# Development and Test of an Open Source Autonomous Sailing Robot with Accessibility, Generality and Extendability

Ulysse Vautier
ulysse.vautier@plymouth.ac.uk

Jian Wan
jian.wan@plymouth.ac.uk

Ming Dai
y.dai@plymouth.ac.uk

Christophe Viel
christophe.viel@plymouth.ac.uk

Robert Hone
robert.hone@plymouth.ac.uk

School of Engineering, University of Plymouth, Plymouth

**Abstract**

This paper introduces the design of an open source autonomous sailing robot with emphasis on its accessibility, generality and extendability. To meet such requirements, a generic control box connecting an Arduino board and a Raspberry Pi computer was tailor-made so as to host the robot operating system (ROS) and to interact with versatile sensors and actuators. The goal of such a project is to create an accessible, generic and expendable platform of autonomous sailboats for wider education and research publicity and engagement. Autonomous sailing test for the developed sailboat was also conducted to validate its design.

## 1 Introduction

Over the last two decades, there are growing interests in developing autonomous sailing robots. Different groups have already made fully autonomous sailboats, ready to deploy for data gathering purposes and almost able to cross the transatlantic ocean in the Microtransat challenge (Meinig et al., 2015; Ghani and Hole, 2014). Sailboat research projects are also growing, most of which are particularly structured for specific problems (Naveau et al., 2013; Rathour et al., 2017; Silva Junior et al., 2016; O'Hara, 2017) or others for generic research (Domínguez-Brito et al., 2015; Miller et al., 2014; Wirz et al., 2015; Plumet et al., 2015). There are also various kinds of software, hardware or mechanical design projects (Santana-Jorge et al., 2017; Neal, 2006; Lam et al., 2016). While the motives of building such robots are multiple with no wide-spread applications for autonomous sailing technologies, the research in this field is still in its early stage compared to other autonomous vehicles such as cars or drones.

An autonomous sailboat is interesting in that it can use all available energy around it to complete its mission i.e. wind, solar and wave energy (Liu et al., 2016). This makes it an important subject for long-duration applications such as oceanography, surveillance and reconnaissance. Moreover, such robots can play a significant role in solving nowadays environmental and ecological problems. The energy needed to operate a sailboat is relatively small in comparison to other autonomous vehicles (Cruz and Alves, 2008). Nevertheless, the market

In: S. M. Schillai, N. Townsend (eds.): Proceedings of the International Robotic Sailing Conference 2018, Southampton, United Kingdom, 31-08-2018

and the commercial applications are yet to be explored to bring greater interest to it. In that context, there is a need to develop a modular sailing robot platform with accessibility, generality and expandability so as to facilitate the development of such robots with more focus on their control and applications.

Most projects of autonomous sailboats are open source. While most of them explain the hardware specifications and some explain the software designs, it is still hard for a researcher to make their own sailboat in a short amount of time and to focus on control algorithms or applications. This asks for an accessible hardware, already available in most research labs and more importantly a friendly software architecture which makes it straightforward for any user to test control algorithms.

In the same way 3D printers are designed, fully open-sourced and with accessible hardware, a control box was made for highly generic hardware and software structures. That way, the control box can be put in any RC (Radio-Controlled) sailboat hull, connect to sensors and servo-motors and transform it into an autonomous sailboat. The software architecture has to be flexible so that in couples of lines of code, the sailboat can respond and act according to the instructions. So this paper will address the development of an accessible, generic and expendable autonomous sailboat with particular emphasis on the aspect of control hardware and software, which is in the format of a control box.

