PROJECT REPORT

BLUE PROJECT

DONE

Trieme (2024)

Author(s):

Famura

Contents

1. Project Description

1.1. Introduction

Trieme V1 is designed for completing Teknofest 2024 Unmanned Surface Vehicle competition course. Unfortunately, project's sources are classified. Thus, this report does not contain any code. However, I (Famura) alone designed the system and have right to explain and visualize the architecture. Thus, images regarding the project are present in the report. Also this report is only focused on the software of the Trieme

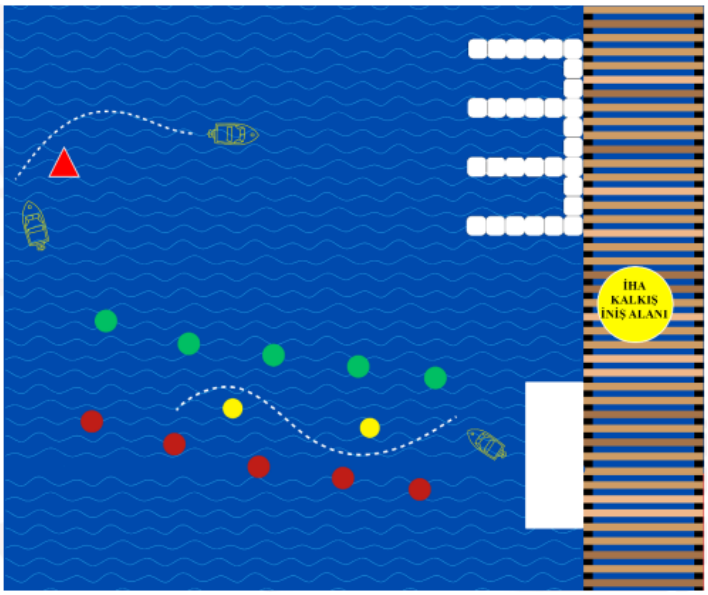
The team was formed in the February, the competition was in early July and with the software team containing only two people (me included) we developed an algorithm for ship's control. With plans of using an AI model for next year.

1.2. Competition Details

1.2.1. Course Details

Course was divided into three different parts:

- Parkour 1
- Parkour 2

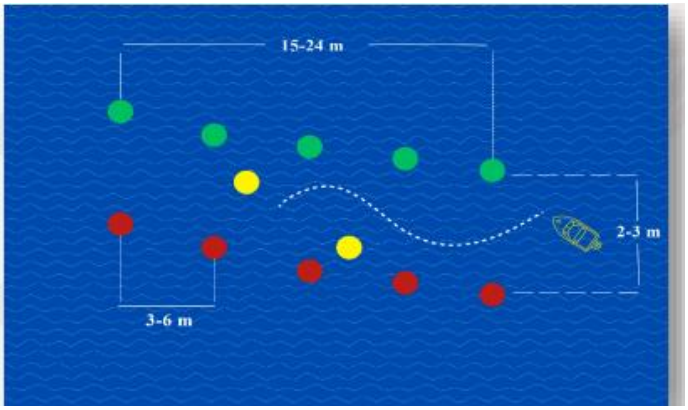


The ship must complete all these parkours in one continuous attempt and should come back.

We were allowed two attempts for completing whole course.

While on the course, the ship was not allowed to touch anything or go off course. If any of these happen, we would get a penalty point for any of these possibilities. Repeating penalty points would mean disqualification.

