ESCOLA NAVAL

talant de bien faire

Pedro Miguel de Castro Fernandes

Projeto e Construção de um Veleiro Autónomo, utilizando materiais compósitos, Impressão a 3D e Aprendizagem Máquina

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Mecânica

Alfeite

2016

Pedro Miguel de Castro Fernandes

Projeto e Construção de um Veleiro Autónomo, utilizando materiais compósitos, Impressão a 3D e Aprendizagem Máquina

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenharia Naval Ramo de Mecânica

Orientação de: Professor Doutor Victor Sousa Lobo

O Aluno Mestrando,

ASPOF EN-MEC Castro Fernandes

O Orientador,

Professor Doutor Victor Lobo

Alfeite

2016

Epígrafe

"Somos assim: sonhamos o voo mas tememos a altura. Para voar é preciso ter coragem para enfrentar o terror do vazio. Porque é só no vazio que o voo acontece. O vazio é o espaço da liberdade, a ausência de certezas, mas é isso o que tememos: o não ter certezas. Por isso trocamos o voo por gaiolas. As gaiolas são o lugar onde as certezas moram."

Os irmãos Karamazov

Fiódor Dostoiévski

Agradecimentos

Só foi possível a realização desta dissertação de mestrado, com o auxílio de inúmeras pessoas, às quais, não poderia de maneira nenhuma deixar de demostrar o meu total reconhecimento por todo o apoio e tempo que me disponibilizaram.

Gostaria de agradecer ao meu orientador, Professor Victor Lobo, pelo apoio, motivação e supervisão prestados ao longo do trabalho.

Ao Sr. Engenheiro Monteiro Marques pela ajuda valiosa na realização de três artigos científicos que valorizaram o trabalho aqui apresentado.

À minha família pelo suporte e motivação. À Ariana e à sua família pelos mesmos motivos.

Ao Sr. Engenheiro Mário Figueiredo pelo apoio e conhecimentos transmitidos na área de compósitos que em muito valorizaram o trabalho.

Ao Sr. Almirante Rodolfo e a toda a estrutura AFCEA Portugal.

À guarnição do NRP D. Carlos I pela ajuda indispensável na realização das provas de mar do veleiro.

Aos meus camaradas ASPOF EN-MEC Lopes Nunes, ASPOF Freire Correia, ASPOF Pereira da Silva, CAD Silva Ferreira e CAD Rocha Araújo.

Aos Professores militares e civis do Departamento de Engenheiros Navais da Escola Naval, que me ministraram conhecimentos indispensáveis para a realização deste trabalho, nomeadamente os Professores das cadeiras: Eletrotecnia, Eletrónica, Arquitetura Naval, Órgãos de Maquinas, Maquinas Marítimas e Mecânica de Fluidos.

Resumo

Este trabalho consistiu no projeto e construção de um veleiro autónomo de pequena escala. No início do trabalho, é feito um estudo acerca dos diferentes tipos de veículos autónomos, dando mais enfase aos veleiros. Em seguida, é iniciado o projeto do casco do veleiro, aplicando conceitos básicos de Arquitetura Naval. A forma do casco é desenhada com recurso ao programa DELFT Ship Free, onde são realizados estudos hidrodinâmicos do mesmo. Posteriormente é retratado a construção do casco projetado, com recurso a materiais compósitos e impressão 3D de componentes do veleiro. São ainda descritos os sensores, controladores, atuadores e programação desenvolvida para o veleiro. É também realizado um estudo sumário da estimativa de consumos e autonomia do sistema. No final, encontram-se os resultados obtidos das provas de mar efetuadas ao veleiro.

Palavras-chave: Veleiro Autónomo, Impressão 3D, Compósitos, Arduíno.

Abstract

This work is the project and construction of a small scale sailboat. It starts with an overview of the major types of unmanned vehicles. Then, we introduce the subject of designing the hull shape using naval architecture principles. The hull shape was drawn using DELFT Ship Free, where some hydrostatic and hydrodynamic studies were also made. The hull was built using steel reinforced composite materials and the peripheral systems were made using a 3D printer. A brief description of the electronic parts of the system is made, with emphasis on the sensors, actuators and microcontroller used. Some estimates are made on the cost of the project and the prototype, as well on it's autonomy. Finally we present and discuss the results obtained in the testes and competitions where the sailboat was used.

Key-Words: Autonomous sailboat, 3D printer, Composite, Arduino.

Índice

Índice de Figuras

Índice de equações

Lista de Tabelas

Abreviaturas

ABS - Acrilonitrila Butadieno Estireno

AUV - Autonomous Underwater Vehicle

CAD – Computer Aided Draw

CMRE - Centre for Maritime Research and Experimentation

EN – Escola Naval

EUA - Estados Unidos da América

FEUP - Faculdade de Engenharia da Universidade do Porto

GPS - Global Position System

Marocup - MAritime RObotic CUP

NATO - North Atlantic Treaty Organization

PLA - Polylactic Acid

REP - Rapid Environment Picture

SCOUT - Surface Craft for Oceanographic and Undersea Testing

UAV - Unmanned Aerial Vehicle

USV - Unmanned Surface Vehicle

UxV - Unmanned “x” Vehicle

1. Introdução

1.1. Motivação

A sociedade de hoje em dia é marcada pelas máquinas. De dia para dia surgem novos engenhos, mais capazes, com mais funcionalidades e melhores propriedades. Observa-se que a execução de tarefas é mais frequentemente delegada a estas máquinas, salvaguardando a vida humana, em tarefas de maior risco, reduzindo custos e aumentando a produtividade. Porém, há situações para as quais ainda não existem máquinas fiáveis e eficientes, sendo necessária intervenção humana.

Os veículos autónomos ganharam enfase recentemente, com a generalização de controladores e sensores de baixo custo. Estes veículos apresentam diversas vantagens inerentes às suas capacidades: podem operar em situações perigosas, têm custo relativamente baixo, são versáteis, podem operar em diversos meios, entre outras. Os veículos autónomos podem ser divididos em vários tipos, sendo a sua designação genérica UxV (*Unmanned "x" Vehicle*). Os veículos autónomos mais frequentes são UUV[1] (*Unmanned Underwater Vehicle),* USV (*Unmanned Surface Vehicle*), UGV (Unmanned Ground Vehicle) e UAV (*Unmanned Aerial Vehicle).*

Os veículos de superfície podem ser de diversos tipos tais como lanchas rápidas, navios para rocega de minas ou embarcações de grande *endurance*. Existem vários exemplos de veículos de superfície com capacidade de permanecerem largos períodos de tempo no mar, dos quais se destaca:

- WaveGliders (Willcox, 2010)

Esta embarcação foi desenvolvida pela empresa *Liquid Robotics*, situada nos EUA (Estados Unidos da América). Este projeto contou com mais de 81 milhões de dólares na sua fase de desenvolvimento (Crunchbase, 2016). A velocidade máxima desta embarcação é 2,25 nós, contudo, a velocidade média situa-se nos 1,5 nós. A grande vantagem deste tipo de embarcação está inerente ao recurso que utiliza para obter

[1] *também chamados AUV-Autonomous Underwater Vehicle pela comunidade que trabalha nesta área há mais tempo*

propulsão, a ondulação, permitindo que execute missões de longa duração. Atualmente, o waveglider detém o "Guinness World Record" para a embarcação não tripulada que percorreu a maior distância em autónomo, 14.703Km's, numa viagem entre São Francisco, EUA e Bundaberg, Austrália (Liquid-robotics, 2016). Na Figura 1 encontra-se uma imagem que demonstra o principio de funcionamento deste veiculo.

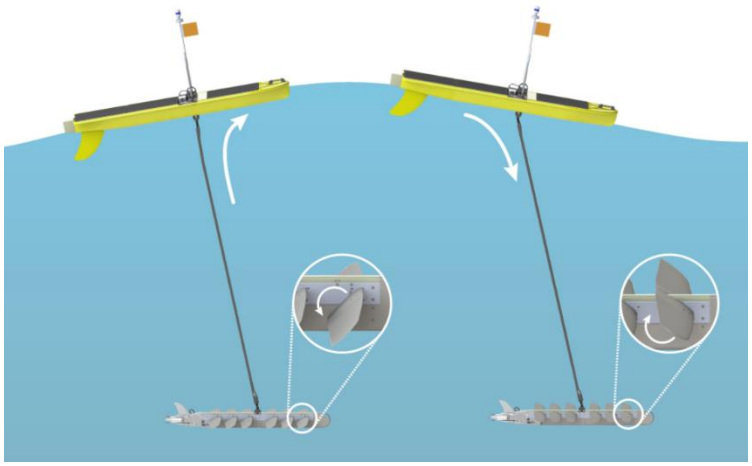


Figura 1 - WaveGlider

A Marinha Portuguesa teve o seu primeiro contacto com waveglider no exercício REP 13, com o waveglider PLOCAN (Centro de Investigação do Governo Regional das Canarias) e este tipo de embarcações foi usado pelo CMRE (Centre for Maritime Research and Experimentation), da NATO (North Atlantic Treaty Organization) durante os exercícios REP (Rapid Environment Picture) 14 e 16 ao largo de Sesimbra.

- Embarcações solares

Este tipo de veículo utiliza painéis fotovoltaicos para alimentar os sistemas de bordo e deste modo poder navegar. Um exemplo deste tipo de veículo é o "SCOUT" (Surface Craft for Oceanographic and Undersea Testing) (Joseph Curcio, 2005). Esta embarcação, mais não é que um Kayak com controladores, sensores e atuadores instalados por forma a atingirem o objetivo de navegar de forma autónoma. Esta embarcação foi desenvolvida no *MIT's Center for Ocean Engineering.* Este projeto iniciou-se com a pretensão de desenvolver uma plataforma capaz de realizar missões oceanográficas, tais como medição de parâmetros físico-químicos da água através de um sensor imerso rebocado, ou a entrega de medicamentos a populações afetadas por fenómenos naturais tais como *Tsunamis* (Mone, 2007).

A Escola Naval francesa, com a qual a Escola Naval tem uma relação próxima, desenvolveu também um veículo solar em 2007 e 2008, que está em exposição no átrio do seu centro de investigação. Na Figura 2 encontra-se uma fotografia da embarcação.

Figura 2 - Embarcação solar francesa

- Veleiros

Outra solução para o problema da propulsão de embarcações autónomas é o vento, através do desenvolvimento de veleiros autónomos. O facto de ser o vento a impulsionar o veleiro constitui uma vantagem, pois consegue-se uma autonomia energética muito grande. A energia elétrica (que pode vir de painéis solares) apenas é necessária para marear as velas, atuar no leme, e alimentar os sensores e computador de bordo. Por outro lado, a dependência do veículo do vento torna-o vulnerável a situações de vento excessivo, ou ausência de vento, que podem comprometer a aptidão do veleiro em cumprir com a sua missão. Porém, em ambiente marinho situações em que se verifica ausência de vento são raras, e é possível dimensionar os componentes do veículo de forma à resistência ser adequada ao meio onde opera, e, complementarmente, criar algoritmos que protejam o veleiro em casos de vento excessivo. Em 2010 a Escola Naval iniciou o desenvolvimento de um veleiro de pequenas dimensões autónomos (Rocha; Cavaco, 2011). O objetivo era o desenvolvimento de conhecimentos nesta área, a partir do desenvolvimento de eletrónica que permitisse a navegação autónoma de pequenos veleiros “Lazer RC”.

A importância de desenvolver as capacidades de veículos autónomos à vela é amplamente reconhecida, sendo que muitos centros de investigação têm investido neste sentido (Alves & Cruz, 2008). Os veleiros são uma solução versátil para responderam a inúmeros problemas, tais como pesquisa oceanográfica, missões de patrulha oceânica, ou até mesmo transporte de mercadorias para áreas remotas, cujos métodos tradicionais de transporte verificam-se economicamente insustentáveis (Alt & Wittinghofer, 2011).

A arquitetura base destes veleiros assenta num módulo de controlo, constituído por sensores, sendo o *Global Position System* (GPS), bússola e anemómetro os principais, e uma unidade de processamento, agrupados de forma generalista em microcontroladores, como o usado no Arduíno (e.g. ARDUINO, 2016), ou microprocessadores tais como o usado *Raspberry Pi* (e.g. Foundation, 2016). Por forma a conseguir navegar, estes veleiros contêm mecanismos, por norma servos, que controlam o ângulo das velas e do leme em função das diretivas da unidade de processamento. Exemplo deste tipo de arquitetura é o *Fast* da Faculdade de Engenharia da Universidade do Porto (FEUP) (Alves & Cruz, 2008), ou o AEOLUS da ETH Zurich (Swiss Federal Institute of Technology) (Wirz, et al., 2015).

A norma é estes veículos serem programados parametricamente, ou seja, são definidas várias funções, e cada uma destas é ativada consoante o *input* recebido pelos sensores. Isto constitui uma limitação, porque tornar o veículo eficiente implica aumentar em grande medida o número de funções, por forma a tirar maior vantagem das condições ambientais para executar a sua missão. A capacidade de adaptação fica igualmente limitada a instruções *hardcoded* na unidade de processamento (i.e. um comportamento pré-definido e não alterável em tempo real). Para ultrapassar esta limitação, é importante o desenvolvimento de algoritmos de *Machine Learning*[2] aplicados ao caso específico da vela autónoma. Com *Machine Learning*, para além de comportamentos pré-definidos, é possível fazer com que o veleiro, a partir dos dados recolhidos em missão e da sua experiência, desenvolva comportamentos novos que o projetista não previu.

Embora existam muitas referências de veículos autónomos de superfície, a implementação de algoritmos ML neste contexto é raro, mas está a surgir cada vez mais dada a sua importância (Tachibana & Fukazawa, 2016).

[2] *Machine Learning* pode ser traduzido em português por "Aprendizagem Máquina", mas nesta dissertação optámos por usar o termo inglês, por estar mais difundido, mesmo na comunidade portuguesa.

1.2. Âmbito da Dissertação

A EN tem desenvolvido veleiros autónomos no âmbito do projeto eVentos desde 2010 (Cavaco, 2011). Neste projeto foram feitas várias parcerias com a FEUP para ajudar a desenvolver o veleiro FAST[3], e foram feitos testes com pequenos veleiros Radio Controlados (RC). No entanto, verificou-se que havia conveniência em desenvolver um veleiro com mais espaço para componentes eletrónicos, mais robusto para aguentar condições de mar adversas e desenvolvido inteiramente pela EN. Assim, propus-me colaborar neste projeto, desenvolvendo uma nova plataforma que englobasse conhecimentos multidisciplinares ensinados em diversas cadeiras na Escola Naval.

O projeto teve vários aspetos inovadores, e implicou a aplicação de boas práticas de engenharia. O primeiro passo foi o dimensionamento do casco e estudos hidrostáticos e hidrodinâmicos do mesmo (capítulo 2, secções 2.1 a 2.7). Seguidamente houve um período de construção física do casco a partir de uma estrutura em aço laminado e materiais compósitos (capitulo 2, secção 2.8). O segundo passo foi fazer o projeto dos sistemas periféricos recorrendo a ferramentas CAD e impressão 3D (capitulo 2, secção 2.9). Foi necessário desenvolver toda a eletrónica de controlo (Capitulo 3) e a programação necessária para o veleiro (Capítulo 4). Depois encontram-se os resultados obtidos (Capitulo 5) e as conclusões deste trabalho (Capitulo 6).

Este projeto tinha entre outros objetivo a participação na regata *World Robotics Sail Conference* (WRSC16) e a apresentação de um artigo nessa conferência internacional. De facto, os objetivo acabaram por ser superados pois o desenvolvimento do veleiro foi integrado nas atividades do *studant chapter* de Almada da AFCEA (do qual o autor é presidente), dando origem a um artigo no seminário da AFCEA Portugal, outro na SEA-CONF16 da Escola Naval Romena e um terceiro na conferência MARTEC16. O quarto e último artigo realizado durante o mestrado foi publicado nos *conference proceedings* do WRSC16, perfazendo um total de três artigos em conferências internacionais, e um nacional (Apêndices M, N e O).

[3] Em 2013 a EN, com o então veleiro Blaus VII, apoiou e escoltou um trànsito autónomo entre Cascais e Sestimbra, e no mesmo ano conduziu testes com o FAST ao largo da Costa da Galé, no âmbito do projecto ROBONOISE. Em 2015 um aspirante da EN fez a sua tese no sistema de gestão de energia do FAST.

A participação na competição WRSC16 foi a primeira de uma equipa da Marinha Portuguesa, e embora tenha ficado em 5º lugar na sua classe, conseguiu pontuar em duas das quatro competições. Esta prova revelou-se uma mais-valia para a maturidade do veleiro, e foi publicamente reconhecida pelos parceiros como um projeto de relevo.

2. Plataforma de Testes – Desenvolvimento de um Veleiro de Pequena escala

2.1. Introdução

Navegar à vela significa estar mais dependente das condições ambientais para atingir o objetivo. Muitos fatores influenciam o comportamento do veleiro, mas o vento e a ondulação são os mais preponderantes. O vento condiciona a navegabilidade, na medida em que a sua direção condiciona a eficiência das velas. O escoamento ao passar pela vela origina duas forças principais: arrasto e sustentação. A conjugação da proa do veleiro e do ângulo da vela, em função da velocidade e direção do vento ditam a magnitude da força de sustentação e de arrasto que são geradas, ou seja, a força que o navio dispõe para ir a vante e a força que adorna o veleiro. A forma e tamanho das velas influenciam igualmente este resultado. A força do vento pode conduzir à cedência dos materiais da vela, originando estragos. A ondulação e vaga afetam a estabilidade e velocidade do veleiro. Dividindo o espectro de frequências e amplitude das ondas nas suas componentes longitudinais *x* (vante/ré) e transversais *y* (estibordo/bombordo), é possível tirar as seguintes conclusões: as componentes em *y* levam o navio a oscilar, podendo atingir adornamentos para além do máximo com estabilidade positiva; com componentes em *x*, se o sentido de propagação for de vante para ré, conduzem a perda de velocidade, e se o sentido de propagação for de ré para vante conduzem a um incremento de velocidade (Journée, 2002). A intensidade com que estes efeitos são sentidos depende não só da ondulação e vaga, mas também da forma e tipo de casco (Misra, 2015).

Dado os fatores ambientais não puderem ser controlados, é importante desenvolver a plataforma no sentido de esta ser o mais permissiva possível a fatores externos e passível de atingir o seu propósito em todas as condições, navegar de forma eficiente.

2.2. Definição das especificações e Dimensionamento do Casco

Considerou-se como prioritário a capacidade de resistência do veleiro ao ambiente imprevisível onde opera, mantendo a dimensão do mesmo reduzida, de modo a que possa ser transportado com facilidade. Assim sendo, definiu-se a dimensão fora a fora do veleiro de 2000mm, e iniciou-se a partir daí o *design* do mesmo.

O próximo passo na definição das especificações do veleiro foi escolher qual o deslocamento (Δ) do veleiro. Quanto maior for a massa do veleiro, maior será a sua

inércia. Em certas situações, a inércia pode ser benéfica, pois, tendo o veleiro maior dificuldade em alterar o seu estado de movimento, menor é a perda de velocidade causada, p.e, por vagas, contudo, sendo maior a inércia, maior será a dificuldade em o veleiro ganhar velocidade ou efetuar guinadas. Ficou decidido que o deslocamento do veleiro seria 20kg. Este valor é bastante reduzido tendo em conta a dimensão fora a fora do casco, o que condiciona o tipo de casco e os materiais utilizados na sua estrutura. Sabendo a massa total que queremos que o sistema tenha, é possível dimensionar o casco do mesmo, sendo o objetivo a estabilidade e as condicionantes a resistência mecânica e a dimensão fora a fora do veleiro.

A força de impulsão é igual em módulo e de sentido oposto ao peso do fluido deslocado (Lewis, 1988). Sabemos que o peso do fluído deslocado é o produto da sua densidade (ρ), pelo volume deslocado (∇) e pela aceleração da gravidade, sendo que a aceleração da gravidade pode ser simplificada pois entra em ambos os lados da equação (Misra, 2015), resultando a Equação 1.

$$\Delta = \rho \nabla$$

Equação 1 – Deslocamento em função do Volume da Carena e da densidade

Deste modo, é possível chegar ao volume que teremos de deslocar, volume da carena, para atingir a impulsão necessária.

Se sobre o veleiro atuar unicamente a impulsão e o peso sobre a mesma vertical, o veleiro está numa situação de equilíbrio e nada acontece. No mundo real, esta situação é extremamente improvável. Sobre ele atuam inúmeras forças, sendo as principais o vento, a ondulação e a vaga. De acordo com a localização do ponto de aplicação dessas forças, são gerados momentos que adornam o veleiro, sendo necessário existir um momento restituidor. As relações entre os momentos inclinantes e os momentos endireitantes definem o comportamento do casco no mar (Mizine, Karafiath, Queutey, & Visonneau, 2009). Neste sentido, queremos que os momentos inclinantes sejam os mais pequenos possíveis e que os momentos endireitantes sejam o maior possível. Este jogo é possível de ser alcançado através do *design* do casco. O sistema de eixos utilizado é o que está representado na Figura 3.

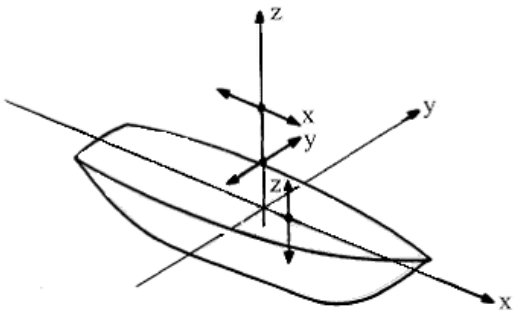


Figura 3 - Sistema de Eixos Utilizado

O objetivo é que pequenos ângulos de adornamento gerem grandes momentos endireitantes. Assim sendo, o metacentro tem de ser o mais elevado possível. A altura do metacentro transversal (Z_M) de um navio (Lewis, 1988) pode ser expressa através da relação expressa na Equação 2.

$$Z_M = Z_B + \frac{I_y}{\nabla}$$

Equação 2 - Altura do metacentro

Onde Z_B é a altura do centro de carena e I_y é o segundo momento de inércia da área da figura de flutuação. Para o navio ser estável a um dado ângulo de adornamento, a altura do metacentro tem de ser superior à altura do centro de gravidade (Z_G) (Lewis, 1988). Esta relação é conhecida como a condição de estabilidade e encontra-se representada na Equação 1.

$$Z_G < Z_M$$

Equação 3 - Condição de Estabilidade

Assim sendo, quanto mais elevada for a altura do metacentro e quanto mais reduzida for a altura do centro de gravidade, maior a estabilidade. Uma forma de aumentar a altura do metacentro é aumentar o segundo momento de inércia da área da figura de flutuação. Esta quantidade pode ser calculada tal como na Equação 4.

$$I_y = \int_A y^2 dA$$

Equação 4 - Segundo Momento de Inércia

Esta relação diz que, quanto maior a dispersão de área no eixo y, maior será a altura metacêntrica, ou seja, quanto maior a boca do veleiro, maior a sua estabilidade. Contudo, tendo já definido o comprimento e o deslocamento do veleiro, é complicado alcançar uma

boca grande, porque, para o mesmo deslocamento e comprimento, aumentar a boca significa diminuir o calado. Sendo este casco para um veleiro, o calado tem um papel de relevo, pois, este aumenta a resistência ao arrasto do veleiro por ação do vento.

A solução adotada foi a forma do casco em V (Mizine, Karafiath, Queutey, & Visonneau, 2009). Esta forma garante baixa resistência hidrodinâmica e menos perda de velocidade causada por vagas (Misra, 2015), porém, não garante a dispersão suficiente de área no eixo y para o veleiro ter estabilidade necessária. Optou-se por manter a forma em V e por aumentar ao tamanho do patilhão, a fim de reduzir ZG, por forma a cumprir com a condição de estabilidade

2.3. Desenho e Simulação em CAD 3D

Utilizou-se o programa *DELFTship™Free* para desenhar o casco principal. Este programa é bastante interativo e versátil, permitindo de forma expedita dimensionar o casco e calcular a resistência hidrodinâmica e parâmetros hidrostáticos do mesmo. A Tabela 1 sumariza as dimensões do casco principal.

Tabela 1 - Principais dimensões do casco

Boca máxima	300mm
Calado	170mm
Comprimento total	1960mm
Deslocamento nominal	21kg
Deslocamento máximo	34kg
Reserva de flutuabilidade	6kg
Coeficiente de bloco	0,1558
Coeficiente Prismático	0,6272

A Figura 4 mostra o casco principal desenhado no *software*.

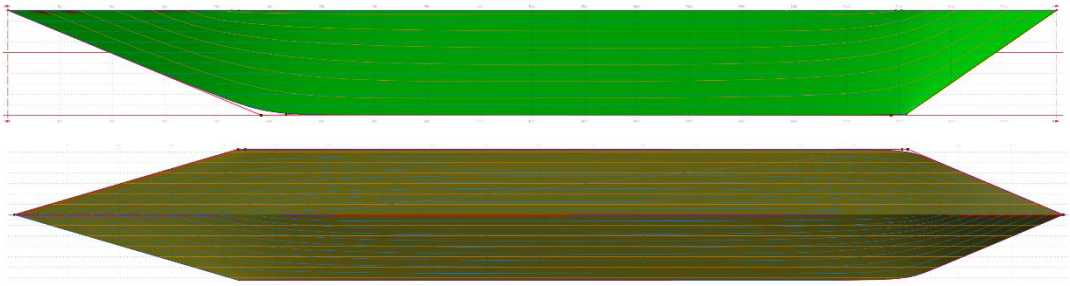


Figura 4 - Desenho da forma do casco

Fazendo uso do mesmo *software* calculou-se a resistência do casco com imersão de 170mm, que corresponde à situação nominal de 21kg de deslocamento. O programa utiliza no cálculo da resistência as séries de *Delft* (Journée, 2002). O gráfico apenas mostra a resistência do casco, não contando com a resistência do leme, nem do patilhão e está representado na Figura 5.

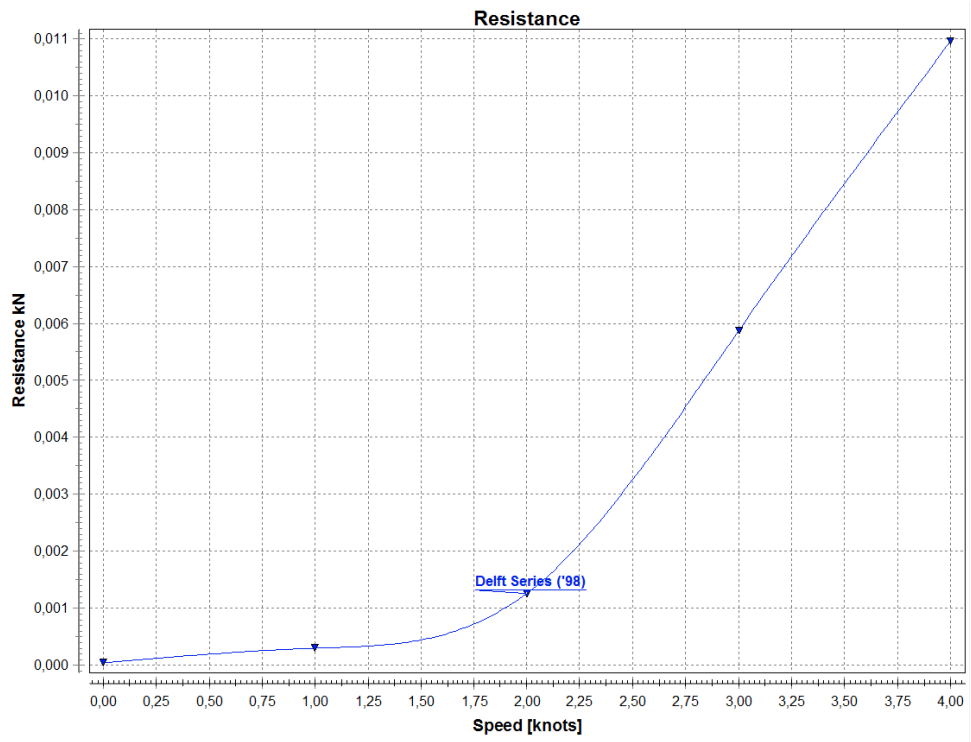


Figura 5 - Resistência ao avanço do casco

Vemos que a resistência ao avanço é bastante reduzida, como seria de esperar, tendo em conta a forma de casco adotada. Este programa permitiu ainda a realização de alguns estudos hidrostáticos e hidrodinâmicos de acordo com o Apêndice Q. Verificou-se que a altura metacêntrica para esta forma de casco é de aproximadamente 16 cm. Contudo, por forma a assegurar a estabilidade em todos os ângulos de adornamento,

decidiu-se que a altura do centro de gravidade iria ficar abaixo da quilha. Desta forma, a condição de estabilidade será sempre assegurada.

Tendo já definido todas as dimensões e forma do casco, recorreu-se ao programa CAD 3D *SolidWorks Student Edition* para desenhar todas as peças que constituem o veleiro. Nesta fase do projeto considerou a resistência mecânica do veleiro e a posição do centro de gravidade, por forma a consegui que o veleiro seja robusto e estável.

2.4. Projeto da Quilha

O primeiro estágio da construção do casco foi a construção da quilha. O material utilizado foi aço laminado, dado as características de resistência mecânica que possui. Este material tem massa volúmica elevada, contudo, reduziu-se a utilização deste ao mínimo para atingir a resistência necessária. Por outro lado, a massa desde componente pode contribuir para a estabilidade do veleiro. Sendo a condição de estabilidade $Z_G < Z_M$, se o material com maior massa volúmica for colocado em cotas mais baixas, o centro de gravidade baixa igualmente a sua cota, contribuindo assim para a estabilidade. O objetivo foi encontrar um equilíbrio entre a massa deste componente por forma a maximizar as características de estabilidade e resistência mecânica, mantendo a massa total do casco o mais reduzido possível. O perfil utilizado foi um perfil "T", com as dimensões representadas na Figura 6.

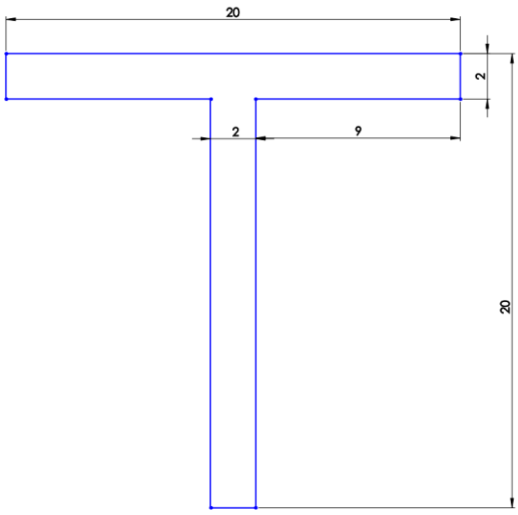


Figura 6 - Perfil utilizado na construção da quilha

Este perfil é bastante utilizado em estruturas pela relação que oferece de resistência a cargas versos quantidade de material utilizado. Todos os esforços que o veleiro sofra, serão transmitidos a este componente da estrutura. Foram montados três apoios diretamente na quilha: um apoio por mastro, totalizando dois; e um apoio para o patilhão. Dado que os esforços dinâmicos nestes componentes serão bastante elevados, estes apoios foram igualmente construídos em aço laminado soldados diretamente à quilha.

Os apoios dos mastros são constituídos por uma viga vertical, encastrada na quilha e soldada no topo a outra viga, horizontal, que contêm o guia do mastro. O guia do mastro é de perfil tubular de diâmetro exterior 25mm e espessura 1,5mm e está soldado entre a viga horizontal e a quilha, sendo que é cortado a 50mm da quilha. A estrutura foi projetada desta forma por necessidades previstas no controlo do mastro que serão explicadas em capítulo posterior. A quilha contêm dois destes apoios. Está representado na Figura 7 um destes suportes.

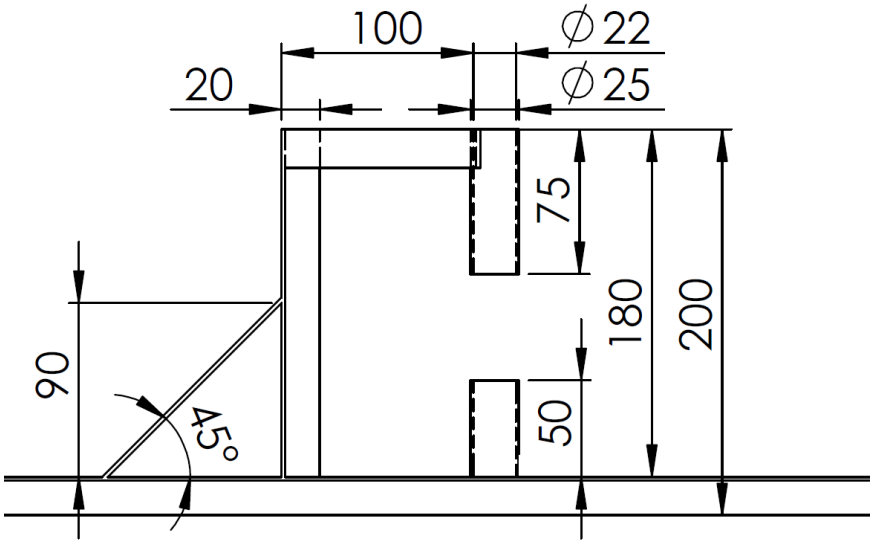


Figura 7 - Suporte dos Mastros

A quilha contêm ainda o apoio para o patilhão. O patilhão mede 1250mm e foi dimensionado para conter um bolbo com 5kg de lastro na sua extremidade. O perfil do patilhão está representado na Figura 8.

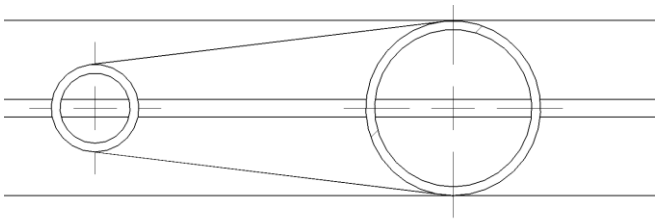


Figura 8 - Perfil do Patilhão

Por facilidade de transporte do casco, foi estudada uma solução para segmentar o patilhão em duas partes. A primeira parte, de 125mm, fica soldada à quilha, a outra parte,

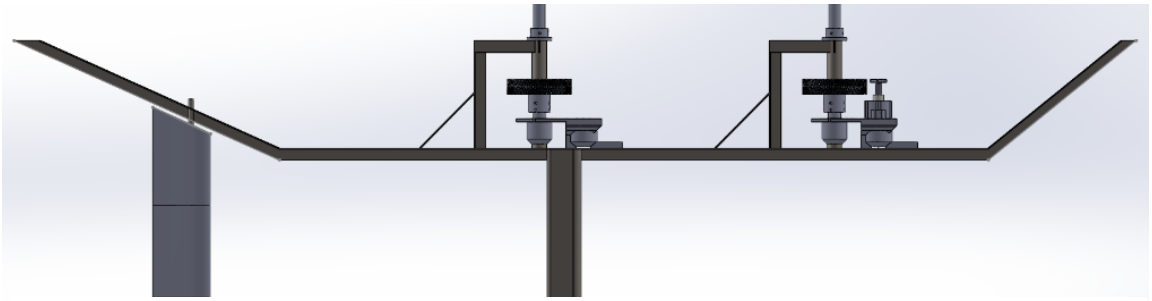


Figura 9 - Arranjo Geral da Quilha

de dimensão que completa os 1250mm totais, encastra através de um vão roscado na primeira parte, a quando da utilização do casco. A quilha ficou então com o formato que se apresenta na Figura 9.

2.5. Projeto do Leme

O leme desempenha uma função importantíssima. Para além de permitir ajustar a proa, este componente também diminui o arrasto lateral do veleiro por ação do vento. A forma típica do leme é *foil* (Vacanti, 2005). Colocando o leme a um determinado ângulo, o escoamento fica com diferentes velocidades nas faces do leme, gerando uma força que faz o veleiro guinar, força de sustentação. É gerada ainda uma força de arrasto que reduz a velocidade do veleiro. A escolha do *foil* a utilizar foi feita recorrendo ao programa *Designfoil Demo* (DreeseCODE Software, 2016). Este programa permite fazer simulações obtendo os coeficientes de sustentação, arrasto 2D e o centro de pressão de um dado perfil, para cada ângulo de ataque. Na Figura 10 encontra-se desenhado o perfil do leme.

