Computing control of an autonomous boat

A second year intership in Aston university

The boat

Noé Lemaistre
Student Engineer
FISE 2026
noe.lemaistre@ensta.fr

2025

Abstract

This report presents the development and testing of an autonomous sailboat. The project focused on the implementation of modular control software where each sensor and actuator is encapsulated within a dedicated class, and higher-level navigation algorithms are managed by a central navigation module. Several control strategies were implemented, including a PID-based heading controller, a waypoint-reaching algorithm based on GPS data, a line-following strategy to minimize cross-track error, and an automatic tacking procedure to handle upwind sailing. A real-time data logging system was also developed using an SD card to record sensor values and control commands for later analysis. Experimental trials were conducted on a pond, where the boat successfully executed a mission consisting of multiple waypoints under realistic wind conditions. Results highlight both the robustness of the control architecture and the challenges posed by sensor noise and environmental disturbances. This work demonstrates the feasibility of affordable autonomous sailing platforms for research and provides a foundation for future improvements such as advanced sail optimization and long-term navigation.

Ce rapport présente le développement et les essais d'un voilier autonome. Le projet s'est concentré sur la mise en place d'un logiciel de contrôle modulaire, où chaque capteur et actionneur est encapsulé dans une classe dédiée, tandis que les algorithmes de navigation de plus haut niveau sont gérés par un module central de navigation. Plusieurs stratégies de contrôle ont été implémentées, notamment un régulateur PID pour le suivi de cap, un algorithme de suivi de points GPS, une stratégie de suivi de ligne pour réduire l'erreur latérale ainsi qu'une procédure automatique de virement de bord pour gérer la navigation face au vent. Un système d'enregistrement en temps réel a également été développé grâce à une carte SD, permettant de conserver les données capteurs et les commandes pour analyse ultérieure. Des essais expérimentaux ont été menés sur un plan d'eau, où le voilier a réussi à exécuter une mission composée de plusieurs points de passage dans des conditions de vent réalistes. Les résultats mettent en évidence la robustesse de l'architecture de contrôle ainsi que les défis posés par le bruit des capteurs et les perturbations environnementales. Ce travail démontre la faisabilité de plateformes de voile autonome à coût réduit pour la recherche et constitue une base solide pour de futures améliorations telles que l'optimisation de la voile et la navigation de longue durée.

Acknowledgments

Je tiens à remercier le docteur Jian Wan qui m'a accueilli et guidé à travers ce stage à l'université d'Aston

Contents

1 Introduction

Sailing technology has been mastered by humanity since ancient times. The earliest known representation of a sailboat dates back to around 5000 BCE, carved on the walls of a temple in Luxor. Throughout history, sailboats have played a decisive role in the development of civilizations, enabling trade, exploration, and warfare across vast distances. However, the advent of the steam engine in the 19th century gradually eclipsed wind power, relegating sailing to a primarily recreational or sporting activity.

Today, in the context of global warming and the urgent need to reduce carbon emissions, sailing has experienced a significant revival. Modern projects seek to harness wind power not only for leisure but also for sustainable trade and scientific research [1]. In this perspective, autonomous sailing vessels represent a particularly promising field of innovation. They combine the age-old knowledge of wind navigation with modern electronics, sensors, and algorithms to create environmentally friendly platforms capable of long-term missions without human presence on board.

This technology has great potential for oceanography, where large parts of the world's oceans remain unmapped and poorly monitored. By deploying autonomous sailboats, researchers can collect valuable data over long periods at low cost and with minimal environmental impact. Several companies and research groups are already exploring this path—for instance, the American company *Saildrone* has developed fleets of autonomous sailboats for defense, climate studies, and oceanographic missions.

2 System Overview

The boat is designed as a single-hull vessel equipped with a single sail. The sail is primarily responsible for controlling the propulsion and therefore the speed of the boat, while the rudder governs its heading and overall maneuverability. Both of these critical components—the sail and the rudder—are actuated by independent servo motors. This configuration allows precise and automated control of the boat's dynamics, enabling the system to adapt to changing wind conditions and maintain the desired course with minimal human intervention.