Here are detailed explanations of these parkours:

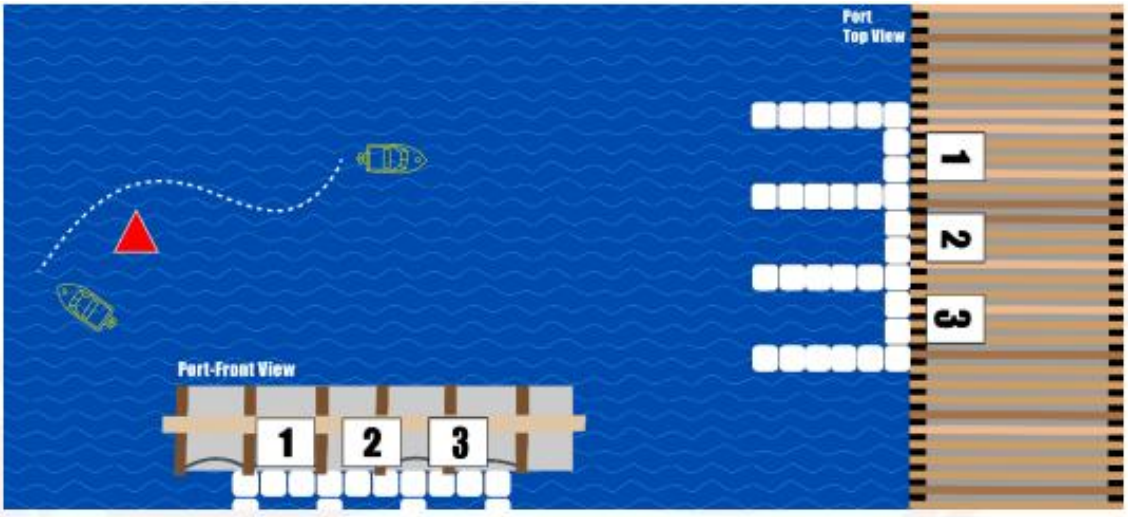


1.2.1.1. Parkour 1

Parkour 1 is added to course for testing route finding, image detection and obstacle avoiding algorithms of the ship. While on route, ship will have border obstacles and avoid obstacles. Left border obstacles are red, right border obstacles are green, middle avoid obstacles are yellow. The ship must complete the parkour while remaining between border obstacles and not colliding with anything.

1.2.1.2. Parkour 2

Parkour 2 is added to the course for testing communication and route-finding algorithms. An autonomous drone should rise and get the data of which parking space should be chosen out of existing three spaces. Then the drone should send this data to ship and ship should park in that parking space.



1.2.2. Main Requirements

There are some main requirements for ship to be able to join the competition that are handled by Software Team:

- There should be a Ground Control Station (GCS)
- The ship must be controllable from GCS
- All the telemetry data should be accessible from GCS
- The ship must record her surroundings as a map. And this map should be accessible from GCS in real time
- The ship also must be controllable manually by GCS or by any RC Controller.
- The ship must have 3 modes; stopping, manual, autonomous. The users must be able to switch between these modes whenever they want
- The ship must flash lights according to her mode. Red if stopping, yellow if manual, green if autonomous

2. Project Technologies

2.1. Module(s)

- **Windows (Windows 10):**
  The GCS is made to run on Windows. It was used on Casper Excalibur G770.
- **Ubuntu (Ubuntu 22.04):**
  The Trieme is made to run on Ubuntu. It was used on Jetson Orin Nano.
- **Rasbery Pi OS:**
  The drone is made to run on Rasbery Pi OS. It was used on Rasbery Pi 4.

2.2. Coding Language(s)

- **Python - Pycharm**
  The entire project is written in Python and is set to interpret in Python3. All of the Venv's were serviced to bios and was starting when the required electronic module got power.

2.3. Libraries & External Additions

- **Pyserial:**
  Allows communication between electronic modules. Telemetry modules were chosen by electronic department of the team regarding their concern frequency conflict with Wi-fi and mobile phone services
- **YOLO:**
  Used image processing.
- **Threading:**
  Using "Threading" for handling the communication in parallel.
- **PyGame:**
  Used for showing telemetry data and live map on the GCS
- **PyMavlink:**
  Used for connecting Jetson Orin with Pixhawk module

3. Project Architecture

The architecture is divided into sequential phases, each controlled by the onboard Jetson Orin computer and coordinated with Pixhawk, PyMavlink, and ROS frameworks. The system ensures continuity by saving progress and dynamically switching between scenarios.

