Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Flavio Marcell Moreira Lemos
Victor Elias de Sousa da Silva

CONTROLE DE UMA EMBARCAÇÃO NÃO TRIPULADA PARA SEGUIMENTO DE UM ALVO

Rio de Janeiro

2019

Flavio Marcell Moreira Lemos
Victor Elias de Sousa da Silva

CONTROLE DE UMA EMBARCAÇÃO NÃO TRIPULADA PARA SEGUIMENTO DE UM ALVO

Projeto de graduação apresentado, como requisito parcial para obtenção do título de Engenheiro Eletricista, à Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. José Paulo Vilela Soares da Cunha

Rio de Janeiro
2019

Ficha elaborada pelo autor através do
Sistema para Geração Automática de Ficha Catalográfica da Rede Sirius - UERJ

L557 Lemos, Flavio Marcell Moreira
Controle de uma embarcação não tripulada para seguimento de um alvo / Flavio Marcell Moreira Lemos, Victor Elias de Sousa da Silva - 2019.
94 f.

Orientador: José Paulo Vilela Soares da Cunha
Projeto Final apresentado à Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia, para obtenção do grau de bacharel em Engenharia Elétrica.

1. Embarcações não tripuladas - Monografias. 2. Controle proporcional-derivativo - Monografias. 3. ZigBee - Monografias. 4. Rastreamento de alvo - Monografias. 5. Sistema de captura de movimentos - Monografias. I. Silva, Victor Elias de Sousa da. II. Cunha, José Paulo Vilela Soares da. III. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. IV. Título.

CDU 621.3

Flavio Marcell Moreira Lemos
Victor Elias de Sousa da Silva

CONTROLE DE UMA EMBARCAÇÃO NÃO TRIPULADA PARA SEGUIMENTO DE UM ALVO

Projeto de graduação apresentado, como requisito parcial para obtenção do título de Engenheiro Eletricista, à Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Aprovado em 09 de dezembro de 2019.

Banca Examinadora:

Prof. Dr. José Paulo Vilela Soares da Cunha (Orientador)
Faculdade de Engenharia - UERJ

Prof. Dr. Paulo Bulkool Batalheiro
Faculdade de Engenharia - UERJ

Prof. Dr. Téo Cerqueira Revoredo
Faculdade de Engenharia – UERJ

Rio de Janeiro
2019

AGRADECIMENTOS

Flavio Marcell Moreira Lemos

A minha mãe, Maria José, por me trazer a este plano e com todo carinho, amor incondicional e paciência ser a responsável pelo que sou hoje e serei no futuro.

A minha co-mãe, Ana Lucia, pelo igualmente amor, carinho, apoio e perseverança em lutar contra minha impontualidade. Daqui a pouco tô saindo, Tia!

A minha um dia namorada, hoje esposa e companheira Caroline por caminhar todo o caminho ao meu lado sem duvidar em nenhum instante que nossos sonhos serão realizados. Por mostrar que sempre haverá um jeito.

A Luciana pelo apoio e palavra de força no momento mais descrente e mostrar que a distância não quebrará os laços que a vida amarra. *LuvU!*

A meu pai, Gilberto, e meu padrinho, Donato, que mesmo em diferenças transmitiram suas experiências, carinho e conselhos para enfrentar os obstáculos que se colocarão a minha frente. Queria poder ter ouvido mais...

A Vanguarda pelo companheirismo, alegria e diversão nos momentos que estamos reunidos ou molhando as palavras! Abaixa a cabeça que o copo vai voar...

Aos amigos de faculdade por ajudar a tornar todo esse tempo leve, fazendo da universidade uma segunda casa.

A Leonardo e Carolina pela irmandade e compartilhar o sangue rubro-negro.

Aos meus líderes Alexandre Bezerra, Ericson Bittencourt, Togo Ribeiro e suas respectivas equipes pelo grande aprendizado, confiança, ensinamentos e por permitir realizar meu curso de graduação com dedicação graças a horários flexíveis.

Ao meu amigo e parceiro de projeto, Victor Elias, pela GRANDE paciência, força, amizade, alegria, tensão, e outros muitos substantivos ao qual não sei o que utilizar para descrever essa fase final de faculdade. *Yes, we can*!

Aos meus professores da universidade pelas horas dedicadas a excelência da formação, mesmo em tempo difíceis, deste que vos agradece humildemente.

A Malu e Vitrola pelas entusiasmadas recepções de todos os dias.

A todos meus amigos, famílias e agregados, a Rapeizy e ao Flamengo.

A quem duvidou ou torceu contra, nem discuto por que estou em outro patamar!

AGRADECIMENTOS

Victor Elias de Sousa da Silva

À Oxalá, Ogum, Iemanjá e a todos os meus guias por me permitirem ter a paciência e perseverança de continuar sempre, mesmo nos momentos mais difíceis.

À minha mãe Márcia por todo esforço empenhado na minha formação, pelos conselhos e apoio incondicional nos momentos que pensei em desistir.

Ao meu irmão Thiago por estar presente em todos os momentos, dividindo experiências e culpas por toda a vida, mesmo agora um pouco mais distante.

À minha avó Sueli por todo amor e carinho, pela paciência com meu mau humor e preguiça.

À minha namorada Leticia pelo companheirismo, amor e toda ajuda nesse projeto e durante a minha graduação, obrigado por estar sempre comigo, por todos os momentos bons e não tão bons, pelos conselhos e por me aturar quando estou insuportável. Obrigado por tornar tudo mais fácil.

Aos meus amigos Fernando, Flavio, Iuri, Jorge, Marco e Robson pelos momentos de descontração e diversão para tornar a vida mais fácil.

Aos meus amigos de final de faculdade Daniel, Lucas e Yago por tornarem as matérias de final de curso mais divertidas e mais fáceis de aturar, Vermelho Líder sempre!!

Ao meu orientador, José Paulo, por todo conhecimento passado, pela paciência e ajuda durante a elaboração desse projeto.

Ao Anderson Dursulina Marques pela elaboração do desenho da placa de circuito impresso desenvolvido nesse projeto.

Aos meus chefes Carlos Pinheiro e Paulo Vaz que permitiram que eu pudesse realizar com muita dedicação meu curso de graduação muitas vezes nos horários de trabalho, pelo suporte e confiança durante todos esses anos.

E a todos que me acompanharam e me ajudaram durante o curso de graduação.

Nada no mundo consegue tomar o lugar da persistência. O talento não consegue; nada é mais comum que homens fracassados com talento. A genialidade não consegue; gênios não recompensados é quase um provérbio. A educação não consegue; o mundo é cheio de errantes educados. A persistência e determinação sozinhas são onipotentes.

Calvin Coolidge

RESUMO

Neste projeto foi desenvolvido e testado experimentalmente um sistema autônomo, com o objetivo de fazer com que uma embarcação não tripulada siga um alvo, recebendo apenas informações sobre o seu posicionamento atual e o alvo desejado. Para este trabalho foi utilizado um barco dotado de um microcontrolador Arduino, que recebe as medições das posições necessárias para realizar o controle de um sistema visual de captura de movimentos de alta precisão. De forma a possibilitar a resolução do problema, desenvolveu-se a modelagem da dinâmica do barco. Foram utilizados dois controladores PD (proporcional derivativo), um para controlar o rumo a fim de direcionar a embarcação ao alvo e outro para controlar a distância entre a embarcação e alvo. A comunicação utilizada entre os elementos do sistema é realizada por uma rede *ZigBee*. Uma placa de circuito eletrônico foi construída de forma a facilitar a integração do microcontrolador, da rede *ZigBee* e a embarcação. Para a validação do sistema desenvolvido, são apresentados alguns resultados experimentais obtidos em um ambiente controlado constituído de uma piscina e do sistema de captura de movimentos.

Palavras chave: Embarcações não tripuladas; Controle proporcional-derivativo; *ZigBee*; Rastreamento de alvo; Sistema de captura de movimentos.

ABSTRACT

In this project an autonomous system was developed and experimentally tested, with the aim of making an unmanned vessel follow a target, receiving only information about its current positioning and the desired target. A boat equipped with an Arduino microcontroller was used to receive the required position measurements from a high precision motion capture system. In order to solve the problem, the dynamics modeling of the boat was developed. Two PD (proportional derivative) controllers were used, one to control the leading to align the vessel with the target and the other to control the distance between the vessel and the target. The communication used between the system elements is made by a *ZigBee* network. An electronic circuit board has been created to facilitate the integration of the microcontroller, the ZigBee network and the vessel. To validation the developed system, some experimental results obtained in a controlled environment consisting of a swimming pool and motion capture system are presented.

.

Keywords: Unmanned vessels; Proportional-derivative control; *ZigBee*; Target tracking; Motion Capture System.

LISTA DE FIGURAS

Figura 1 - Carro autônomo. Extraído de (Guerra, 2017). 17
Figura 2 - Frota idealizada de barcos. Extraído de (Lorenzetti, 2016). 17
Figura 3 - Diagrama geral do sistema proposto. 18
Figura 4 - Diagrama interno da embarcação utilizado. 19
Figura 5 – Sistemas de coordenadas. 23
Figura 6 – Dimensões do barco. 24
Figura 7 – Distâncias entre ponto desejado e ponto atual. 25
Figura 8 – Forças geradas pelos propulsores. 26
Figura 9 – Camadas de uma rede *ZigBee*. Extraído de (Ramos, 2012, p.48). 30
Figura 10 – Topologias de rede *ZigBee*: a) Topologia estrela. b) Topologia em malha. c) Topologia em árvore. Extraído de (Tennina, 2013, p.5). 33
Figura 11 – Tela inicial do software XCTU. 39
Figura 12 – Tela para adicionar dispositivos. 40
Figura 13 – Tela com parâmetros. 41
Figura 14 – Diagrama do sistema de controle. 48
Figura 15 – Zona morta. 55
Figura 16 – Resposta dos controladores com o compensador. 56
Figura 17 - Câmera de precisão do sistema VICON. 57
Figura 18 – Ambiente dos experimentos. 58
Figura 19 – Testes com os LED's. 59
Figura 20 – Montagem final da embarcação. 60
Figura 21 – Ângulo de rumo do barco no experimento 1. 62
Figura 22 – Distância entre o barco e o alvo no experimento 1. 63
Figura 23 – Trajetória do barco no experimento 1. 64
Figura 24 – Ângulo de rumo do barco no experimento 2. 65
Figura 25 – Distância entre o barco e o alvo no experimento 2. 66
Figura 26 – Trajetória do barco no experimento 2. 67
Figura 27 – Ângulo de rumo do barco no experimento 3. 68
Figura 28 – Distância entre o barco e o alvo no experimento 3. 69
Figura 29 – Trajetória do barco no experimento 3. 70
Figura 30 – Projeto da placa. 90

Figura 31 – Vista inferior da placa da embarcação.....91
Figura 32 – Vista superior da placa da embarcação.....91
Figura 33 – Alterações realizadas.....92
Figura 34 – Alterações implementadas na placa. 93
Figura 35 – Componentes. a) Xbee. b) Garadino. c) L298N.....93
Figura 36 – Montagem final da placa.....94