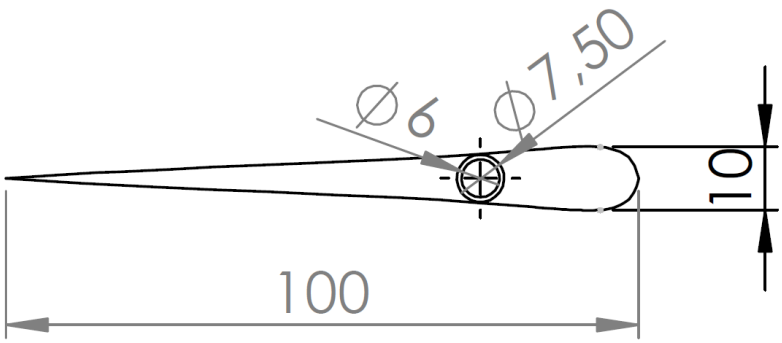


Figura 10 - Perfil do Leme

O tamanho do leme pode ser calculado através do critério de Det Norske Veritas (Vacanti, 2005), expresso na Equação 5.

$$A_r \approx \frac{d.L_{pp}}{100}\left\{1 + 25\left(\frac{B}{L_{pp}}\right)^2\right\}$$

Equação 5 - Critério de Det Norske Veritas

Na qual: Ar = Área do Leme; Lpp = Comprimento entre Perpendiculares; B = Boca e d = Calado

Esta equação prevê o valor mínimo da área de leme. Caso o leme não esteja colocado diretamente atras do hélice, a área deve ser aumentada no mínimo em 30%. O resultado da equação para o caso do veleiro em estudo, já com o incremento devido, é de 48cm^2. É importante que o leme tenha dimensão ajustada, pois, demasiado pequeno não é suficiente para guinar o navio, demasiado grande gera demasiada força de arrasto, condicionando a velocidade do mesmo. Resolvendo a equação com os parâmetros definidos para o veleiro em projeto, a área do leme deve ser, não inferior a 20000mm^2.

O leme está em contato permanente com a água, sendo uma zona de risco de entrada de água para o interior do casco através do bucim da madre do leme. Para minimizar este risco, na parte inferior e superior da guia da madre do leme serão colocados retentores de borracha, comprimidos por uma união roscada colocada na parte superior da madre do leme.

2.6. Projeto do Patilhão

Pretende-se que o patilhão tenha um alongamento elevado, pois, quanto mais elevado este coeficiente for, menos força de arrasto longitudinal é gerada, mantendo a força de arrasto lateral suficiente para que o veleiro não derive por ação do vento (Vacanti, 2005). Por razões de estabilidade também é benéfico que este coeficiente seja elevado, uma vez que, tendo o patilhão um bolbo na sua extremidade, quando maior este for, mais baixo será o centro de gravidade, melhorando a estabilidade. O problema que se coloca é o dimensionamento do mesmo por forma a equilibrar o arrasto longitudinal, transversal e massa deste componente.

O *foil* utilizado neste componente é igual ao *foil* utilizado no leme, contudo, sendo que este componente é mais alongado, prevê-se problemas de resistência dos materiais. Assim sendo, utilizou-se na sua construção um perfil de duplo tubo de aço laminado.

O bolbo será do tipo projétil encastrado na extremidade do patilhão. Este componente será impresso em 3D, e revestido o *GelCoat*[4] por forma a assegurar a sua estanqueidade. No seu interior terá materiais de grande massa volúmica por forma a obter-se a massa de 5kg.

2.7. Mastro e Velas

Pretendeu-se criar uma plataforma passível de melhoramentos. Assim sendo, desenvolveu-se um sistema que permite a fácil troca do sistema mastro e vela. Este sistema é útil pois permite a troca deste conjunto em caso de este se danificar e permite experimentar diferentes tipos de vela, variando a altura do mastro e/ou o comprimento da retranca, estando apenas limitado o diâmetro do mastro. O controlo do ângulo do mastro é feito através de uma correia dentada ligada entre uma roda dentada encastrada no mastro e um sistema de motor elétrico com caixa redutora. O controlo do ângulo é efetuado por um potenciómetro ligado à roda dentada do mastro.

Optou-se por desenvolver umas velas simples de montar, pois, mais tarde poderia ser feito o melhoramento da mesma. Neste sentido, desenhou-se em CAD o perfil da vela

[4] *Gelcoat* é uma resina, geralmente de *Epoxy* ou *Poliester*, utilizada por norma para efetuar revestimento de materiais compósitos.

pretendido. Depois seriam impressos vários destes perfis, unidos através de um varão de alumínio de diâmetro 16mm e forrados com plástico Termo retrátil. Esta solução serial estruturalmente menos resistente que o suporte do mastro. Desta forma garante-se que a vela funciona como um fusível mecânico, danificando-se primeiro, antes de comprometer o sistema de controlo dos mastros, que é mais complexo e de substituição mais difícil.

2.8. Construção do Casco

2.8.1. Construção da Quilha e dos Reforços Metálicos

A construção do casco iniciou-se através da construção da quilha. Foram medidos e cortados os perfis necessários, e iniciou-se a soldadura dos mesmos. A soldadura foi realizada através de arco elétrico. O elétrodo utilizado foi de 2.0mm e utilizou-se uma corrente de 100A a 24V. Na Figura 11 encontra-se a primeira parte construída, a roda de proa.

Figura 11 - Quilha, pormenor da roda de proa

Após a conclusão da roda de proa, e ajuste do ângulo por forma a corresponder com o projeto, iniciou-se a construção da parte mais a Ré da quilha. Foi soldado diretamente à quilha o guia do leme. O objetivo foi que a parte superior do guia do leme fica-se acima da linha de água projetada, 170mm, por forma a, ser uma salvaguarda no que diz respeito à entrada de água para o interior do casco. Na Figura 12 encontra-se representado o arranjo geral desta parte do componente.

Figura 12 - Guia do Leme

O passo seguinte foi a construção do guia dos mastros. Para isso, previamente cortou-se todos os perfis e verificou-se se seria possível a soldadura de acordo com o planeado. Verificado a exequibilidade, iniciou-se a soldadura do mesmo. Iniciou-se por soldar verticalmente o guia do mastro, um perfil tubular de diâmetro externo 25mm e 1,5 mm de espessura. Tendo em conta a espessura deste perfil, reduziu-se a corrente de soldadura para 80A, por forma a evitar o colapso da mesma durante a soldadura. Após concluída a soldadura deste perfil, iniciou-se a soldadura dos perfis de reforço de acordo com o projeto. Nesta fase, o guia do mastro ficou com o aspeto representado na Figura 13.

Figura 13 - Guia do Mastro

Para finalizar a estrutura metálica da quilha, iniciou-se a soldadura da estrutura de reforço do patilhão. Para isso, soldou-se um varão roscado de 10mm de diâmetro e

1000mm de comprimento na localização do bordo de ataque. Este componente tem a função de fixar o bolbo. Outra aplicação deste componente advém da necessidade de, à posteriori, se dividir o patilhão por forma a facilitar o transporte. Quando se efetuar essa divisão, será este componente que permitirá a junção das duas partes. Finda esta soldadura, utilizou-se dois varões metálicos de diâmetro 16mm e 6 mm para efetuar, respetivamente, o bordo de ataque e de fuga do patilhão. Por fim, soldou-se ligações entre os dois bordos, por forma a aumentar a resistência mecânica dos mesmos. O resultado final encontra-se na Figura 14.

Figura 14 - Estrutura metálica de reforço da quilha

2.8.2. Construção do Convés

O próximo passo foi a construção do convés do veleiro. Para isso foi realizado um molde em poliestireno azul[5]. Verificou-se esse molde não estava de acordo com a tolerância pretendida, tendo sido rejeitado. Na Figura 15 encontra-se o molde rejeitado.

[5] Tipo de esferovite utilizada em isolamento.

Figura 15 - Molde rejeitado por defeito na Geometria

Construi-se outro molde, em madeira. Este molde foi cortado numa mesa de corte de madeira profissional, sendo que a sua geometria ficou correta, tendo sido utilizado para a construção do convés. Colocou-se uma camada de plástico fino, por forma a evitar que os materiais compósitos ficassem presos ao molde. Em seguida, colocaram-se três camadas de fibra de vidro, previamente impregnada em *Epoxy*, biaxial e com 380g/m^2. Por não ser possível a utilização de vácuo, colocou-se novamente plástico a envolver o molde, e pressionou-se o mesmo até a cura estar efetuada, a fim de evitar a delaminação do material. Findas 6 horas de cura, a estrutura foi retirada do monde. Verificou-se que a estrutura tinha robustez, e que a sua geometria estava de acordo com os padrões pretendidos. Na Figura 16 encontra-se uma imagem da utilização deste molde.

Figura 16 - Molde em madeira com a estrutura metálica da quilha

O próximo passo foi a instalação do convés na estrutura metálica da quilha. Verificou-se importante acrescentar à estrutura metálica um reforço transversal nos guias dos mastros, por forma a garantir a esquadria do veleiro, e melhor assegurar a ligação entre o convés e a quilha. Efetuada a soldadura desses reforços, foi colocado o convés na

respetiva posição, e preso através processo de costura por fio de aço. Na Figura 17 é visível o aspeto final que tomou o veleiro após este processo.

Figura 17 - Ligação entre o Convés e a estrutura da Quilha

2.8.3. Construção do costado e revestimento do patilhão

Findas todas as soldaduras e trabalhos ligados à construção do convés, iniciou-se a construção do costado do veleiro e revestimento a compósito do patilhão. Esta fase foi a mais complicada de todas, pois, não existia nenhum molde, dificultado o processo de construção. A estratégia de construção utilizada foi prender seguimentos de fibra de vidro biaxial com 380g/m^2 previamente impregnados com *epoxy* entre o convés, passando por baixo da quilha, e voltando ao convés no bordo contrário. O objetivo era existirem três camadas de fibra de vidro, à exceção de meio navio, onde existiria uma camada extra para reforço estrutural, devido à concentração de tensões de esforços dinâmicos no patilhão.

A primeira camada foi a mais complicada de aplicar, porque se verificava que a fibra de vidro não aderia totalmente ao convés, não ficando o tecido completamente esticado. Cuidadosamente e recorrendo pincéis pequenos, foi possível colocar em todo o costado a primeira camada de fibra, completamente esticada. Esperou-se 6 horas por forma ao compósito estar totalmente curado para verificar a consistência do trabalho efetuado. Tal como é possível verificar na Figura 18, o veleiro começava a ganhar forma.

Figura 18 - Primeira Camada de Fibra no Costado

Paralelamente, iniciou-se o revestimento do patilhão. Colocou-se fibra de vidro em volta da estrutura metálica. Impregnou-se a fibra e revestiu-se com material plástico não aderente à resina, por forma a pressionar as camadas de fibra e evitar a delaminação do material.

Pequenas rugosidades existentes na primeira camada foram corrigidas com recurso a material de enchimento de fibra de vidro, impregnado em Epoxy. Continuou-se o revestimento do costado, contudo, cada camada adicional era revestida com plástico por forma a pressionar contra as camadas anteriores e evitar a delaminação. Antes de iniciar nova camada, este material plástico era removido, e a superfície limpa.

2.8.4. Revestimento com Gelcoat e Processo de Pintura

Deu-se muita importância à estanquidade do veleiro. Num esforço para garantir a longevidade do mesmo, foi selecionado um material *Gelcoat* à base de *epoxy* para fazer um revestimento estanque do mesmo. Este material tem propriedades excelentes de aderência, resistência mecânica e estabilidade química.

Esta fase foi igualmente complicada. A primeira camada aplicada foi demasiado espeça, sendo que a superfície não ficou completamente lisa, devido a, parte do *Gelcoat* ter escorrido antes de se ter dado a cura total do material. O veleiro teve de ser todo lixado, antes de se realizar nova pintura. Verificou-se que a nova pitura repetiu o problema anterior, tendo-se repetido o processo de lixa e alterado o processo de pintura. Passou-se a realizar a pintura através de rolo de pintura ao invés de trincha. O problema foi solucionado, e foram aplicadas três camadas extra de *Gelcoat* finas. Devido aos

problemas de pintura iniciais, verificou-se que o veleiro ficou com algumas irregularidades na pintura, contudo, a estanquidade estava assegurada, pelo que não se realizou nova lixa e pintura do casco, tendo sido estabelecido que o objetivo primordial de assegurar a estanquidade estava atingido, e que o problema de estética seria colocado de parte devido ao tempo que iria demorar a corrigi-lo. Nesta fase, o veleiro tinha o aspeto apresentado na Figura 19.

Figura 19 - Casco revestido a gelcoat de cor branca

Após este processo, realizou-se a pintura do casco recorrendo a *sprays* em lata de várias cores. O objetivo era obter uma pintura do tipo camuflado pixilizado em tons de azul. Para isso, recortou-se em papel autocolante vários moldes de pintura em forma de pixel, de diferentes tamanho e formas. Pintou-se o tom azul mais escuro, colaram-se alguns desses moldes, pintou-se novamente o veleiro no segundo tom mais escuro, repetindo o processo até ter o casco totalmente pintado. Na Figura 20 está explícito o resultado final.

Figura 20 - Casco Após Processo de Pintura

Após a remoção de todos os moldes de pintura colocados, finalizou-se o processo de pintura com um verniz especial de proteção, que evita o desgaste das camadas de tinta de *spray* e protege contra a ação dos raios ultravioleta da luz solar.

O interior do casco foi revestido com madeira balsa, e foi colocado espuma expansiva poliuretano para assegurar que, mesmo em avaria com o casco totalmente cheio de água, este ainda garantia a sua flutuabilidade. Foram também construídas anteparas estaques a delimitar seis compartimentos distintos, por forma a assegurar que a entrada de água para um dos compartimentos não se alastrava ao casco todo.

2.9. Tecnologia CAD 3D e 3D *Printing*

Esta fase foi desenrolada em paralelo com a construção do casco, por forma a garantir a adaptação dos sistemas desenvolvidos à plataforma construída. Foram utilizadas um total de 42 peças desenvolvidas em CAD e impressas em 3D por método de extrusão. As peças desenvolvidas têm várias funções, desde o comando dos mastros e do leme até às portas de visita dos diferentes compartimentos. O *software* de desenho utilizado foi o *Solidworks 2014 Student Edition*, pela versatilidade e ferramentas que dispõe. Este programa permite guardas os projectos em *.stl[6]. É importante este formato, pois é o formato nativo de uma outra aplicação, *Repetier-Host* (Littwin, 2016). Esta

[6] *.stl significa *STereoLithography*, é um tipo de ficheiro que permite o intercambio de projetos entre programas CAD.

aplicação permite efetuar o slice (slic3r, 2016) da peça 3D, convertendo a mesma em gCode (cnccookbook, 2016). A impressora utilizada foi PrusaI3. Esta impressora tem as características expressas na Tabela 2.

Tabela 2 - Características da Impressora 3D Prusa i3 (Mundo Reader, 2016)

VELOCIDADE DE IMPRESSÃO	Velocidade recomendada: 40-60 mm/s Velocidade máxima recomendada: 80-100 mm/s
RESOLUÇÃO	Resolução Nominal mecânica no eixo X: 0,015 milímetro Resolução Nominal mecânica no eixo Y: 0,015 milímetro Resolução Nominal mecânica no eixo Z: 0.781µm
MATERIAIS	Extrusora com *design* próprio Dissipador com aspas e ventilador axial Boquilha de 0,4 mm para filamento de 1,75 mm Bico de refrigeração da peça
ELECTRÓNICA	*Ramps* 1.4 Ecrã LCD com *encoder* rotativo com pulsador para a navegação Termístores 100k na extrusora Mega 2560 Base fria em vidro tamanho 220 x 220 x 3 mm Fonte de alimentação de 220 AC 12 DC 100W Cartucho aquecedor 40W 12V
SOFTWARE	*Firmware* derivado do Marlin Ficheiros admitidos: .gcode Ambiente recomendado: *Cura Software*, *Slic3r*, *Repetier*, *Kisslicer* Sistemas operativos compatíveis: Windows XP e superiores, Mac OS X e superiores e Linux

2.9.1. Controlo dos Mastros

O controlo dos mastros é um aspeto de elevada importância, sendo necessário que a estabilidade e fiabilidade do sistema seja elevada, por forma a se garantir que o veleiro é capaz de ajustar o ângulo das velas de forma precisa, tirando maior partido das condições envolventes, navegando de forma eficaz e eficiente. Ficou estipulado que os mastros seriam comandado através de rodas dentadas, e que o feedback da posição das mesmas seria efetuado através de um potenciómetro.

Iniciou-se o projeto do sistema de controlo dos mastros através da definição do tipo de rodas dentadas que se iria utilizar. Recorrendo a estudos prévios, definiu-se que se iria utilizar rodas dentadas helicoidais duplas, também conhecidas como dentes em espinha (Barros, 2011). Estas rodas dentadas tem a vantagem de ter baixo escorregamento, induzindo baixas pressões axiais. Isto significa que as rodas dentadas não têm tendência a se afastarem uma da outra no sentido axial. Outra vantagem deste tido de roda dentada é a folga rotacional entre duas rodas rentadas ser pequena, o que neste sistema em especifico se traduz em maior precisão no ângulo da vela (Barros, 2011).

O próximo passo foi a definição do arranjo geral do sistema. Definiu-se que o sistema seria constituído por duas rodas dentadas principais, uma encastrada diretamente ao mastro, e outra encastrada ao motor de acionamento do mastro. Após consulta de mercado, a melhor solução encontrada foi a utilização do conjunto motor e caixa redutora de uma pistola a parafusadora marca Ikea com a referência 202.141.99, conforme Figura 21.

Figura 21 - A Parafusadora utilizada no projeto

A vantagem da utilização destes componentes advém do facto de o motor ter a cupulado um sistema de redução por engrenagem planetárias, que reduz as rotações do motor para aproximadamente 200 rotações por minuto, e aumenta consideravelmente o binário. Não existe informação específica a cerca do valor do binário. Outra vantagem é este sistema já trazer incorporado uma bateria de lítio 3.6 V/1.3 Ah, bem como a placa e transformador de carregamento da mesma. Definiu-se ainda que, embora este motor tivesse binário suficiente para rodar o mastro, através das rodas dentadas, se faria uma redução ainda maior.

Ficou decidido que a roda dentada que encastro no mastro teria 100 mm de diâmetro, por motivos de espaço, ou seja, para ser de dimensão compatível com o espaço onde irá trabalhar. Ficou ainda definido que a roda dentada pequena teria 40 mm de diâmetro. O rácio ou a redução (Smith, 2001) obtida por este sistema está expresso na Equação 6.

$$r = \frac{100}{40} = \frac{2,5}{1}$$

Equação 6 - Redução no Sistema

É possível efetuar o cálculo através do diâmetro da roda dentada porque o número de dentes utilizado é proporcional ao raio da roda dentada (Robots, 2016). O desenho deste tipo de roda dentada é muito complexo, porque tem de se ter em atenção a forma e o número de dentes nas duas rodas dentadas, para que trabalhem sem interferência, mas com a maior área de contacto possível, por forma a diminuir a pressão de contacto, e consequentemente, a probabilidade de falha da mesma. Na Figura 22 encontra-se o desenho da roda dentada que encaixa no motor do sistema.

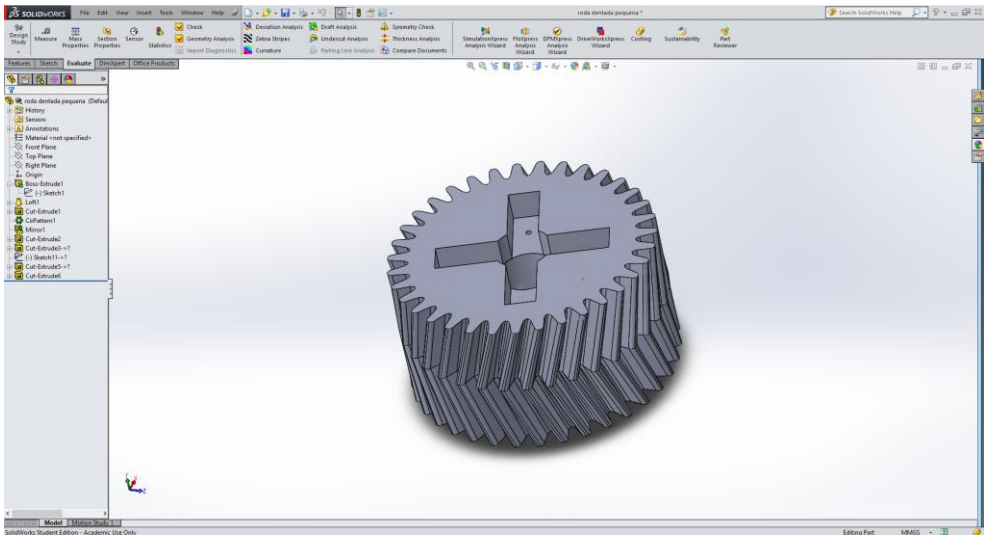


Figura 22 - Roda dentada de controlo do mastro pequena

O entalhe em forma de cruz serve para encaixe de uma peça que garante que não existe movimento relativo entre o veio do motor e a roda dentada. Esta peça foi igualmente desenhada recorrendo ao mesmo programa e encontra-se na Figura 23

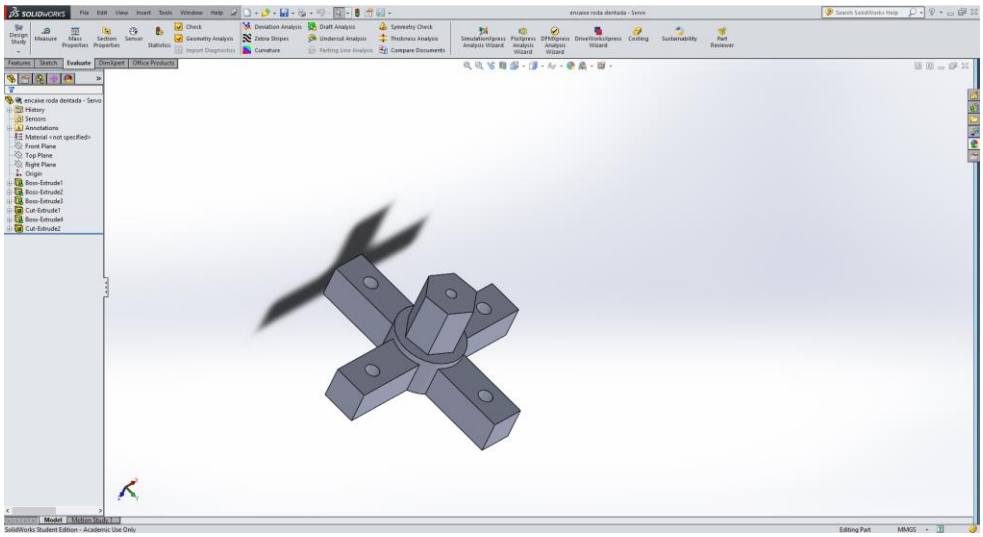


Figura 23 - Peça de fixação da roda dentada

Em seguida desenhou-se a roda dentada que fica solidária com o mastro. Esta roda dentada tem diâmetro de 100 mm. A forma e tamanho dos dentes é igual à roda dentada que encaixa no servo, por forma a assegurar o funcionamento de ambas. Foi ainda previsto e desenhado uma extensão da roda dentada em perfil tubular, por forma a esta ser aparafusada ao mastro. Na Figura 24 encontra-se o desenho da peça.

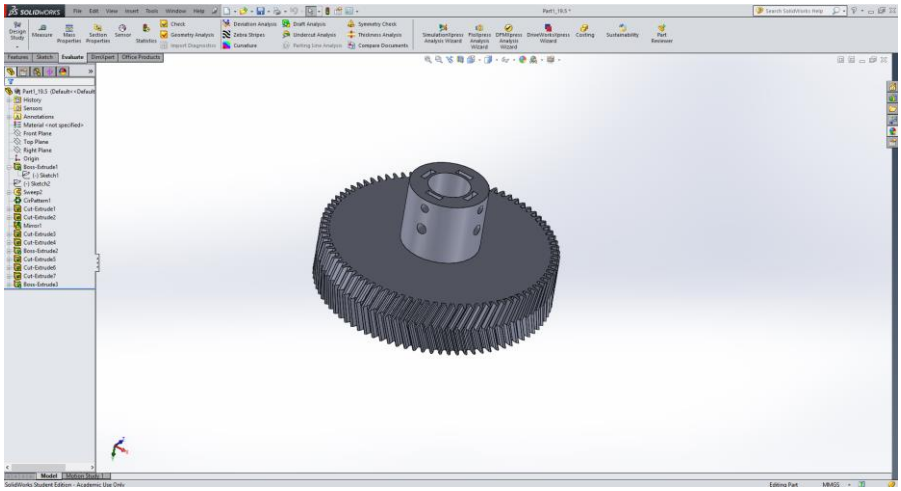


Figura 24 - Roda dentada de controlo do mastro grande

Após as duas peças estares desenhadas, foram impressas em 3D de modo a verificar a efetividade das peças. Utilizou-se PLA (Polylactic acid) para ambas as peças, com um enchimento de 20% em forma de hexágono, a temperatura foi 210 ºC e os *layers* de 0,4 mm. Devido à forma complexa que as rodas dentadas apresentam, o *slicer* das mesmas foi uma tarefa demorada, pois exige muita capacidade computacional. Na Figura 25 encontra-se a roda dentada mais pequena após *slice* e pronta a imprimir.

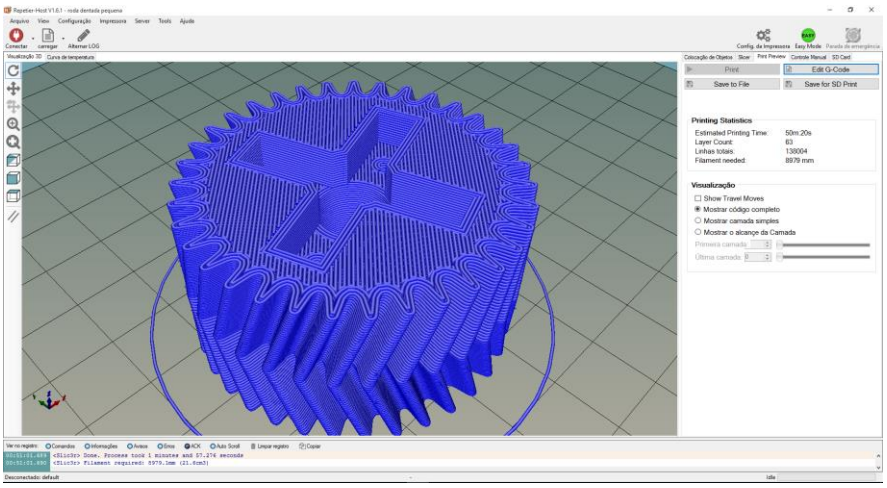


Figura 25 - Roda dentada após Slice no programa Repetier-Host, pronta a imprimir

Verificou-se que devido ao coeficiente de dilatação do material, a peça contraia ligeiramente. Foi necessário aumentar ao furo da roda dentada pequena 0,25 mm por forma a este encaixar no veio do motor. As restantes características das peças estavam satisfatórias, verificando-se que possuíam boas capacidades mecânicas, ou seja, foi efetuado um teste no qual o sistema foi montado. Verificou-se que os dentes das rodas dentadas não entravam em falha, mesmo quando se parava o veio do mastro e se colocava o motor em carga. Assumiu-se que este seria o máximo esforço que as peças estariam sujeitas, embora, seja importante em trabalho futuro realizar testes mecânicos complementares, a fim de se obter o valor máximo de binário que o conjunto suporta.

Iniciou-se então o projeto dos restantes componentes. Primeiro desenhou-se a estrutura que serviria de base a todo o sistema. Inicialmente esta estrutura era constituída por uma peça única, contudo, verificou-se que seria mais pertinentes este suporte estar dividido em duas peças, por forma a possibilitar a afinação. Nesta estrutura existe o suporte para o potenciómetro, e o suporte para as restantes rodas dentadas que fazem a ligação entre o mastro e o potenciómetro.

As rodas dentadas de ligação ao potenciómetro foram desenhadas utilizado dentes retos. Foi feita uma desmultiplicação de 2:1 entre o mastro e o potenciómetro. O potenciómetro entre a resistência máxima e mínima corresponde apenas a meia volta do seu veio. Em caso de falha do sistema de controlo do mastro, rapidamente o potenciómetro poderia chegar ao limite mecânico, danificando-se. Fazendo a desmultiplicação, garante-se que existe mais folga até este atingir o máximo. O sistema de feedback da posição do mastro tem três rodas dentadas. Uma solidaria com o veio do potenciómetro, outra solidaria com o mastro e uma intermédia que faz a desmultiplicação

das rotações. A roda dentada intermédia tem ainda a função de permitir um arranjo que permite uma fácil montagem dos componentes.

Existe ainda uma peça que faz a fixação do motor ao suporte do sistema. Esta peça é fixa através de quatro parafusos e tem como função manter o motor na posição devida, permitindo ainda o ajuste do ângulo do mesmo, por forma a um melhor encaixe das rodas dentadas. O sistema assume a forma apresentada na Figura 26.

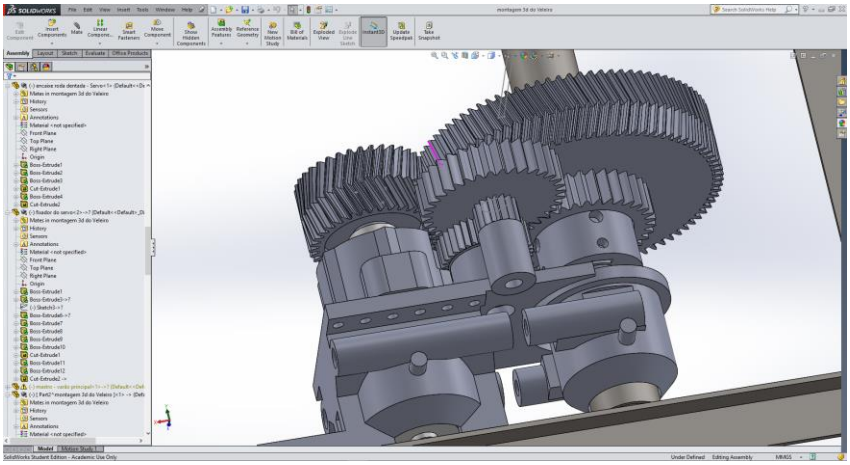


Figura 26 - Sistema de controlo do mastro

Após impressão e montagem dos componentes, o arranjo do sistema é o representado na Figura 27.

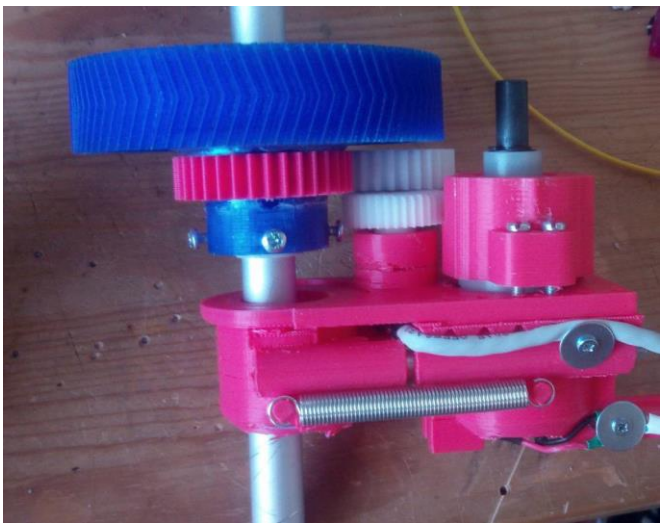


Figura 27 - Arranjo final do sistema de controlo dos mastros

2.9.2. Construção do Leme

A área do leme foi calculada de acordo com o critério de Det Norske Veritas explicado em capítulo anterior. De forma a cumprir com a área estipulada pelo critério, a

área do leme deve ser superior a 20000 mm^2. Tendo em conta que o veleiro irá operar sempre a baixa velocidade, prevê-se uma baixa eficiência do mesmo, pelo que se estipulou que se daria uma margem maior por forma a assegurar a efetividade do mesmo. Na Figura 28 encontra-se o desenho do leme em *Solidworks*.

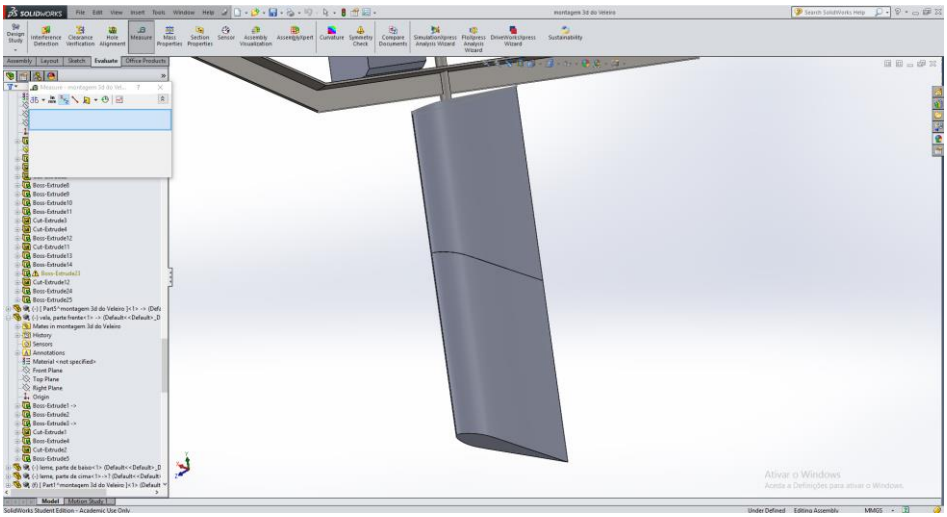


Figura 28 - Desenho do leme em CAD

A área do mesmo é 26742mm2, tendo 274mm de altura e 97,6mm de cumprimento. Esta peça foi impressa em 3D. Por limitações no volume de impressão, o leme teve de ser dividido em duas peças separadas. Após impressão, o leme foi montado e colado recorrendo a resina de *Poliéster*. No seu interior, está um varão roscado de 6mm também colado recorrendo à mesma resina. O varão roscado entra dentro do casco através do guia do leme. É aparafusado a um tirante do mecanismo de movimentação do leme. Por forma a assegurar a vedação deste sistema, foram colocados quatro vedantes de forma tórica. Um no topo do guia do leme e outro na cota mais inferior deste componente. Dois a meio espaçados 20mm. Por forma a facilitar o movimento do sistema e ajudar na redução do atrito, todo o sistema foi impregnado com massa de lítio.

O mecanismo de movimentação assenta em dois tirantes e dois braços, que transferem o binário do servo para o leme. Na Figura 29 encontra-se uma fotografia do arranjo final do sistema de acionamento do leme.

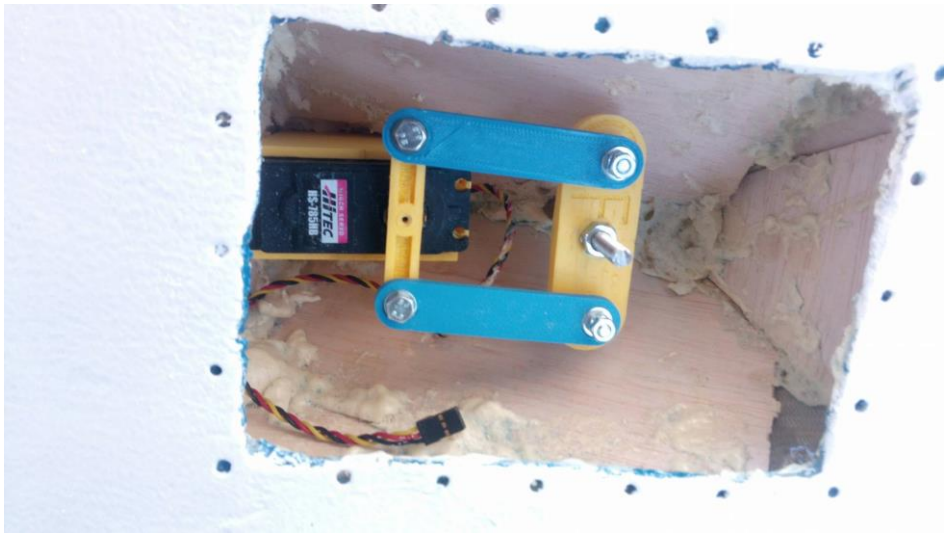


Figura 29 - Mecanismo de acionamento do leme

O servo utilizado é da marca Hitec com a referência HS-765HB. Este servo tem um binário máximo de 13,2 Kg/cm @ 6,2 V, contudo, por forma a desgastar menos o mecanismo, optou-se por utilizar a configuração 11,0 Kg/cm @4,8 V. O varão roscado fica solidário com o sistema através da utilização do sistema de dupla união roscada. Na Figura 30 encontra-se uma fotografia do sistema totalmente montado.

Figura 30 - Leme do veleiro

2.9.3. Portas de visita

Outra necessidade foi o desenvolvimento de portas de visita por forma a ser prático trabalhar no interior do veleiro. A solução encontrada foi o desenvolvimento de um suporte, encastrado ao convés do veleiro, no qual é colocada a respetiva porta de visita através de união roscada. O veleiro conta com quatro portas de visita grandes, de dimensão 180mm por 160mm e uma pequena, de dimensão 150mm por 100mm. De vante para ré, os compartimentos do veleiro são: Casa do controlo do mastro de vante; Casa das baterias e do comando dos mastros; Casa do controlo do mastro de ré, Casa dos sistemas principais e Casa da máquina do leme. Na Figura 31 encontra-se o encaixe da porta de visita. Esta peça fica presa ao convés através de parafusos e silicone, com dupla função, ajudar a fixação do componente e garantir a estanqueidade.