## 2 Accessible Hardware Configuration

The hardware architecture is similar to other sailboats. To stay low on budget and to use reliable resources, most projects use Arduino boards and Raspberry Pis, which can be easily accessed by most robotics laboratories (Krauss, 2016; Lazarin and Pantoja, 2015; Miller et al., 2014). The hardware structure is also mostly the same (Lam et al., 2016; Plumet et al., 2015), it consists of a low-level computer and a high-level computer. In this case, the Arduino acts as a low-level controller acquiring data from sensors and acting on the actuators and the Raspberry Pi acts as a high-level controller with more intensive algorithms as shown in Figure 1. Making a specific purpose board can lower the price and simplify the architecture (Alvira et al., 2013; Cabrera-Gámez et al., 2013), but it would decrease the accessibility of such a hardware. Others would use a bigger computer instead of the Raspberry Pi to have more resources for the algorithms (Naveau et al., 2013; Lam et al., 2016) but this will increase the cost and reduce accessibility. The newest Raspberry Pis are good enough to process complex control algorithms (Benchoff, 2016; Larabel, 2018). It would also be possible to use the Raspberry Pi alone. Adding low-level microcontrollers provide an extra security and free valuable resources of the high-level computer for main control tasks.

Along with the technology development, the price for Arduino boards and Raspberry Pis will continue to decrease while their performance will continue to improve, which makes such a hardware architecture more attractive in the long term.

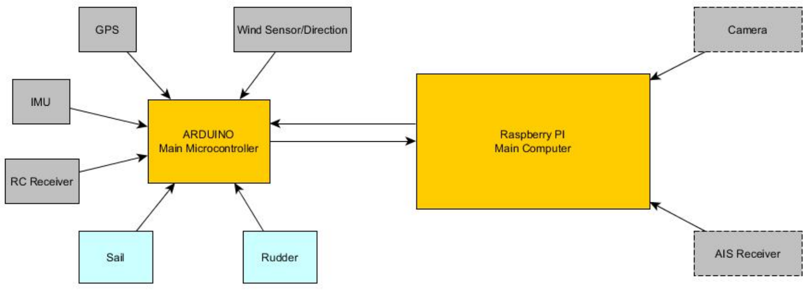

Figure 1: Hardware Architecture.

### 2.1 Specifications and Architecture

The control box was designed to comply with as many standards as possible. While the boards presented here comply with our configurations, changes can be made to accept other configuration of power necessities. The sailboats are powered with a 2S (Two cells) LiPo battery which outputs a maximum of 8.4V (4.2V per cells) and nominal at 7.4V (3.7V per cells). We set our requirements to at least two hours of continuous operation without any other energy input such as solar panels. From all the sensors and actuators the sailboats have, we are at about 2A usage at full use. This means we must have at least a 4000mAh battery. Of course, this also depends on other factors such as the boat size, weight and usage. For standardization, it is considered that 3S (maximum 12.6V) or even 4S (maximum 16.8V) batteries can be used.

Particular sensors or actuators are not discussed in this paper. But based on RC models of sailboats and expertise of users, a standard as generic as possible was designed. As such, a 3 pin connector was considered for small servomotors connection. Other connectors were chosen arbitrarily for robustness while keeping standard pin configurations. Those configurations are to be discussed further. For a sailboat to be autonomous, it needs at least a way to know its position, heading and wind direction. We will use a GPS, IMU and a wind direction sensor in our examples. This control box is designed to be able to connect with other sensors. Concerning actuators, it has to control the sail and the rudder.

### 2.2 Custom Interface Board

While the platforms chosen are accessible, they are development platforms and are unfit for industrial applications. Pin headers are the connectors in both boards. They are not suitable for robust and waterproof connections. To solve this problem, a custom board is made to have better industry-grade connectors. This custom board decreases the accessibility for researchers but it highly improves the robustness of the control box. Making such a board can hinder the accessibility with some laboratories which do not have the facilities to make such a PCB board. We suggest the use of third-parties industries, accessible to anybody, to make those boards.

This custom board is connected to the Arduino and the Raspberry Pi. It manages the power and extracts all connectors from the Arduino to sensors and actuators. It is also designed to contain all the boards in a small sized box and be mechanically robust. The following sections will describe in detail the choices in the different parts of the boards. All the details and documents for DIY can be found in the *Meca* Repository of (Sailboat, 2018).

#### 2.2.1 Power Management

The power management is simple and efficient. The unit manages the energy supply for sensors, actuators and the Raspberry Pi. It is simply made of regulators for each group of loads. We use switching regulators because of its high efficiency. These regulators can input a large range of voltage (6.5-32V) and output a constant voltage (5V). They are used for the actuators and the Raspberry Pi. One linear regulator can manage the power supply to multiple sensors because of the low energy consumption of sensors. While these regulators were chosen based on our configuration, the footprints on the custom boards are standard, making it possible to solder any regulators.

