UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Máster Oficial en Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería

Master Thesis

A-TIRMA: A small autonomous sailboat

Ángel Ramos de Miguel

Tutores: Jorge Cabrera Gámez

Antonio Carlos Domínguez Brito

29 July 2013

Agradecimientos

A toda mi familia por todo el ánimo que me han dado en este trabajo, así como en todos los aspectos de la vida. Por prestarme atención y entenderme cada vez que les explicaba cualquier aspecto del proyecto.

Al todo los que trabajan en el instituto universitario de sistemas inteligentes y aplicaciones numéricas que me han hecho disfrutar de un año adquiriendo conocimientos que seguramente utilice el resto de mi vida profesional.

Por supuesto a mis tutores D. Antonio Carlos Domínguez Brito por proponerme el proyecto y toda la ayuda prestada en todas las fases, y a D. Jorge Cabrera Gámez por todas las horas, días y meses que ha dedicado en explicarme y ayudarme durante todas las fases del proyecto.

No puedo olvidar a los compañeros del equipo que han participado y espero que participen durante mucho tiempo, a José Daniel Rodríguez Sosa y a Bernardino Valle Fernández por las horas de diseño, debates de como debería comportarse el barco y las horas en la playa haciendo pruebas.

Por último a mi novia, por todas las horas que me ha esperado estando yo en el laboratorio.

Abstract

This master thesis describes the development of the A-TIRMA, an autonomous sailboat based on an One Meter Class RC vessel. The A-Tirma navigates autonomously with no human intervention except for establishing the route it will follow. One of the distinctive features of sailing robots is that they constitute really interesting operative platforms for monitoring and data sampling in aquatic environments, mainly due to its low power consumption requirements which allows them great operative autonomy. In addition, in the case of A-TIRMA, it is also easily deployable due to its small size and weight.

The control system of A-TIRMA consists of two main parts: a base station and the sailboat. From the base station (a notebook) we can command straightly the sailboat's sails and rudder remotely in RC (Remote Control) mode using a wireless link. In autonomous mode, we can make the boat to follow autonomously a route, which is a list of waypoints specified on a Google Maps interface. The boat is endowed with an embedded on board system, which, in autonomous mode, bases its operation in an autonomous fuzzy-logic based navigation algorithm for controlling the boat's sails and rudder, in order to navigate autonomously following the route of waypoints specified from the base station. During its development we have made intensive sea trials to check the operational capabilites of A-TIRMA and its control system at sea, and its autonomy in terms of power consumption and operative endurance.

Chapter 1

Introduction

This master thesis is concerned with the development of unmanned vehicles suitable for performing environmental monitoring and data sampling at sea.

The importance of the oceans in the overall health our planet Earth can't be underestimated. For a long time seas were considered vast spaces whose, once thought, inexhaustible resources could be blindly exploited for fishing, hunting, mining or even used for dangerous wastes disposal. Luckily, human kind is now conscious about the importance that the health of the seas has for the whole planet and its population.

The international scientific community has expressed clearly an repeatedly the necessity of improving the knowledge we have about the oceans. Specifically, this knowledge is crucial to address the problem of global climate change as evidences continue to accumulate in that respect.

Three are the key problem associated with in situ data gathering at seas. The first two are related to spatial and temporal data resolution. The other one is cost. Until very recently, data could only be obtained from localized points using moorings and buoys with very poor spatial resolution. Research and opportunity vessels have been the other classical mean for sampling at sea with good spatial and temporal resolution. However, the very high cost of a research vessel normally limits the duration of campaigns and impedes

the acquisition of data series with the desirable temporal and spatial density.

1.1 Unmanned Marine Vehicles

The necessity of cutting costs of data gathering and sampling for longer periods is pushing forward the development and utilization of unmanned vehicles in oceanographic campaigns. Unmanned marine vehicles can be broadly classified into two different categories: underwater vehicles and surface vehicles.

1.1.1 Underwater Vehicles

This category includes profilers, propelled underwater vehicles, also known as AUVs, (Autonomous Underwater Vehicles), underwater gliders.

Profilers and floats are devices capable of traversing the column of water regulating their buoyancy. They can only control its motion in depth and, in the long term, they moved in the ocean carried away by marine currents. Their power economy makes them very interesting instruments for long deployments. The ARGOS program [1] is perhaps the best known example of a large scale program based on this kind of devices.

An AUV [2] is a propelled torpedo-shaped robot which travels underwater autonomously. Their range and payload capacity is basically limited by its dimensions. Numerous examples of this kind of vehicles can be found on [3]. AUVs are now being used for more and more tasks with roles and missions constantly evolving. Meaningful cases of use can be found in science missions and commercial tasks in the oil and gas industries which use AUVs to make detailed maps of the seafloor before start building a subsea infrastructure. Also in the military sector, AUVs are used routinely to perform acoustic vigilance or to localize mines.

Underwater gliders [4] are a relatively new technology that is revolu-

tionizing this field. They move by gravity, regulating their buoyancy, but are normally equipped with wings that transform the vertical gravitational force into forward motion. Due to this form of impulsion they move relatively slowly, typically below 0.6 m/s, but their range autonomy is in the order of thousands of kilometers. In practice this autonomy is limited nowadays by biofouling and the power demands of the scientific instruments carried on board.

1.1.2 Surface vehicles

The term ASV (Autonomous Surface Vehicle) refers to any vehicle that operates on the surface of the water without a crew. ASVs have been tested since World War II and numerous examples exist of military vehicles of this type, but it has not been until recently that ASVs have started to be considered for scientific missions.

ASVs are valuable in oceanography, as they are more capable than drifting buoys, but far cheaper than research vessels, and more flexible than opportunity ship contributions [5].

We can find three different types of ASV depending on the source of energy they use for impulsion.The widest class is that of motorized vehicles power by electric motors or fuel engines. Another recent and revolutionary type of vehicle is the wave-glider [6], a vehicle impulsed by sea waves. Finally, there have been since ancient times vehicles that move impulsed by the wind, that is, sailboats. The main advantage of these two lasta types of ASV have over motorized ASVs is their virtually unlimited range autonomy. Wave gliders' main drawbacks is their limited payload capacity and relatively low speed of 1 m/s.

Sailboats offer good payload capacity and, depending on sail dimensions,

Table 1.1: Oceanographic platform relative capabilities in normal operating conditions (taken from [16]).

Vehicles	Manned Ship	Glider	Drifter	Sailboat
Platform Specs.				
Survey speed (m/s)	0-8	0-0.3	0-0.3	0-2.5 [1]
Survey time scale	Weeks	Months	Months	Months
Cost ($USD)	10,000/day	60,000	5,000	50,000
Adaptive sampling	Yes	Yes	No	Yes
Max. 24h range (km)	600	25	25	200
Payload vol/weight	$100m^3$/500kg	$0.1m^3$/5 kg	$1m^3$/25kg	$1m^3$/50kg

are capable of larger speeds. The main disadvantage is their limited capacity to surpass storm conditions unattended, at least with conventional sailboat designs.

Table 1.1. taken from [16], summarizes the previous discussion on several types of marine vehicles, comparing them across a number of characteristics.

1.2 Outline of This Thesis

This master thesis is committed with the design and implementation of an autonomous low cost and low power sailboat. The main goal of the project has been to transform a radio control sailboat into a complete autonomous system. Through this document we will discuss its sensors and actuators, describe the software on board and the implementation of a base station application to communicate with the sailboat.

Extensive tests have been carried out in real conditions, concretely in Alcaravaneras beach, in Las Palmas de Gran Canaria, Spain. Data collected during those sea trials will be used to describe her control system and discus her performance

[1] Survey speed varies with specific design and weather conditions. The presented values represent the range of current autonomous sailboats in an approximate sea state of 3 or less.

Chapter 2

The vessel

The One Meter Class sailboat is a developmental class, which means that there are very few design restrictions. The basic design restrictions include overall hull length, keel depth and sail area. The hull can be self designed or built from the scratch, and easily planked from wood or any other suitable material. The structure of the class allows the designer/builder to experiment with a design, and then try it out on the water.

The design restrictions for the One Meter Class are the following:

1. LOA(Length Over All): 99-100 cm
2. Maximum Mast Height: 1.65 m
3. Maximum Mast Diameter: 1.90 cm
4. Rudder must be aft of keel fin.
5. Maximum Boom Diameters: 1.90 cm

The One Meter Class is light weight, very fast and responsive to the controls. Two radio channels are required for control of sails and rudder. The boat is easily disassembled and fits in a small size automobile for transportation.

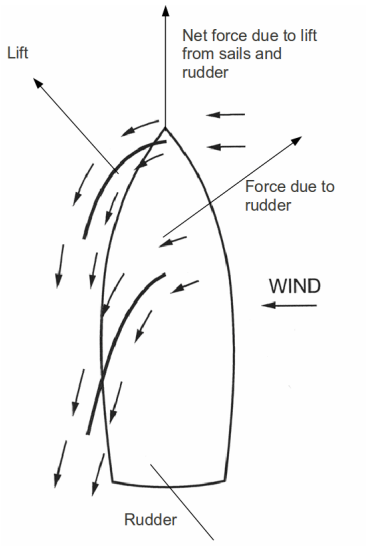


Figure 2.1: Sails physics

2.1 How Does A Sailboat Work?

The sailboats have been used since the Egyptians 5000 years ago [27]. The sailing technology has changed a lot since then but the main idea remains the same. The sails of a sailboat work like the wind of an aircraft, Figure 2.1. The power to move the boat is produced by a pressure difference between both sides of the sails, and this difference depends on the position of the sails in relation to the direction of the wind.

Figure 2.2 shows different positions of the sails with respect to the wind. For example, if the wind comes from the bow, we have to place the sails in up wind position, if we do not set the sail on that position we will lose all the power of the wind, and the sails will flutter. The sail positions might work also like a brake, in order to make the sailboat reduce its velocity. Fine sails regulation, i.e. how they are sheeted, strongly depends on wind intensity. For example, when sailing close hauled with strong winds, placing the sails in an optimum position may provoke over heeling, consequently reducing boat's velocity and even changing its bearing. Thus, depending on

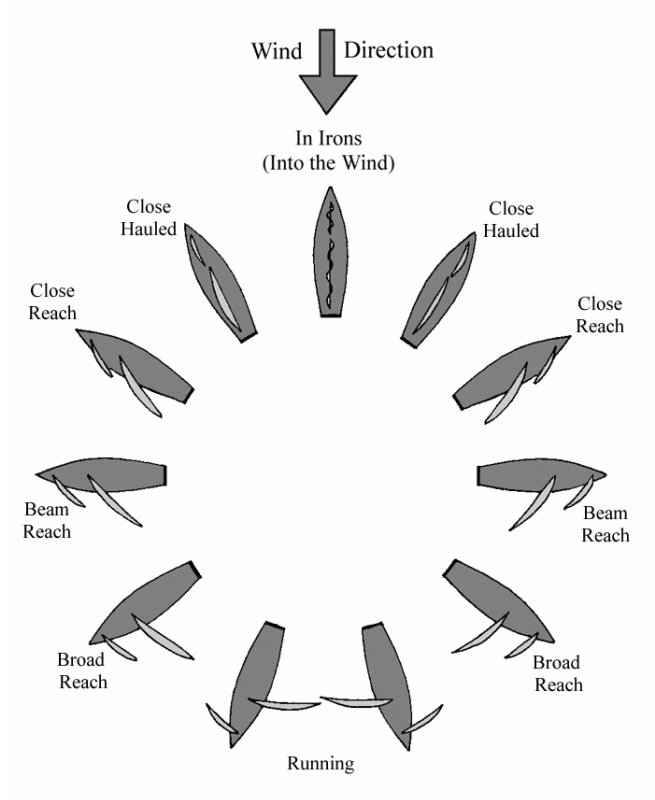


Figure 2.2: Sails Positions

wind direction, wind velocity and heeling we should place the sails and the rudder in a way that we avoid that situation.

2.2 Sails

Sails are the most important part of a sailboat. They provide the large potential autonomous sailboats have as high speed vehicles of virtually unlimited autonomy for environmental monitoring and sampling. Depending on their net displacement and dimensions, they can accept scientific payloads, that maybe, are too large or too power demanding to be integrated in other types of autonomous marine vehicles.

All in all, sails are the source of power of sail boats, and they have many

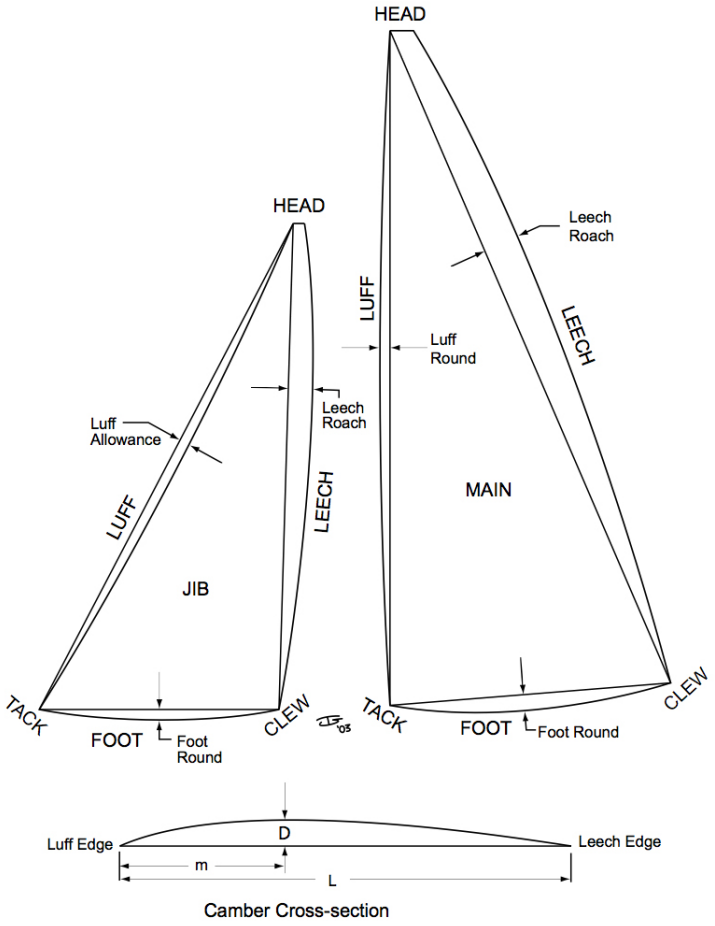


Figure 2.3: Sail parts [28]

parts as we can see in Figure 2.3.

2.2.1 Sail Construction

Sails come in two general varieties of construction; single panel, where the entire sail is one piece of cloth without seams, gores, or cuts in the body of the sail; and paneled, where each sail is made up of panels or strips of cloth, attached edge to edge with tapered seams to induce three dimensional shape in the sail when filled with wind.

Single panel sails of a woven material are typically encountered in the

construction of "kit" boats, because single panel sails are much less costly for the kit maker to provide. The details of successful setting, trimming and tuning of single panel sails is an entire subject in itself. Single panel sails must be full of air, so that the cloth can stretch under the wind loading and can begin to take on the cambered shape necessary for drive.

Paneled sails are used in a greater proportion in racing classes, and also in scale models that are going to be operated on the water. They provide superior performance because the airfoil shape that produces drive for propelling the hull is built into the sail. Most use a membrane material such as a mylar sandwich with load carrying fibers, or mylar film. These modern materials do not stretch, so the cambered shape must be built into the sail with tapered seams that hold the panels together.

We use single panel sails because they are less complicate to trim than the paneled ones, and, in addition, with only one configuration we can navigate in a variety of conditions.

2.2.2 Sail Attachment

A sail is attached to the spars by each of the three corners and by the luff of the sail to either the mast or the jibstay. In the case of the jib, luffs are fitted with a hem into which the jibstay slides. This supports the sail along its length and makes for a smooth and clean entry where the wind meets the sail. The luff hem method eliminates scalloping which can happen with the tubes, and it distributes the wind load evenly over the entire luff. The sail tubes provide for excellent swiveling as the sail tacks, but produce point loads where they attach to the sail, and this can cause the sail to wear out prematurely.

The head of the jib usually contains a grommet to which is attached a halyard. For ease of adjustment, the halyard is fixed, usually up at the jibstay attachment point. Tension is adjusted on the sail luff by use of a

downhaul, which is tied to the tack grommet, led to the jib club, and back about half way down the club to some sort of a tension adjustment like a cleat, 3-hole bowsie,or other device. The clew is constrained in the vertical direction by something as simple as a loop of sheet line tied through the clew grommet and led under and around the club. A second line, called the clew outhaul, leads aft to the end of the jib club, and provides for adjusting the position of the clew along the club, controlling the fullness of the foot and the bottom third of the jib. The corners of the mainsail are rigged in the same way.

More details on model sailboat rigging techniques can be found on [28].

2.3 A-TIRMA Sailboat

The sailboat described in this master thesis has been named "A-TIRMA", or Autonomous-TIRMA, after the famous canary TIRMA sailboat [29]. This vessel was built in 1910 in the shipyards of San Telmo, in Las Palmas de Gran Canaria, Spain. It won its first race on October 25, 1911. One of its most memorable achievements took place in 1960 when, under the command of skipper D. Ventura Quevedo, managed to circumnavigate the island of Gran Canaria in 27 hours and 8 minutes, beating the previous record at the time. In November 10^{th} 2000 it was rebuilt and nowadays it is exposed at the building entrance of the Real Club Nautico of Gran Canaria.

A-TIRMA is based on a carbon fiber One Meter Class vessel with mainsail and jib displayed in Figure 2.4. It is equipped with two analog RC servos, acting as actuators for rudder and sail sheets, that are powered from a six NiMH rechargeable AA cell battery set with a capacity of approximately 20 Ah. The boat has been equipped with a custom-made wind vane situated on the top of the mast for sensing the apparent wind direction, and other electronics that will be described later.

Figure 2.5 details the dimensions of the sailboat. The most important

Figure 2.4: Our sailboat

dimensions of the vessel are:

LOA	100 cm
Beam	24.5 cm
Draft	14 cm
Sail Area	0.61 m^2
Displacement	4.3 kg
Mast Height	1.6 m

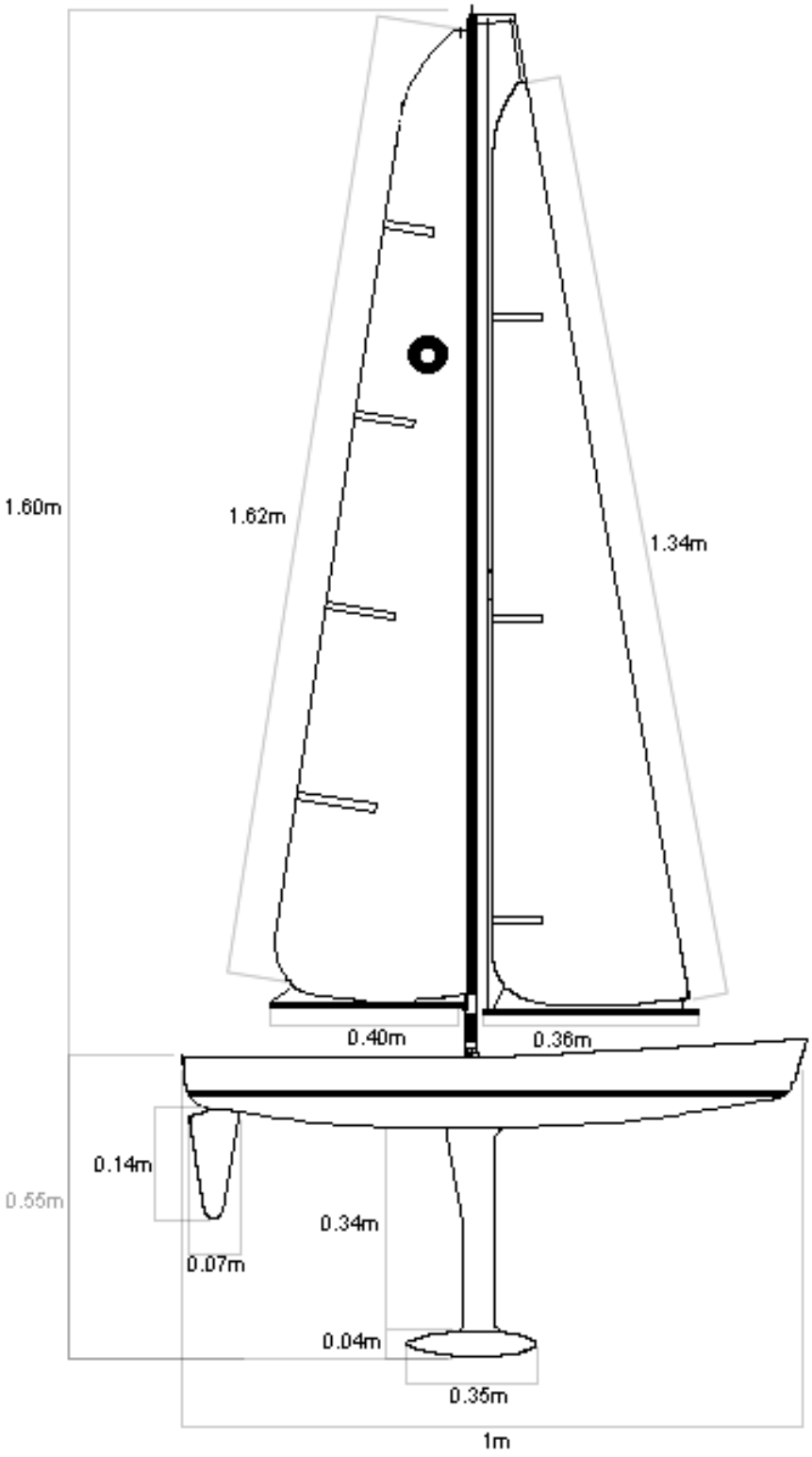


Figure 2.5: Sailboat dimensions

Chapter 3

State of Art

This chapter is dedicated to review the evolution of the technical developments in sailing. The historical antecedents of robotic sailing date back to the automatization of the rudder control and the utilization of self-steering mechanisms, but continue nowadays with the recent appearance of kite-based proposals to use the wind to propel high-tonnage vessels with fuel cost reductions of up to 20% [8]

3.1 Self-Steering Gear

The first task to be automated was the rudder control [9] which is also referred to as autopilot. Self-steering gears can be divided on mechanical or electronic solutions.

3.1.1 Mechanical Self-Steering

The first approach to a mechanical self-steering system was a solution created by fishermen who set the rudder of their boat to a fixed position.

Then Herbert Hasler designed a more sophisticated mechanical solution, the wind vane, an example is shown in Figure 3.1. It basically consists of a wind vane which is connected to the rudder, and when the angle of the apparent wind changes, the wind vane detects the change and activates the steering device to return the boat to the selected course.

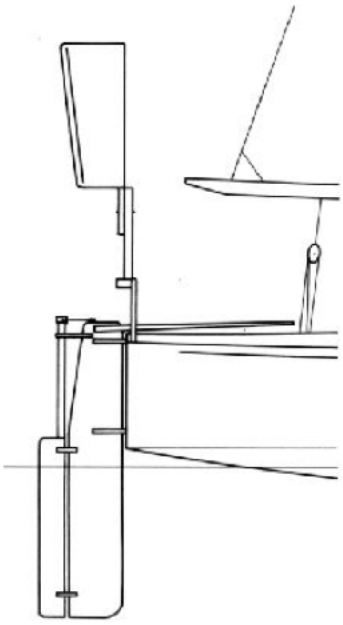


Figure 3.1: Wind Vane [18]

3.1.2 Electronic Self-Steering

Electronic self-steering controls the rudder position based on various sensors data. This system needs a compass, a wind direction sensor and additionally a GPS receiver. The major contribution to the development of a self-steering system was made by Sperry Gyroscope Company in 1911 [21]. This system compensated for varying sea states using a feedback control and automatic gain adjustments.

