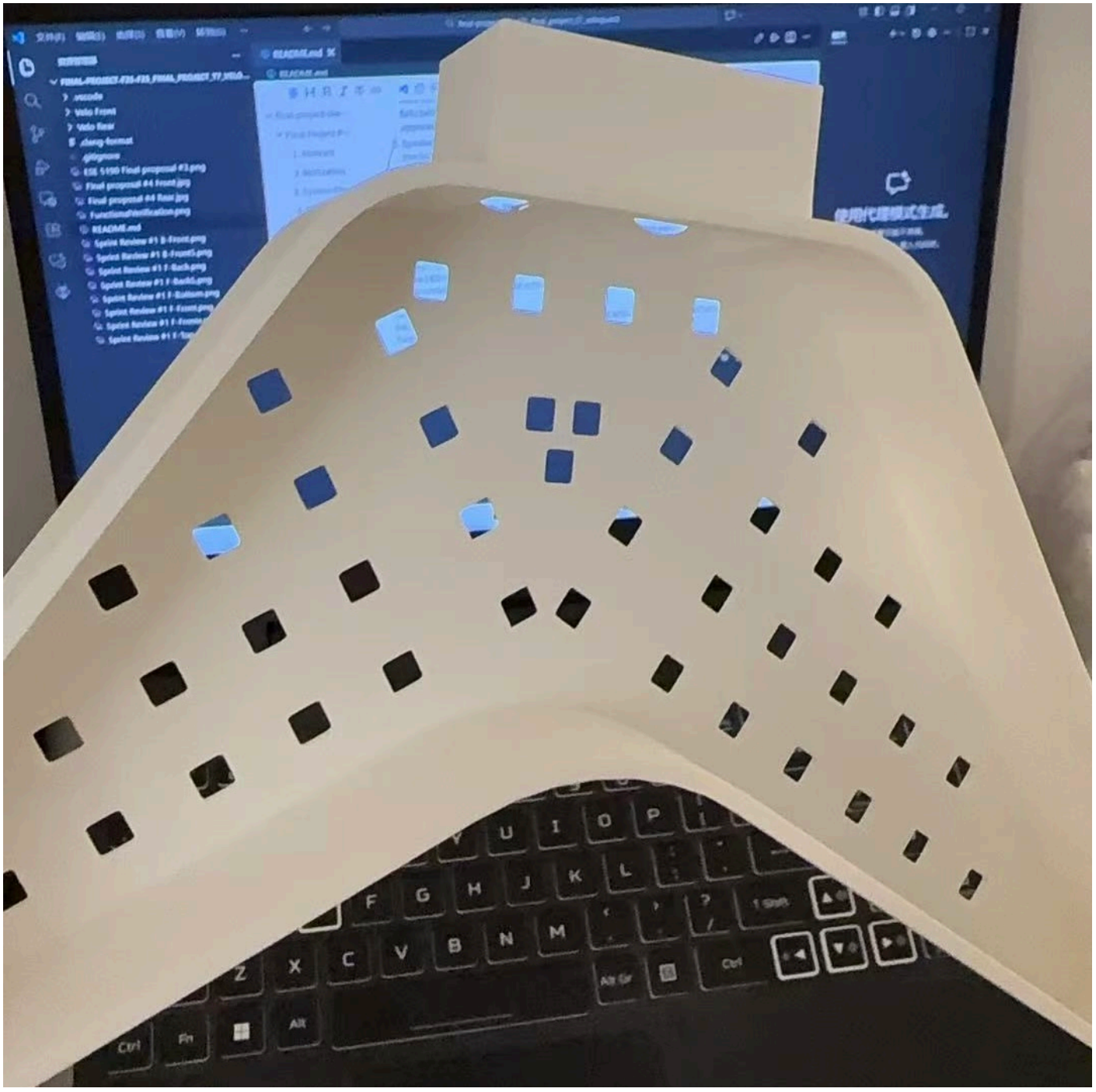
During the past week, the team made substantial progress on both the front-end and back-end modules of the project. The 3D-printed enclosures for both modules have been completed and received. The printed parts largely match our expectations in terms of fit and overall design, although a few minor issues were identified that will require small revisions. These adjustments are not expected to take much time.