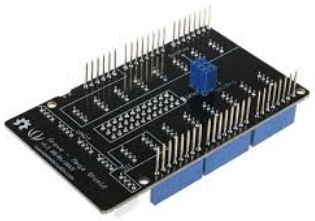
3 Materials

3.1 Hardware

The hardware platform was built around an **Arduino Mega 2560** [2] microcontroller, chosen for its large number of digital pins, multiple UARTs, I2C, and SPI ports, which are essential for interfacing with multiple sensors simultaneously. The main hardware components are described below:

(a) Mega 2560

(b) PCA Board



(c) Shield

Figure 1: Hardware components

- **GPS module** (Adafruit GPS on Serial2) for geolocation and UTC time retrieval.

- **CMPS12 IMU** for heading measurement, including calibration routines to minimize magnetic disturbances.

- **Custom wind sensor** for wind speed and direction, with interrupts used for pulse counting.

- **Servo motors** driven via a PCA9685 board for rudder and sail control.

- **RC receiver** for manual override and safety fallback.

- **SD card module** connected over SPI for persistent mission data logging.

3.2 Box

A dedicated enclosure has been designed to house all the different electronic components of the system. The box not only protects the hardware from external conditions such as humidity, shocks, or dust, but it also provides a clean and organized layout that makes it easier to identify and troubleshoot potential issues. By grouping all modules in a single accessible structure, maintenance and debugging become more efficient, while ensuring the overall reliability of the sailboat. The box has been designed by my partner Ewen Mélé.

4 Software

4.1 general description

The software was written in C++ within the Arduino ecosystem. A class-based approach was adopted: each sensor or actuator is represented by a class (e.g., GPS, IMU, WindSensor, ControlMotor, Controller, SDcard). The Navigation class integrates them to implement higher-level behaviors. Logging functions write mission data to an SD card in plain text, which can later be converted into GeoJSON for trajectory visualization.

The autonomous sailboat system was designed with a strong emphasis on modularity. Each physical sensor and actuator is encapsulated within its own dedicated software class, while a central Navigation class is responsible for coordinating the data flow and executing the control algorithms. This architecture ensures a clear separation of concerns, which greatly simplifies debugging, facilitates the replacement or upgrading of individual components, and enhances scalability for future extensions.

The system operates in two distinct modes: manual control, where the boat is remotely piloted, and autonomous navigation, where decision-making is fully handled by the onboard algorithms. All class instances are created in the main file, which also contains dedicated functions to display sensor data via the RX/TX serial terminal and to log mission data onto the SD card. In practice, raw data from the sensors is continuously acquired within the main loop and then passed to the Navigation class, where it is processed and transformed into control commands for the actuators.

4.2 Classes

4.2.1 controlMotor class

The `controlMotor` class is responsible for driving the sail and rudder servomotors. It relies on the `Adafruit PWMServoDriver` library to generate PWM signals. The PWM duty cycle corresponds to an angle command, and is computed using the following formula:

$$PWM = (PWM_{\max} - PWM_{\min}) \cdot x + PWM_{\min}$$

where:

- x is a float in the range $[0.0, 1.0]$, representing the normalized control command.
- $PWM_{\min}$ and $PWM_{\max}$ are the calibration limits of the servomotor.

This abstraction allows the navigation algorithms to set rudder and sail angles in a normalized way, without directly handling PWM values.

4.2.2 IMU class

The `IMU` class interfaces with the CMPS12 sensor, which provides heading measurements. Communication is handled through a serial interface. The class supports two main features:

- **Heading acquisition:** The IMU sends a 16-bit value representing the absolute heading. This is converted into degrees in the range $[-180°, +180°]$.
- **Self-calibration:** At startup, the IMU executes an internal calibration routine, ensuring that heading measurements remain accurate despite sensor drift or environmental disturbances.

The class exposes an `update()` function to refresh sensor values, and `get_cap()` to retrieve the current heading.

4.2.3 GPS / GPS2 classes

The GPS classes provide geographic positioning. Two versions exist:

- `GPS`: a minimal implementation that parses raw NMEA sentences to extract latitude and longitude.
- `GPS2`: a higher-level version based on the Adafruit GPS library, providing robust parsing and additional information such as the number of satellites and fix validity.