Nicholas Minorsky did a contributions to autonomous ship steering, he presented a detailed analysis of a position feedback control. He formulated the specification of a proportional-integral-derivative (PID) controller in [22]. But this type of controller has two disadvantages:

1. It is difficult to adjust it manually, because the operator usually does not have enough insight about control theory.

2. The optimal adjustment varies depending on the situation. Changing circumstances require readjustment of settings.

Due to the dynamic and ever-changing environment, artificial intelligence and fuzzy logic has been studied for rudder control. Various publications have shown the suitability of fuzzy logic for rudder control and also for sail control [23]. Polkinghorne et al. [24] did a comparison between a conventional PID and a fuzzy logic controller. The experiments shown a much smoother rudder action for the fuzzy logic controlled rudder.

3.2 Automatic Sail Control

Most sail control strategies published for autonomous sailboats are based on measuring the apparent wind. Many of them have virtually an infinite number of sail positions, when the system sets a sail position, the real position depends of the resolution of the actuator. Some researchers use only 10 discrete positions on their system, MOOP (University of Aberystwyth, UK) [25], to avoid permanent switching between adjacent positions to save

power on the sail actuators [26].

There is another method described by Stelzer et al. [23] which does not calculate directly the sail positions using wind data. It first calculates the desired heeling of the sailboat, and then a feedback-loop implemented as a Mamdani type fuzzy inference system controls the sail positions to reach the desired heeling of the boat.

3.3 Recent Technical Innovations

Sailing is an ancillary art but still in constant evolution. Every year international sailing races are used to announce new, sometimes radical, technical innovations based on new designs, exotic materials or CFD results, all them aimed at achieving faster vessels.

Autonomous sailboats may, of course, benefit from these technological advances but the goals that drive the development in this field of robotics are completely different. Here, much more important than speed are issues like robustness, seaworthiness and energetic efficiency. In this section we will briefly review some ideas and technological innovations that are nowadays of interest for autonomous sailboats.

Rig innovations. Rigid wind sails and balanced rigs [10] are two of the innovations that have been experimented recently on autonomous sailboats. Wind sails are rigid sails that have been used in some designs developed at Aberystwyth. They are robust yet light sail designs whose wing-like profile is promoted as aerodynamically more efficient that the classical soft sails. On the downside, rigid sails demand a careful and expensive design based on high tech materials, they can not be reefed and are considered fragile

under bad weather conditions. On the other hand, the balanced rig is self tacking, reduces mechanical loads on the rig and, overall, can be more energy efficient than an equivalent sloop rig[11].

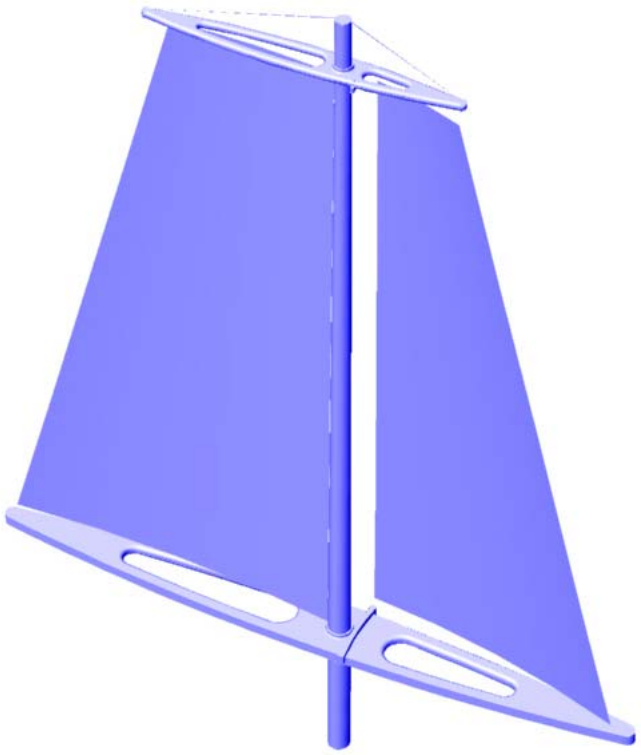


Figure 3.2: Balanced rig concept [10]

Flexible hull. Protei [12] is an international volunteer-based organization that is designing a new concept of sailboat that can be applied on autonomous cleaning of oil spills. In this kind of application, the boat must tow a large floating structure designed to absorb the oil at surface. This structure renders useless the classical stern rudder to head the boat. To deal with this basic problem, a flexible shape-shifting hull has been proposed by the Protei team. According to the authors, its benefits are:

- Better trajectory control, greater "pulling capacity".
- Better maneuvering, smaller radius of turning, dynamic stability.
- Lateral lift (due to curved hull profile, at high speed).
- Less resistance and turbulences (no centerboard, no rudder), compensating environmental noise with flexible body.

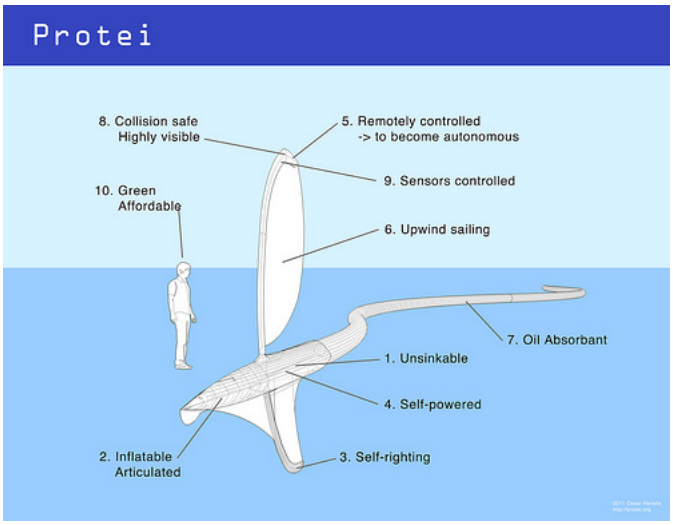


Figure 3.3: Protei, a shape-shifting hull sailboat [?]

Energy saving: Most of the energy consumed at a sailboat is used in sheeting sails and, in general, energy-aware control systems do not adjust the sails except when necessary. A good example of this approach taken to the extreme, that is, no sail actuator is used, is Sailbuoy [15]. In this 2m boat the wing sail is free to swing from side to side following the wind and its angle is not regulated. Accordingly, no wind sensor exists on board. Sailbuoy is also remarkable because it is - to our knowledge - the only commercial product in this ambit, designed to act as a surface sampling vehicle.

Figure 3.4: Sailbuoy [15]

3.4 Autonomous Sailboat Competitions

Two international competitions, Sailbot [13] in North America, and the World Robotic Sailing Championship [14] in Europe, are common meeting forums for researchers and students working this field.

It is beyond the scope of this document to extensively review all sailboats that have participated on this competitions and that information can be obtained from the competitions' web sites. The following tables just provide some examples of autonomous sailboats that have taken part in the last editions of the World Robotics Sailing Championship [46] to provide the reader a summary on the main characteristics of the boats.

Table 3.1: FAST - University of Porto (FEUP)

Team: FAST - University of Porto (FEUP) and INESC TEC
Location: Porto, Portugal
Class: Microtransat
Length: 2.5 m
Displacement: 60 kg
Draft: 1.25 m
Beam: 0.67 m
Hull Type: Based on a scaled down version of a Class 40 hull.
Sensors and Computers: FPGA-based computer running Linux
Power Systems: 190 W/h of Lithium Ion batteries and 45 W peak of photovoltaic solar panels.

Figure 3.5: FASt - FEUP

Table 3.2: BeagleB - Aberystwyth University

Team: Department of Computer Science, Aberystwyth University **Location:** Aberystwyth, Wales, UK **Class:** Microtransat **Length:** 3.65 m **Displacement:** 280 kg **Draft:** 0.65 m **Beam:** 0.86 m **Hull Type:** Based on a Mini-J disabled sailors dinghy hull. **Sail Type:** 2 square metre carbon fibre wing sail. **Sensors and Computers:** GPS, Fluxgate Compass, Ultrasonic windsensor, Gumstix Single Board Computer **Power Systems:** 2.8 KW/h of lead acid batteries and 90 W peak of photovoltaic solar panels˙ **Communications:** Wifi, Iridium Satellite Modem, optional GSM.

Figure 3.6: Beagle - Aberystwyth

Table 3.3: Gill the Boat - US Naval Academy

Team: US Naval Academy
Location: Annapolis, Maryland, USA
Class: Sailbot
Length: 2 m
Displacement: 25 kg
Draft: 1.5 m
Beam: 0.6 m
Hull Type: Custom designed hull built for the SailBot class.
Sail Type: Bermudan style fabric mainsail and jib.
Sensors and Computers: Rabbit controller mounted on a in-house designed and built board that includes a GPS receiver. Relative wind is measured by an optical encoder and she has a working ultrasonic collision avoidance system.
Other notes: She was wholly designed, built and is operated by the midshipmen of the United States Naval Academy. She has reached speeds of over 6 knots and has taken voyages up to 21 nautical miles.

Figure 3.7: Gill The Boat - USNA

Table 3.4: Avalon - Swiss Federal Institute of Technology Zurich (ETH)

Team: Swiss Federal Institute of Technology Zurich
Location: Rämistrasse, Zürich, Swiss
Class: Microtransat
Length: 3.95 m
Hull Type: A monohull design.
Sensors and Computers: The control system is implemented on a MPC2120 industrial computer running Linux.
Power Systems: The power supply is realized with four solar panels of 90 Wp, four lithium-manganese batteries of 600 Wh each and a direct-methanol fuel cell for back-up power.

Figure 3.8: Avalon - Swiss Federal Institute of Technology Zurich (ETH)

Table 3.5: Roboat I - Austrian Society for Innovative Computer Sciences (INNOC)

Team: Austrian Society for Innovative Computer Sciences
Location: Viena, Austria, Europe
Class: Microtransat
Length: 1.38 m
Height: 1.73 m
Displacement: 17.5 kg
Hull Type: Based on yacht model of type Robbe Atlantis.
Sensors and Computers: The boat is equiped with a 800 MHz PC running Linux, a GPS receiver, a tilt-compensated electronic compass and sensors for wind speed and wind direction.
Power Systems: It is equipped with solar panels providing up to 285 Wp of power during conditions of full sun and a direct methanol fuel cell delivering 65 W as a backup energy source.
Communication Systems: WLAN, UMTS/GPRS and an IRIDIUM satellite communication system.

Figure 3.9: Roboat I - Austrian Society for Innovative Computer Sciences (INNOC)

Table 3.6: VAIMOS - École Nationale Supérieure de Techniques Avancées Bretagne (ENSA)

Team: École Nationale Supérieure de Techniques Avancées Bretagne
Location: Bretagne, Berst cedex, France, Europe
Class: Microtransat
Length: 3.65 m
Hull Type: Based on yacht model of type Robbe Atlantis.
Sensors and Computers: A Linux-based embedded computer, a weather station (that measures the wind speed and direction as well as GPS position), an AHRS (Altitude and Heading Reference System), a Iridium communication system and actuators for sail and rudder control.

Figure 3.10: VAIMOS - École Nationale Supérieure de Techniques Avancées Bretagne (ENSA)

3.5 Field Applications

Autonomous sailboats are silent vessel with a virtually unlimited operational range, restricted only by wind and sea conditions, that may host large payloads and measure atmospheric and ocean parameters simultaneously. All these features confer them an enormous potential to serve as sampling and monitoring marine vehicles.

All in all, the field has still to mature and produce reliable and robust designs, capable of dealing with hard conditions at sea on their own means. Regardless of this state of affairs, field applications developed with autonomous sailboats have been published. Here we summarize some of them:

- **Acoustic mapping and cetacean tracking**. Recently, the INNOC's ROBOAT sailboat has been used to acoustically track cetaceans in the Baltic Sea [19] under strong sea conditions.

- **Water quality sampling measurements**. Rynne and von Ellenrieder [17] have described the development of the Wasp, a 4.2m monohull keelboat with wing sail, and its application in seawater quality control.

Chapter 4

Navigation And Control

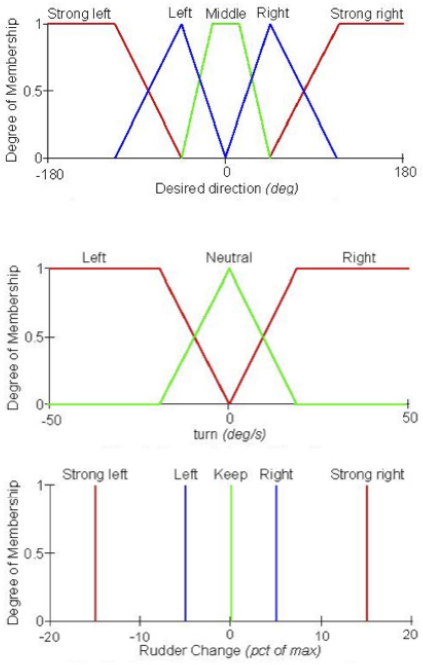
This master thesis has focused on the development of the low level control of the rudder and sails and a bearing selection algorithm, i.e. the determination of the optimal bearing to navigate the sailboat to a destination taking into account the wind direction. This work has been based on two influential and recognized articles of the field: "Fuzzy logic control system for autonomous sailboats" [23] and "Autonomous sailboat navigation for short course racing" [30]. These papers have been reviewed in the recent PhD thesis of Roland Stelzner[20]

The first paper, "Fuzzy logic control system for autonomous sailboats"[23], describe how to transform the sailors knowledge into a Mamdani-type fuzzy system, to control a sailboat with two actuators, rudder and sails. It provides the low level sailboat controller.

The second paper, "Autonomous sailboat navigation for short course racing"[30], proposes a method to select the optimal bearing to reach a destination given the true wind direction and the current sailboat position and heading. This will provide a second control level for the sailboat.

4.1 Low Level Control

The authors of this article [23] propose an algorithm to control the rudder and sails of a sailboat. The paper describes a fuzzy logic system to manage the control surfaces of the vessel (sails and rudder) like a sailor would do. It is described a fuzzy method to control the rudder that takes into account the difference between the sailboat bearing and the desired bearing, and also the angular velocity of the sailboat to avoid oversteering.



FUZZY RULES FOR RUDDER-FIS

Rudder Change		Turn		
		Left	**Neutral**	**Right**
Desired direction	**Strong left**	Left	Strong left	Strong left
	Left	Keep	Left	Strong left
	Middle	Right	Keep	Left
	Right	Strong right	Right	Keep
	Strong right	Strong right	Strong right	Right

Figure 4.1: Fuzzy rudder position [23]

On the illustrations shown in Figure 4.1 shows the fuzzy sets for the different sails positions. Where, on the first fuzzy set, we see the difference between the actual bearing and the desired bearing. On the second we can see the angular velocity in degrees per second, positive if the sailboat is turning to the right side (starboard) or clockwise and negative on the left side (port) of the sailboat or anticlockwise. On the third one, the different positions of the rudder appear. Finally, on the last illustration we have the fuzzy rules applied to set the rudder to a given position depending on the current bearing and the desired bearing, and the turning rate.

For controlling the sails the authors propose to use a function 4.2 that, given a wind direction and a wind speed, returns the optimum heeling. Finally, the heeling angle if fixed adjusting sails sheeting, Figure 4.3.

$$h = \max\left(0, \left(h_{max} - k \cdot |\alpha|\right) \cdot \frac{\min(v, v_{max})}{v_{max}}\right)$$

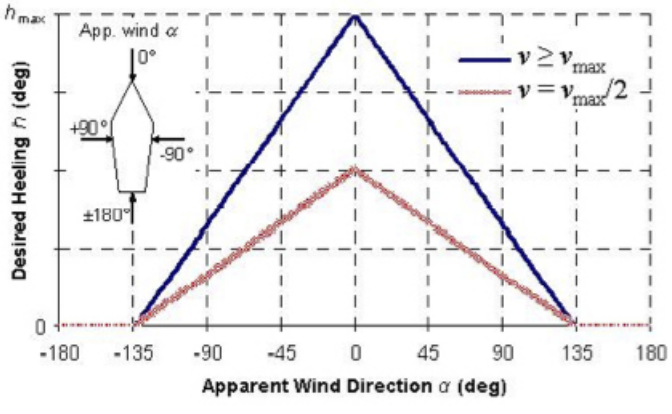


Figure 4.2: Desired heeling function [23]

The optimal heeling function depends on the variables detailed on Table 4.1.

Table 4.1: Variable meaning

h_{max}	Maximum heeling of the sailboat.
v	Actual velocity of the sailboat.
v_{max}	Maximum velocity of the sailboat.
α	Wind direction.
k	boat specific constant determined by experiments.

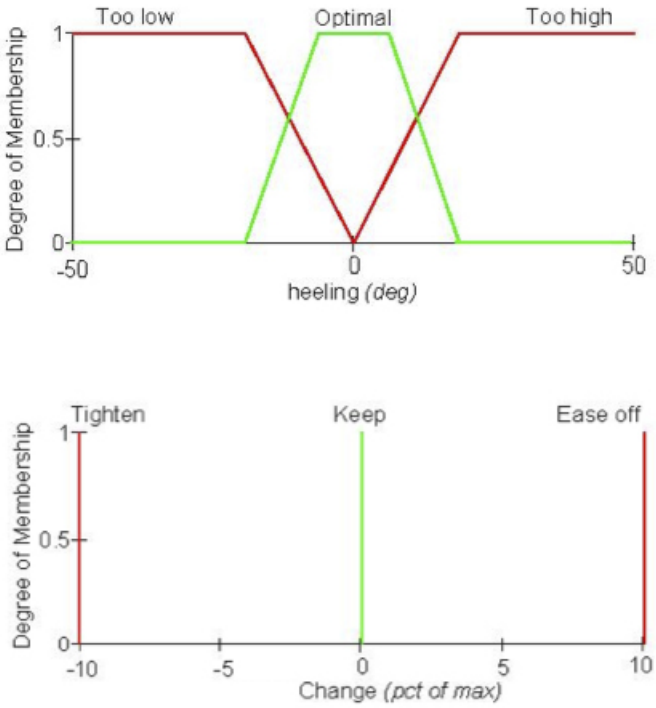


Figure 4.3: Fuzzy sails [23]

The fuzzy sets included in Figure 4.3 define the vocabulary for the desired heeling. Using the values given by the function shown in Figure 4.2, the actual heeling of the sailboat is subtracted to the desired heeling, and the difference is used to calculate the new sails position with the fuzzy sails

control rules.

4.2 Autonomous Sailboat Navigation

In [30] it is proposed an algorithm to calculate the next best bearing for a sailboat. The basic idea behind the algorithm is to select the bearing that maximizes the velocity made good or velocity projected in the direction to destination. This basic idea is modulated by a hysteresis parameter to avoid tacking constantly. The algorithm uses as basic parameters the real wind direction and speed, the destination coordinates and the sailboat position and polar. Figure 4.4 shows a flowchart of the bearing selection algorithm.

This algorithm is considered by the authors as a short term routing algorithm. In fact, that is possible as far as the following two conditions hold:

- The true wind is the same all over the area between the current boat position and destination.
- The true wind will remain for the whole leg as it is in the moment.

The flowchart starts with the initialization of the variables that the function will use. The data needed to select the next bearing are:

Table 4.2: Flowchart symbols

B	Current boat position
T	Target position
$\varphi(v_b)$	Current boat heading
$\varphi(-W_{abs})$	True wind direction and speed
n	Hysteresis factor

The current boat position and orientation are needed to decide if it is necessary to do a tack maneuver, then the wind speed it necessary if the

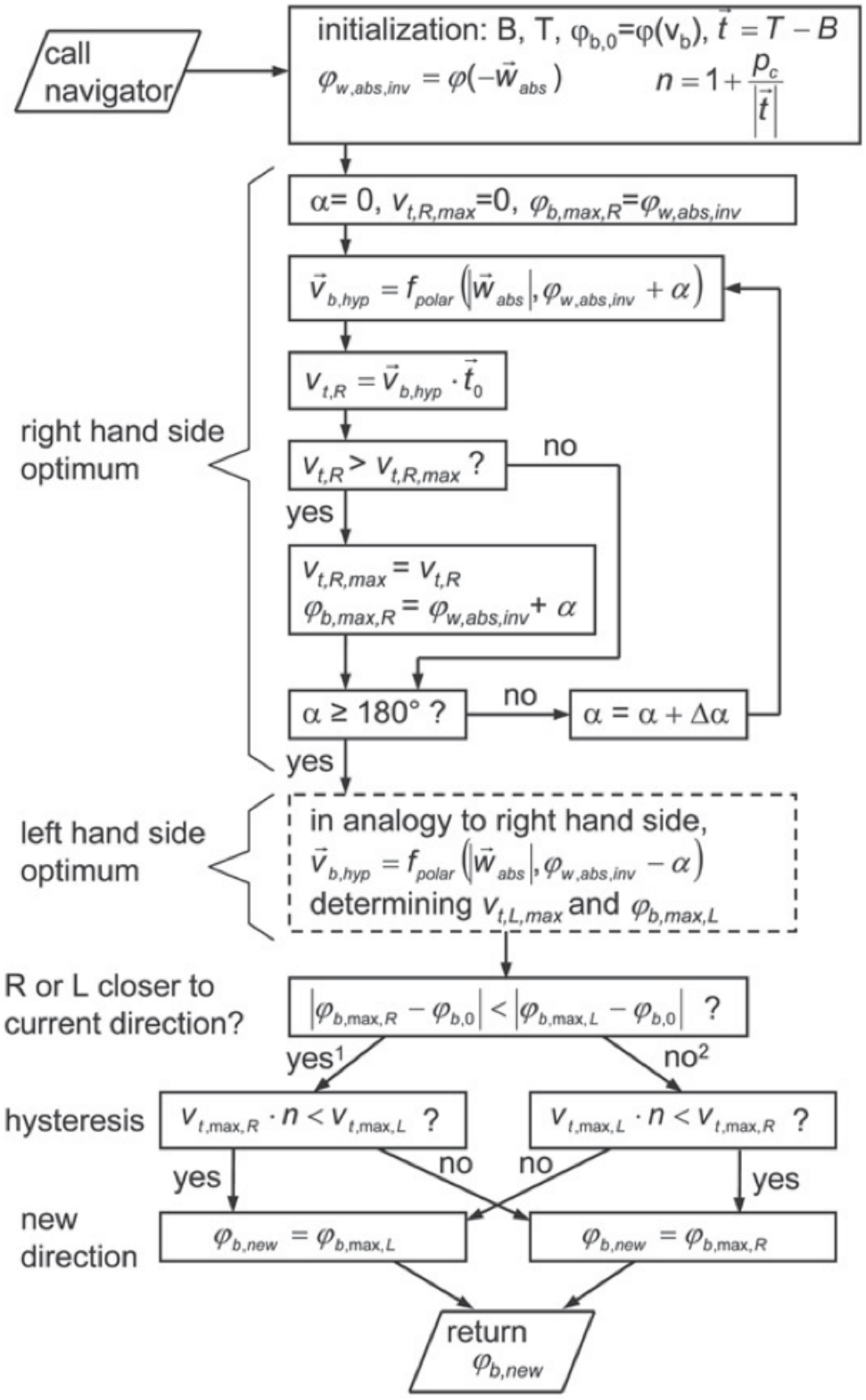


Figure 4.4: Bearing selection algorithm [30]

polar diagram of the boat it significantly non-linear and finally we need the true wind direction.

