DD-Boat Navigation and Control

ENSTA Brest 17-24/10/2026

Team Members

Professor: Luc Jaulin

The project was carried out by Group 1, also known as the Banana Team. The team consisted of four members:

- Csaba – Cluj-Napoca, Romania
- Ahmet – Izmir, Turkey
- Aron Tamas – Budapest - Hungary
- Christos – Chania, Greece



Introduction

The objective of this project was to configure and operate a DD-Boat and develop the software required for autonomous navigation. Our team, Group 1 (Banana Team), worked on sensor calibration, heading estimation, remote control, and autonomous navigation missions. Our boats name was Messi because it had the number 10 on it.(We all loved him)

The practical work consisted of three main navigation tasks. First, we calibrated the boat's magnetometer and programmed it to travel 20 metres north. A remote-control client/server system was also used to bring the boat back to the dock. The second task was to make the boat follow a square trajectory. Finally, the third task was a race in which the boat had to navigate to a specified point and return.



Us in class working

Digital “Messi”

Before touching the water, we modelled the boat. Its state is $X = (x, y, \theta)$: position and heading. The two thrusters u_1 (left) and u_2 (right) give:

$\dot{x} = v \cdot \cos\theta$, $\dot{y} = v \cdot \sin\theta$, $\dot{\theta} = \omega$, with $v = k_2 \cdot (u_1 + u_2)/2$ and $\omega = k_1 \cdot (u_2 - u_1)$

We simulated it with Euler steps, $X(k+1) = X(k) + dt \cdot f(X(k), u(k))$, with $dt = 0.05$ s. Equal thrusts give a straight line, and unequal thrusts give a circle. With $u_1 = -u_2$ the boat spins on the spot.

Compass Calibration

The first step was to calibrate the magnetometer so that the boat could obtain more reliable heading measurements. We implemented the Python script `my_calib.py`, which used the `imu9_driver_v2` driver to read raw magnetic measurements from the onboard IMU.

The calibration followed the four-pose method presented in the course slides:

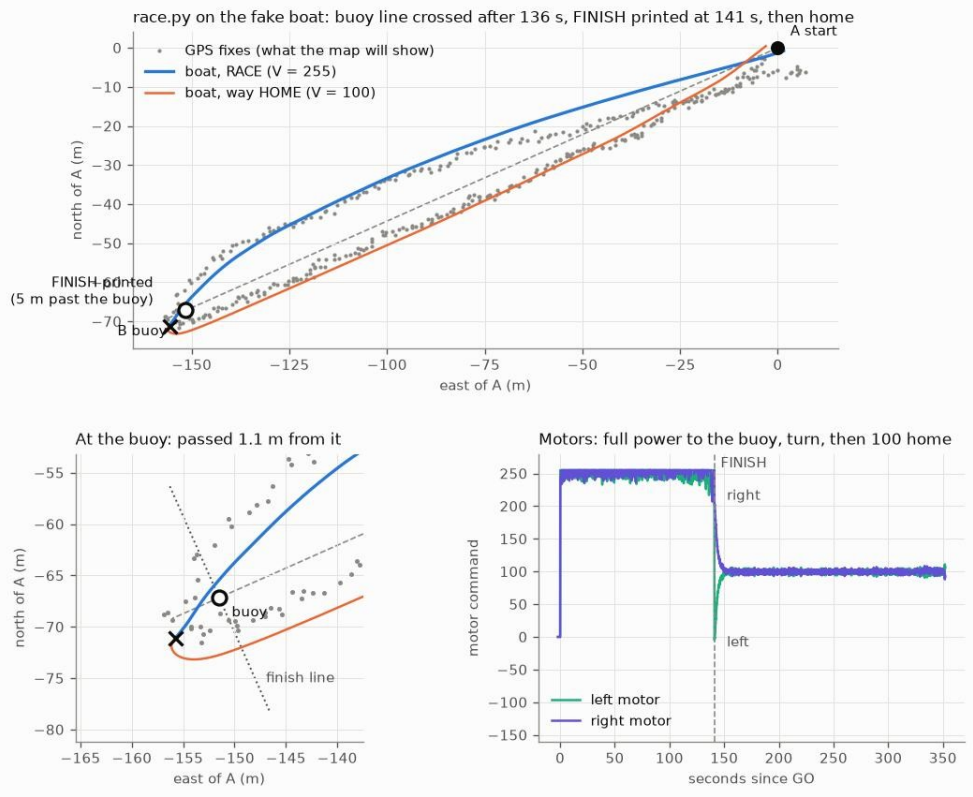
1. The boat was placed flat, with its nose pointing north.
2. The boat was turned upside down, with its nose pointing south.
3. The boat was placed flat, with its nose pointing west.
4. The boat was placed vertically, with its nose pointing towards the sky and its deck facing south.

