CHALLENGE 1

AUTONOMOUS RESCUE ROVER FOR DISASTER ZONES

Overview

- The project is a Flask web application that allows users to control an autonomous rescue rover, retrieve its status, and view sensor data through a web interface.
- The app establishes a session with an external API and makes API calls to move the rover, check its status, and charge it if the battery is low.
- The application fetches and displays real-time sensor data, including position (X, Y), battery level, ultrasonic, infrared, RFID detection, and accelerometer readings.
- The web interface provides directional movement controls (Up, Down, Left, Right) and visual elements like a radar scanner, battery indicator, and sensor status table to enhance user experience.
- The system integrates an RFID-based alert mechanism where detected RFID tags trigger a visual pulse animation that turns red for critical proximity warnings.
- A Pygame-based simulation where an autonomous rover navigates the world to locate and "rescue" humans while avoiding obstacles. The rover interacts with an external API for movement, sensor data, and battery management.

Tech Stack

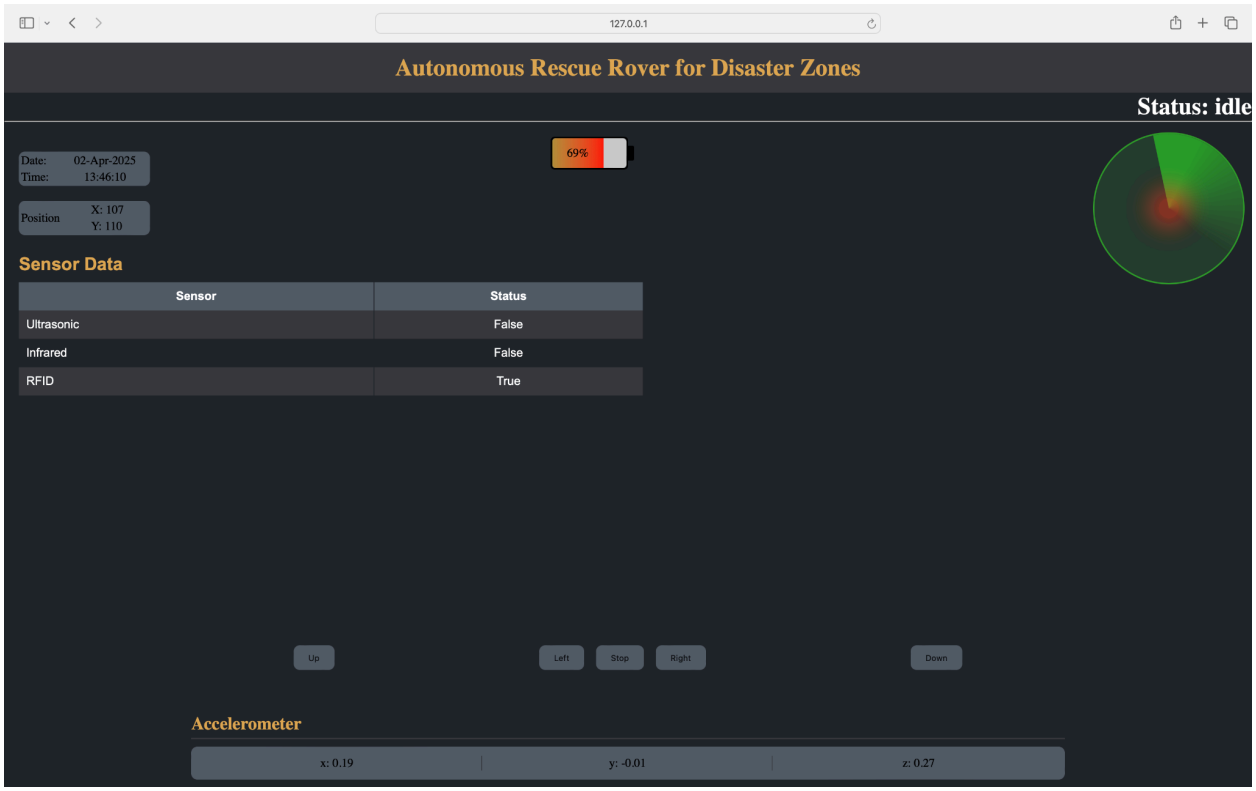
1. Backend Tech Stack:

HTML, CSS, Javascript, Python, Flask

2. Front End Tech Stack (Simulation):

   Python, Pygame

Sensorical Dashboard



- ➔ The dashboard includes sensor data such as position, obstacles, battery status, RFID detection, and manual override.
- ➔ Users can easily control and maneuver the rover as needed.
- ➔ Also indicates whether the rover detected any RFID nearby visually using a Radar chart.

