
Four-Day Project Roadmap (UI-First Approach)

Team Roles:

- **Hardware Lead (Person A):** Physical assembly and wiring.
- **Rover Software Lead (Person B):** Coding the physical rover.py.
- **Mission Control Lead (Person C):** Coding the mission_control.py backend.
- **UI & Virtualization Lead (Person D):** Coding the HTML/JS and the virtual_rover.py simulator.

Day 1: UI Mockup & Full System Simulation

Goal of the Day: Create a fully functional web dashboard that communicates with a "virtual" rover, simulating the entire project in software.

Team Member	Primary Tasks	Details & Objectives
UI & Virtualization Lead (D)	**1. Build index.html & history.html:** Create the complete, polished user interface. This is the **priority task for the day.** **2. Write virtual_rover.py:** Create a new Python script. This script will connect to the MQTT broker and publish a stream of *fake* but correctly formatted JSON telemetry every second. It will also subscribe to the command topic and print any commands it receives.	**Objective:** A complete and visually appealing web UI. A working "simulator" that the Mission Control Lead can use for backend development.
Mission Control Lead (C)	**1. MC Pi Setup & MQTT Broker:** Set up the Mission Control Pi, install all libraries, and configure the Mosquitto broker to accept network connections. **2. Develop Flask Backend:** Create the mission_control.py script with all Flask routes (/, /history, /api/data, /command). **3. Integrate with Virtual Rover:** Write the MQTT subscription logic to receive data from the **virtual_rover.py** script, update the rover_state, and log the fake data to odyssey_log.csv.	**Objective:** A running web server that can be accessed by the team. The UI, when opened, should display the fake data being generated by the virtual_rover.py script. Clicking UI buttons should log commands in the backend's console.

Hardware Lead (A)	**1. Assemble Rover Chassis & Mount Core Components:** Build the frame, mount motors, wheels, L298N, both Pis, and the breadboard. **2. Fritzing Diagram Review:** Sit with Person D to review and confirm every connection on the Fritzing diagram, ensuring it's a perfect blueprint.	**Objective:** A physically assembled, rollable chassis ready for wiring. A 100% verified wiring diagram.
Rover Software (B)	**1. Rover Pi Setup:** Flash the SD card for the Rover Pi, install OS, all necessary Python libraries, and configure network access. **2. Code Skeleton & Pin Definitions:** Create the rover.py file. Define all the GPIO pin variables according to the verified Fritzing diagram. Create the empty function stubs (e.g., def move(): pass, def read_sensors(): pass).	**Objective:** A Rover Pi that is fully configured and ready for hardware-specific code. A clean code structure to build upon.

End-of-Day Milestone:

> **The UI is complete.** Any team member can open a web browser, connect to the Mission Control Pi's IP address, and see a fully functioning dashboard. The telemetry fields will be updating with realistic (but fake) data, and the history page will show charts based on this fake data. This proves the entire software architecture, from UI to data logging, is working.

Day 2: Hardware Assembly & "First Movement"

Goal of the Day: Wire the physical rover and replace the "virtual rover" with the real one, achieving first movement.

Team Member	Primary Tasks	Details & Objectives
Hardware Lead (A)	**1. Wire the Rover:** Meticulously wire the entire rover according to the Fritzing diagram—motors, sensors, ADC, button, and LEDs. **2. Power System Check:** Test the motor power and logic power circuits independently.	**Objective:** A fully wired, "electrically complete" rover. This is the main hardware task for the project.

Rover Software (B)	**1. Implement Motor Control:** Flesh out the move() and stop() functions in rover.py. **2. Implement Basic MQTT Integration:** Write the on_message handler to receive commands from the real Mission Control and call the appropriate motor functions. **3. Standalone Motor Test:** Write a small, temporary script (motor_test.py) to directly test motor functions without MQTT, ensuring the hardware is wired correctly.	**Objective:** The Rover Pi can now physically control the motors. The rover.py script can receive a "forward" command from Mission Control and make the rover move.
Mission Control (C)	**1. Support & Integration:** Assist the Rover Software Lead (B) in debugging the MQTT connection between the real rover and Mission Control. **2. Refine Logging:** Ensure the log file is correctly capturing the initial data streams from the real hardware.	**Objective:** A smooth transition from the virtual rover to the real rover. The backend is now receiving and logging real hardware commands.
UI & Virtualization (D)	**1. UI/UX Refinements:** Based on team feedback from Day 1, make any necessary tweaks to the index.html and history.html files. **2. Create Presentation Slides:** Begin creating the PowerPoint presentation, focusing on the UI, project goals, and architecture.	**Objective:** A polished and finalized UI. A head start on the final presentation.