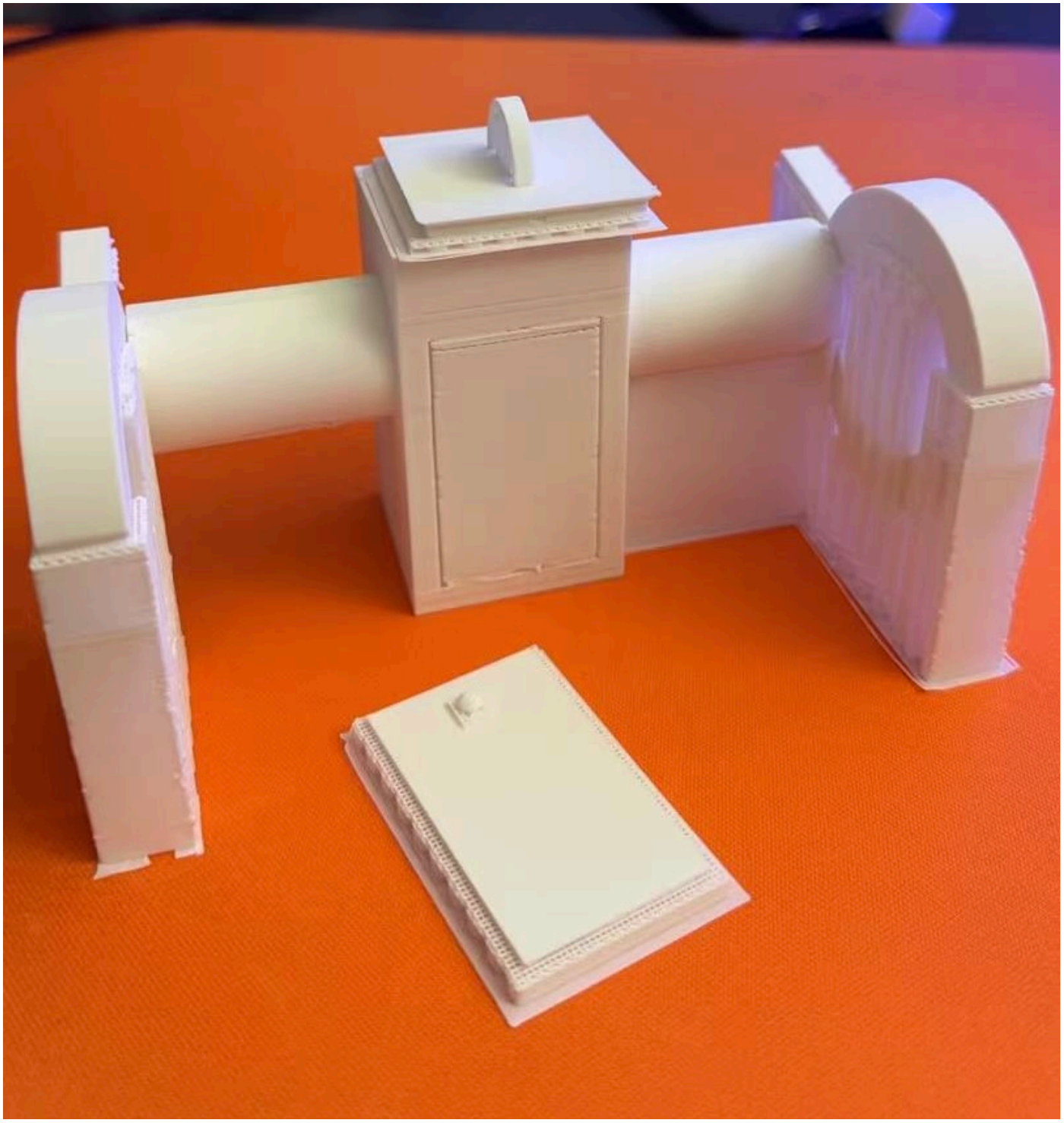
Because the BOM components still have not arrived this week, the team was unable to fully integrate the electronic components with the 3D-printed enclosures. As a result, our main focus shifted to the code development.