LISTA DE TABELAS

Tabela 1 – Dimensões da embarcação. 24
Tabela 2 – Principais características das topologias de rede *ZigBee*, adaptado de (Tennina, 2013, p.6).....34
Tabela 3 – Características do módulo XBee-Pro S2, adaptado de Sousa (2016, p.66).....37
Tabela 4 – Configuração Coordenador.....41
Tabela 5 – Configuração Dispositivo Final.....42
Tabela 6 - Montagem do quadro *ZigBee.* 42
Tabela 7 – Alteração da frequência dos PWMs.....49
Tabela 8 – Controlado PD implementado. 50
Tabela 9 – Limitador da saída do controlador PD.....51
Tabela 10 – Leitura do quadro *ZigBee.*.....52
Tabela 11 – Habilitador do controlador de distância. 53
Tabela 12 – Acionamento completo dos motores. 54
Tabela 13 – Ganhos/Parâmetros dos controladores. 61
Tabela 14 - Programa do Arduino completo. 76
Tabela 15 – Programa Principal do Matlab. 83
Tabela 16 – Programa da comunicação *ZigBee* no Matlab. 87

LISTA DE SIGLAS

API	*Application Programming Interface*
ISA	International Society of Automation
LED	Diodo emissor de luz
P	Proporcional
PAN	*Personal Area Network*
PD	Proporcional-derivativo
PID	Proporcional-derivativo-integral
UART	*Universal Asynchronus Receiver/Transmitter*

SUMÁRIO

1. Introdução 16

1.1. Objetivo 18

1.2. Descrição do sistema 18

1.3. Organização do texto 20

2. Modelagem da Dinâmica do Barco 21

2.1. Sistema de coordenadas 22

2.2. Equações dinâmicas do sistema 24

3. Comunicação sem fio – *ZigBee* 29

3.1. Padrão *ZigBee* 29

3.1.1. IEEE 802.15.4 30

3.1.2. Camadas 31

3.1.2.1. Função dos dispositivos 32

3.1.3. Topologia de rede *ZigBee* 32

3.1.4. Formação, endereçamento e transmissão de uma rede *ZigBee* 34

3.1.4.1 Enderaçamento 35

3.1.4.2 Transmissão 35

3.1.5. Modos de operação 36

3.1.6. Módulos XBee 36

3.1.6.1. Modos de comunicação dos módulos XBee 37

3.2. Motivação e utilização do padrão *ZigBee* 38

3.3. Configuração da rede e dos dispositivos 39

3.4. Extensão da rede para uma frota de barcos 43

4. Controle 44

4.1. Descrição do controle 45

4.2. Controlador PD 45

4.3. Implementação do controle no microcontrolador Arduino 49
5. Resultados experimentais 57
5.1. Experimento 1 61
5.2. Experimento 2 64
5.3. Experimento 3 67
6. Conclusões 71
Referências 73
Apêndice A – Código Arduino 76
Apêndice B – Códigos Matlab 83
Apêndice C – Montagem eletrônica 89

1. Introdução

É perceptível que o interesse em veículos não tripulados aumenta em uma velocidade proporcional à tendência mundial de automatização, conforme explicita Rosário (2017, p. 27). Nesse sentido, um grande exemplo desses veículos são os carros autônomos (Figura 1), os quais podem utilizar as faixas ou calçadas como referência para manter o carro no trajeto pré-definido.

Outro exemplo, é a utilização de veículos terrestres não tripulados controlados a distância por câmeras, para utilização para finalidades militares conforme citado por Wolf (2015, p. 1).

No caso de embarcações podemos verificar esse interesse conforme descrito no texto de Rosário (2016, p.1), em que o autor menciona que existe um aumento no desenvolvimento de pesquisas relacionadas a embarcações de superfícies não tripuladas, com sistemas de posicionamento mais precisos e com menor custo, além da contribuição acrescida pela evolução das tecnologias de redes sem fio.

No entanto, embarcações não possuem esses tipos de objetos para facilitar o seu posicionamento, uma vez que, na maioria dos seus trajetos, como no mar ou em um rio largo, é possível que não haja nenhum ponto de referência próximo. Nestes casos, é utilizado um Sistema de Posicionamento Dinâmico, que é definido por Amaral (2008, p.13) como “{...} um sistema que controla automaticamente a posição e a orientação de uma embarcação por meio de propulsão ativa”. Este último pode ser encontrado em plataformas de exploração e perfuração, pesquisa geofísica, suporte a veículos submarinos, lançamento de dutos e cabos, bem como em lançamento de foguetes em alto-mar.

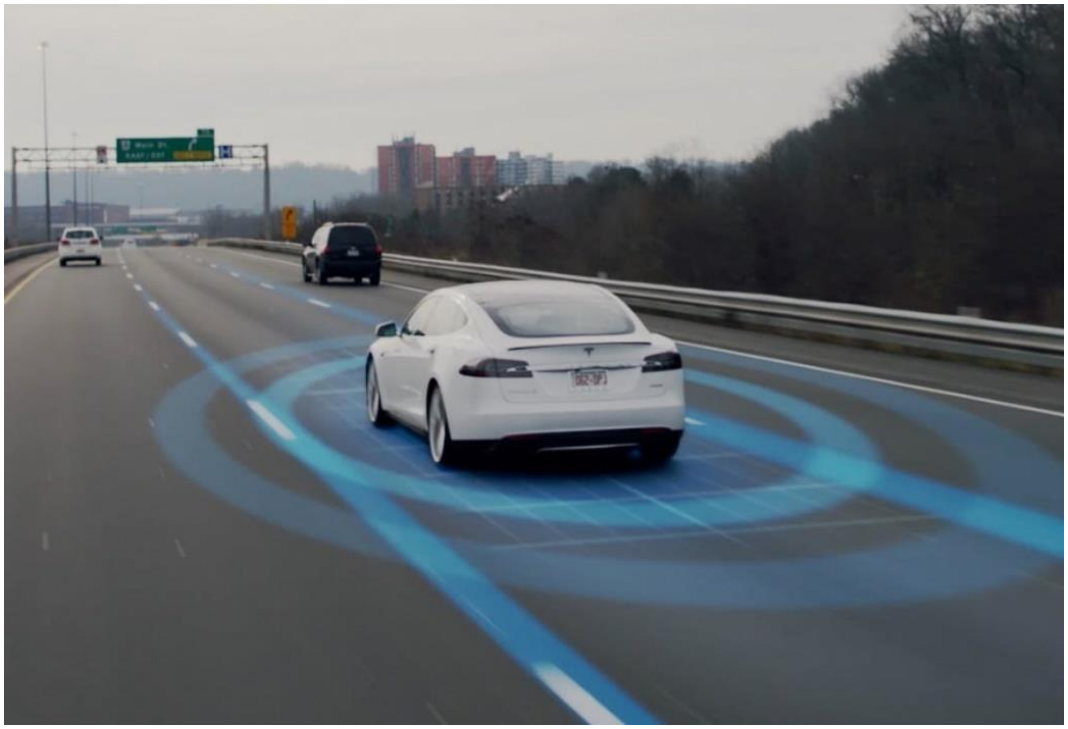


Figura 1 - Carro autônomo. Extraído de (Guerra, 2017).

Inclusive, uma aplicação direta de veículos autocontrolados seria para o deslocamento de grandes distâncias ou para transporte de cargas, sendo muito interessante a possibilidade de que vários veículos possam seguir a mesma rota controlando suas distâncias, mas mantendo uma proximidade entre suas rotas, o que, por conseguinte, otimizaria as trajetórias (Figura 2).

Figura 2 - Frota idealizada de barcos. Extraído de (Lorenzetti, 2016).

1.1. Objetivo

Os objetivos deste Projeto de Graduação são: (i) desenvolver algoritmos de controle integrados à embarcação e (ii) controlar o barco de forma que siga um outro objeto utilizando informações enviadas do sistema de medição de posições.

1.2. Descrição do sistema

O sistema é composto pela embarcação, onde está inserido o controlador Arduino, o microcomputador, que irá transmitir as posições adquiridas pelo sistema de monitoramento visual e a trajetória pré-definida; e o sistema de monitoramento visual, que será utilizado para o posicionamento do barco. A comunicação que ocorrerá entre os três elementos supracitados será feita conforme a Figura 3.

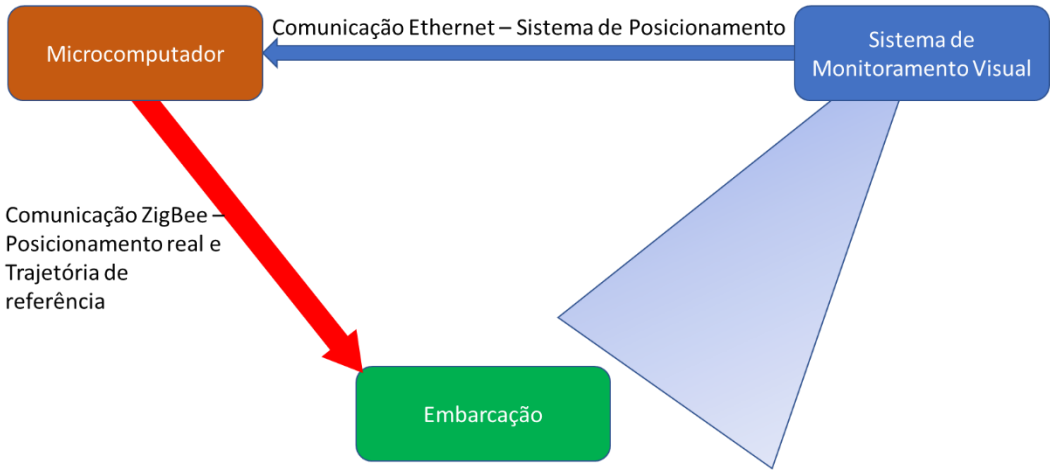


Figura 3 - Diagrama geral do sistema proposto.

Conforme demonstrado na Figura 4, o sistema é baseado na implementação de um controle diretamente no Arduino responsável pela atuação dos propulsores. Desta forma, o controlador mencionado terá uma comunicação sem fio com um microcomputador, onde se encontram o sistema de posicionamento e um percurso pré-definido como referência de trajetória.

Para a comunicação entre a embarcação e o microcomputador, será utilizada uma rede *ZigBee* (ZIGBEE, 2006), por ser uma rede de baixo custo de implementação e existir interface pré-definida com o Arduino, também apresentando a possibilidade de arquitetura de rede auto ajustável, para que seja possível a comunicação com um dispositivo da rede através de qualquer outro elemento desta.