When the function has initialized all the variables, it calculates the best bearing to the target by the right side and then by the left side. To calculates de best direction, the function calculates the projected velocity to the target. If the selected direction gives a faster velocity to the target, we update the new direction and then we test with other direction.

Now we have the best bearing by the right side and the left side. The function will return the closer bearing to current direction to minimize the tack maneuvers.

Chapter 5

Onboard Electronics

The vessel's electronics is made up of the following main components:

1. An 8-bit microcontroller board
2. A XBee PRO 868MHz RF module
3. A GPS receiver
4. An electronic compass with inclinometers
5. A wind vane
6. A current sensor

5.1 System Hardware

The sailboat controller hardware is based on a Waspmote V1.1. It is a commercial credit-card size board based on a microcontroller ATmega1281 running at 8MHz. The microcontroller integrates 8KB of SRAM for data, 128 KB of FLASH program memory for program code and a 4 KB EEPROM [31]. The board provides several UARTs, an I^2C bus, a micro SD card reader, a real-time clock, a three-axis accelerometer and several other sensors for measuring, for example, the board temperature or the battery level.

This board is prepared to accept external hardware modules like a GPS receiver, a GSM/GPRS modem or different XBee RF communication modules. It is powered from a 3.7V 6000 mAh Li-Ion battery and consumes 9 mA under normal operating conditions. Suitable photovoltaic panels can be connected directly to the board to recharge the main battery.

5.1.1 Waspmote

The Waspmote board is an embedded system inspired in the microcontroller based electronic prototyping platform Arduino [32], but, Waspmote gives us more sensors and sockets in only one board. Its hardware architecture is modular in a way that, we can add or remove off-the-shelf hardware modules easily to the device.

The modules available for integration in Waspmote can be categorized in:

- ZigBee:/802.15.4 modules (2.4GHz, 868MHz, 900MHz).
- GSM - 3G/GPRS module (Quadband: 850MHz/ 900MHz/1800MHz/1900MHz)
- GPS module
- Several sensor modules (sensor boards)
- Storage Module: micro SD Memory Card
- Photovoltaic solar panels

Figures 5.1 and 5.2 show the main components and sockets available for connecting external devices and power sources.

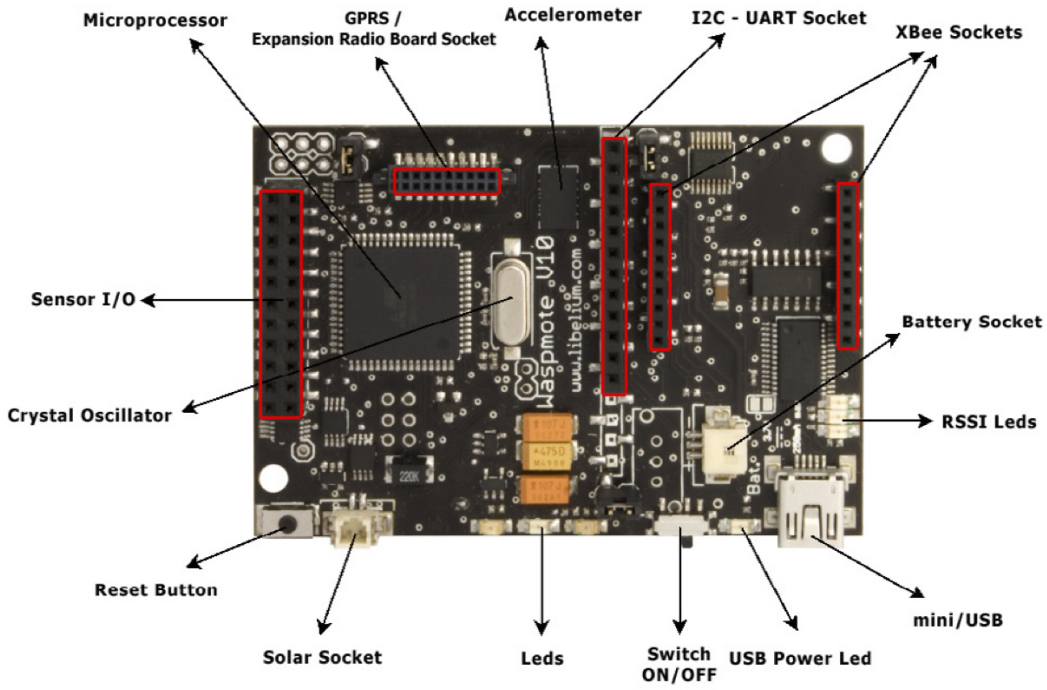


Figure 5.1: Waspmote board V1.1 – Top side

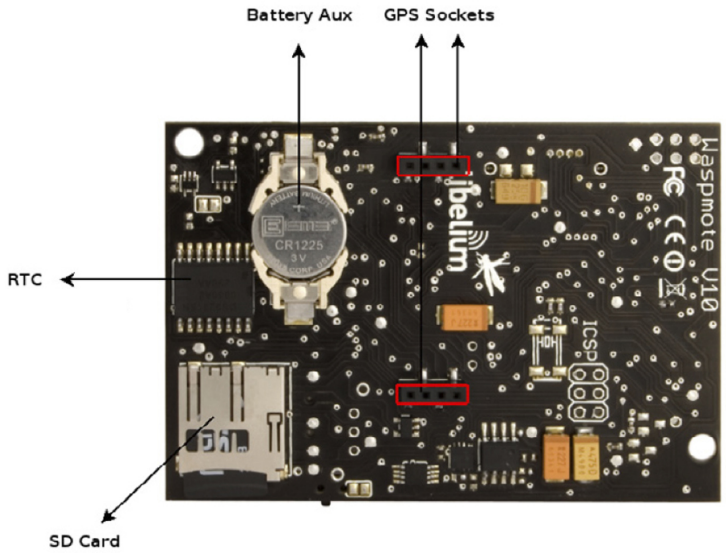


Figure 5.2: Waspmote board V1.1 – Bottom side

5.1.2 Electrical Characteristics

The electrical characteristics of the Waspmote board are detailed in the following list:

Operational values

- Minimum operational battery voltage	3.3V
- Maximum operational battery voltage	4.2V
- USB charging voltage	5V
- Solar panel charging voltage	6-12V
- Battery charging current from USB 100 mA (max)	100mA
- Battery charging current from solar panel	280mA
- Button battery voltage	3V

Absolute maximum values

- Voltage in any pin	[-0.5 V, +3.8 V]
- Maximum current from any digital I/O pin	40mA
- USB power voltage	7V
- Solar panel power voltage	18V
- Charged battery voltage	4.2V

As we can see, we must to be careful with all the modules that we connect to the Waspmote, because we should not exceed the maximum current and voltage.

5.1.3 Input/Output

A Waspmote may communicate with other external devices, that it is possible because the board has many input/output pins as we can see in Figure 5.3. The board is provided with digital input/output pins, analog inputs, PWM and power outputs, SPI, I^2C and UART interfaces, etc. This is one of the most important features for us, because we have to control servos, read analog sensors and communicate with sensors like GPS or compass over

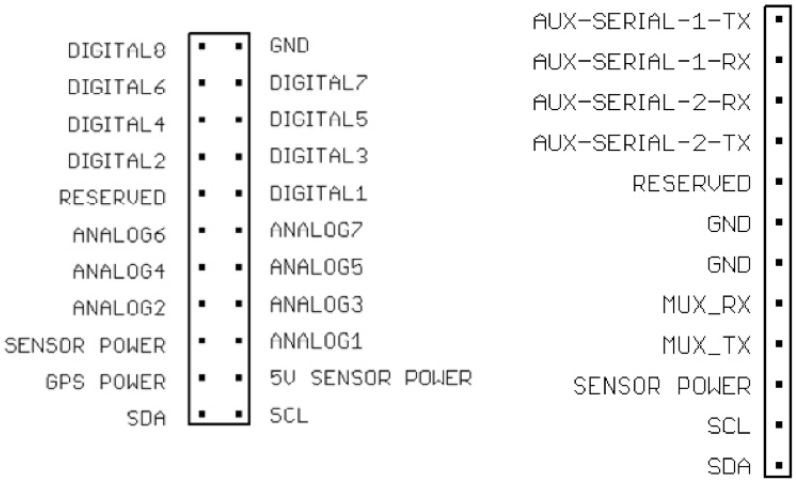


Figure 5.3: Waspmote pinout

serial interfaces.

5.1.4 Working Environment

To develop software for the waspmotes we have used the Waspmote IDE [31] which is based on the open source IDE for the Arduino platform [32], following the same style of libraries and operation. It includes all the API libraries necessary to compile the programs.

As we can see in Figure 5.4 the Waspmote-IDE has a code editor, an area for messages, a terminal, a toolbar with buttons for the main functions and a menu. Program loading on the Waspmote is carried out using a serial connection over an USB port .

In Figure 5.5 we can see the toolbar of buttons where we have the most important functions of the IDE. The next list describes the functionality associated to each button.

- Verify/Compile. Verifies the code and compile it.
- Stop. Stops the compilation.

Figure 5.4: Waspmote IDE

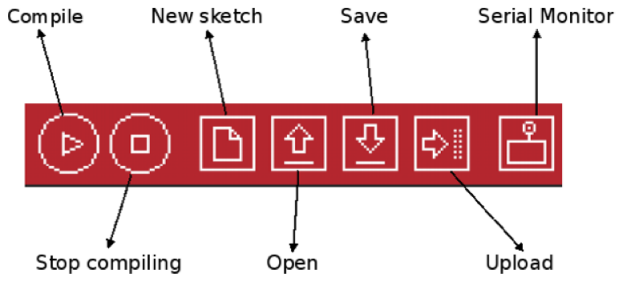


Figure 5.5: Waspmote IDE - Buttons

- New. Creates a new sketch.
- Open. Opens a menu to find a sketch and open it.
- Save. Saves the changes of current sketch we are editing.
- Upload to Waspmote board. Compiles the code and upload it to the Waspmote board.
- Serial Monitor. Starts the Waspmote's serial console.

5.1.5 Over The Air Programming (OTAP)

The concept of *Over The Air Programming* [33] or OTAP is commonly used in the scope of sensor networks for denoting the possibility of programming a sensor node wirelessly. In this project this capability has alleviated the hassle of having to open the deck to access to the inner of the vessel, and to extract the plastic container that hosts the Waspmote, just to connect a USB cable every time a new firmware had to be uploaded to the sailboat.

The OTAP process involves the following steps:

- Locate the node to upgrade
- Check current software version
- Send the new program
- Store the new program on the SD card
- Reboot and start with the new program
- Restore the previous program if the process fails

More details on the OTAP process can be found in Libelium's OTAP manual [33].

5.2 RF Radio Module

We have based all communications with the sailboat on XBee 868 Pro RF modules, Figure 5.7. These modules operate at the 868 MHz ISM band using only one channel. The bandwidth is 24 Kbps and the communications can be encrypted. The nominal range using a 4.5 dB dipolar antena in a free field is 40 km, but more realistic estimations are in the range of 10 km. It is possible to adapt the transmission power in five levels from 1 mW till a maximum of 315 mW. It works at 3.3 V and its current consumption is 500 mA in transmission and 65 mA in reception [34].

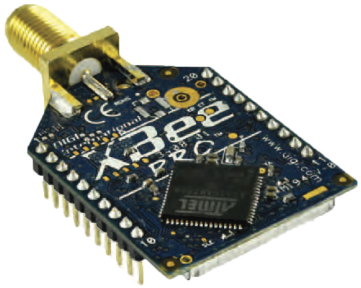


Figure 5.6: XBee Module

The position of the antenna on the sailboat is very important to have a good communication with the station base. We first placed it on the mast, but with that configuration the aluminum mast blocked the reception in some courses impeding the communication with the sailboat. After some testing, we found that the antenna could be fixed to the backstay, without disturbing the wind vane using a deformable support, with good results.

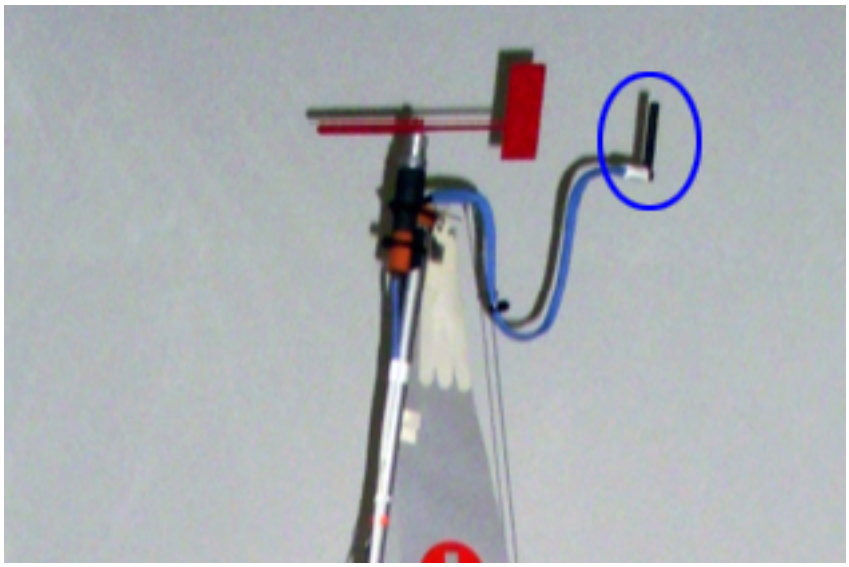


Figure 5.7: The 868 MHz 0dBi antenna in its final placement

5.3 Electronic Compass

The electronic compass is a legacy TCM2.50 board [35]. Basically, it provides tilt-compensated heading information and instantaneous pitch and roll angles over a RS232 interface. The board temperature and raw readings from three magnetometers can be also obtained. The compass readings are tilt and roll compensated till 50 degrees. The TCM2.50 can not operate for heeling or pitching angles over that limit. The TCM2.50 has a maximum update rate of 20Hz. It is powered at 5 V and consumes 20 mA.

This board is connected to one of the microcontroller TTL serial ports using a simple level converter circuit based on a MAX3232 integrated circuit (IC).

The MAX3232 [36] is an IC, first created by Maxim Integrated Products, that converts signals from a RS-232 serial port to signals suitable for use in TTL compatible digital logic circuits and vice versa. The MAX3232 is a dual driver/receiver and typically converts the RX, TX, CTS and RTS

signals.

The receivers reduce RS-232 inputs (which may be as high as ± 25 V), to standard 5 V to 3 V TTL levels. These receivers have a typical threshold of 1.3 V, and a typical hysteresis of 0.5 V.

In Figure 5.8, we can see the connection diagram, to connect the TCM2.50 to the Waspmote board.

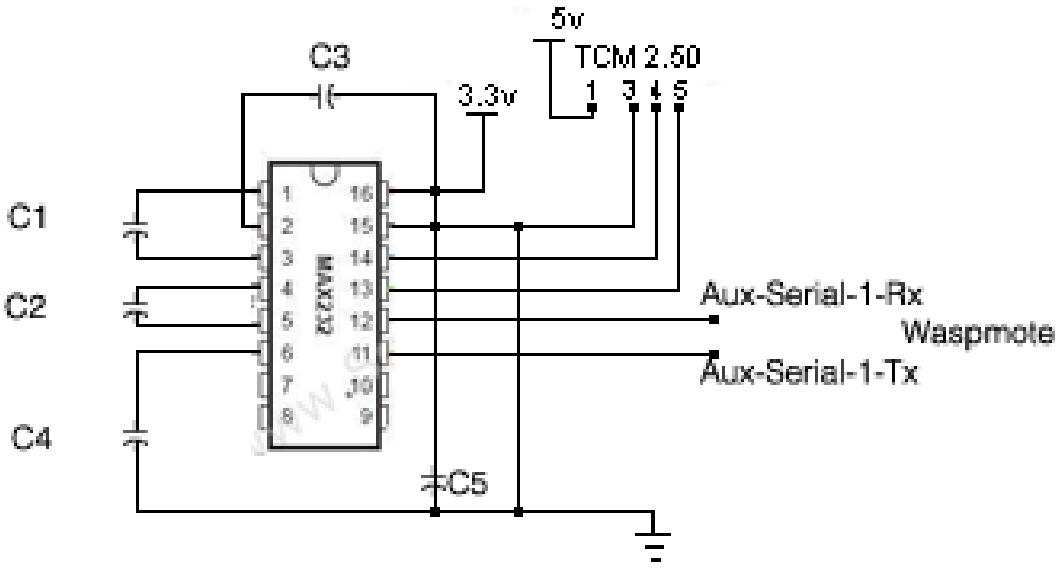


Figure 5.8: TCM adapter

5.4 GPS Receiver

The Waspmote board is prepared to accept an A1084, Figure 5.9, 20 channel GPS receiver with an external antenna. This receiver is based on the SiRF III chipset and supports the NMEA0183 [37] and SiRF binary serial protocols. We use the binary protocol to configure the receiver (elevation mask, signal strength mask, messages rates, ...) and rely on NMEA RMC and GGA messages for obtaining information about position, altitude, horizontal dilution of precision (hdop), ground speed, course and time. It has

a nominal accuracy of less than 10 meters. However, in our tests at sea, imposing elevation and signal strength masks, the accuracy has been quite stable and typically better that 3 meters. It is powered by the on board 3.3 V regulators and consumes 26 mA [38].

In the Table 5.1 we can see the main characteristics of the GPS system.

Table 5.1: GPS receiver features

-Model	A1084 (Vincotech)
-Movement sensitivity	-159dBm
-Acquisition sensitivity	-142dBm
-Hot Start time	<1s
-Warm Start Time	<32s
-Cold Start Time	<35s
-Antenna connector	UFL
-External Antenna	26dBi

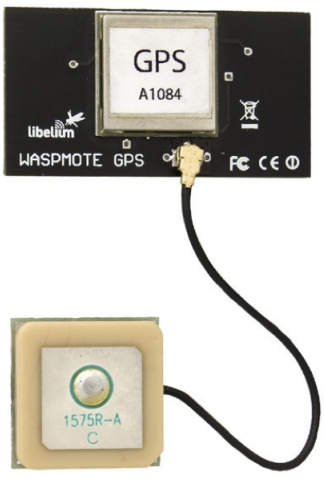


Figure 5.9: GPS receiver and external antenna

5.5 Wind Vane

The wind vane has been custom built from an Optimist wind vane attached to a US Digital's MA3 miniature absolute magnetic encoder [39] which is shown in Figure 5.10. The encoder is installed in an aluminum enclosure with a floating cap on top of the mast and connected to one of the analog inputs of the microcontroller, Figure 5.11. It allows to detect the direction of the apparent wind but not its speed. It works at 5 V and consumes 16 mA.

Figure 5.10: MA3 absolute angular encoder

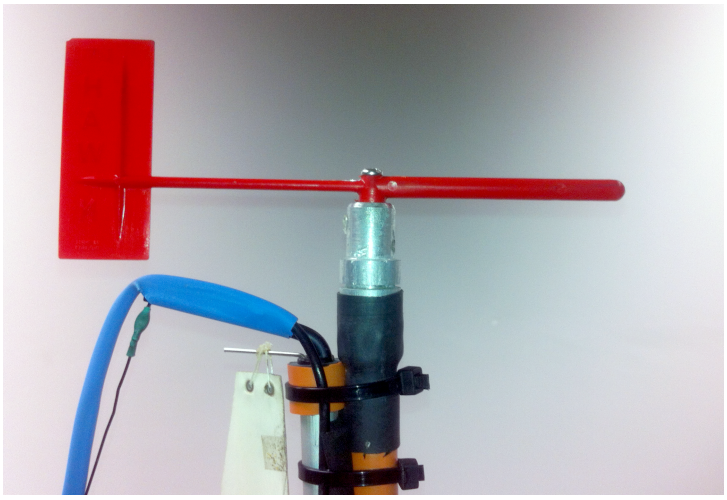


Figure 5.11: Home-made wind vane

5.6 Current Sensor

The current consumption at the actuator that control the sails' sheets is measured by means of an ACS712 board, Figure 5.12, [40]. The instantaneous current consumption is read as a voltage at a microcontroller's analog input. The ACS712 board integrates two potentiometers to adjust the intensity range being sensed and the acceptable levels of output voltage. This reading is used as an indirect measure of wind pressure in the sails. It is powered from 5 V and consumes 7 mA.

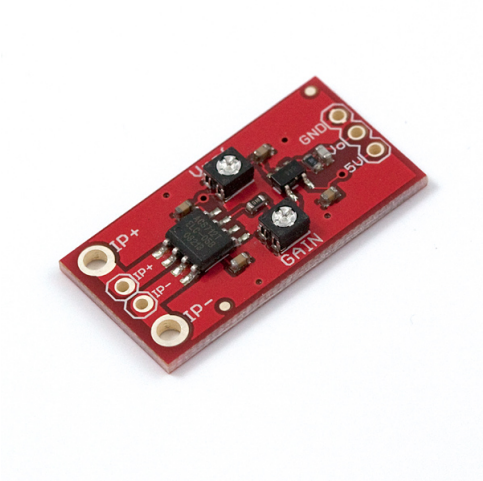


Figure 5.12: ACS712

5.7 Power Demands

A summary of power demands of the main components of the system, along with the capacity of both batteries, is detailed in Table 5.2.

Table 5.2: Power demands of system components.

Component	Volt(V)	Current(mA)	Power(mW)
Microcontroller	3.3	9	29.7
GPS	3.3	26	85.8
XBee 868 PRO	3.3	65-500	~ 330
TCM2.50	5	20	100
MA3 encoder	5	16	80
ACS712 board	5	7	35
		Electronics Total	660.5
	Electronics battery	3.7V - 6000mAh	22200 mWh

Component	Volt(V)	Current(mA)	Power(mW)
Rudder servo	5	10-500	~ 100
Sail winch	5	10-800	~ 500
		Actuators Total	600
	Actuators battery	7.4V - 2700 mAh	19980 mWh

Chapter 6

Control System

The system consists of two main parts: a base station, and a sailboat. The base station is a laptop equipped with a XBee USB adapter board used to communicate with the microcontroller onboard the sailboat over a 868 MHz RF link. On the sailbot, an onboard system treats the received data and commands the sailboat to reach the waypoints.

Both systems communicate regularly at a predefined but modifiable frequency. Using this radio link, the vessel can be monitored and controlled from the base station.

6.1 Base Station

The base station is a Linux application with a Qt [41] front end that relies on the libXBee [42] library to support the radio communications using XBee radio modules. Through the graphical user interface (GUI), just by clicking on a Google map (see Figure 6.1), it is possible to add, edit or delete sequences of waypoints to define a route for the sailboat.

The interface displays the telemetry data received from the sailboat with information about sailboat's state parameters, like bearing, speed or position, along with sensor readings. It is also possible to modify from the base

station some thresholds and parameters, like the frequency at which the telemetry packets are remitted or the minimum frequency at which the bearing selection function must be invoked. This capability has demonstrated its usefulness during sea trials.

The application can switch between autonomous or RC control modes. In this last control mode, a wireless game pad is used to control the rudder and the sails.