Heat and energy is a problem for underwater and surface vehicle. Using linear regulators or regulators with low efficiency can hinder the boat capacity. Switching regulators are used with efficiency up to 96% to lower the heat generated. The Raspberry Pi is the biggest consumer of energy. In a linear regulator, the ballast lowers the energy by outputting heat. This would end up wasting $(8.4V - 5V) \times 2A = 6.8W$ of energy while a switching regulator would waste only $5V \times 2A \times (\frac{1}{0.96} - 1) = 0.42W$ (considering 96% efficiency) for the Raspberry Pi.

#### 2.2.2 Connectors and Communication

Connectors play a big part in robustness for robots. In mobile robots, vibration and movement can remove the cables from the pins. Robust, industry-grade connectors that are easy to plug in and out frequently was necessary for sensors. JST connectors were chosen purposely for the latter, based on usage experience and industry standards. Concerning the battery connections, two pin holes are placed on the custom board on which any wire connectors can be soldered. In our case, our battery had HXT 4mm connectors and as such HXT wire adapters were soldered, as shown in Figure 2. Finally, the custom boards also have a special compartment for radio receivers, for which a row of 3 pin holes is made available. Up to 6 channels are pluggable.

There are some standards in connection and communication protocols for most sensors and actuators. Most notable communication protocols range from Serial communication, I2C protocol to Analog signal and Digital interrupts. Each of these communications come with a standard wiring. Sensors have 2 wires for power - Ground and Power (5V) - and the rest for communication - RX, TX for Serial, SDA, SCL for I2C, or 1 pin for analog and digital signal. Of course, other protocols and wirings exist but those are the most common.

This custom board accepts all of those different connectors and communications and allows access to the corresponding Arduino pins with mechanical connectors. Once connected, the software part arranges the connection inside the system through a configuration file.

#### 2.2.3 Waterproofing

One of the challenges in sailboats is waterproofing all the electronic components. Several 3D printed boxes were made for the sensors and the control box, all available online at the *Meca* repository of (Sailboat, 2018). A key problem arising from waterproofing is heat. Any electronic components generate heat. An enclosed system in plastic or wood insulates it and keep the heat inside, which may cause the components to overheat. For low-power sensors such as GPS or IMUs, plastic is enough to dissipate the heat. But for heat-generating components such as the main computer, a high dissipating material is needed. For this case, a side of the box (close to the main computer) is metal.

The 3D printed case for the boards' stack is a frame-like box which leaves open the connectors. While the box is not fully waterproof, it is enough to protect the boards from the small amount of water coming inside the boat. Batteries were chosen to be ROAR approved, meaning they are approved for racing. These batteries are hard-cased and protected against high impact (Rules, 2013). The hard case in itself provides a weatherproof package for batteries. The hulls of the boat are sealed as much as possible to waterproof the whole sailboat.

### 2.3 Hardware

Figure 2 shows the board and its components. Two boards were made to form the overall structure. The overall board stack dimensions are 70x130x40mm. It was designed to be placed in our 1m-long boat[1]. Table 1 shows the overall components used and their specifications. A complete detailed BOM (Bill of Material) can be found in the *Meca* repository of (Sailboat, 2018). It is based on the high prices of each component, for instance, the Arduino Mega can be found at a lot cheaper cost. The cost of the custom boards is at £40.1 using a third-party service to print the circuit boards. The biggest cost comes from the switching regulators. The remaining cost of the whole sailboat would be on the hull, the sensors, actuators and battery.