Both classes implement:

- `getLatitude()`, `getLongitude()`, `getPoint()` → return coordinates.
- `conversion()` → converts geographic coordinates into local Cartesian coordinates for navigation.

4.2.4 WindSensor class

The `WindSensor` class measures two key variables:

- **Wind speed:** counted by measuring pulses from an anemometer, attached via an interrupt. The number of pulses per unit time is converted into speed (in m/s).
- **Wind direction:** read from an analog sensor (wind vane). The voltage is converted into an angular direction in degrees.

Filtering methods (e.g., exponential moving average) can be applied to smooth noisy data caused by boat motion.

4.2.5 Controler class

The `Controler` class interfaces with a remote controller (manual mode). It reads PWM signals from RC channels (rudder, sail, mode selection) using `pulseIn()`.

Key features:

- Manual sail and rudder control: maps RC inputs to servo angles.
- Mode switching: a dedicated RC channel toggles between manual and autonomous modes after a short delay (safety mechanism).

4.2.6 Navigation class

The `Navigation` class is the core of the autonomous logic. It integrates data from the GPS, IMU, and WindSensor, and computes the commands to be applied to the actuators via the `controlMotor`.

Main algorithms:

- **Heading control (follow_cap):** Proportional-Derivative (PD) controller that minimizes heading error.
- **Reach point:** computes bearing between current and target GPS points, then calls `follow_cap()`.
- **Line following:** projects the boat's position onto a virtual line between two waypoints, and computes lateral error to correct the trajectory.
- **Tacking:** manages zig-zag sailing when the target is upwind.

The class is designed to remain modular and easily extensible.

4.2.7 SDcard class

The `SDcard` class is dedicated to data logging. It uses the Arduino SD library to write mission data into a `log.txt` file. Each entry contains:

- Timestamp (from `millis()`)
- GPS coordinates
- Heading (from IMU)
- Wind speed and direction
- Rudder and sail positions
- Current control mode (manual/autonomous)

The resulting file can later be converted into `GeoJSON` format for visualization on platforms like `geojson.io`.

4.3 Navigation Algorithms

Autonomous sailing requires algorithms that transform sensor inputs into rudder and sail commands. Several algorithms were implemented:

- **Heading following:** This is the fundamental control algorithm of the sailboat. It is implemented as a Proportional–Derivative (PD) controller acting on the rudder angle. The algorithm takes a desired heading (in degrees) as input and continuously compares it with the current heading measured by the IMU (CMPS12). The heading error is computed and corrected through the rudder, with proportional action ensuring responsiveness and derivative action damping oscillations. This allows the sailboat to maintain a stable course despite disturbances from wind gusts or water currents.

- **Waypoint reaching:** This higher-level navigation algorithm builds on heading following. It computes the bearing (azimuth) from the current GPS position to a target waypoint using geodesic formulas. The bearing then serves as the input to the heading following controller. In other words, waypoint reaching translates a positional objective into a heading command, allowing the boat to autonomously navigate toward a GPS-defined location.

- **Line following:** Line following is designed to guide the boat along a virtual straight path defined by two GPS waypoints. Instead of simply heading toward the next waypoint, the algorithm projects the current boat position onto the desired line and computes the cross-track error (the lateral distance to the line). The control law then corrects this error by adjusting the commanded heading so that the boat converges toward and tracks the line, which is particularly useful for survey missions or systematic transects.

- **Tacking:** When the wind comes from a direction close to the desired heading, the sailboat cannot sail directly into it. In such cases, the tacking procedure is activated. The boat alternates its heading relative to the wind (typically at about 45° on each side), following a zig-zag pattern until progress toward the target waypoint becomes feasible again. In this project, tacking was implemented as a timed maneuver: when the wind was detected within a forbidden "no-go zone," the algorithm adjusted the heading for a fixed duration before resuming waypoint reaching. If the boat remained blocked by the wind, the tack would automatically switch to the opposite side.