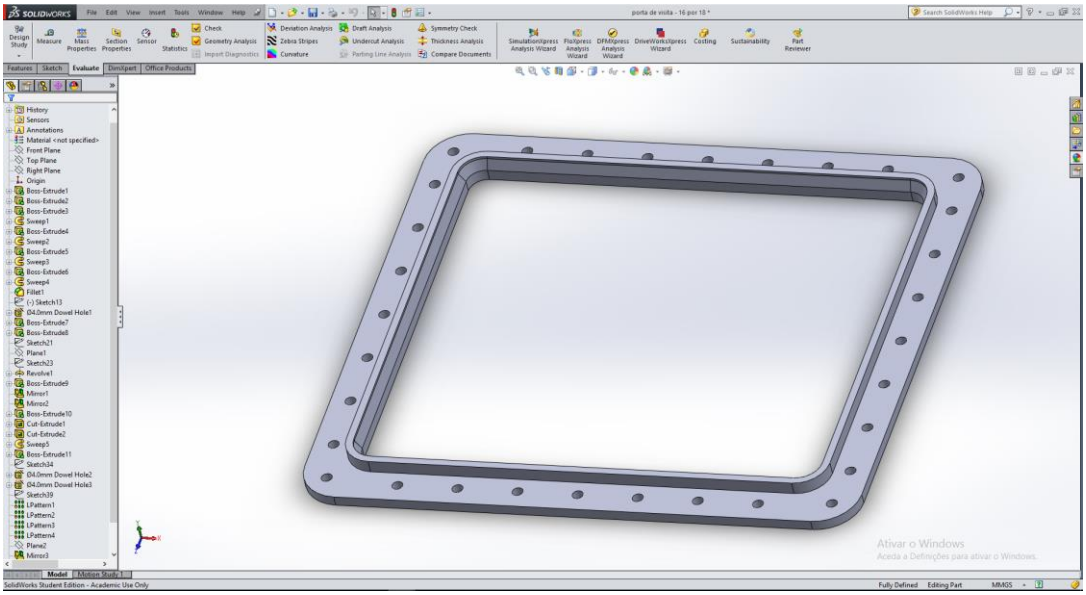


Figura 31 - Encaixe da porta de visita

Na Figura 32 encontra-se a porta de visita. Este componente foi impresso em PLA com 25% de enchimento a 220 ºC. Por forma a garantir a durabilidade do componente, este foi pintado utilizando Gelcoat e posteriormente com tinta *spray* de forma a condizer com o camuflado do veleiro.

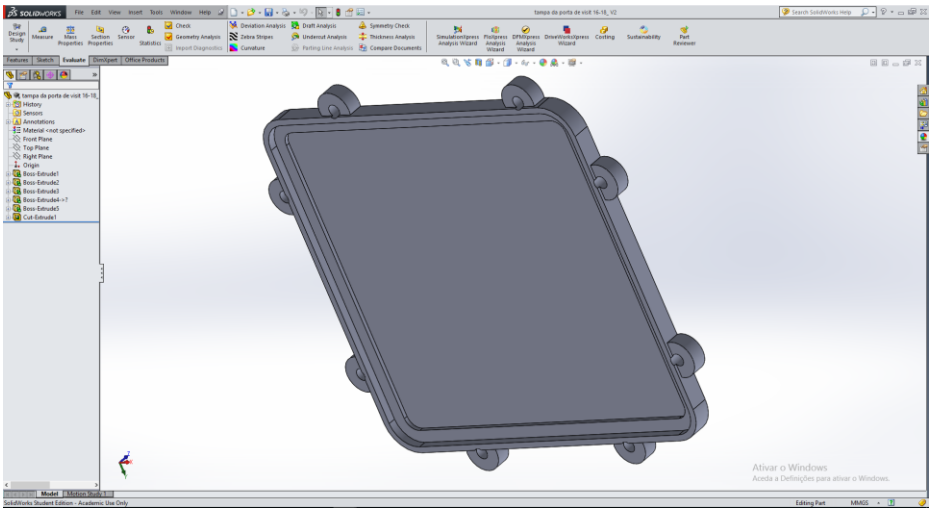


Figura 32 - Porta de visita

Na Figura 33 é possível ver as diversas portas de visita já instaladas no veleiro durante o processo de pintura. Por forma a assegurar a estanquidade, foi colocado um material de borracha entre os dois componentes. Quando a porta de visita é fechada, a borracha é esmagada e a vedação é garantida.

Figura 33 - Portas de visita durante o processo de pintura

Verificou-se que o material PLA não é o mais adequado para este tipo de peças, devendo as mesmas ser substituídas por ABS (Acrilonitrila Butadieno Estireno). O PLA perde resistência mecânica a temperaturas acima de 40ºC, tendo as portas de visita ficado com a geometria afetada após o veleiro ter sido exposto a temperaturas elevadas.

3. Eletrónica do Sistema

3.1. Introdução

Os componentes eletrónicos, sensores, atuadores e controladores, representam uma parte importante do sistema, sendo responsáveis por todas as ações tomadas pelo veleiro. Neste capítulo, serão discutidos os sensores utilizados e a relação que estes estabelecem com atuadores através do código implementado nos controladores.

O processo de projeto e implementação dos sistemas eletrónicos do veleiro iniciou-se com a escolha dos sensores a utilizar. Para isso, baseamo-nos nos sensores utilizados em projetos anteriores, tais como o lazer RC Marocup (MAritime RObotic CUP) do projeto eVentos da Escola Naval iniciado em 2010. Como sensores, o veículo tem: anemómetro, bússola eletrónica e GPS. O controlador escolhido foi o Arduíno Mega 2560, devido á versatilidade do mesmo. Como atuadores, o veleiro tem: um motor para controlo de cada um dos dois mastros e servo do leme. Para efetuar o controlo dos mastros, foi necessário a adição de um outro controlador, um *Arduíno Nano*.

Por forma a facilitar o *debug* e controlo do veleiro, foram adicionadas comunicações ao mesmo e contruída uma estação de controlo em terra. Esta estação foi desenvolvida em C#, e será abordada especificamente em subcapítulo posterior. É verdade que existem estações em terra muito desenvolvidas, tais como, *neptus* da FEUP (Dias, Gomes, & Pinto, 2008), mas isso envolveria a utilização de protocolos *standart* e mais algum processamento do controlador de bordo, pelo que se deixou este desenvolvimento para uma fase futura.

3.2. Arquitetura do Sistema

3.2.1. Microcontrolador

É necessário ter uma unidade computacional para processar os dados que vem dos sensores e tendo em conta a missão a executar, atue sobre os sensores. Existem muitas alternativas populares, como por exemplo, placas PC/104 (e.g. Consortium, 2016), a correr Linux ou mesmo Windows, ou Raspberry pi. A unidade central de processamento escolhida para este projeto foi um Arduíno Mega 2560 devido ao seu baixo custo, simplicidade de programação, baixo consumo energético, capacidade de processamento suficiente para as tarefas e ser uma solução frequentemente utilizada em projetos na Escola Naval. As suas principais características encontram-se expressas na Tabela 3.

Tabela 3 - Características do Arduíno Mega 2560 (ARDUINO, 2016)

Microcontrolador	ATmega2560
Tensão de Funcionamento	5V
Limites de Alimentação	6-20V
Pins I/O Digitais	54 (of which 15 provide PWM output)
Pins Analógicos	16
Corrente DC por I/O Pin	20 mA
Corrente DC dos Pins 3.3V	50 mA
Memória Flash	256 KB
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
Comprimento	101.52mm
Largura	53.3mm
Massa	37g

Este microcontrolador é muito versátil e permite a integração de inúmeros sensores e atuadores facilmente.

3.2.2. Sensores utilizados

Pode-se efetuar uma comparação entre os sensores do veleiro e os sentidos de um ser humano. Os sensores definem o estado envolvente ao veleiro, e é com base na informação que estes transmitem ao microcontrolador que o veleiro consegue navegar. Por este motivo, os sensores têm de ser de qualidade e estáveis, de modo a se ter um sistema fiável.

O *Global Positioning System* (GPS) tem uma função muito importante. É com base nas coordenadas GPS que o veleiro sabe a sua posição, o rumo e a velocidade. Estes parâmetros são muito importantes para o veleiro conseguir navegar de forma autónoma, pois, desta forma, consegue saber onde está, o azimute e distância para o sítio onde quer ir, a velocidade com que navega e o rumo que está a fazer.

O módulo GPS utilizado é da marca Ublox e tem a referência GY-NEO6MV2. Este módulo pode ser alimentado entre 3 a 5V, utiliza comunicação serial a 9600 Bits por

segundo e demora 26 segundos a obter a primeira posição quando visualiza todos os satélites com 130 dBm. Na Figura 34 encontra-se uma imagem do módulo utilizado

Figura 34 - Módulo GPS utilizado

A bússola digital utilizada tem a referência HMC5883L e baseia-se no princípio de funcionamento de um magnetómetro (Silva, 2013). Este sensor é alimentado a 5V e comunica com o Arduíno através de porta série com *baud rate* de 19200 bits por segundo. Este sensor devolve *strings* em formato *National Marine Electronics Association* (NMEA) (NMEA, 2002), contendo informações relativas à proa, adornamento e caimento do veleiro. Na Figura 35 encontra-se uma fotografia deste sensor.

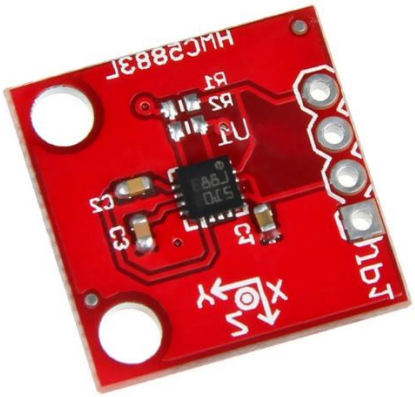


Figura 35 - Bíssula digital utilizada

O anemómetro é da marca *Argent Data Systems* e tem a referência SEN10008. Este sensor é ligado ao Arduíno através da adaptação de uma porta do tipo RJ45 (e.g. CableOrganizer, 2016). A informação relativa à direção do vento é medida através da

variação de tenção num pin analógico do Arduíno. Na Figura 36 encontra-se o esquema elétrico da ligação deste sensor ao *Arduíno*.

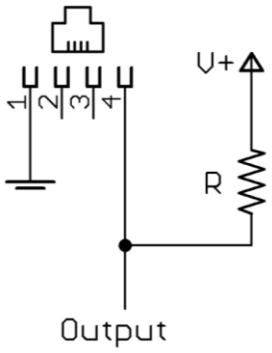


Figura 36 - Esquema elétrico da ligação do sensor, no qual: R = 210 Ohm, V+ = 5V e 0,32V < OUTPUT < 4,78V

O valor de cada tensão lida no pin analógico do Arduíno pode ser associada a uma direção através de uma matriz de correspondência fornecida no *datasheet* deste sensor, de acordo Anexo A. Na Figura 37 encontra-se uma imagem deste sensor

Figura 37 - Sensor de vento utilizado

Também foram colocadas no veleiro um sistema de comunicações, por forma a ser possível a partir de um computador saber quais os valores medidos nos sensores, as ações que o veleiro estava a tomar e até controlar o mesmo através de protocolo definido e explicado em capítulo posterior. Na Figura 38 encontra-se uma figura do dispositivo utilizado.

Figura 38 - Dispositivo de comunicação

Este módulo é da marca *Silicon Lab RF* e tem a referência sv611v2 (NiceRF Wireless Technology Co., 2016). As principais características encontram-se na Tabela 4.

Tabela 4 - Características do módulo de comunicação (NiceRF Wireless Technology Co., 2016)

Distância máxima de transmissão	1400 metros
Frequências de trabalho	433/470/868/915 MHz
Interface	TTL/RS232/RS485
Ganho da antena	-121dBm
Potência de transmissão	100 mW
Baud rate	9600

Embora a distância de transmissão anunciada seja de 1400 metros, verifica-se que a partir dos 500 metros a transmissão de dados é muito afetada.

3.2.3. Atuadores utilizados

O servo utilizado é da marca *Hitec* com a referência HS-765HB (Robotzone, 2016). Este servo tem um binário máximo de 13,2 Kg/cm @ 6,2 V. Contudo, por forma a desgastar menos o mecanismo, optou-se por utilizar a configuração 11,0 Kg/cm @4,8 V. A posição do servo pode variar entre 0º e 180º. A posição em que o servo se encontra é controlada através de uma onda quadrada escrita numa porta digital do Arduíno. Na Figura 39 encontra-se uma imagem do servo utilizado.

Figura 39 - Servo utilizado no controlo do leme

O controlo da posição dos mastros é feito atuando sobre um motor que através de uma caixa de engrenagens move os mastros. Na Figura 40 pode se ver uma fotografia da placa de controlo tal como ela foi montada.

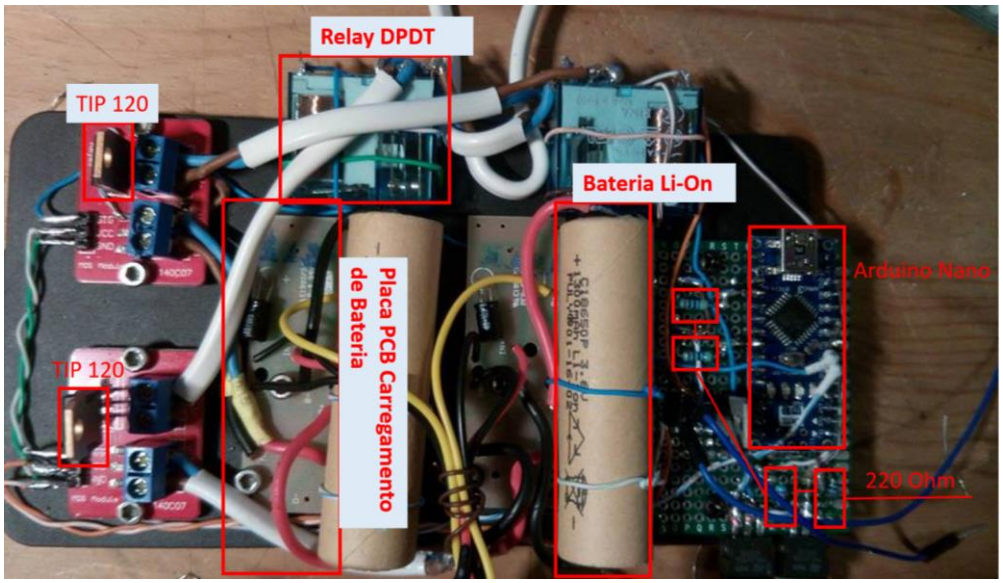


Figura 40 - Placa de Controlo dos Mastros

Esta placa segue o esquema elétrico apresentado na Figura 41.

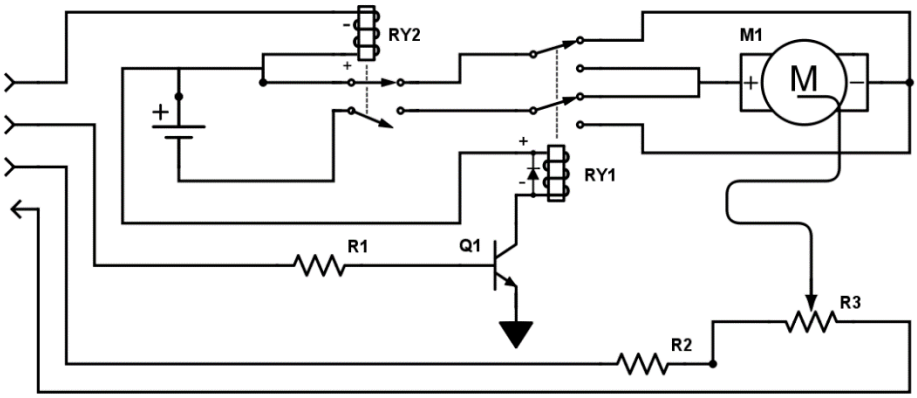


Figura 41 - Esquema elétrico da placa de controlo dos mastros

Esta placa é controlada pelo Arduíno Nano. Este microcontrolador recebe do outro Arduíno a posição em que deve colocar os mastros por porta série. Em seguida, através do feedback dado pelo potenciómetro de posição do mastro, calcula se o ângulo do mastro está inferior, superior, ou dentro da margem pretendida. Se estiver na margem pretendida, não faz nada. Se estiver abaixo ou acima do ângulo pretendido, efetua a correção através do acionamento do motor, através do *transístor* TIP 120, e da inversão ou não do sentido de rotação do mesmo, através do relé RY1 da Figura 41.

Durante a participação na *World Robotic Sail Conference* o conector Universal Serial Bus (USB) do Arduíno Nano partiu, efetuando um curto-circuito que levou a um sobreaquecimento dos transístores TIP120 e consequente falha dos mesmos. Tendo em conta o nível de degradação desta placa, decidiu-se efetuar a troca da mesma pelo controlador *Veyron Driver* com a referência MOT01014. A troca desta placa por este controlador conduziu a uma alteração do código do Arduíno do sistema central de bordo. O novo código encontra-se no Apêndice L – Código ficheiro CPP *Main* após incorporação do constrolador Veyron Driver. Dado o código se encontrar dividido em CPP Files, apenas foi necessário alterar o ficheiro *Main*. Verificou-se que este controlador permite um controlo mais suave da posição dos mastros e uma simplificação do sistema. Na Figura 42 encontra-se uma figura do controlador utilizado.

Figura 42 -Controlador Veyron Driver

3.2.4. Desenvolvimento de um Arduíno *Shield* – Placa PCB

Outra necessidade foi o desenvolvimento de uma placa em PCB por forma a integrar todos os sensores e atuados e fazer a ligação com o microcontrolador. Esta placa garante uma maior fiabilidade do mesmo. Esta placa integra os sinais oriundos dos seguintes sistemas:

- Anemómetro;
- Servo do Leme;
- Comunicações serial a 433Mhz;
- Leitor Cartão SD;
- Serial interface com a placa de controlo dos mastros;
- GPS;
- Bússola digital.

O desenho da placa foi realizada com recurso ao programa *EAGLE* 7.2.0. Este programa é desenho de placas PCB e circuitos eletrónicos. Na Figura 43 encontra-se o desenho da placa utilizada no projeto.

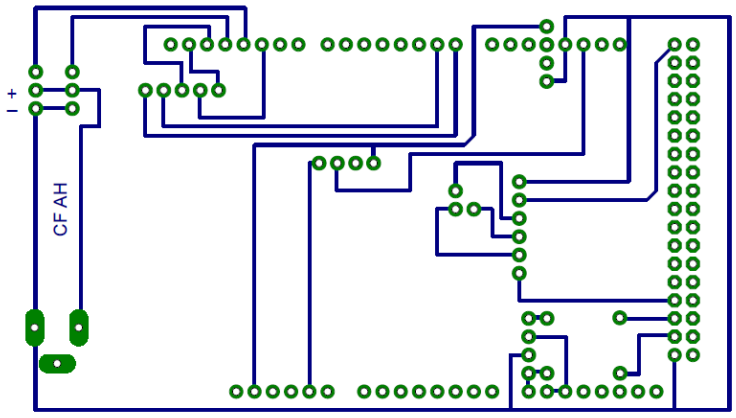


Figura 43 - Placa PCB

Após montagem dos sensores, conetores e outros componentes eletrónicos, o arranjo final da placa encontra-se na Figura 44.

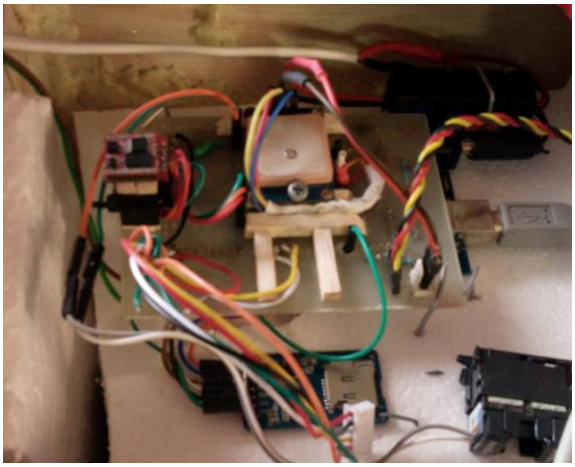


Figura 44 - Placa PCB após montagem dos componentes

3.3. Consumo Energético Estimado

O estudo pormenorizado do consumo energético do veleiro tem de ter em conta inúmeros fatores tais como a variação das necessidades do veleiro de acordo com a missão, carga colocada no servo dos lemes e motores do controlo do mastro, ou até mesmo, a variação de consumo energético do controlador de acordo com o algoritmo implementado. Contudo, verifica-se importante ter uma estimativa do consumo do mesmo, por forma a ser possível efetuar uma estima da autonomia. Nesse sentido, foi construída a Tabela 5, na qual estão expressos os principais sistemas de bordo e a respetiva energia que necessitam. A corrente que cada sistema consome foi retirada do *datasheet* fornecido pelo fabricante do sistema. No caso do Arduíno, a corrente consumida vária de acordo com inúmeros fatores, pelo que se mediu a corrente que o mesmo estava a consumir num instante de funcionamento normal. O anemómetro funciona por variação da resistência aos seus terminais, pelo que a corrente que o mesmo consome varia com o ângulo de vento medido. Para efeitos de cálculo foi utilizado o valor correspondente ao maior consumo energético. Relativamente ao servo de controlo do leme e os motores de controlo da posição do mastro, considerou-se valores medidos, num determinado instante, no qual estes se encontravam em carga. Considerou-se a percentagem em *idle*, o tempo estimado que estes funcionam em operação normal do veleiro.

Tabela 5 - Consumos energéticos

Item	Corrente (mA)	Tensao (V)	Potencia idle (W)	% idle	Corrente (mA)	Tensão (V)	Potencia nominal (W)	Potencia Média(W)
anemometro	21	5	0,105	100	-	-	-	0,105
GPS	35	5	0,175	100	-	-	-	0,175
modulo comunicações	0	0	0	90	25	5	0,125	0,1125
bussula	22	5	0,11	100	-	-	-	0,11
arduino	65	12	0,78	100	-	-	-	0,78
servo leme	8,7	5	0,0435	75	230	5	1,15	0,320125
motores controlo mastros (x2)	0	0	0	90	500x2	12	12	1,2
sistema controlo mastros	50	12	0,6	100	-	-	-	0,6
							TOTAL	**3,40**

O valor estimado de consumo energético médio do veleiro é 3,40W. A bateria utilizada no veleiro foi a bateria WP3-12 12Volt 3Ah. As características desta bateria encontram-se no Anexo B. Um consumo de 3,40W à tensão de 12V equivale a uma corrente de 283mA. De acordo com o fornecedor, o período de tempo entre a bateria totalmente carregada, até esta chegar a 10,5V para este consumo energético é de aproximadamente 10 horas.

4. Programação

Este capítulo apresenta os programas desenvolvidos no âmbito desta dissertação. Os programas podem ser divididos em três conjuntos:

- Código do Sistema Central de Bordo (Arduíno Mega 2560)
- Código do Controlador dos Mastros (Arduíno Nano)
- Código da Estação em Terra (PC a correr MS Windows)

O código do Sistema Central de Bordo foi escrito em C++ com recurso ao Arduíno *Integrated Development Environment* (IDE). Este código contempla a integração dos sensores de vento, GPS, bússola, cartão Secure Digital (SD), servo do leme e comunicação serie com a estação em terra e o Arduíno Nano de controlo dos mastros.

O código do Controlador dos Mastros foi escrito de forma análoga ao do Arduíno Mega 2560. Este código contempla a comunicação serie com o Arduíno Mega 2560, a leitura dos potenciómetros indicadores da posição dos mastros e o controlo dos motores que movimentam os mastros através de sinais enviados à placa de controlo dos mastros.

De forma a facilitar a interpretação do código, este foi escrito por módulos, divididos em diferentes ficheiros. Obrigatoriamente, um dos ficheiros tem de se chamar "main" e tem de conter duas funções:

- void setup(): o código desta função apenas corre uma vez aquando inicialização do Arduino.
- void loop(): quando termina o código da função, o Arduíno reinicia a leitura do mesmo, operando, tal como o nome indica, em *loop* até que o microcontrolador seja desligado

4.1. Sistema Central de Bordo

Todas as interfaces com sensores têm de fazer o parse das mensagens geradas por estes (NMEA para bússola e GPS, sinal analógico para sensor de vento)

4.1.1. Código da Bússola

No Apêndice A encontra-se o código utilizado neste sensor. Este fragmento de código tem a função de atualizar as seguintes variáveis inteiras de acordo com a informação lida no sensor:

```
int head = 0;
int roll = 0;
int pitch = 0;
```

Na inicialização estas variáveis são iguais a 0, e, caso haja algum problema com o sensor, a proa, o adornamento e o caimento serão forçados a zero. Estas variáveis foram definidas como “public”, ou seja, são visíveis em todo o programa.

Este ficheiro define uma classe, chamada “Compass” e uma função dentro dessa classe, chamada “Update()”. A função Update não tem variáveis de entrada nem de saída (é void), mas actualiza as variáveis de sistema “roll”, “pitch”, “head”. O sensor da bússola está ligado à “Serial Port 2” do Arduíno, definida para funcionar a 19200 bits por segundo de acordo com o protocolo de comunicação deste sensor.

Quando a função update e chamada, ela tenta ler a informação do sensor chamando “**Serial2**.available()”. Se esta função retornar “1”, inicia a leitura do sensor através da função “**Serial2**.read()” até esta devolver o char “'\n'”, significando fim de leitura. A informação é guardada na variável “proa_string”. Quando a leitura está completa, esta variável é enviada para os métodos: “update_proa”, “update_roll” e “update_pitch”, que através da função “substring” extrai as variáveis head, roll e pitch.

4.1.2. Código GPS

No Apêndice B encontra-se o código utilizado neste sensor. Por forma a facilitar a aquisição das coordenadas GPS, foi utilizada uma biblioteca Arduíno, “TinyGPS” (Arduiniana, 2016). O código do GPS é constituído por uma classe, chamada GPS. Dentro desta classe existem três funções: “Update”, “CalcDist”, “CalcAzi”.

4.1.2.1. Função *“Update”*

Esta função tem como objetivo a atualização das seguintes variáveis:

- “Date”: guarda o valor relativo à data.
- “time”: guarda o valor relativo às horas (em segundos desde 1JAN1970)
- “lat”: latitude da posição (long int, em graus, com parte decimal, multiplicado por 100000)
- “lon”: longitude da posição.
- “Speed_kts”: velocidade em Nós.
- “GPS_head”: rumo do veleiro.

A libraria “TinyGPS” faz a atualização automática dos valores, bastando indicar a variável onde deve de ser guardado a informação. A parte do código responsável pela atualização é a seguinte:

```
if(gps_nr1.encode(Serial1.read())){
    gps_nr1.get_datetime(&date, &time, &fix_age);
    gps_nr1.get_position(&lat,&lon,&fix_age);
    Speed_kts = gps_nr1.f_speed_knots();
```

```
GPS_head = gps_nr1.f_course();
}
```

4.1.2.2. Função *"CalcDist"*

Esta função tem por objetivo a determinação da distância a que o veleiro se encontra do *waypoint* para o qual se está a deslocar. Esta função é chamada no "voidloop()" através do seguinte código:

- gps.CalcDist(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));

Na qual:

- "gps" representa a classe principal.
- "CalcDist" a função de calculo da distância.
- "gps.lat" a latitude atual do veleiro.
- "gps.lon" a longitude atual do veleiro.
- "wp_number" o número do waypoint para o qual o veleiro se está a deslocar.
- "wp.wplat[wp_number]" a latitude do waypoint para o qual o veleiro se está a deslocar.
- "wp.wplon[wp_number]" a longitude do waypoint para o qual o veleiro se está a deslocar.

O valor da distância, em metros, é guardado na variável "DistCalc".

4.1.2.3. Função *"AziCalc"*

Esta função tem por objetivo a determinação do azimute entre o veleiro e o waypoint para o qual está a navegar. Esta função é chamada no "voidloop()" através do seguinte código:

- gps.CalcAzi(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));

Na qual:

- "gps" representa a classe principal.
- "CalcDist" a função de calculo da distância.
- "gps.lat" a latitude atual do veleiro.
- "gps.lon" a longitude atual do veleiro.
- "wp_number" o número do waypoint para o qual o veleiro se está a deslocar.
- "wp.wplat[wp_number]" a latitude do waypoint para o qual o veleiro se está a deslocar.
- "wp.wplon[wp_number]" a longitude do waypoint para o qual o veleiro se está a deslocar.

O valor do azimute é guardado na variável "AziCalc".

4.1.3. Código Sensor do Vento

Este código tem por função a atualização da direção do vento. A medição da direção do vento é feita através da variação da tensão no sensor do vento. A medição é feita através do pin analógico A8 do Arduíno.

Este ficheiro de código é constituído por uma classe, ”Wind”, e uma função *void* “Update”. A função verifica qual o valor de tensão no pin A8 através do seguinte código “analogRead(A8)”. Depois compara essa tensão com as tensões definidas pelo fabricante para cada ângulo. Este ângulo é guardado na variável “wind.Direction”

Para chamar esta função no voidloop do programa, utiliza-se o seguinte código:

- wind.Update(sail.angle);

No qual:

- “wind” representa a classe;
- “Update” representa a função que se quer chamar da class “wind”;
- “sail.angle” é o ângulo no qual o mastro se encontra[7].

4.1.4. Ficheiro de código “main”

Este é o ficheiro de código principal, a partir do qual são chamadas todos os outros ficheiros nomeadamente:

- #include "Main_GPS.cpp"
- #include "compass.cpp"
- #include "WayPoints.cpp"
- #include "Wind.cpp"
- #include "Rudder.cpp"
- #include "Sail.cpp"

Este ficheiro obrigatoriamente tem de conter as funções void setup() e void loop(). Na função void setup() são inicializados os seguintes sistemas:

- Porta Serie 0: Sistema de comunicações, com baud rate de 9600.
- Porta Serie 1: Modulo GPS, com baud rate de 9600;
- Porta Serie 2: Bíssula digital, com baud rate de 19200;
- Porta Serie 3: Sistema de controlo dos mastros, com baud rate de 4800;
- Servo do Leme: no pin digital 12 do Arduíno. É escrito o valor 98 neste pin, correspondendo à posição central do leme.
- Cartão SD: Cria o filheiro “LOG.txt”, senão escreve (“initialization of SD Card FAILED!”) na porta serie 0.

[7] Devido ao sensor estar colocado no topo do mastro e este rodar, é necessário efetuar a subtração do valor do ângulo para se obter a direção verdadeira do vento.

Em seguida, o Arduíno corre interruptamente a função "void loop()", correndo as seguintes funções por ordem, repetindo quando chega ao fim:

- compass.Update();
- gps.Update();
- gps.CalcDist();
- gps.CalcAzi();
- wp_test();
- head_to_go();
- rudder.Update();
- sail.Update();
- UpdateBS();

As funções sail.Update() e rudder.Update() estão associadas à variável boliana "Autonomous". Se esta variável for igual a "false", os parâmetros de entrada na função "rudder.update()" são as variáveis compass.head e "azi_manual[8]". Na função sail.Update() o parâmetro de entrada é a variável "sail_manual[9]".

4.2. Código do Controlador dos Mastros

Os mastros são controlados individualmente, embora o ângulo dos mastros tenha de ser igual para ambos. O ângulo de vela é definido pelo ângulo da retranca e a proa do veleiro. Os valores do ângulo para as velas encontram-se limitados de 90º a 270º

O mastro de vante é controlado pelos pins:

- Pin digital 3: para rodar o mastro valor 1, definido a 0 por default;
- Pin digital 6: define o sentido de rotação do mastro, valor 1 o mastro aumenta o ângulo , valor 0 o mastro reduz o ângulo.

O mastro de Ré é controlado pelos pins:

- Pin digital 4: para rodar o mastro valor 1, definido a 0 por default;
- Pin digital 5: define o sentido de rotação do mastro, valor 1 o mastro aumenta o ângulo , valor 0 o mastro reduz o ângulo.

Na função void setup() são chamadas as seguintes funções

- Serial.begin(4800): inicia comunicações seria com baud rate de 4800;
- pinMode(3, OUTPUT): inicia o pin 3 como output;
- pinMode(4, OUTPUT); inicia o pin 4 como output;
- pinMode(5, OUTPUT); inicia o pin 5 como output;
- pinMode(6, OUTPUT); inicia o pin 6 como output;

[8] Esta variável inteira é definida pelo utilizador, e será a proa que o veleiro ira seguir.
[9] Esta variável inteira é definida pelo utilizador, e será o ângulo a que o veleiro colocará as velas.

4.2.1. Função void safe();

Esta função tem como objetivo garantir que em caso de avaria os mastros são desligados. Se for detetado através dos potenciómetros de posição do mastro que pelo menos uma das velas se encontra fora dos valores aceitáveis para a posição das velas, estas são desligadas, sendo enviada a seguinte mensagem por porta serie: "SistemFail - Forced Shutdown", complementada pela leitura do ângulo das velas.

4.2.2. Função void loop();

Esta função é responsável por atuar diretamente nos mastros, corrigindo a posição dos mesmos. A sequência de passos que a função efetua é a seguinte:

I. Serial.parseInt(): Verifica se foi definido pelo Sistema Central de Bordo um novo ângulo para as velas.
II. sail_read1 = analogRead(A4): Verifica o ângulo a que se encontra o mastro de vante.
III. diference1 = sail_read1 – angle: verifica a diferença entre a instrução do Sistema Central de Bordo e o ângulo atual da vela de vante.
IV. if (diference1 > 5): testa se a diferença entre o ângulo da vela de vante é superior a 5º, se for escreve no pin 3 o valor 1 e no pin 6 o valor 1, nesta configuração a rotação aumenta o ângulo da vela.
V. if (diference1 < -5): testa se a diferença entre o ângulo da vela de vante é inferior a -5º, se for escreve no pin 3 o valor 1 e no pin 6 o valor 0, nesta configuração a rotação diminui o ângulo da vela.
VI. Efetua os passos IV, V e VI para a vela de Ré;
VII. Delay(75): espera 75 milissegundos por forma a garantir que os mastro dão um pequeno passo na direção pretendida.
VIII. Escreve os valores 0 nos pins 3 e 4 por forma a parar de rodar os mastros.

4.3. Código da Estação em Terra

Este programa foi desenvolvido em C# com recurso ao *VisualStudio2015*. O objetivo deste programa é funcionar em concordância com as funções ReadComand() e UpdateBS() do Sistema Central de Bordo. A Estação em Terra envia instruções para o Sistema Central de Bordo através da função ReadComand() e recebe as informações relativas a sensores e atuadores através da função UpdateBS().

Este programa tem o *design* gráfico representado na Figura 45

Figura 45 - Design Gráfico da Estação em Terra

Para utilizar o programa basta:

- Instalar este em qualquer computador com Windows, versão igual ou superior à versão XP.
- Conectar o módulo de comunicações numa entrada USB
- Selecionar o baud rate para 9600.
- Carregar no botão ligar. Se a ligação falhar, irá receber a mensagem “Algo correu mal, não foi possível ligar ao Arduino! :/”

O código deste programa funciona da seguinte forma:

- Liga à porta Serie e com o Baud rate indicados pelo utilizador;
- Tenta receber Strings pela porta Serie;
- Todas as strings recebidas são mostradas na “Listbox” geral, campo mais à direita da Figura 45.
- Cada String corresponde a uma variável distinta enviada pela função UpdateBS(), sendo mostrada no respetivo campo da Estação em Terra. Estas variáveis podem ser guardadas através da Estação em Terra em ficheiro “.txt” na pasta “C:\LCA Logs”. Se esta pasta não existir, o programa cria a pasta. Para selecionar quais variáveis guardar, deve se selecionar as checkbox correspondente de acordo com Figura 46. Esta form, “Criar log”, aparece se o utilizador carregar no botão “Criar log”.
- É possível enviar comandos para o veleiro, escrevendo os mesmos no campo “Enviar Comando”, carregando de seguida no “Enter” do teclado.