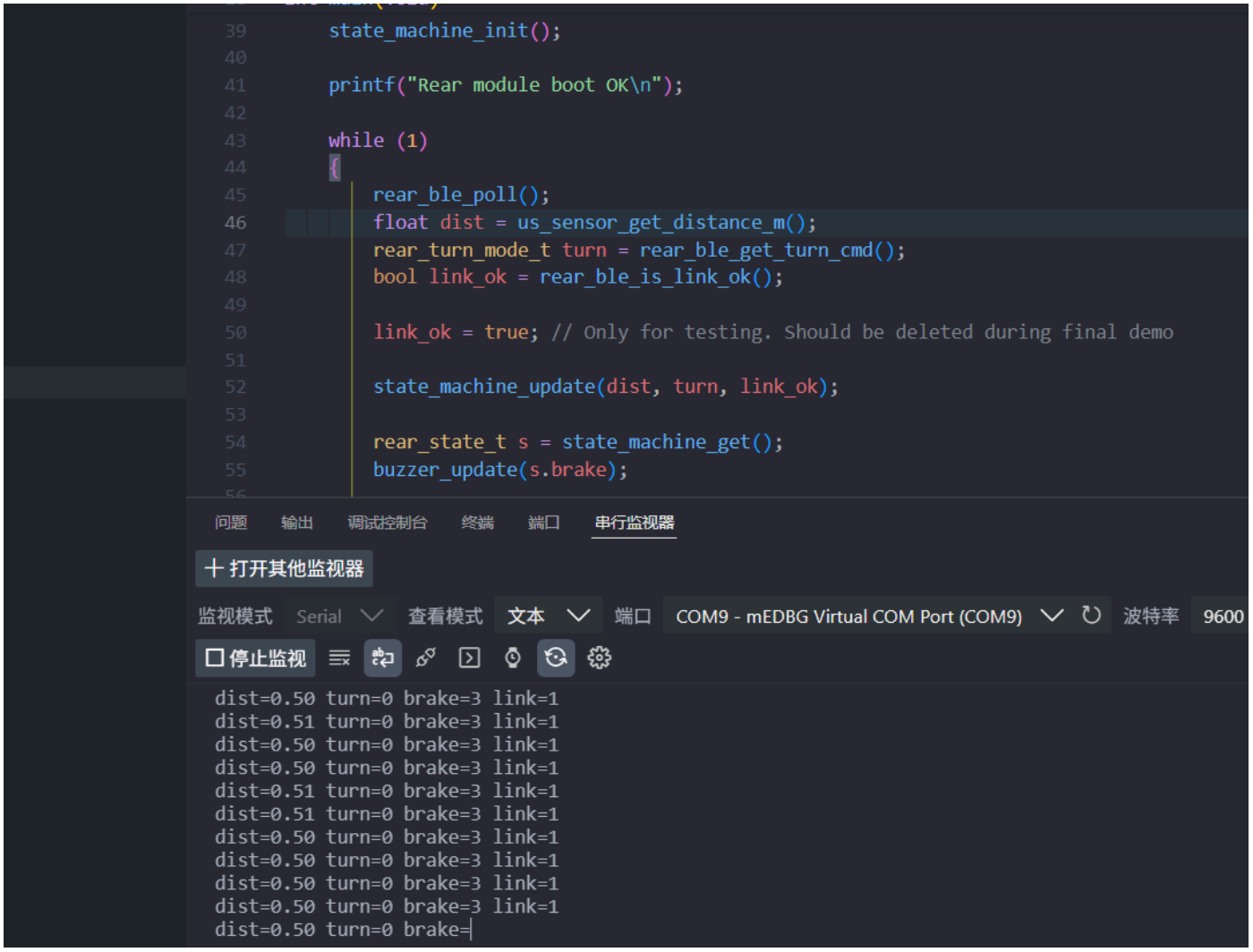
On the front-end side, Jiaan worked on the code and successfully implemented the functionality that allows the user to control the turn signals via the joystick. When the user moves the joystick, the corresponding blinking turn indicator icon is displayed on the LCD. This behavior was tested and validated using spare components available in the lab. In addition, the Bluetooth code for the front module has been completed, including both transmitting turn-signal commands and receiving speed data from the accelerometer. Once the actual components arrive, this code will be ready for full system testing. (Code is documented in the Velo Front file.)

On the back-end side, Tiancheng and Zibo focused on the core low-level functional coding. They implemented the main framework for the rear module, including BLE command parsing, ultrasonic distance measurement, the basic system state machine, buzzer alert logic, and the system-level timer. These features have also been validated using lab hardware, providing a solid foundation for integrating additional components such as LEDs and the accelerometer in the next phase. (Code is documented in the Velo Rear file.)









Current state of project

At this stage, the 3D-printed enclosures and the majority of the code for both the front and rear modules are essentially complete. The remaining work mainly depends on the arrival of the BOM components, which are required to finish a small portion of the code and to carry out full functional verification.

Although the Bluetooth logic for both the front and rear modules has already been implemented, the team is currently unable to perform end-to-end communication tests between the two modules because the actual Bluetooth modules have not yet arrived.