3.1. System Flow

3.1.1. System Initialization

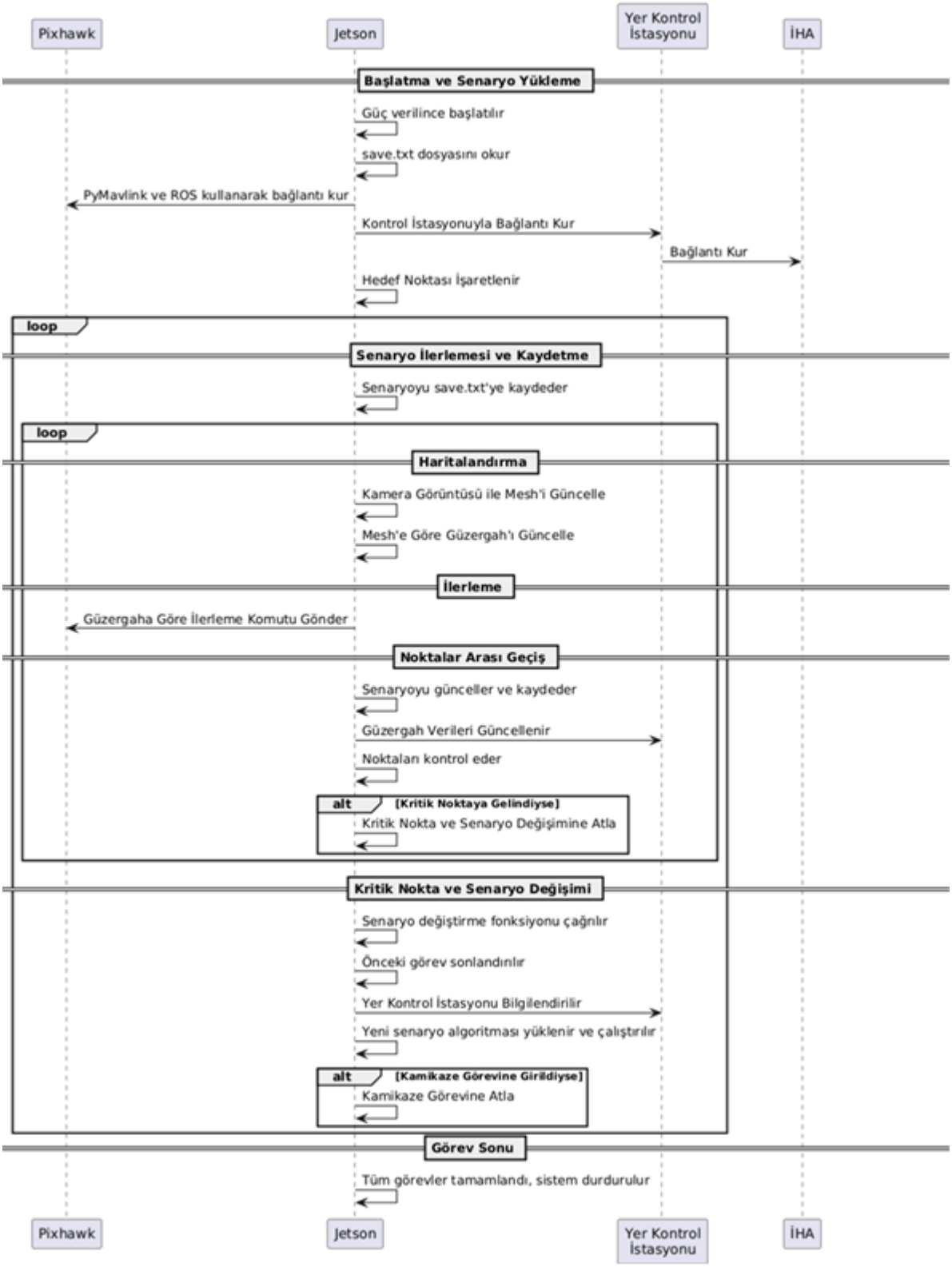
- **Power On**
  When power is supplied, the Jetson Orin board is activated and the system starts.
- **Load Save File**
  The system reads the stored save file to determine the current mission scenario. This ensures that after any interruption, the vehicle can resume from the last known state.
- **Scenario Awareness**
  Each scenario is defined by a fixed number of checkpoints. The vehicle uses these checkpoints to recognize its current mission and to transition between scenarios