2D Simulation of Rover

Grid-Based Navigation

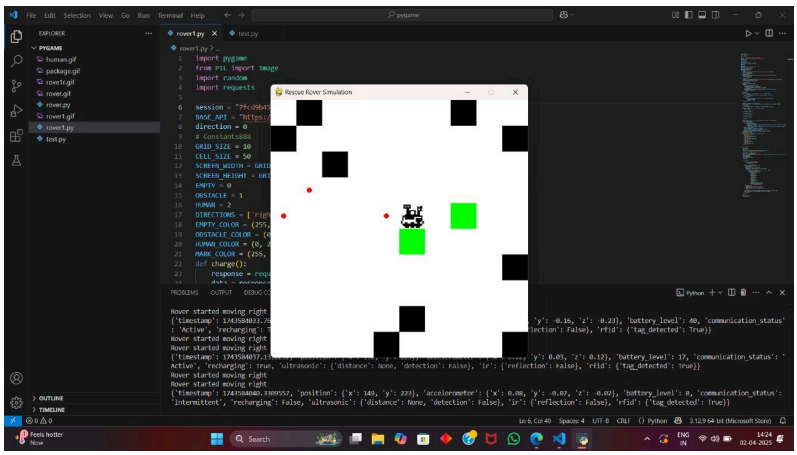
- The grid contains **empty spaces, obstacles, and humans**.
- The rover moves based on API commands (right, left, up, down).
- Movement is visually represented using a **GIF animation**.

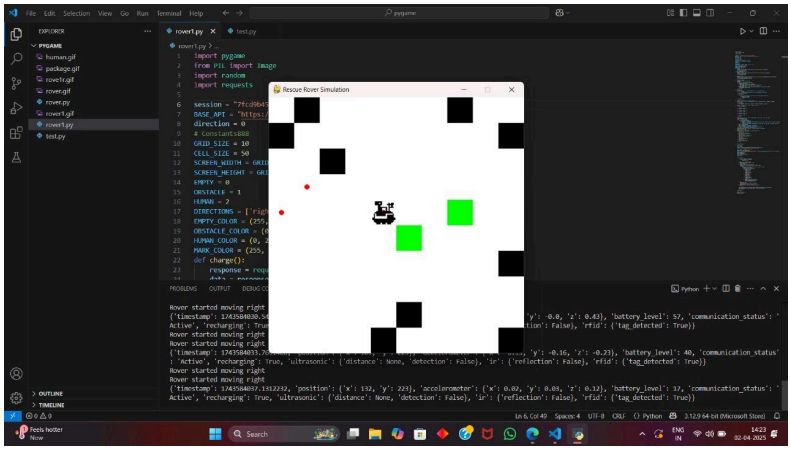
Obstacle & Human Detection

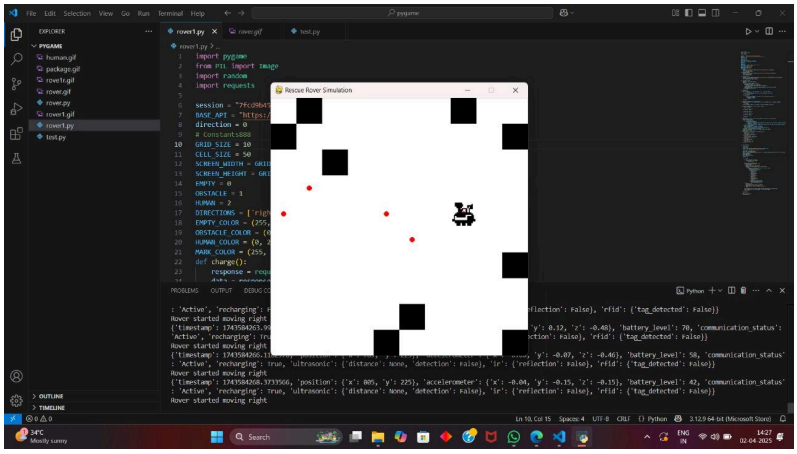
- Uses **sensor data API** (ultrasonic and RFID) to detect obstacles and humans.
- If an obstacle is detected, movement stops.
- If a human is detected, the rover moves towards them using **basic directional logic**.

Simulation Execution

- Uses **Pygame for visualization** (grid rendering, animations).
- Continuously updates the grid state and rover position.
- Ends if the user exits or an obstacle blocks further movement.







TEAM ID/NAME: 78965 - Fanaticus

Sivarathinam S D

Harshavardhan V

Joel Kirubakaran G

Lalith Adithya S

Github link: https://github.com/SivarathinamSD/ARRDZ

Thank You