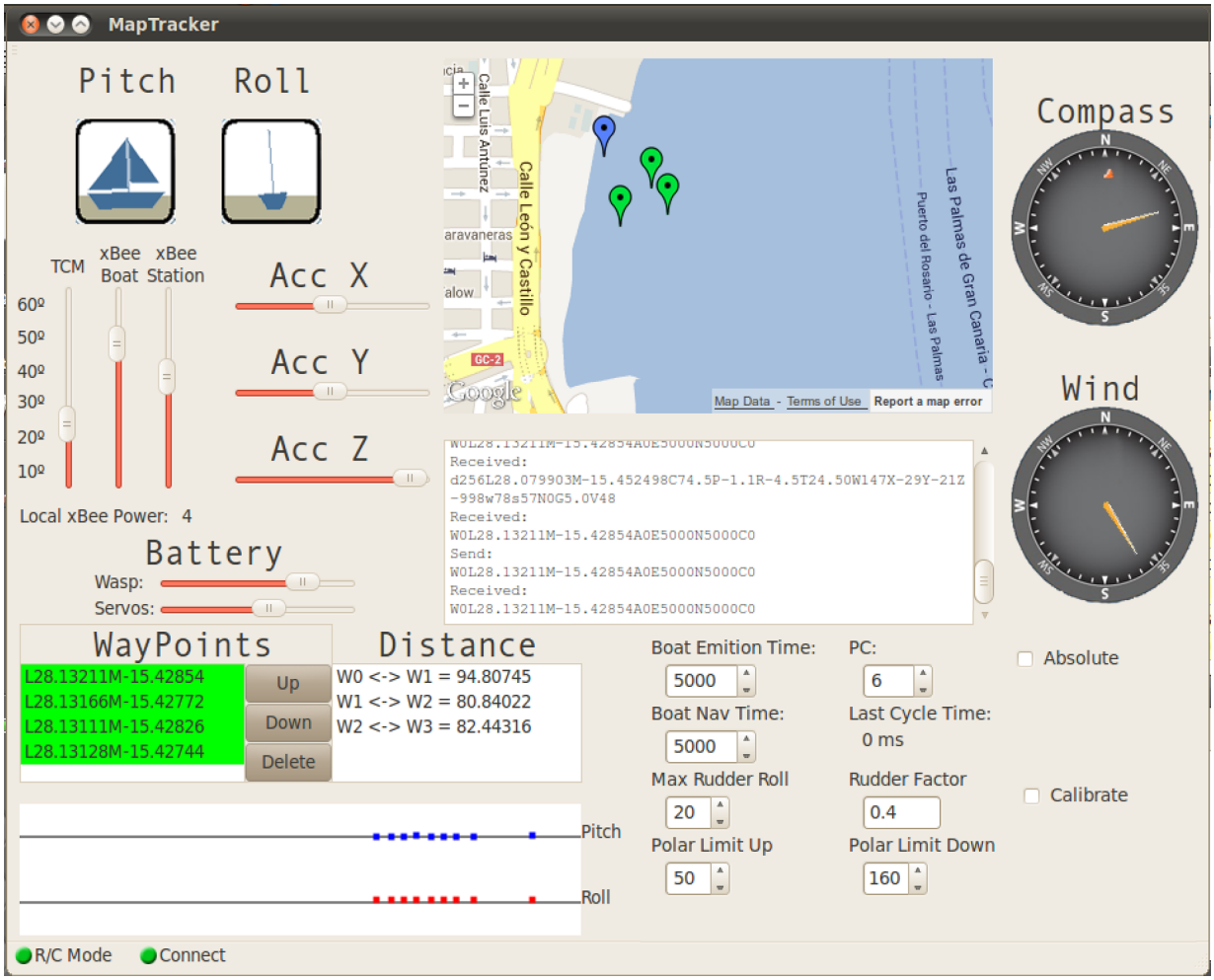


Figure 6.1: Base Station GUI

6.1.1 Base Station Hardware

To use the base station application we need a XBee gateway and a wireless game pad to communicate with the sailboat and to command the boat in RC control mode. As XBee gateway we have selected the one provided with the Waspmote.

Figure 6.2: XBee gateway

The selected wireless game pad was a Logitech WingMan Cordless.

Figure 6.3: Logitech WingMan Cordless

6.1.2 Software Architecture

The application receives and transmits information to the sailboat by means of the XBee gateway. Depending of the mode selected by the user, the application transmits different information to the sailboat. If the application is in RC control mode, every 500 milliseconds it transmits to the sailboat the commanded position of rudder and sails, but if it is in autonomous mode the base station resends the Home coordinates and some other parameters described on Table 6.2 every 2,5 seconds.

Depending on the working mode of the application, namely, RC control or autonomous mode, the system goes through the following stages:

- Initialize.
- Close.
- Update widget.
- Edit the waypoints list.
- Send command to sailboat.

This application uses the libXBee and the Sample Directmedia Layer library to support, respectively, the radio communications and the wireless game pad.

libXBee v1

LibXBee [42] is a C/C++ library that simplifies the usage of Digi XBee radios. It provides a friendly interface for creating connections, and sending and receiving data packets. Eventhough, this library is now in version 3, we have kept using version 1 because versions 2 and 3 were not backwards compatible. It is planned to update the base station software to the most recent version of this library.

Sample Directmedia Layer (SDL)

The Simple DirectMedia Layer [43] is a cross-platform multimedia library designed to provide low level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer. It is used by MPEG playback software, emulators and many popular games.

SDL supports Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2, but these are not officially supported.

SDL is written in C, but works with C++ natively, and has bindings to several other languages, including Ada, D, Eiffel, Erlang, Euphoria, Go, Guile, Haskell, Java, Lisp, Lua, ML, Oberon/Component Pascal, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk, and Tcl.

SDL is distributed under GNU LGPL version 2. This license allows you to use SDL freely in commercial programs as long as you link with the dynamic library.

In the sequel we detail the different stages of the Base Station operation.

6.1.3 Initialize

The initialization of the system it is done by $MainWindow :: MainWindow($ $QWidget * parent)$. This part of the application initializes the variables, starts the timers to update the widgets and send the commands, connects to the XBee module, starts the background thread that controls the wireless game pad and at the end it loads the Google map corresponding to the area of interest.

Firstly, the timers are configured to send data to the sailboat every 500 milliseconds and to update the widgets every 100 milliseconds. Then, the XBee radio is initialized and the function to connect to the XBee gateway is invoked, indicating the USB port where the gateway is connected.

The control of the wireless game pad is carried out with a background thread, using the QT class QFuture to start and check the wireless game pad.

At the end of the initialization phase, we load the html file, map.html, that controls the Google map and waypoints interface.

6.1.4 Close

The function $MainWindow :: MainWindow()$ stops the thread that controls the wireless game pad and the XBee communications, when the user closes the application.

6.1.5 Update Widget

The procedure $voidMainWindow :: updateWidget()$, every 100 milliseconds, checks if we have received any data packet from the sailboat.

First, it checks if we have received any packet from the sailboat during the last 15 seconds. If that is not the case, it changes in the interface to red the color of the connection state, and changes also the control mode to autonomous, because, on board, the sailboat changes automatically to

autonomous mode and activates the Return To Home behavior in these circumstances. If this is the case, the application also updates the LED light that shows on the GUI if the vessel is in autonomous or in RC control mode.

When a packet is received, the application displays the content of the message on the text browser and parses it to extract the data or interpret the message.

Received messages can be GPS connection message, a summary of the state of the sensors onboard or a confirmation reply to a new waypoint previously sent.

Telemetry messages are sent from the sailboat routinely. They contain a summary of on board's sensors readings and follow the following format:

$$\boxed{dxxCxxPxxRxxTxxWxxXxxYxxZxxwxxsxxNxxGxxVxx}$$

In Table 6.1 the meaning of the fields of the formate of telemetry packets is clarified.

To display the actual position of the sailboat, we call the $showNewMarker()$ function which, along with the $evaluateJavaScript()$ function, allows us to send javascript commands from QT to a javascript code in a html file.

Finally, the application checks if the user has introduced a new waypoint in the map, and in that case, the new waypoint's coordinates are sent to the sailboat using $sendXbee()$. Every time waypoints are updated, the Base Station recalculates the distances between consecutive waypoints and updates the GUI with the new distances.

Table 6.1: Format of telemetry messages

[d|D]xx	If it is in capital letters it means that the boat its in autonomous mode, and xx indicates the duration of the previous cycle in milliseconds.
Cxx	Indicates the compass value.
Pxx	Indicates the pitch value.
Rxx	Indicates the roll value.
Txx	Indicates the temperature value.
Wxx	Indicates the apparent wind position.
Xxx	Indicates X accelerometer value.
Yxx	Indicates Y accelerometer value.
Zxx	Indicates Z accelerometer value.
wxx	Indicates Waspmote battery value.
sxx	Indicates servos battery value.
Nxx	Indicates the index of the actual waypoint.
Gxx	Indicates the desired bearing.
Vxx	Indicates the temperature of the XBee radio module in Celsius degrees.

6.1.6 Editing The List Of Waypoints

The user has the possibility of performing some operations over the list of waypoints. It is possible to:

- Add
- Reorder
- Delete
- Resend

To add a new waypoint the user has to click on a place in the map where he or she wants to add the new waypoint. The map.html file will register the point and when $updateWidget()$ asks for new waypoint it will return the new point that will be sent to the sailboat.

The user can also reorder the list of waypoints doing click on Up or Down buttons, and the system will instruct the sailboat controller to reflect also the new arrangement of waypoints.

To delete a waypoint, the user has to select it and click on delete and the system will send the command to the sailboat to delete the waypoint.

If we want to resend a waypoint, we have to select it by clicking on it in the waypoints list and it will be resent to the sailboat. This it is useful because if there were any connection problem, it gives us the possibility to resend the waypoints that have not been acknowledged from the sailboat.

Next sections describe the commands that can be sent to the sailboat from the Base Station.

All the commands related with waypoints we will be detailed in the next section.

6.1.7 Messages Sent To The Sailboat

There are two different commands that can be sent to the sailboat, autonomous mode command or RC control mode command. These commands are prepared in $voidMainWindow :: sendCommBoat()$.

Autonomous mode command

The format of autonomous mode command messages is:

$$\boxed{WxxLxxMxxAxxExxNxxCxx}$$

Not all fields need to be set on the message, thus the format of this type of messages is flexible. Basically, it is used to fix a waypoint,command the sailboat to transit into autonomous mode, set some operational frequencies or combinations of these. The meaning of the fields are defined in table 6.2.

Table 6.2: Format of an autonomous mode message

Wxx	Indicates that this is an autonomous mode command and the position of the waypoint on the list that will be set on this message
Lxx	Indicates the longitude of a waypoint
Mxx	Indicates the latitude of a waypoint
Axx	Indicates how far (in meters) can be the sailboat off the line that connects the current position of the boat and the waypoint
Exx	Indicates the telemetry emission rate from the boat
Nxx	Indicates the execution rate of the navigation procedure
Cxx	Indicates if the on board inclinometers and wind vane zero must be recalibrated

In this mode, the base station sends every 5 seconds short messages to check the state of the radio link. If these packets are not received at the vessel for 20 seconds, the active waypoint is cleared and substituted by the coordinates that identify the "Home Point" and the sailboat tries to arrive to that point autonomously. This constitutes the "Return To Home" or RTH behavior that has proved a valuable capability during sea tests. This situation can be reverted from the base station as soon as the radio link is reestablished. In that moment, new waypoints and navigation parameters can be transmitted to the boat.

RC Control command

The format of RC control commands is:

$$\boxed{Crxxsxx}$$

An RC control command command is sent every 500 milliseconds from the base station to the vessel if the system is in radio control mode. This

message specifies the position of the rudder in degrees and the percentage in which the sails must be tightened, as detailed in Table 6.3.

Table 6.3: Format of a control message

C	Indicates that it is a control command
rxx	Indicates the desired rudder position
sxx	Indicates the desired sails position

6.1.8 Use Cases

This section defines the different cases of usage of the base station.

Name	Create waypoint	Id	UC1
Actor			
-User			
Description			
-Creates a new waypoint, displays it on the map, and sends it to the sailboat.			
Trigger			
-To do click on a place on the map			
Precondition			
-The map has been loaded -The XBee gateway has been connected			
Result			
-The waypoint is displayed on the map -The new waypoint is sent to sailboat along with its position on the list			
Flow			
-Do click on the map -Update map.html and save the waypoint -Wait to load the waypoint from map.html to QT -Create the message -Send the message			

Table 6.4: Create waypoint

Name	Up waypoint	Id	UC2
Actor			
-User			
Description			
-Set the selected waypoint one position upper.			
Trigger			
-To do click on up			
Precondition			
-One waypoint of the list has been selected. -The selected waypoint can not be the first one.			
Result			
-The waypoint will be set one position up and. the upper waypoint one position down.			
Flow			
-Do click on the up -The waypoint selected is set one position up -The upper waypoint will set one position down -The list will be updated -Send the two modified waypoints			

Table 6.5: Up waypoint

Name	Down waypoint	Id	UC3
Actor			
-User			
Description			
-Set the selected waypoint one position down.			
Trigger			
-To do click on up			
Precondition			
-One waypoint of the list has been selected. -The selected waypoint can not be the last one.			
Result			
-The waypoint will be set one position down and the bottom waypoint one position up.			
Flow			
-Do click on the up -The waypoint selected it set one position up -The bottom waypoint will set one position down -The list will be updated -Send the two modified waypoints			

Table 6.6: Down waypoint

Name	Delete waypoint	Id	UC4
Actor			
-User			
Description			
-Delete the selected waypoint.			
Trigger			
-To do click on delete			
Precondition			
-One waypoint of the list has been selected.			
Result			
-The waypoint will be deleted.			
Flow			
-Do click on delete -The selected waypoint is deleted -The list will be updated -Send to delete the waypoint selected			

Table 6.7: Delete waypoint

Name	Send autonomous message	Id	UC5
Actor			
-User -Application			
Description			
-Send an autonomous mode message to the sailboat. -It will be sent if the user clicks on a waypoint on the list or every 2,5. seconds.			
Trigger			
-Every 2,5 seconds -Do click on a waypoint			
Precondition			
-To be in autonomous mode -The waypoint have to exist			
Result			
-The waypoint will be sent to the sailboat and the vessel will answer with a confirmation of reception.			
Flow			
-Do click on the waypoint or reach the timeout of 2.5 seconds -Configure the message format $WxxLxxMxxAxxExxNxxCxx$ -Send the waypoint -If message is confirmed, set the waypoint as received			

Table 6.8: Send autonomous message

Name	Send control message	Id	UC6
Actor			
-Application			
Description			
-Send to the sailboat a control message. -It will be sent every 0,5. seconds.			
Trigger			
-Every 0,5 seconds			
Precondition			
-To be in control mode			
Result			
-The control message will be sent to the sailboat			
Flow			
-Reach the timeout of 0,5 seconds -Set the package format $Crxxsxx$ -Send the command			

Table 6.9: Send control message

Name	Receive sensors message	Id	UC7
Actor			
-Application			
Description			
-Receives a message from the sailboat, parses the message and display the content.			
Trigger			
-Receive something from the sailboat. -The message must begin with D or d.			
Result			
-The message will be parsed and GUI will be updated			
Flow			
-Receive a message. -Detect if the message begins with D or d. -Parse the message $dxxCxxPxxRxxTxxWxxXxxYxx$ $ZxxwxxsxxNxxGxxVxx$ -Display the information			

Table 6.10: Receive sensors message

Name	Calibrate sensors	Id	UC8
Actor			
-User			
Description			
-To calibrate the wind vane, pitch and roll sensors			
Trigger			
-Check the calibration option.			
Result			
-Sensors will be calibrated.			
Flow			
-To set the calibration option. -The next autonomous message will send the calibration command. -When the sailboat receives the command it will calibrate the sensors -The calibration will be stored on the EEPROM			

Table 6.11: Calibrate sensors

Name	Set sailboat emission rate	Id	UC9
Actor			
-User			
Description			
-To set emission rate of the sensors message			
Trigger			
-Set the emission rate option.			
Result			
-The emission rate will be set on the sailboat.			
Flow			
-To set the emission rate. -The next autonomous message will send the emission rate command. -When the sailboat receive the command it will set the emission rate			

Table 6.12: Set sailboat emission rate

Name	Set navigation execution rate	Id	UC10
Actor			
-User			
Description			
-To set navigation execution rate			
Trigger			
-Set the navigation execution rate option.			
Result			
-The navigation execution rate will be set on the sailboat.			
Flow			
-Set the emission rate. -The next autonomous message will send the navigation execution rate command. -When the sailboat receives the command it sets the navigation execution rate			

Table 6.13: Set navigation execution rate

Name	Set sailboat mode	Id	UC11
Actor			
-User			
Description			
-Sets the sailboat mode			
Trigger			
-Push C on the wireless game pad.			
Result			
-The sailboat will change the mode.			
Flow			
-Push C on the wireless game pad. -Sets the station base application in the opposite mode than the actual mode. -Starts to send messages in the selected mode.			

Table 6.14: Set sailboat mode

6.2 Sailboat Control System

The on board system communicates with the base station through the exchange of messages. Messages are parsed to detect if it is a remote control or an autonomous mode message. If the system is in autonomous mode, given the actual position, the waypoint coordinates, the actual heading and the instantaneous wind direction it selects the best bearing to reach the waypoint goal.

6.2.1 Software Architecture

The on board software has been developed modularly and all modules have been first developed and tested in isolation. Because the development environment lacks any kind of debugging facilities we have designed a basic methodology to tackle this problem. It is based of defining a set of compilation directives to select which parts of the code are to be compiled and in this way "tune" the on board control code to different debugging condi-

tions. An additional set of compilation directives allows to select the level of "verbosity" of modules' debugging outputs.

The next table describes all the compilation directives and how they configure the on board control program. Obviously, some possible configurations are useless in practice as the control program demands all modules to be included.

GPS	If defined, GPS support will be included. Otherwise, the GPS receiver will not be used and the sailboat will be always on the same position. (Latitude 0. Longitude 0)
TCM	If defined, the code related to the TCM2.50 sensor will be included. Otherwise, the TCM board will not be operative and the sailboat will have always the same value of pitch, roll, compass and temperature, all of them 0
XBEE	If defines, the RF XBee module will be operative. Otherwise, the radio will not operative and the sailboat will not be able to communicate with the base station.
SD	If defined, support for operating the SD card will be included. Otherwise, the micro SD card will not be usable and logging will be canceled.
DEBUG SENSOR	This directive permits debugging the code related with on board sensors. Output is displayed on the serial monitor through the USB connection.

DEBUG NAV	This directive permits debugging the on board navigation algorithms. Output is displayed on the serial monitor through the USB connection.
DEBUG XBEE	This directive permits debugging the code related with radio communications. Output is displayed on the serial monitor through the USB connection.
DEBUG SD	This directive permits debugging the code related with SD logging services. Output is displayed on the serial monitor through the USB connection.
DEBUG TELE	This directive permits debugging the telemetry packets. Output is displayed on the serial monitor through the USB connection.
DEBUG TIME	This directive activates the profiling sentences embedded in the code to know the time consumed in the main blocks of the control code. Measurements are displayed on the serial monitor.
DEBUG TIME SD	This directive is complementary of the DEBUG_TIME directive and, when defined, it makes the profiling data to be saved on the SD card.

DEBUG MEM	This directive activates memory checking functions that are invoked at selected locations in the code to assure memory consistency. Information gathered is displayed on the serial monitor.

The control application is structured around the following modules:

- Initialization.
- Fuzzy Logic System.
  - Rudder Control.
  - Sails Control.
- Control loop.
  - Remote control mode.
  - Autonomous mode.
- Robust radio connectivity.
  - Send.
  - Receive
- Sensor sampling.
- SD Card
  - Full sensors message.
  - Timing measurements message.

The flowchart of the software on board is detailed on Figure 6.4.

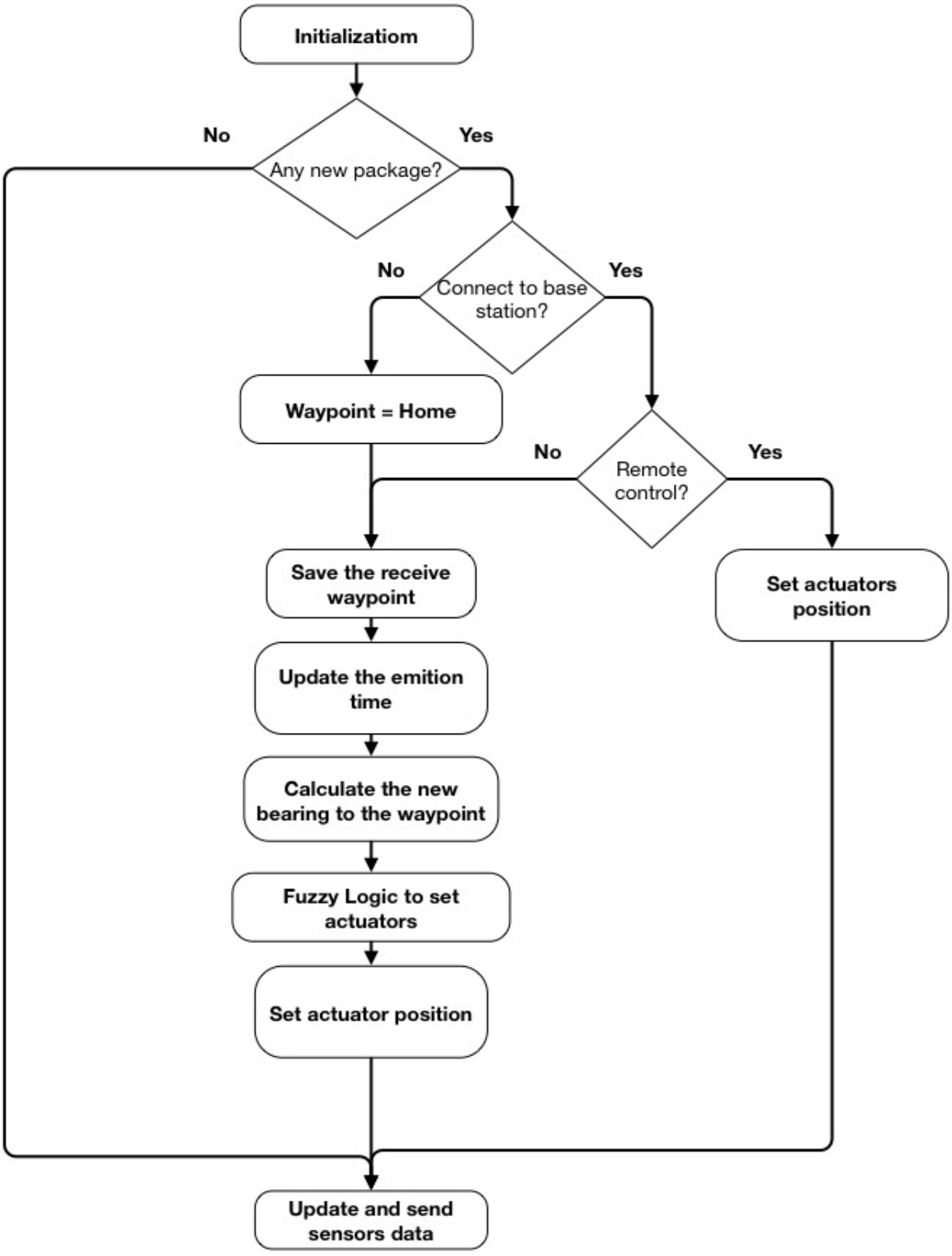


Figure 6.4: On Board Flowchart

6.2.2 Initialization

The initialization of the system is carried out normally with the vessel at shore, but could be done remotely as far as the radio link is possible. In this phase, the operational state of all onboard subsystems are verified and some on board sensors may be calibrated, namely the wind vane and the inclinometers. The calibration steps can be omitted using the base station interface.

