***Graphical User Interface***

**GitHub:**  <https://github.com/yaswanthsingampalli/Endure_robotics/tree/main/robot_ui_launcher_ws>

|  |
| --- |
| **Prerequisites**   * ROS 2 Humble installed and sourced. * Python 3 with Flask and Gunicorn installed (Check Readme). * Both robot\_ui\_launcher\_ws and airport\_navigation\_ws workspaces cloned and accessible. |

**Workspace Overview:** The robot\_ui\_launcher\_ws is a ROS 2 workspace focused on providing a web-based user interface (UI) for controlling and monitoring the autonomous robot system. It hosts a Flask-based backend server and a frontend web UI, enabling operators to launch, stop, or manage robot navigation missions easily through a browser.This workspace acts as a bridge between the operator and the robot's core ROS 2 navigation stack, providing an intuitive interface for mission management without needing direct command-line interaction with ROS 2.

**Purpose and Benefits**

**1) Simplifies Mission Control:**  
Operators can start, stop, and restart navigation missions with a click, improving operational efficiency.

**2) Removes Need for Direct ROS Interaction:**  
The UI abstracts away complex ROS 2 commands, making robot control accessible to non-expert users.

**3) Supports Remote Access:**  
The server runs on port 8081 and can be accessed remotely via any web browser on the same network.

**4) Production-ready Backend:**  
Uses Gunicorn WSGI server for stable and performant HTTP request handling.

**Repository Structure:**

**Scripts Description**

**Startup Automation (ui\_server.py):**

**A Flask-based web server implementing API endpoints for:**

Launching ROS 2 navigation missions

Stopping or restarting existing launches

Monitoring mission status in real-time

Communicates with ROS 2 using subprocess calls to run launch files and system commands.

Serves the web UI frontend (HTML/JS) to clients.

**Backend (Flask Server -** **start\_server.sh):**

**Automates the startup sequence:**

Builds both the UI workspace (robot\_ui\_launcher\_ws) and the main robot navigation workspace (airport\_navigation\_ws).

Sources the necessary ROS 2 environment.

Starts the UI server using Gunicorn on port 8081.

Opens the default web browser automatically to the UI page (http://localhost:8081).

**Web Interface (Frontend – index.html, styles.css, app.js):**

**Buttons:**

**Start Mission:** Calls /start, launches navigation.

**Stop Mission:** Calls /stop, kills the launch process.

**Restart Mission:** Stops and then starts navigation.

**Manual Control:** Placeholder for future manual override feature.

**Status Display:**

A live text indicator (**Status: Idle / Launched / Stopped / Running**) with color-coded updates reflects backend responses.

|  |
| --- |
| **Example JS Function:**  function sendCommand(cmd) {  fetch(`/${cmd}`)  .then(response => response.json())  .then(data => {  document.querySelector('.status-text').innerText = data.status;  // Update color based on state  });  } |

**Launching the UI Server**

1. **Make sure ROS 2 is sourced:**

source /opt/ros/humble/setup.bash

1. **Run the startup script:**

chmod +x start\_server.sh

./start\_server.sh

1. **This will:**

Build necessary workspaces

Start the UI server on http://localhost:8081

Open the UI automatically in your default browser

**Accessing the UI**

**Open your browser and navigate to:**

http://<robot-ip>:8081

This interface lets you manage robot missions visually.

**How It Helps the Robot System**

Bridges Human-Robot Interaction: Operators don’t need to know ROS commands or manage multiple terminal windows.

Improves Reliability: Centralized mission control reduces manual errors.

Enables Remote Operation: Run or rerun missions from anywhere on the network.

Supports Rapid Development: Easy to integrate new features or diagnostics in the UI backend.