**ERC Hackathon Overall Report File**

**2) Mechanical**

**2) Electronics**

**1. TinkerCAD Simulation**

The entire electronics system for the autonomous electrical surveillance robot has been designed and simulated using TinkerCAD. This simulation ensures that all electronic components function correctly and interact as intended. The TinkerCAD simulation link is provided inside document for Electronics.

**2. Electronics Design**

The electronics system consists of several key components and considerations:

**2.1 Components and Specifications:**

1. **Arduino UNO:**
   * **Purpose:** Acts as the central controller for managing sensors, motors, and overall robot operations.
   * **Details:** Handles the logic for movement control, sensor data processing, and communication.
2. **Motor Drivers:**
   * **Selection:** Chosen based on the power and control requirements of the motors.
   * **Purpose:** Controls the speed and direction of the motors according to commands from the Arduino.
3. **Power Supply:**
   * **Battery:** Lithium-Polymer (LiPo) or Lithium-Iron-Phosphate (LiFePO4) battery is selected based on the power requirements and operational needs.
   * **Voltage and Capacity:** Chosen to match the power needs of the robot and provide sufficient runtime.

**2.2 PCB Design:**

**The PCB design has been created using KiCad and includes three main files:**

1. **KiCad Project File (.kicad\_pro):**
   * Contains the project configuration and setup details. It includes references to all design files and settings used in the PCB design.
2. **Schematic File (.sch):**
   * Provides the circuit schematic, detailing the electrical connections and component placements. It represents the logical design of the electronics system.
3. **PCB Layout File (.pcb):**
   * Contains the physical layout of the PCB, including the placement of components, routing of traces, and design of the PCB layers. It translates the schematic design into a printable and manufacturable circuit board.

**3) Autonomous**

**1. Path Planning and Autonomous Operation**

The autonomous electrical surveillance robot utilizes the A\* (A-star) algorithm for efficient path planning. The robot is designed to navigate to various cones, where it will detect whether the site is affected or not. The system is composed of three primary nodes, each responsible for distinct tasks:

1. **path\_planner.py**: This node calculates the optimal path to the next cone using the A\* algorithm. It continuously updates the path based on real-time data and publishes it for the controller to use.
2. **color\_detection.py**: Once the robot approaches a cone, this node handles color detection to assess the condition of the site. It starts color detection based on signals received from the path planning node.
3. **controller.py**: This node oversees the overall control of the robot, including movement based on the path provided by path\_planner.py and coordinating tasks like color detection through color\_detection.py.

**2. Node Execution Order**

To ensure smooth operation, the nodes should be run in the following order:

1. **Path Planner Node (path\_planner.py)**:
   * Responsible for computing the path to each cone and publishing this information.
   * This node initiates the path planning process and updates the path data for the robot's movement.
2. **Color Detection Node (color\_detection.py)**:
   * Starts color detection when the robot reaches a designated proximity to a cone, based on a signal from path\_planner.py.
   * The node processes camera images to detect the color and determine if the site is affected.
3. **Controller Node (controller.py)**:
   * Manages the robot’s movement according to the path received from the path planner.
   * Coordinates with the color detection node to start and stop detection as necessary.

**3. Launch File Issues**

There were challenges with implementing the nodes in the launch file. To circumvent these issues, the nodes can be manually executed in sequence to ensure there are no errors. The manual execution order is as follows:

1. Start path\_planner.py to begin path computation.
2. Follow with color\_detection.py to set up color detection capabilities.
3. Finally, run controller.py to manage robot movements and integrate path and detection functionalities.

Running the nodes in this specified sequence ensures that the robot performs its tasks autonomously without encountering errors related to node communication or launch file configurations.

**4. Design Considerations**

* **Communication Between Nodes:** The nodes are designed to communicate via ROS topics. path\_planner.py publishes path information, which controller.py uses for movement control. The color\_detection.py node listens for proximity signals to start detection.
* **Node Dependencies:** Ensure all required dependencies and ROS packages are correctly installed and configured for seamless node operation.
* **Testing and Debugging:** Independent testing of each node is crucial to verify functionality. Tools like rostopic and roslaunch can be used to monitor and debug node interactions and data flow.