3.1.2. Mission Execution

- **Progression Function**
  Once the progression function is triggered, Jetson Orin establishes communication with the Pixhawk flight controller via PyMavlink and ROS. This enables the ASV to move forward autonomously.
- **Scenario Transition**
  At the end of each scenario, the vehicle reaches a *critical point*. Passing this point triggers the

scenario-switching function, which terminates the previous mission and loads the algorithms required for the next one.

- **Finalization**
  After the last critical point is reached, the USV concludes all missions and shuts down operations.



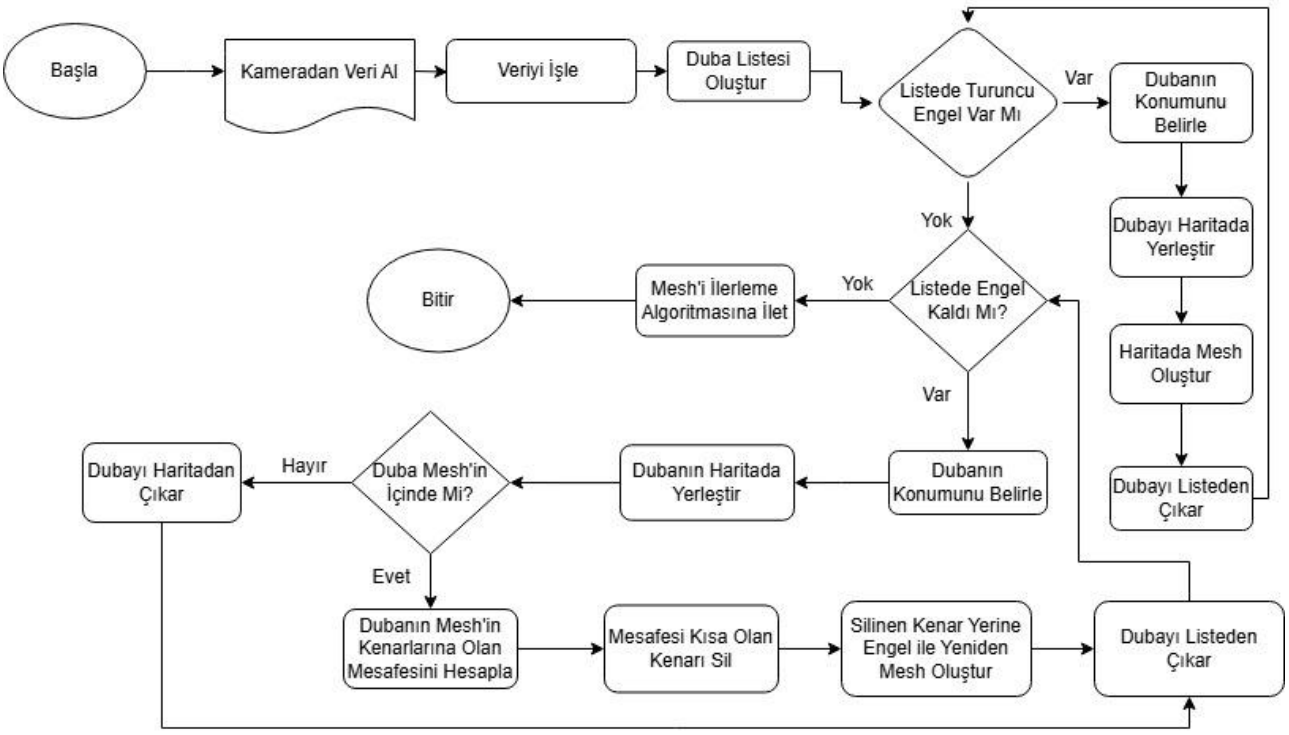
