PROJECT REPORT

BLUE PROJECT

DONE

Göksu (2025)

Author(s):

Famura

Contents

1. Project Description

1.1. Introduction

Göksu is designed for completing Teknofest 2025 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 Göksu.

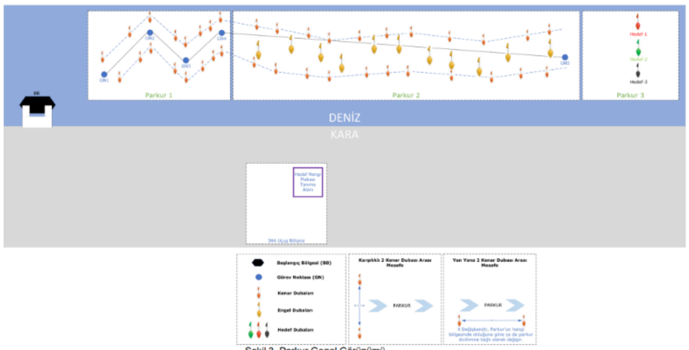
Unlike last year (Trieme), Göksu is designed to run with an AI model. This year, we had 8 months and 4 people to work on this project and we delivered.

1.2. Competition Details

1.2.1. Course Details

Course was again divided into three different parts:

- Parkour 1
- Parkour 2
- Parkour 3



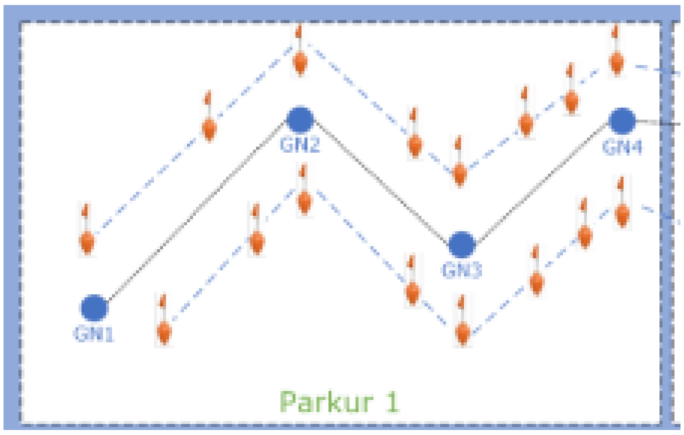
Şekil 3. Parkur Genel Görünümü

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:

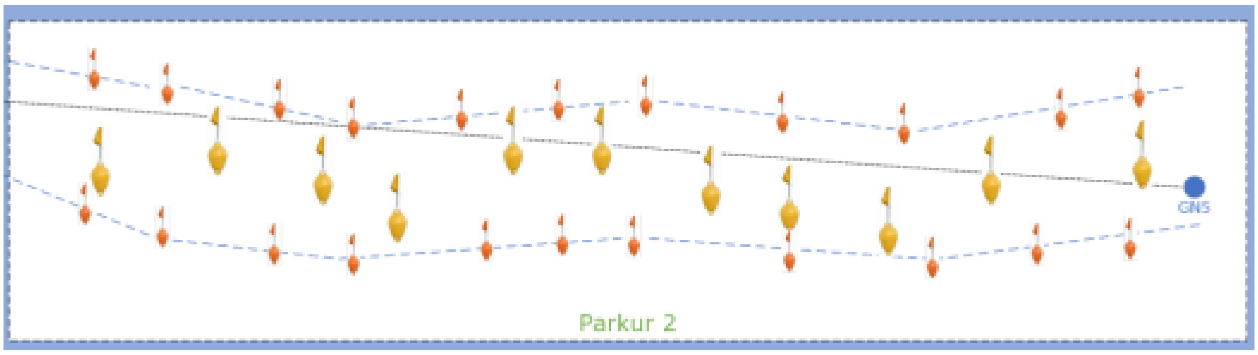


1.2.1.1. Parkour 1

Parkour 1 is added to course for testing route finding and image detection algorithms of the ship. While on route, ship will have border obstacles. Both, left border obstacles and right border obstacles are orange. Which makes classifying border obstacles hard. The ship must complete the parkour while remaining between border obstacles and not colliding with anything.

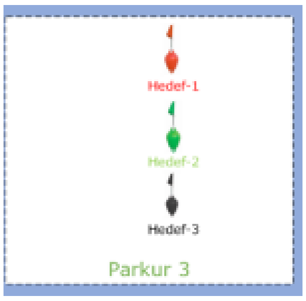
1.2.1.2. Parkour 2

Parkour 2 is added to the course for testing route finding and image detection and obstacle avoiding algorithms. Again; while on route, ship will have border obstacles. Both, left border obstacles and right border obstacles are orange. Which makes classifying border obstacles hard. And there is middle obstacles which are yellow. The ship must complete the parkour while remaining between border obstacles and not colliding with anything.