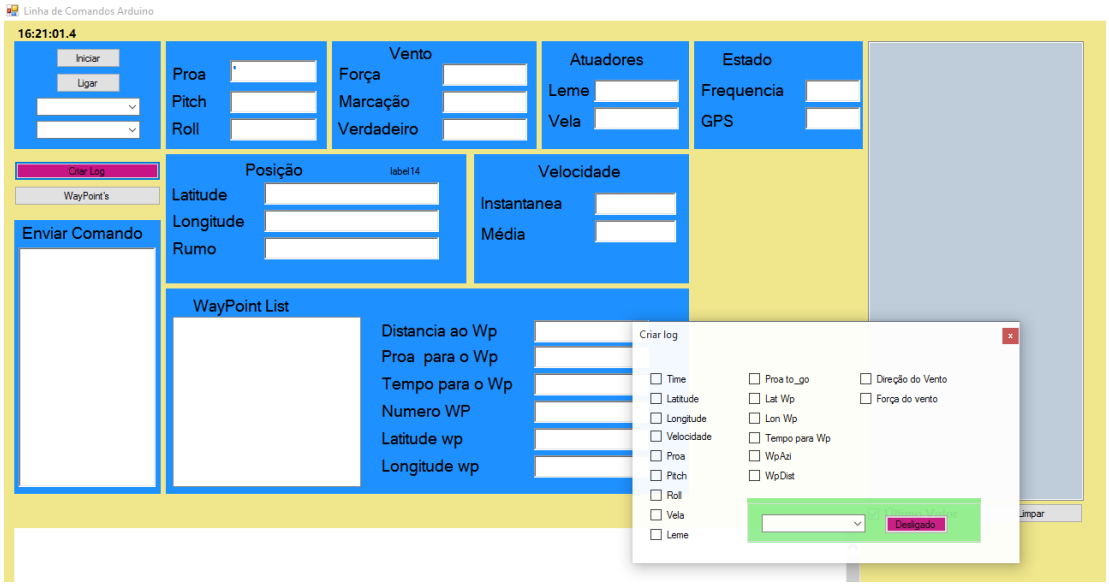


Figura 46 - Form Criar log

5. Testes Efetuados

5.1. Primeira Fase de Testes

O objetivo da primeira fase de testes é a verificação da estabilidade, estanquidade do casco e operacionalidade e fiabilidade de todos os sistemas.

O primeiro teste efetuado à plataforma tinha como objetivo a verificação das condições de estanqueidade e estabilidade do veleiro. A bordo do NRP D. Carlos I, o veleiro foi colocado na água através da grua *Palfinger*. O veleiro comportou-se de acordo com o que foi projetado. O calado aproximado, não contando com o patilhão e bolbo, foi de aproximadamente 18cm. O Caimento ficou a ré, não tendo sido possível a determinação do valor exato, pois, o veleiro ainda não continha a eletrónica a bordo, de modo a garantir a salva guarda da mesma, caso se verifica-se a inconformidade da estanqueidade e consequente entrada de água para bordo. Na Figura 47 encontra-se uma fotografia tirada a bordo do NRP D. Carlos I a quando do início do teste.

Figura 47 - Primeiro teste efetuado ao veleiro

Foi detetada a entrada de uma pequena quantidade de água para o bolbo do veleiro. O bolbo encontra-se fisicamente isolado do restante veleiro, desse modo, a estanqueidade não foi comprometida, porém, de modo a confirmar o ponto de entrada de água e proceder à sua reparação, o veleiro foi removido da água aproximadamente 5 minutos após a sua colocação.

Verificou-se que a entrada de água estava a dever-se a uma fissura na peça impressa a 3D que constitui a parte central do bolbo. A reparação foi efetuada com recurso a silicone cola. Após a cura deste produto, foi aplicada uma camada de tinta por forma a garantir que não se voltaria a verificar este problema.

O segundo teste teve como objetivo verificar se o problema de estanqueidade no bombo estava resolvido, e testar os sistemas de bordo, nomeadamente o sistema de comunicações, Controlo dos Mastros e o leme. Outro objetivo era verificar o tempo que as baterias aguentavam o veleiro em funcionamento. Na Figura 48 encontra-se uma fotografia da colocação do veleiro na água para início do segundo teste.

Figura 48 - Segundo teste em água efetuado

Verificou-se que o problema de estanqueidade estava resolvido. Todos os sistemas funcionaram como previsto. A parte mecânica do sistema de controlo do mastro de ré ganhou uma pequena folga, e o potenciómetro de controlo da posição moveu-se da posição inicialmente calibrada. A parte mecânica foi revista, tendo-se procedido a reaperto dos parafusos de ligação da roda dentada grande ao mastro. Voltou a calibrar-se o mastro, deixando para mais tarde a resolução definitiva do problema, pois esta envolver a desmontagem de todo o sistema mecânico devido ao difícil acesso do potenciómetro.

O veleiro ficou na água aproximadamente 3h20m. A tensão das baterias antes do teste era 13,2V e após teste 12,7V. O fabricante indica que a bateria totalmente carregada tem uma tensão de 12V, no entanto, foram medidos valores superiores. Verifica-se necessária a realização de testes complementares para determinar com maior exatidão a autonomia

do veleiro, embora, estes valores sejam de todo um fator positivo pois indicam que o veleiro tem uma larga autonomia, mesmo sem ter instalado painéis solares.

O terceiro teste tinha como objetivo a verificação da resolução do problema da folga na vela de ré e testar o envio de comandos para o veleiro a partir da estação em terra.

Verificou-se que o mastro de ré ainda continha um erro na posição devido ao potenciómetro de controlo da posição não estar devidamente fixo, como é visível na Figura 49. O envio de comandos para o veleiro requeria em média três tentativas até ser recebido sem erros no veleiro.

Figura 49 - Terceiro teste em água

O mastro de ré foi de novo calibrado, mas ficou decidido que se iria desmontar a parte mecânica na totalidade por forma a aceder ao potenciómetro de controlo da posição e fixa-lo corretamente. A dificuldade no envio de comandos é inerente à baixa potência do sistema de comunicações, potência de envio 100mW. O problema pode ser minimizado através do envio automático e repetido de qualquer comando por parte da estação em terra. Alternativamente, pode ser colocado no veleiro um sistema de 500mW ou até mais, que irá garantir uma distância eficaz maior.

5.2. Provas de Mar

O objetivo desta fase de testes era verificar o comportamento do veleiro em navegação autónoma e a implementação de melhorias no algoritmo de controlo do sistema central de bordo. Verificou-se que em condições de pouco vento, inferior a 5kts, o veleiro ganhava seguimento aproximado de 2kts, o que demostra a consistência das simulações hidrodinâmicas efetuadas, uma vez que, a resistência ao avanço até essa velocidade é muito reduzida, aumentando significativamente a partir dessa velocidade. Na Figura 50 encontra-se o *tracking* do veleiro na primeira navegação autónoma realizada. O veleiro largou do NRP D. Carlos I, atracado no cais 4 pos.1S da BNL, navegou até ao primeiro *waypoint*, uma boia do canal do Alfeite, tendo em seguida alterado rumo para o segundo waypoints, a "rampa do CNOCA".

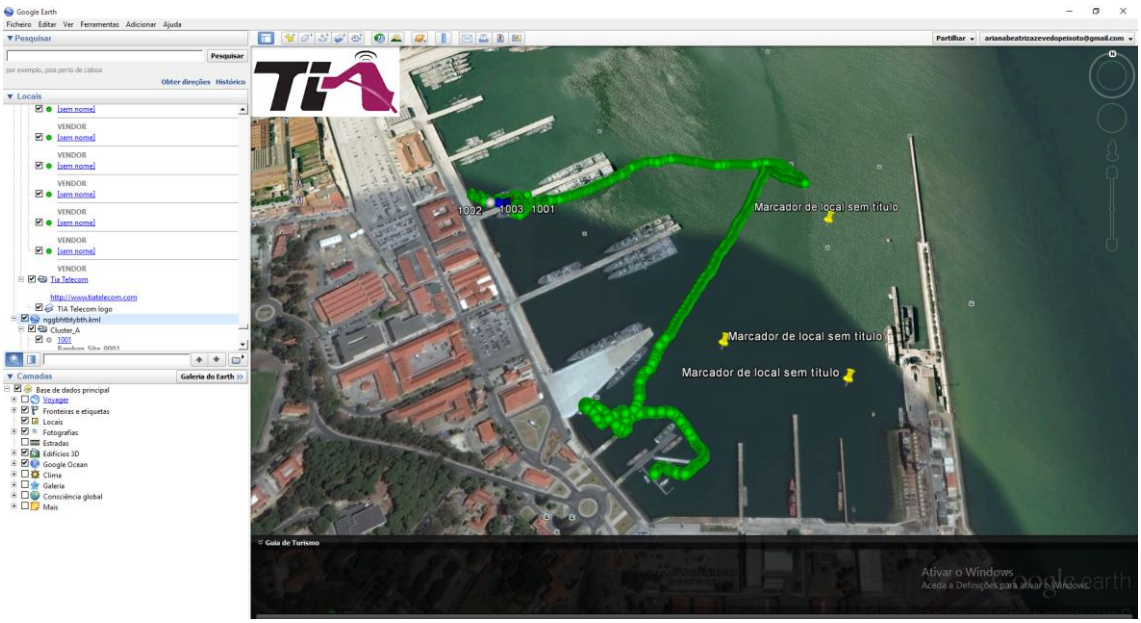


Figura 50 - Primeiro teste de navegação corrida

Foram definidos dois waypoints diferentes para a segunda navegação, marcados pelos alfinetes amarelos na Figura 51. O veleiro foi rebocado até posição mais a norte do *tracking* a verde, onde iniciou a sua navegação em modo autónomo.

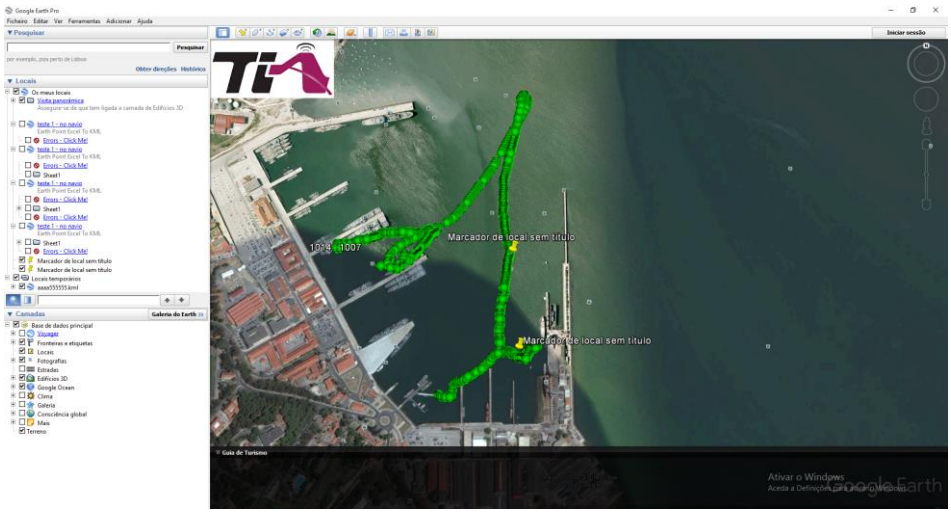


Figura 51 - Segunda navegação do veleiro

Verificou-se que o algoritmo de escolha da proa para navegação era adequado, contudo, seria pertinente a inclusão no mesmo do abatimento do veleiro, para que a aproximação aos waypoints fosse uma reta, e não uma curva como é visível na Figura 52.

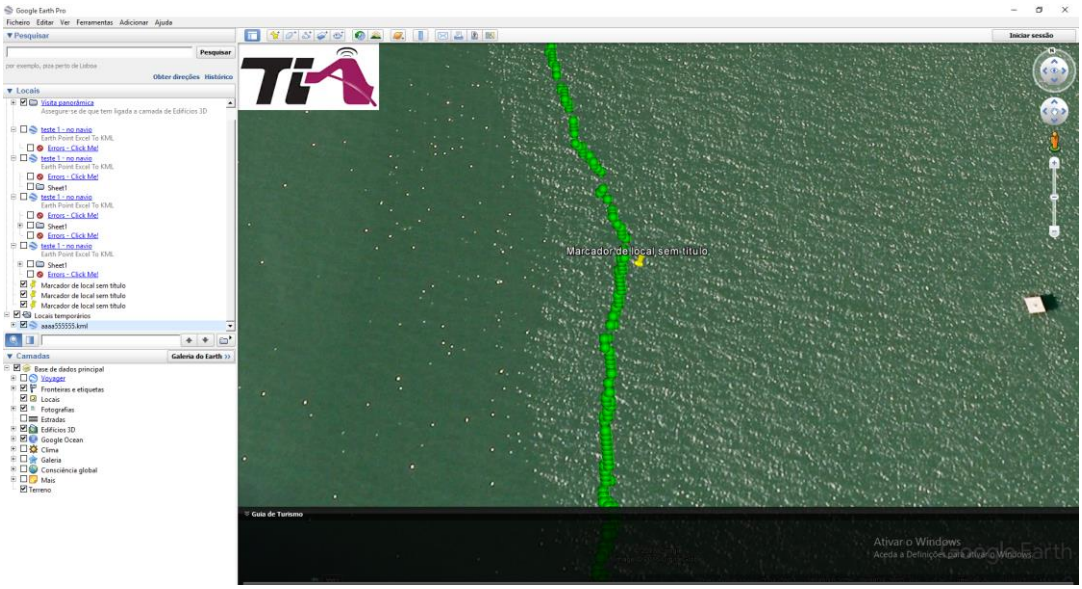


Figura 52 – Passagem no primeiro waypoint da navegação

O veleiro passou a 3 metros do waypoints definido. Na Figura 53 encontra-se a passagem do veleiro pelo segundo waypoints. Verificou-se dificuldade em o veleiro rodar devido às velas não auxiliarem o leme. É necessário a inclusão de uma condição no código do veleiro que detete a necessidade de o veleiro guinar, e que ajuste as velas de maneira a facilitar a manobra.

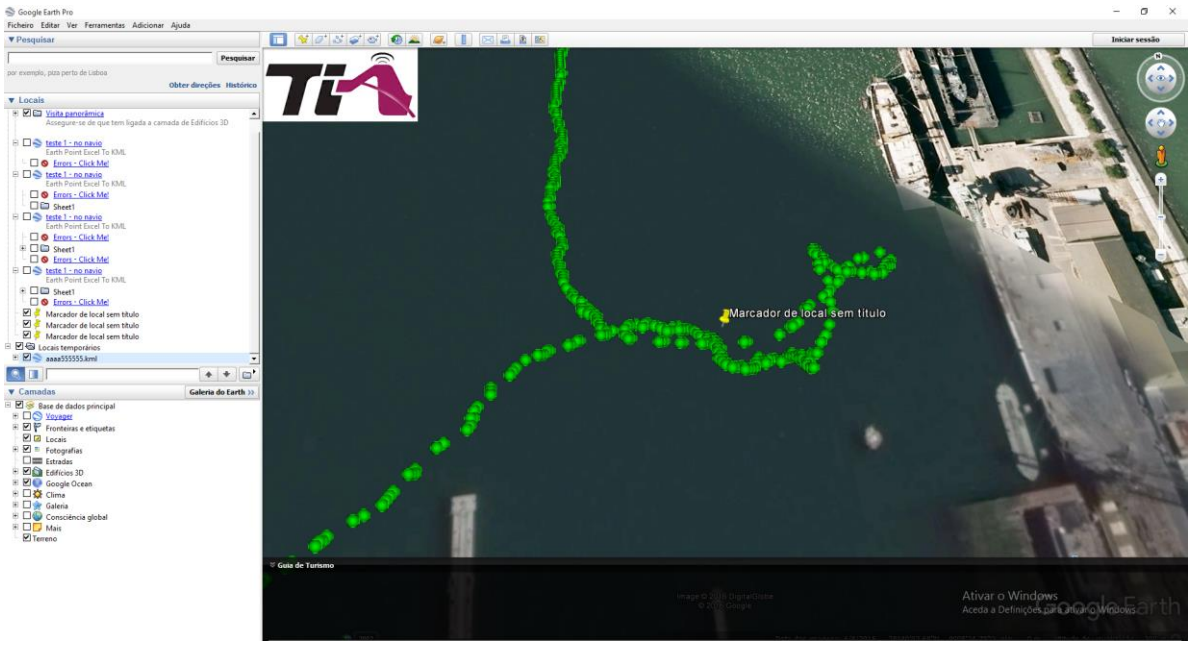


Figura 53 - Passagem pelo segundo waypoint da navegação

5.3. Participação na regata WRSC

Como referido anteriormente, o veleiro sofreu uma avaria que o impossibilitou de participar nos dois primeiros dias de prova. Com esforço da equipa da Escola Naval, foi possível restituir a operacionalidade do veleiro, tendo este participado nas provas *collision avoidance* e *area scanning*, tendo falhado na participação das provas *Station keeping* e *fleet race*. O veleiro não estava preparado para participar na prova de *collision avoidance*, devido a não ter sistema de deteção de obstáculos. Contudo, no dia da prova foi feito um sistema rudimentar com um detetor de obstáculos ultrassónico por forma a o veleiro participar na prova. O *tracking* está visível na Figura 54, estando representado pela linha verde mais a oeste. O veleiro abateu para sul, não tendo pontuado na prova. A linha verde mais a este é o percurso da *fleet race*. A prova não foi válida porque não foi feita no dia da mesma, contudo a fim de testes decidiu-se colocar o veleiro a realizar a mesma. Verificou-se que o mesmo se aproximou dos waypoints pretendidos, e que se tivesse realizado a prova no dia correto, teria tido uma boa prestação.

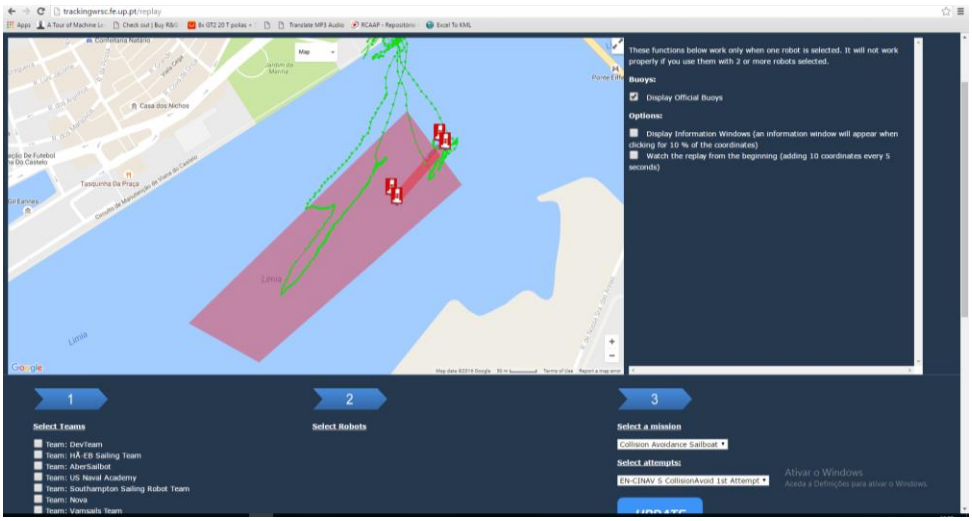


Figura 54 - Tracking da prova colision avoidance

O veleiro participou ainda na prova *area scanning*. O veleiro pontuou na prova, contudo verificou-se que deve ser desenvolvidos algoritmos específicos para cada prova. O tracking desta prova encontra-se na Figura 55, onde é possível ver que o mesmo abateu para sul, tendo por esse motivo terminado a sua prova.

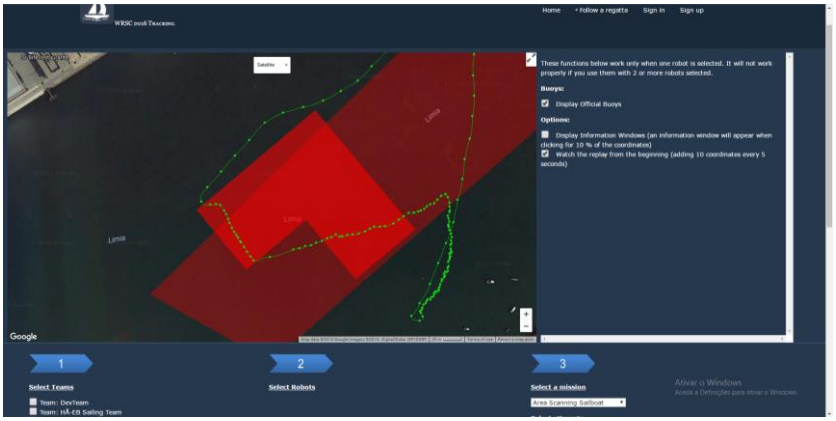


Figura 55 - Tracking da prova Area Scanning

6. Conclusão e trabalho futuro

Este trabalho inicia-se com um estudo dos diferentes tipos de veículos autónomos existentes, dando maior enfase aos veleiros autónomos. Em seguida, inicia-se o projeto do casto, efetuando para isso estudos hidrostáticos e hidrodinâmicos, e desenhando o mesmo na aplicação *Delft Ship Free*. Em seguida, iniciou-se a construção do casco com recurso a uma estrutura metálica em aço laminado e materiais compósitos. Sistemas periféricos tais como portas de visita e sistema de controlo da posição do mastro foram desenvolvidos em *Solidworks Student Version* e impressos utilizando uma impressora 3D. Em seguida é explicado os sistemas eletrónicos do veleiro e é feito um estudo sumário do consumo energético do veleiro. Depois, é estudado o código CPP e C# desenvolvido para o projeto. Por fim, são mostrados os resultados obtidos nas provas de mar e tiradas conclusões acerca das mesmas.

Deste projeto resultou uma plataforma autónoma robusta e testada, que pode servir de base para futuros trabalhos de vela robótica, tendo como objetivo últimos os objetivo traçados para o projeto (Lobo, 2014). Resultou também numa experiencia prática de utilização de impressão 3D para prototipagem de mecanismos mecânicos complexos. Para além desta tese, foram produzidos quatros artigos científicos e a EN foi representada em inúmeros eventos, sendo o principal o WRSC. Os artigos encontram-se nos Apêndices M, N e O. Este evento, bem como o seminário *The Armed Forces Communications and Electronics Association* (AFCEA) Portugal, teve uma cobertura mediática significativa, da qual resultaram inúmeras notícias em jornais e televisão, sendo que uma das notícias se encontra no Anexo C – Noticia sobre o veleiro no jornal economia do mar.

Em qualquer trabalho de engenharia é fundamental ter em conta os custos humanos e materiais para atingir um dado fim, assim sendo, neste projeto, à que distinguir o custo de desenvolvimento deste projeto, incluindo desenvolvimentos experimentais que não surgiram efeito, de construir um sistema igual ao protótipo, a partir das lições aprendidas. Os custos em recursos humanos foram elevados mas constituíram na minha dedicação durante o decorrer do projeto, e vou-me abster de os quantificar. Os custos materiais são mais objetivo, e podem reduzir-se ao expresso no Apêndice K – Relação dos Custos. O custo deste projeto teve o valor aproximado de 1769€, sendo que a

construção de um novo veleiro segundo o mesmo processo de construção, teria apenas o custo de 655€.

Pessoalmente, foi uma oportunidade de crescimento pessoal multidisciplinar. O autor teve oportunidade de estudar e melhorar as suas competências em inúmeras áreas, tais como: Soldadura, compósitos, programação, arquitetura naval, vela, circuitos elétricos, entre outros. Foi realizado um estudo sobre a implementação de um algoritmo de *Machine Learning*, nomeadamente Agente de Procura, o qual se encontra no Apêndice P.

Considero que este projeto deve ser continuado, nomeadamente com a realização dos seguintes trabalhos, os quais podem ser enquadrados em diversas disciplinas.

- Algoritmo que otimiza *area scanning* de acordo com as condições ambientais;
- Algoritmo para otimizar a proa de acordo com o rumo que o veleiro está a fazer;
- Construção de velas novas;
- Corte do patilhão por forma a facilitar o transporte;
- Construção de uma caixa de transporte para o veleiro;
- Desenvolvimento de um sistema de carregamento das baterias a partir de painéis solares instalados a bordo;
- Estudo energético do veleiro e otimização do consumo;
- Desenvolvimento de um novo *shield* para o Arduíno central de bordo;
- Substituição das portas de visita feitas em ABS;
- Melhoramento da estação em terra, com a introdução de mais comandos para controlo do veleiro;
- Substituição do módulo de comunicações por um com mais potência;
- Otimização do algoritmo de controlo dos mastros de forma ao veleiro tirar maior partido do vento, quer para obter maior velocidade, quer para auxiliar em guinadas e proteger em situação de mau tempo.

7. Referencias Bibliográficas

Alt, C., & Wittinghofer, N. (2011). *Autonomous Sailing Boats.* Salzburg: Institut of Computer Science Salzburg.

Alves, J. C., & Cruz, N. A. (2008). FASt - An autonomous sailing platform for oceanographic missions. Quebec : IEEE.

Arduiniana. (2016). *TinyGPS*. Obtido de Arduiniana: http://arduiniana.org/libraries/tinygps/

ARDUINO. (2016). *arduino*. Obtido em 21 de 03 de 2016, de Arduino MEGA 2560 & Genuino MEGA 2560: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Barros, P. L. (2011). *Montagem por Interferência de Rodas Dentadas nos Veios de Redutores Industriais.* Lisboa: Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa.

Blum, A. (1998). *On-Line Algorithms - The State of the Art.* Pittsburgh: Fiat and Woeginger eds.

CableOrganizer. (2016). *Network Wiring Instructions for RJ11 and RJ45*. Obtido de cableorganizer: http://www.cableorganizer.com/telecom-datacom/network-instructions.htm

Cavaco, G. (2011). *Projecto eVEntos - Sensores.* Almada: Escola Naval.

cnccookbook. (2016). *CNCCookbook's G-Code Tutorial and Course*. Obtido de cnccookbook: http://www.cnccookbook.com/CCCNCGCodeCourse.htm

Consortium, P. (2016). *What is PC/104*. Obtido de http://pc104.org/

Costa, E., & Simões, A. (2008). *Inteligência Artificia - Fundamentos e Aplicações.* FCA.

Crunchbase. (2016). *Liquid Robotics*. Obtido de https://www.crunchbase.com/organization/liquid-robotics#/entity

Dias, P. S., Gomes, R. M., & Pinto, J. (2008). *Neptus – A Framework to Support Multiple Vehicle Operation.* Porto: Faculdade de Engenharia da Universidade do Porto .

DreeseCODE Software, L. (2016). *User-Friendly DesignFOIL Features*. Obtido de dreesecode: http://www.dreesecode.com/

Foundation, R. P. (2016). *RASPBERRY PI HARDWARE GUIDE*. Obtido de raspberrypi: https://www.raspberrypi.org/learning/hardware-guide/

Fuchang Gao, L. H. (January 2012). Implementing the Nelder-Mead simplex algorithm. *Volume 51*(Issue 1, pp 259-277).

Grollman, D. H., Dept. of Comput. Sci., B. U., & Jenkins, O. C. (2007). Dogged Learning for Robots. Roma: IEEE.

Joseph Curcio, J. L. (2005). *SCOUT — A Low Cost Autonomous Surface Platform for Research in Cooperative Autonomy.* Cambridge: Department of Mechanical Engineering, Massachusetts Institute of Technology.

Journée, J. e. (2002). INTRODUCTION IN SHIP HYDROMECHANICS. Delft: Delft University of Technology.

Koshel, R. J. (2002). Enhancement of the downhill simplex method of optimization. Tucson: Proceedings of SPIE - The International Society for Optical Engineering.

Lewis, E. V. (1988). *Principles of Naval Architecture - Volume I. Stability and Strength.* Jersey: The Society of Naval Architects and Marine Engineers.

Liquid-robotics. (2016). *Liquid robotics*. Obtido de About us: http://www.liquid-robotics.com/company/about-us/

Littwin, M. (2016). *Repetier*. Obtido de Repetier: https://www.repetier.com/about-us/

Lobo, V. (2014). *http://ptdocz.com/doc/177168/6_comunica%C3%A7%C3%A3o_prof-sousa-lobo---cinav.* Obtido de ptdocz: http://ptdocz.com/doc/177168/6_comunica%C3%A7%C3%A3o_prof-sousa-lobo---cinav

Madhulatha, T. S. (2012). AN OVERVIEW ON CLUSTERING METHODS . *Vol. 2(4)*(pp: 719-725).

Misra, S. C. (2015). *Design Principles of Ships and Marine Structures.* CRC Press.

Mizine, I., Karafiath, G., Queutey, P., & Visonneau, M. (2009). INTERFERENCE PHENOMENON IN DESIGN OF TRIMARAN SHIP . *10th International Conference on Fast Sea Transportation.* Greece .

Mone, G. (2007). *MIT technology review*. Obtido de Autonomous Kayaks: https://www.technologyreview.com/s/407126/autonomous-kayaks/

Mundo Reader, S. (2016). *BQ Prusa i3 Hephestos*. Obtido de bq: https://www.bq.com/pt/prusa

NiceRF Wireless Technology Co., L. (2016). *100mW Anti-interference wireless transceiver module SV611*. Obtido de http://www.nicerf.com/product_view.aspx?id=5

NMEA. (2002). *NMEA 0183 Standard For Interfacing Marine Electronic Devices* . National Marine Electronics Association .

Ray, C., Mondada, F., & Siegwart, R. (2008). What do people expect from robots? Nice: IEEE.

Robots, S. (2016). *Applied Physics – Gears Math.* Obtido de Storming Robots: http://www.stormingrobots.com/prod/tutorial/gearsWksheets%20p1-13%20packet.pdf

Robotzone. (2016). *ServoCity*. Obtido de https://www.servocity.com/hs-765hb-servo

Rocha, C. d. (2011). *Projecto eVentos - Sistema de Controlo.* Alfeite: Escola Naval.

Silva, S. C. (2013). *Desenvolvimento de uma metodologia para realização de levantamentos magnéticos marinhos para deteção de objetos.* Lisboa: Universidade de Lisboa.

slic3r. (2016). *slic3r*. Obtido de slic3r: http://slic3r.org/

Smith, D. (2001). Understanding Gear Ratios. *Racing Lines*.

Tachibana, K., & Fukazawa, R. (2016). Effect of an Ensemble Algorithm in Reinforcement Learning for Garbage-Collection Sailing. *WRSC.* Tokyo.

Vacanti, D. (2005). Keel and Rudder Design. Professional BoatBuilder.

Willcox, J. M. (2010). The Wave Glider: A Persistent Platform for Ocean Science. *IEEE OCEANS.* Sydney.

Wirz, J., Tranzatto, M., Liniger, A., Colombino, M., Hesse, H., & Grammatico, S. (2015). AEOLUS, the ETH Autonomous Model Sailboat. *8th International Robotic Sailing Conference.* Springer International Publishing.

Apêndice A – Código C++ da bússola

```
#include "Arduino.h"


class Compass{
  String proa_string="";
  public:
  //****************************************
  int head = 0;
  int roll = 0;
  int pitch = 0;
  //****************************************

void Update(){
    while (Serial2.available()){
    char c = Serial2.read();
    if (c== '\n'){
      update_proa(proa_string);
      update_roll(proa_string);
      update_pitch(proa_string);
      proa_string ="";
      }
    else{
      proa_string = String(proa_string + c);
      }
    }
}
//_____________________________________________
//_____________________________________________
void update_proa (String stringGIRO){
for (int i = 0 ; i<7;i++){
  if(stringGIRO.substring(i,i+1)=="C"){
     for(int j = 0; j<8;j++){
          if(stringGIRO.substring(j,j+1)=="P"){
              head=string_to_float(stringGIRO,i+1,ji);
             }
      }
  }
}
}
void update_roll (String stringGIRO) {
for (int i = 4 ; i<23;i++){
  if(stringGIRO.substring(i,i+1)=="R"){
      for(int j = 0; j<8;j++){
          if(stringGIRO.substring(j+i,j+i+1)=="T"){
              roll = string_to_float(stringGIRO,i+1,j-i);
             }
      }
  }
}
}
void update_pitch (String stringGIRO){
for (int i = 4 ; i<16;i++){
  if(stringGIRO.substring(i,i+1)=="P"){
      for(int j = 0; j<8;j++){
          if(stringGIRO.substring(j+i,j+i+1)=="R"){
```

```
              pitch = string_to_float(stringGIRO,i+1,j-i-1);
            }
      }
   }
}
}

int string_to_float(String valor_string, int xpos, int lpos){
String a = valor_string.substring(xpos,lpos);
char buf[a.length()];
a.toCharArray(buf,a.length());
int WVal=atof(buf);
return WVal;
}

};
```

Apêndice B – Código C++ do GPS

```
#include "Arduino.h"
#include <TinyGPS.h>

class GPS{
TinyGPS gps_nr1;
public:
  //*********************************************
  long lat,lon;
  int GPS_head;
  float Speed_kts;
  unsigned long fix_age;
  unsigned long time, date;
  bool GPS_LOCK = false;
  //*********************************************
    void Update(){

    while(Serial1.available()){
       if(gps_nr1.encode(Serial1.read())){
          gps_nr1.get_datetime(&date, &time, &fix_age);
          gps_nr1.get_position(&lat,&lon,&fix_age);
          Speed_kts = gps_nr1.f_speed_knots();
          GPS_head = gps_nr1.f_course();
          }

          if (fix_age == TinyGPS::GPS_INVALID_AGE){
            GPS_LOCK = false;
          }
            else if (lat < 1){
                  GPS_LOCK = false;
            }
                  else{
                  GPS_LOCK = true;
                  }
          }
       }

   float CalcDist(float flat1,float flon1,float x2lat,float x2lon){
// flat1 = our current latitude. flat is from the gps data.
// flon1 = our current longitude. flon is from the fps data.

    flat1=flat1/100000;
    flon1=flon1/100000;

    float dist_calc=0;
    float dist_calc2=0;
    float diflat=0;
    float diflon=0;

    diflat=radians(x2lat-flat1);
    flat1=radians(flat1);
    x2lat=radians(x2lat);
    diflon=radians((x2lon)-(flon1));
    dist_calc = (sin(diflat/2.0)*sin(diflat/2.0));
    dist_calc2= cos(flat1);
    dist_calc2*=cos(x2lat);
    dist_calc2*=sin(diflon/2.0);
    dist_calc2*=sin(diflon/2.0);
    dist_calc +=dist_calc2;
```

```
    dist_calc = (2*atan2(sqrt(dist_calc),sqrt(1.0-dist_calc)));
    dist_calc*=6371000.0;
    return(dist_calc);
    }

    float CalcAzi(float flat1,float flon1,float x2lat,float x2lon){

    flat1=flat1/100000;
    flon1=flon1/100000;
    flat1 = radians(flat1);

    flon1 = radians(flon1);
    x2lat = radians(x2lat);
    x2lon = radians(x2lon);

    float heading;
    heading = atan2(sin(x2lon-flon1)*cos(x2lat),cos(flat1)*sin(x2lat)-sin(flat1)*cos(x2lat)*cos(x2lon-
flon1)),2*3.1415926535;
    heading = heading*180/3.1415926535;
    float head = heading;

        if(head<0){
                head+=360;
                }
        if(head>360){
                head-=360;
                }
        return(head);
      }

    void Test(){
    bool a = true;
    String stringOne= "";
    while (a){
    while (Serial1.available()){
    char c = Serial1.read();
    if (c== '\n'){
      Serial.println(stringOne);
      stringOne ="";
    }
    else{
    stringOne =  String(stringOne + c );
      }
     }
    if (Serial.available()) {
    String Comand = Serial.readString();
    if (Comand == "Normal"){ // End Cicle, if the user send the comand "Normal";
      Serial.println("Normal mode");
      a = false;
      }
     }
    }
}

};
```

Apêndice C – Código C++ do Sensor de Vento

```
#include "Arduino.h"


class Wind {

  public:
    //****************************************
    int Direction;
    int Speed;
    //****************************************

    void Update(int ang) {
      float sensorValue;
      pinMode(46, OUTPUT);
      digitalWrite(46, HIGH);

      sensorValue = (analogRead(A8));
      sensorValue =  ((sensorValue * 500) / 1023);

      float aux = Direction;

      if (sensorValue > 20 && sensorValue < 36) {
        Direction = 112.5;
      }
      if (sensorValue > 36 && sensorValue < 43) {
        Direction = 67.5;
      }
      if (sensorValue > 43 && sensorValue < 54) {
        Direction = 90;
      }
      if (sensorValue > 54 && sensorValue < 76) {
        Direction = 157.5;
      }
      if (sensorValue > 76 && sensorValue < 105) {
        Direction = 135;
      }
      if (sensorValue > 105 && sensorValue < 130) {
        Direction = 202.5;
      }
      if (sensorValue > 130 && sensorValue < 169) {
        Direction = 180;
      }
      if (sensorValue > 169 && sensorValue < 212) {
        Direction = 22.5;
      }
      if (sensorValue > 212 && sensorValue < 259) {
        Direction = 45;
      }
      if (sensorValue > 259 && sensorValue < 301) {
        Direction = 247.5;
      }
      if (sensorValue > 301 && sensorValue < 326) {
        Direction = 225;
      }
      if (sensorValue > 326 && sensorValue < 369) {
```

```
      Direction = 337.5;
    }
    if (sensorValue > 369 && sensorValue < 399) {
      Direction = 0;
    }
    if (sensorValue > 399 && sensorValue < 433) {
      Direction = 292.5;
    }
    if (sensorValue > 433 && sensorValue < 470) {
      Direction = 270;
    }
    if (sensorValue > 470 && sensorValue < 490) {
      Direction = 315;
    }


    Direction = Direction + ang - 180;
    if (Direction > 359) {
      Direction -= 360;
    }
    if (Direction < 0) {
      Direction += 90;
    }

    Direction = (aux * 0.9 + 0.1 * Direction);
  }
};
```