3.1.3. Communication with UAV

- **Data Request**
  For the parking mission, the USV requests target information from the GCS
- **Relay Mechanism**
  The GCS autonomously collects data from the UAV and relays it to the USV. This design ensures that both UAV and USV operate under a unified communication structure, even though the link is extended through the GCS.

3.2. Algorithmic Components

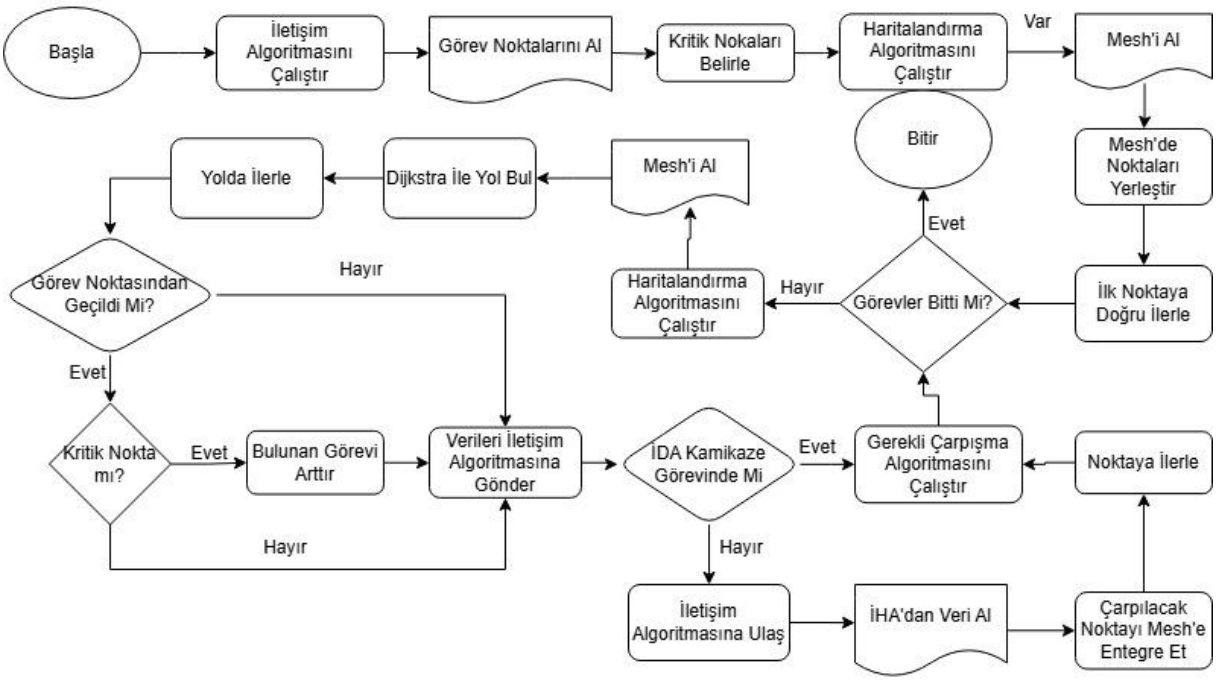
3.2.1. Mapping Algorithm

- Integrates camera input and communication data to build a mesh-based map.
- Recognizes orange buoys as barriers and constructs meshes between them.
- Provides spatial data to the progression and collision control algorithms.