First radio communication, SD logging and battery levels are checked and afterwards on board power regulators are switched on. Then the GPS receiver is configured and the elevation and signal strength masks are configured in order to minimize noise in GPS readings. Once the GPS is configured, a first valid fix is awaited and then it will wait 30 additional seconds to stabilize the GPS measurements. Finally, the fuzzy logic control system is initialized.

An optional final stage in initialization deals with the calibration of some sensors offsets. It requires to keep the sailboat in a horizontal position with 0^o of pitch and roll and the wind vane pointing forward. This is done only once at the beginning of the experiment but can be avoided if previously calibrated offsets are valid. These offsets are stored at fixed addresses in EEPROM.

At the conclusion of this stage the sailboat will be in remote control mode and it will start sending telemetry data through the radio every 5 seconds by default.

6.2.3 Fuzzy Logic System

The sailboat's actuators, rudder servo and sails winch, are controlled using a fuzzy control system[23]. Setting this kind of controller involves defining the control vocabulary, and associated membership functions, for inputs and outputs and the rules to control the rudder and the sails. The fuzzy

controllers used in the A-TIRMA has been inspired in those described in [23], even though they differ in important aspects.

Rudder Control

The input data for the rudder control circuit are the current boat direction and the desired direction from the routing system. The difference between these two gives the necessary course correction which enters directly into the fuzzy system as input variable, Figure 6.5. Also in order to avoid over-steering we use the position of the rudder, Figure 6.6, as an additional input variable. Finally we have a output fuzzy set with the differents posible positions of the rudder, Figure 6.7.

The definition of membership functions have been adapted from [23] to accommodate the characteristics of our sailboat. Our fuzzy sets for the rudder control are:

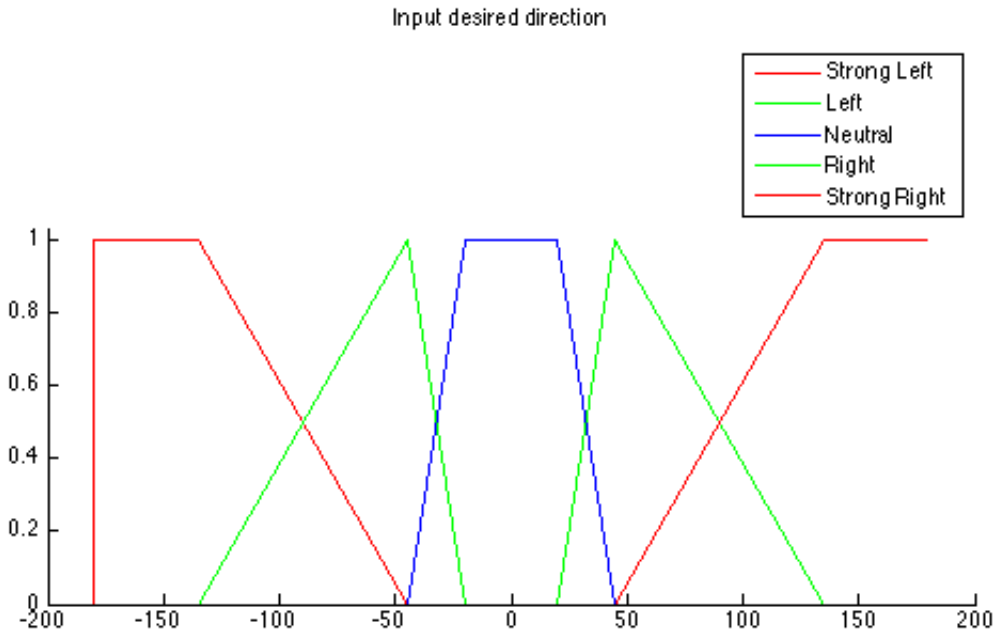


Figure 6.5: Fuzzy input: Desired heading

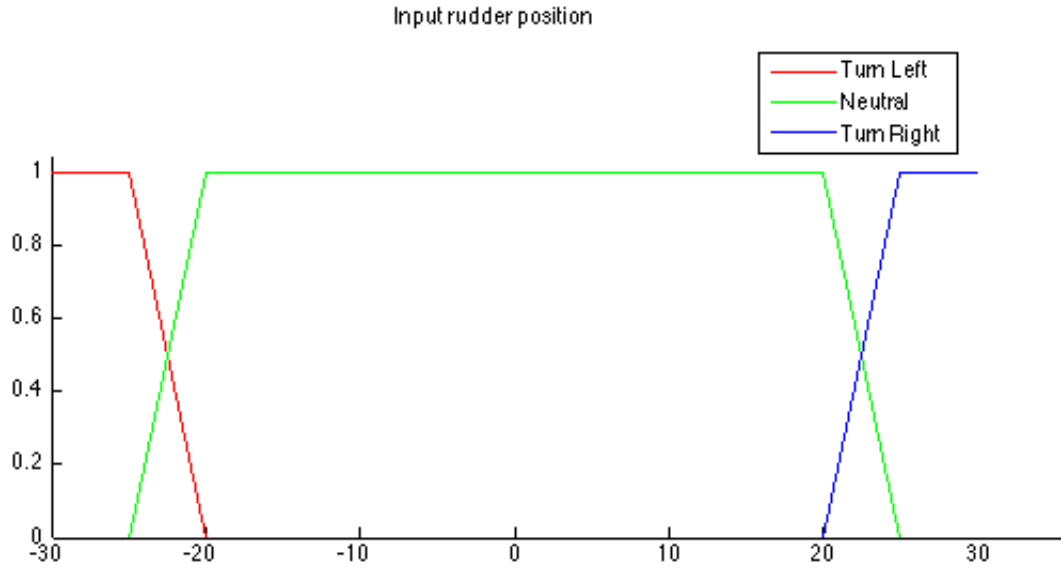


Figure 6.6: Fuzzy Input: Current rudder angle

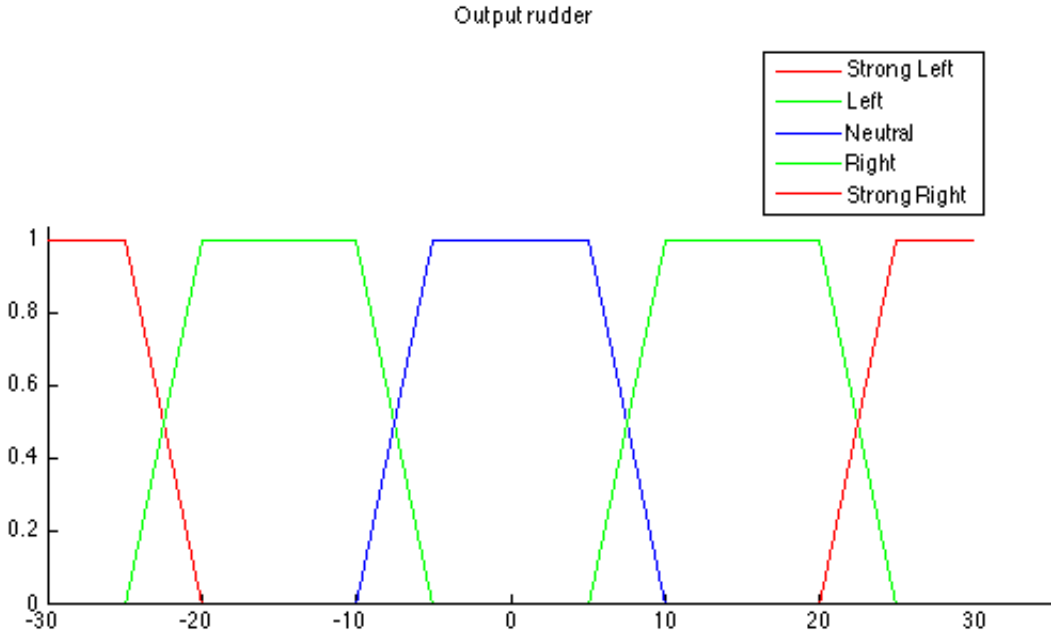


Figure 6.7: Fuzzy output: Commanded rudder angle

The rules implemented were:

If Direction Strong Left And Turning Left **Then** Rudder Change Left
If Direction Strong Left And Turning Neutral **Then** Rudder Change Strong Left
If Direction Strong Left And Turning Right **Then** Rudder Change Strong Left
If Direction Left And Turning Left **Then** Rudder Change Keep
If Direction Left And Turning Neutral **Then** Rudder Change Left
If Direction Left And Turning Right **Then** Rudder Change Strong Left
If Direction Middle And Turning Left **Then** Rudder Change Right
If Direction Middle And Turning Neutral **Then** Rudder Change Keep
If Direction Middle And Turning Right **Then** Rudder Change Left
If Direction Right And Turning Left **Then** Rudder Change Strong Right
If Direction Right And Turning Neutral **Then** Rudder Change Right
If Direction Right And Turning Right **Then** Rudder Change Keep
If Direction Strong Right And Turning Left **Then** Rudder Change Strong Right
If Direction Strong Right And Turning Neutral **Then** Rudder Change Strong Right
If Direction Strong Right And Turning Right **Then** Rudder Change Right

Also to avoid problems with overheeling, we check if the heeling of the vessel it higher than a maximum threshold. If it is higher then the system apply a correction to the rudder to compensate the sailboat's tendency to haul upon the wind due to excessive heeling.

Sails Control

The inputs for sail control are the desired point of sailing, Figure 6.8, and the roll angle, Figure 6.9. The desired point of sailing, determined by the wind direction for the established bearing, defines the position of the sail, even though this position is adjusted depending on the roll angle to avoid excessive heeling.

As output we have five positions of the sail, Figure 6.10.

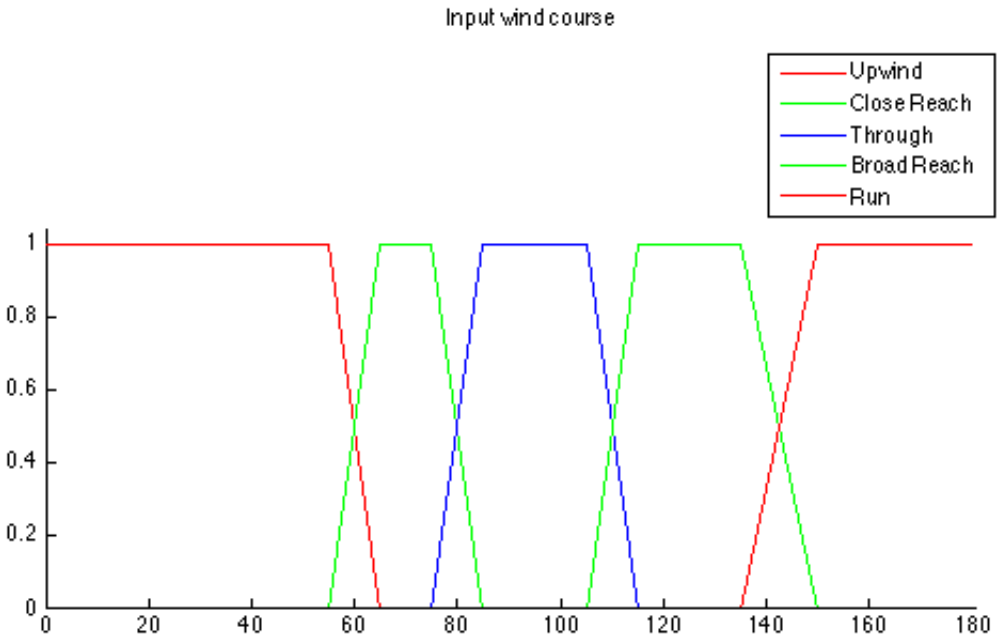


Figure 6.8: Fuzzy input: Point of sailing

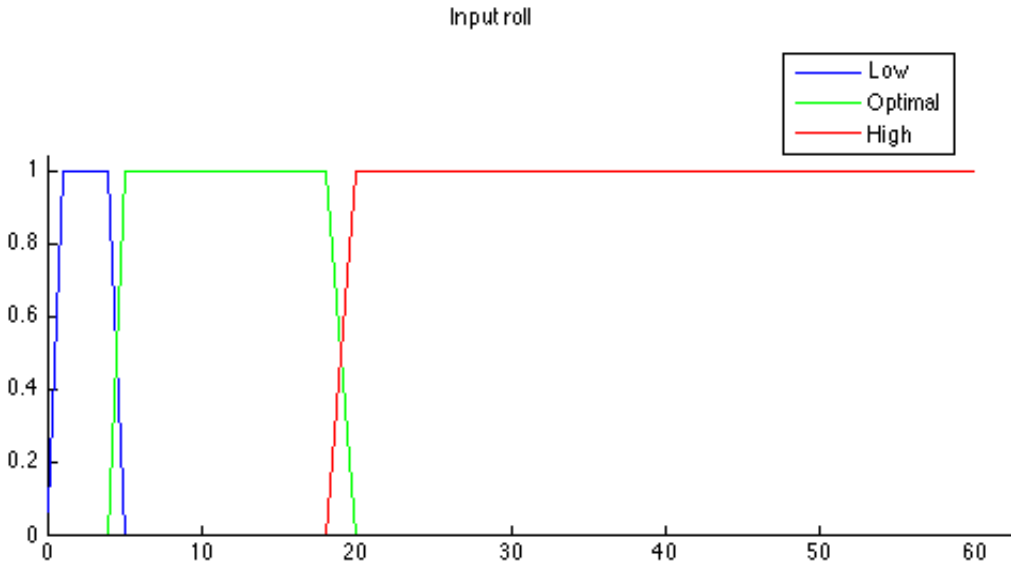


Figure 6.9: Fuzzy input: Roll angle

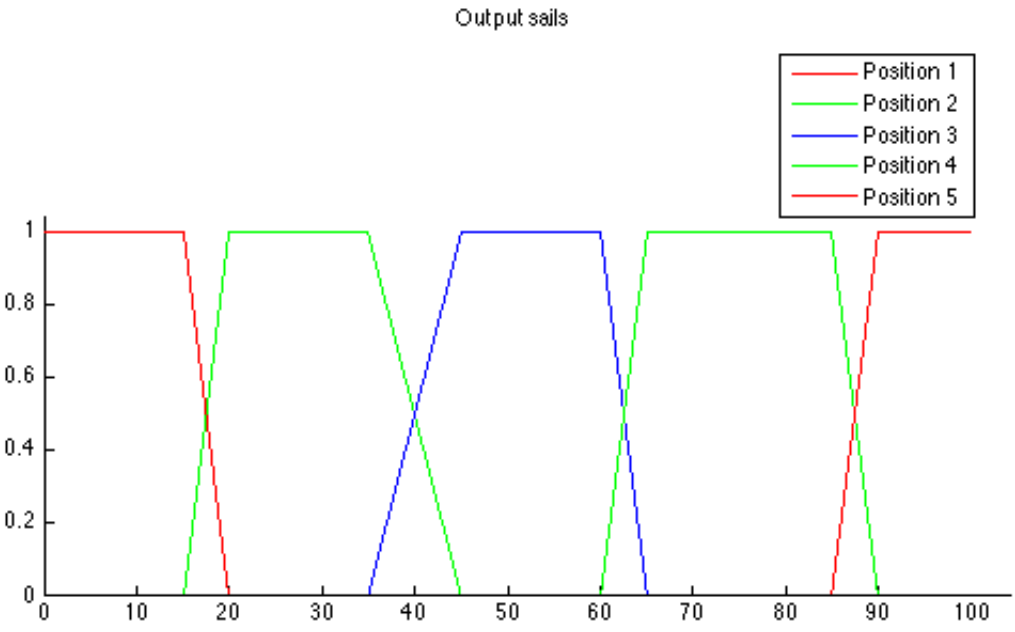


Figure 6.10: Fuzzy output: Sail position

The rules implemented were:

If Upwind And Heel Too Low **Then** Sail position 1
If Upwind And Heel Optimal **Then** Sail position 1
If Upwind And Heel Too High **Then** Sail position 2
If Close Reach And Heel Too Low **Then** Sail position 1
If Close Reach And Heel Optimal **Then** Sail position 2
If Close Reach And Heel Too High **Then** Sail position 3
If Through And Heel Too Low **Then** Sail position 2
If Through And Heel Optimal **Then** Sail position 3
If Through And Heel Too High **Then** Sail position 4
If Broad Reach And Heel Too Low **Then** Sail position 3
If Broad Reach And Heel Optimal **Then** Sail position 4
If Broad Reach And Heel Too High **Then** Sail position 5
If Run And Heel Too Low **Then** Sail position 4
If Run And Heel Optimal **Then** Sail position 5
If Run And Heel Too High **Then** Sail position 5

6.2.4 Control loop

Once the initialization has been accomplished, the main control loop will proceed in any of two possible modes of operation. In the autonomous mode the sailboat's control system selects the best bearing to arrive to the active waypoint. Alternatively, the remote control or teleoperation mode permits to take full control of the sailboat from the base station. In both modes the telemetry is kept active. Figure 6.11

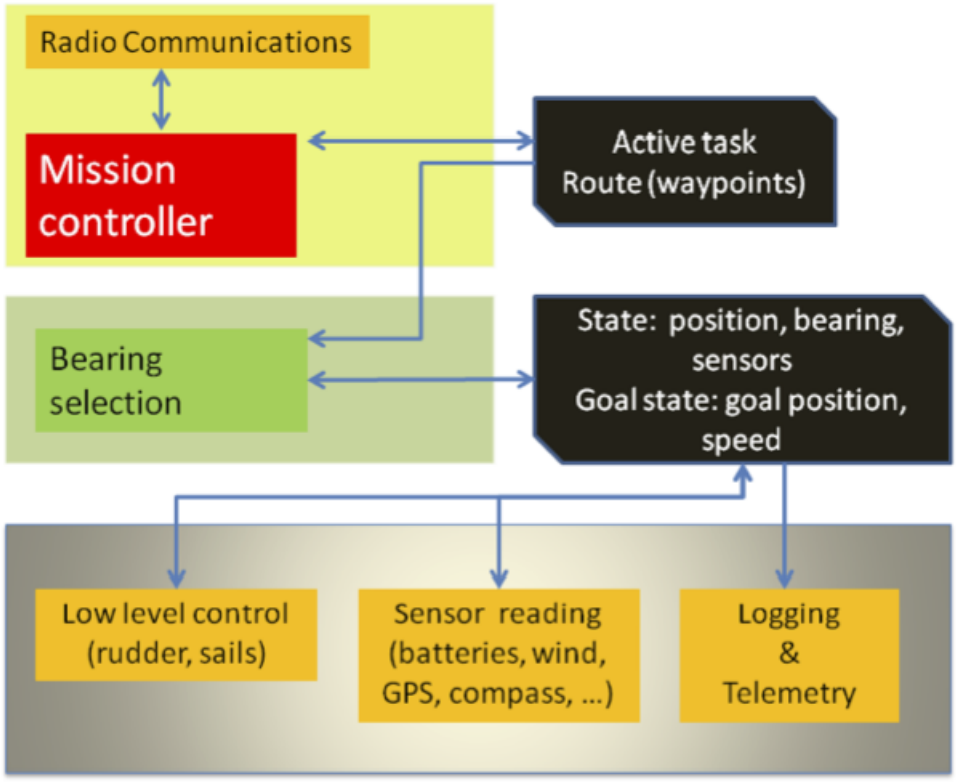


Figure 6.11: Control architecture

Remote Control Mode

While the boat is in remote control mode it obeys the sail and rudder position commands sent from the base station using a wireless game pad connected to the laptop running the base station application. In this mode, short radio packets are sent to the vessel at a frequency of 5 Hz. The transmission rate can not be too high because in this mode, the on board sensors are sampled, logged and telemetry packets are sent at the specified frequency to the base station.

As we mentioned before on Section 6.1.7, control messages follow the format $\boxed{Crxxsxx}$, with xx denoting the commanded rudder and sail posi-

tions. The control program on board monitors the reception of these packages several times per control cycle and, if received, they are parsed and the servos commanded.

Autonomous Mode

In the autonomous mode the navigation is fully under the control of the on board microcontroller. The control is organized around three levels of control. At the highest, the route controller simply manages the list of waypoints that defines a route and selects the active waypoint. When the sailboat is inside the radius of precision of the waypoint, the route controller will change the active waypoint to the next one in the route. The list of waypoints is treated as a cyclic route by default, so when the last waypoint is reached, it will start again with the first one and the route is repeated.

The list of waypoints has six positions. The first one, the zero position, gives the location of the Home waypoint. The other five points are the waypoints that define the route and are used by the bearing selection algorithm.

The bearing selection algorithm [30] is used at the next control level to obtain the best (i.e. fastest) bearing to reach the active waypoint, given the current wind direction, boat position and heading. This control level runs at an adjustable frequency but can be triggered also by a sudden wind roll. Note that as far we lack an estimation of wind speed on board, we run this algorithm using only the apparent wind. The algorithm reacts in real-time to wind changes, like a sailor does.

The parameters needed for the calculation of the desired boat heading are:

- Target position
- Current boat position

- Current boat heading
- Apparent wind direction

The bearing selection algorithm is the slowest function on the on board system, roughly 150 milliseconds, and it does not make sense to calculate the goal bearing on every iteration, that is, each 250 milliseconds. Thus, the bearing selection function is run by default every five seconds, although it can be modified using the base station application, or if the wind direction has changed. Note also that the execution of the bearing selection function will be delayed if the sailboat is turning in that moment because wind vane readings would be unreliable.

The bearing selection algorithm described in [30] is based on absolute wind direction. In our sailboat we only have a wind vane, so we only have the apparent wind. Figure 6.12 illustrates the difference between the true wind and the apparent wind in the case where the sailboat speed is 1m/s and the wind speed is 3m/s.

Considering the apparent wind direction as the absolute wind direction provokes that the boat will never sail as close hauled as it could as determined by its polar. In downwind courses, the situation is just the opposite: the vessel can go into a too closed downwind course inadvertently.

At the lowest level, the fuzzy controller described previously (see 6.2.3) runs at the highest frequency of the control system and controls the sail and rudder positions to keep the sailboat under control on the desired bearing. This controller has been implemented using the EFLL library [45].