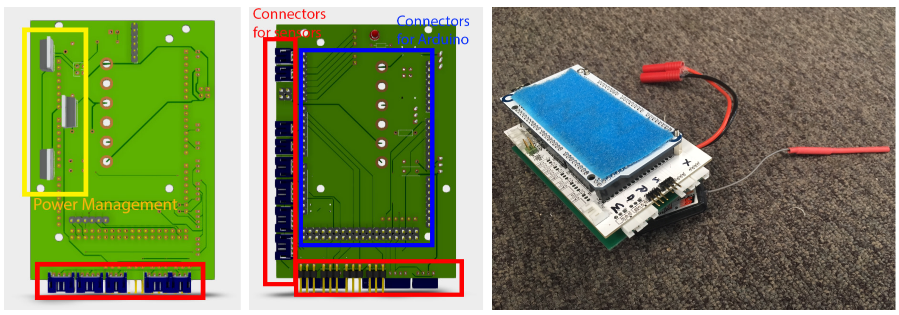

Figure 2: Back and front of the custom board and the assembled control box.

Table 1: Hardware Specifications and Approximate Price of the Control Box (without sensors and motors)

| Part | Description | Total Price (£) |
|---|---|---|
| Board Components | Resistors, Capacitors, Inductors, diodes and regulators | 19.1 |
| Connectors | Industrial Grade Connectors - Arduino and Raspberry Pi Connectors | 12 |
| Circuit Board | Third-Party Service from SeeedStudio | 8 |
| Screws and Spacers | Mechanical constraints - M3 | 1 |
| Arduino Mega | Low-level Microcontroller Board | 28 |
| Raspberry Pi 3 Model B | High-level SBC Computer | 32 |
| Total | | 100.1 |

## 3 Generic Software Architecture

Sailboats come in different formats, small or long, one or two rudders, winch motors for sails or direct actuation. As the hardware can change among different sailboats, the software architecture should be able to adapt to

[1]Length Over All

those varying configurations. In a similar way with 3D printers that built upon the marlin firmware (Marlin, ), a configuration file is used to configure the customized hardware specifications. Not only does the hardware configuration change but different brands and different protocols of sensors and actuators are used. This is solved by using a component-based architecture, highly used in the video game industry and robotics (Gregory, 2014; Elkady and Sobh, 2012; Santana-Jorge et al., 2017). It simplifies the development of libraries of different components while maintaining uniformed standard and compatibility with the overall system.

Concerning the Raspberry Pi, no configuration is needed. The focus is on the usability and interfaces for users to add their own controllers. **ROS** (Robot Operating System) has been increasingly used for robotics research and development (Cousins, 2011). Using ROS as the base of the software architecture, the project can benefit from existing work done by other researchers in robotics and be able to share its specific outcomes with the community as well. With **ROS**, a software architecture has been done for users to easily and rapidly integrate their controllers into the robot.

## 3.1 Arduino Programming

The Arduino is the low-level controller. It acquires all the data from different sensors and acts upon the actuators. Once all the data is updated, it sends them to the main computer and also receives the instructions from it. The code and the wiki for installation instructions can be found in *Arduino Interface* repository of (Sailboat, 2018).

A Component-Based architecture is exploited to be as generic as possible with regards to the hardware. Such architecture fully uses the **OOP** (Oriented-Object Programming) aspect of C++, the language used for Arduino. The principle is based on the semantics of the architecture. We have a main class, the *Sailboat*, which contains component objects : *Sensors*, *Actuators* and *Basic Controllers*. Those components have their own methods to conduct certain tasks. *Sensors* have the common methods of acquiring data from sensors and sending them to the main computer. *Actuators* have the common method of acting upon the actuators to the desired set-point. Finally, *Basic Controllers* have the common methods of controlling, taking the data from sensors and acting upon the actuators. The implementation of those methods might differ among different products but the inputs and outputs are the same. Using **OOP**, you can derive from these components class and would only need to implement the methods. The *Sailboat* class would automatically call the right methods depending on the derived component class. This architecture makes it simple for any user, using a new sensor or actuator, to implement the methods and stay compatible with the sailboat hardware configuration.

The *Basic Controllers* are low-level controllers such as rudder control, sail control or heading control. This enables the main computer to send different inputs to the Arduino, depending on the controller, e.g. A Potential Field controller might output a heading while a Line Follower controller might directly act upon the rudder without adjusting the sail.

On top of those basic controllers are security controllers such as the RC controller and the Return Home controller. They will be discussed further in section 3.3 on safety.

