Requirements Document

Human-autonomous teamwork of ground and air vehicles

Team Members
Yavanni Ensley, yensley2022@my.fit.edu
Younghoon Cho, ycho2021@my.fit.edu
Jaylin Ollivierre, jollivierre2022@my.fit.edu

Faculty Advisor
Thomas Eskridge, teskridge@fit.edu

Table of Contents

1. Introduction

1.1 Purpose

- The purpose of this document is to outline the functional, interface, and performance requirements for our project 'The Human-autonomous teamwork of ground and air vehicles'. This project is meant to create a means for the continuous compositional control of ground and air vehicles to allow agents to work together with/without the assistance of human operators. This document serves as a reference for both the customer and the development team, ensuring that both parties are clear on what the product will have the ability to do from a functional standpoint.

1.2 Scope

- The system will support the control of drones, tanks, and cars across different mediums.There will be an interface allowing for collaboration between the agents, creating a way for them to communicate, with each other, their needs and abilities. This will be able to be done locally and remotely.

1.3 Definitions, Acronyms, and Abbreviations

Agent – A software or hardware autonomous entity (e.g., ground vehicle, aerial drone) capable of executing specific tasks and exposing its capabilities via ROS2.

Mission Operator – A user responsible for monitoring and directing autonomous vehicles, with authority to control multiple agents.

Field Operative – A user responsible for monitoring mission status and receiving updates, with limited system control access.

ROS2 (Robot Operating System 2) – A middleware framework used for robot communication, discovery, and coordination.

SLAM (Simultaneous Localization and Mapping) – A computational method for robots to build a map of an unknown environment while tracking their own location within it.

GPS (Global Positioning System) – A satellite-based navigation system used to determine the global position of agents.

HTMX – A lightweight JavaScript library that allows for dynamic HTML updates without requiring full page reloads.

TailwindCSS – A utility-first CSS framework used to style the frontend of the application.

PostgreSQL – An open-source relational database management system (RDBMS).

MongoDB – An open-source NoSQL database system that stores data in JSON-like documents.

Frontend – The client-side portion of the system where users interact with the application interface.

Backend – The server-side portion of the system responsible for data management, communication with agents, and business logic.

Node (ROS2) – A process that performs computation in the ROS2 ecosystem. Each agent is represented as a node.

Namespace (ROS2) – A logical grouping of nodes, topics, and services in ROS2 to organize communication.

1.4 References

[1] "ROS 2 Documentation: Foxy." *ROS.org*, Open Robotics, https://docs.ros.org/en/foxy/index.html.

[2] "Simultaneous Localization and Mapping (SLAM)." *MathWorks*, The MathWorks, Inc., https://www.mathworks.com/discovery/slam.html.

[3] *Tailwind CSS Documentation.* Tailwind Labs, https://tailwindcss.com.

[4] *HTMX Documentation.* Big Sky Software, https://htmx.org/docs.

[5] *PostgreSQL Documentation.* PostgreSQL Global Development Group, https://www.postgresql.org/docs.

[6*MongoDB Documentation.* MongoDB, Inc., https://www.mongodb.com/docs.

2. Overall Description

2.1 Perspective

- This system will be a web-based platform that integrates with ROS2 to manage an environment between autonomous ground and aerial vehicles. This is our middle layer between the human operator and the independent agents, providing human oversight, and compositional control to provide guidance when necessary. The interface will be a combination of existing frameworks: ROS2, GPS/SLAM mapping, wireless communication and

2.2 Functions

- This system will support multi-agent monitoring through a map and status overview. The agents in the system will communicate their status on the interface to help the others build their behaviors. This allows the agents to help each other

based on how they interpret the statuses of the neighboring agents. Offer the ability to control one or more agents at a time. Outside users will be able to externally access the system through a remote server.

2.3 User Characteristics

- Mission operators: Trained users that will know how to control and reassign the robots with different tasks. Operators will have access to all system functions, live feeds, and interfaces to interact with the agents.
- Field Operatives: Observers of mission/task progress. They will not have complex/administrative access to control vehicles, but will be able to see what is happening and what the directives are for the agents.

2.4 Constraints

- Network latency when operating over networks that could be unstable/inconsistent.
- ROS2 Compatibility with all of the agents involved.
- The system has to run on modern browsers with HTMX/TailwindCSS frontend support.
- The database must have compatibility with PostgreSQL or MongoDB for the project's infrastructure.

2.5 Assumptions and Dependencies

- Each robot can send data relevant to its location.
- GPS or SLAM mapping are consistently available for mapping and give accurate information
- The database is reliable, available, and configured.
- All agents can run ROS2 nodes compatible with our backend.
- Need robots to be on a synchronized clock.
- Secure authentication for role-based access.
- ROS2 discovery of active agents.
- The system can be scaled to a larger or smaller number of robots based on foundation.

3. Specific Requirements

3.1 Functional Requirements

The system shall:

- Authenticate users and ensure proper permissions are enforced.
- Discover the available agents using ROS2
- A dashboard to show, agents, their locations, and the status of their mission

- Allow the control of select agents using the input schemes required for each of the vehicles.
- Enable monitoring of multiple agents
- Support the handoff/collaboration of tasks from agent-agent
- Report/Notify of any issues encountered during tasking

3.2 External Interface Requirements

- User: Web application with an interactive design (desktop and mobile)
- Hardware: GPS modules, cameras, and sensors
- Software: Integrates with ROS2, PostgreSQl/MongoDB, and our backend python server

3.3 Performance Requirements

- The system shall update agent status and map location with latency ≤ 1 second in stable network conditions.
- The system shall support at least **10 simultaneous active agents** without significant degradation.
- Map and control feeds shall refresh at least **5 times per second** for smooth monitoring.

3.4 Security Requirements

- All remote connections must use encrypted channels (SSL/VPN).
- Role-based access control must prevent field operatives from sending commands.
- The system shall log all operator interventions for accountability.

3.5 Non-Functional Requirements

- Scalability: The system must support the addition of agents and robot types with minimal reconfiguration.
- Usability: Interface should be intuitive and user-friendly, with simple navigation for both operators and field operatives.
- Maintainability: System codebase should be modular, leveraging ROS2 packages and well-documented APIs to ensure consistent upkeep.