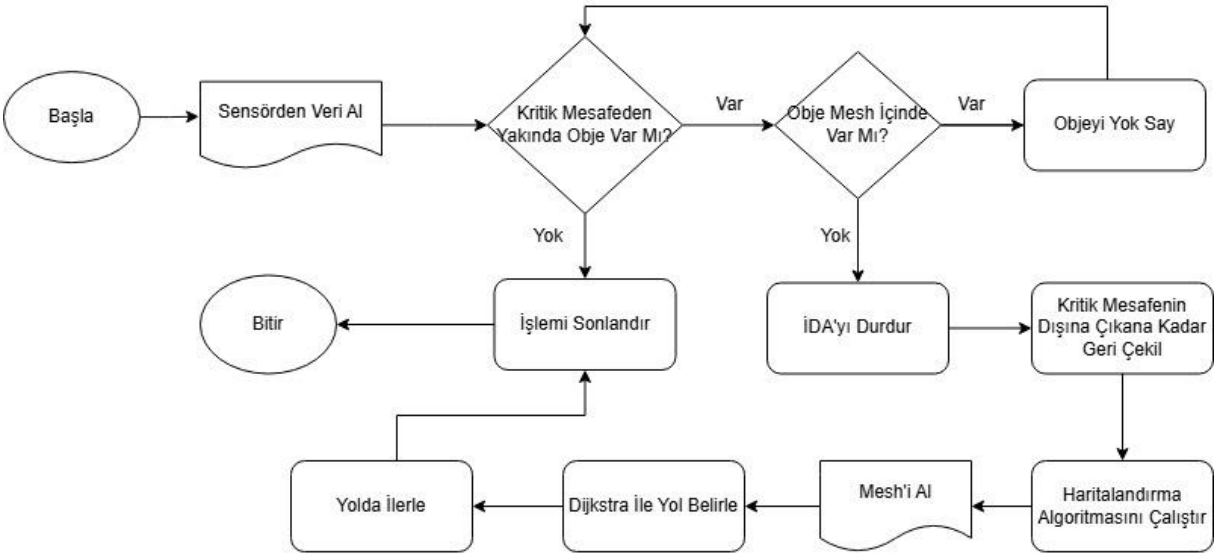
3.2.2. Progression Algorithm

- Core of the system, responsible for navigation and mission control.
- Uses Dijkstra's algorithm on the mesh structure to determine optimal paths.
- Manages the execution of other algorithms.



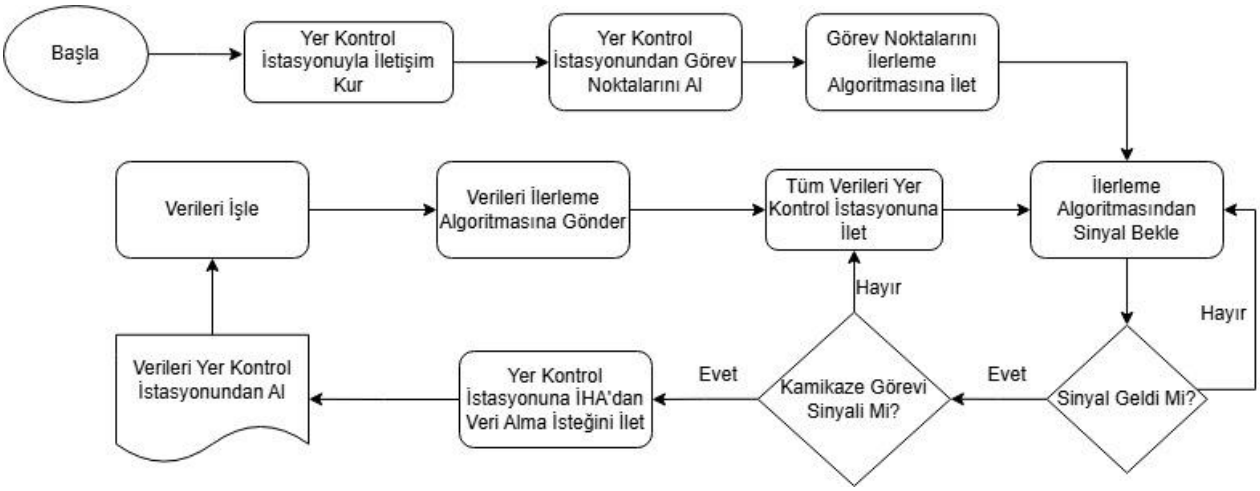
3.2.3. Collision Control Algorithm

- Active in the first two missions.
- Detects unexpected nearby objects not included in the map and avoids them.
- If triggered, it indicates a failure in earlier stages and requires system reset.