Figure 3 shows a UML (Unified Modeling Language) of the different classes in the Arduino architecture. For the optimal sail angle and the rudder angle, we are constructing our code on (Jaulin and Le Bars, 2012) for our basic controllers. You can see the implementation of all the methods at *Arduino Interface* repository of (Sailboat, 2018).

## 3.2 Raspberry Pi Programming and ROS

The Raspberry Pi is the main computer in this architecture. It is loaded with a lightweight *Ubuntu* Operating System. Its goal is to calculate the complex control algorithms using the data sent by the Arduino and act on the actuators by commanding the Arduino. In both the Arduino and the Raspberry Pi, the architecture is made so that any users will be able to focus on the algorithm and not the integration of it. The architecture also uses the **OOP**. **ROS** is compatible with Python and C++ languages and the same architecture is used for both languages. The code related to the Raspberry Pi can be found on *ROS* repository of (Sailboat, 2018).

This architecture is different from the Arduino software architecture in that it focuses on usability and ergonomics for users. The goal of such an architecture is to free the user from integration problems and focus only on the important part of the research, which is the control of the sailboat.

For that reason, the architecture automatically takes over communication tasks with the Arduino, acquiring and parsing the data from it and making it available to the user. All the user needs to do is to implement two methods : a setup method to initialize all the variables and a control method where the algorithm has to be implemented, which sends the results to the Arduino.

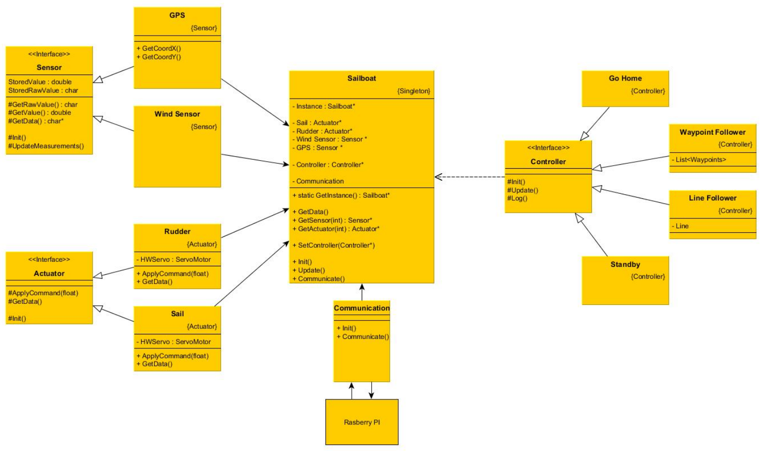

Figure 3: UML of the Component-Based Architecture.

The newest Raspberry Pis come with a WiFi chip. This enables WiFi communication with another computer without cables. When it boots up, it creates a hotspot, it is then possible to connect to it, SSH (Secure Shell) into the Raspberry Pi and controls it directly from a laptop.

**ROS** is used for multiple reasons. It is first highly used by the robotics community around the world and some users might consider it standard. It also helps in making robust protocol communication with the Arduino and among different components of the sailboat. Finally, it already has various libraries/applications (called *nodes* in **ROS**) used in robotics. For the latter to work, a certain standard had to be met while working on the architecture to make it compatible with most nodes, particularly on the message structures of each sensor.

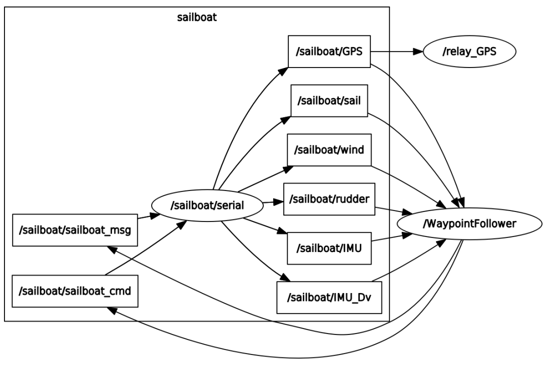

Figure 4: A graph of ROS topics and nodes when applying a waypoint-following control.

All the topics concerning the sensors are published by the Arduino. The Raspberry Pi only launches a node to connect to the Arduino and the controller node. The controller node then sends commands to the Arduino through the */sailboat/sailboat_cmd* and */sailboat/sailboat_msg* topics at a regular frequency, chosen by the user.

