DDBoat Autonomous Navigation

Team name: Divas.
Members: Sofiia Manko (Ukraine), Mariia Sokolovska (Ukraine), Saulė Bardauskaitė (Lithuania).

Introduction

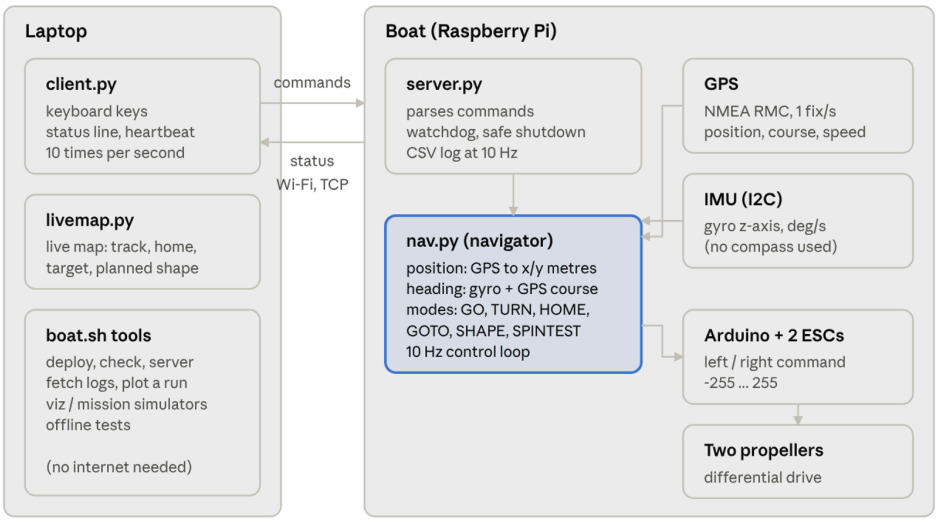
Our team made the DDBoat drive itself on the lake: it holds a course, turns by a given angle, returns to its start point, draws a square, triangle or circle, and crosses to a GPS point picked on Google Maps and comes back. All of this runs on the boat and works without a magnetic compass.

The DDBoat is a small twin-propeller catamaran. On board are a Raspberry Pi, an Arduino driving the two motor controllers (ESCs), a 9-axis IMU (gyroscope, accelerometer, magnetometer) and a GPS receiver. We started from the course driver repository (`drivers-ddboat-v2`) and built a navigation layer, safety features, field tools and a simulator on top of it.

This report explains the architecture (section 2), how the boat knows where it is and where it points (3), how it is controlled (4), how we kept it safe and tunable on the water without internet (5), how we tested everything in simulation first (6), and what happened on the water (7).

System architecture

The boat runs its own navigation; the laptop is only a remote control and a display. If the Wi-Fi link drops, the boat still knows what to do.



The laptop and the boat talk over the boat's Wi-Fi with a simple line-based text protocol on TCP port 5000. The laptop sends short commands (`M 50 50` for manual motors, `GO 80`, `TURN 90`, `HOME`, `GOTO lat lon`, `SHAPE square 20`, `S` to stop) and receives a status line twice per second. Every on-board component degrades gracefully: without an IMU or GPS the server still accepts manual driving.

The navigation logic (`nav.py`) is pure Python with no hardware access. The same code runs on the boat, in the simulator and in the automated tests, so what we tested on the laptop is exactly what ran on the water.

Localization and heading without a compass

The boat needs two things: where it is (GPS) and where its bow points (gyroscope, corrected by GPS). We chose not to use the magnetometer, because it needs a careful calibration for each boat and is disturbed by the motor currents, while the gyro and GPS were available and reliable.

Position

GPS gives latitude ly and longitude lx. We convert them to a flat local frame in metres (x = east, y = north) around a reference point m, the first GPS fix, with the formulas from the course (ρ0 = Earth radius, 6 371 km):

$$x = \rho_0 \cos(l_y) \sin(l_x - l_x^m), \qquad y = \rho_0 \sin(l_y - l_y^m)$$

The inverse is exact, so a target picked on Google Maps as (lat, lon) is converted with the same formula. Distance and bearing to a target follow directly: d = √(Δx² + Δy²) and bearing = atan2(Δx, Δy), measured clockwise from north. The home point is the average of the first 20 GPS fixes while the boat sits still, which removes most of the GPS noise.