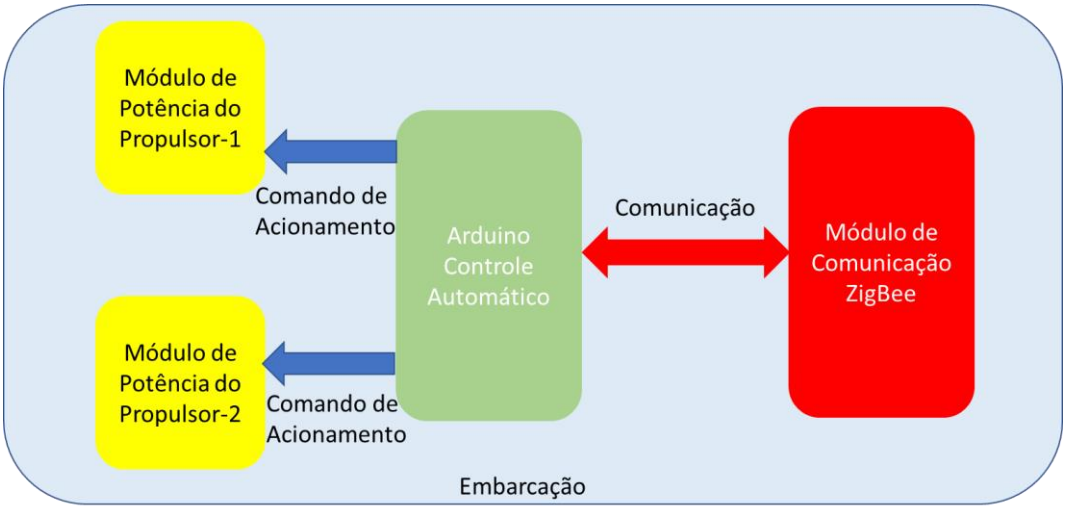


Figura 4 - Diagrama interno da embarcação utilizado.

Por conseguinte, a partir do diagrama, pode-se perceber que o propósito do sistema descrito é controlar os propulsores do barco por meio das saídas do Arduino que irão atuar nos módulos de potência de acionamento dos motores.

O Sistema de Posicionamento Dinâmico foi feito com a utilização de um sistema visual de câmeras, o qual é capaz de gerar e processar o posicionamento do veículo, sendo possível utilizá-lo em algoritmos de controle de alto desempenho (VICON, 2006). Esta informação é enviada para o microcomputador, através da rede Ethernet entre o switch do sistema VICON e o microcomputador.

Após o protótipo receber as informações sobre sua posição e a trajetória desejada, o microcontrolador utilizará estas para corrigir seu movimento, a fim de se aproximar com maior perfeição da rota prevista. Esta comunicação será feita através dos módulos *ZigBee*, que serão acoplados na porta de comunicação paralela do Arduino e no microcomputador, possibilitando a recepção e transmissão destes dados.

A utilização do Arduino em conjunto com a rede *ZigBee* é utilizada em várias iniciativas, por ser uma rede de baixo consumo de energia, e com várias possibilidades de configuração, temas a serem aprofundados no Capítulo 3. Um exemplo de aplicação utilizando o conjunto Arduino/ *ZigBee* é o robô móvel para coleta de variáveis em ambientes não-estruturados de (Neves, 2016).

1.3. Organização do texto

O texto foi organizado como segue:

I. O Capítulo 2 apresenta a modelagem da dinâmica do barco e todos os elementos necessários para isto.
II. O Capítulo 3 descreve a comunicação sem fio, bem como o protocolo de rede *ZigBee*, suas configurações e dispositivos.
III. O Capítulo 4 descreve as características de controle assumidas para solução do problema proposto e a sua implementação na programação do Arduino.
IV. O Capítulo 5 apresenta os resultados experimentais obtidos.
V. O Capítulo 6 apresenta as conclusões e contribuições que o projeto pode oferecer e propostas para trabalhos futuros.

2. Modelagem da Dinâmica do Barco

Neste Capítulo serão abordados a modelagem da dinâmica do barco e todos os elementos necessários para que seja possível esta descrição do sistema de posicionamento da embarcação, bem como o sistema de coordenadas utilizado e as equações que descrevem o movimento do veículo.

A modelagem da dinâmica do barco consistirá em definir algumas premissas, o equacionamento e a definição do tipo de controle que será exercido sobre o veículo, conforme é possível verificar em (Amaral, 2008, p.38), a modelagem de um sistema consiste em definir um conjunto de equações, desenvolvidas por meio de modelos matemáticos, que sejam capazes de descrever o comportamento físico do sistema.

Assim, esta etapa do projeto versará sobre a construção das equações as quais possibilitam a descrição do modelo matemático que descreve o comportamento da embarcação. Por isso, em um primeiro momento, serão desconsideradas as outras partes do sistema, como a rede *ZigBee* e a programação do Arduino.

A complexidade da modelagem utilizada para descrever os fenômenos de interesse é diretamente ligada à precisão desejada no final do controle ou sistema. Nesse sentido, conforme explicitado por Ogata (2000, p.48), “é possível melhorar a precisão de um modelo matemático aumentando sua complexidade”, porquanto é relevante manter o compromisso entre esta complexidade e os resultados que são desejados a partir do experimento.

Por isso, o mencionado autor evidência que para obter um modelo matemático, é importante estabelecer uma relação entre a simplicidade do modelo e a precisão dos resultados de análise. Isto porque, quando não for verificada a necessidade de uma precisão extrema, “{...} é preferível obter apenas um modelo razoavelmente simplificado”, sendo, em regra, suficiente, “a obtenção de um modelo matemático adequado ao problema em consideração” (2000, p. 48).

Além disso, Ogata (2000, p. 48) preleciona que afim de se alcançar um modelo matemático que represente de forma significativa, muitas vezes é preciso que ignoremos alguns parâmetros intrínsecos ao sistema de forma a obter um modelo linear e com parâmetros aglutinados, mas sempre com o compromisso de representar as propriedades físicas de forma mais precisa possível.

Sendo assim, o sistema descrito neste projeto será estruturado atendendo à todas as especificações consideradas, mas com o maior grau de simplicidade possível.

Fossen (2002, p.9) apresentada duas modelagens clássicas para embarcações, em que a primeira é motivada pela segunda lei de Newton e pode ser representada pelo somatório generalizado da matriz de inercia em seis graus de liberdade, três da relação das velocidades lineares e três das angulares. Já no tocante à segunda, esta consiste na representação vetorial apresentada pelo mencionado autor em (Fossen, 1991).

Por fim, salienta-se que foram utilizados apenas dois sentidos de deslocamento para atender às especificações de controle da embarcação, tendo em vista que o experimento será realizado em ambiente controlado, não existindo ondas ou nenhuma outra perturbação externa que faça com que este se desloque em outros sentidos além dos considerados.

2.1. Sistema de coordenadas

Para a análise do sistema de coordenadas do barco no ambiente controlado onde a parte experimental foi realizada, é possível a aproximação de três graus de movimento, conforme citado por Rosário (2017, p. 38). A partir disso, serão consideradas apenas as coordenadas X e Y para descrever o plano estacionário e o ângulo Ψ para indicar o rumo em que a embarcação está direcionada.

Para desenvolver a modelagem e o controle da embarcação, será adotado o sistema de coordenadas no espaço de trabalho apresentado na Figura 5.

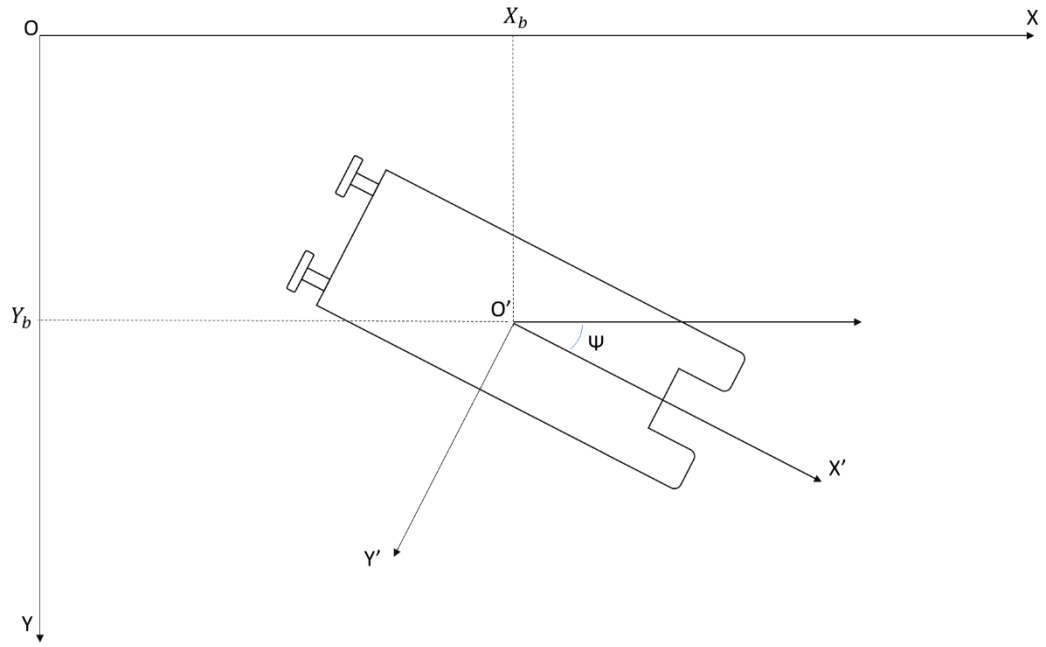


Figura 5 – Sistemas de coordenadas.

- Sistema inercial (estacionário):

O – Origem do sistema de coordenadas estacionário;

X, Y – Eixos que definem o plano horizontal no sistema estacionário;

- Sistema móvel (fixo à embarcação):

O' – Origem do sistema de coordenadas móvel;

X', Y' – Eixos do sistema móvel. Neste projeto despreza-se o balanço da embarcação, desta forma os eixos Z e Z' são paralelos.

A posição da embarcação pode ser definida como as coordenadas inerciais do ponto *O'* (origem do sistema de coordenadas móvel) em relação ao ponto *O* (origem do sistema de coordenadas estacionário). O ângulo de orientação da embarcação (Ψ) pode ser definido como a inclinação entre os eixos *X* e *X'*, conforme a Figura 5.

A partir deste ponto, essas coordenadas serão responsáveis por posicionar e possibilitar o equacionamento do sistema de interesse conforme demonstrado no próximo tópico.

2.2. Equações dinâmicas do sistema

Em primeiro lugar, o sistema que será equacionado a seguir pode ser subdividido em dois problemas segregados, quais sejam: o posicionamento da embarcação e sua direção de rumo. Sendo assim, as equações utilizam estes termos para descrever o deslocamento do veículo.

Para o equacionamento do deslocamento do barco, foram necessárias as dimensões externas e a distância entre os propulsores, medidas estas as quais estão descritas abaixo, conforme a Figura 6 e a Tabela 1.

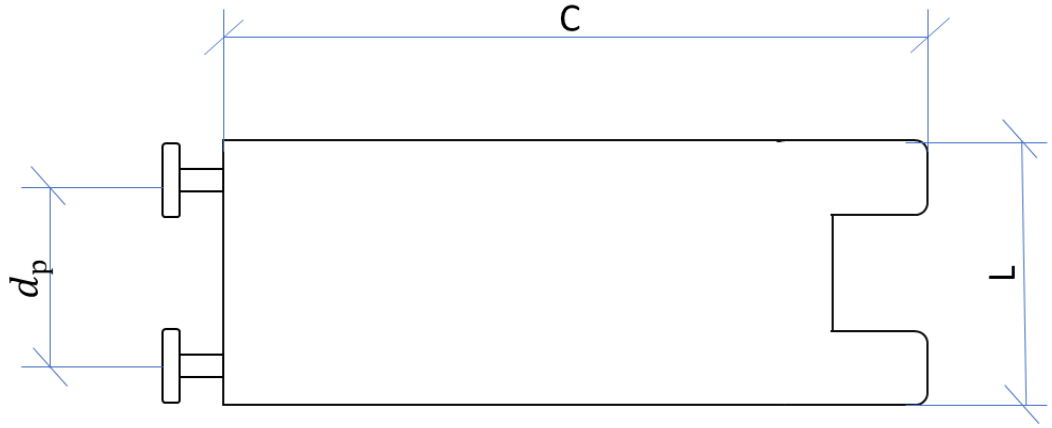


Figura 6 – Dimensões do barco.