The sailboat can transit into autonomous mode if a prolonged failure of radio communications is detected or because this control mode is explicitly commanded from the base station through a radio packet with the format $\boxed{WxLxxMxxAxxExx}$. The preamble W identifies this packet as an autonomous mode command packet; the L and the M fields indicate the latitude and the longitude of the active waypoint; A indicates how far (in

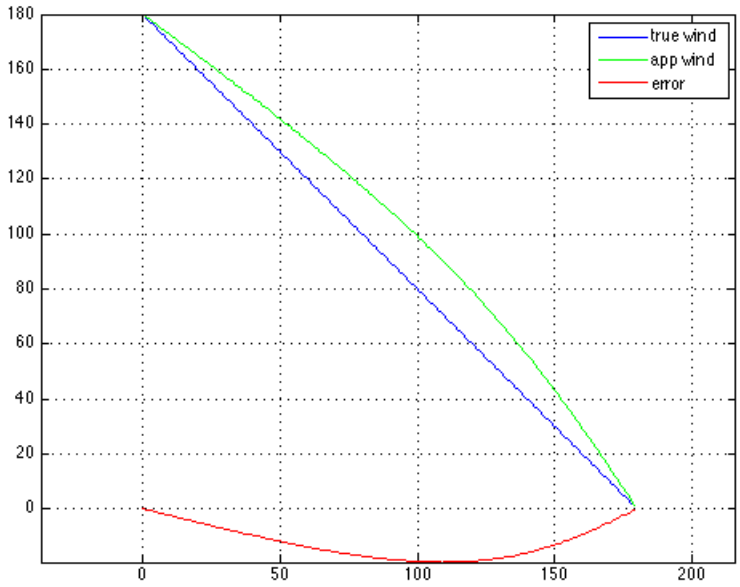


Figure 6.12: Apparent wind error

meters) can be the sailboat off the line that connects the current position of the boat and the waypoint (it is equivalent to the PC parameter in [30]), finally, E indicates the emission period for telemetry messages.

When the vessel is in this mode, the base station sends every 5 seconds short messages to verify the radio link. If these packets are not received at the vessel for 20 seconds, the active waypoint is deactivated and substituted by the coordinates that identify the "Home Point" and the sailboat will try to arrive to that point autonomously. This constitutes the Return To Home" or RTH behavior that has proved a valuable capability during field tests.

6.2.5 Robust radio connectivity

Loss of radio connectivity is something that may happen during sailing due to a variety of reasons and it is important to endow the sailboat with some recovery and continuity strategies to deal with these situations. In order to

increase the robustness of radio communications on this uncertain scenario, the XBee radios are used in API mode and all exchanged messages have been limited in extension to make them fit within the payload of XBee API frames. Basically, this constraint reduces the complexity of recovering partially lost packets as all messages involve a single radio frame. The payload size is 100 bytes, so we have to adjust the data transmitted in packages to not exceed this limit.

Accordingly, at the lowest level, the XBee radio modules have been programmed to resend automatically dropped or incorrect radio packets for a number of times. The loss of telemetry packets is not critical because they are still logged on the micro SD card available on board. More critical is the loss of command packets sent from the base station and this is why these packets must be acknowledged explicitly from the sailboat.

Send

The sailboat sends to types of packages, telemetry packages and waypoint reception acknowledgment packages. Telemetry messages have the format that described previously in Table 6.1. Waypoint acknowledgment packages are just copies of the message received from the base station and described in Table 6.2.

Receive

When the sailboat receives a package, it is parsed and the message type detected. Unrecognized messages are silently ignored. The vessel receives two different messages:

- Autonomous mode packages. Described on Table 6.2
- Remote control mode packages. Described on Table 6.3

6.2.6 Sensor Sampling

Sensors are sampled at different rates depending on its potential rate of change and the temporal cost of a new reading in order to reduce the mean sampling time and hence, the duration of a control cycle.

The GPS sensor available on board has a maximum update rate of 1 Hz and it does not make sense to read it faster because it will deliver old estimates. Even more, while reading at the nominal rate of 1 Hz, timestamps of new readings must be checked against the timestamp of the last delivered message to verify that it is indeed a new reading. If that is not the case, a new reading is attempted.

The compass board is programmed to produce a continuous flow of measurements at a default frequency of 10 Hz. This approach reduces the cost of reading from the TCM board. Each data packet contains the compass, pitch and roll angles, the temperature of the board and, eventually, an error code. Commonly, error codes appear when the magnetometers have become saturated or the pitch and roll angle limits have been exceeded. In those cases, these measurements are discarded.

TCM packets may accumulate and overflow the microcontroller serial buffer if it is not read fast enough. This is not a problem because the serial port buffer is has been designed as a circular buffer. At the same time, when new measurements are read from the serial buffer, all messages but the last one are discarded.

It is important to know that the GPS receiver and the compass are connected to two different serial ports that are in fact multiplexed on the same microcontroller's UART. This implies that it is not possible to receive continuously and simultaneously data from both devices. Our solution is to read first the TCM 2.50 sensors board and then flush the serial port and set the multiplexer to read from the GPS receiver.

The navigation is critically dependent on the adequate sampling rate of

the set of on board sensors. With the limited computing power available and high temporal cost of sampling some sensors, a multirate, smart sampling strategy is necessary. Sensors like the wind vane and the compass have a high update rate and the reading cost is very small. On the other side, GPS has a low update rate and interrogating the GPS receiver takes about 60 msecs. To deal with this situation, fast sensors, and in particular the wind vane, are sampled several times within a single control cycle and filtered to produce better estimates of these magnitudes.

Sensors readings are monitored and they may trigger some alarms. For example, a sudden roll in wind direction over a predefined threshold will trigger the execution of the bearing selection algorithm during the next control loop. If the vessel is turning we do not execute the bearing selection algorithm. Also, battery levels are checked against low level thresholds and if low battery alarms are triggered they are notified to the base station.

With the current hardware, the temporal cost of executing one control cycle is dominated by the temporal cost of the actions carried out during one control cycle. The subtasks that have the higher temporal cost are reading the GPS (60 milliseconds) and SD logging (60 milliseconds). Taking into account that some subtasks do not execute in every cycle, the shortest cycle time takes approximately 160 milliseconds and the largest 500 milliseconds.

6.2.7 SD Card

The micro SD card is used to log all data collected and generated by the sailboat. This information give us the opportunity to record the state of the vessel on each control iteration. The system store in the SD card two types of messages. A full sensor message and a timing measurements message.

The SD card specifications:

- FAT16

- Allocation table size of 16 bits.
- 4GB
- HC cards are compatible.

Full Sensors Message

The format of this message it is an extension of the one detailed on Table 6.1.

The structure of a telemetry or sensor data message is:

[d|D]xx	If it is in capital letters it means that the boat its in autonomous mode, and xx indicates the duration in milliseconds of the last control cycle.
Cxx	Indicates the compass value.
Pxx	Indicates the pitch value.
Rxx	Indicates the roll value.
Txx	Indicates the temperature value.
Wxx	Indicates the apparent wind position.
Xxx	Indicates X accelerometer value.
Yxx	Indicates Y accelerometer value.
Zxx	Indicates Z accelerometer value.
wxx	Indicates Waspmote battery voltage.
sxx	Indicates servos battery voltage.
Nxx	Indicates the index of the active waypoint.
Gxx	Indicates the desired bearing.
Vxx	Indicates XBee temperature value.
Hxx	Indicates the active waypoint latitude.

Axx	Indicates the active waypoint longitude.
Ixx	Indicates winch intensity.
Sxx	Indicates sailboat speed as estimated by the GPS receiver.
Uxx	Indicates speed calculated from GPS coordinates.
Jxx	Indicates distance calculated to the previous point.
Fxx	Indicates azimuth.
Kxx	Indicates GPS time.
Qxx	Indicates milliseconds from the initialization.
axx	Indicates rudder position.
bxx	Indicates sails position.

Timing Measurements Message

Timing measurements message contains the execution cost in milliseconds of selected parts of the on board control system. The structure of this type of messages is as follows:

Dxx	Cost in milliseconds of the full cycle.
axx	Cost in milliseconds of the first check for incoming packets.
bxx	Cost in milliseconds of invoking *checkTimeOut*().
cxx	Cost in milliseconds of invoking the bearing selection algorithm.

dxx	Cost in milliseconds of reading all sensors.
fxx	Cost in milliseconds of checking a second time for incoming packets.
gxx	Cost in milliseconds preparing a telemetry packet.
hxx	Cost in milliseconds of sending a message.
ixx	Cost in milliseconds of logging a data packet.
jxx	Cost in milliseconds of obtaining a new reading from the TCM2.50 board.
kxx	Cost in milliseconds of obtaining a new GPS reading.
lxx	Cost in milliseconds of some numerical computations.
mxx	Cost in milliseconds of reading the wind vane, accelerometers and batteries.
nxx	Cost in milliseconds of reading the wind load current sensor.

As commented previously, all collected sensor data are packed and logged on board on the micro SD, but only a fraction of the logged packet is transmitted to the ground station as a telemetry packet at a predefined frequency.

Chapter 7

Results

The data collected across several sea trials will be used in this chapter to evaluate the behavior of the sailboat under different circumstances. It will also rise the opportunity to discuss the evolution of the control code and its timing performance. We will base some of the tests on tasks proposed for the 2013 edition of the WRSC (World Robotic Sailing Championship) [47].

7.1 Code Improvements

7.1.1 Timing

As we described before on Section 6.2.7, during sea trials we have recorded the temporal cost of executing specific blocks of the code. In a first stage that information was used as profiling data to improve the code and minimize the time spent in slow functions. In Figure 7.1 we can see a non-optimized time code. It should be recalled here that our system is a 8 MHz 8 bits microprocessor with only 8 KB of RAM memory. When we developed the code we did it knowing the capabilities of our system, trying to do a fast code with low memory requirements.

It's worth commenting also that neither the development environment nor the avr-gcc compiler offers any kind of profiling support. Hence, timing

data have been obtained inserting timing functions in the code, as explained in detail in the previous chapter. This approach certainly degrades full cycle measurements, like the cycle time, because these measurements will be distorted by the cost of invoking the timing functions themselves, but it is to our knowledge the only viable approach.

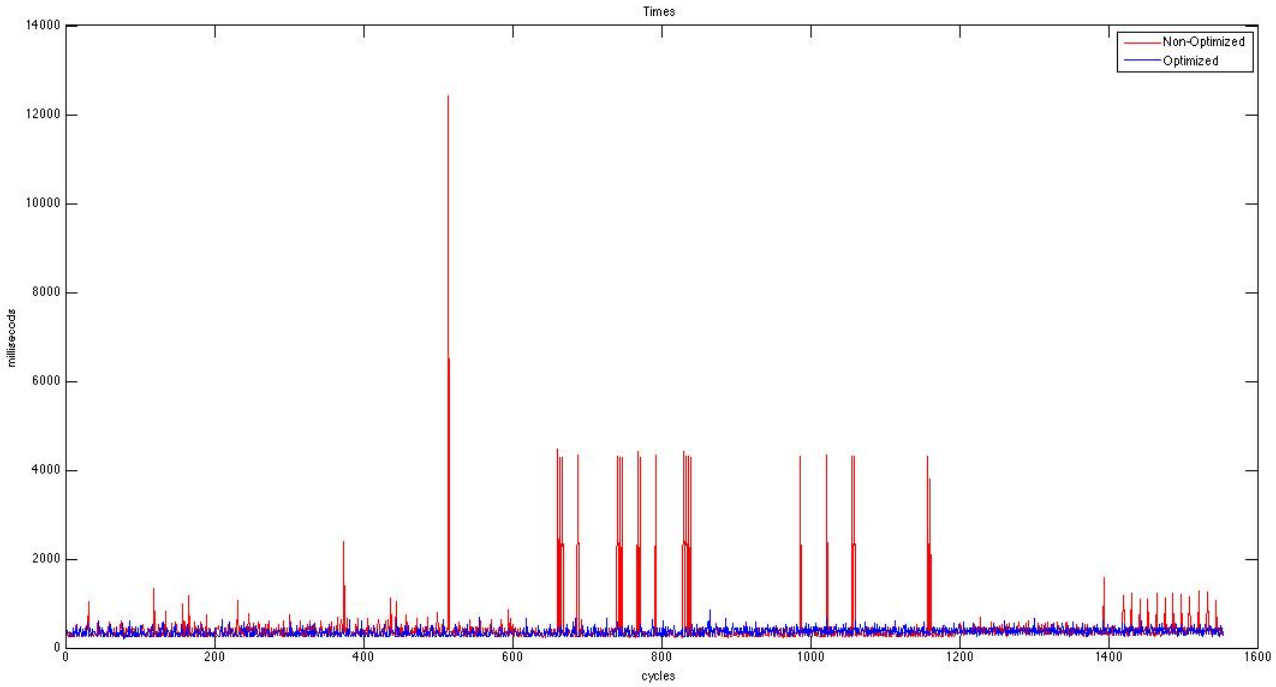


Figure 7.1: Timing with and without code optimizations

We can see in Figure 7.1 that, the non-optimized sometimes had huge time peaks, the biggest of 12409 milliseconds. We analyzed the data looking for the culprit functions that incurred in these delays. It was discovered that the functions used to send data through the XBee radio module could get locked under certain circumstances. The data analysis also showed that the function used for getting the GPS position could incur in occasional delays of 1 to 3 seconds. We rewrite some parts of these Waspmote's API functions to reduce the time of each delay to a maximum of 100 milliseconds.

In the first versions of the on board control system the cost of logging on the SD card was one of the more expensive invocations (45 msecs). Luckily, it was found that it could be reduced to just a few milliseconds only changing from asynchronous to synchronous writing as shown in Table 7.1.

Table 7.1: SD logging time improvements

	Before	After
Min	35 ms	0 ms
Max	152 ms	144 ms
Mean	45.33 ms	4.13 ms
Standard deviation	6.52 ms	7.14 ms

Figure 7.2 displays graphically the timing results after code optimization. Table 7.2 summarizes the effect of timing improvements after code optimization on cycle time. Similarly, Table 7.3 shows the effects over the main parts of the code.

Table 7.2: Cycle time statistics

	Before	After
Min	238 ms	140 ms
Max	12409 ms	635 ms
Mean	424.39 ms	285.86 ms
Standard deviation	556.05	93.76

Table 7.3: Times

Function	Min	Max	Mean	Standard deviation
checkForNewPackets(1)	0	59	0.93	5.89
checkTimeOut	0	3	0.14	0.56
checkBearing	8	172	23.64	43.90
updateSensors	52	312	174.76	65.02
checkForNewPackets(2)	0	60	3.53	13.95
prepareTelegram	60	121	65.93	10.51
radioSend	0	27	6.64	11.57
logTelegram	0	28	3.84	4.14

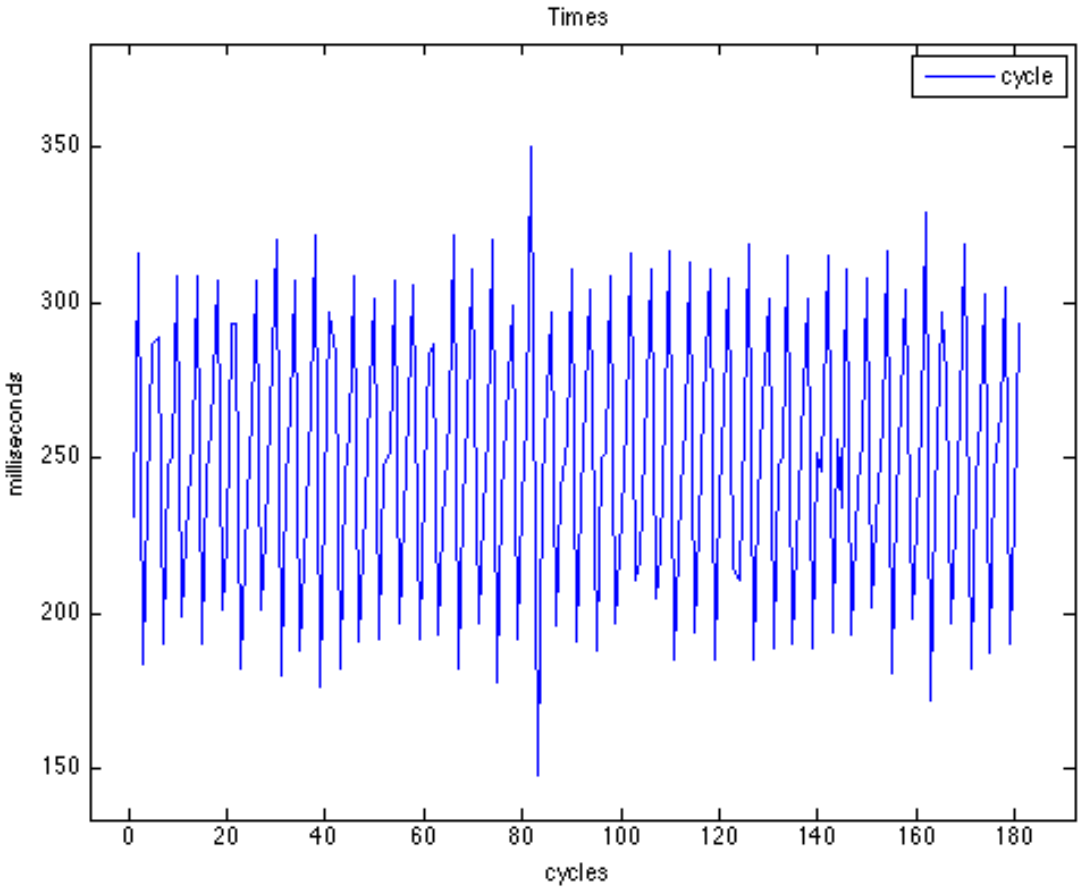


Figure 7.2: Timing with and without code optimizations

7.1.2 Multirate Control Cycle

The A-TIRMA sailboat does not exhibit fast dynamics that demand a control frequency much higher than the average 3-4 Hz discussed in the previous section. It is basically determined by the cost of executing certain tasks and reading slow sensors as the GPS.

However, we can improve the response of the system to external commands, for example when in remote control mode, or the quality of data provided by some fast but noisy sensors, if some fast operations are performed several times per cycle.

This simple idea has been applied to duplicate the frequency at which we check the reception of new data packets on the radio and also to sample wind direction. In the first case, we get a much more responsive behavior of the A-TIRMA to external remote control commands; in the second case, the wind direction filtered estimates are better. With this approach we can double (8Hz) the operational frequencies for some parts of the code.

7.2 Sensors tests

7.2.1 Wind Load Sensor

The A-TIRMA lacks a wind speed sensor or anemometer. On this work we have tested the usage of a current sensor, an ACS712 board Section 5.6, to indirectly estimate the wind pressure or wind load on the sails from the current consumed at the winch servo, while it tries to keep the sails in position under wind load.

This approach has to deal with some difficulties. Firstly, the intensity consumed at the winch servo is not constant as it's under PWM control. This issue has been solved integrating the measurements provided by the ACS712 during 40 milliseconds, the double of the PWM period. Secondly, the current consumed is critically dependent on the state of motion of the

servo. Only if the servo is at rest, if it has arrived to the commanded position, the current consumed will be used to maintain the servo in position under the wind load. In the current implementation the wind load measurement is only produced when the sails position has been maintained along the last two cycles.

Figure 7.3 depicts a typical situation, note that the highest the current consumed at the winch, the lowest the reading returned by current sensor. First the winch is commanded to a new position, denoted by the red line. Once the servo is in position, and using the roll angle as a secondary evidence of wind strength, it can be appreciated how the ACS712 measurement follows approximately the evolution of the roll angle.

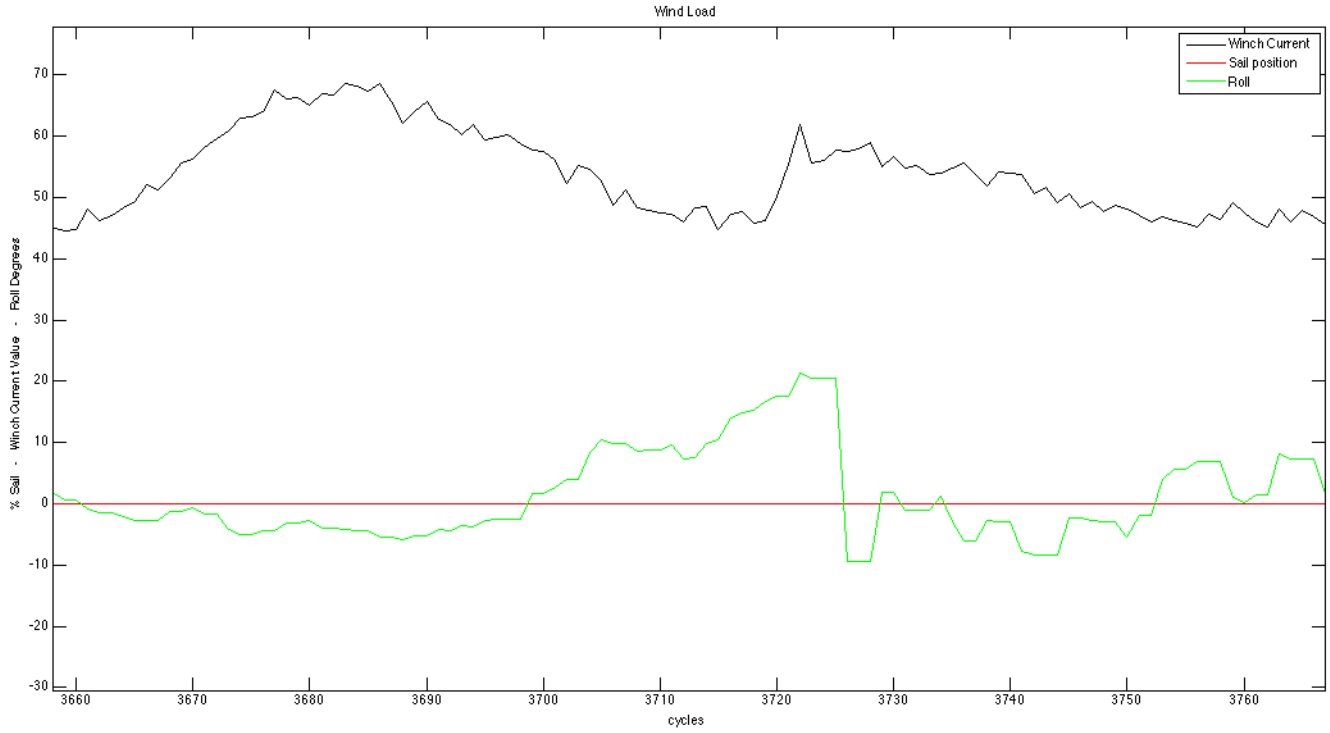


Figure 7.3: Wind load estimation

7.2.2 Wind Vane Noise

In the A-TIRMA control system, it is of paramount importance to have available a stable estimate of wind direction to adjust the sails position and to select the best bearing to reach a waypoint. However, the wind vane is in essence a noisy sensor due to its principle of operation and also due to it is placement on top of the mast, that makes it quite sensitive to pitch and roll movements of the sailboat and sudden wind gusts.

To cope with these circumstances some form of filtering of the raw data provided by the wind vane is unavoidable. In the A-TIRMA a median filter with window of length 7 has been used with good results. The median filter was selected by its well known capability to eliminate impulsive noise. Figure 7.4 shows the raw readings obtained from the wind vane and the resulting signal after applying the median filter. The outcome is a more stable wind direction estimate that translates in a better control of the sailboat.