Heading

The gyroscope's z axis gives the turn rate. Integrating it gives a precise but relative heading that slowly drifts. Two corrections make it usable:

- Bias: at start-up the boat stays still for 2 s and the mean gyro reading is subtracted (rejected if the noise is above 2 °/s, meaning the boat was moving). The bias keeps being tracked slowly while the boat is idle.
- Absolute direction: when the boat moves faster than 0.5 m/s and is not turning, the GPS course over ground tells where it really goes. The difference between this course and the integrated gyro heading is filtered (gain 0.3) into an offset.

A useful side effect: because the offset follows the course over ground, the boat steers its real path over the water. In a crosswind it automatically points its bow slightly into the wind (section 6 shows this).

Control and autonomous behaviours

Every behaviour reduces to one question asked 10 times per second: which heading do I want now? A single heading controller then turns that into two motor commands.

Heading controller

With e = desired heading − estimated heading (wrapped to ±180°) and a forward throttle u:

$$s = \mathrm{sat}_{\pm 60}(k_p\, e), \qquad u_{\mathrm{left}} = u + s, \qquad u_{\mathrm{right}} = u - s, \qquad k_p = 1.2$$

If |e| > 60° the boat first spins on the spot (one propeller forward, one backward) until it faces the target. The spin power is bounded between spin_min and spin_max, which the boat measures itself (section 5). Every command is capped at 120 of 255.

Key	Mode	Desired heading	Ends when
G	GO	heading at the moment of the key press	stopped, or after 5 min
J / L / U	TURN	current heading ± 90° (U: 180°)	within 6° for 0.6 s
H	HOME	bearing to the home point	mean of last 3 GPS fixes within 4 m
T	GOTO	bearing to a lat/lon target, then to home	same, at the target and then at home
1 / 2 / 3	SHAPE	along the current side of a square, triangle or circle	back at the start corner

Going to a point (HOME, GOTO)

The throttle shrinks near the target and when the heading error is large: u = min(80, 25 + 8d) · max(0.3, cos e). If the absolute heading is still unknown (the boat has not moved yet), it first drives straight for a few metres until the GPS course gives it one.

Drawing shapes against the wind

A shape is a list of corners defined in the boat's own frame: the first side goes where the bow points, all turns are to the right, and a circle is a polygon of sides about 3 m long. Steering straight to the next corner would let the wind push the boat off the side. Instead the boat follows the line between corners (line-of-sight guidance with an integral term), where xte is the signed distance to the line:

$$\psi_{\text{desired}} = \psi_{\text{line}} - \arctan\frac{xte}{L} - k_i \int xte\ dt, \qquad L = 6\ \text{m},\ k_i = 0.4$$

The arctan term pulls the boat back to the line; the integral learns a steady drift and cancels it. The next side starts 1 m before a corner. To stop GPS noise from skipping short sides, the position is averaged over the last three fixes and the boat may advance by at most one side per GPS fix.

Safety, calibration and simulation

At the lake the only network was the boat's Wi-Fi, so everything had to work offline and be safe if that link dropped.

Safety

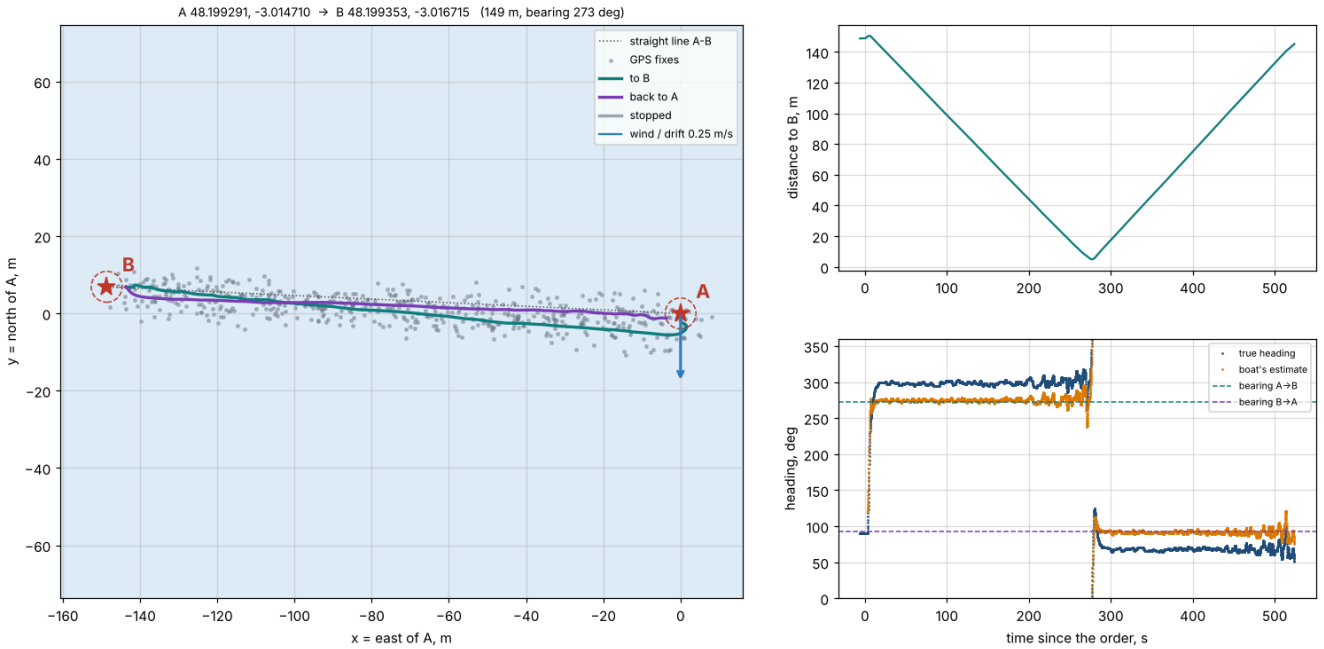
Manual driving, GO and turns stop by themselves after 1 s without a message from the laptop. HOME, the crossing and the shapes keep going without the link, because they end at a known point, but they stop if the GPS is lost or after a time limit. SPACE stops everything, and the server sets both motors to 0 if its terminal is closed.