- **Sail trimming:** The sail angle is controlled independently of the rudder through a simple linear mapping between the apparent wind direction (measured by the wind vane) and the sail servo angle. The aim is to keep the sail just on the verge of luffing, ensuring that air always flows across it for maximum efficiency. While the present implementation relies on a static mapping, more advanced strategies could incorporate dynamic adjustment based on wind speed and course stability.

5 Results

The purpose of the test was to evaluate the reach point algorithm by making the autonomous sailboat complete a full loop around the pond. In order to structure the naviga-

tion, three checkpoints were defined along the course, which the boat had to pass through sequentially before returning to its initial position. This setup allowed us to assess the precision of the algorithm in guiding the vessel toward intermediate targets rather than simply following a straight line. During the trial, the wind was blowing steadily from the southwest and the boat had to continuously adjust its rudder and sail settings to maintain an efficient trajectory while still heading toward the checkpoints.

Figure 2: Path of the Boat

My colleague Ewen implemented the algorithm described in the article by Jaulin and Lebars [3], which is specifically designed for autonomous sailboat navigation. In parallel, I developed a simpler yet effective control strategy based on a PD controller applied to the rudder. The sail, on the other hand, was adjusted dynamically according to the wind direction, with the goal of always keeping airflow in the sail to generate propulsion.

The real challenge lay in the tacking maneuver. When the wind was coming directly from the bow, the boat was unable to advance efficiently, so it automatically triggered a tacking procedure. During this phase, the boat altered its heading to avoid sailing directly into the wind. After maintaining this new heading for about ten seconds, the system would command the boat to resume its course toward the target waypoint. If the vessel was still facing the wind after resuming course, it would initiate another tack in the opposite direction, alternating sides until it could effectively continue its trajectory toward the objective. On figure 4, the three point are the point reached by the boat. The circle around represent a 5 meter interval within the boat can consider he reached the point.