Apêndice D – Código C++ do Leme

```
#include "Arduino.h"


class Rudder {
  public:
    int angle = 68;

    int Update(int compass_head, int azi) {
      int delta = compass_head - azi;

      if (delta < (-180)) {
        delta = 98 + ( 360 + azi - compass_head);
      }
      if (delta > 180) {
        delta = 98 - ( 360 - compass_head + azi);
      }

      angle = (98 + (2 * delta));


      if (angle > 128) {
        angle = 128;
      }
      if (angle < 58) {
        angle = 58;
      }


      return (angle);
    }
};


// LEME A MEIO CALIBRADO 98;
```

Apêndice E – Código C++ do Ajuste da Vela

```
#include "Arduino.h"


class Sail{
public:
int angle = 180;
void Update(int WDirection, bool autonomous, int value_test){
  if (autonomous == false){
    angle = WDirection;
    Serial3.println(WDirection);
    }

  if (autonomous == true){
       if(WDirection > 180){
          angle = WDirection - value_test;
            if(angle < 90)
            angle = 90;
       }
       if(WDirection < 180){
          angle = WDirection + value_test;
            if(angle > 270)
            angle = 270;
      }
     Serial3.println(angle);
    }
  }
};
```

Apêndice F – Código C++ da *header file* “main” do Arduino Mega 2560

```
#include <math.h>
#include <Servo.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Servo.h>

#include "Main_GPS.h"
GPS gps;
#include "compass.h"
Compass compass;
#include "WayPoints.h"
WP wp;
#include "Wind.h"
Wind wind;
#include "Rudder.h"
Rudder rudder;
#include "Sail.h"
Sail sail;


//*****************************************
float DistCalc = 0;
float AziCalc = 0;
float target_head = 180;

bool autonomous = true;
bool manual = false;
bool sail_stop = false;
bool sail_auto = true;
bool sail_upwind = true;
double millisBS = 0;
String Comand = "autonomous";
int wp_number = 0;
int wp_min_dist = 12;

int AziManual = 180;
int sailManual = 180;
int sail_teste_value = 135;

File myFile;
Servo servo_rudder;

//*****************************************
void setup() {
  Serial.begin(9600); // connect serial
  Serial1.begin(9600); // connect gps sensor
  Serial2.begin(19200); // connect compass
  Serial3.begin(4800); // sail control arduino
  wp.Update();
  servo_rudder.attach(12);
  servo_rudder.write(98);
```

```
  if (!SD.begin(8)) {
      Serial.println("initialization of SD Card FAILED!");
      return;
  }
  Serial.println("Creating LOG File.");
  myFile = SD.open("LOG.txt", FILE_WRITE);
}

//*****************************************
void loop() {

  ReadComand();

  compass.Update();

  wind.Update(sail.angle);

  gps.Update();

  if(gps.GPS_LOCK){
              DistCalc =
gps.CalcDist(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));   // Calcula a distância
entre os Wp's e a posição atual
              AziCalc =
gps.CalcAzi(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));     // Calcula o Azimute
entre os Wp's e a posição atual
              wp_test();

              if(sail_upwind){
              target_head = head_to_go(AziCalc, wind.Direction);
              AziCalc = target_head;
                      }
              }



  if(autonomous == true){
     servo_rudder.write(rudder.Update(compass.head, AziCalc));
  if(sail_stop == false)
     sail.Update(wind.Direction, autonomous, sail_teste_value);
  }

  if(autonomous == false){
     servo_rudder.write(rudder.Update(compass.head, AziManual));
  if(sail_stop == false)
     sail.Update(sailManual, autonomous, sail_teste_value);
  }

  UpdateBS(500);  // Send to function the update rate in millis wanted
  }



void ReadComand(){
  if (Serial.available()) {
    Comand = Serial.readString();

    if (Comand == "sail_stop"){
      sail_stop == true;
    }
```

```
    if (Comand == "sail_go"){
      sail_stop == false;
    }
    if (Comand == "w"){
    wp_number++;
    }

    if (Comand == "sail_upwind"){
      sail_upwind == false;
    }
    if (Comand == "go_upwind"){
      sail_upwind == true;
    }

    if (Comand == "TestGPS"){
      Serial.println("Testing GPS");
      gps.Test();
     }
    if (Comand == "Manual"){
    autonomous = false;
    Serial.println("Manual mode engaged");
     }
    if (Comand == "Autonomous"){
    autonomous = true;
    Serial.println("Autonomous mode engaged");
     }
  }
    if (( Comand.substring(0,1) == "s") && (autonomous == false)){
    sailManual = Comand.substring(1,4).toInt();
    }
    if (( Comand.substring(0,1) == "h") && (autonomous == false)){
    AziManual = Comand.substring(1,4).toInt();
    }
    if (Comand.substring(0,3) == "stv")
    sail_teste_value = Comand.substring(2,6).toInt();
}


void UpdateBS(int update_time){
    if (millis() - millisBS > update_time){
    millisBS = millis();
    Serial.print("Autonomous: ");     Serial.println(autonomous);

if (autonomous == false){
    Serial.print("Azi_Manual: ");     Serial.println(AziManual);
    Serial.print("Sail_Manual: ");    Serial.println(sailManual);
}
if (autonomous == true){
    Serial.print("sailAuto: ");       Serial.println(sail.angle);
}
    Serial.print("c.head: ");         Serial.println(compass.head);
    Serial.print("c.roll: ");         Serial.println(compass.roll);
    Serial.print("c.pitch: ");        Serial.println(compass.pitch);

    Serial.print("w.dir: ");          Serial.println(wind.Direction);
    Serial.print("rudder: ");         Serial.println(rudder.angle);

    Serial.print("GPS LOCK = ");      Serial.println(gps.GPS_LOCK);
    if(gps.GPS_LOCK){
    Serial.print("gps.fix_age: ");    Serial.println(gps.fix_age);
```

```
   Serial.print("gps.lat: ");        Serial.println(gps.lat);
   Serial.print("gps.lon: ");        Serial.println(gps.lon);
   Serial.print("gps.Speed_kts: ");  Serial.println(gps.Speed_kts);
   Serial.print("gps.GPS_head: ");   Serial.println(gps.GPS_head);
   Serial.print("distância: ");      Serial.println(DistCalc);
   Serial.print("AziGPS: ");         Serial.println(AziCalc);
   Serial.print("Time: ");           Serial.println(gps.time);
   Serial.print("Date: ");           Serial.println(gps.date);
        }
   Serial.print("wpnumber: ");       Serial.println(wp_number);
   Serial.print("wplat: ");          Serial.println(wp.wplat[wp_number]);
   Serial.print("wplon: ");          Serial.println(wp.wplon[wp_number]);
     }

     myFile = SD.open("LOG.txt", FILE_WRITE);
     myFile.print(gps.lat); myFile.print(","); myFile.print(gps.lon); myFile.print(",");
myFile.print(gps.Speed_kts); myFile.print(",");
     myFile.print(wind.Direction); myFile.print(","); myFile.println(sail.angle);
     myFile.close();
  }


 void wp_test(){
          if(DistCalc < wp_min_dist)
            wp_number++;
          if(wp_number>9)
            wp_number = 0;
          if((wp.wplat[wp_number] == 0) || (wp.wplon[wp_number] == 0))
            wp_number = 0;
 }

 float head_to_go(float AziCalc, float windDirection){

   float head;
   if(AziCalc - windDirection < 30 && AziCalc - windDirection > - 30){
    if(AziCalc >= windDirection){
     head += 35 - (AziCalc - windDirection);
    }
    if(AziCalc <= windDirection){
     head -= 35 - (windDirection - AziCalc);
    }
   }

   if(AziCalc - windDirection < - 335 && AziCalc - windDirection > 335){
    if(AziCalc >= windDirection){
     head -= 35 - (360 - AziCalc + windDirection);
    }
    if(AziCalc <= windDirection){
     head += 35 - (360 - windDirection + AziCalc);
    }
   }

   if(head > 360){
    head -= 360;
   }
   if(head < 0){
    head += 360;
   }
   return(head);
 }
```

Apêndice G – Código C++ do Vetor de *Waypoints*

```
#include "Arduino.h"

class WP{
public:
  //*********************************************
    float wplat[9];
    float wplon[9];
  //*********************************************

    void Update(){
      wplat[0]= 38.6695027;
      wplon[0]= -9.1430388;

      wplat[1]= 38.6676250;
      wplon[1]= -9.1448777;

      wplat[2]= 38.6671666;
      wplon[2]= -9.1428444;

    }
};
```

Apêndice H – Código C++ do Arduino Nano

```
int angle = 600;
int sail_read1, sail_read2, sail_angle1, sail_angle2;
int diference1, diference2;
int test = 180;
int aux;

unsigned long millisS1 = 0;
unsigned long millisS2 = 0;


void setup() {

  Serial.begin(4800);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {

      digitalWrite(5, 0);
      analogWrite(4, 0);
      digitalWrite(6, 0);
      analogWrite(3, 0);

      // Estabilizar codigo, média flutuante para as ordens;
      // mais suave na mudança de atitude de velas

      if (Serial.available()) {
        aux = Serial.parseInt();
        if(aux > 89 && aux < 271){
        test = aux;
        angle = ((aux*2.0000)+280);
        }
        else{
          Serial.println("not a valid angle! Try again mate.");
          delay(1000);
        }
      }
     debug();

     Serial.print("received information: "); Serial.println(test);
     Serial.print("order given: ");       Serial.println(angle);

      delay(50);
      sail_read1 = analogRead(A4);
      sail_read2 = analogRead(A0);

   //    safe();

      diference1 = sail_read1 - angle;
      diference2 = sail_read2 - angle;

    if(angle > 200 && angle < 900){
    if (diference1 > 20){
```

```
          digitalWrite(6, 255);
          analogWrite(3, 255);
          }
      if (diference1 < -20){
          digitalWrite(6, 0);
          analogWrite(3, 255);
          }

      if (diference2 > 20){
          digitalWrite(5, 255);
          analogWrite(4, 255);
          }
      if (diference2 < -20){
          digitalWrite(5, 0);
          analogWrite(4, 255);
          }
}
      delay(50);
}

void safe(){
  if (sail_read1 > 1100 || sail_read1 <100||sail_read2 > 1100 || sail_read2 <100){
          while(true){
          Serial.println("SistemFail - Forced Shutdown");
          Serial.print("sail_read1: "); Serial.print(sail_read1); Serial.print(", sail_read2: ");
Serial.println(sail_read2);
          delay(500);
          }
        }
      }
void debug(){
        Serial.println(angle);
        Serial.print("sail_read1: "); Serial.print(sail_read1); Serial.print(", sail_read2: ");
Serial.println(sail_read2);
        Serial.print("angle: "); Serial.println(angle);
}
```

Apêndice I – Código C# da Estação em terra

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Windows.Forms;
using System.IO;


namespace LCA_1._0
{
   public partial class Form1 : Form
   {
      SerialPort ComPort = new SerialPort();

      internal delegate void SerialDataReceivedEventHandlerDelegate(object sender,
SerialDataReceivedEventArgs e);
      internal delegate void SerialPinChangedEventHandlerDelegate(object sender,
SerialPinChangedEventArgs e);
      delegate void SetTextCallback(string text);
      string InputData = String.Empty;

      public Form1()
      {
         InitializeComponent();
         ComPort.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(port_DataReceived_1);
         label30.Text = DateTime.Now.ToString("HH:mm:ss.f");
      }

      private void timer1_Tick_1(object sender, EventArgs e)
      {
         timer2.Enabled = Variables.stat;

         label30.Text = DateTime.Now.ToString("HH:mm:ss.f");
         if (Variables.stat == false)
         {
            button3.BackColor = Color.MediumVioletRed;
         }

         else
         {
            button3.BackColor = Color.MediumSpringGreen;
         }
      }

      private void button1_Click(object sender, EventArgs e)
      {
         try
        {
            string[] ArrayComPortsNames = null;
            int index = -1;
            string ComPortName = null;
            cboPorts.Items.Clear();

            //Com Ports
```

```
            ArrayComPortsNames = SerialPort.GetPortNames();
            do
            {
               index += 1;
               cboPorts.Items.Add(ArrayComPortsNames[index]);


            } while (!((ArrayComPortsNames[index] == ComPortName) || (index ==
ArrayComPortsNames.GetUpperBound(0))));
            Array.Sort(ArrayComPortsNames);

            if (index == ArrayComPortsNames.GetUpperBound(0))
            {
               ComPortName = ArrayComPortsNames[0];
            }
            //get first item print in text
            cboPorts.Text = ArrayComPortsNames[0];
            //Baud Rate
            cboBaudRate.Items.Clear();
            cboBaudRate.Items.Add(57600);
            cboBaudRate.Items.Add(9600);
            cboBaudRate.Items.Add(14400);
            cboBaudRate.Items.Add(19200);
            cboBaudRate.Items.Add(38400);
            cboBaudRate.Items.Add(115200);
            cboBaudRate.Items.ToString();
            //get first item print in text
            cboBaudRate.Text = cboBaudRate.Items[0].ToString();
         }
         catch
         {

            MessageBox.Show("Algo Correu mal, Não foi encontrado nenhum Arduino :/", "ERRO",
MessageBoxButtons.OK);
         }

      }

      private void port_DataReceived_1(object sender, SerialDataReceivedEventArgs e)
      {
         try
         {
            InputData = ComPort.ReadLine();
            if (InputData != String.Empty)
            {
               this.BeginInvoke(new SetTextCallback(SetText), new object[] { InputData });
            }
         }
         catch
         {

            MessageBox.Show("Algo Correu mal, erro ao receber data :/", "ERRO", MessageBoxButtons.OK);
         }
      }

      private void SetText(string text)
      {
         {
            try
            {
               listBox1.Items.Add(text);

               if(checkBox1.Checked==true)
               this.listBox1.SelectedIndex = this.listBox1.Items.Count - 1;
```

```
                if (text.Substring(0, 4) == "ERRO")
                {
                   richTextBox5.Text = "ERRO";
                }
                else
                {
                   //richTextBox5.Text = "GOOD";
                }

 switch (text.Substring(0,6))
{
   case "A in":
       checkedListBox1.Items.Clear();
       break;
   case "c.head":
       richTextBox1.Text = text.Substring(7, text.Length - 7);
        Variables.proa = text.Substring(7, text.Length - 7);
       break;
   case "c.roll":
       richTextBox3.Text = text.Substring(7, text.Length - 7);
       break;
   case "AziGPS":
       richTextBox15.Text = text.Substring(8, text.Length - 8);
       break;
   case "wplat:":
       richTextBox21.Text = text.Substring(7, text.Length - 7);
       break;
   case "wpnumb":
       richTextBox19.Text = text.Substring(10, text.Length - 10);
        break;
   case "gps.fi":
        richTextBox8.Text = text.Substring(13, text.Length - 13);
        break;
   case "wp_c":
        richTextBox19.Text = text.Substring(10, text.Length - 10);
        break;
   case "wplon:":
        richTextBox20.Text = text.Substring(7, text.Length - 7);
        Variables.WpLongitude = text.Substring(7, text.Length - 7);
        break;
   case "w.dir:":
        richTextBox11.Text = text.Substring(7, text.Length - 7);
        Variables.DirVento = text.Substring(7, text.Length - 7);
        break;
   case "c.pitc":
        richTextBox2.Text = text.Substring(9, text.Length-9);
        Variables.pitch = text.Substring(9, text.Length - 9);
        break;
   case "gps.la":
     richTextBox7.Text = text.Substring(9, text.Length - 9);
     Variables.latitude = text.Substring(9, text.Length - 9);
     break;
   case "gps.lo":
     richTextBox6.Text = text.Substring(9, text.Length - 9);
     Variables.longitude = text.Substring(9, text.Length - 9);
     break;

     break;
   case "rudder":
     richTextBox13.Text = text.Substring(8, text.Length - 8);
     Variables.leme = text.Substring(8, text.Length - 8);
     break;
   case "gps.Sp":
```

```
    richTextBox18.Text = text.Substring(15, text.Length - 15);
    Variables.Velocidade = text.Substring(15, text.Length - 15);
    break;
  case "distan":
    richTextBox14.Text = text.Substring(11, text.Length-11);
    Variables.WpDist = text.Substring(11, text.Length-11);
    break;

  case "sailAu":
    richTextBox12.Text = text.Substring(10, text.Length - 10);
    Variables.vela =  text.Substring(10, text.Length - 10);
    break;
  default:
    text = "";
    break;
}
            }

            catch
            {

            //    MessageBox.Show("Algo Correu mal, não foi possivel atualizar os Valores  :/", "ERRO",
MessageBoxButtons.OK);
            }

        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        try
        {
            if (button2.Text == "Ligar")
            {
                ComPort.PortName = Convert.ToString(cboPorts.Text);
                ComPort.BaudRate = Convert.ToInt32(cboBaudRate.Text);
                ComPort.Open();
                button2.Text = "Desligar";
            }
            else if (button2.Text == "Desligar")
            {
                ComPort.Close();
                button2.Text = "Ligar";
            }
        }
        catch
        {

            MessageBox.Show("Algo Correu mal, não foi possivel ligar ao Arduino :/", "ERRO",
MessageBoxButtons.OK);
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {

    }

    private void richTextBox4_KeyPress_1(object sender, KeyPressEventArgs e)
    {
        try
        {
            if (e.KeyChar == (char)13) // enter key
            {
```

```
                ComPort.Write("\r\n");
                richTextBox4.Text = "";
            }
            else if (e.KeyChar < 32 || e.KeyChar > 126)
            {
                e.Handled = true; // ignores anything else outside printable ASCII range
            }
            else
            {
                ComPort.Write(e.KeyChar.ToString());
            }
        }

        catch
        {

            MessageBox.Show("Algo Correu mal, não foi possivel enviar o comando :/", "ERRO",
MessageBoxButtons.OK);
        }
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs e)
    {

    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void button3_Click_1(object sender, EventArgs e)
    {
        Criar_log Log = new Criar_log();
        Log.ShowDialog();
    }

    private void button5_Click(object sender, EventArgs e)
    {
        listBox1.Items.Clear();
    }

    public void timer2_Tick(object sender, EventArgs e)
    {
        try
        {
            if (File.Exists(Variables.pathString))
            {
                ////// Escrever aqui o codigo para guarda na base de dados os valores proa ....
                string linha_log = "";

                if (Variables.BTime == true)
                {
                    linha_log += DateTime.Now.ToString("HH:mm:ss.f") +" $ ";
                }
                if (Variables.Bproa == true)
                {
                    linha_log += Variables.proa + " $ ";
                }
                if (Variables.Bpitch == true)
                {
                    linha_log += Variables.pitch + " $ ";
                }
```

```
if (Variables.Broll == true)
{
   linha_log += Variables.roll + " $ ";
}

if (Variables.Blatitude == true)
{
   linha_log += Variables.latitude + " $ ";
}

if (Variables.Blongitude == true)
{
   linha_log += Variables.longitude + " $ ";
}

if (Variables.BForçaVento == true)
{
   linha_log += Variables.ForçaVento + " $ ";
}

if (Variables.BDirVento == true)
{
   linha_log += Variables.DirVento + " $ ";
}

if (Variables.Bleme == true)
{
   linha_log += Variables.leme + " $ ";
}

if (Variables.Bvela == true)
{
   linha_log += Variables.vela + " $ ";
}

if (Variables.BVelocidade == true)
{
   linha_log += Variables.Velocidade + " $ ";
}

if (Variables.BWpDist == true)
{
   linha_log += Variables.WpDist + " $ ";
}

if (Variables.BWpAzi == true)
{
   linha_log += Variables.WpAzi + " $ ";
}

if (Variables.BProaToGo == true)
{
   linha_log += Variables.ProaToGo + " $ ";
}

if (Variables.BWpTime == true)
{
   linha_log += Variables.WpTime + " $ ";
}

if (Variables.BWpLatitude == true)
{
   linha_log += Variables.WpLatitude + " $ ";
}
```

```
            if (Variables.BWpLongitude == true)
            {
               linha_log += Variables.WpLongitude + " $ ";
            }


            TextWriter tw = new StreamWriter(Variables.pathString, true);
            tw.WriteLine(linha_log);
            tw.Close();
          }
        }
        catch
        {

        }
    }

    private void webBrowser1_DocumentCompleted(object sender,
WebBrowserDocumentCompletedEventArgs e)
    {

    }
  }

  class Variables
  {
    public static bool stat = false;
    public static string pathString;

    public static string proa;
    public static string pitch;
    public static string roll;
    public static string latitude;
    public static string longitude;
    public static string ForçaVento;
    public static string DirVento;
    public static string leme;
    public static string vela;
    public static string Velocidade;
    public static string WpDist;
    public static string WpAzi;
    public static string ProaToGo;
    public static string WpTime;
    public static string Time;
    public static string WpLatitude;
    public static string WpLongitude;

    public static bool Bproa;
    public static bool Bpitch;
    public static bool Broll;
    public static bool Blatitude;
    public static bool Blongitude;
    public static bool BForçaVento;
    public static bool BDirVento;
    public static bool Bleme;
    public static bool Bvela;
    public static bool BVelocidade;
    public static bool BWpDist;
    public static bool BWpAzi;
    public static bool BWpLatitude;
    public static bool BWpLongitude;
    public static bool BProaToGo;
```

```
        public static bool BWpTime;
        public static bool BTime;

    }
}
```

Apêndice J – Código C# da form "Create Log" da Estação em Terra

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;


namespace LCA_1._0
{
   public partial class Criar_log : Form
   {
      public Criar_log()
      {
         InitializeComponent();
         comboBox1.Items.Add(0.1);
         comboBox1.Items.Add(0.25);
         comboBox1.Items.Add(0.5);
         comboBox1.Items.Add(1);
         comboBox1.Items.Add(3);
         comboBox1.Items.Add(5);
         comboBox1.Items.Add(10);
         comboBox1.Items.Add(30);
         comboBox1.Items.Add(60);
         comboBox1.Items.Add(120);

          if (Variables.stat == true)
          {
             button2.BackColor = Color.MediumSpringGreen;
             button2.Text = "Ligado";
          }
          else
          {
             button2.BackColor = Color.MediumVioletRed;
             button2.Text = "Desligado";
          }
      }

      private void Criar_log_Load(object sender, EventArgs e)
      {

      }


      private void button2_Click(object sender, EventArgs e)
      {

         // Verifica quais as variaveis que deve guardar
         bool AnyChecked = false;

         if (checkBox1.Checked == true)
         {
            Variables.BTime = true;
            AnyChecked = true;
         }
         else
```

```
{
   Variables.BTime = false;
}

if (checkBox2.Checked == true)
{
   Variables.Blatitude = true;
   AnyChecked = true;
}
else
{
   Variables.Blatitude = false;
}

if (checkBox3.Checked == true)
{
   Variables.Blongitude = true;
   AnyChecked = true;
}
else
{
   Variables.Blongitude = false;
}

if (checkBox13.Checked == true)
{
   Variables.BVelocidade = true;
   AnyChecked = true;
}
else
{
   Variables.BVelocidade = false;
}

if (checkBox4.Checked == true)
{
   Variables.Bproa = true;
   AnyChecked = true;
}
else
{
   Variables.Bproa = false;
}

if (checkBox7.Checked == true)
{
   Variables.Bpitch = true;
   AnyChecked = true;
}
else
{
   Variables.Bpitch = false;
}

if (checkBox8.Checked == true)
{
   Variables.Broll = true;
   AnyChecked = true;
}
else
{
   Variables.Broll = false;
}
```

```
if (checkBox5.Checked == true)
{
   Variables.Bvela = true;
   AnyChecked = true;
}
else
{
   Variables.Bvela = false;
}

if (checkBox6.Checked == true)
{
   Variables.Bleme = true;
   AnyChecked = true;
}
else
{
   Variables.Bleme = false;
}

if (checkBox11.Checked == true)
{
   Variables.BProaToGo = true;
   AnyChecked = true;
}
else
{
   Variables.BProaToGo = false;
}

if (checkBox12.Checked == true)
{
   Variables.BWpLatitude = true;
   AnyChecked = true;
}
else
{
   Variables.BWpLatitude = false;
}

if (checkBox14.Checked == true)
{
   Variables.BWpLongitude = true;
   AnyChecked = true;
}
else
{
   Variables.BWpLongitude = false;
}

if (checkBox15.Checked == true)
{
   Variables.BWpTime = true;
   AnyChecked = true;
}
else
{
   Variables.BWpTime = false;
}

if (checkBox9.Checked == true)
{
   Variables.BDirVento = true;
   AnyChecked = true;
```

```
        }
        else
        {
            Variables.BDirVento = false;
        }

        if (checkBox10.Checked == true)
        {
            Variables.BForçaVento = true;
            AnyChecked = true;
        }
        else
        {
            Variables.BForçaVento = false;
        }

        if (checkBox16.Checked == true)
        {
            Variables.BWpAzi = true;
            AnyChecked = true;
        }
        else
        {
            Variables.BWpAzi = false;
        }

        if (checkBox17.Checked == true)
        {
            Variables.BWpDist = true;
            AnyChecked = true;
        }
        else
        {
            Variables.BWpDist = false;
        }


        if (AnyChecked == true)
        {
            string folder = @"C:\LCA Logs";

            if (!Directory.Exists(folder))
            {
                Directory.CreateDirectory(folder);
            }
            Variables.pathString = System.IO.Path.Combine(folder,
(DateTime.Now.ToString("yyyy/MM/dd_HHmmss") + ".txt"));

            if (Variables.stat == true)
            {
                Variables.stat = false;
                button2.Text = "Desligado";
                button2.BackColor = Color.MediumVioletRed;
            }
            else
            {
                try
                {
                    if (!File.Exists(Variables.pathString))
                    {
                        File.Create(Variables.pathString);
                        TextWriter tw = new StreamWriter(Variables.pathString);
                        tw.WriteLine("The very first line!");
                        tw.Close();
```

```
                    File.Delete(Variables.pathString);
                }
            }
            catch
            {
            }

            Variables.stat = true;
            button2.Text = "Ligado";
            button2.BackColor = Color.MediumSpringGreen;
        }
    }
    else
    {
        MessageBox.Show("Seleciona pelo menos 1 item para guardar ;)", "ERRO",
MessageBoxButtons.OK);
    }

  }
 }
}
```

Apêndice K – Relação dos Custos

item	quantidade	Preço Total	Qauntidade para construção de um novo veleiro	Preço para construção de novo veleiro	URL material
PLA	5	74,75	2	29,9	https://www.pccomponentes.com/bq-pla-premium-1-75-mm-transparente-1kg-fabricado-en-espana
Fibra de vidro	12m2	270	7	70,25	http://shop1.r-g.de/en/art/190148
Epoxy	5 kg	220	2	40,22	http://shop1.r-g.de/en/art/100110
Gelcoat	4kg	130	2	65	http://shop1.r-g.de/en/art/120105
Tubo de Carbono	4m	57,28	4	57,28	http://shop1.r-g.de/en/art/740908-HM
Perfis Tem Aço Laminado	5m	61,25	2,5	30,625	
Perfis Tubulares em aço laminado	2m	25,26	0,5	6,315	
Arduino Mega 2560	2	19,26	1	9,63	https://www.aliexpress.com/item/Mega-2560-R3-Mega2560-REV3-ATmega2560-16AU-Board-USB-Cable-compatible-for-arduino/32238296519.html?spm=2114.30010308.3.10.6o2b43&s=p&ws_ab_test=searchweb201556_7,searchweb201602_5_10057_10056_10065_10068_10055_10054_10069_10059_10058_10073_10017_10070_10060_10061_10052_10062_10053_10050_10051,searchweb201603_3&btsid=de7e7891-66a8-4f2b-990d-6da812a1a99a8
Arduino Nano	2	5,06	0	0	https://www.aliexpress.com/item/NANO-3-0-controlador-compatible-con-arduino-NANO-CH340-turno-USB-controlador-ninguna-CABLE-V3-0/32574307428.html?spm=2114.30010308.3.1.IpG98c&ws_ab_test=searchweb201556_7,searchweb201602_5_10057_10056_10065_10068_10055_10054_10069_10059_10058_10073_10017_10070_10060_10061_10052_10062_10053_10050_10051,searchweb201603_3&btsid=e1ee96fa-7523-49a9-b35b-0c9571e592da
Trasistors / Reles / Resistencias	-	120	-	40	
Conectores	-	25	-	5	
GPS	3	25,35	1	8,45	https://www.aliexpress.com/item/GY-NEO6MV2-new-NEO-6M-GPS-Module-NEO6MV2-with-Flight-Control-EEPROM-MWC-APM2-5-large/32613723691.html?spm=2114.30010308.3.66.IhyIgA&ws_ab_test=searchweb201556_7,searchweb201602_5_10057_10056_10065_10068_10055_10054_10069_10059_10058_10073_10017_10070_10060_10061_10052_10062_10053_10050_10051,searchweb201603_3&btsid=1d77652b-7d8a-415a-a1b9-22e51df2adb2
Bussula	1	2,24	1	2,24	https://www.aliexpress.com/item/ADXL335-3-Axis-Compass-Accelerometer-Module-GY-61-for-arduino/32725068981.html?spm=2114.30010308.3.47.5yFVs9&ws_ab_test=searchweb201556_7,searchweb201602_5_10057_10056_10065_10068_10055_10054_10069_10059_10058_10073_10017_10070_10060_10061_10052_10062_10053_10050_10051,searchweb201603_3&btsid=1420799b-9a96-4288-82c5-2f6ec7eebaac
modulo comunicações	2	42,32	1	26,16	https://www.aliexpress.com/item/Hot-Sale-High-Quality-APC220-Wireless-Data-Communication-Module-USB-Adapter-Kit-For-Arduino/32657825283.html?spm=2114.30010308.3.83.jqzRtM&ws_ab_test=searchweb201556_7,searchweb201602_5_10057_10056_10065_10068_10055_10054_10069_10059_10058_10073_10017_10070_10060_10061_10052_10062_10053_10050_10051,searchweb201603_3&btsid=285e6eec-14a8-4253-a464-a8a45b8ef15c
outros materiais de acabamento	-	150		50	
outros materiais eletronicos		150		50	
Veyron Driver	1	99,99	1	99,99	http://www.botnroll.com/pt/controladores/1261-veyron-driver-para-2-motores-dc-c-escovas-ate-25a.html?search_query=controlador&results=127
Servo leme	2	78,2	1	39,1	https://www.hpmodelismo.com/gb/servos/1990-servo-hitec-645mg-96kg-020s-60.html?search_query=hitec&results=40
motores controlo matro	3	36	2	24	
Sprays de pintura	5	87	-	-	
perfis alumínio	-	50	-	-	
Plastico cobertura mastros	4m2	40			
		1768,96		654,16	

Apendice L – Código ficheiro CPP *Main* após incorporação do constrolador Veyron Driver

```
#include <math.h>
#include <Servo.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <Servo.h>
#include "DFRobotVeyronBrushed.h"

DFRobotVeyronBrushed VeyronBrushed1;


#include "Main_GPS.cpp"
GPS gps;
#include "compass.cpp"
Compass compass;
#include "WayPoints.cpp"
WP wp;
#include "Wind.cpp"
Wind wind;
#include "Rudder.cpp"
Rudder rudder;
#include "Sail.cpp"
Sail sail;

#define RUDDER_PIN 12 // the rudder is connected to pint 12



//*****************************************
float DistCalc = 0; // float dist2wp = 0; // dist2wp is the distance (in meters) to the next Waypoint2
float AziCalc = 0;
float target_head = 180;  // target_head is the aymuth that we should follow to reach the WP (which may be different
                    // from azi2wp because we may have to tack.

bool autonomous = true;
bool manual = false;
bool sail_stop = false;
bool sail_auto = true;
bool sail_upwind = true;
long double millisBS = 0;
long double millis_update_sail = 0;
long double millis_desvia = 0;


String Comand = "autonomous";
int wp_number = 0;
int wp_min_dist = 10;

int AziManual = 180;
int sailManual = 180;
int sail_teste_value = 135;
  int id = 1;            //The ID of the Veyron
  int motorNumber = 1;    //The motor number: 1 for M1 and 2 for M2
```

```
  int speed;              //Store the speed of the motor in RPM (Revolutions Per Minute)
 int sail_read1, sail_read2;
int sail_error = 15;

File myFile;
Servo servo_rudder;

//*********************************************
void setup() {
  Serial.begin(9600); // connect serial
  Serial1.begin(9600); // connect gps sensor
  Serial2.begin(19200); // connect compass
  Serial3.begin(57600); // mast control arduino
  VeyronBrushed1.begin(Serial3);
  VeyronBrushed1.setSpeed(id, 1, 0);
  VeyronBrushed1.setSpeed(id, 2, 0);

  wp.Update();

  servo_rudder.attach(RUDDER_PIN);    // the rudder is connected to pin 12
  int rudder_center_value = 98;
  servo_rudder.write(98);     // initilise the rudder to the center position (which we estimate will be 98
degrees)

  if (!SD.begin(8)) {
      Serial.println("initialization of SD Card FAILED!");
      return;
  }
  Serial.println("Creating LOG File.");
  myFile = SD.open("LOG.txt", FILE_WRITE);
}

//*********************************************
void loop() {


  ReadComand();
  compass.Update();
  wind.Update();
  gps.Update();
  if(gps.GPS_LOCK){
              DistCalc =
gps.CalcDist(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));   // Calcula a distância
entre os Wp's e a posição atual
              AziCalc =
gps.CalcAzi(gps.lat,gps.lon,(wp.wplat[wp_number]),(wp.wplon[wp_number]));     // Calcula o Azimute
entre os Wp's e a posição atual
              wp_test();

              if(sail_upwind){
              target_head = head_to_go(AziCalc, wind.Direction);
              AziCalc = target_head;
                 }
              }

  if(autonomous == true){
     servo_rudder.write(rudder.Update(compass.head, AziCalc));
  if(sail_stop == false)

     if(millis()- millis_update_sail > 7500){
```

```
    sail.Update(wind.Direction, autonomous, sail_teste_value);
    millis_update_sail = millis();
    }
    sail_ajust();
  }
  else{
    servo_rudder.write(rudder.Update(compass.head, 180));
    sail.Update(wind.Direction, true, sail_teste_value);
  }

    if(millis() - millis_desvia > 30000)
    {
      autonomous = true;
      millis_desvia = millis();
    }

  if(sail_stop == false){

    if(millis()-millis_update_sail > 7500){
    sail.Update(sailManual, autonomous, sail_teste_value);
    sail.angle = sailManual;
    millis_update_sail = millis();
    }
    sail_ajust();
  }
  UpdateBS(500);  // Send to function the update rate in millis wanted

  if(analogRead(A12) > 900){
    autonomous = false;
    servo_rudder.write(rudder.Update(compass.head, 180));
    sail.Update(wind.Direction, autonomous, sail_teste_value);
  }

  }



void ReadComand(){
  if (Serial.available()) {
    Comand = Serial.readString();

    if (Comand == "sail_stop"){
      sail_stop == true;
    }
    if (Comand == "sail_go"){
      sail_stop == false;
    }
    if (Comand == "w"){
    wp_number++;
    }

    if (Comand == "a"){
    autonomous = false;
    servo_rudder.write(rudder.Update(compass.head, 180));
    sail.Update(wind.Direction, autonomous, sail_teste_value);
    }

    if (Comand == "sail_upwind"){
      sail_upwind == false;
    }
```

```
   if (Comand == "go_upwind"){
     sail_upwind == true;
   }

   if (Comand == "TestGPS"){
     Serial.println("Testing GPS");
     gps.Test();
    }
   if (Comand == "Manual"){
   autonomous = false;
   Serial.println("Manual mode engaged");
    }
   if (Comand == "Autonomous"){
   autonomous = true;
   Serial.println("Autonomous mode engaged");
    }
 }
   if (( Comand.substring(0,1) == "s") && (autonomous == false)){
   sailManual = Comand.substring(1,4).toInt();
   if(sailManual > 90 && sailManual < 270)
   sail.angle = sailManual;

   }
   if (( Comand.substring(0,1) == "h") && (autonomous == false)){
   AziManual = Comand.substring(1,4).toInt();
   }
   if (Comand.substring(0,3) == "stv")
   sail_teste_value = Comand.substring(2,6).toInt();
}


void UpdateBS(int update_time){
   if (millis() - millisBS > update_time){
   millisBS = millis();
   Serial.print("Autonomous: ");     Serial.println(autonomous);

if (autonomous == false){
   Serial.print("Azi_Manual: ");     Serial.println(AziManual);
   Serial.print("Sail_Manual: ");    Serial.println(sailManual);
}
if (autonomous == true){
   Serial.print("sailAuto: ");       Serial.println(sail.angle);
}
   Serial.print("c.head: ");         Serial.println(compass.head);
   Serial.print("c.roll: ");         Serial.println(compass.roll);
   Serial.print("c.pitch: ");        Serial.println(compass.pitch);

   Serial.print("w.dir: ");          Serial.println(wind.Direction);
   Serial.print("rudder: ");         Serial.println(rudder.angle);

   Serial.print("GPS LOCK = ");      Serial.println(gps.GPS_LOCK);
   if(gps.GPS_LOCK){
   Serial.print("gps.fix_age: ");    Serial.println(gps.fix_age);
   Serial.print("gps.lat: ");        Serial.println(gps.lat);
   Serial.print("gps.lon: ");        Serial.println(gps.lon);
   Serial.print("gps.Speed_kts: ");  Serial.println(gps.Speed_kts);
   Serial.print("gps.GPS_head: ");   Serial.println(gps.GPS_head);
   Serial.print("distância: ");      Serial.println(DistCalc);
   Serial.print("AziGPS: ");         Serial.println(AziCalc);
   Serial.print("Time: ");           Serial.println(gps.time);
```

```
   Serial.print("Date: ");          Serial.println(gps.date);
        }
   Serial.print("wpnumber: ");      Serial.println(wp_number);
   Serial.print("wplat: ");         Serial.println(wp.wplat[wp_number]);
   Serial.print("wplon: ");         Serial.println(wp.wplon[wp_number]);
      }

      myFile = SD.open("LOG.txt", FILE_WRITE);
      myFile.print(gps.lat); myFile.print(","); myFile.print(gps.lon); myFile.print(",");
myFile.print(gps.Speed_kts); myFile.print(",");
      myFile.print(wind.Direction); myFile.print(","); myFile.println(sail.angle);
      myFile.close();
    }


 void wp_test(){
            if(DistCalc < wp_min_dist)
              wp_number++;
            if(wp_number>9)
              wp_number = 0;
            if((wp.wplat[wp_number] == 0) || (wp.wplon[wp_number] == 0))
              wp_number = 0;
 }

 float head_to_go(float AziCalc, float windDirection){
   float head;
   if(AziCalc - windDirection < 45 && AziCalc - windDirection > - 45){
     if(AziCalc >= windDirection){
       head += (25 - (AziCalc - windDirection));
     }
     if(AziCalc <= windDirection){
       head -= (25 - (windDirection - AziCalc));
     }
   }

   if(AziCalc - windDirection < - 315 && AziCalc - windDirection > 315){
     if(AziCalc >= windDirection){
       head -= (25 - (360 - AziCalc + windDirection));
     }
     if(AziCalc <= windDirection){
       head += (25 - (360 - windDirection + AziCalc));
     }
   }

   if(head > 360){
     head -= 360;
   }
   if(head < 0){
     head += 360;
   }
   return(head);
 }

 void sail_ajust(){

      sail_read1 = analogRead(A12);
      sail_read2 = analogRead(A14);

      if(sail.angle == 666){
       speed = sail_read1+30 - 790; // Update sail 1
```

```
    if ( speed > sail_error ||  speed < -sail_error){
       speed_normalization();
       VeyronBrushed1.setSpeed(id, 1, speed);//Stop the motor 1
          }
        else{
        VeyronBrushed1.setSpeed(id, 1, 0);
        }

         speed = sail_read2-20 - 460;   //Update sail 2
     if ( speed > sail_error ||  speed < -sail_error){
       speed_normalization();
       VeyronBrushed1.setSpeed(id, 2, speed);//Stop the motor 1
          }
        else{
        VeyronBrushed1.setSpeed(id, 2, 0);
        }
   }

    else{

         speed = sail_read1 + 30 - (((sail.angle)*2.0000)+300);// Update sail 1
    if ( speed > sail_error ||  speed < -sail_error){
       speed_normalization();
       VeyronBrushed1.setSpeed(id, 1, speed);//Stop the motor 1
          }
        else{
        VeyronBrushed1.setSpeed(id, 1, 0);
        }

         speed = sail_read2 - (((sail.angle)*2.0000)+300);   //Update sail 2
     if ( speed > sail_error ||  speed < -sail_error){
       speed_normalization();
       VeyronBrushed1.setSpeed(id, 2, speed);//Stop the motor 1
          }
        else{
        VeyronBrushed1.setSpeed(id, 2, 0);
        }
     }
   }

   void speed_normalization(){
       if(speed > 200)
         speed = 200;
       if(speed > 0 && speed < 130)
         speed = 130;
       if(speed < 0 && speed > -130)
         speed = -130;
       if(speed < -200)
         speed = -200;

}
```

Apêndice M – Artigo submetido na conferência SEA-CONF16

Implementation of a Machine Learning Algorithm in an Autonomous Sailboat

Pedro Castro Fernandes, Mario Monteiro Marques,
Victor Lobo CINAV – Escola Naval
pedro.castro.fernandes@marinha.pt

Abstract

The sea always pumped up human's curiosity. We have been exploring it since the beginning of time. It had always an important role in the society since we make use of it in several activities like collecting resources, dispatching merchandise or just for recreating activities. The number of ships crossing the oceans is incredible high, and there are a lot of illegal activities.