## 3.3 Safety

Multiple security measures are incorporated in the Arduino and Raspberry Pi architectures. For that purpose, watchdogs are both implemented inside the Arduino and the Raspberry Pi.

To be sure that the Raspberry Pi is still on or is not freezing, it sends a message every 30 seconds to the Arduino. If the latter does not receive anything for 5 minutes, it will consider the Raspberry Pi to be malfunctioned. Such safety measures have been implemented automatically in the software architecture and the user does not need to write any further code for them.

The Arduino itself have securities in case of blockage. With a watchdog, the Arduino will set itself in a safe mode if nothing is happening. A watchdog is present in most microcontrollers. This feature is present in the

Arduino and acts as a hardware timer that has to be reset frequently. If the timer goes over a certain amount, it will restart the microcontroller. This timer is independent of the software and will run whether the software is blocked or not.

When a watchdog is triggered i.e. when the Raspberry Pi is not responding or the Arduino is blocked, the lowest-level controller, the Arduino, will set itself into Return Home control where it will come back to the point where it was turned on.

Another emergency trigger is the Radio Controller. If the radio controller is turned on, it takes control over the autonomous mode and ignores the Raspberry Pi commands. Two modes have been incorporated. One manual where both the sail and the rudder has to be controlled using the RC transmitter. Another semi-automatic where the sail uses an optimal sail angle based on the wind sensor, the user just needs to control the rudder.

## 4 Open Source

The main objective of this work is to create an accessible, generic and expandable platform for autonomous sailboat development. Everything detailed here is available online at `https://github.com/Plymouth-Sailboat`. We use the Github platform to share publicly all the work, including the electronic circuits and the source code. A lot of efforts were taken to write a detailed guide for new users to integrate this control box into their sailboat so as to develop their own controllers. This open-source work will always be maintained, updated and upgraded along the researches and any feedback to be received. This will give other laboratories the opportunity to build their own sailboats at a low cost and a reduced effort.

## 5 Tests and Experiments

We tested this control box on two sailboats : A 1m mono-hull sailboat and a 2m mono-hull sailboat as shown in Figure 5. Actuators and sensors were bought and installed along the control box inside empty sailboat hulls. Two tests were done, first in a lake near the University of Plymouth, then a second test was performed on the west coasts of France. Each of these sailboats uses the same GPS and AHRS sensors but the wind sensors are different. The small boat has only a wind direction sensor, while the other has also an anemometer. The actuators on each boat are different. Two different configuration files were created for those boats but the same code was running on both the Arduino and the Raspberry Pi on each boat. This involved changing only one file called the *config-Sailboat.h* which contains all the different parameters of the hardware such as : minimum and maximum actuator angles, minimum and maximum values sent by the different sensors, numbers of sensors and actuators, etc...

A couple of well-known algorithms were tested. On the Arduino side, as stated above, only low-level controllers were used. One which takes the rudder and sail command directly from the Raspberry Pi, and another one receiving the cap and applying an optimal sail angle and an appropriate angle for the rudder. When receiving the cap, for the sail angle $\delta_s$ and the rudder angle $\delta_r$ we have :

$$\delta_s = \frac{\pi}{2}\left(\frac{\cos(\psi)+1}{2}\right) \tag{1}$$

$$\begin{cases} \delta_r = r_{max}\sin(\theta-\bar{\theta}), & \cos(\theta-\bar{\theta}) \geq 0 \\ \delta_r = r_{max} sign(\sin(\theta-\bar{\theta})), & else \end{cases} \tag{2}$$

with $\psi$ being the wind coming from the wind sensor (in the boat referential), $r_{max}$ being the maximum rudder angle, $\theta$ the boat heading and $\bar{\theta}$ the cap. We consider the sail maximum angle of $\frac{\pi}{2}$. A tack strategy is also applied in case of upwind navigation, similar to the one in (Jaulin and Le Bars, 2012).