Tabela 1 – Dimensões da embarcação.

Componente	Descrição	Dimensão (mm)
C	Comprimento da embarcação	375
L	Largura da embarcação	125
d_p	Distância entre os propulsores da embarcação	70

O deslocamento será dividido em duas partes. A primeira é o ângulo (θ), necessário para alcançar o ponto desejado, e a outra parte é a distância real (d) entre o ponto atual da embarcação e o ponto de destino. Este primeiro equacionamento está representado pela Figura 7.

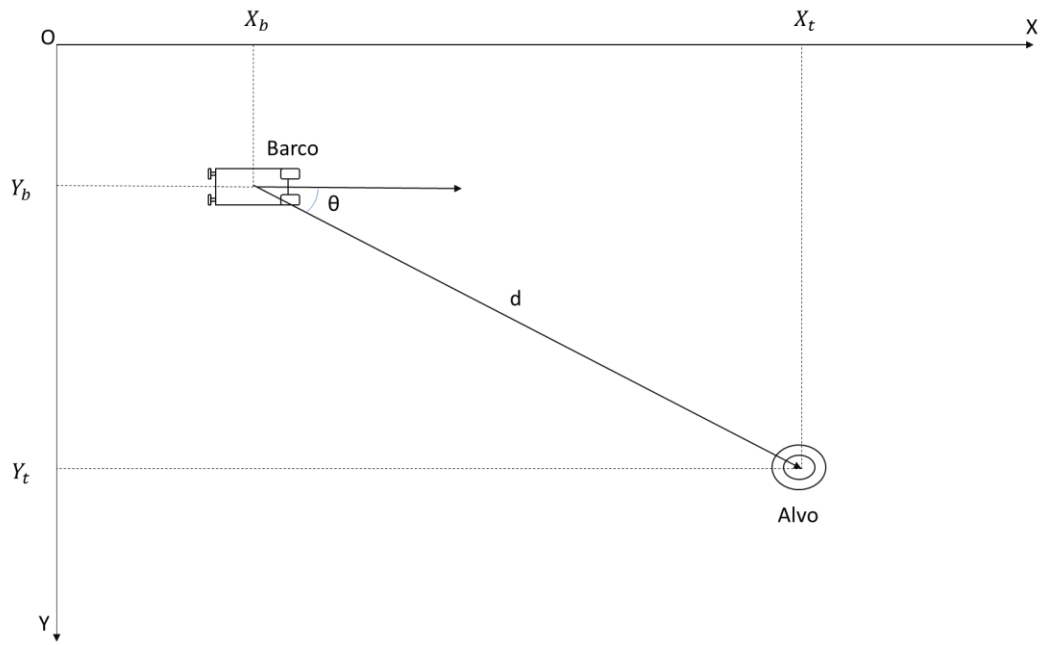


Figura 7 – Distâncias entre ponto desejado e ponto atual.

Na Figura 7, o ponto (X_b, Y_b) é o ponto atual e o ponto (X_t, Y_t) é o ponto de destino e, de acordo com as definições presentes nesta imagem, foram formuladas as seguintes equações:

$$d = \sqrt{(X_t - X_b)^2 + (Y_t - Y_b)^2}, \tag{1}$$

$$\theta = \tan^{-1}\left(\frac{Y_\mathrm{t} - Y_\mathrm{b}}{X_\mathrm{t} - X_b}\right), \tag{2}$$

Nas quais:

X_b, Y_b – Coordenadas da embarcação no sistema de coordenadas estacionário;

X_t, Y_t – Coordenadas do alvo no sistema de coordenadas estacionário.

Os termos descritos pelas equações (1) e (2) serão utilizados junto à segunda lei de Newton para descrever o movimento da embarcação. As forças geradas pelos propulsores, conforme demonstrado na Figura 8, serão as forças responsáveis tanto pelo deslocamento, quanto pela rotação do veículo, e, com isso, serão as forças que compõem as forças resultantes do equacionamento.

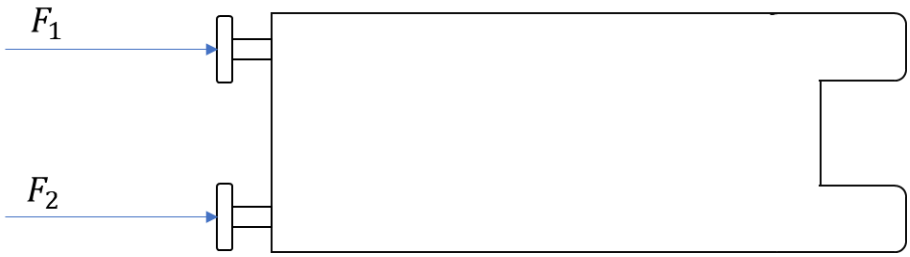


Figura 8 – Forças geradas pelos propulsores.

$$F = F_1 + F_2 = \ddot{d} \times m \tag{3}$$

$$M = (F_1 - F_2) \times d_{\mathrm{p}} = \ddot{\theta} \times J \rightarrow F_1 - F_2 = \frac{\ddot{\theta} \times J}{d_{\mathrm{p}}} \tag{4}$$

Onde:

F - Força de deslocamento;

M - Momento angular de propulsão;

F_{n} - Força do Propulsor n;

m - massa do barco;

J - Momento de inércia do barco em torno do eixo vertical;

d_{p} - Distância entre os propulsores da embarcação.

Manipulando as equações (3) e (4) descritas acima, podemos concluir que as forças F_1 *e* F_2 serão responsáveis pelo deslocamento e pela direção do movimento (rumo), conforme previsto anteriormente. A seguir, demonstra-se de forma explícita nas formulas (5) e (6) as supracitadas conclusões.

$$F_1 = \frac{(d_{\mathrm{p}} \times \ddot{d} \times m) + (J \times \ddot{\theta})}{2 \times d_{\mathrm{p}}} \tag{5}$$

$$F_2 = \frac{\left(d_{\mathrm{p}} \times \ddot{d} \times m\right) - (J \times \ddot{\theta})}{2 \times d_{\mathrm{p}}} \tag{6}$$

Destas equações, pode se chegar a representação simplificada do modelo clássico presentado por Fossen (2002, p.9), a simplificação desconsiderando três graus de liberdade, restando apenas três. Esta representação está indicada nas equações (7) e (8):

$$\mathrm{M}\dot{\mathrm{v}} = \sum_{i=1}^{n} F_i \tag{7}$$

Na qual:

M – Matriz de inercia;

$\dot{\mathrm{v}}$ – Vetor de aceleração genérico;

$$\mathrm{v} = [\dot{\mathrm{X}}', \dot{\mathrm{Y}}', \dot{\Psi}]^T; \tag{8}$$

X', Y' – Eixos do sistema de coordenadas móvel;

$\dot{\Psi}$ - Ângulo de orientação da embarcação.

Na aproximação de três graus de liberdade utilizada, podemos assumir que a matriz de inércia do barco pode ser simplificada à matriz de inércia do corpo rígida apresentada por Rosário (2017, p.39) somada a matriz de inércia de massas adicionais, que é simétrica e positiva se consideramos um fluido ideal, sem ondas nem correnteza. A matriz resultante dessas aproximações é apresentada na equação (9).

$$\mathrm{M} = \begin{bmatrix} m_b - X_{\dot{u}} & 0 & 0 \\ 0 & m_b - Y_{\dot{u}} & m_b x_g^b - Y_{\dot{r}} \\ 0 & m_b x_g^b - Y_{\dot{r}} & I_{zb} - N_{\dot{r}} \end{bmatrix} \tag{9}$$

Onde:

m_b– Massa da embarcação.

x_g^b – Coordenada da posição do centro de gravidade no sistema de coordenadas móvel.

I_{zb}– Momento de inércia sobre o eixo z.

A partir destes equacionamentos, será possível a construção das leis de controle que estarão descritas no Capítulo 4.

3. Comunicação sem fio – *ZigBee*

A comunicação eletrônica sem fio consiste basicamente em fazer dois ou mais dispositivos eletrônicos se comunicarem usando o espaço como meio físico de transmissão de informações (Ramos, 2010, p.1). A utilização dessa tecnologia ganha cada vez mais espaço na indústria, segundo Ramos (2010, p.17) devido aos baixos custos de implementação e infraestrutura quando comparado as redes cabeadas.

A mesma percepção é também citada por Lee (2007, p.1), que afirma que as redes sem fio são um dos elementos com maior crescimento na indústria de forma a proporcionar maior flexibilidade e mobilidade.

Historicamente, o crescimento das comunicações móveis caminhou devagar e dependeu principalmente do desenvolvimento tecnológico de seus componentes, como contextualiza Rappaport (2009, p.2) em sua abordagem histórica na definição do conceito do celular criado pela *Bell Laboratories* nos anos 1960 e 1970.

Já na atualidade, há uma grande variedade de protocolos para utilização de comunicações móveis e utilizando Lee (2007, p.1) que comparou os protocolos *Bluetooth, Ultra-wideband* (UWB), *ZigBee* e *Wi-fi* mostrando os benefícios do protocolo *ZigBee* por atender uma maior gama de problemas da indústria, graças a sua autonomia e flexibilidade.

A comparação de Lee (2007, p.6) também aponta que o protocolo *ZigBee*, juntamente com o *Bluetooth*, consomem menos potência quando comparado ao *Wi-fi* e UWB.

Assim, no presente projeto, será utilizado o protocolo *ZigBee* devido às vantagens mencionadas, bem como pela limitação de acesso a outras tecnologias disponíveis.

3.1. Padrão *ZigBee*

O padrão *ZigBee* é um protocolo de comunicação sem fio desenvolvido após um conjunto de empresas do mercado de eletrônica criarem um conglomerado (*ZigBee Alliance*) visando a criação de um sistema de baixo custo e potência de

transmissão voltado principalmente para os sistemas de automação (Sousa, 2016, p.57). O *ZigBee* não utiliza todas as camadas usualmente existentes nas redes de comunicação, tendo apenas a camada de aplicação e rede, mais altas e definidas pelo padrão *ZigBee*, e as camadas mais baixas de acesso (MAC) e física definido pelo padrão IEEE 802.15.4.