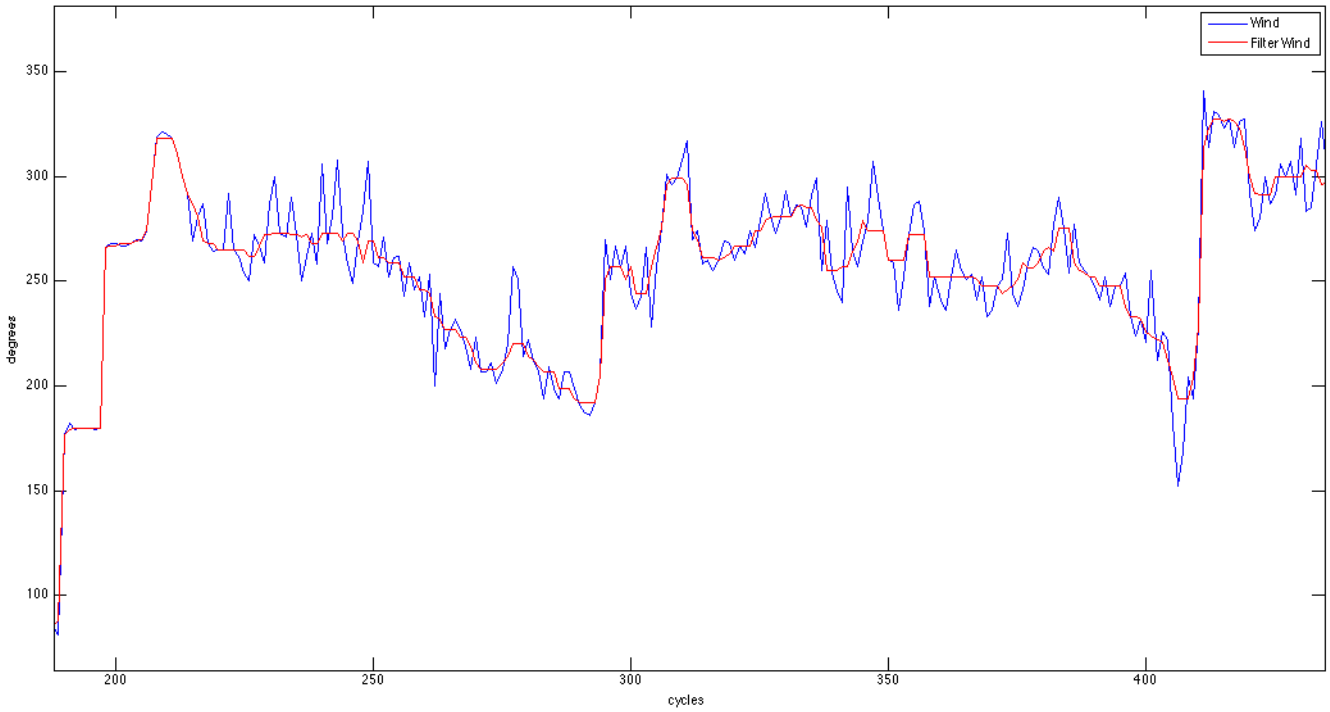


Figure 7.4: Wind Filter

7.2.3 TCM2.50

The TCM2.50 board provides compass, pitch, roll and temperature measurements. Although minimized by the placement of this board close to the base of the mast, these measurements also suffer from unintended movements produced during sailing. The TCM2.50 can filter its sensors reading and provide cleaner signals. The filtering parameters can be set by the on board control system. This possibility has been exploited to obtain cleaner measurements of bearing, pitch and roll angles directly from the TCM board.

The TCM2.50 board can be configured to output measurements only when interrogated or continuously. In this case it was configured to output continuously the readings of all sensors, except the 3-axis magnetometer readings, at 8 Hz. In that way the time required to obtain measurements is reduced considerably, making the whole system more agile.

7.2.4 GPS Measurements

The GPS available for the Waspmote is a slow sensor and it can be quite inaccurate if it is not properly configured.

We performed some previous tests to characterize the level of noise in GPS position estimates placing a static GPS receiver in an open area with a clear and equilibrated view of the sky. The results of these tests determined that better results were obtained if satellites that were too low over the horizon or provided very weak signal were not used to compute the position fix, as recommended in the GPS literature.

Figure 7.5 depicts a typical downwind-upwind test course under constant bearing, where the elevation mask was set to 15^{o} and the signal strength mask was set to 20 dBi, the standard GPS configuration in the A-TIRMA. This figure shows a vast majority of clean position estimates that have been characteristic of all sea trials under this configuration.

Nevertheless, we developed a Kalman filter to combine the measurements

of the GPS with the bearing angle provided by the compass and vessel's estimated speed to achieve more robust and faster position estimates. Even though the Kalman filter could provide some immunity against occasional large GPS errors, finally this filter has not been integrated in the current version of the on board control system due to RAM memory restrictions.

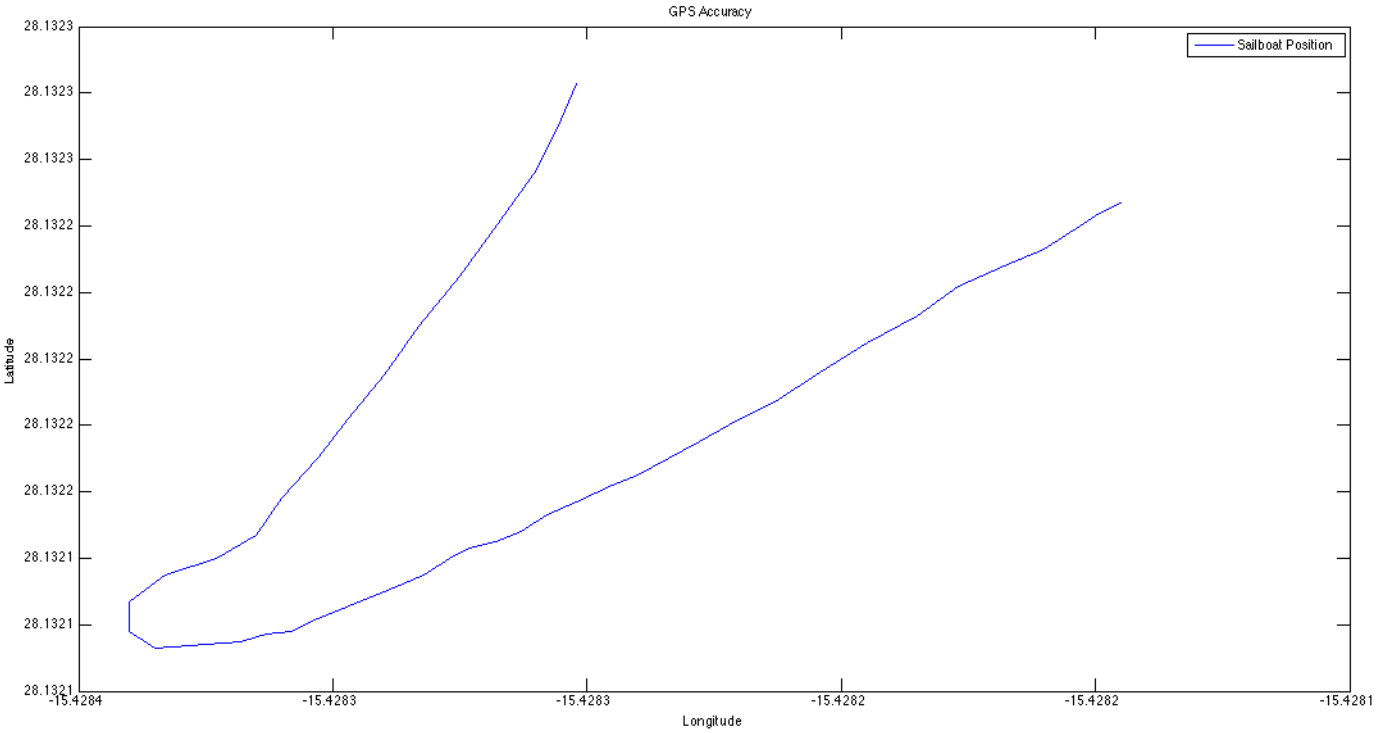


Figure 7.5: GPS Accuracy

7.3 Navigation Tasks

In order to test some of the navigation capabilities of the sailboat we have taken inspiration from tasks proposed for the World Robotic Sailing Championship (WRSC). Namely, these were a station keeping task, an accuracy task and an endurance task. A short description of the tasks proposed for WRSC'13 can be found in [47].

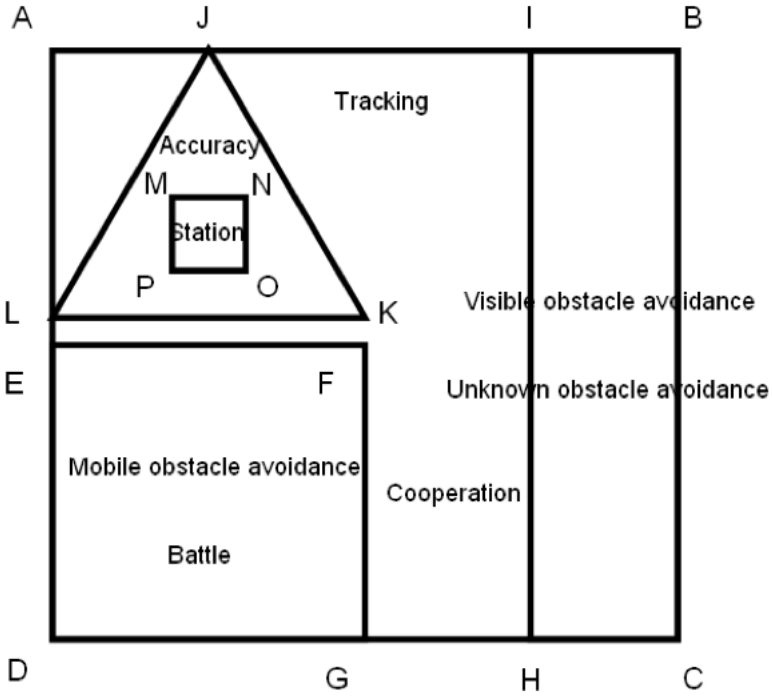


Figure 7.6: WRSC arena

7.3.1 Accuracy Task

In this task the sailboat has to follow autonomously a triangle, see Figure 7.6, visiting its vertices in a precise order and trying to keep the course as close as possible to the triangle sides. The objective here is to test how the sailboat sails under different points of sailing.

Figure 7.7 represents a test done on a triangle of the approximate dimensions included in Table 7.4. Wind direction along the route is also depicted. The route is colored differently for each side of the triangle. That day the wind was quite unstable, with frequent change of direction and intensity, with some strong gusts.

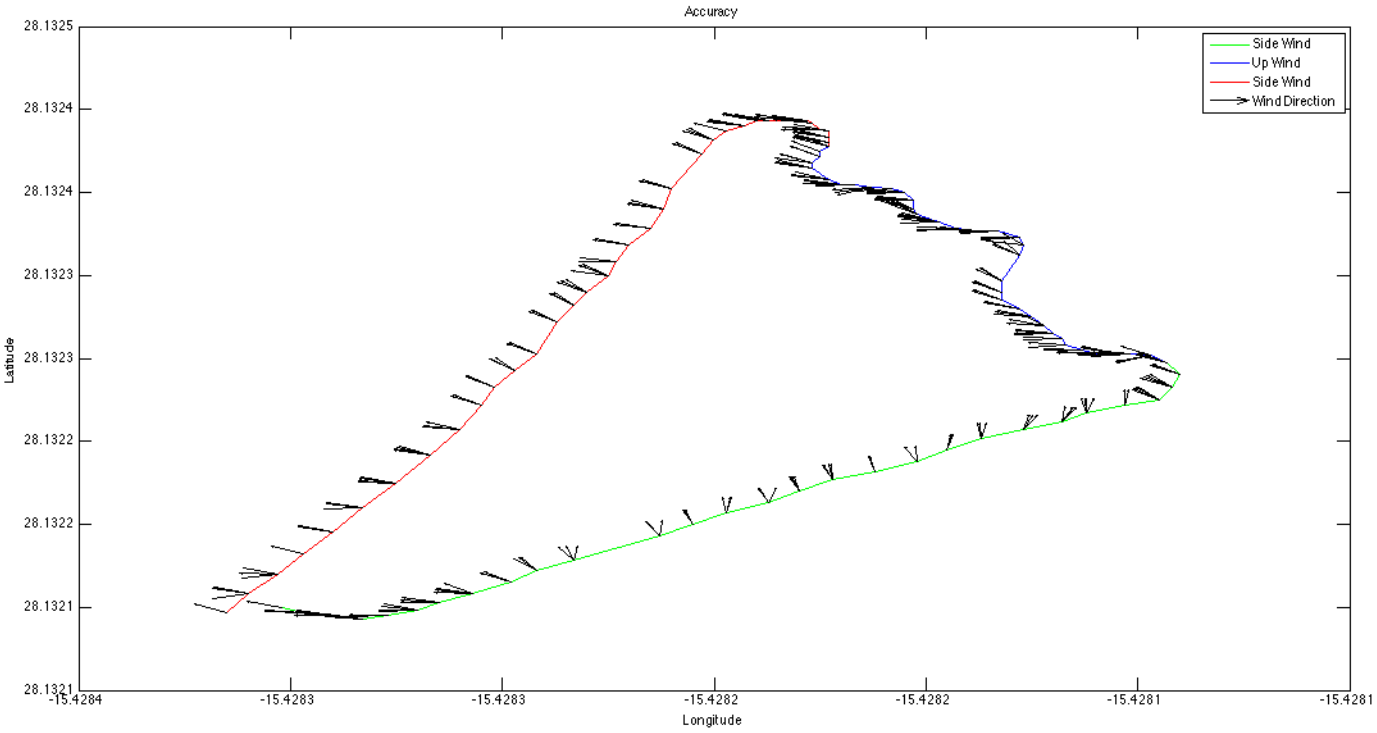


Figure 7.7: An accuracy test

The red line it is completely a sidewind course and we can see that the sailboat kept the course till it arrived to the waypoint. Then, in green, the sailboat started a downwind course that turned into a sidewind course. And, finally, the blue line is a upwind course and we can see how the sailboat do a series of tacks to reach the initial waypoint.

Table 7.4: Accuracy task

JK length	31.48 m
KL length	21.43 m
LJ length	37.74m

7.3.2 Return To Home (RTH)

The RTH it is an important behavior of the sailboat because it guarantees that if the sailboat loses the connection with the base station, it will initiate automatically the return to the Home waypoint.

On Figure 7.8 we can see how the sailboat automatically returns to home after losing the radio link. The sailboat was performing in a triangular route drawn in blue. After one round, we disconnect the radio antenna at the base station, when the sailboat was still close to the upper vertex. After 20 seconds without any reception from the base station, the sailboat changed the active waypoint to the Home waypoint, initiating the trajectory depicted in red. After reaching the Home waypoint, it tries to keep the station awaiting the restitution of radio communications.

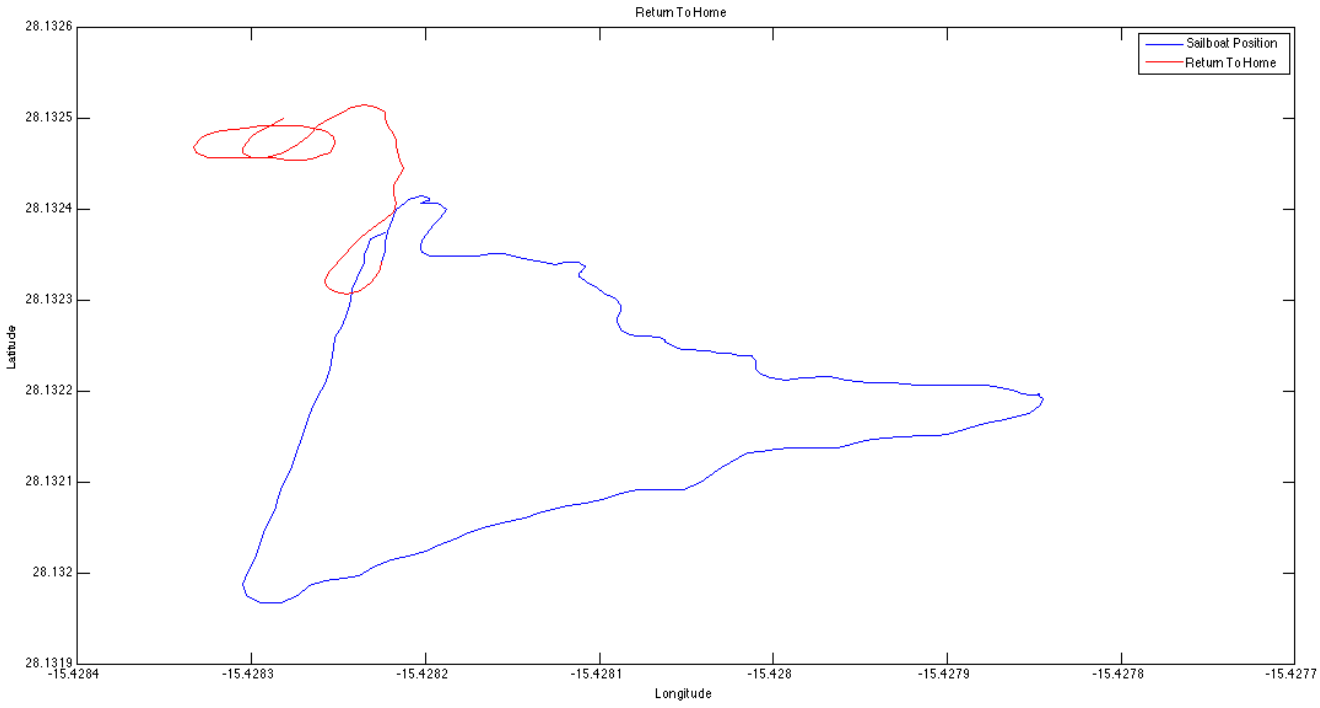


Figure 7.8: Return To Home

7.3.3 Excessive Heeling Effects

The A-TIRMA exhibits a behavior that is common to many sailboats when they are sailing close hauled and they heel in excess. Under those conditions, the observable behavior is a clear tendency to luff up or haul upon wind, loosing the bearing completely.

To avoid this type of instability, it is not only necessary to avoid extreme heeling angles, but also compensate the tendency to luff up with the rudder. To neutralize this effect, the rudder position computed by the fuzzy controller is modified when a heeling angle threshold is reached. The rudder correction is directly proportional to the heeling of the sailboat. It is worth noting that this kind of maneuver is routinely done by helmsmen on board of full-scale sailboats.

Figures 7.9 and 7.10 depict the consequences of not compensating for this effect. In Figure 7.10 it is evident how abrupt bearing changes follow once extreme roll (i.e. heeling) angles are reached. All this end up in a unstable course of polygonal appearance, see Figure 7.9, characterized by close hauled course segments, followed by hauls upon the wind, lose of speed, tack for bearing correction and so on continuously.

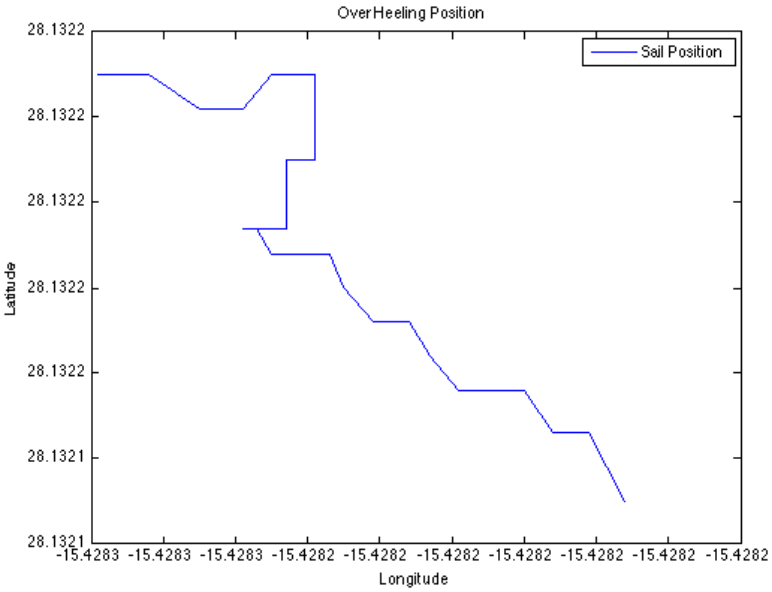


Figure 7.9: Over Heeling Position Effect

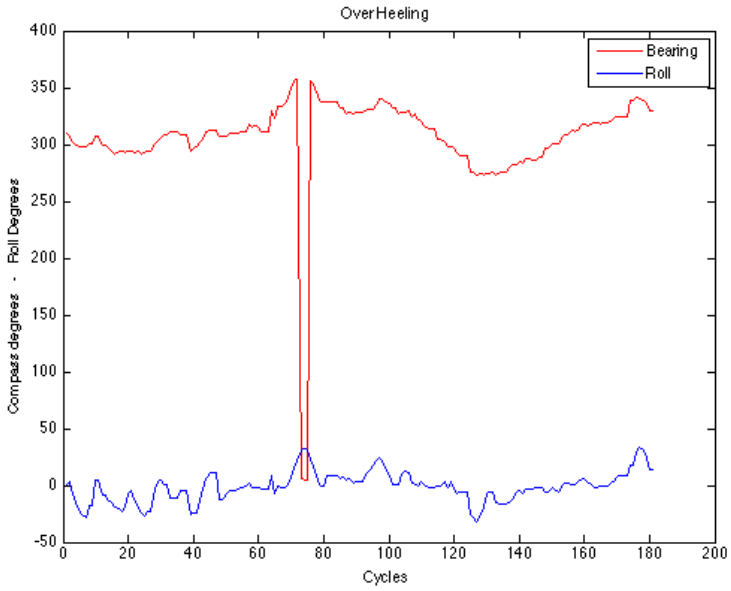


Figure 7.10: Over Heeling Effect

7.3.4 Station Keeping Task

Station keeping is the capacity of any vehicle to maintain is position close to a designated location. It has also been proposed as a test task for WRSC'13. In that case, it is required to go autonomously inside a square of side of 50 m and stay in the middle for a certain amount of time.

To test the station keeping task we set two waypoints with a distance of the center of the square of 1.25 meters, and we kept that point during 5 minutes. Figure 7.11 and Table 7.5 summarize the results.

Table 7.5: Mesurements

Distance between waypoints	2.50 m
Minimum distance to the center	0.38 m
Maximum distance to the center	7.83 m
Mean distance to the center	4.00 m
Standard deviation	1.79 m
Minimum square	10.45 x 13.33 m

We tried also this task using only one waypoint and kept the task active during 12 minutes with the results depicted Figure 7.12 and Table 7.6.