End-of-Day Milestone:

> The virtual_rover.py script is retired. The real rover, running rover.py, connects to Mission Control. Clicking the "Forward" button on the **final web UI** makes the **physical rover** move.

Day 3: Full Sensor Integration & Autonomy

Goal of the Day: Bring the rover's senses online and implement its "smart" features.

Team Member	Primary Tasks	Details & Objectives
Hardware Lead (A)	**1. Hardware Troubleshooting:** Work with the Rover Software lead to diagnose any sensor-related hardware issues (e.g., bad connections, incorrect wiring). **2. Chassis Refinement:** Secure all wiring and ensure all components are robustly mounted for movement.	**Objective:** All sensors and user components are confirmed to be wired correctly and are providing data to the Pi.

Rover Software (B)	**1. Implement read_sensors():** Write the full code to read from the DHT22, HC-SR04, and the MQ-135 via the ADS1115 ADC. **2. Implement Full Telemetry:** Integrate the real sensor data into the main loop's JSON payload. **3. Implement Autonomous Modes:** Code the logic for Assisted and Autonomous modes.	**Objective:** The rover.py script is now sending a complete, real-time stream of all sensor data to Mission Control. The autonomous modes are functional.
Mission Control (C)	**1. Data Validation:** Monitor the incoming real data stream and the odyssey_log.csv file. Check for null values or errors that might indicate sensor problems on the rover. **2. Finalize Backend Code:** Add comments and clean up mission_control.py.	**Objective:** A robust and finalized backend script. Confirmation that the data pipeline is handling real, and sometimes messy, sensor data correctly.
UI & Virtualization (D)	**1. Finalize Chart.js:** Ensure the history page correctly renders the real data being logged. **2. Continue Presentation:** Build the slides for the Fritzing diagram, hardware, and the live demo plan.	**Objective:** A history page that visualizes real mission data. A nearly complete presentation deck.

End-of-Day Milestone:

> The "Live Telemetry" panel on the dashboard is now populated with real, live sensor data. The rover can successfully navigate autonomously, and all mission data is being correctly logged and can be viewed on the history page.

Day 4: Final Integration, Testing, & Presentation Prep

Goal of the Day: Add the final user features (button/LEDs), conduct full system testing, and prepare for the final presentation.

Team Member	Primary Tasks	Details & Objectives
Hardware Lead (A)	**1. Endurance Testing:** Run the rover in autonomous mode for an extended period. Monitor for any hardware failures or loose connections. **2. Final Assembly:** Charge batteries and ensure the rover is "demo ready."	**Objective:** A physically robust rover that has been tested for a flawless live demo.

Rover Software (B)	**1. Implement Power Button & LEDs:** Add the button interrupt, power_on(), power_off(), and update_leds() functions to rover.py. **2. Bug Hunting & Code Cleanup:** Fix any bugs discovered during endurance testing. Add final comments to the code.	**Objective:** The project is "feature complete." The power button and LEDs work as intended. The rover software is stable and well-documented.
Mission Control (C)	**1. Presentation - Backend & Architecture:** Finalize the technical slides of the presentation. **2. Assist with Full System Test:** Help conduct a full "dress rehearsal" of the final demo, monitoring the backend for any errors.	**Objective:** The technical explanation for the presentation is finalized.
UI & Virtualization (D)	**1. Presentation - Final Polish:** Complete all remaining slides and rehearse the presentation flow. **2. Lead Full System Test:** Run through the complete demo script, from powering on the rover to showing the history page, to ensure a smooth presentation.	**Objective:** A complete, polished, and rehearsed presentation. A confirmed and reliable demo plan.

End-of-Day Milestone:

A fully tested, feature-complete rover ready for a live demonstration. A completed and rehearsed PowerPoint presentation. A clean, commented, and finalized codebase. **The project is complete.**