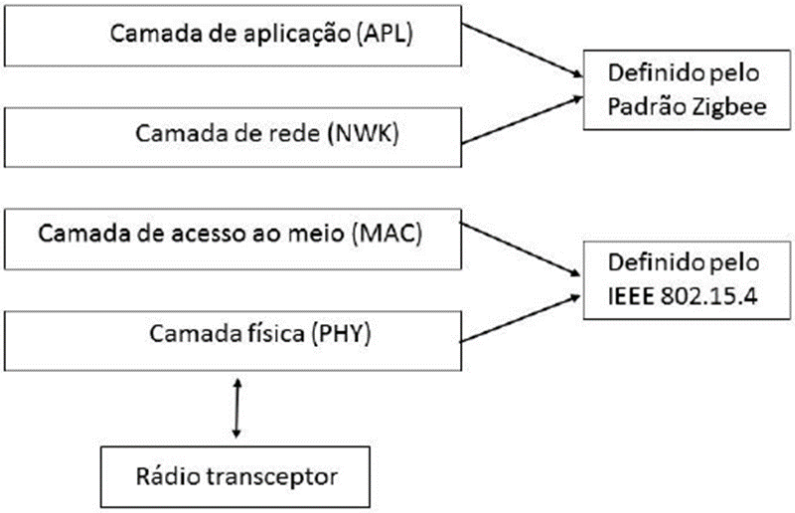


Figura 9 – Camadas de uma rede *ZigBee*. Extraído de (Ramos, 2012, p.48).

3.1.1. IEEE 802.15.4

O padrão IEEE 802.15.4 é uma subclasse das redes WPAN (*Wireless Personal Area Network)*, que são redes sem fio inicialmente desenvolvidas para comunicação de dispositivos em um raio de pouco alcance, normalmente na faixa de 10 metros (Sousa, 2016, p.54). Tal padrão foi desenvolvido para englobar redes que não necessitem de taxas tão altas de transmissão aliado a um baixo consumo de energia, visto que sua utilização tem como objetivo redes de sensores, por exemplo.

Como mencionado anteriormente, o padrão 802.15.4 foi criado pelo IEEE (*Institute of Electrical and Eletronics Engineers*), definindo as especificações da camada física (PHY) e do controle de acesso ao meio das redes LR-WPAN (Low-Rate WPAN) (Sousa, 2016, p.56). Suas vantagens, segundo IEEE (2011), são a fácil

instalação, a maior confiabilidade na transferência de dados, o protocolo simplificado e o baixo consumo de potência.

3.1.2. Camadas

A camada física é responsável pela transmissão e recepção de dados utilizando um canal de rádio através de técnicas de espalhamento e modulação. O IEEE (2011) permite utilização nas frequências de operação de 2400, 915 e 868 MHz, divididos em 16 canais (2,4 até 2,4835 GHz), 10 canais (902 até 928 MHz) ou canal único (868 até 868,16 MHz). A camada física tem por função a recepção e transmissão de dados, controle do transceptor de rádio, detecção de energia no canal a ser usado, seleção do canal a utilizar, indicador da qualidade da conexão e utilização de técnicas de acesso múltiplos canais (Sousa, 2016, p.56).

A camada de acesso ao meio controla a transferência dos dados entre os dispositivos vizinhos (ponto a ponto) utilizando técnicas de prevenção de colisão (CSMA-CA[1]), geração e sincronismo de *beacon* e segurança do dispositivo.

A camada de rede *ZigBee* é responsável pelo gerenciamento, segurança e roteamento da rede como definição da tabela de vizinhança. Os dispositivos *ZigBee* são classificados de acordo com sua funcionalidade: *Full Function Device* (FFD) e *Reduced Function Device* (RFD). O primeiro é capaz de formar uma rede, gerenciar dados e roteamento com total utilização do protocolo definido pelo IEEE 802.15.4 enquanto o RFD possui menos recursos, podendo apenas acampar na rede e realizar funções mais simples (Sousa, 2016, p.58).

[1] *Carrier sense multiple access with collision avoidance* – é um método que possui parâmetros restritivos, visando a redução da ocorrência de colisões numa rede.

3.1.2.1. Função dos dispositivos

Os dispositivos em uma rede *ZigBee* podem assumir três tipos de função:

- *ZigBee Coordinator* (ZC)

Em uma rede *ZigBee* haverá apenas um coordenador (ZC), que possuirá a capacidade de inicializar e de configurar uma rede específica atuando como um PAN *Coordinator* (Tennina, 2013, p.4). O dispositivo configurado como coordenador atua como um roteador após a rede ser estabelecida. Por ser um dispositivo FFD, possui uma pilha de instrução completa em sua memória (Sousa, 2016, p.59).

- *ZigBee Router* (ZR)

O dispositivo roteador tem como função facilitar o tráfego presente na rede oferecendo caminhos alternativos para o dado passante. O ZR pode se associar a um dispositivo coordenador (ZC) ou a outro dispositivo roteador (ZR), como ocorre na topologia tipo árvore. Devido seu perfil de atuação, o ZR também é um dispositivo FFD.

- *ZigBee End Device* (ZED)

O dispositivo final possui tarefas reduzidas e não tem a possibilidade de exercer tarefas de roteamento, não é possível associar um ZED a outro ZED pertencente na rede. Portanto é um RFD (possui reduzido conjunto de instruções) que atua como um elemento atuador, como sensores por exemplo.

3.1.3. Topologia de rede *ZigBee*

A topologia de rede *ZigBee* permite três tipos de configuração da rede: estrela, malha e árvore (Tennina, 2013).

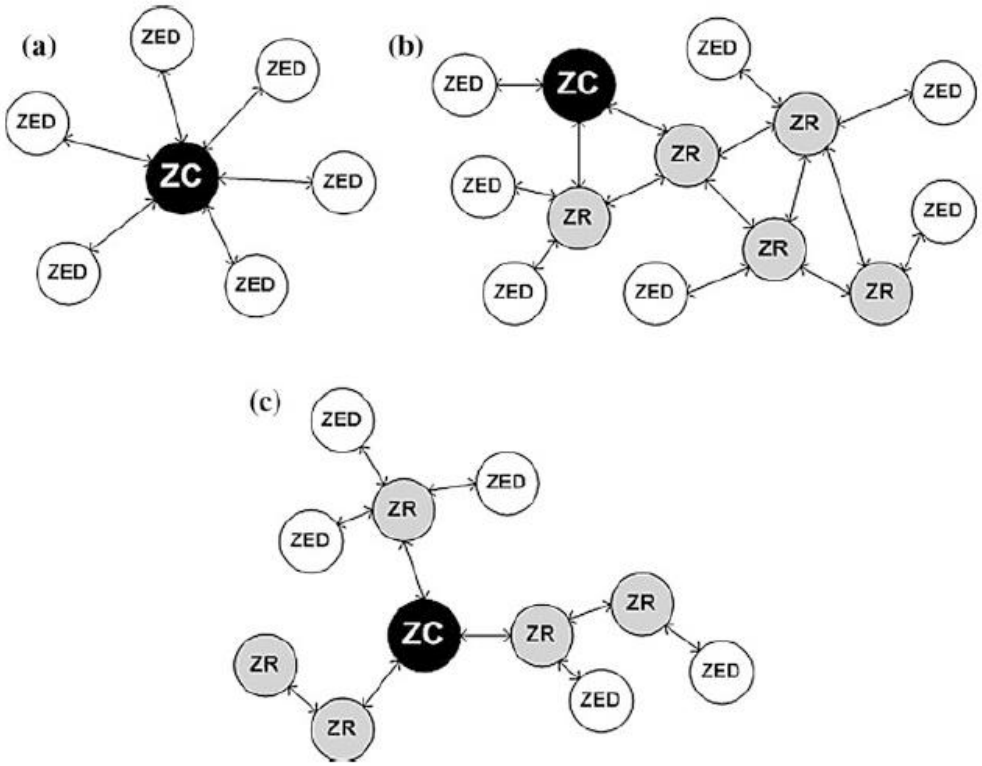


Figura 10 – Topologias de rede *ZigBee*: a) Topologia estrela. b) Topologia em malha. c) Topologia em árvore. Extraído de (Tennina, 2013, p.5).

A topologia estrela possui um dispositivo único ZC que fará a escolha do identificador da rede PAN. A topologia tem perfil centralizado, isto é, todos os elementos pertencentes a rede devem prioritariamente comunicar-se antes com o ZC que por usa vez transmitirá o dado ao outro dispositivo, podendo possuir elementos FFD ou RFD.

A topologia em malha possui um ZC responsável pela configuração de toda a rede, mas possui perfil descentralizado visto que os elementos podem comunicar-se diretamente entre si desde que estejam a um alcance possível. Os elementos FFD podem contatar outros elementos FFD e dispositivos finais podem utilizar roteadores para se comunicar com o coordenador (Sousa, 2016, p.60). A topologia malha permite maior flexibilidade a rede, mas apresenta maior complexidade devido a conectividade ponto-a-ponto de todos os elementos.

A topologia do tipo árvore é um caso variante do tipo malha em que há apenas uma rota entre dois elementos presentes na rede. Nesta rede, cada ZR controla seu próprio *cluster* de dispositivos finais.

A Tabela 2 retirada de *IEEE 802.15.4 and ZigBee as Enabling Technologies for Low-Power Wireless Systems with Quality-of-Service Constraints* (2013, p.6) compara algumas características entre os tipos de topologia.

Tabela 2 – Principais características das topologias de rede *ZigBee*, adaptado de (Tennina, 2013, p.6).

Característica	Estrela	Malha	Árvore
Escalabilidade	Não	Sim	Sim
Sincronização	Sim	Não	Sim
Período de Inatividade	Todos os nós	ZEDs	Todos os nós
Garantia de largura de banda	Sim	Não	Sim
Rotas de Redundância	Não se aplica	Sim	Não
Sobrecarga do protocolo de roteamento	Não se aplica	Sim	Não

3.1.4. Formação, endereçamento e transmissão de uma rede *ZigBee*

Para inicialização da rede *ZigBee* é necessário um dispositivo FFD que atuará com a função de coordenador criando uma rede PAN único com identificação, também exclusiva, chamado PAN ID (Sousa, 2016, p.61). Pela PAN ID que é possível diferenciar as redes *ZigBee* existentes, assim cada elemento que vá juntar-se a rede terá sua PAN ID selecionada.

Configurada a rede PAN, o coordenador ao iniciar fará uma varredura do espectro em busca dos níveis de energia dos canais disponíveis, descartando os que apresentarem níveis energia excessivos (DIGI *International*, 2008, p.19). Realizada a varredura, o ZC enviará pacotes piloto nos canais disponíveis detectados que permitam a comunicação e entrada na rede de outros dispositivos configurados com a PAN ID escolhida no coordenador.

3.1.4.1 Enderaçamento

Há dois tipos de endereçamento numa rede de *ZigBee*:

- Endereço 64 bits: É um endereço único e imutável definido pelo fabricante do dispositivo;

- Endereço 16 bits: É um endereço dinâmico que os dispositivos devem receber do ZC (ou ZR dependendo da topologia) assim que é permitida a entrada na rede PAN disponível.

3.1.4.2 Transmissão

A transmissão dos dados na rede *ZigBee* pode ocorrer no método *broadcast* ou *unicast.* No primeiro caso, a transmissão ocorre de um dispositivo para toda a rede, sem que seja necessário especificar o destinatário. Nesse caso todos os dispositivos finais receberão os dados enviados e sua utilização ficará a cargo de sua própria necessidade.