Next week's plan

The team has scheduled the MVP demo for Monday, which means that the necessary components ideally need to arrive before then in order to complete full system integration and testing.

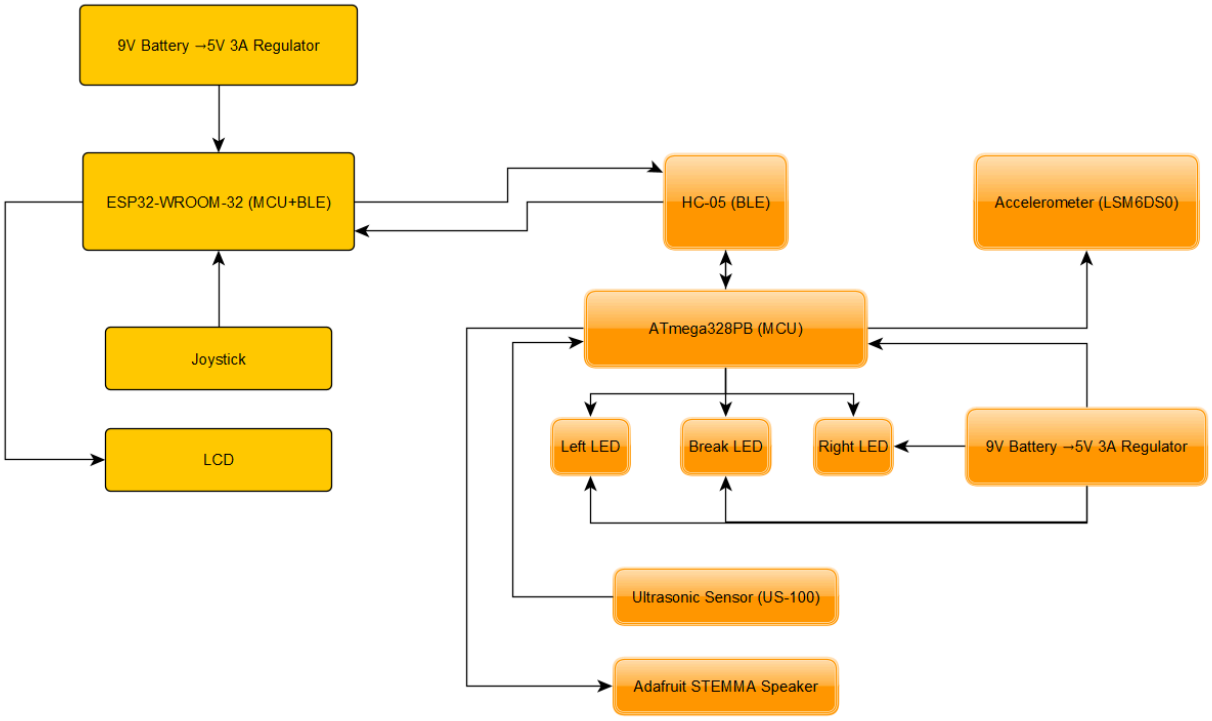
If the materials arrive on time , next week the team will focus on integrating all electronic components with the 3D-printed enclosures, establishing and testing Bluetooth communication between the front and rear modules, and conducting comprehensive functional tests of the overall system.

If the materials are still delayed , the team will use alternative devices available in the lab to temporarily test the overall system behavior, prioritize verifying the connection and interaction between the front-end and back-end modules, and use FreeRTOS tasks to simulate the BLE link, periodically generating bicycle turn-signal telemetry data to validate the system logic in the absence of the final hardware.

This approach will allow the team to continue making meaningful progress toward the MVP, even under hardware delivery constraints.

MVP Demo

1. *Show a system block diagram & explain the hardware implementation.*



Compared with our initial project design, we had to make several adjustments because the ordered components have not arrived yet. To keep the project moving, we switched to parts available in the lab and still managed to implement all the basic functions. On the hardware side, for the front-end we replaced the original MCU with the lab's ESP32-WROOM, which also integrates Bluetooth and can communicate with the HC-05 module. We also simplified

the user interface: instead of using two separate buttons for left and right turn signals as in the original design, we now use a single joystick to control both directions. This allows the user to operate the system with one hand and reduces overall complexity. For the display, we chose a smaller LCD screen that was available in the lab.