1.2.1.3. Parkour 3

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 obstacles. Then the drone should send this data to ship and ship should do a kamikaze mission to required obstacle.



1.1.1. 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

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
- **TensorFlow:**
  Powers the learning and decision-making modules within Aelita. It supports neural network training and inference for route prediction and adaptive behavior.
- **WhisperDock:**
  Enables voice input processing. It converts radio-transmitted voice commands into text, allowing Aelita to receive instructions during competition.
- **XTTS-v2:**
  Used for voice output. Aelita responds to commands and communicates status updates via synthesized speech, enhancing human-AI interaction.
- **Unity (C#):**
  Used for simulation testing. Custom environments were built to evaluate Aelita's behavior under controlled virtual conditions before physical deployment.
- **Robot Operating System (ROS):**
  Integrated for modular communication between subsystems. It supports message passing and sensor fusion, especially in multi-threaded environments.

3. Project Architecture

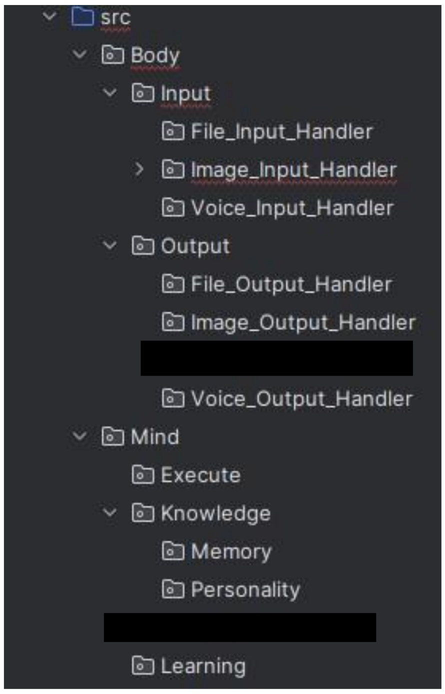
3.1. Overview

The project divides into two main sections: body and mind. The body represents the interfaces for interacting with the external environment, handling inputs like images, voice, and system files, as well as producing corresponding outputs. The mind encompasses the internal cognitive processes, enabling the AI to analyse inputs, store knowledge, communicate, and learn dynamically. The seamless integration of these components ensures Aelita can perform reliably in various challenging marine scenarios.

The team's approach is inspired by the human body's and brain's interaction principles but has been designed uniquely without directly emulating any pre-existing AI architecture. Aelita's functionality reflects real-time decision-making, error handling, and self-correction to navigate small-scale ship designs smoothly. Key tools like TensorFlow for machine learning, Pixhawk for electronic route management, and MAVLink for communication solidify the development strategy.

Adopting Python as the programming language due to its compatibility with hardware and software libraries simplifies development and ensures efficient system integration. As the project progresses, iterative improvements in the AI structure and capabilities ensure adaptability to emerging challenges while meeting competitive expectations.

For our proposed plan, we divided Aelita (thus our work) for different subdivisions and different approaches. This division is made by our software group and our software group alone. It is not inspired by any existing AI. But of course, it is inspired by how human brain and how human body works.



```
src
  Body
    Input
      File_Input_Handler
      Image_Input_Handler
      Voice_Input_Handler
    Output
      File_Output_Handler
      Image_Output_Handler
      [redacted]
      Voice_Output_Handler
  Mind
    Execute
    Knowledge
      Memory
      Personality
    [redacted]
    Learning
```

As seen on the left, we divided Aelita in two main parts and subparts inside of it. We can think their work principle as how class type works in Java.

In the body part, we aim to make Aelita interact with her environment. Since our engine and control components are not made yet, their handlers are not made yet.

In the mind part, we aim to build the Aelita. This part will be like her brain, and we will allow her to make decisions, learn, understand different language in this part.

Let’s dive into each subdivision one by one:

3.2. Body

3.2.1. Input

1) **File_Input_Handler:** This package allows her to read different system inputs from the system itself. She reads system logs; this package comes in handy in these conditions. Also, from time to time we need to communicate with her, and we do it using “.txt” format (in cases like voice communication breaks).

2) **Image_Input_Handler:** This package mainly handles the image processing for Aelita.