On the Raspbbery Pi side, a potential field method based on (Petres et al., 2012), a simple waypoint follower and a new position keeping algorithm were tested (Viel et al., 2018) with each sending commands at 10Hz. Some results are shown in Figure 5, where the sailboats are shown in the left two figures while the right two figures show the potential field method and way-point following control, respectively. The result figures show the GPS trajectory of the sailboats recorded through ROS in a *rosbag*. These primary test results have confirmed the feasibility and the functionality of the current hardware and software design in terms of accessibility, generality and extendability. The sailboats were able to follow the commands from the high-level controllers on the Raspberry Pi. For the potential field test, only an attractive goal point was put. The test was stopped and retrieved using the RC controller at the middle of the test in the bottom of the picture for emergency reasons.

Concerning the waypoint following controller, we put 3 different GPS coordinates to follow, shown in green in the picture. The boat was able to use a tack strategy to reach the first point at the bottom of the figure and then attain the other two GPS points. It went on to the first GPS coordinate again before we collected the boat. While the results show a sub-optimal route to get to the waypoints, it shows nonetheless the feasibility of such a control box, configured easily to apply controllers to the sailboat. After examinations of the test results, the navigation problem on the waypoint-follower was due to a mechanical fault of the sailboat, constraining a maximum rudder angle on one side.

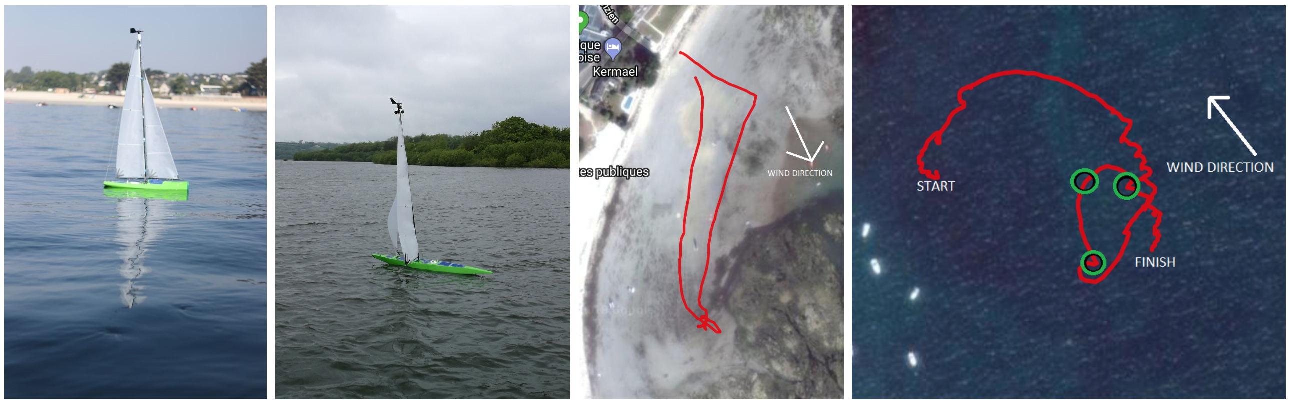

Figure 5: Sailboats with the control box and test results.

## 6 Conclusion

We have demonstrated the development and the test of an open source autonomous sailing robot with emphasis on its accessibility, generality and extendability. The hardware architecture is designed to accept most common sensors and communication protocols. It is made of accessible and economic products and a customized PCB board. The software architecture is designed to comply with the hardware by being generic and extensible. The goal is to create a platform of autonomous sailing robots for generic education and research purpose. This platform can also be beneficial for other relevant researches. Future work will focus on making the whole system more robust, economic and smaller while maintaining the same level of accessibility. It will also be tested on other formats of sailing robots such as catamarans and trimarans. The research will be shared along the work progresses and more control algorithms for various autonomous missions such as area scanning and obstacle avoidance will be implemented on it in the future work.

#### 6.0.1 Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of the U.K. under Grant EP/R005532/1.
We would also like to thank Professor Luc Jaulin from Ensta-Bretagne, France for providing testing facilities for us to test the developed sailboats in Brest.

## References

Alvira, M. et al. (2013). *Small and inexpensive single-board computer for autonomous sailboat control.*

Benchoff, B. (2016). Pi 3 benchmarks: The marketing hype is true. https://goo.gl/Nx7df2. Accessed: 01/06/2018.