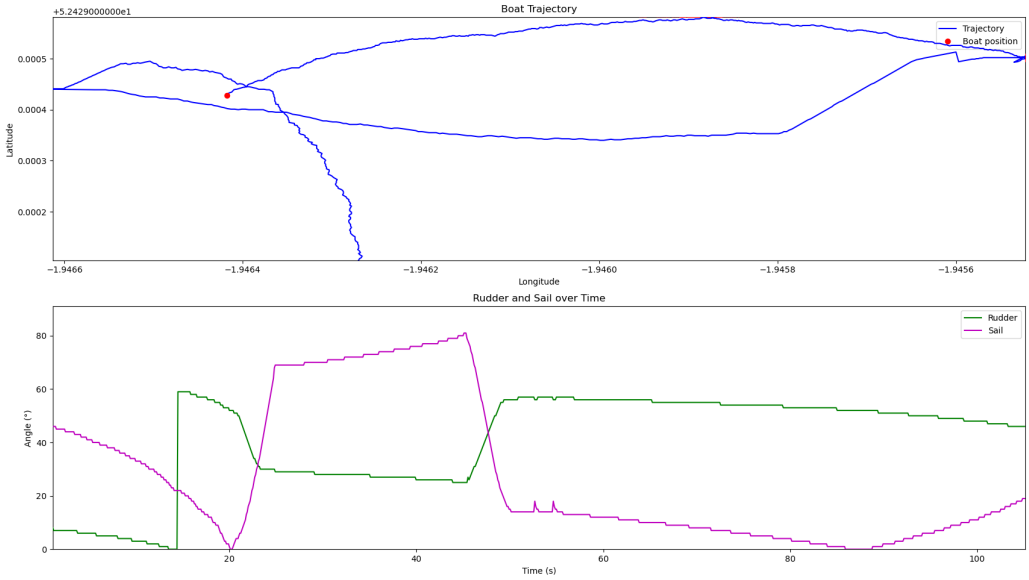


Figure 3: Position of the boat (top) and sail and rudder position

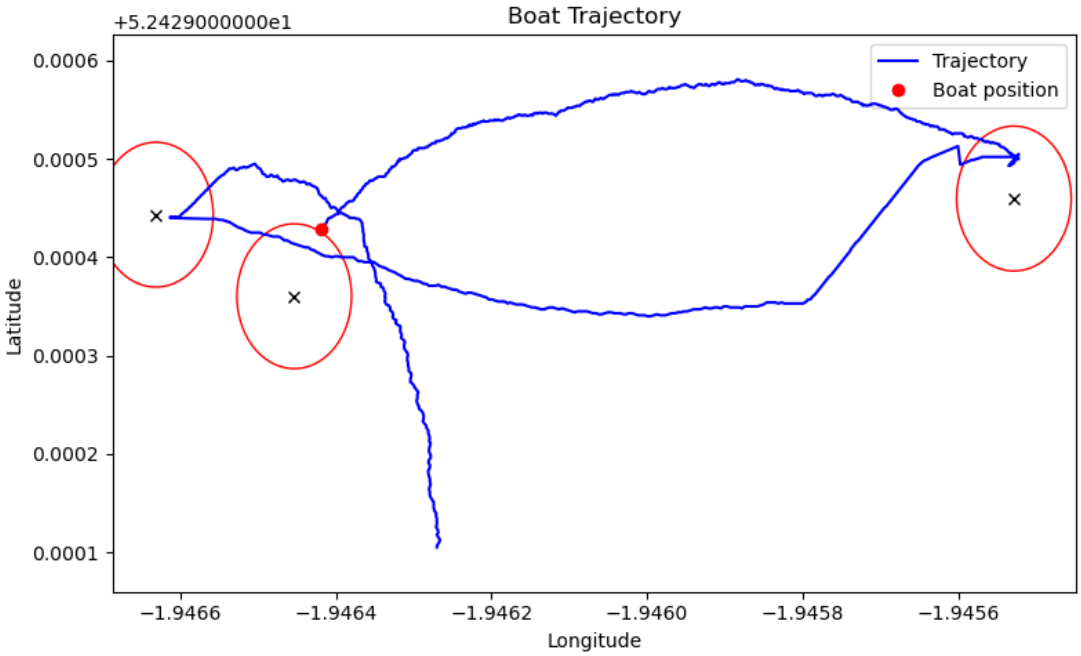


Figure 4: Position of the boat with the three reached point

6 Conclusion

The development of the autonomous sailboat represented a multidisciplinary challenge, combining electronics, embedded programming, control theory, and marine dynamics. Throughout this project, a modular software architecture was designed and implemented to ensure flexibility, scalability, and ease of debugging. Each physical component—IMU, GPS, wind sensor, servomotors, and SD logging system—was encapsulated in its own class, allowing clear communication and independence between modules.

The navigation system was structured hierarchically, from low-level control loops (rudder and sail) to higher-level decision-making algorithms such as heading following, waypoint reaching, and tacking. Field tests demonstrated that the sailboat was capable of following simple trajectories under real wind conditions, validating the robustness of the implemented algorithms. However, the experiments also highlighted limitations, such as sensor noise, calibration instability, and the difficulty of tuning PID gains in a dynamic marine environment.

Despite these challenges, the project successfully achieved its main objective: designing and testing a functional prototype capable of autonomous navigation. The results provide a solid foundation for future improvements, including the integration of advanced wind estimation techniques, adaptive control for better sail efficiency, and possibly the use of machine learning for decision-making under variable conditions. In a broader sense, this work contributes to the growing field of sustainable maritime robotics and demonstrates the potential of wind-powered autonomous systems for research, exploration, and environmental observation.

Bibliography

References

[1] Exemple of industrial use of autonomous sailboat *Autonomous Ocean Data Collection Platforms*, 2023. Available at: `https://www.saildrone.com/`

[2] Arduino, *Arduino Mega 2560 Rev3 Technical Reference*, 2024. Available at: `https://docs.arduino.cc/hardware/mega-2560/`

[3] Jaulin L and Lebars F, *A Simple Controller for Line Following of Sailboats* Available at: `https://link.springer.com/chapter/10.1007/978-3-642-33084-1_11`

[4] Melin, J, *Modeling, control and state-estimation for an autonomous sailboat (Dissertation)* Available at: `https://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-261553`