3.2.4. Communication Algorithm

- Handles all data exchange with the Ground Control Station.
- Continuously transmits and receives mission data until power is cut.
- Ensures synchronization between USV, UAV, and GCS.



4. Project Analysis

4.1. Test

The project was tested under some physical and simulation testing. Our algorithms were tested under Unity (thus C#) and physical conditions. A video of physical test is in the disk.

The tests showed that our algorithms are working properly but our image processing has some problems. It was working flawlessly at nighttime, but it had some issues during the daytime. Which is interesting because you would expect the opposite. But we later found out that our image processing gets confused with the sun reflections from the sea. This did not cause any problems in the competition.

4.2. Upgradable Things

We wanted to upgrade the Trieme with AI model after seeing other competitors on the field. But we knew that doing that with only two people was going to be a challenge. So, we requested some additional manpower from our team captain.

Regarding our algorithms, mesh system was overengineering. We did not needed this complicated system. We had a second approach to the problem at hand, setting middle points. We could use some edge tracking algorithms and create edges around map. And we could find middle points in these maps so that we can give it to ArduPilot software and have a simpler solution

5. Project Comments

5.1. Creator Comments

2024 competition teached me importance of many things. Including; preparing project reports, managing time in a busy calendar, working under stressful environments, obeying deadlines, usage of AI, usage of software in embedded systems, team communication, working under choises that your boss did but you didn’t agree with etc.

I’m still so grateful for the opportunity. It allowed me to develop myself and my students like Anatoli.

Hooah - Famura

6. Credits

Famura – Software Department Leader

Musab Kılıç – Software Team Member (Mainly Worked on Image Processing)

7. References

- MathWorks. (n.d.). *Dijkstra's algorithm*. In *MATLAB & Simulink documentation*. Retrieved October 18, 2025, from https://www.mathworks.com/help/matlab/ref/graph.shortestpath.html
- OpenCV. (n.d.). *YOLO object detection with OpenCV*. Retrieved October 18, 2025, from https://docs.opencv.org/
- PyGame Community. (n.d.). *PyGame documentation*. Retrieved October 18, 2025, from https://www.pygame.org/docs/
- PyMavlink Developers. (n.d.). *pymavlink: Python MAVLink library*. Retrieved October 18, 2025, from https://www.ardusub.com/developers/pymavlink.html
- Python Software Foundation. (n.d.). *Python 3 documentation*. Retrieved October 18, 2025, from https://docs.python.org/3/
- Raspberry Pi Foundation. (n.d.). *Raspberry Pi OS documentation*. Retrieved October 18, 2025, from https://www.raspberrypi.com/software/
- Robot Operating System (ROS). (n.d.). *ROS documentation*. Retrieved October 18, 2025, from https://www.ros.org/
- Serial Communication Library (PySerial). (n.d.). *PySerial documentation*. Retrieved October 18, 2025, from https://pyserial.readthedocs.io/
- Unity Technologies. (n.d.). *Unity documentation*. Retrieved October 18, 2025, from https://docs.unity.com/

8. Disk Contents

- Trieme.pdf
- Trieme.docx
- Ekip_Fotoğrafı.jpg
- Yarışma_Günü_Videosu.mp4
- Otonom_Testi.mp4
- Video_Mülakat_Videosu.mp4
- Yarışma_Önü_Test_Günü.mp4
- Let It Be by The Beatles