On the back-end side, we replaced the original Bluetooth module with an HC-05, which is more common and easier to pair with the front-end ESP32-WROOM. At the same time, we changed the accelerometer to the lab's SparkFun LSM6DS0 6DoF IMU. Because the speaker module hasn't arrived, we used a buzzer instead to test the system. These substitutions keep our system architecture close to the original plan while making sure we can test and demonstrate all core functions even before the final components arrive.

2. *Explain your firmware implementation, including application logic and critical drivers you've written.*

   Our firmware is organized around a simple main loop plus a few key drivers for sensing, communication, and LEDs/buzzer control. We initialize the timer, ultrasonic sensor, IMU, buzzer, rear BLE link, and brake/turn LEDs. Then we run a 50 ms loop where we read distance from the ultrasonic sensor, check new turn commands from BLE, read the IMU X-axis acceleration to detect braking, and pass these signals into a small rear state machine. The state machine decides the current brake mode (idle, warning, burst, link-loss) and turn mode (left, right, hazard, off), and the outputs are then mapped to the brake LED, turn LEDs, and buzzer.

   The critical drivers are kept small and focused. The timer driver gives us a millis() function using an ISR, and we use this timebase for non-blocking patterns like the buzzer beeping and BLE timeouts. The ultrasonic driver uses the input-capture unit to measure the echo pulse width and convert it into distance. The IMU driver configures the SparkFun LSM6DS0 over I^2C, reads the acceleration registers, and provides a simple deceleration flag for the state machine. The BLE/UART driver parses single-character commands from the front unit and periodically sends back a short status message so we can also detect link loss. Finally, the LED and buzzer drivers just expose simple functions like “set brake mode” or “set turn mode” and apply the correct pin outputs, which keeps the application logic easy to follow.

3. *Demo your device.*

Our group completed the demo with the teaching team on Monday, 11/24.

4. *Have you achieved some or all of your Software Requirements Specification (SRS)?*

   We have achieved most of the SRS items, and we have already collected preliminary data using the lab-available hardware. For SRS-01, although we replaced the original button design with a joystick, the functionality is fully preserved. By sending left/right/hazard commands through the joystick, we verified the correct LED behavior on the rear module. Using our rear BLE logs and visual checks, we confirmed that the system consistently responds with the correct turn-signal mode.

   For SRS-03 , we did not implement the “warning flash” state. After re-evaluating the real-world use case, we felt that introducing a mid-level brake signal was not necessary for safety. Instead, we directly use the burst strobe mode for close-range alerts.

   For SRS-07 , we have not finished the speed display. The accelerometer readings from the LSM6DS0 are still noisy, so we cannot yet compute stable velocity for LCD output. We plan to complete IMU filtering and velocity estimation next week.

   All other SRS items have been implemented and tested. To collect data, we logged ultrasonic readings at ⩾10 Hz, monitored IMU deceleration flags. The BLE link-loss behavior was validated by intentionally powering off the front module and observing that the brake LED enters the correct fail-safe pattern.

5. *Have you achieved some or all of your Hardware Requirements Specification (HRS)?*

   Most HRS items have also been satisfied using the components currently available in the lab. For HRS-01 , we could not test the full battery-powered setup because the DC-DC buck converter module has not arrived. However, we ran the entire system from USB power and confirmed that the voltage remained stable even under maximum load. This gives us confidence that the power requirement will still hold once the final regulator arrives.

   For HRS-03, the accelerometer output still fluctuates more than expected, and this affects speed calculation. We need another round of tuning to stabilize the raw acceleration data.

All remaining hardware requirements have been met. We validated these by taking short walk-tests for BLE, placing fixed objects at 0.2–3 m for ultrasonic calibration, performing brown-out/reset tests to check the default brake LED behavior. The LCD control also works reliably at 5 Hz refresh during basic tests.

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

   Next, we will redesign both the front and rear enclosures based on the updated hardware choices and send the new models for 3D printing. Once all components arrive, we will assemble them with the printed parts. For the LED section in particular, we also need to finalize the LED programming. In parallel, we will continue the final round of functional debugging to make sure all requirements are met, with a special focus on the acceleration/velocity pipeline and its accuracy.

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

   the acceleration data from the IMU is currently not stable enough to compute accurate real-time speed. This is because the LSM6DS0 is capturing not only linear acceleration but also components related to angular motion. In our updates next week, we plan to either remove the influence of angular acceleration in the calculation or add an appropriate filter to suppress it and then evaluate the effect on the speed estimation.

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

   We hope our necessary components can arrive in time.

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.