[5] Link to the porject's git : `https://github.com/BZprogrammeur/Ewensboat.git`

Annex

Class diagram

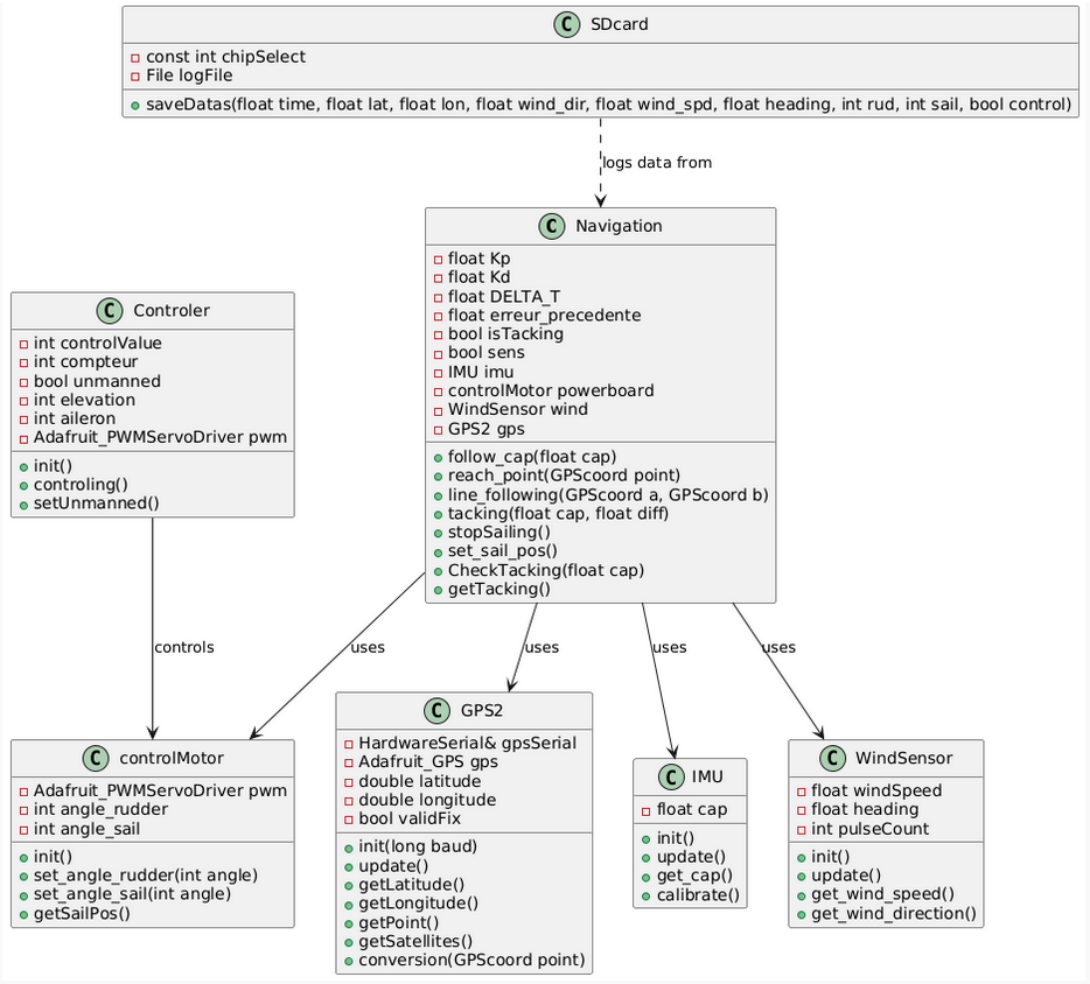


Figure 5: Class diagram

How to launch ?

Begin by assembling the sailboat's mechanical components. Install the rudder securely in its mount and attach the sail to the mast, ensuring that the control lines are properly connected to their respective servomotors. Verify that all mechanical elements move freely and without friction.