Já na transmissão *unicast,* o dado é enviado de forma direta a um dos dispositivos presente na rede utilizando-se o endereço de 16 bits do destinatário. Para descobrir tal endereço, o módulo remetente faz uma busca na rede pelo endereço 64 bits do destinatário em uma transmissão *broadcast*, os receptores comparam o endereço recebido e, caso a comparação seja positiva, enviam seu endereço dinâmico para o transmissor que, posteriormente, enviará o pacote de dados transmitidos.

3.1.5. Modos de operação

Os dispositivos em uma rede *ZigBee* possuem até cinco modos diferentes de operação que variam diretamente conforme a necessidade, considerando principalmente o consumo de energia (Sousa, 2013, p.62).

O modo de repouso é utilizado quando ainda não há demanda de operação para determinado dispositivo. Este último fica em estado de espera aguardando a notificação para operar, visando principalmente a economia de energia. Por outro lado, também há o modo inativo, que é um estado transitório em que o dispositivo aguarda qual estado deverá ir para realizar a tarefa que se aproxima. Após o modo inativo, o dispositivo pode voltar ao estado de repouso ou ir para o modo de recepção, transmissão ou comando.

Como o nome descreve, o modo de transmissão e recepção leva o dispositivo ao estado em que um dado será enviado ou recebido e armazenado no *buffer* da transmissão serial. O modo de comando serve para modificar ou ler parâmetros dos módulos dos dispositivos a partir da interpretação de linhas de comando.

3.1.6. Módulos XBee

Os módulos XBee são dispositivos de rede sem fio que utilizam o protocolo *ZigBee*, cuja diferença é descrita no manual do usuário, onde é possível verificar que *ZigBee* é apenas o protocolo de rádio frequência que pertence a *ZigBee* Alliance, que foi desenvolvido para aplicações de baixa potência e baixa taxa de dados. Já o Xbee é o nome do módulo físico desenvolvido pela Digi para a utilização do protocolo.

A junção de hardware, protocolo e firmware define o produto XBee final e a qual série ele pertence. Desta forma, ressalta-se que existem diversas séries, todavia, um módulo pertencente a uma série não se comunica com outro de uma série diferente (Sousa, 2016, p.65).

Outra diferenciação nos módulos ocorre entre os modelos XBee e XBee-Pro. A principal característica que muda entre eles é que o XBee-Pro possui uma maior

potência de transmissão, o que o faz superior na utilização de comunicação sem fio em distâncias elevadas.

Os módulos disponibilizados para realização dos experimentos deste projeto são XBee-Pro Série 2 (S2). A Tabela 3 apresenta algumas das suas características.

Tabela 3 – Características do módulo XBee-Pro S2, adaptado de Sousa (2016, p.66).

Característica	XBee-Pro S2
Distâncias internas	90 m
Distâncias externas	3,2 km
Potência de transmissão	50 mW (17 dBm)
Sensibilidade de recepção	-102 dBm
Taxa de transmissão	250 kbps

Os módulos XBee se comunicam com outros dispositivos por meio de comunicação serial assíncrona podendo se comunicar com qualquer nível lógico compatível com Receptor/Transmissor Universal Assíncrono (tradução livre de UART). Assim, um dispositivo que tenha interface UART pode enviar dados para os módulos XBee através do pino *data in* (Sousa, 2016, p.69).

3.1.6.1. Modos de comunicação dos módulos XBee

Há dois modos de comunicação serial suportáveis pelos módulos XBee:

- Modo transparente: O modo transparente realiza a transmissão dos dados que saem do UART similar a uma transmissão serial cabeada. Este modo é mais simples e indicado a redes menos robustas.

- Modo API: Este modo de comunicação é alternativo ao modo transparente e possui maior complexidade porque a comunicação e envio das informações são feitos através de pacotes de dados. Por

> conseguinte, é possível endereçar os pacotes para um destinatário específico, aplicando métodos que diminuam a perda de informação no meio.

Um quadro do modo API contém todos os dados que definirão operações ou eventos que serão utilizados pelo módulo (DIGI User Guide, 2018, p.28). Nesse sentido, o quadro de informação transmitido (que entra no *data in*) inclui o quadro com as informações da comunicação de rádio e o quadro com comandos do modo transparente (AT).

A operação em modo API também permite a facilitação de outras operações como a transmissão de dados para múltiplos destinos sem a necessidade de entrar no modo de comando, notificação de sucessos ou falhas na recepção de cada pacote de dados e identificação do remetente de cada pacote recebido (DIGI User Guide, 2018, p.28).

3.2. Motivação e utilização do padrão *ZigBee*

No presente projeto a utilização do padrão *ZigBee* para comunicação entre a embarcação e o sistema de captura de movimento foi escolhido devido as vantagens comentadas no início desse capítulo (menor consumo de energia e por ser também um protocolo indicado para taxa de dados mais baixa) aliado a disponibilidade dos componentes para uso imediato.

A topologia escolhida para o experimento será a do tipo estrela devido ao número de dispositivos que serão conectados: a embarcação e o emissor de sinal conectado ao sistema de monitoramento. Por ser o elemento que enviará os dados de posicionamento, o computador sincronizado com a VICON será o coordenador (ZC, elemento FFD) enquanto a embarcação será configurada como dispositivo final (ZED, elemento RFD).

O modo de comunicação será o modo API por possuir uma gama maior de utilidades, ainda mais considerando que será necessário o envio continuo de diferentes variáveis de posicionamento. Outro ponto vantajoso é o modo API ser mais

eficiente para construir o quadro com os dados processados pelo sistema de monitoramento e enviá-los a embarcação por meio de um *script* que será executado no MatLab (detalhes de sua construção ao final da seção 3.3).

3.3. Configuração da rede e dos dispositivos

Nesta subseção, será demonstrado o passo a passo para a configuração dos módulos XBee que foram utilizados para a comunicação do sistema. Para a realização desta configuração, foi necessário usar o programa XCTU na versão 6.4.2.

A seguir, serão apresentadas as telas onde se realizam as configurações. Na Figura 11 é mostrada a tela inicial do *software* XCTU.

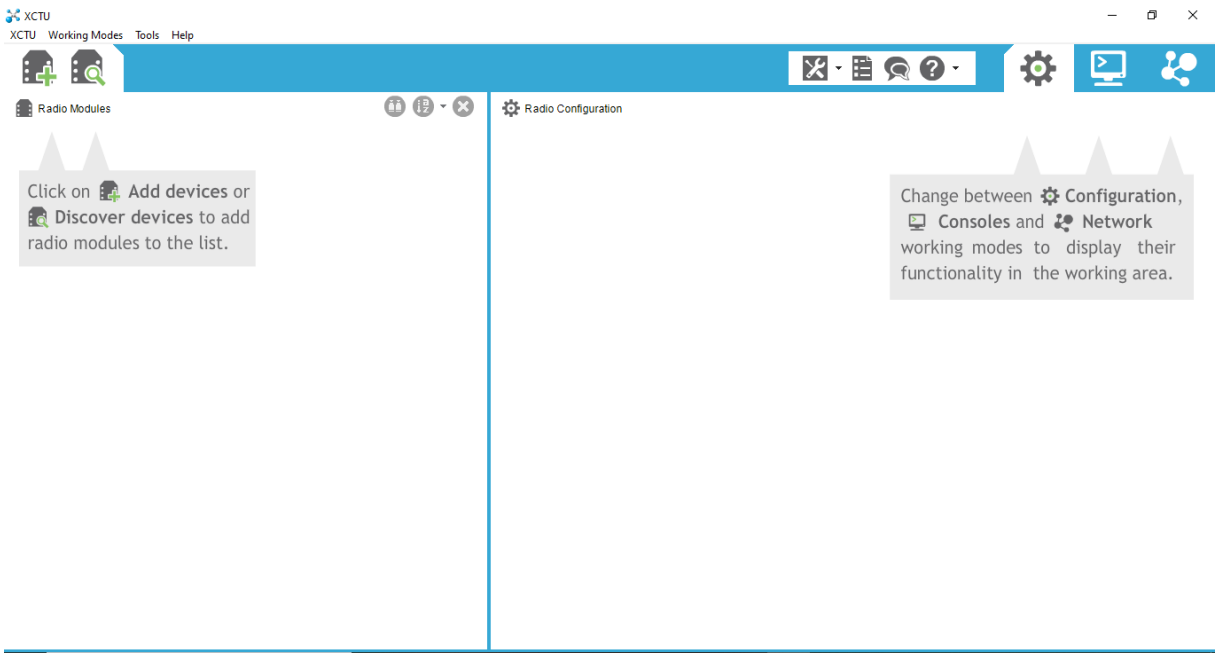


Figura 11 – Tela inicial do software XCTU.

Para iniciar a configuração, devemos conectar o módulo com seu adaptador na porta USB do microcomputador e, após, clicar em adicionar dispositivos, no canto superior esquerdo da tela. Surgirá uma tela (Figura 12) para selecionar as configurações da porta em que o dispositivo XBee está conectado.

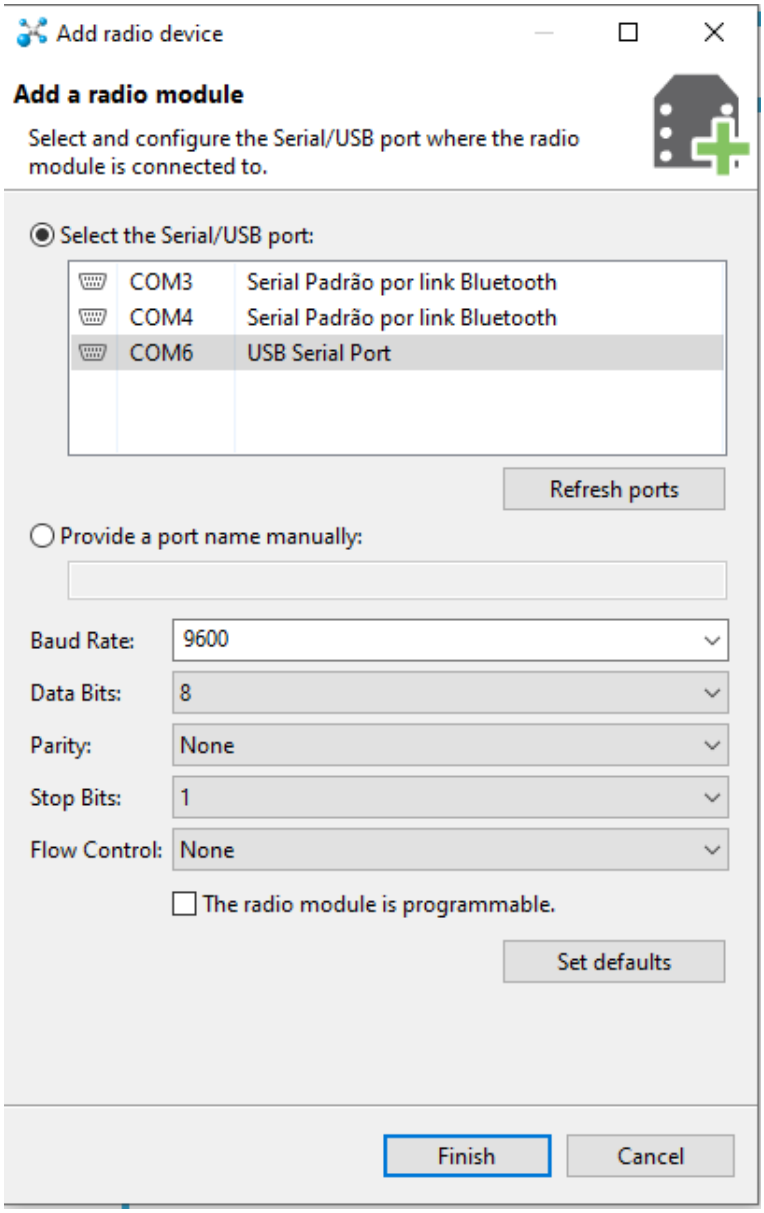


Figura 12 – Tela para adicionar dispositivos.