Self-calibration

Instead of guessing constants, the boat measures itself: Z measures the gyro bias with the boat still, K the gyro scale while we turn the boat by hand exactly once, and Y the minimum propeller power needed to turn on the spot (it also detects swapped propellers). Results are saved on the boat. One script, `boat.sh`, runs every step from the laptop: copy the code, check each propeller, the IMU and the GPS, start the boat, and bring back the logs.

Simulation first

A simulated boat with gyro bias, GPS noise, a motor dead zone and wind ran exactly the same navigation code, and 59 automated checks run in seconds. For the crossing we picked A (48.199291, −3.014710) on our side and B (48.199353, −3.016715) on Google Maps: 149 m almost due west. With a 0.25 m/s crosswind and 3 m GPS noise the simulated boat reached B in 4.6 min, within 5 m, and came back to A within 4 m.



Simulated crossing A - B - A with a crosswind to the south. Left: the track (green to B, purple back) and GPS fixes in grey. Bottom right: the bow (dark blue) points about 25° into the wind while the course over the water (orange) stays on the bearing to B.

Results on the water

On the lake every behaviour worked on the real boat: manual driving, holding a course, turning by an angle, returning home, drawing the three shapes and the autonomous crossing to the GPS point on the other side and back.

At start-up the server found the Arduino and the IMU (I2C addresses 0x1e and 0x6b), calibrated the gyro (bias −3.36 °/s, noise 0.63 °/s) and received a GPS fix every second. The bias is large enough that an uncorrected gyro would drift by about 200° per minute, which shows why the start-up calibration matters.

Behaviour	Key	Worked on the water
Manual driving	W/A/S/D	yes
Hold course, turns	G, J/L, U	yes
Return home	H	yes
Square / triangle / circle, 20 m	1 / 2 / 3	yes
Crossing A - B - A, 149 m	T	yes







Difficulties, lessons learned and conclusion

Most of our problems were not in the control theory but in the practical chain between the laptop and the propellers.

The hardest part was the the chain between the laptop and the propellers. At the lake there was no internet, so we built tools to calibrate and tune the boat on site. Early on the propellers did not spin, which led us to write a pre-flight check that tests each propeller, the IMU and the GPS separately. We also often mixed up commands meant for the laptop and for the boat, so one script now runs everything from the laptop. Simulation caught several bugs before the water, for example noisy GPS making the boat skip the short sides of the circle.

Simulating first saved us time on the water: most bugs above were found on the laptop. Letting the boat measure its own constants (gyro bias, gyro scale, spin power) worked better than guessing them. Safety needs several independent layers, because each failure mode (lost link, closed terminal, lost GPS, refused order) is different.

Without a compass, by combining a calibrated gyroscope with the GPS course, the DDBoat holds a course, turns, returns home, draws shapes against the wind and crosses the lake to a point chosen on a map. Possible next steps: a calibrated magnetometer to know the heading before the boat moves, line following also for the crossing, and avoiding the shore and obstacles.