Final Project '26: Control Sub-Team

1 OVERVIEW

Welcome to the ultimate test to determine the **Best Embedded Control Engineer**! You are three candidates competing to program the next generation of Martian exploration rovers. Your mission is to implement critical control systems on a prototype rover, showcasing your skills in real-time control, communication, and robustness.

Your mission scenario: The rover, piloted by an A-Tier virtual driver, must navigate the rough Martian terrain. It will then encounter a crucial segment—a narrow, unstable bridge conveniently marked with a **black guideline**. Precision is paramount here; any deviation means mission failure. The erratic Martian atmosphere and uneven ground demand a finely tuned **PID system** to ensure the rover's stability and survival across the bridge.

You will work with a provided rover equipped with an **ESP32 microcontroller** and necessary sensors (IR for line following). You must develop the software for both the rover (ESP32) and a ground control station, utilizing **ROS 2** for commanding and telemetry. This is a **solo project**.

2 REQUIRED CHALLENGES (TOTAL: 400 POINTS)

2.1 CHALLENGE #1: REMOTE PILOT INTERFACE (150 POINTS)

Demonstrate your ability to establish robust, low-latency, real-time control over the rover.

1. **Ground Station Control:** Develop the **ROS 2** ground station software to translate operator input (e.g., joystick/keyboard) into velocity commands.
2. **Wireless Command Link:** Implement reliable, wireless communication between the **ROS 2 ground station** and the **ESP32**.
3. **Rover Actuation:** Program the **ESP32** to receive commands and translate them into motor control signals (e.g., PWM) to precisely manage the rover's speed and direction.

2.2 Challenge #2: Autonomous Bridge Navigation (150 points)

Program the rover for mission-critical autonomous navigation across the unstable bridge.

1. **Sensor Integration:** Use the rover's **IR line-following sensors** to detect the black guide line.
2. **Line Following Logic:** Implement the necessary logic on the **ESP32** to ensure the rover autonomously follows the detected line.
3. **Mode Switching:** Implement a safe transition mechanism between **Remote Control** and **Autonomous** modes, commanded from the ROS 2 ground station.

2.3 Challenge #3: Stability Control System (100 points)

Prove your expertise in closed-loop control by implementing a PID system to maintain stability.

1. **PID Implementation:** Integrate a **Proportional-Integral-Derivative (PID) control loop** into the autonomous line-following system (Challenge #2).
2. **Tuning for Precision:** Tune the PID gains (Kp, Ki, Kd) to achieve **smooth, precise line tracking** and minimal oscillation, essential for crossing the narrow bridge.

2.4 Bonus Challenges (Up to 100 Points)

These tasks are optional but crucial for securing the title of the Best Embedded Engineer.

2.4.1 Bonus #A: Vision-Assisted Precision (50 points)

Enhance the line-following capability by incorporating a camera (if available) and image processing to increase line-tracking robustness and precision over the IR sensors.

2.4.2 Bonus #B: Command and Telemetry Interface (50 points)

Develop a simple **Graphical User Interface (GUI)** on the ground station that interfaces directly with the **ROS 2** system:

1. Displays key rover telemetry (e.g., current mode, sensor readings, motor commands).
2. Allows the operator to send control commands and switch modes via the GUI.

---

3 DELIVERABLES

Your final submission will consist of a live demonstration of the working system and a concise **Technical Report**.

3.1 CODE SUBMISSION

- All source code for the **ESP32** and the **ROS 2 Ground Station**.
- Clear inline comments and a README for build instructions.

3.2 TECHNICAL REPORT (MAXIMUM 5 PAGES)

The report must focus purely on the control system and software engineering aspects:

- **Software System Architecture:**
  - Explanation of the **ROS 2 communication architecture** (nodes, topics, message types) between the ground station and the ESP32.
  - Diagram illustrating the data flow for both Remote Control and Autonomous modes.
  - Discussion of any **special ESP32 microcontroller features** (e.g., timers, low-power modes, specific peripherals) utilized to optimize performance or timing-critical tasks.
- **PID Control System Deep Dive:**
  - Detailed explanation of the **PID error definition** (e.g., based on line deviation) and how the output is applied to motor control.
  - Documentation of the **PID tuning methodology** and the final Kp, Ki, Kd values used, with a focus on stability and response time.
- **Autonomous Logic:**
  - Description of the **line-following algorithm** implemented.
- **Bonus Tasks (if completed):**
  - Brief description of the implementation for any completed bonus tasks.