3) **Voice_Input_Handler:** This package allows us to communicate with Aelita using voice communication. We use *(1)WhisperDock for this. It allows us to order Aelita using radio during competitions.

3.2.2. Output

1) **File_Output_Handler:** This package is mainly for Aelita's log. She provides her log in ".txt" format from here. She also uses this to create different types of files when needed.

2) **Image_Output_Handler**: This package allows Aelita to output her image processing results so we can check her progress. Since image processing is a tough part of this project, we created a checkpoint for this. She does not constantly do this but delivers when we ask her. This way, we can check if she is doing something wrong or if we did something wrong in the image processing process.

3) **Voice_Output_Handler:** As mentioned before, we communicate with Aelita using radio. And Aelita communicates with us as well. We use *(3) XTTS-v2 for this

3.3. Mind

3.3.1. Execute

This part allows Aelita to connect with her body for outputs. It gathers information about where to connect within the body and, if necessary, connects with the Learning module to decide which body part to use. This is not needed for inputs, because the required body packages are directly used by mind packages.

3.3.2. Knowledge

This part allows Aelita to store data. It is divided into two components. Memory allows her to store all of the data. Personality simulates a person to make her decision-making and risk-taking easier. For creating her personality, we base it on three individuals: Niki Lauda (for an engineer's approach to problems), Margaret Thatcher (for a strong and brave approach), and Misato Katsuragi (for a woman's approach).

3.3.3. Language

This part allows Aelita to understand and communicate with users in both Turkish and English. However, we are likely abandoning this and communicating with her only in English.

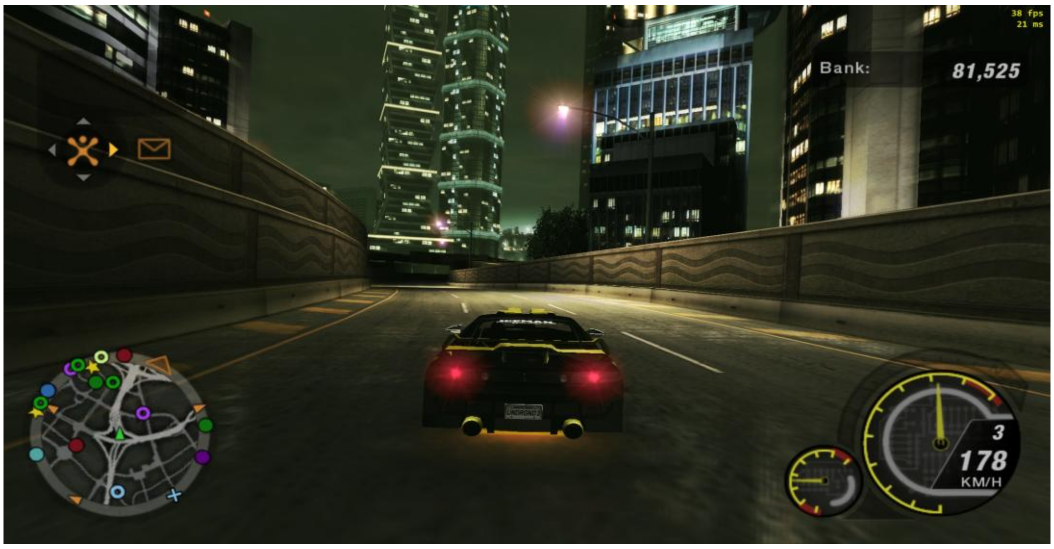
3.3.4. Learning

This part contains the machine learning and decision-making capabilities. It is the most important part of the whole project. We use *(4)Tensorflow for this part. We also divide this part into two: decision and learning, as the project progresses. Since we don't have enough information for decision-making yet, we have not made a specific classification.