Table 7.6: Mesurements

Minimum distance to the center	0.22 m
Maximum distance to the center	12.76 m
Mean distance to the center	3.48 m
Standard deviation	2 m
Minimum square	15.2 x 17.24 m

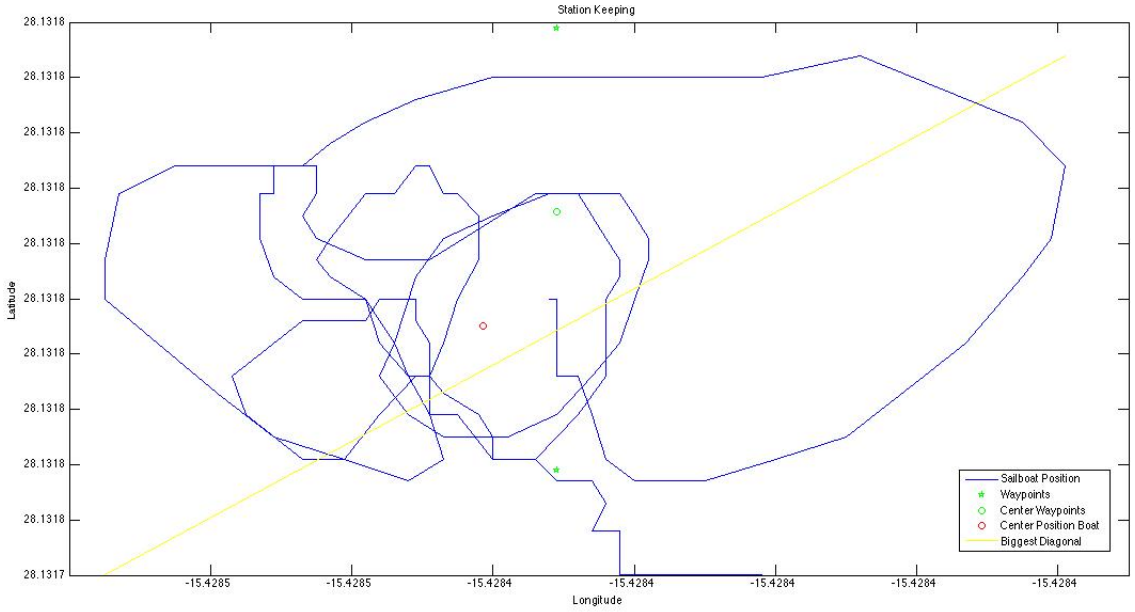


Figure 7.11: Station keeping results

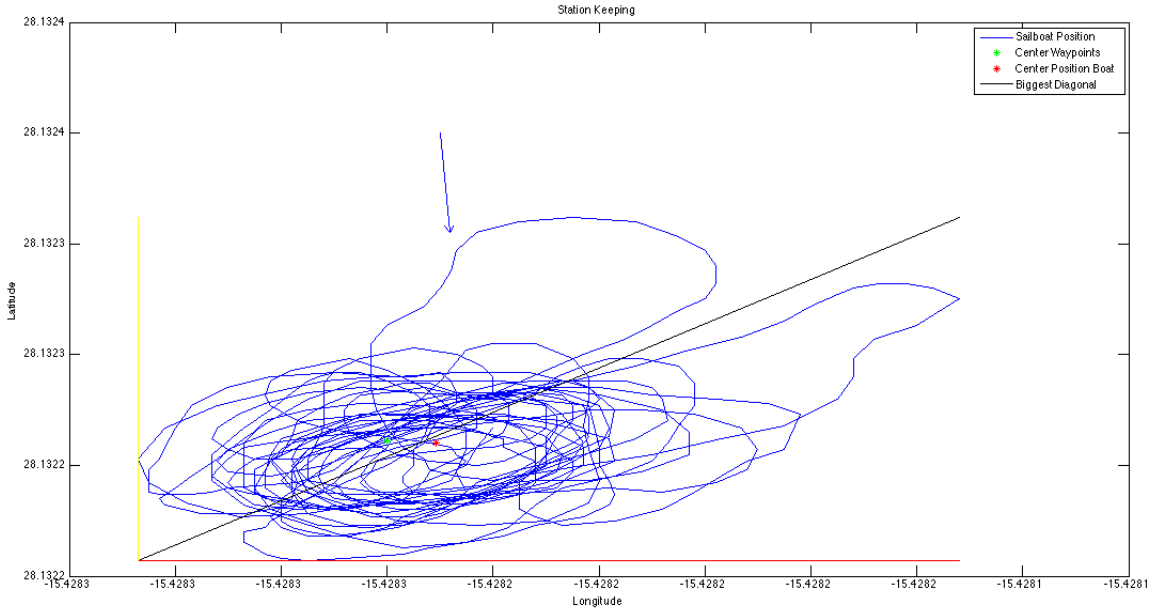


Figure 7.12: Second station keeping results

7.3.5 Endurance Tests

Endurance tests are typical of WRSC. For the 2013 edition, the sailboat will have to demonstrate its capacity to sail for 8 hours while performing some sensor sampling tasks.

Our sailboat has two sets of batteries, one powers the waspmote board and sensor meanwhile the second one powers the servos only. We did done an endurance test over three and a half hours and got the discharge curves shown in Figures 7.13 and 7.14.

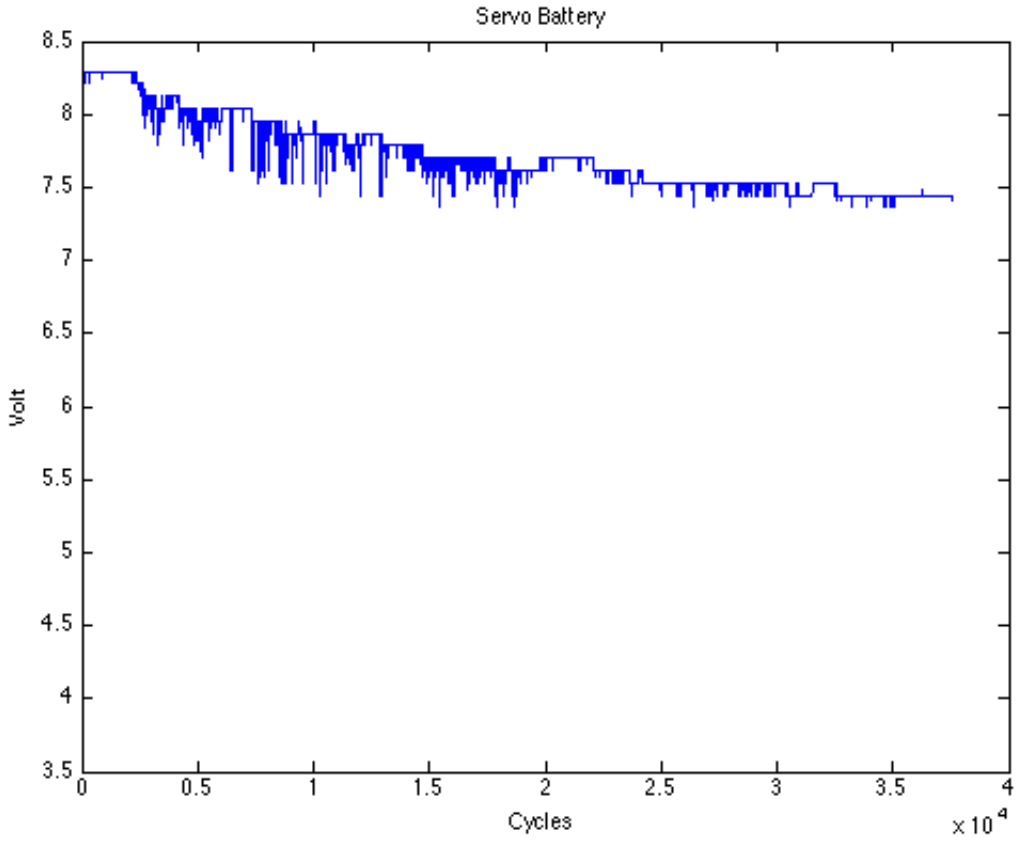


Figure 7.13: Servo Battery

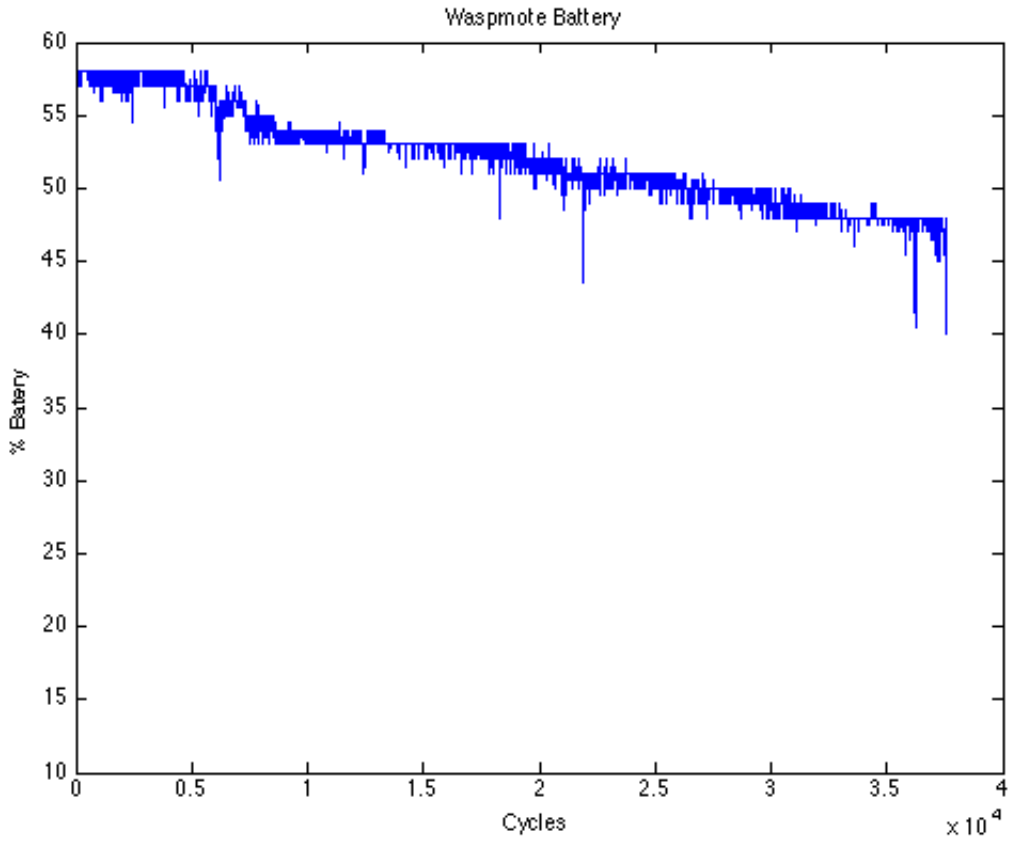


Figure 7.14: Waspmote Battery

We can see in both graphics that in three and a half hours the waspmote consumed only 10% of battery's capacity, and the voltage of the servo battery decreased in 0.5 volts. Note that the minimum operating voltage of servos is 4 volts.

Chapter 8

Conclusions

This work has been motivated by the necessity of developing an small and affordable autonomous sailboat platform that could be transported and operated by one or two people without any special means. In agreement with those objectives, this master thesis has described the design of a low cost autonomous sailboat whose development has been based on a standard RC One Meter class vessel and off the shelf low power hardware components.

The main features of the described system are its flexibility as experimental platform, its large power autonomy and its robustness in case of communication failures. Extensive sea trials in autonomous mode have been performed to confirm the stability and robustness of the A-TIRMA control system. The sailboat, in spite of her dimensions, has exhibited seaworthiness and demonstrated to be an interesting experimental platform.

The amount of space and displacement available in a One Meter class sailboat severely restricts the volume and weight of sensors and control electronics that can be installed on board. These restrictions have a negative impact in terms of computing power and sensors that can be used in the development of the sailboat's control system.

The main limitations of the control system described in this thesis are its scarce computing power and reduced RAM memory. These restrictions

have demanded a careful analysis and design of the control system to make it "fit" within the microcontroller memory and processing power, trying at the same time to reduce the span of a control cycle as much as possible.

This control system is very similar in scope to that described in [48], where it was presented a control system for autonomous sailboats based on a 50 MHz (64KB RAM) Cortex-M3 ARM7 processor board. The main difference between both systems, aside from the smaller computing power and memory of our system, is that our sailing control system is completely embedded on the on board processor, while in [48] the sailboat controller executes in a laptop outside of the boat.

Perhaps the most fragile element of the whole system is the wind vane, as noted by many others [49]. Wind vanes with movable mechanical parts are intrinsically prone to failure. Whilst commercial solutions exist for full scale sailboats, they are unpractical for a boat of small dimensions. Some solutions have been explored and described in the literature but a truly robust design is still to be achieved. An alternative design for a wind sensor (direction and intensity) has been described in [50].

Finally as a result of the work carried out in this master thesis, a publication have been accepted to the IRSC'2013:

- J. Cabrera-Gámez, A. Ramos de Miguel, A.C. Domínguez-Brito, J.D. Hernández-Sosa, J. Isern-González and E. Fernández-Perdomo
- An Embedded Low-Power Control System for Autonomous Sailboats
- 6^{th} IRSC (International Robotic Sailing Conference)
- 2 to 6 September, Brest, France, Europe

And also that A-TIRMA is going to take part in the contests organized during the IRSC'2013.

8.1 Future Work

This area of field robotics is still in its childhood and we think there are many opportunities to contribute to its development.

Autonomous sailboats have many appealing characteristics to make them optimal vehicles for marine sampling and monitoring applications. First of all, this kind of vehicles allows simultaneous sampling of atmospheric and sea variables. They may offer a virtually unlimited range autonomy at relatively high speeds and ample payload capacity.

However, autonomous sailboats still have to evolve in many aspects to become more robust and reliable vehicles. Here, robustness implies multiple dimensions. On one side, it demands that the sailboat must be seaworthy, capable of standing strong sea conditions and tolerant to failure of on boards systems and sensors. On the other hand, they must be designed to be safe vessels that do not pose any threats to other boats. This requires better perception capabilities and the incorporation of collision avoidance tactics. It is important also to limit their speed and net displacement so they can be seen as potentially not harmful in case of failure.

Specific to the work that has been described in this master thesis, future work might address some of the following lines of development.

The current version of the A-TIRMA is quite limited due its physical dimensions. We would like to apply the A-TIRMA control system on a larger sailboat which might offer better navigation conditions and applicable on real field applications.

As commented before, sailing a larger boat requires endowing it with improved sensing capabilities and security mechanisms to perceive the environment and avoid collisions. A first step in that direction would be to develop a collision avoidance module based on an class B AIS receiver.

In the short term, we foresee to continue this work with the development of field application of autonomous sailboats like optimal area sampling in

environmental monitoring applications.

Appendix A

Improvements on Waspmote API

The Waspmote's API used in this project has been version 0.32 for hardware version 1.1. However, the API has been modified extensively in a number of aspects.

A.1 XBee Support

The XBee support in the Waspmote's API for hardware version 1.1 was substituted with and adapted version of the equivalent for hardware version 1.2 that has been modified in several aspects to improve reliability, memory consumption and speed.

One aspect that has been addressed in depth is the OTAP (Over The Air Programming) support included in the Waspmote's API. This feature was an important need in our context because it might facilitate enormously uploading new programs on board the sailboat during field trials. In order to adapt it to our needs, we implemented a new OTAP shell for Linux.

A.2 Timer

In order to control the sails positions with an adequate resolution, it was necessary to change the duty cycle of the PWM drive because with the the API's default duty cycle we could only discriminate among 6 positions. We adapted the TimerThree Arduino library [51] that easily allows to set the period and the duty cycle of the any of the ATMega timers. Now we have 20 different sail positions. This number of positions makes the system to be more accurate than before.

A.3 GPS

The GPS API module has been completely rewritten to achieve better control of the GPS receiver. Currently, the GPS receiver can operate at different UART speed, supports a broader range of binary commands and is much more robust.

A subtle problem related to GPS that has been addressed and solved is related to the roundup conversion error that is normally present when converting longitude and latitude coordinates, obtained from NMEA sentences, to equivalent single precision coordinates in decimal degrees. Note that the avr-libc does not provide support for double precision real numbers.

Latitude and longitude coordinates are obtained from NMEA messages as strings with the format $[-]dddmm.cccc$, where d stands for digits of "degrees", m for digits of "minutes" and c are the decimal part of minutes. Parsing these strings implies converting the minutes to degrees and adding it to the degrees.

The operation is straightforward in principle, but due to the limited capacity of single precision real numbers to represent real numbers up to six decimal digits, it requires a more careful conversion procedure. Otherwise, the roundup errors are normally below a meter but occasionally the

combined error in latitude and longitude can reach 2 meters.

On Table A.1, we can see the difference between the previous function and the new one, and the error in meters.

Table A.1: GPS improvement

NMEA coordinate	Decimal deg Before	Decimal deg After	Diff. in meters
2830.2789	28.504646	28.504648	0.22m
-12847.3562	-128.789270	-128.789276	0.6672m
8147.0002	81.783337	81.783333	0.4448m
12847.0001	128.783335	128.783340	0.556m
-01547.0001	-15.783335	-15.783336	0.1112m
-2806.7944	-28.113240	-28.113241	0.1112m

A.4 AVR-Libc Error

We saw that the avr-libc[52] version used in Waspmote API had a bug on the management of the dynamic memory. This error appeared when we get and free memory during a lot of time. It produced fragmentation of the memory when trying to free acquired memory, the problem was that really the memory was not being freed. So we lost memory because avr-libc's functions malloc and free did not work correctly. To solve the problem we adapted the solution for Arduino to our system. We used a new implementation for memory management developed by Joerg Wunsch [53], and we placed it in our API directory.

Bibliography

[1] ARGOS Program Information Center http://wo.jcommops.org/cgi-bin/WebObjects/Argo

[2] Autonomous Underwater Vehicle. http://en.wikipedia.org/wiki/Autonomous_underwater_vehicle

[3] Autonomous Undersea Vehicle Apllication Center. Portal on AUV technology.http://auvac.org/

[4] Underwater gliders. http://en.wikipedia.org/wiki/Underwater_glider

[5] Autonomous Surface Vehicle. http://en.wikipedia.org/wiki/Unmanned_surface_vehicle

[6] Liquid Robotics. http://liquidr.com/technology/wave-glider.html

[7] Global Ocean Drifter Program. http://www.aoml.noaa.gov/phod/dac/index.php

[8] SkySails GmbH. Web site. http://www.skysails.info/english/

[9] Roberts,G.:Trends in marine control systems.Annual reviews in control 32(2), 263–269 (2008), http://202.114.89.60/resource/pdf/2236.pdf

[10] BalancedRig: BalancedRig website. Available online: http://balancedrig.com/description.html (last accessed: Julyl 2013) (2013)

[11] Stelzer, R., Estarriola Dalmau, D.: A Study on Potential Energy Savings by the User of a Balanced Rig on a Robotic Sailing Boat, in Proceedings of International Robotic Sailing Conference, pp. 87-94, Cardiff, Wales, UK, 2012

[12] Protei: Open Source Sailing Drone. Web site, http://www.protei.org

[13] Sailbot competition website. http://sailbot.org/

[14] World Robotic Sailing Championship web site. http://www.roboticsailing.org/

[15] Sailbuoy web site. http://sailbuoy.no

[16] Rynne, P. F., von Ellenrieder, K. D.: Unmanned autonomous sailing: Current status and future role in sustained ocean observations, Mar. Technol. Soc. J., vol. 43, no. 1, pp. 21–30, 2009

[17] Rynne, P. F., von Ellenrieder, K. D.: Development and Preliminary Experimental Validation of a Wind- and Solar-Powered Autonomous Surface Vehicle, IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 35(4), pp.971-983,October 2010

[18] Stelzer, R., Jafarmadar, K.: History and Recent Developments in Robotic Sailing in in Proceedings of International Robotic Sailing Conference, pp. 3-23, Lübeck, Germany, 2011

[19] Stelzer, R.; Jafarmadar K.:The Robotic Sailing Boat ASV Roboat as a Maritime Research Platform in Proceedings of 22nd International HISWA Symposium on Yacht Design and Yacht Construction, Amsterdam, The Netherlands. 2012

[20] Stelzer, R.: Autonomous Sailboat Navigation – Novel Algorithms and Experimental Demonstration, PhD Thesis, Centre for Computational Intelligence, De Montfort University, UK, 2012

[21] Allensworth, T.: A short history of Sperry Marine (1999), `http://www.sperrymarine.northropgrumman.com/Company-Information/Corporate-History/Sperry-History/`

[22] Ninorsky, N.: Directional stability of automatic steered bodies. Journal of American Society of Naval Engineers 34(2), 280-309 (1922)

[23] Stelzer, R., Pröll, T., John, R.I.: Fuzzy Logic Control Sysrtem for Autonomous Sailboats. IEEE.(2007)

[24] Polkinghorne, M., Roberts, G., Burns, R., Winwood, D.: The implementation of fixed rulebase fuzzy logic to the control of small surface ships. Control Engineering Practice 3(3), 321–328 (1995), http://202.114.89.60/resource/pdf/2231. pdf

[25] Sauzé, C., Neal, N.: MOOP: A Miniature Sailing Robot Platform. Robotic Sailing. Proceedings of the 4th International Robotic Sailing Conference. Part II. Pages 39-53, 2011.

[26] Burnie, M. (ed.): Participant Package – World Robotic Sailing Championship 2010 and International Robotic Sailing Conference 2010. Queen's University, Kingston (2010)

[27] Introduction to sailing concepts. `http://en.wikipedia.org/wiki/Sailing`

[28] Carr, R., Sails: The Power of Source, Model Yachting Magazine, Winter Issue, 2006. Available on line at `http://www.theamya.org/Intro_Sails.pdf`

[29] An article about the TIRMA yacht. http://escoben.blogspot.com.es/2005/04/el.html

[30] Stelzer, R., Pröll, T.: Autonumous sailboat navigation for short course racing. Robotocs and Autonomous Systems., **56**, 604-614(2008)

[31] Libelium's Waspmote manual. http://www.libelium.com/v11-files/documentation/waspmote/waspmote-technical_guide_eng.pdf

[32] Arduino web site. http://playground.arduino.cc

[33] Libelium's OTAP manual. http://www.libelium.com/downloads/documentation/ove_the_air_programming.pdf

[34] XBee 868 Pro specifications. http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-pro-868\#specs

[35] PNI's legacy TCM2.5 electronic compass manual. http://www.tri-m.com/products/precisionnav/files/specs/tcm25_spec.pdf

[36] MAX3232 Datasheet. http://www.ti.com/lit/ds/slls410i/slls410i.pdf

[37] NMEA 0183 Interface Standard, National Marine Electronics Association. http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp

[38] A1084 GPS receiver hardware manual. http://ec-mobile.ru/user_files/File/Tyco/A1084_HM_V1.0.pdf

[39] US Digital absolute encoder MA3. http://www.usdigital.com/products/encoders/absolute/rotary/shaft/ma3

[40] ACS712 product page. https://www.sparkfun.com/products/8883

[41] Qt UI framework. `http://qt-project.org`

[42] LibXBee library. `http://code.google.com/p/libxbee/`

[43] SDL library web site. `http://www.libsdl.org`

[44] J.A. Spaans, Windship routeing, Journal of Windship Engineering and Industrial Aerodynamics 19 (1985) 215–250.

[45] EFLL fuzzy logic library. `https://github.com/zerokol/eFLL`

[46] Web site of the World Robotic Sailing Championship 2012, Cardiff, Wales, September 2012, `http://www.microtransat.org/wrsc2012`

[47] Rules for the 2013 edition of the World Robotic Sailing Championship. `http://www.ensta-bretagne.eu/wrsc13/pdf/Rules\_2013-07-17.pdf`

[48] Koch, M., Petersen, W.: Using ARM7 and C/OS-II to Control an Autonomous. Sailboat Robotic Sailing 2011, pp 101-112, Springer 2012

[49] Neal, M. , Sauze,C. , Thomas, B. and Alves, J. C.:Technologies for Autonomous Sailing: Wings and Wind Sensors, in proceedings of the 2nd IRSC , Matosinhos, Portugal, July 6th-12th 2009, pages 23-30, 2009.

[50] Alvira, M., Barton, T.: Small and Inexpensive Single-Board Computer for Autonomous Sailboat Control, Robotic Sailing 2012, pp 105-116, Springer, 2013

[51] Arduino Timer library. `http://playground.arduino.cc/Code/Timer1`

[52] AVR-Libc Documentation `http://www.nongnu.org/avr-libc/user-manual/index.html`

[53] AVR-Libc Solution by Joerg Wunsch http://code.google.com/p/arduino/issues/detail?id=857

Contents

List of Figures

List of Tables