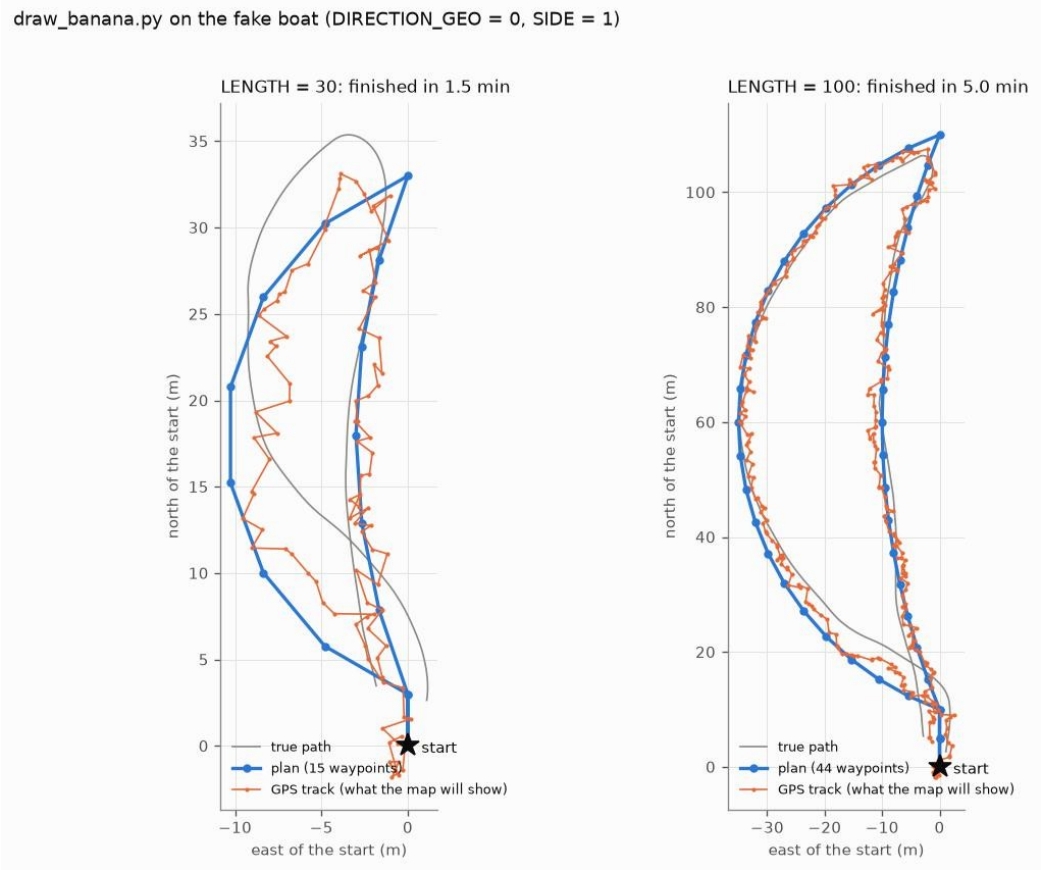
For each pose, the script collected 50 measurements at intervals of 0.1 seconds and calculated their mean to reduce measurement noise.

From the four poses we got an offset b and a matrix A, and corrected every reading with $x = A^{-1}(y + b)$.

Simulation

Before each practical experiment, we tested our control algorithms through simulations in Python. These simulations allowed us to visualize the boat's movement, examine its trajectory, and verify the expected behaviour of the controller before running the code on the physical boat. We used graphical representations of the simulated trajectories to observe the boat's path, heading changes, and turns. This approach helped us identify potential problems and adjust our code before testing it in real conditions.





The Banana shape to honor our team's name

Remote-Control Client/Server

The remote-control client/server system was used to control the boat remotely, particularly to bring it back to the dock after the first navigation task.

The system provided an alternative to autonomous navigation by allowing commands to be sent remotely to the boat. This was useful for retrieving the boat after completing a mission that did not include an autonomous return.

The remote-control functionality complemented the autonomous control system and helped us operate the boat during the practical exercises.

Autonomous Navigation 20 Metres North

The first navigation task was to make the boat travel 20 metres north. Unlike the later missions, this task did not require the boat to autonomously return to its starting position.

The calibrated magnetometer measurements provided the basis for estimating the boat's heading. The objective was to make the boat follow the desired northward direction while travelling the specified distance.

The boat turns harder the further it points away from the target: left = V − K·e, right = V + K·e. V = 100 and K = 40 worked well.

To recover the boat after the mission, we used a remote-control client/server system. This allowed us to control the boat remotely and bring it back to the dock.



This is us , returning Messi(our boat) to the dock with the remote control system

Square Trajectory

The second task was to make the boat follow a square trajectory. This required the boat to travel along straight segments and change its heading at the corners.

Compared with the first task, the square mission required more complex navigation because the boat had to follow multiple directions in a specific sequence. The control system needed to adjust the heading at each turn before continuing along the next side.

This task provided experience with sequential navigation commands and the coordination of straight-line movement and heading changes.

Navigation Race

The final task was a race in which the boat had to navigate to a specified point and then return.

This mission combined the main elements developed during the earlier tasks: heading estimation, directional control, and navigation between target points. Unlike the square trajectory, which required the boat to follow a predefined geometric path, the race focused on reaching a particular destination and returning.

We turned the GPS coordinates into metres with the formula from the board (the buoy was 166 m away). Our plan: full power, and aim 5 m behind the buoy so we don't slow down before the line. In simulation that was 136 s instead of 218 s.

The task tested the boat's ability to execute a navigation objective while following the required route.



Our boat resting after the hard race!

Trajectory Recording and Visualization

During the experiments, we collected the geographical coordinates recorded along the boat's course. We then used these coordinates to visualize the boat's trajectory on Google Maps. This allowed us to observe the actual path followed by the boat and compare its movement with the intended navigation route. The visualization provided a useful way to evaluate the boat's navigation behaviour during the different tasks.

Conclusion

During the project, our team worked on preparing the DD-Boat for navigation and implementing several increasingly complex tasks. We began with magnetometer calibration and a 20-metre northward journey, using a remote-control client/server system to recover the boat at the dock.

We then progressed to following a square trajectory and participating in a race that required reaching a specified point and returning. These tasks introduced different aspects of autonomous navigation, including sensor calibration, heading control, trajectory following, and movement between target points.

The project provided practical experience with the configuration and operation of the DD-Boat, as well as the software components needed to support autonomous and remote-controlled movement.