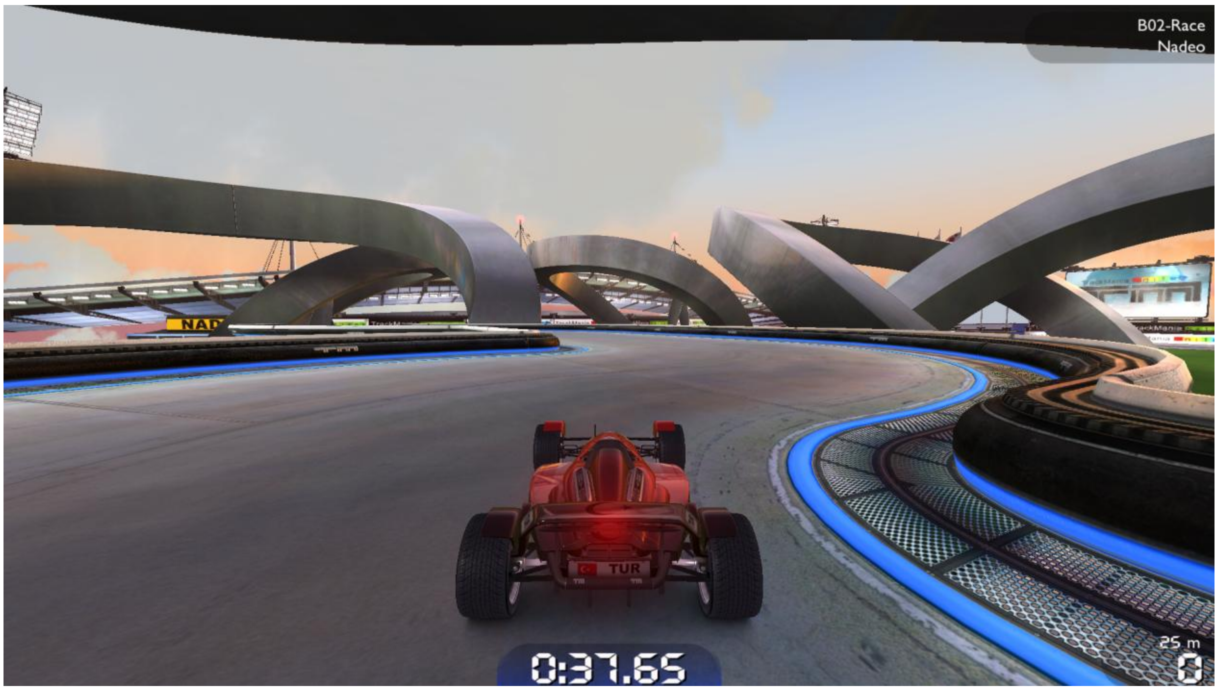
4. Project Analysis

4.1. Tests

The project was tested under some physical and simulation testing. Our model was tested under Unity (thus C#) and physical conditions. Also, we tested Aelita using Need for Speed Underground 2 and Trackmania Nations Forever. Here is a figure of the game:



For NFSU2; on the highways of the game, there is clear borders for Aelita to follow. Thus, we tested and teached Aelita to stay in middle of these borders and stay clear of traffic on the way.



Again, for Trackmania as you can see there is clear obstacles on the left and right side of most of the tracks. This is used for same reason for NFSU2.

4.2. Upgradable Things

In future versions, Aelita’s learning and decision-making modules can be further enhanced using reinforcement learning techniques to improve adaptability in dynamic environments. The voice communication system may also be upgraded to support multi-language interaction and improved natural speech output. Additionally, integrating a more advanced 3D visualization system into the Ground Control Station could offer better situational awareness and real-time map interaction for operators.

5. Project Comments

5.1. Creator Comments

2025 Competition teached me so many things, just like 2024. This time, I learned how to be a veteran, how to be a leader. Not just how to manage everyone that is working for you. Every individual that only wants same thing as you but counting on YOU. Responsibility of messing up and working with constant pressure of it. Hooah - Famura

6. Credits

Famura – Software Department Leader

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

Ali Haydar Sucu – Software Team Member

Abdullah Aksoy – Software Team Member

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/
- Google. (n.d.). TensorFlow: An end-to-end open source machine learning platform. Retrieved October 18, 2025, from https://www.tensorflow.org/
- NVIDIA Corporation. (n.d.). Jetson Orin Nano developer kit. Retrieved October 18, 2025, from https://developer.nvidia.com/embedded/jetson-orin
- ElevenLabs. (n.d.). XTTS-v2: Cross-lingual text-to-speech model. Retrieved October 18, 2025, from https://elevenlabs.io/
- WhisperDock Developers. (n.d.). WhisperDock: Voice recognition and processing toolkit. Retrieved October 18, 2025, from https://github.com/openai/whisper
- Unity Technologies. (n.d.). Unity real-time development platform. Retrieved October 18, 2025, from https://unity.com/
- Electronic Arts. (2003). Need for Speed: Underground 2 [Video game]. Electronic Arts.
- Nadeo. (2006). TrackMania Nations Forever [Video game]. Ubisoft.

8. Disk Contents

- Göksu.pdf
- Göksu.docx
- Ekip_Fotoğrafı.jpg
- Otonom_Testi.mp4
- Image_Proccessing_Test.mp4
- Testing_Photo.jpg
- Uluslararası_Kurula_Sunum.mp4
- Canon In D by Evangelion Series