A robotic sailing boat is a complex system. It has several parts that come together in a specific order to achieve the goal of sailing. The important sensors are the wind meter, the compass and the GPS sensor, but, this project, as also other sensors like a SD card reader sensor and a 433MHz trans-receiver. This sensors describes the environment. This paper presents an online machine learning agent developed to control a small scale sail autonomous sail boat. Implement this is a big challenge, as we are running our project into an Arduino mega, clocking at the speed of 16 Mhz. Design this agent is complex, because, sailing depends on a

lot of variables, and we have restricted processing capacity. In this paper we describe the variables that we used to construct the different matrix and how they become a usable information to the boat successfully sail.

Keywords: Sailboat, autonomous, Machine Learning.

I. Introduction

The ocean not only provides a way of communication and transportation, it also is used to collect important resources with major impact in economies. Nowadays, countries are interested in maintain a constant surveillance on the oceans in order to guaranty the sovereignty and the control of that resources. The idea of using autonomous sailing boats isn't new, since there are other people developing similar devices [1], however it is always good to have different approaches to the same problem, because each solution cuts both ways as anything in engineering, having strong

as weak points, being better in some aspects, and worst in another's.

A boat to stay long periods of time in the sea as to be self-sustainable in energy, and being a sailboat is a good solution, although, being environmental dependent can be bad, as the wind can stop, and we lose the propulsion, however, most of the times we got enough wind to sail.

People see an autonomous device as a physical platform that receive inputs and perform a given task, in function of internal instructions implemented on the device [2]. Our goal is to save human labour, so, our device, as to be independent as possible. In general people see as a good thing the use of these devices to do this kind of jobs [3], although, at this moment, we are only using the sail boat in competitions. We expect that, in the future, it can be used in real missions of surveillance, patrol or search and rescue.

This paper focus on the development of an agent. This agent tries to learn the best way to sail the boat in which is installed. Our main goal was to develop an agent that can be used in different types of sailboats and take advantage on the particularities of the platform in which is performing.

This paper is divided as follows: in the section II is described the architectures used in the system, the sensors, actuators and processor unit used. On the section III, we talk about the Arduino shield and how it was programed to be able to read all sensors data and communicate. In this section is also explained the shield developed in this project that groups all sensors signals and output them in the Arduino. In section IV we talk about the agent developed. The agent is an online Machine Leaning algorithm that performs hill climbing process to converge to the best sail configuration. In the section V, we talk about future work that is going to be implemented on the work develop. The section VI is the conclusion of the paper.

II. System Architecture

Sailing is a complex job. We got a lot of things to take care before we successfully sail. For a ship to sail, there are a lot of different jobs to be profited. In a really ship, we got different persons taking care of the different jobs, in order to everything run smoothly. In our boat, we have only one processor, so, metaphorical we got only one person, although, that person think fast, and can make a lot of jobs quickly. The processor unit we are using is an Arduino Mega 2560[4]. This is a microcontroller board based on the microprocessor ATmega2560. It has 54 digital input/output pins, 16 analog inputs and 4 universal asynchronous receiver/transmitter (UART), which can be used as hardware serial ports. We use three UART ports to sensors: Compass; GPS and wireless communications. The anemometer works with one Hall Effect sensor, connected to an analogic port.

Different directions or speed of wind are characterized for different voltages outputs. Those voltages are read in the Arduino and converted to numerical values.

In our sailing boat, we are using the following list of parts, some shown on the Fig. 1:

- GPS – Updates positions with a frequency of 1 Hz
- Compass – Besides de head, it also indicates the pitch and roll of the boat
- Rudder – powered by an servo
- Sail – powered by a servo
- Anemometer
- Communications – 433MHz (100mW)
- Batteries – 12.1V and 3000 mAh
- Arduino Mega ATM 2560
- SD card reader

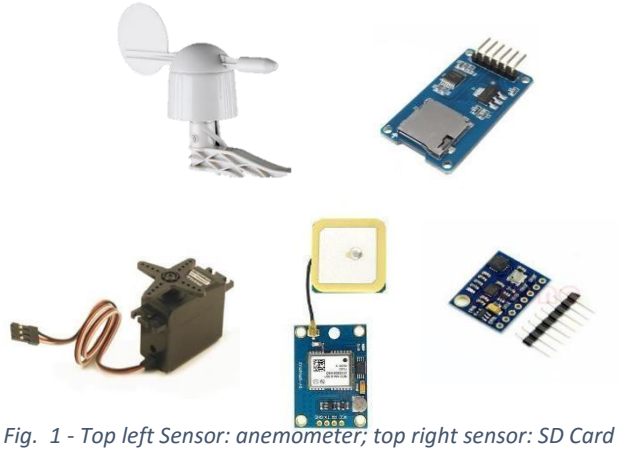


Fig. 1 - Top left Sensor: anemometer; top right sensor: SD Card Reader; botom left: Rudder Servo; botom center: GPS sensor; botom right: Compass sensor

We implemented this architecture on a lazer sailboat with one meter length, but this solution can be used in other type of boats.

We used two programming languages: C++ and C#. The C++ is the language that was used to program the Arduino that controls the boat. The C# was used in the development of a debugger. We call this application backstation

The boat has three modes of operation. The first one is totally user dependent. On the back station the user define the rudder and sail angle. This mode is used to debug and testing the boat, or to collect information, for instance the coordinates of a certain place. Other mode, the Patrol mode, is used to develop the knowledge of the agent, by testing sail angles and update the matrix. The last mode, the compete mode, is used to the boat follow waypoints using the knowledge stored in the matrix. The modes of function of the boat will be described in detail further in this article. The backstation Software is a program wrote in C#. The boat can function completely independent from this, although, we have several vantages in having this program running with the boat. It was developed in order to help detect malfunctions, and to manual control the boat, sending commands directly to the rudder or to the sail.

This application displays all data read or generated by the boat. The information gets to the application by a COM port where we attach the communication hardware. The link between the boat and the back station is assured by the transreceiver sensor, shown in Fig. 2.

Fig. 2 - Comunication Sensor

This is a 100mW communications sensor, with a range of 1km in open space. If the boat goes out of the connection range it continues doing

its job, as it is completely independent from the back station.

The information read by the *backstation* can be saved in log files and is as shown in Fig. 3.

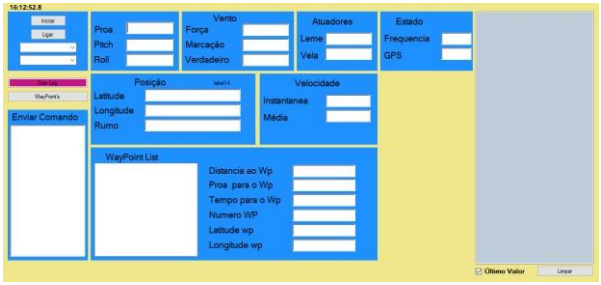


Fig. 3 - Backstation main page

III. Development of the Arduino shield and C++ Code

The first step integrate all sensors in the Arduino. To do that, we develop a PCB board. It was draw using the freeware application *EAGLE schematics*. Using the data sheet of the sensors, we stablish which pins of the Arduino would be used with which sensor. The final board is as represented in figure Fig. 4.

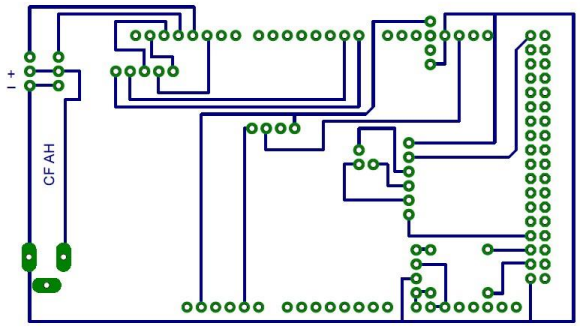


Fig. 4 - Shield used in the arduino

This PCB board, or Arduino Shield, clusters the signals of the different sensors. The next step was to develop the Arduino code, in order to the Arduino interpret the signal of which sensor. The sensors wore connected to the Arduino by UART serial ports. The Arduino Mega has four Serial ports, making the job easy. For the GPS, we used an open library that convert the GPS signal in variables: Angular latitude and longitude, age of the signal, speed in knots and course. The anemometer generates a different voltage in function of the wind direction. It can be measured with an analogic pin in the Arduino. The control of the servos was made using an Arduino open library.

The code was written in C++ using the Arduino Integrated Development Environment (IDE). We used different header files to different parts of the code, in order to simplify the comprehension of the code. The main function call the other ones when needed. Which sensor has it own header file. If we need to use a different sensor, is only necessary to change this part of the code. The main function as Boolean variables to set the different modes of function the boat. By default, the boat acts in compete mode. The waypoints can be stored in the SD Card, or written in hardcode directly in the microcontroller.

IV. The Agent

Sailing depends on two major components: the ones that are related with the platform and the ones that related with the environment and the surroundings. We can study and deeply understand the way that our platform works. We know its characteristics, we know the code, we know the components and the way they are relate in function of the circuits we created. We

can also know the surrounding, but, this is harder to define, the sea is a severe environment and it is constantly changing. It is hard to define it, and, it is even harder to define the relation that the boat will create with it. Most autonomous sail boats use reactive agents [5]. Those algorithms tell the boat how to sail, but, if we want the sailboat to be effective as possible, we have to put itself exploring that interacting and discovering the best sail configuration in each situation.

On-Line Algorithms and Machine Learning are areas that study the problematic of making decisions from limited information [6]. This method is characterized by the data used in the learning process become available in a sequential order, step by step. This area of machine learning has been extensively studied [6].

The ability to adapt to the surroundings as a very important role in autonomous vehicles. This could be particularly important in the situation that something goes wrong with the boat, like for instance, the sail gets damaged and the boat begins to behaviour differently. In this case, if the boat as the ability to test, it could adapt and still be able to get to its destination.

We implemented an online searching agent [5]. The first step of this kind of algorithm is looking for the environment stat, the sensors readings data. This data is saved as a vector of a space in R^n. We attached a SD Card Reader to the Arduino that controls the boat, this way, we can store there all data. The readings that we are saving are: difference between the head and the wind direction, speed of approaching to a waypoint, and sail angle. Those are the ones that most influence the sailing. We could use more variables, for instance: wind speed, pitch and roll, but, the space would become significantly larger. The processor unit wouldn't react fast enough and it would take much time to collect data to fill the space. One set of this readings is a vector, and all te vectors put together are the matrix that we store in the SD Card.

The next step is filling the matrix. The method that we are using to do this is the downhill simplex method [7]. This method iteratively approaches an optimal solution [8]. We set that, for the same conditions of difference of head and wind direction, the boat would try the sail full at starboard, full at portside, and half position. Then, it would choose the best two sail angles, the sail angles that provide more velocity of approaching, and try the sail middle position between and repeat this five times, narrow the sail angle to a range of about 5 degrees. The wind and the head are always changing, so, we set an interval of 10degrees. If the value goes off this range, that data is kept way, and it restarts the algorithm.

It will take much time to save all possible situations, and it all depends on the environmental conditions like the wind. We set that we would fill variable difference between head and wind direction in steps of 5 degrees.

There are a range of differences that are impossible to sail. Normally, the sail boat can't sail upwind with less than 30 degrees. In this cases the speed is going to be null or negative, and the agent will steep to the next test frame.

The final step is to organize all data in a way that the sail boat can use the information to sail. It creates a new matrix, and stores only the sail angles that give the best speed, in function of the values of difference between head and wind speed.

The boat will have two function modes:

- Patrol – Is in this mode that the boat can run the algorithms of machine learning, and try new things and update the matrix of decisions. The diagram shown in Fig. 5 represents the used agent.

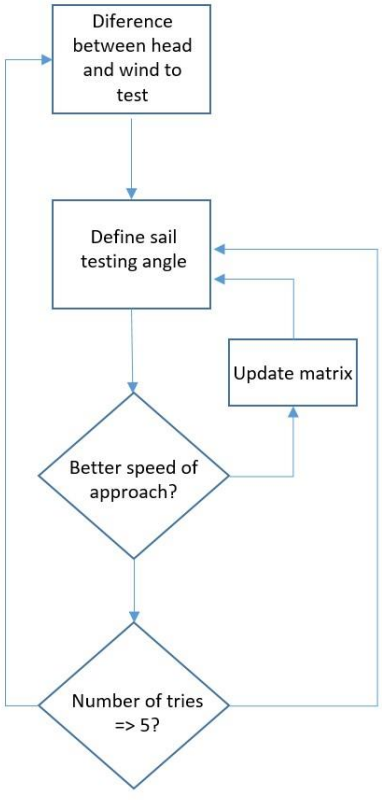


Fig. 5 - Agent Fluxogram

- Compete – In this mode the boat don't try to improve the sailing knowledge, it uses the previous values from the matrix. In there, isn't all possible situation of values.
  When the agent can't find one equal, it uses the nearest neighbour, using Manhattan distance [9] between the situations saved on the matrix and the present situation and applies the one with the less distance.

V. Future Work

Nowadays we see a big growth in the number of autonomous systems. There are several systems, since another's autonomous boats, UAV's among others, and it will be a good idea to put all this systems connected together in somehow. There have been made an effort in order to achieve this goal with the development of Joint Architecture for Unmanned Systems (JAUS), for instance the Standard: AS5684 (SAE). It is a protocol to communicate between autonomous systems. It is important to develop and implement that kind of solution in this project.

It is also necessary to develop a management system to the electrics, at this moment the sail boat is only powered by the on board batteries. It is necessary to think about the implementation of renewable power sources, in order to, give to the boat a better performance in endurance tasks.

VI. Conclusion

In this article we describe the agent we implemented in our sail boat. We start talking about the concept of autonomous sailing boats and the requirements of such a device. Next, we focused on the Machine Learning concept, and how it should be implement on an autonomous sail boat. Finally we exposed the architecture of our system and the software's developed.

Build an agent to autonomous sail boat is complex. It has to mine a lot of data to successfully sail. We were able to implement algorithms capable of handle such complex task in a microcontroller running with a processor clocking at 16 MHz. Thus, we develop a new approach to the problem of autonomous sailing. Our goal was to make it efficient as possible, making it possible to be used in other project or in real life operations.

References

[1] Alves, José C. and Cruz, Nuno A. "FASt - An autonomous sailing platform for oceanographic missions" , Quebec, 2008.

[2] Grollman, D. H. Brown University, Providence, RI Dept. of Comput. Sci., and O. C. Jenkins, "Dogged Learning for Robots" , Roma, 2007.

[3] Ray, C.; Mondada, F. and Siegwart, R. "What do people expect from robots?", Nice, 2008.

[4] ARDUINO. (2016) Arduino MEGA 2560 & Genuino MEGA 2560. [Online]. https://www.arduino.cc/en/Main/ArduinoBoardMega2560

[5] Costa, Ernesto and Simões, Anabela. *Inteligência Artificia - Fundamentos e Aplicações*.: FCA, 2008.

[6] Blum, Avrim. *On-Line Algorithms - The State of the Art*. Pittsburgh: Fiat and Woeginger eds, 1998.

[7] Koshel, R. John. "Enhancement of the downhill simplex method of optimization," , Tucson, 2002.

[8] Gao, Lixing Han Fuchang."Implementing the Nelder-Mead simplex algorithm," vol. Volume 51, no. Issue 1, pp 259-277, January 2012.

[9] Madhulatha, T. Soni. "An Overview On Clustering Methods" vol. Vol. 2(4), no. pp: 719-725, 2012.

[10] Groenen, Patrick J.F.; Kaymaky, Uzay and Rosmalen, Joost van. "Fuzzy clustering with Minkowski distance functions" Rotterdam, EI 2006-24, July 6, 2006.

[11] Blum, Avrim and Mitchell, Tom. "Combining labeled and unlabeled data with co-training" , New York, 1998.

Apêndice N – Artigo submetido na conferência MARTECH16

Barlavento – Construção de um Veleiro Autónomo

Pedro Castro Fernandes, Mário Monteiro Marques, Victor Lobo

Centro de Investigação Naval, Escola Naval
Email: pedro.castro.fernandes@marinha.pt

Abstract – A monitorização persistente do oceano é algo extremamente importante para compreender melhor o clima e dinâmica dos oceanos, para melhorar a segurança da navegação, para fazer uma melhor gestão dos seus recursos e para evitar e combater abusos ao meio ambiente. No entanto essa monitorização persistente é extremamente cara, e só pode ser conseguida com o recurso, entre outros, a meios não tripulados sejam eles aéreos, de superfície ou subaquáticos. No entanto este tipo de veículos necessita de ter uma grande autonomia energética, ou recolher a energia que necessita do próprio meio ambiente, neste sentido, os veleiros autónomos podem ser uma boa solução para a monitorização persistente do oceano. O objetivo deste trabalho é o desenvolvimento de um veleiro autónomo capaz cumprir esse propósito. Desde de 2010 a Escola Naval tem um projeto de veleiro autónomo. Contudo esse demonstra fragilidades dado à plataforma não ser resistente às intempéries marinhas e não dispor de espaço interior para a integração de tecnologias diversas. Este artigo explicita os desenvolvimentos que a Escola Naval tem feito desde 2015 com vista ao desenvolvimento do novo veleiro autónomo. Este artigo descreve a criação do casco com recurso a tecnologias CAD 3D e simulação hidrodinâmica.

Introdução

Inserido numa linha de desenvolvimento de interesse do Centro de Investigação Naval (CINAV), iniciou-se o projeto de desenvolvimento de um veleiro capaz de navegar autonomamente. Navegar à vela significa estar mais dependente das condições ambientais para atingir o objetivo. Todos os fatores influenciam, mas o vento e a ondulação são os mais preponderantes. O vento condiciona a navegabilidade, na medida em que, a sua direção condiciona a eficiência das velas. O escoamento ao passar pela vela

origina duas forças principais: arrasto e impulsão. A conjugação da proa do veleiro e do angulo da vela, em função da velocidade e direção do vento ditam a quantidade de força de impulso e de arrasto que são geradas, ou seja, a força que o navio dispões para ir a vante e a força que adorna o veleiro. A forma e tamanho das velas influenciam igualmente este resultado. A força do vento pode conduzir à cedência dos materiais da vela, originando estragos. A ondulação e vaga afetam a estabilidade e velocidade do veleiro. Dividindo o espectro de frequências e amplitude das ondas nas suas componentes x (vante/ré) e y (Estibordo/Bombordo) [1], é possível tirar as seguintes conclusões: as componentes em y levam o navio a oscilar, podendo atingir adornamentos para além do máximo com estabilidade positiva, com componentes em x, se o sentido de propagação for de vante para ré, conduzem a perda de velocidade, se o sentido de propagação for de ré para vante conduzem a um incremento de velocidade. A intensidade com que estes efeitos são sentidos depende não só da ondulação e vaga, mas também da forma e tipo de casco.

Dado os fatores ambientais não puderem ser controlados, é importante desenvolver a plataforma no sentido de esta ser o mais permissiva possível a fatores externos e passível de atingir o seu propósito em todas as condições, navegar de forma eficiente.

Este artigo encontra-se organizado da seguinte forma: Na secção II encontram-se definidas os prossupostos tidos em conta no projeto do casto, definese a forma do casco e as dimensões principais do mesmo; Na secção III descreve-se o trabalho de desenho realizado e as simulações feitas.

As secções IV, V e VI discutem aspetos relativos ao projeto da quilha, do leme e do patilhão respetivamente. A secção VII contém aspetos relativos aos mastros, velas e ao sistema de controlo. O artigo termina com a secção VIII onde são tiradas conclusões sobre este trabalho e os resultados obtidos.

Definição das especificações e Dimensionamento do Casco

Considerou-se como prioritário a capacidade de resistência do veleiro ao ambiente imprevisível onde opera, mantendo a dimensão do mesmo reduzida, de modo a que possa ser transportado com facilidade. Assim sendo, definiu-se a dimensão fora a fora do veleiro de 2000mm, e iniciou-se a partir daí o *design* do mesmo.

O próximo passo na definição das especificações do veleiro foi escolher qual o deslocamento (Δ) do veleiro. Quanto maior for a massa do veleiro, maior será a sua inércia. Em certas situações, a inércia pode ser benéfica, pois, tendo o veleiro maior dificuldade em alterar o seu estado de movimento, menor é a perda de velocidade causada, p.e, por vagas, contudo, sendo maior a inércia, maior será a dificuldade em o veleiro ganhar velocidade ou efetuar guinadas. Ficou decidido que o deslocamento máximo do veleiro seria 20kg. Este valor é bastante reduzido tendo em conta a dimensão fora a fora do casco, o que condiciona o tipo de casco e os materiais utilizados na sua estrutura. Sabendo a massa total que queremos que o sistema tenha, é possível dimensionar o casco do mesmo, sendo o objetivo a estabilidade e as condicionantes a resistência mecânica e a dimensão fora a fora do veleiro.

A força de impulsão é igual em módulo e de sentido oposto ao peso do fluido deslocado [2]. Sabemos que o peso do fluído deslocado é o produto da sua densidade (ρ), pelo volume deslocado (∇) e pela aceleração da gravidade, sendo que a aceleração da gravidade pode ser simplificada pois entra em ambos os lados da equação [3], resultando a equação 1.

$$\Delta = \rho \nabla \quad (1)$$

Deste modo, é possível chegar ao volume que teremos de deslocar, volume da carena, para atingir a impulsão necessária.

Se sobre o veleiro atuar unicamente a impulsão e o peso sobre a mesma vertical, o veleiro está numa situação de equilíbrio e nada acontece. No mundo real, esta situação é extremamente improvável. Sobre ele atuam inúmeras forças, sendo as principais o vento, a ondulação e a vaga. De acordo com a localização do ponto de aplicação dessas forças, são gerados momentos que adornam o veleiro, sendo necessário existir um momento restituidor. As relações entre os momentos inclinantes e os momentos endireitantes definem o comportamento do casco no mar [4]. Neste sentido, queremos que os momentos inclinantes sejam os mais pequenos possíveis e que os momentos endireitantes sejam o maior possível. Este jogo é possível de ser alcançado através do *design* do casco. O sistema de eixos utilizado é o que está representado na figura 1.

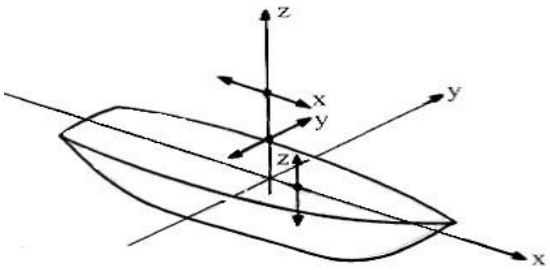


Figura 1- Sistema de eixos utilizado

O objetivo é que pequenos ângulos de adornamento gerem grandes momentos endireitantes. Assim sendo, o metacentro tem de ser o mais elevado possível. A altura metacêntrica transversal (Z_M) de um navio [2] pode ser expressa através da relação expressa na equação 2.

$$Z_M = Z_B + \frac{I_y}{\nabla} \quad (2)$$

Onde Z_B é a altura do centro de flutuação e I_y é o segundo momento de inércia da área da figura de flutuação. Para o navio ser estável a um dado angulo de adornamento, a altura do metacentro tem de ser superior à altura do centro de gravidade (Z_G) [3]. Esta relação é conhecida como a condição de estabilidade e encontra-se representada na equação 3.

$$Z_G < Z_M \quad (3)$$

Assim sendo, quanto mais elevada for a altura metacêntrica e quanto mais reduzida for a altura do centro de gravidade, maior a estabilidade. Uma forma de aumentar a altura metacêntrica é aumentar o segundo momento de inércia da área da figura de flutuação. Esta quantidade pode ser calculada tal como na equação 4.

$$I_y = \int_A y^2 dA \quad (4)$$

Esta relação diz que, quanto maior a dispersão de área no eixo y, maior será a altura metacêntrica, ou seja, quanto maior a boca do veleiro, maior a sua estabilidade. Contudo, tendo já definido o comprimento e o deslocamento do veleiro, é complicado alcançar uma boca grande, porque, para o mesmo deslocamento e comprimento, aumentar a boca significa diminuir o calado. Sendo este casco para um veleiro, o calado tem um papel de relevo, pois, este aumenta a resistência ao arrasto do veleiro por ação do vento.

A solução adotada foi a forma do casco em V [4]. Esta forma garante baixa resistência hidrodinâmica e menos perda de velocidade causada por vagas [4], porém, não garante a

dispersão suficiente de área no eixo y para o veleiro ter estabilidade necessária. Optou-se por manter a forma em V e por adicionar dois cascos unidos rigidamente ao casco principal, obtendo assim o tipo de casco conhecido por Trimarã. Não foram realizados testes comparativos entre esta solução e a solução mono-casco convencional, contudo, esta solução tem a vantagem de garantir a estabilidade do veleiro, mantendo a resistência hidrodinâmica baixa. Se não existir interação entre os escoamentos em volta dos diferentes cascos do trimarã, a resistência total é a soma da resistência de cada casco [5]. Mantendo a resistência de cada casco reduzida, obtém-se uma resistência total pequena.

Desenho e Simulação em CAD 3D

Utilizou-se o programa DELFTship™Free para desenhar o casco principal. Este programa é bastante interativo e versátil, permitindo de forma expedita dimensionar o casco e calcular a resistência hidrodinâmica e parâmetros hidrostáticos do mesmo.

A tabela I sumariza as dimensões do casco principal.

Tabela I - Principais dimensões do casco

Boca máxima	200mm
Calado	130mm
Comprimento total	1900mm
Deslocamento nominal	12kg
Deslocamento máximo	20kg
Coeficiente de bloco	0,1558
Coeficiente Prismático	0,6272

A figura 2 mostra o casco principal desenhado no *software*.

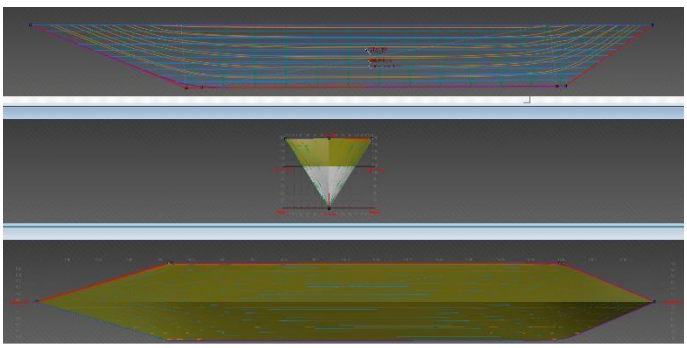


Figura 2 – Casco principal desenhado em DELFTship™Free

Fazendo uso do mesmo *software* calculou-se a resistência do casco com imersão de 130mm, que corresponde à situação nominal de 12kg de deslocamento. O programa utiliza no cálculo da resistência as séries de Delft [6]. O gráfico apenas mostra a

resistência do casco, não contando com a resistência do leme, nem do patilhão e está representado na figura 3.

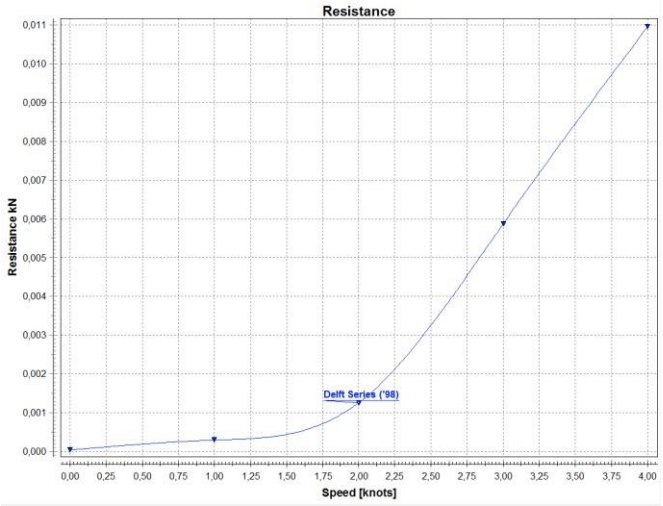


Figura 3 - Resistência em função da velocidade para o casco principal

Vemos que a resistência ao avanço é bastante reduzida, como seria de esperar, tendo em conta a forma de casco adotada.

Por simplicidade, utilizou-se a mesma forma de casco para o casco principal e para os flutuadores laterais. A adição de flutuadores laterais permite aumentar o deslocamento do veleiro acima dos 20kg referidos. Manteve-se este valor como valor máximo de deslocamento. Os flutuadores laterais irão fornecer apenas uma reserva de flutuabilidade. Estando o veleiro sem adornamento, a parte inferior dos cascos encontram-se no limite da linha de água. A função destes flutuadores é apenas a de aumentar a estabilidade do veleiro. À medida que o veleiro adorna, o flutuador lateral do bordo do adornamento entra em imersão, gerando uma força de impulsão, que dado o seu afastamento no eixo y em relação ao centro de gravidade gera um momento inclinante. A localização destes cascos foi estudada de forma à imersão dos mesmos, quando o veleiro esteja adornado, não afete o caimento. Isto é possível através da colocação do centro de flutuação de todos os cascos na mesma posição no eixo x. O resultado do conjunto flutuadores e casco principal encontra-se representada na figura 4.

Figura 2 - Vista por baixo do Veleiro

Tendo já definido todas as dimensões e forma do casco, recorreu-se ao programa CAD 3D *SolidWorks Student Edition* para desenhar todas as peças que constituem o veleiro. Nesta fase do projeto considerou a resistência mecânica do veleiro e a posição do centro de gravidade, por forma a consegui que o veleiro seja robusto e estável.

Projeto da Quilha

O primeiro estágio da construção do casco foi a construção da quilha. O material utilizado foi aço laminado dado as características de resistência mecânica que possui. Este material tem massa volúmica elevada, contudo, reduziu-se a utilização deste ao mínimo para atingir a resistência necessária. Por outro lado, a massa desde componente pode contribuir para a estabilidade do veleiro. Sendo a condição de estabilidade $Z_G < Z_M$, se o material com maior massa volúmica for colocado em cotas mais baixas, o centro de gravidade baixa igualmente a sua cota, contribuindo assim para a estabilidade. O objetivo foi encontrar um equilíbrio entre a massa deste componente por forma a maximizar as características de estabilidade e resistência mecânica, mantendo a massa total do casco o mais reduzido possível. O perfil utilizado foi um perfil "T", com as dimensões representadas na figura 5.

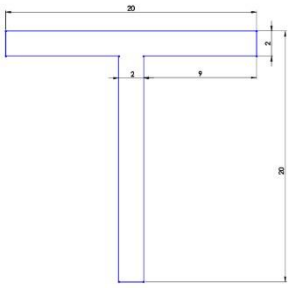


Figura 5 - Perfil Utilizado na Quilha

Este perfil é bastante utilizado em estruturas pela relação que oferece de resistência a cargas versos quantidade de material utilizado. Todos os esforços que o veleiro sofra, serão transmitidos a este componente da estrutura. Foram montados três apoios diretamente na quilha: um apoio por mastro, totalizando dois; e um apoio para o patilhão. Dado que os esforços dinâmicos nestes componentes serão bastante elevados, estes apoios foram igualmente construídos em aço laminado soldados diretamente à quilha.

Os apoios dos mastros são constituídos por uma viga vertical, encastrada na quilha e soldada no topo a outra viga, horizontal, que contem o guia do mastro. O guia do mastro é tubo de diâmetro exterior 25mm e espessura 1,5mm. O guia do mastro está soldado entre a viga horizontal e a quilha, sendo que é cortado a 50mm da quilha. A estrutura foi projetada desta forma por necessidades previstas no controlo do mastro que serão explicadas em secção posterior. A quilha contem dois destes apoios. Está representado na figura 6 a um pormenor da vista de frente da quilha.