Cabrera-Gámez, J. et al. (2013). An embedded low-power control system for autonomous sailboats. In *Robotic Sailing*.

Cousins, S. (2011). Exponential growth of ros [ros topics]. *Robotics & Automation Magazine, IEEE.*

Cruz, N. A. and Alves, J. C. (2008). Autonomous sailboats: An emerging technology for ocean sampling and surveillance. In *OCEANS*.

Domínguez-Brito, A. C. et al. (2015). A-tirma g2: An oceanic autonomous sailboat. In *Robotic Sailing*.

Elkady, A. and Sobh, T. (2012). Robotics middleware: A comprehensive literature survey and attribute-based bibliography. *Journal of Robotics*.

Ghani, M. H. and Hole, a. o. (2014). The sailbuoy remotely-controlled unmanned vessel: Measurements of near surface temperature, salinity and oxygen concentration in the northern gulf of mexico. *Methods in Oceanography*, 10:104–121.

Gregory, J. (2014). *Game engine architecture*. AK Peters/CRC Press.

Jaulin, L. and Le Bars, F. (2012). A simple controller for line following of sailboats. In *Robotic Sailing 2012*.

Krauss, R. (2016). Combining raspberry pi and arduino to form a low-cost, real-time autonomous vehicle platform. In *ACC*.

Lam, T. L. et al. (2016). System design and control of a sail-based autonomous surface vehicle. In *ROBIO*, pages 1034–1039.

Larabel, M. (2018). Raspberry pi 3 benchmark. https://goo.gl/LkDz46. Accessed: 01/06/2018.

Lazarin, N. M. and Pantoja, C. E. (2015). A robotic-agent platform for embedding software agents using raspberry pi and arduino boards. *9th Software Agents, Environments and Applications School*.

Liu, Z. X. et al. (2016). Unmanned surface vehicles.: An overview of developments and challenges. *Annual Reviews in Control*.

Marlin. Open source 3d printer firmware. http://marlinfw.org/. Accessed: 01/06/2018.

Meinig, C. et al. (2015). The use of saildrones to examine spring conditions in the bering sea: Vehicle specification and mission performance. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–6.

Miller, P. et al. (2014). *MaxiMOOP: A Multi-Role, Low Cost and Small Sailing Robot Platform*.

Naveau, M. et al. (2013). Marius project: Design of a sail robot for oceanographic missions.

Neal, M. (2006). A hardware proof of concept of a sailing robot for ocean observation. *Ieee Journal of Oceanic Engineering*.

O'Hara, W. J. (2017). ASTERiaS: Autonomous Sailboat for Titan Exploration and Reconnaissance of Ligeia Sea. In *Lunar and Planetary Science Conference*, Lunar and Planetary Science Conference.

Petres, C. et al. (2012). A potential field approach for reactive navigation of autonomous sailboats. *Robotics and Autonomous Systems*.

Plumet, F. et al. (2015). Toward an autonomous sailing boat. *Ieee Journal of Oceanic Engineering*.

Rathour, S. S. et al. (2017). *Development of a Robotic Floating Buoy for Autonomously Tracking Oil Slicks Drifting on the Sea Surface (SOTAB-II): Experimental Results*.

Rules, R. (2013). Rules for roar racing. https://goo.gl/2XjucQ. Accessed: 01/06/2018.

Sailboat, P. (2018). Open-source source code and documentation of the plymouth sailboat project. https://github.com/Plymouth-Sailboat. Accessed: 01/06/2018.

Santana-Jorge, F. J. et al. (2017). A component-based c++ communication middleware for an autonomous robotic sailboat. In Øvergård, K. I., editor, *Robotic Sailing*.

Silva Junior, A. G. et al. (2016). Towards a real-time embedded system for water monitoring installed in a robotic sailboat. *Sensors (Basel)*.

Viel, C., Vautier, U., et al. (2018). Position keeping control of an autonomous sailboat. In *Proceedings of the 11th IFAC CAMS 2018*.

Wirz, J. et al. (2015). Aeolus, the eth autonomous model sailboat. In Friebe, A. and Haug, F., editors, *Robotic Sailing*.