Next, connect the Arduino board to the computer via USB. Open the project folder and upload the program located in: Arduino Code/Allsensor/main/main.ino You can also access the latest version of the source code directly from the GitHub repository: https://github.com/BZprogrammeur/Ewensboat.git

Here is scheme to help pluging

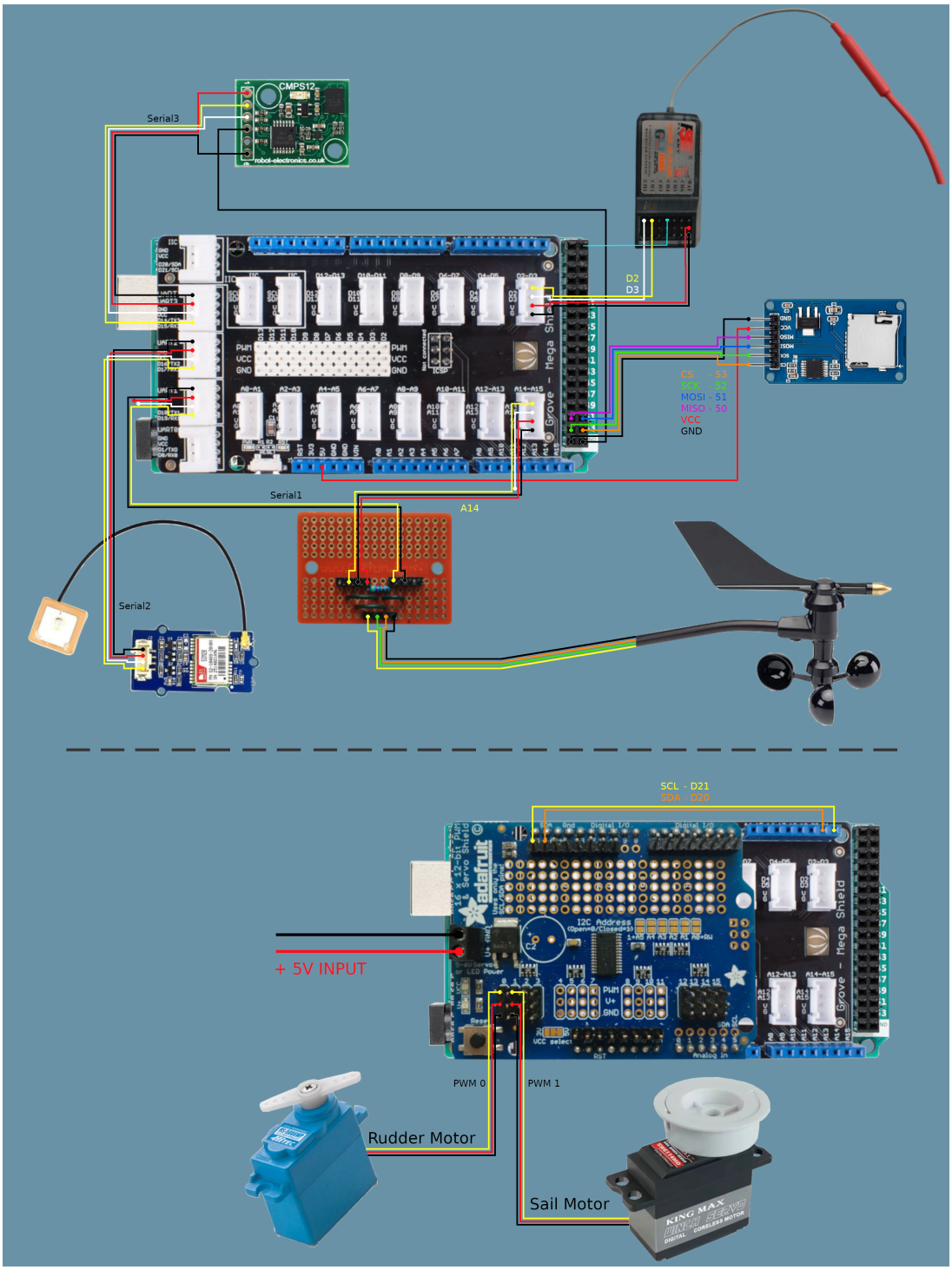


Figure 6: Pluging scheme