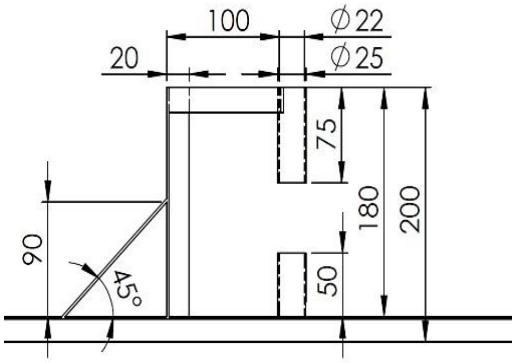


Figura 6 - Desenho de Pormenor do Suporte do Mastro

A quilha contem ainda o apoio para o patilhão. O patilhão mede 1250mm e foi dimensionado para conter um bolbo com 3kg de lastro na sua extremidade. O perfil do patilhão está representado na figura 7.

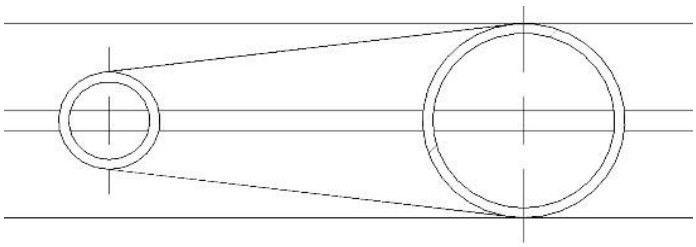


Figura 7 - Encaixe do Patilhão

Por facilidade de transporte do casco, optou-se por segmentar o patilhão em duas partes, sendo então que fica dividido em uma parte de 125mm que se encontra soldada à quilha, e outra parte, de dimensão que completa os 1250mm totais, e que se encastra na primeira

a quando da utilização do casco. A quilha ficou então com o formato que se apresenta na figura 8.

Projeto do Leme

O leme desempenha uma função importantíssima. Para além de permitir ajustar a proa, este componente também diminui o arrasto lateral do veleiro por ação do vento. A forma típica do leme é *foil* (6). Colocando o leme a um determinado angulo, o escoamento fica com diferentes velocidades nas faces do leme, gerando uma força que faz o veleiro guinar, força de sustentação. É gerada ainda uma força de arrasto que reduz a velocidade do veleiro. A escolha do *foil* a utilizar foi feita recorrendo ao programa *Designfoil Demo*. Este programa permite fazer simulações obtendo os coeficientes de sustentação, arrasto 2D e o centro de pressão de um dado perfil, para cada angulo de ataque.

Na figura 9 encontra-se desenhado o perfil do leme.

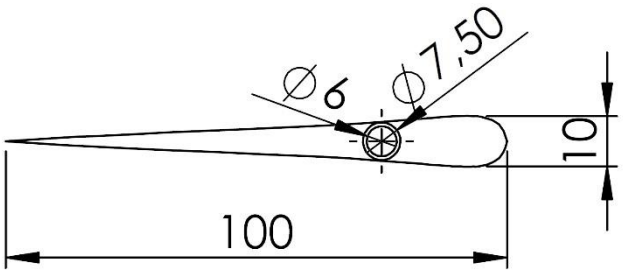


Figura 9 - Planta do Leme

O tamanho do leme pode ser calculado através do critério de Det Norske Veritas [7], expresso na equação 5.

$$A_r \approx \frac{}{100} \left\{1 + 25\left(\frac{B}{}\right)^2\right\} \; d.L^{pp} \qquad Lpp \qquad (5)$$

Na qual: A_r = Área do Leme; L_{pp} = Comprimento entre Perpendiculares; B = Boca e d = Calado

Esta equação prevê o valor mínimo da área de leme. Caso o leme não esteja colocado diretamente atrás do hélice, a área deve ser aumentada no mínimo em 30%. O resultado da equação para o caso do veleiro em estudo, já com o incremento devido, é de 48cm^2. É importante que o leme tenha dimensão ajustada, pois, demasiado pequeno não é suficiente para guinar o navio, demasiado grande gera demasiada força de arrasto, condicionando a velocidade do mesmo.

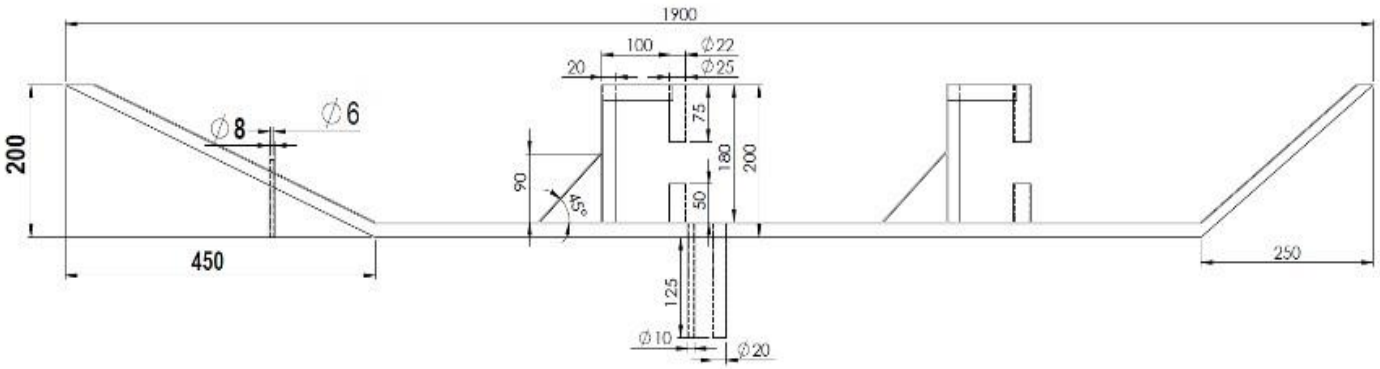


Figura 8 - Desenho da quilha

O leme está em contato permanente com a água, sendo uma zona de risco de entrada de água para o interior do casco através do bucim da madre do leme. Para minimizar este risco, na parte inferior e superior da guia da madre do leme serão colocados retentores de borracha, comprimidos por uma união roscada colocada na parte superior da madre do leme.

Projeto do Patilhão

Pretende-se que o patilhão tenha um alongamento elevado, pois, quanto mais elevado este coeficiente for, menos força de arrasto longitudinal é gerada, mantendo a força de arrasto lateral suficiente para que o veleiro não derive por ação do vento [8]. Por razões de estabilidade também é benéfico que este coeficiente seja elevado, uma vez que, tendo o patilhão um bolbo na sua extremidade, quando maior este for, mais baixo será o centro de gravidade, melhorando a estabilidade. O problema que se coloca é o dimensionamento do mesmo por forma a equilibrar o arrasto longitudinal, transversal e massa deste componente.

O *foil* utilizado neste componente é igual ao *foil* utilizado no leme, contudo, sendo que este componente é mais alongado, prevê-se problemas de resistência dos materiais. Assim sendo utilizou-se na sua construção um perfil de duplo tubo de aço laminado.

Este componente está dividido em duas porções por motivos já referidos. A união das porções é feita por encaixe tipo telescópio. As razões de se ter escolhido este tipo de união é dividir a transição de esforços por uma maior área e impedir a entrada de água para o interior do componente.

O bolbo será do tipo projétil encastrado na extremidade do patilhão. Este componente terá a estrutura em aço laminado apresentada na figura 10, revestida a fibra de vidro.

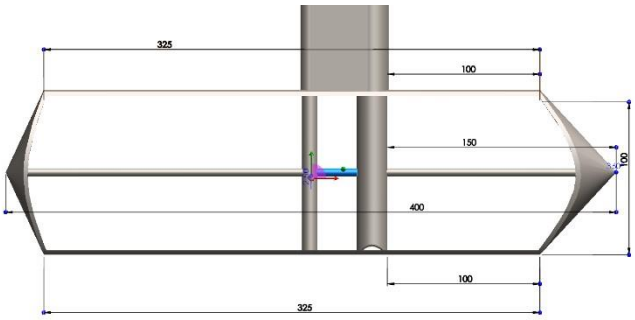


Figura 10 - Bolbo

Dado a massa deste componente permitir o abaixamento do centro de gravidade, no interior do bolbo serão colocadas baterias que alimentarão o veleiro. Seria de esperar a utilização de um material com maior massa volúmica, tal como o chumbo, para lastrar este componente, contudo optou-se pelas baterias, com o objetivo de diminuir a massa total do sistema e diminuir a altura do centro de gravidade.

Mastro, velas e controlo

Pretendeu-se criar uma plataforma passível de melhoramentos. Assim sendo, desenvolveu-se um sistema que permite a fácil troca do sistema mastro e vela. Este sistema é útil pois permite a troca deste conjunto em caso de este se danificar e permite experimentar diferentes tipos de vela, altura de mastro e comprimento da retranca, estando apenas limitado o diâmetro do mastro. O controlo do angulo do mastro é feito através de uma correia dentada ligada entre uma roda dentada encastrada no mastro e um sistema de motor elétrico com caixa redutora. O controlo do angulo é efetuado por um potenciómetro ligado à roda dentada do mastro.

Comando e controlo do veleiro. No centro do controlo do veleiro está um *arduino* mega 2560. O *arduino* dispõe de uma bússola, GPS, anemómetro e comunicações RF 433mhs. Estes sensores transmitem as informações necessárias ao algoritmo de controlo para o veleiro ser capaz de navegar autonomamente. O *input* das coordenadas dos waypoints pode ser efetuado através de hard-core ou então através de mensagem formatada enviada

por RF. O algoritmo esta programado para segui sequencialmente a ordem dos *waypoins* a partir do estado observado através dos sensores. O algoritmo calcula o azimute ao próximo *waypoint* através das coordenadas GPS, e em conjunto com a direção e velocidade vento define a ou as proas a seguir para se aproximar desse *waypoint*. A atitude das velas e o angulo do leme é ajustado igualmente pelo algoritmo à medida que o veleiro navega.

Conclusões

Este trabalho fala sobre o projeto e as considerações tidas em conta no desenvolvimento do novo veleiro pertencente ao CINAV, Escola Naval. Iniciou-se o artigo falando sobre as dimensões e forma que o veleiro deveria de ter. Definiu-se o deslocamento máximo de 20kg e o comprimento fora a fora de 1900mm. A forma do casco foi outro assunto estudado, ficando definido a utilização de um casco trimarã, com forma em V de cada um dos cascos. Em seguida, falou-se do *design* de cada um dos componentes principais do veleiro: quilha, leme, patilhão e bolbo. Foram ainda feitas considerações sobre o sistema de controlo do veleiro.

Conclui-se que a dispersão de área no eixo y da carena seria um especto determinante na estabilidade do veleiro, contudo, por razões de resistência hidrodinâmica, optou-se por flutuadores laterais para garantir a estabilidade do veleiro, ao invés da utilização de um casco com maior boca. Conclui-se ainda a importância de o leme e o patilhão terem um alongamento elevado por forma a manterem a força de arrasto longitudinal baixa, e a transversal suficientemente elevada para que o veleiro possa, guinar e não ser arrastado lateralmente pelo vento. Foi ainda utilizado o critério de Det Norske Veritas para o dimensionamento do leme.

É reconhecida a importância dos veleiros autónomos [9], sendo importante o desenvolvimento de novas abordagens e a procura de novas soluções para o mesmo problema. A crescente generalização e descida do custo de autómatos e sensores permite uma aplicabilidade muito grande em veículos autónomos, contudo, se não for desenvolvida a plataforma, os autómatos e sensores não conseguem atingir o propósito de navegar de forma eficiente e fiável.

Referências

[1] Osgood, Brad. *The Fourier Transform and its Applications*. Stanford: Stanford University.

[2] Benny, Lautrup. *Physics of continuous matter*. Copenhagen: The Niels Bohr Institute, 1998–2010.

[3] Lewis, Edward V. *Principles of Naval Architecture - Volume I. Stability and Strength*. Jersey: The Society of Naval Architects and Marine Engineers, 1988.

[4] Misra, Suresh Chandra. *Design Principles of Ships and Marine Structures*.: CRC Press , 2015.

[5] Mizine, Igor; Karafiath, Gabor; Queutey, Patrick; Visonneau, Michel. "Interference Phenomenon In Design Of Trimaran Ship," , Greece, 2009.

[6] Keuning, J. A.; Sonnenberg, U.B. "Developments in the Velocity Prediction based on the Delft Systematic Yacht Hull Series," Portsmouth, 1132-P, 1998.

[7] Journée, J.M.J.; Pinkster, Jakob. *Introduction In Ship Hydromechanics*.: Delft University of Technology, 2002.

[8] Vacanti, David. "Keel and Rudder Design," *Professiona BoatBuilder*, no. 95, 2005.

[9] Alves, J. C. and Cruz, N.A. "FASt - An autonomous sailing platform for," , 2008.

Apêndice O – Artigo submetido na conferência WRSC16

Barlavento – Considerations about the Design of an Autonomous Sailboat

Pedro Castro Fernandes[10], Mario Monteiro Marques[11], Victor Lobo[12]

Abstract Persistent monitoring of the ocean is important for several reasons, such as to: better understand the climate and ocean dynamics, improve navigation safety, make a better management of their resources and to prevent and combat abuse to the environment. However persistent monitoring can be extremely expensive. One way of decreasing costs is to use unmanned air, surface or underwater vehicles. However, energy autonomy is a major issue for this type of vehicles. For this reason, autonomous sailboats may be a good solution because they can collect renewable energy from the sea and atmosphere, thus being self-sustainable. The objective of this work is to develop and test an autonomous sailboat capable of performing persistent monitoring of the ocean.

The Portuguese Naval Academy has been working on autonomous sailboats since 2010. However, the first autonomous sailboats, that used commercially available hulls, were not resistant enough to bad weather, had little available space for electronics, and were not very efficient. We now decided to design a radically different boat: a very thin and long monohull. We did so using freely available (or very low cost) software, low-cost off-the-shelf components, and simple 3D printers when necessary. This paper describes the creation of the hull using 3D CAD technologies and hydrodynamics simulation.

Introduction

There has been a lot of interest in developing unammed systems persist monitoring of the oceans. The main applications of such systems are environmental monitoring, border and security control (mainly for preventing illegal immigration and smuggling), search and

[10] Pedro Castro Fernandes, Centro de Investigação Naval, Base Naval de Lisboa Alfeite 2810-001 Almada, e-mail: pedro.castro.fernandes@marinha.pt

[11] Mario Monteiro Marques, Centro de Investigação Naval, Base Naval de Lisboa Alfeite 2810-001 Almada, e-mail: mario.monteiro.marques@marinha.pt

[12] Victor Lobo, Centro de Investigação Naval, Base Naval de Lisboa Alfeite 2810-001 Almada, e-mail: vlobo@novaims.unl.pt

rescue, communication relay, etc. One of the main problems of such systems is energetic autonomy. For truly persistent ocean monitoring, the energy must be harvested from the environment. Many different approaches have been made, using solar power (e.g. Scout Boat [1]), wave power (e.g. waveglider [2]), or windpower (e.g. FASt [3], AEOLUS [4], etc). Windpower systems, and conventional sailboats in particular, have proven to be particularly fragile in ocean environments, particularly if they are small, and most have followed conventional designs used in yachts or larger boats.

The Portuguese Navy Research Center (CINAV), after some initial trials with conventional and commercially available model sailboats, started to develop a radically different design for an autonomous oceangoing sailboat. The aim is that sailboat be able to navigate independently of any human intervention. Sail is very dependent on the environment and is influenced by several factors, but the wind and the curl are the most important.

The wind speed and its direction defines the efficiency of the sails. The air flow passing through the boat sails generates two major forces: drag and lift. The conjugation of the boat's head, sail's angle, wind speed and direction, dictate the amount of drag and lift forces generated in each sail [5]. The shape, size, attack angle and number of sails also influence this result.

The curl affects the stability and speed of sailboat. By splitting the sea waves frequency spectrum in its components (see figure 1) x (forward - aft) and y (Starboard - Portside) [6], it is possible to take the following conclusions: i) the components in y lead the ship to oscillate, and it may reach a roll angle beyond the maximum with positive stability; ii) the components in x, if the direction of propagation of the wave is from forward to aft, leads to a decrease of speed, and otherwise leads to an increase of speed. The intensity with which those effects are felt depends not only on the sea, conditions, but also of the dimensions and type of hull [7].

Given the fact that environmental factors cannot be controlled, it is important to develop the sailboat characteristics in other to achieve its purpose even in heavy weather conditions.

This article is organized as follows: In Section II we define the most important factors taken into account in the hull design, and define the shape and dimensions of the hull; in section III we describe the design and hydrodynamic simulations performed; in the

sections IV, V AND VI we discuss aspects related to the design of the keel, rudder and centreboard respectively; section VII contains aspects related to peripheral systems; and finally in section VIII, we draw some conclusions about the results obtained.

Specifications Definition and Sizing

In the design process, the priority was the sailboat's capacity to resist to unpredictable weather conditions, keeping the dimensions as small as possible so that it can be easily transported. We defined the sailboat length overall (LOA) to 2000mm, and started from there.

The next step was to define the displacement (Δ) of the sailboat. The greater the mass, the greater the inertia. This quantifies the difficulty of the sailboat to change its motion status. In certain situations, high inertia can be a benefit. On one hand, the greater the inertia, the lower is the loss of speed caused, for instance, by waves. On the other hand, the greater the inertia, the greater is the difficulty of the sailboat to gain speed or to make turns. Taking this in account, we decided that the maximum displacement of the sailboat would be 20kg. This value is expressively low, taking into account the LOA. The hull type, materials and the stability have to be carefully chosen, in other to satisfy this parameters and still be able to perform the mission. With the LOA and the mass of the system defined, it is possible to start the shape design.

The buoyancy force is equal in module and has opposite direction to the weight of the fluid displaced [8]. We know that the weight of fluid displaced is the product of its density (ρ), by the volume displaced (∇) and by the acceleration of gravity [9]. Having the gravity acceleration in both sides of the equation, the equation (1) simplifies as shown:

$$\Delta = \rho \nabla \quad (1)$$

Using this equation, we can compute the volume of the sailboat underwater.

If the only forces acting on the sailboat were the buoyancy and the weight on the same vertical, the boat would be in balance and no movement would happen. In the real world, this is very unlikely. Many forces act on the sailboat, the main ones being related with wind and ocean waves. Depending on the localization of the application point of those

forces, moments are created that tend to roll the sailboat, being necessary an opposing moment to balance it. The relations between the upsetting moment and the righting moment define the behaviour of the boat at sea [10]. The higher the righting moment produced when the boat is subject to a roll angle, the better, since, this way, the boat will be able to handle bigger upsetting moments. One way to manage this is through the design of the hull shape and size. The axles system used is represented in figure 1.

The goal is that small angles of roll produce large righting moments, meaning, the transverse metacentric height (Z_M) ought to be as high as possible. This can be

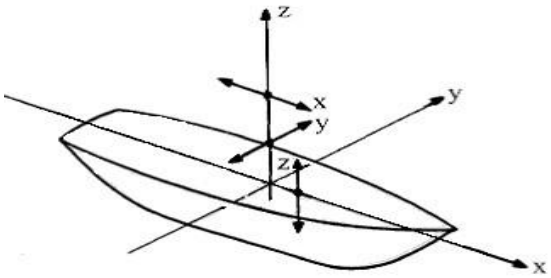


Figure 1- Axles system

expressed through the relation expressed in equation (2) [8].

$$Z_M = Z_B + \frac{I_y}{\nabla} \quad (2)$$

Where ZB is fluctuation centre height and Iy is the second inertia moment of the fluctuation figure area.

For a given roll angle, the sailboat is in a stable condition if Z_M is higher than the gravity centre height (Z_G) [9]. This relationship is known as the stability condition and is represented in the equation 3.

$$Z_G < Z_M \quad (3)$$

Thus, the higher the Z_M and the lower the Z_G, the greater the stability. Z_M depends on the second inertia moment of the fluctuation area I_Y, and can be calculated using the equation 4.

$$I_y = \int_A y^2 dA \quad (4)$$

This relation proves that the more area the sailboat has dispersed on the y axis, the higher will be the metacentric height, meaning it is beneficial to have a large beam. However, the length and the displacement of the sailboat are already set. If we want to maintain these properties and have a larger beam, we have to reduce on the draft. The draft is

important in a sailboat. This type of boat relies on the wind to sail. The wind tends to drag the sailboat with it. A reaction force is required to propel the boat forward, instead of simply being dragged with the wind. This reaction force is generated between the water and the underwater body of the sailboat. The higher this force, the more it will maintain the desired course, resulting in a more precise sailing. In figure 2 we have a comparison between a bigger water drag force resistance and a smaller one. We consider that the force that propels the sailboat forward is the sum of the water drag resistance force and the wind force. We ignore other forces acting on it. In figure 2A we consider a larger drag resistance force then in figure 2B. It is evident that the angle between the forward force and the middle ship line is smaller in figure 2A, resulting in better sailing when compared with figure 2B.

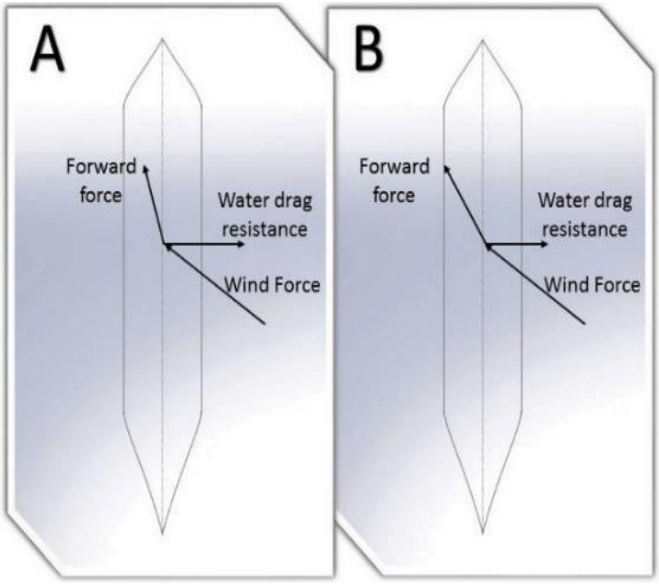


Figure 2 - Comparison between different water drag forces

Thus, we have to dimension the draft and the beam in order to maximize the stability and the lateral water drag resistance force.

The solution adopted was the hull in V shape [10], ensuring low hydrodynamic resistance and less speed loss caused by waves [10]. This form may not guarantee I_Y high enough to have a balanced Z_M on the sailboat. In equation 3 we stated the stability condition. We can achieve positive stability either by higher the Z_M or either by lower the Z_G. As we're not able yet to calculate this value, we design assuming that with ballast management we can ensure a low value for Z_G, resulting in positive stability. To avoid the possibility of

this not be enough and small upsetting moments developing big roll angles, we added two lateral hulls to the main one transforming the sailboat in a Trimaran.

If there is no interaction between the water flow around the different hulls of a trimaran, the total resistance is the sum of the resistance of each hull [11]. Keeping the resistance of each reduced hull, you get a small total resistance. This solution has the advantage of ensuring the sailboat's stability, keeping the hydrodynamic resistance low.

Design and simulation in 3D CAD

We used the program DELFTship™Free to design the main hull. This program is very versatile and user friendly, allowing to design the hull and calculate hydrostatic and hydrodynamics parameters easily.

The dimensions of the main hull are summarized in Table I.

Maximum beam	200mm
Draft	130mm
Length over all	1900mm
Normal displacement	12kg
Maximum displacement	20kg
Cylinder block Coefficient	0,1558
Prismatic Coefficient	0,6272

Table I - hull main dimensions

Figure 3 shows the main hull drawn on this software.

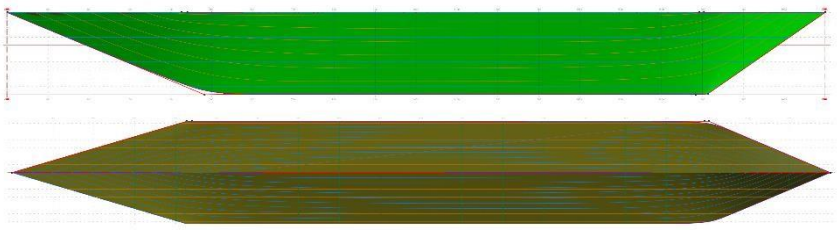


Figure 3 - Main Hull designed in DELFTship™Free

Making use of the same software, the hull drag with draft of 130mm was calculated, which corresponds to the nominal situation of 12kg of displacement. The program makes

use of the resistance series of Delft [12] in this calculation. The graphic only contemplates the main hull drag, not counting with the drag caused by the rudder, hull roughness, keel and other underbody parts. This graphic is represented in figure 4.

We see that the forward drag is low, about 11N at the speed of 4Kts. The real drag is going to be higher than this. Nevertheless, the value still points to an efficient hull.

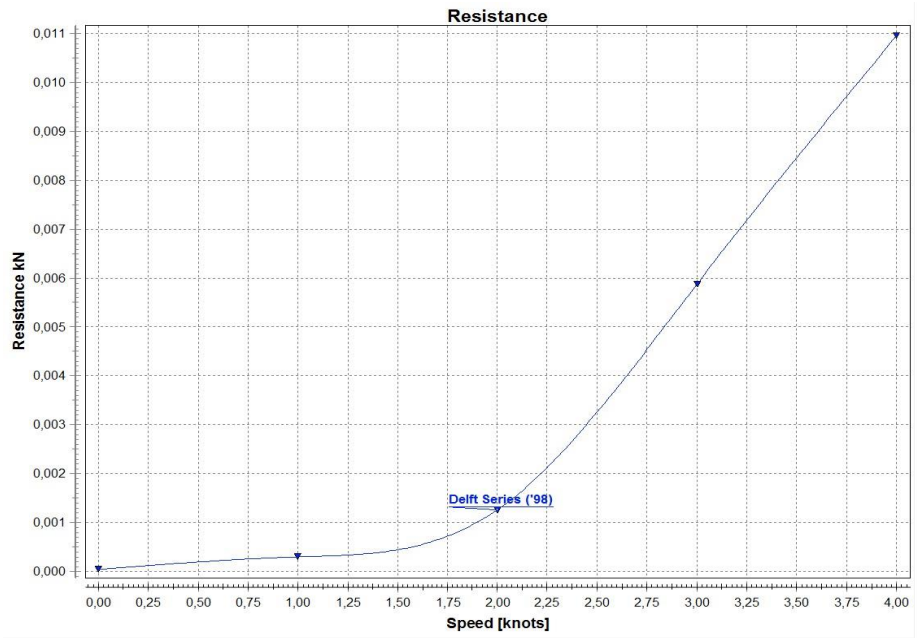


Figure 4 - resistance as a function of speed for the main

For simplicity, we used the same hull shape for the lateral ones. This addition increases the maximum displacement of sailboat over the 20kg's fixed before. The function of the lateral hull is only to increase stability, so when the sailboat in upright the lower part of the lateral hulls is on the water line. The lateral hulls also provide a buoyancy reserve. When the sailboat gets a roll angle, the lateral hull on the adorned side submerge, producing a local buoyancy force. This force generates an upright moment. This can be seen as an increase in I_Y, conducting to a bigger Z_M and better stability.

Having already defined all the dimensions and shape of the hull, we used the CAD program 3D SolidWorks Student Edition to draw all the parts that of the sailboat. In this phase of the project, we stated that the mechanical resistance of the sailboat and the position of the centre of gravity are the major concern in order to achieve a robust and stable sailboat.

The first stage of the hull construction project was the keel design. The material used was rolled steel, since it has good mechanical resistance characteristics. However, this material has high density. Thus, we decided to use the minimum required to achieve a solid and robust part. On the other hand, the Z_G component contributes to the stability of the sailboat. Being $Z_G < Z_M$, the condition of stability, if the heavier materials are placed in lower positions of the sailboat, Z_G will also be lower, contributing to improve stability. It was defined that the keel would be constructed with "T" profile bars, with the dimensions shown in figure 5.

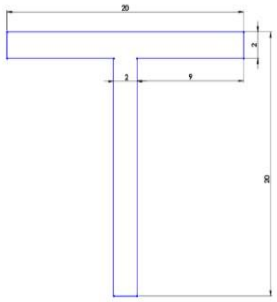


Figure 5 - Profile used in Keel

The "T" profile is widely used in structures for the relationship that it offers between resistances to loads verses quantity of material used. The aim is to transfer all loads applied to the sailboat to this part.

We defined that in the keel there would be three supports: two for the masts and one for the centreboard. Given major dynamic efforts in these components, these parts will also be constructed in rolled steel and directly welded to the keel.

The mast's supports are constituted by a vertical beam, welded in the bottom to the keel. On top of it, there is another beam, welded horizontally. This beam is welded to the mast guide. The mast guide is a tube with 25 mm external diameter, 1.5mm thickness and 75mm long. Concentrically to this tube, and directly welded to the keel, we have got another mast guide constructed with the same material and 50mm long, leaving a gap of 75mm between the tubes. This gap is going to be used

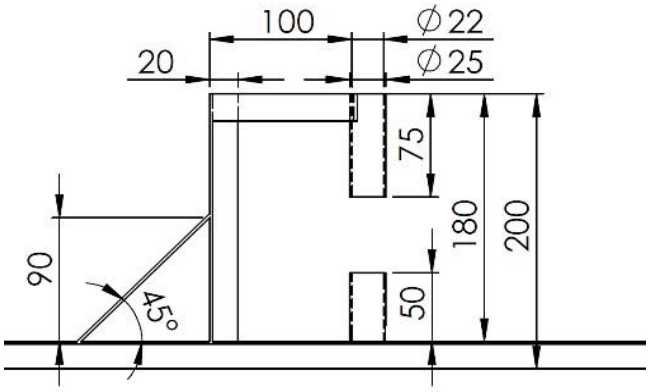


Figure 6 - Exploded View of Mast Support

to control the mast. Figure 6 is the front view of mast support. The keel contains also the support for the centreboard.

Rudder Project

The rudder plays an important role. It allows the sailboat to helm a course and increases the side drag, which is important so sail efficiently. The typical shape of the rudder is a foil.

By seting a certain angle to the rudder, the flow velocity on the two rudder faces will be different, generating a lift and a drag force. The lift force is responsible for generating a turning moment, and the drag force decreases the velocity of the sailboat. The decision of the foil shape was made using the program Designfoil Demo. This program can run simulations on a given foil at a certain angle, providing the lift and drag coefficient, and the pressure centre. The rudder profile is shown in Figure 7.

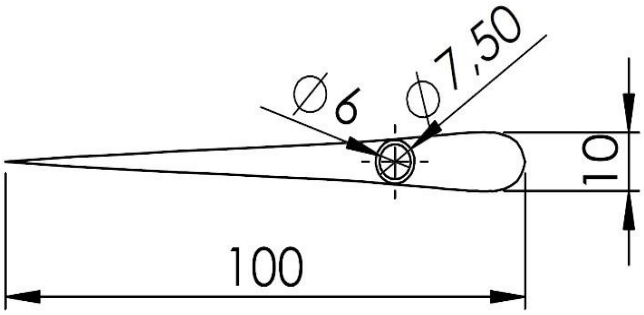


Figure 7 - Rudder Plant

The rudder size can be calculated by the Det Norske Veritas criterion [13], expressed in equation 5.

$$A_r \approx \frac{d \cdot L_{pp}}{100} \left\{1 + 25\left(\frac{B}{L_{pp}}\right)^2\right\} \quad (5)$$

In which: Ar = rudders area; Lpp = Length between perpendiculars; B = beam and d = draft.

This equation provides the minimum rudders area. If the rudder is not placed directly behind the propeller, the area should be increased at least 30%. The equation result for this sailboat is 48cm2. The rudder size should be well-adjusted because, on one hand, if it is too small it won't be able to turn the sailboat; on the other hand, if it is too big, it will generate too much drag force, slowing down the boat.

The rudder is in permanent contact with the water. This could be a problem because it's control would normally involve a hull passage under the waterline allowing water to go inside the sailboat. To minimize this risk, the guide of the rudder inside the sailboat goes higher than the water line, and has rubber retainers on top and bottom of the rudders guide.

Centreboard Project

It is intended that the centreboard has a high aspect ratio, because the higher this coefficient, the less drag is generated, maintaining the lift high enough to balance the lateral force generated by the sails [14]. Without this component, the sailboat would only be able to go in directions down wind. The efficiency of the centreboard is proportional to the aspect ratio. Also, for stability it is beneficial to have a big aspect ratio since need less ballast to achieve a lower Z_G position. The foil used in this component is shown in figure 8.

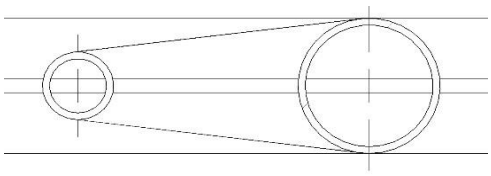


Figure 8 – Centreboard profile.

This component is 1000mm long. It has to resist to the dynamical loads. In the construction of it, we decided to use the same material of the keel. This heavy material can resist to the loads, and it lowers the Z_G. This part measures 1000mm and will have 3kg of ballast on its edge.

In order to simplify the transportation of the sailboat, this component can be divided in two parts, one welded to the keel, 125mm long, and other 875mm long that can be attached to the first part by a screw joint. The keel is represented in figure 10.

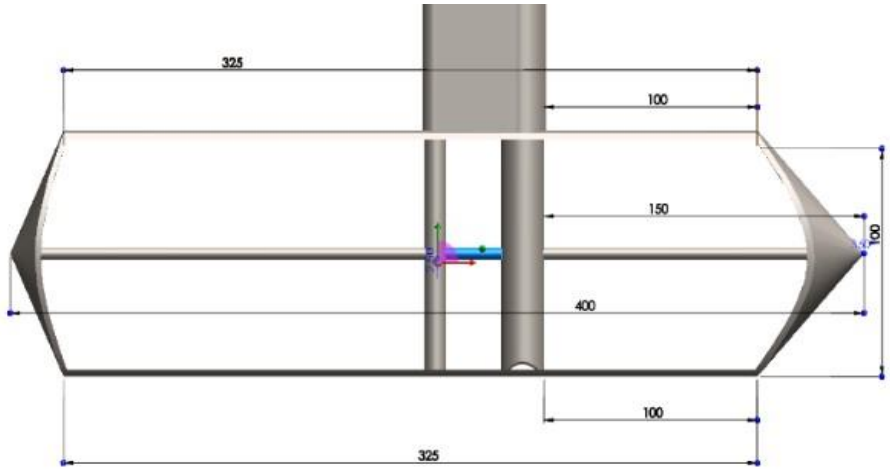


Figure 10 - Bulb

The bulb will be located in the end of the centreboard and will be projectile type, as shown in figure 9.

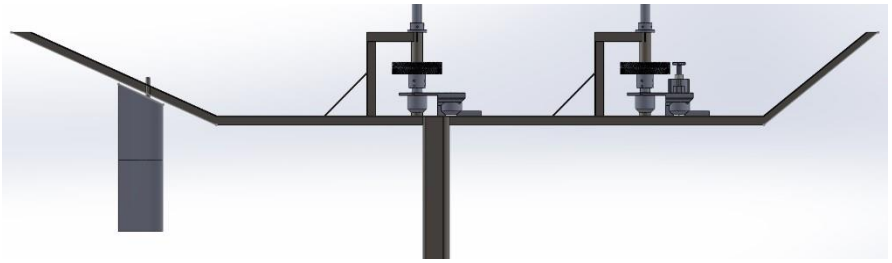


Figure 9 – Keel drawn in Solidworks Stu-

Peripherals Systems

We intended to create an improvement capable platform, and so developed a system that easily allows the exchange of components, such as mast and rudder, in case of material damage or for testing different components.

The control of the mast angles is made using a potentiometer. This potentiometer is linked to the mast by a gear. This system was developed in 3D CAD and printed in PLA. An electrical motor with a planetary gearbox was used. This system has a stall torque of 25Kg/cm. The torque is elevated by a set of gears also printed in PLA with a ratio of 2,14:1. The final result is shown in figure 11.

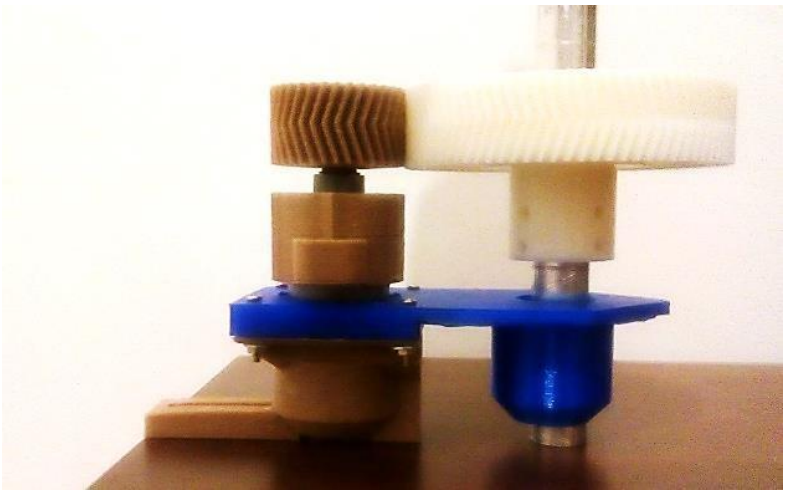


Figure 11Mast control System

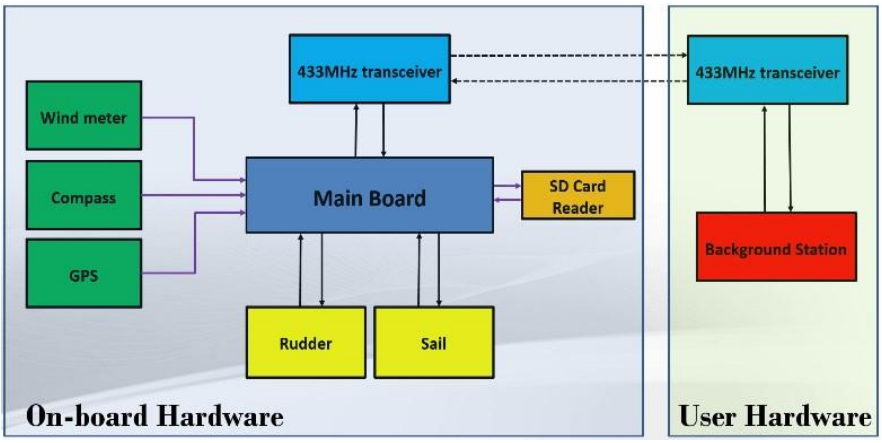


Figure 12System Architecture

We use an Arduino Mega 2560 in the control centre. Attached to it we have got a compass, GPS, anemometer, 100mw RF communications, and other sensors as shown in figure 12.