Após selecionar para finalizar as configurações da porta do computador, será adicionado o dispositivo no campo a esquerda da tela inicial, e, ao clicar sobre este dispositivo, será possível configurá-lo, conforme demonstrado na Figura 13.

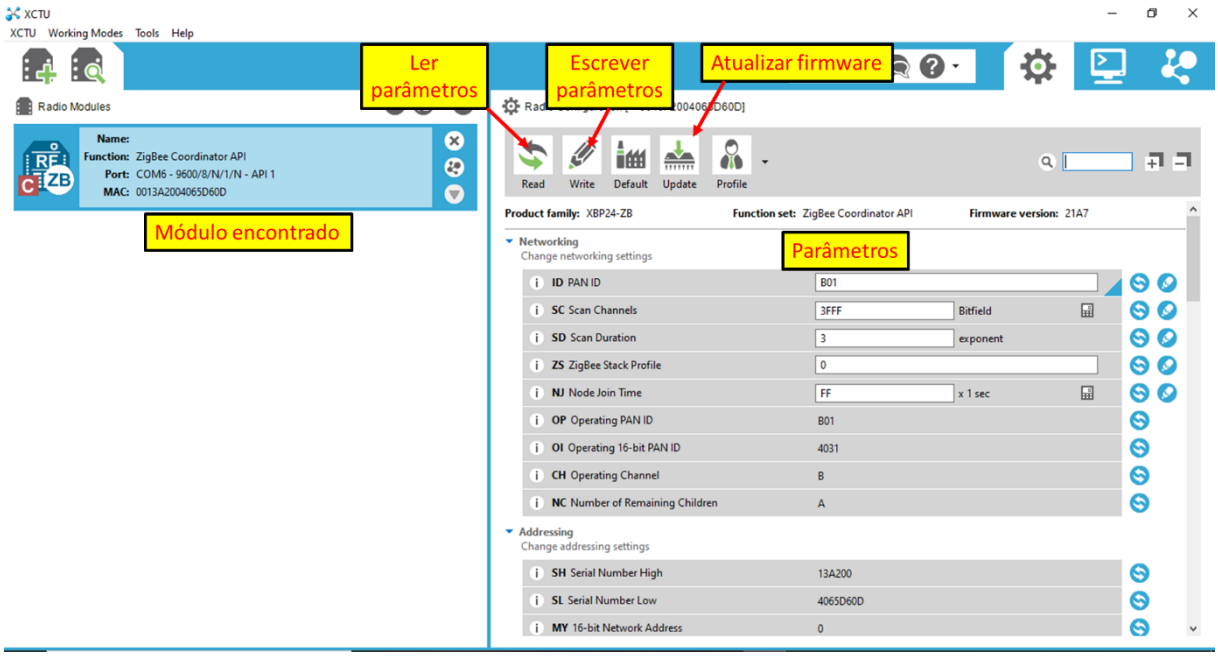


Figura 13 – Tela com parâmetros.

Na atualização do *firmware* definimos qual dispositivo será o coordenador, que ficará conectado no microcomputador e terá a função de enviar as informações para a embarcação, assim como qual dispositivo será o dispositivo final que ficará conectado ao Arduino no barco.

As configurações dos parâmetros foram realizadas de acordo com as Tabela 4 e Tabela 5 demonstradas abaixo:

Tabela 4 – Configuração Coordenador.

Parâmetro	**Valor**	**Função**
PAN ID	B01	Identificação da rede
NI	Coord	Nome do dispositivo na rede
AP	1	Modo API – (1) ativo
DH	0	Parte alta do endereço de 64 bits do módulo de destino
DL	FFFF	Parte baixa do endereço de 64 bits do módulo de destino

Tabela 5 – Configuração Dispositivo Final.

Parâmetro	Valor	Função
PAN ID	B01	Identificação da rede
NI	Barco_1	Nome do dispositivo na rede
AP	1	Modo API – (1) ativo
DH	13A200	Parte alta do endereço de 64 bits do módulo de destino
DL	4065D60D	Parte baixa do endereço de 64 bits do módulo de destino

A configuração do endereço de 64 bits do coordenador definida na Tabela 4 é utilizada para transmissão *broadcast*. Terminada a configuração dos módulos, foi utilizado o Matlab para a construção dos quadros a serem enviados pelo coordenador. O quadro é estruturado conforme demonstrado na Tabela 6.

Tabela 6 - Montagem do quadro *ZigBee.*

Programa do Matlab – Montagem do quadro *ZigBee*

```
APIframe(1,:) = '7E'; % Frame Class Name for reading only
APIframe(2,:) = '00';  % length 1
APIframe(3,:) = '20'; % length 2
APIframe(4:5,:) = ['10';'01'];   % type & ID
APIframe(6:13,:) = ['00'; '13'; 'A2'; '00'; '40'; '63'; 'D2'; '2F'];
%address
APIframe(14:15,:) = ['FF'; 'FE']; %address 16BIT
APIframe(16,:) = '00';
APIframe(17,:) = '00';
APIframe(18:35,:) = [ x_barco_h ; x_barco_l ; y_barco_h ; y_barco_l ;
rumo_barco_h ; rumo_barco_l ;
                      x_ref_h ; x_ref_l ; y_ref_h ; y_ref_l; Kc_1_h;
Kc_1_l; Td_1_h; Td_1_l; Kc_2_h;
                      Kc_2_l; Td_2_h; Td_2_l];% APIframe;
checksumframe = hex2dec(APIframe);
checksum = dec2hex(hex2dec('FF') - (mod((sum(checksumframe(4:35))),
256)));
APIframe(36,:) = checksum;
APIframe = hex2dec(APIframe);
```

O quadro é estruturado por conjuntos de 16 bits, escritos de forma hexadecimal, e, dentre estes, constam as informações necessárias a serem enviadas para embarcação as quais se encontram nos conjuntos 18 à 35. Além disso, vale destacar que o último conjunto contém o *checksum* para verificação do quadro quando este é recebido.

3.4. Extensão da rede para uma frota de barcos

Por último, vale salientar que para a inclusão de novos elementos na rede, como no caso de se querer transformara rede para uma aplicação que controle uma frota de barcos e não apenas um veículo não tripulado, há duas soluções sendo mantida a comunicação *ZigBee*. A primeira é apenas utilizarmos os novos elementos com a configuração de dispositivo final, neste caso será mantida a topologia estrela demonstrada na Figura 10 – (a), assim o coordenador conectado no microcomputador terá a responsabilidade de enviar e receber os pacotes de todas as embarcações.

A outra solução é utilizar a topologia em malha representada na Figura 10 – (b), desta forma todas as embarcações deveriam ter seus módulos configurados como roteadores e assim poderiam se comunicar entre si, além de se comunicarem com o coordenador.

4. Controle

A ação de controle consiste em modificar o valor de uma variável para que esta atinja um ponto desejado. Todavia, para que o controle seja possível, é necessário que o controlador seja capaz de regular o fornecimento de energia de forma a ser suficiente para que a mudança pretendida seja alcançada, como pode-se concluir a partir da explicação desenvolvida por Castrucci (2011, p. 1):

Por outro lado, o controlador automático pode ser definido como um controlador o qual produz sua ação de controle a partir do erro calculado entre a variável de controle e o valor de referência desejado. Nesse sentido, explicita Ogata (2000, p. 177), que este controlador compara a variável medida no processo com a grandeza de referência, este gerará um sinal que tentará reduzir o erro entre essas variáveis a zero, e o que é denominado ação de controle.

O processo de elaboração de um projeto de controlador que seja adequado para a aplicação desejada é realizado em duas etapas. Na primeira delas, há a determinação da estrutura do controlador; enquanto na segunda, há o ajuste dos parâmetros dos controladores para a resposta desejável do sistema. Sendo assim, no presente capítulo, será apresentada a primeira etapa e, já a segunda, será desenvolvida no Capítulo 5, visto que os parâmetros serão obtidos de forma experimental.

O objetivo pretendido quanto ao controlador da embarcação consiste em fazer com que o veículo não tripulado siga um outro objeto, conforme proposto na seção 1.1. Para isso, serão utilizados controladores automáticos com o propósito de comandar a distância d e o ângulo Θ (vide Figura 7), de acordo com o que foi apresentado durante o Capítulo 2.

Assim, conforme o que foi desenvolvido nos parágrafos anteriores, os subcapítulos a seguir englobarão as definições da lógica utilizada para geração da ação de controle, o tipo de controlador utilizado e sua implantação no Arduino.

4.1. Descrição do controle

Passando-se à descrição do controle desenvolvido ao longo deste trabalho, observa-se que, com o objetivo de facilitar o controle do posicionamento do barco, foi utilizado o modelo matemático descrito no Capítulo 2, cujo movimento foi divido em duas partes, quais sejam: o rumo e a distância para o alvo.

Em um primeiro momento, é realizado o cálculo da distância dada pela equação (1), que já é, por definição, o erro de posição sem a consideração do rumo. Entretanto, para evitar a singularidade no caso do cálculo do rumo desejado, uma vez que o rumo desejado é calculado através da tangente, o valor das coordenadas não podem ser iguais porque o resultado do cálculo de rumo seria indeterminado, neste caso será utilizado um valor a ser subtraído da distância calculada anteriormente, para que o barco não fique exatamente sobre o ponto desejado e sim apontando para o ponto de destino.

No controle de rumo, leva-se em consideração apenas o ângulo Θ, calculado através das coordenadas do barco (X_b, Y_b) e do alvo (X_t, Y_t) de acordo com a equação (2), sendo o valor desejado e o valor atual do rumo, ângulo Ψ.

Todas as variáveis reais utilizadas para controle são adquiridas pelo sistema de monitoramento visual Vicon e enviadas para os controladores através da comunicação *ZigBee* estabelecida entre o microcontrolador e o microcomputador. Além disso, vale mencionar que o formato dessa comunicação para aquisição destes dados será apresentado na seção 4.3.

4.2. Controlador PD

O controlador PID (ações Proporcional, Integral e Derivativa) é um dos controladores mais famosos por conta da facilidade de construção e de ajustes dos seus parâmetros. Nesse sentido, conforme descrito por Castrucci (2011, p.229), é um controlador que apresenta ajuste fácil dos seus parâmetros e uma atuação robusta o suficiente para ambientes industrias.