These sensors transmit the necessary information to the control algorithm. The waypoints coordinates can be saved directly to the SD Card, or can be sent by RF in a formatted message. The algorithm follows the waypoins sequentially, calculating the azimuth the sailboat should go by the GPS coordinates, adjusting the head by the compass readings, and the sails by the anemometer information.

Conclusion

This work is about the construction project of a new sailboat for CINAV. We recognized the importance of autonomous sailboats [3] and the importance to develop new approaches to this problem.

We started this article by talking about some concerns taken into account in the sizing the hull of the sailboat. We defined the LOA, the beam and draft of the sailboat. Then, we discussed the design and simulations done in 3D CAD, in order to achieve an efficient hull shape. After that, we explained the project of the keel, rudder and centreboard. We discussed the peripherals used and finally we end this article with some conclusions.

It is important to develop an efficient underbody shape in order to achieve low drag, but generate enough lateral resistance to avoid the sailboat simply drifting with the wind. The area dispersion in y plays an important role in the sailboat stability. The aim is to have the most area dispersed in this direction, in order for the metacentre to be as high as possible. Another way to increase the sailboat stability is to lower the Z_G.

The draft and the centreboard are very important in sailing. Those components have to generate enough lift to balance the sails lateral force. The rudder not only provides steering to the sailboat, it also contributes to avoid the sailboat to drift with the wind. The rudder can by sized by the Det Norske Veritas criterion. This criterion gives a minimum value to the rudder size, in function of the boat size and characteristics.

We are currently performing sea trials with the sailboat, and the results seem promising.

References

[1] Pete Danko. (2013) "'Scout,' Robotic Solar Boat, On Transatlantic Voyage Thanks To Group Of College Students". [Online]. http://www.huffingtonpost.com/2013/07/10/scout-robotic-solar-boat_n_3575669.html

[2] Justin Manley and Scott Willcox, "The Wave Glider: A persistent platform for ocean science," in OCEANS 2010 IEEE, Sydney, 2010, pp. 1 - 5.

[3] Alves J. C. and Cruz. N.A, "FASt - An autonomous sailing platform for," , 2008.

[4] M. Tranzatto, A. Liniger, M. Colombino, H. Hesse, and S. Grammatico J. Wirz, "AEOLUS, the ETH Autonomous Model Sailboat," Automatic Control Laboratory, Zurich, Switzerland,.

[5] Ryan M. Wilson, The Physics of Sailing. Colorado: JILA and Department of Physics, University of Colorado, Boulder, 2010.

[6] Prof. Brad. Osgood, The Fourier Transform and its Applications. Stanford: Stanford University.

[7] Hal Roth, Handling Storms at Sea.: Adlard Coles, 2009.

[8] Benny. Lautrup, Physics of continuous matter. Copenhagen: The Niels Bohr Institute, 1998–2010.

[9] Edward V. Lewis, Principles of Naval Architecture - Volume I. Stability and Strength. Jersey: The Society of Naval Architects and Marine Engineers, 1988.

[10] Suresh Chandra Misra, Design Principles of Ships and Marine Structures.: CRC Press , 2015.

[11] Igor Mizine, Gabor Karafiath, Patrick Queutey, and Michel. Visonneau, "Interference Phenomenon In Design Of Trimaran Ship," , Greece, 2009.

[12] J. A. Keuning and U.B. Sonnenberg, "Developments in the Velocity Prediction based on the Delft Systematic Yacht Hull Series," Portsmouth, 1132-P, 1998.

[13] J.M.J. Journée and Jakob. Pinkster, Introduction In Ship Hydromechanics.: Delft University of Technology, 2002.

[14] David. Vacanti, "Keel and Rudder Design," Professiona BoatBuilder, no. 95, 2005.

Apêndice P – Estudo sobre Inteligência Artificial.

Inteligência Artificial – O Inicio

O desenvolvimento da eletrónica e da computação na década de 1950, conduziu a especulações se seriam capazes os computadores de desenvolver capacidades cognitivas. Os estudos realizados sobre essas especulações, conduziram à criação desta disciplina, tendo sido formalmente criada em 1956.

Não existe uma definição categórica para inteligência artificial. Diferentes autores defendem diferentes definições para esta disciplina, consoante a visão que têm sobre o tema, embora, se possa diferenciar em duas abordagens: as afetas ao processo e ao raciocino lógico, e as afetas ao comportamento. O sucesso de cada abordagem pode ser medido comparativamente ao desempenho dos seres humanos, ou, relativamente ao conceito ideal de inteligência. Da conjunção das diferentes abordagens com os diferentes modos de medição do sucesso, derivam quatro definições de inteligência artificial:

- Sistemas que agem como humanos;
- Sistemas que pensam como humanos;
- Sistemas que pensam racionalmente;
- Sistemas que agem racionalmente.

Agir como humanos: O teste de Turing

Alan Turing foi precursor na disciplina de Inteligência Artificial. Este génio Inglês escreveu inúmeros trabalhos na área da Inteligência artificial, sendo os mais relevantes: *Intellegent Machinary*, 1950, onde o autor se foca em provar a possibilidade de "máquinas" adquirirem inteligência e *Computing Machinery and Intelligence*, 1950, sendo que neste artigo o autor já propõe um teste para verificar se uma máquina é ou não "inteligente". Este teste ficou conhecido como "o teste de Turing". Ele defendia que um comportamento inteligente por parte de um sistema AI seria o comportamento capaz de atingir desempenho semelhante a humanos em tarefas cognitivas. A partir desta premissa ele formulou o teste, que se caracteriza por um avaliador humano em conversa com dois

interlocutores ausentes, via um terminal. Um dos interlocutores é humano, o outro é o agente de AI. Se o desempenho do agente for bom o suficiente para enganar o avaliador humano, esse sistema é um sistema AI.

Para um Sistema AI passar no teste de Turing tem de ter varias capacidades:

- **Armazenador de conhecimento:** o sistema tem de ser dotado de memória
- **Processador de linguagem e de fala:** por forma a conseguir estabelecer comunicação com o avaliador;
- **Pensamento racional:** por forma a utilizar os dados em memória para formular conclusões.
- **Aprendizagem Máquina:** por forma a adaptar a novas circunstancias e a extrapolar padrões dos dados armazenados.

Estas são as capacidades que um Sistema AI necessita para passar no teste de Turing, contudo, foram desenvolvidas outras versões deste teste, de salientar o "teste total de Turing". Este teste é semelhante ao anterior, mas mais genérico, uma vez que, é efetuada transmissão de vídeo para os interlocutores por forma a testar a perceção visual dos mesmos.

Até à atualidade, ainda nenhum agente passou este teste, contudo, existe referências de agentes dotados de armazenador de conhecimento, processador de linguagem e fala e aprendizagem máquina, faltando apenas o pensamento racional para conseguir passar o teste. Em 2011 foi criado o projeto online *Cleverboot*, que é um sistema que permite o utilizador manter uma conversa com o agente. A arquitetura deste sistema baseia-se no modelo neuronal de conversa. Foi perguntado a este agente *online* se concordava com a

referência desse projeto neste trabalho, sendo que a resposta se encontra na figura que se segue.

Pensar Humanamente: Modelo Cognitivo

Para aferir se um computador pensa de forma semelhante a humanos, é necessário determinar a forma como os humanos pensam. Isso é possível de realizar de suas formas: introspeção e/ou experiencias psicológicas. Após a recolha desta informação é possível efetuar a comparação entre a forma de pensar humana e a forma de pensar do agente de AI. Se os processos racionais entre a entrada e a saída de informação nos humanos e no agente for semelhante, é possível aferir que o agente cumpre o modelo cognitivo. Um exemplo deste modelo é o General Problem Solver (GPS), proposto por Simon, Shaw, e Newell em 1959. Este, que foi o primeiro programa de AI, pretendia, tal como o nome indica, ser uma solução geral para todos os problemas. A arquitetura do programa assentava numa série de operações de memória baseada em regras de logica. O problema era dividido em vários problemas mais pequenos, e, encontrada a solução para cada um desses problemas o problema inicial estava resolvido. A metodologia de teste deste programa passou pela implementação do mesmo num computador e estudo dos resultados obtidos. Neste programa em especifico, os resultados obtidos não foram satisfatórios, uma vez que, para entradas mais complexas, o número de processos até à solução aumentam exponencialmente, tornando o programa obsoleto para a maioria dos problemas do mundo real. Contudo, para problemas mais simples, foram encontradas semelhanças entre o processo de resolução do problema por parte de humanos e do programa.

Pensar Racionalmente: as leis do pensamento

Esta definição de AI assenta numa visão mais teórica, na qual, os problemas são formalizados matematicamente e o método de resolução é dedutivo e racional, baseado em regras logicas tais como as definidas pelo filósofo grego Aristóteles. Porém, formalizar problemas reais ou formalizar o senso comum é tarefa complexa, dado que, muitas variáveis não são têm uma notação precisa, e, algumas relações entre essas variáveis não são explícitas, dificultando o processo de formalização do problema. Isto

contrasta com problemas matemáticos e/ou geométricos, os quais são, normalmente, mais fáceis de definir e de encontrar relações entre variáveis.

Machine Learning – Derivação da Inteligência Artificial

Machine Learning (ML), ou Aprendizagem Maquina surgiu na tentativa de criar máquinas inteligentes, e pode ser definido como a capacidade de um sistema melhorar a sua capacidade de resposta a uma dado problema, ao longo do tempo. Para isto acontecer, é necessário o sistema adquirir novos conhecimentos. Estes conhecimentos são extraídos de grandes quantidades de informação, *data*. Aqui encontramos a diferença maior entre IA e ML. IA prende-se com o comportamento inteligente do agente, definido explicitamente no algoritmo, enquanto ML tenta melhorar o comportamento do agente a partir de experiencia.

A história do ML é quase tao longa quanto a AI, estendendo-se desde a década de 1950 até à atualidade. Existem vários exemplos de agentes deste campo de investigação, sendo os principais:

- Aprendizagem por comparação;
- Aprendizagem por procura;
- Aprendizagem por ajuste de parâmetros;
- Agentes Aprendizes;
- Agentes Adaptativos

Agentes de Procura

Introdução

O primeiro passo na resolução de um problema é a caracterização do mesmo rigorosamente. Existem várias formas de o caracterizar, contudo, o método adotado nesta dissertação, será o exposto no livro de Ernesto Costa – Inteligência Artificial, no qual o autor defende que um problema pode ser definido por um conjunto de configurações ou estados. Um desses estados é o estado inicial. Por forma a se resolver o problema, um conjunto de operadores é requerido, a fim de mutar o estado inicial em estados intermédios e finais. O conjunto de todos os estados é o espaço de estados ou espaço de procura. Este espaço tem de obedecer a um conjunto de restrições implícitas ao problema específico.

O problema que o agente estudado nesta dissertação tenta responder é a forma mais eficiente, entenda-se rápida, de um veleiro de pequenas dimensões se deslocar entre dois pontos. Navegar à vela pressupões a agilização de diversos fatores tais como o vento relativo e o estado do mar com as características orgânicas da própria embarcação. Para caracterizar o problema, é necessário definir quais as variáveis que serão visíveis para o agente, e quais serão desprezadas, quer por simplificação do modelo, quer por impossibilidade de medição das mesmas. É necessário considerar a dimensão do espaço de estados, sendo que este é tanto maior quanto o número de variáveis visíveis pelo agente. Navegar à vela pressupões a gestão do vento relativo por forma a tirar maior vantagem deste. Deste modo o vento relativo é fator imprescindível no espaço de estados. A configuração das velas, ou seja, o seu ângulo , tem de ser igualmente visível para o agente. Por outro lado, a intensidade deste, mesmo sendo de importância inquestionável, pode ser desprezado, pois, é espectável que para o mesmo ângulo de velas, quanto maior a intensidade do vento relativo, maior o impulso. Seria ainda possível gerar quantificadores do estado do mar, através da implementação de variáveis que complementassem o balanço e cabeceio do veleiro, contudo a sua utilização não é imprescindível, pelo que não serão consideradas. Por fim, outra variável que tem de ser visível para o agente é a velocidade. Este objeto será o objetivo de otimização do agente, ou seja, a partir do vento relativo e da atitude das velas, o agente pesquisa dentro de todo o espaço de estados quais os mais viáveis, ou seja, qual a atitude das velas para um determinado vento relativo que produz maior velocidade.

Um algoritmo geral de procura assume a seguinte representação em pseudo-código:

Função: Procura Geral (Problema, Estratégia): Solução ou Falha

8. Inicia a árvore de procura, estado inicial do problema;
9. **Repete**:
   - 9.1. **Se** não existe mais espaço de procura, então:
     - 9.1.1. F Devolve Falha;

     Fim de Se;
   - 9.2. Escolhe outro nó na fronteira para expansão (de acordo com estratégia);
   - 9.3. **Se** nó contêm o **objetivo**, então:
     - 9.3.1. Devolve a solução correspondente;

     Senão:
     - 9.3.2. Expande o nó à árvore de procura e acrescenta os seus sucessores;

     Fim de Se;

10. Fim de Repete;
Fim de Função;

Apêndice Q – Estudos Hidrostáticos

Hydrostatics

Trim: 0,000 (m)

Draft (m)	Lwl (m)	Bwl (m)	Vol. mould (m^3)	Volume (m^3)	Displ. (tonnes)	Displ. (tonnes)	LCB (m)
0,010	1,235	0,012	0,000	0,000	0,000	0,000	1,092
0,020	1,290	0,024	0,000	0,000	0,000	0,000	1,083
0,030	1,334	0,037	0,001	0,001	0,001	0,001	1,079
0,040	1,376	0,050	0,001	0,001	0,001	0,001	1,078
0,050	1,416	0,062	0,002	0,002	0,002	0,002	1,076
0,060	1,456	0,075	0,003	0,003	0,003	0,003	1,075
0,070	1,495	0,087	0,004	0,004	0,004	0,004	1,075
0,080	1,535	0,100	0,005	0,005	0,005	0,005	1,074
0,090	1,574	0,112	0,006	0,006	0,006	0,006	1,072
0,100	1,613	0,125	0,008	0,008	0,008	0,008	1,071
0,110	1,652	0,137	0,009	0,009	0,010	0,010	1,070
0,120	1,691	0,150	0,011	0,011	0,012	0,012	1,068
0,130	1,730	0,162	0,013	0,013	0,014	0,014	1,066
0,140	1,768	0,175	0,016	0,016	0,016	0,016	1,065
0,150	1,807	0,187	0,018	0,018	0,019	0,019	1,062
0,160	1,846	0,200	0,021	0,021	0,022	0,022	1,060
0,170	1,884	0,212	0,024	0,024	0,025	0,025	1,058
0,180	1,923	0,225	0,028	0,028	0,028	0,028	1,056
0,190	1,962	0,237	0,031	0,031	0,032	0,032	1,054

NOTE 1: Draft (and all other vertical heights) is measured above base Z=0,000 NOTE

2: All calculated coefficients based on project length, draft and beam.

Nomenclature	
Draft	*Moulded draft, measured from baseline*
Lwl	*Length on waterline*

Bwl	*Beam on waterline*
Vol. mould	*Moulded volume*
Volume	*Total displaced volume*
Displ.	*Displacement*
Displ.	*Displacement*
LCB	*Longitudinal center of buoyancy, measured from the aft perpendicular at X=0.0*

Design hydrostatics report

Designer			
Created by			
Comment			
Filename	barlavento.fbm		
Design length	2,000 (m)	**Midship location**	1,000 (m)
Length over all	2,001 (m)	**Relative water density**	1,0250
Design beam	0,300 (m)	**Mean shell thickness**	0,0000 (m)
Maximum beam	0,250 (m)	**Appendage coefficient**	1,0000
Design draft	0,170 (m)		

Volume properties		**Waterplane properties**	
Moulded volume	0,024 (m^3)	Length on waterline	1,884 (m)
Total displaced volume	0,024 (m^3)	Beam on waterline	0,212 (m)

Location	**Area**	**Thickness**	**Weight**	**LCG**	**TCG**	**VCG**
	(m^2)	*(m)*	*(tonnes)*	*(m)*	*(m)*	*(m)*
Layer 0	0,769	0,000	0,000	1,043	0,000 (CL)	0,110

Sectional areas

Location	**Area**	**Location**	**Area**	**Location**	**Area**	**Location**	**Area**	**Location**	**Area**
(m)	*(m^2)*	*(m)*	*(m^2)*	*(m)*	*(m^2)*	*(m)*	*(m^2)*	*(m)*	*(m^2)*
0,100	0,000	0,500	0,014	0,900	0,018	1,300	0,018	1,700	0,013
0,200	0,002	0,600	0,016	1,000	0,018	1,400	0,018	1,800	0,006
0,300	0,006	0,700	0,017	1,100	0,018	1,500	0,017	1,900	0,001
0,400	0,011	0,800	0,018	1,200	0,018	1,600	0,016		

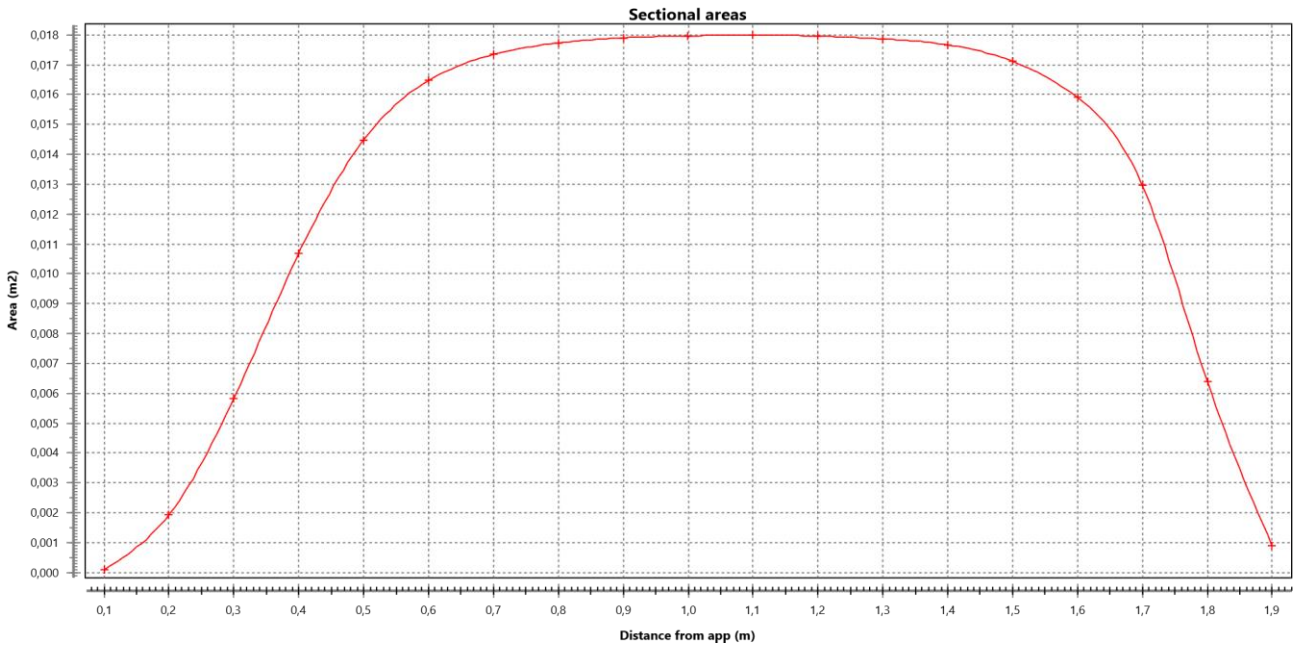
Displacement	0,025 (tonnes)	Entrance angle	22,575 (Degr.)
Block coefficient	0,2386	Waterplane area	0,326 (m^2)
Prismatic coefficient	0,6774	Waterplane coefficient	0,5426
Vert. prismatic coefficient	0,4398	Waterplane center of floatation	1,041 (m)
Wetted surface area	0,623 (m^2)	Transverse moment of inertia	0,001 (m^4)
Longitudinal center of buoyancy	1,058 (m)	Longitudinal moment of inertia	0,069 (m^4)
Longitudinal center of buoyancy	3,093 %		
Vertical center of buoyancy	0,117 (m)		

Midship properties		Initial stability	
Midship section area	0,018 (m^2)	Transverse metacentric height	0,160 (m)
Midship coefficient	0,3522	Longitudinal metacentric height	2,939 (m)

Lateral plane

Lateral area	0,259 (m^2)
Longitudinal center of effort	1,055 (m)
Vertical center of effort	0,092 (m)

The following layer properties are calculated for both sides of the ship



NOTE 1: Draft (and all other vertical heights) is measured above base Z=

NOTE 2: All calculated coefficients based on project length, draft and beam.

Anexo A – *Datasheet* do Anemómetro

Weather Sensor Assembly p/n 80422

Imported by Argent Data Systems

Usage Notes

This kit includes a wind vane, cup anemometer, and tipping bucket rain gauge, with associated mounting hardware. These sensors contain no active electronics, instead using sealed magnetic reed switches and magnets to take measurements. A voltage must be supplied to each instrument to produce an output.

Assembly

The wind sensor arm mounts on top of the two-piece metal mast and supports the wind vane and anemometer. A short cable connects the two wind sensors. Plastic clips on the underside of the arm hold this cable in place. Screws are provided to secure the sensors to the arm.

The rain gauge may be mounted lower on the mast using its own mounting arm and screw, or it may be mounted independently.

Rain Gauge

The rain gauge is a self-emptying tipping bucket type. Each 0.011" (0.2794 mm) of rain causes one momentary contact closure that can be recorded with a digital counter or microcontroller interrupt input. The gauge's switch is connected to the two center conductors of the attached RJ11-terminated cable.

Anemometer

The cup-type anemometer measures wind speed by closing a contact as a magnet moves past a switch. A wind speed of 1.492 MPH (2.4 km/h) causes the switch to close once per second.

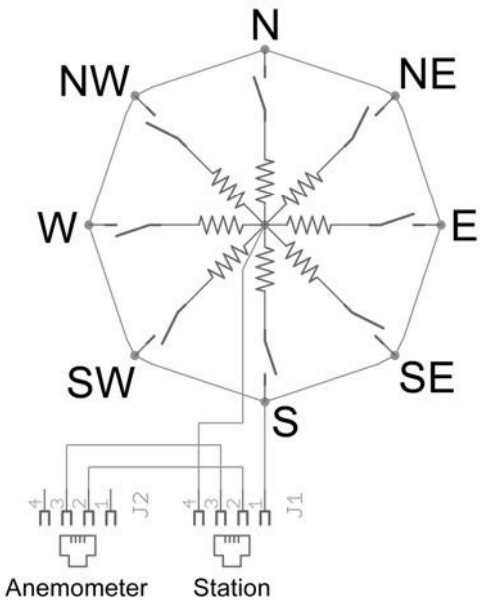
The anemometer switch is connected to the inner two conductors of the RJ11 cable shared by the anemometer and wind vane (pins 2 and

3.)

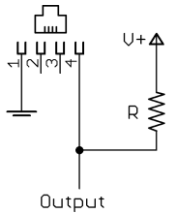
Wind Vane

The wind vane is the most complicated of the three sensors. It has eight switches, each connected to a different resistor. The vane's magnet may close two switches at once, allowing up to 16 different positions to be indicated. An external resistor can be used to form a voltage divider, producing a voltage output that can be measured with an analog to digital converter, as shown below.

The switch and resistor arrangement is shown in the diagram to the right. Resistance values for all 16 possible positions are given in the table.



Resistance values for positions between those shown in the diagram are the result of two adjacent resistors connected in parallel when the vane's magnet activates two switches simultaneously.



Example wind vane interface circuit. Voltage readings for a 5 volt supply and a resistor value of 10k ohms are given in the table.

Direction (Degrees)	Resistance (Ohms)	Voltage (V=5v, R=10k)
0	33k	3.84v
22.5	6.57k	1.98v
45	8.2k	2.25v
67.5	891	0.41v
90	1k	0.45v
112.5	688	0.32v
135	2.2k	0.90v
157.5	1.41k	0.62v
180	3.9k	1.40v
202.5	3.14k	1.19v
225	16k	3.08v
247.5	14.12k	2.93v
270	120k	4.62v
292.5	42.12k	4.04v
315	64.9k	4.78v
337.5	21.88k	3.43v

Anexo B – Caracteristicas da Bateria WP3-12 12Volt 3Ah

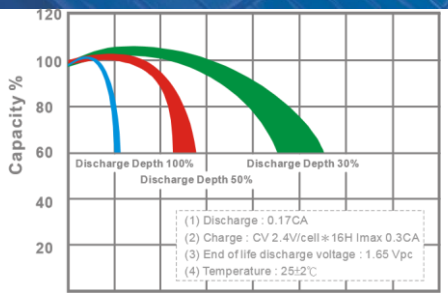
Specifications

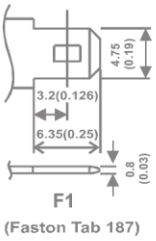
Nominal Voltage(V)				**12V**
Nominal Capacity				
20 hour rate	**(0.15A**	**to**	**10.50V)**	**3Ah**
10 hour rate	**(0.285A**	**to**	**10.50V)**	**2.85Ah**
5 hour rate	**(0.51A**	**to**	**10.20V)**	**2.55Ah**
1 C	**(3A**	**to**	**9.60V)**	**1.6Ah**
3 C	**(9A**	**to**	**9.60V)**	**1.05Ah**
Weight				**Approx. 1.3kg(2.86Lbs.)**
Internal Resistance (at 1KHz)				**Approx. 45 mΩ**
Maximum Discharge Current for 5 seconds:				**45A**
Charging Methods at 25℃(77℉)				
Cycle use:				
Charging Voltage				**14.4 to 15.0V**
Coefficient -5.0mV/℃/cell				
Maximum Charging Current :				**0.9A**
Standby use:				
Float Charging Voltage				**13.5 to 13.8V**
Coefficient -3.0mV/℃/cell				

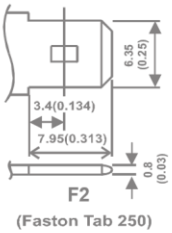
Dimensions mm(inch)

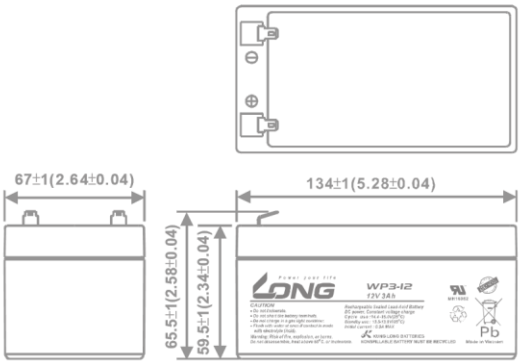
WP3-12 12Volt 3Ah

Operating Temperature Range Charge -15℃(5℉) to 40℃(104℉) Discharge -15℃(5℉) to 50℃(122℉) Storage -15℃(5℉) to 40℃(104℉) Cycle Service Life Trickle (or float) Service Life

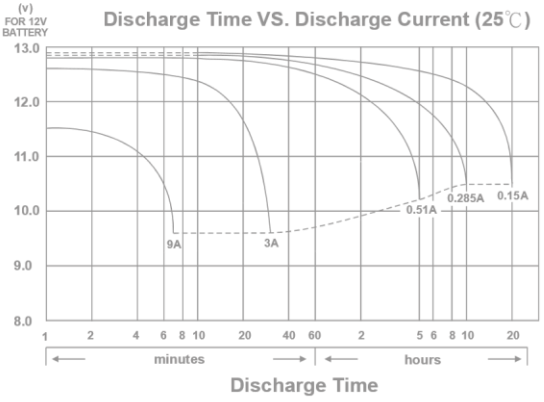


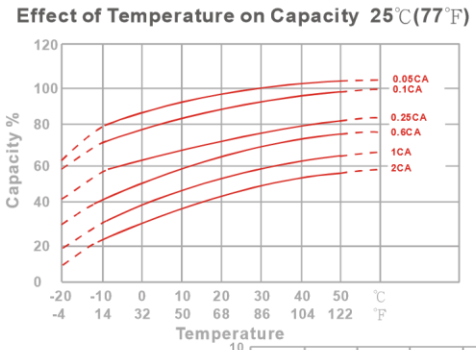


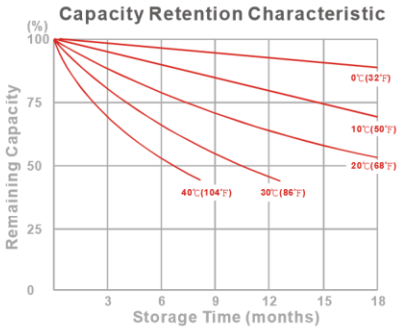


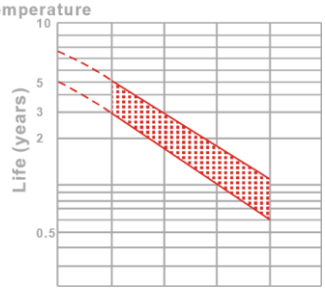


Charge Retention (shelf life) at 20℃ (68℉)	
1 month	92%
3 month	90%
6 month	80%
Case Material	ABS
(Option: UL94 HB & UL94 V-0 flame retardant)	
Terminal	F1 or F2 (Faston Tab 187 or 250)









0 200 400 600 800 1000 1200 1400 1600
Number of Cycles (times)

- PERFORMANCE DATA

Discharge Rates in Watts to Various End Voltages at 25℃(77℉)

Time \ End Voltage		1.85V	1.80V	1.75V	1.70V	1.67V	1.65V	1.60V
5	min	94.2	106	114	121	125	128	132
10	min	68.1	74.7	80.1	84.1	86.2	87.9	90.0
15	min	51.8	57.6	60.5	63.7	65.3	67.0	69.1
30	min	30.2	33.1	34.9	37.1	37.9	38.6	39.2
60	min	17.3	19.2	20.6	21.8	22.4	22.8	23.3
120	min	10.2	11.1	11.9	12.7	13.1	13.5	14.1
180	min	7.62	8.73	9.34	9.76	9.94	10.2	10.6
240	min	5.81	6.53	7.02	7.26	7.39	7.61	7.95
300	min	4.94	5.55	5.97	6.17	6.28	6.41	6.62
600	min	3.28	3.49	3.58	3.65	3.69	3.72	3.77
1200	min	1.79	1.85	1.90	1.92	1.93	1.95	1.98

- Discharge Rates in Amperes to Various End Voltages at 25℃(77℉)

Time \ End Voltage		1.85V	1.80V	1.75V	1.70V	1.67V	1.65V	1.60V
5	min	7.95	9.24	10.3	11.1	11.5	12.0	12.6
10	min	5.87	6.52	7.07	7.46	7.62	7.79	8.08
15	min	4.49	5.03	5.37	5.71	5.86	6.02	6.23
30	min	2.56	2.87	3.03	3.10	3.12	3.14	3.17
60	min	1.54	1.72	1.81	1.88	1.92	1.94	1.99
120	min	0.945	0.993	1.04	1.08	1.10	1.12	1.15
180	min	0.679	0.718	0.752	0.786	0.807	0.826	0.861
240	min	0.556	0.581	0.604	0.625	0.634	0.646	0.663
300	min	0.472	0.493	0.511	0.526	0.533	0.542	0.554
600	min	0.273	0.289	0.301	0.306	0.308	0.311	0.313
1200	min	0.147	0.154	0.158	0.161	0.163	0.165	0.167

All data on the spec. sheet is an average value:

The tolerance range : X<6min(+15%~-15%), 6min≦X<10min(+12%~-12%),10min ≦X < 60min(+8% ~-8%), X≧60min(+5%~-5%)

廣隆光電科技股份有限公司
KUNG LONG BATTERIES INDUSTRIAL CO., LTD.
Head Quarter Service- TEL: +886-49-2254-777 FAX: +886-49-2255-139 E-mail: sales@mail.klb.com.tw
USA Customer Service- TEL: +1-949-328-7936 FAX: +1-949-266-9917 E-mail: sales@kunglong.com
Vietnam Customer Service- TEL: +84-72-3779666 FAX: +84-72-3779476 E-mail: sales@mail.klb.com.tw

Anexo C – Notícia sobre o veleiro no jornal economia do mar

Veleiro da Escola Naval no mundial de veleiros robóticos

Escola Naval aposta na inovação

Começa hoje o Campeonato do Mundo de Veleiros Robóticos (**World Robotic Sailling Competition**, ou WRSC), que decorre ate 10 de Setembro, no estuário do rio Lima. Entre outros participantes, estarão os veleiros autónomos FASt, desenvolvido por estudantes do departamento de Engenharia Electrotécnica e de Computadores (EEC) da Faculdade de Engenharia da Universidade do Porto (FEUP), de que já demos conta neste jornal, e outro, para salvamento, desenvolvido por um investigador da Escola Naval (EN), no âmbito da sua dissertação de mestrado que realizará naquela instituição.
O veleiro para salvamento já foi apresentado em conferências e, de acordo com o responsável pelo projecto, pode ser adaptado "por forma a realizar missões de vigilância na costa portuguesa, ou até mesmo em missões oceanográficas". O protótipo que participará na WRSC já navega de forma eficiente (versão 1.0) e começou a ser desenvolvido em 2015. No total, teve um custo de quase três mil euros, essencialmente em materiais compósitos, electrónica e outros componentes.

Trata-se de um veleiro com 1,9 metros de comprimento, 21kg's de peso, dois mastros com velas rígidas de um metro de altura e um patilhão com 1,25 metros de comprimento. Conforme nos explicou Pedro Castro Fernandes, o seu responsável, "o sistema é controlado por dois Arduínos, programados para desempenharem autonomamente a missão". Todos os componentes "foram projectados e contruídos por nós, contendo 42 peças construídas utilizando uma impressora 3D", acrescenta. Neste momento, a autonomia do veleiro "é de aproximadamente 8 horas em funcionamento continuo, que será futuramente estendida, usando painéis solares, que permitiram funcionamento interrupto", esclareceu-nos.

O mesmo responsável admitiu que entre as vantagens do projecto estão a navegação pelo vento, o que o torna "uma solução muito ecológica", e "a possibilidade de ser utilizado como uma ferramenta de aprendizagem", que os alunos da EN podem usar para diversas finalidades, como arquitectura naval, electrónica ou programação, entre outras. Reconheceu igualmente que reflecte uma solução inovadora, na medida em que tendo um algoritmo de controlo baseado em algoritmos Machine Learning, de que existem várias tentativas de implementação à escala mundial, não concorre directamente com nenhuma solução "que tenha provado ser universalmente melhor que as restantes".

A principal desvantagem reside nas limitações operacionais, "inerentes ao facto de o veleiro estar dependente de condições ambientais, o vento principalmente, para poder navegar", explica. Esclareceu-nos igualmente que até ao momento "não foi identificado um plano de negócio que justifique uma patente", no entanto, reconheceu que o protótipo "tem potencial para ser produzido em larga escala", visando "fazer uma monitorização persistente da nossa actual e futura ZEE". Mas isso "ainda está apenas nos planos", concluiu.

De acordo com Pedro Castro Fernandes, embora desenvolvido na EN, ou seja, associado à Marinha, os progressos científicos que dele resultam "ficam disponíveis para toda a comunidade académica e, caso haja alguma empresa interessada em fabricar e comercializar o protótipo, certamente que será possível encontrar uma solução". Até agora, dois grupos de trabalho da Marinha (um dedicado a veículos autónomos e outro relacionado com o conhecimento situacional marítimo), já demostraram interesse "em utilizar os resultados deste projecto como mais um passo para a vigilância" dos oceanos com veículos autónomos. Um passo que se torna importante com a extensão da plataforma continental portuguesa, cujo processo está em curso nas Nações Unidas.