Dorf (2001, p.545) afirma que a grande popularidade destes controladores automáticos é atribuída a seu desempenho robusto sob uma grande variedade de situações operacionais, com uma funcionalidade simples que permite fácil operação por conta dos seus responsáveis na indústria.

Nas duas definições, é utilizado o termo “robusto” para descrever esse tipo de controlador, cujo sentido se refere à capacidade deste de absorver as incertezas e perturbações incluídas no processo que se deseja realizar o controle.

De acordo com o padrão ISA (Castrucci, 2011, p.229), o modelo matemático utilizado para descrever um controlador PID é:

$$u(t) = \mathrm{K}_c \left(e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + T_D \frac{de(t)}{dt} \right), \tag{10}$$

Na qual:

$u(t)$ – Sinal de saída do controlador, ação de controle;

$e(t)$ – Sinal de entrada do controlador, erro;

K_c – Ganho Proporcional;

T_I – Tempo Integral;

T_D – Tempo Derivativo;

A ação proporcional é a relação direta de um ganho, parâmetro do controlador, em relação ao erro calculado entre o valor desejado e o valor real da variável de controle de acordo com a equação.

$$u(t) = \mathrm{K}_c e(t). \tag{11}$$

Conforme representado no modelo anterior, observa-se que esta ação proporcional pode ser entendida, basicamente, como o efeito de um amplificador com ganho ajustável para seleção do parâmetro do controlador.

Por outro lado, a ação derivativa atua na velocidade de variação do erro, ou seja, na taxa de variação, diminuindo assim o tempo de resposta do controlador. Já acerca do tempo derivativo T_D , pode-se defini-lo como o tempo em que a resposta derivativa será adiantada da resposta proporcional. A referida ação pode ser resumida na equação.

$$u(t) = \mathrm{K}_d \frac{de(t)}{dt}, \tag{12}$$

Na qual:

$\mathrm{K}_d = \mathrm{K}_c \mathrm{T}_D$ – Ganho Derivativo;

No tocante à componente integral do controlador PID, deve-se observar que este atua na taxa proporcional do sinal de entrada e visa eliminar o erro de regime permanente, ou seja, o erro de *offset*. Tendo em vista que este não é um problema para a trajetória desejada e não há perturbações ambientais, pudemos simplificar a implementação do controlador retirando essa ação.

Com isso, o sistema de controle que tem como objetivo apontar e deslocar a embarcação até um destino pré-definido, o qual será composto por dois controladores do tipo Proporcional e Derivativo (PD), um para comandar o rumo e outro para a distância entre o veículo não tripulado e o alvo desejado.

A soma das ações de controle dos dois controladores foi decomposta no acionamento dos dois motores conforme as equações:

$$U_1 = \frac{U_d}{k+1} + \frac{k \times U_{Psi}}{k+1}, \tag{13}$$

$$U_2 = \frac{U_d}{k+1} - \frac{k \times U_{Psi}}{k+1}, \tag{14}$$

Nas quais:

U_x – Sinal de acionamento do motor x ;

U_d– Sinal de saída do controlador de distância;

U_{Psi}– Sinal de saída do controlador de rumo;

$k = 4$ – Peso da ação de controle de rumo em relação à ação de controle de distância.

Os pesos das ações de controle das equações acima, foram obtidos de forma experimental, por meio de aproximações sucessivas até o alcance de valores satisfatórios para a resposta do sistema.

Desta forma, a Figura 14 representa o diagrama de blocos do sistema de controle completo da embarcação, considerando todos os termos desenvolvidos anteriormente.

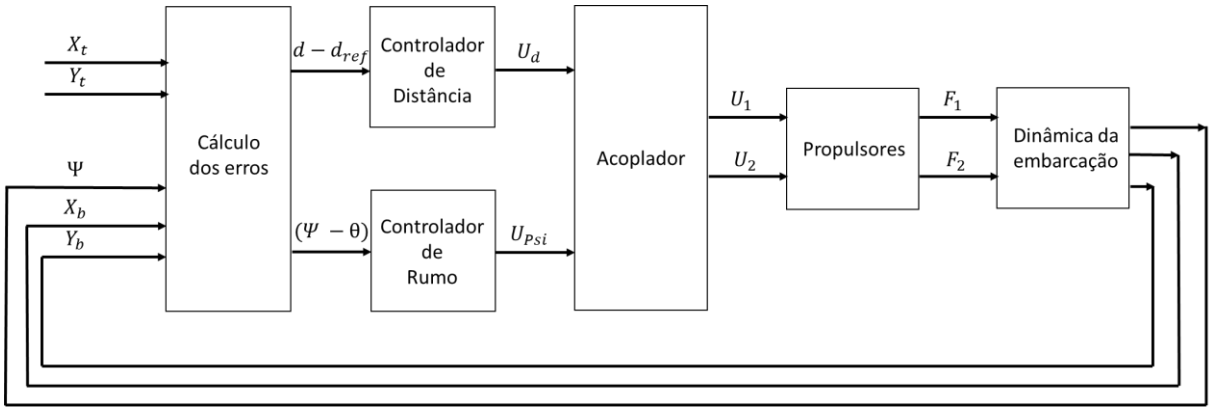


Figura 14 – Diagrama do sistema de controle.

O cálculo dos erros demonstrado no diagrama de bloca anterior, foram solucionados pelas equações a seguir e estes valores foram utilizados como *input* nos seus respectivos controladores.

$$erro_d = d - d_{ref}, \tag{15}$$

$$erro_\Psi = \Psi - \theta\ , \tag{16}$$

Nas quais:

d – Distância entre o barco e o alvo;
$d_{ref} = 45\ cm$ – Distância desejada entre o barco e o alvo;
Ψ – Rumo atual da embarcação;
θ – Rumo desejado.

O valor d_{ref} foi calculado utilizando o tamanho da embarcação e de maior alvo utilizado nos testes, o valor de 45 cm é para que a embarcação fique a 5 cm deste alvo.

4.3. Implementação do controle no microcontrolador Arduino

Nesta seção, será apresentado como foi realizada toda a implementação do sistema de controle embutido no microcontrolador Arduino.

O primeiro ponto a ser observado é que, uma vez que utilizaremos as saídas PWM deste microcontrolador para o acionamento dos motores de corrente continua, foi necessária uma preocupação com a frequência utilizada nessas saídas, de forma a evitar ruídos excessivos e vibrações nos motores CC. Por isso, tivemos que proceder a uma elevação na frequência base do microcontrolador.

Para o aumento da frequência dos PWM, foram utilizados os códigos demonstrados na Tabela 7.

Tabela 7 – Alteração da frequência dos PWMs.

Trecho de programa do Arduino

TCCR0B = TCCR0B & B11111000 | B00000010; // Troca a frequência do PWM das portas D5 e D6 para 7812.50 Hz
 	
TCCR1B = TCCR1B & B11111000 | B00000010; // Troca a frequência do PWM das portas D9 e D10 para 3921.26 Hz
 	
TCCR2B = TCCR2B & B11111000 | B00000010; // Troca a frequência do PWM das portas D3 e D11 para 3921.26 Hz

Os controladores PD resultantes da retirada da ação integral da equação (7), conforme descrito anteriormente, os quais foram necessários para atender o modelo matemático previsto, foram implementados de acordo com a Tabela 8. Já para o cálculo da derivada, foi utilizada uma aproximação da velocidade de variação do erro através da subtração entre o erro do ciclo anterior e erro atual, utilizando o resultado para multiplicar pelo inverso da frequência de amostragem do sistema de monitoramento visual.

Tabela 8 – Controlado PD implementado.

Trecho de programa do Arduino

```
//================================================implementação para o PD
float PD_barco(float Kc, float Td, float T, float erro, float erro_ant)
{
float controle;
float veloc;
float P;
float D;

P= Kc*erro;

veloc= (erro-erro_ant)*T;

D= Kc*Td*veloc;

controle = P + D;

return controle;

}
```

Além disso, uma vez que a variável de controle da frequência do PWM é limitada em 255, foi necessária a implementação de um limitador para evitar erros quando as ações de controle forem maiores do que o valor citado anteriormente e, para realização desta função, foi utilizada uma comparação básica entre a ação de controle e o limite permitido no PWM. A Tabela 9 apresenta o limitador desenvolvido.

Tabela 9 – Limitador da saída do controlador PD.

Trecho de programa do Arduino

```
PD_d = PD_barco(Kc_d, Td_d, Freq, erro_d, d_ant);

if (PD_d>(255.0-zm))
        U_d=(255.0-zm);
else if(PD_d<(-255.0+zm))
        U_d=(-255+zm);
else U_d=PD_d;

PD_Psi = PD_barco(Kc_Psi, Td_Psi, Freq, erro_Psi, Psi_ant);

if (PD_Psi>(255.0-zm))
        U_Psi=(255.0-zm);
else if(PD_Psi<(-255.0+zm))
        U_Psi=(-255+zm);
else U_Psi=PD_Psi;
```

Conforme evidenciado no início deste capitulo, todas as variáveis reais necessárias para a realização do controle são obtidas através da rede *ZigBee* estabelecida entre o microcomputador e o microcontrolador e, além desses valores, os parâmetros dos controladores, o ganho proporcional e o tempo derivativo também serão enviados pelo microcomputador, para tornar mais fácil a calibração dos controladores automáticos. Após, estes dados são lidos pelo Arduino bit a bit por meio da decodificação do quadro de comunicação. O código para verificação desses quadros e obtenção desses valores é feito pela rotina apresentada na Tabela 10.

Tabela 10 – Leitura do quadro *ZigBee.*

Trecho de programa do Arduino

xbee.readPacket(); //=== Ler continuamente algum pacote ZigBee (ZB)
 	
if (xbee.getResponse().isAvailable()) //==recebeu algo
{
if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)//== É um pacote ZB (API ID 0x90 - received packet)
{
xbee.getResponse().getZBRxResponse(rx); //==Aloca os dados recebidos na calsse rx
if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED)
{

Para realização da sobreposição das ações de controle realizadas pelos controladores de rumo e distância, foram necessárias algumas premissas e cuidados.

O primeiro deles foi a definição do controle prioritário, ou seja, determinar qual das ações deveria ser mandatória ou se as duas teriam a mesma capacidade de controle dos propulsores dos barcos. Assim, foi definido que o rumo deve ser mandatório, com o objetivo de evitar que houvessem movimentos extensos para direções equivocadas, e, desta maneira, a ação do controlador de rumo é 4 vezes maior que a ação do controlador de distância.

Ademais, para que a embarcação não comece a se deslocar no sentido oposto ao do alvo, foi necessária a utilização de um habilitador para o controle de distância, evitando, assim, que ele inicie o movimento antes de estar a, pelo menos, noventa graus do alvo. O habilitador foi desenvolvido de acordo com a Tabela 11 o acionamento completo dos motores está ilustrado na Tabela 12.

Tabela 11 – Habilitador do controlador de distância.

Trecho de programa do Arduino

```
//verifica se a diferenca do rumo e o angulo da trajetória é menor que o angulo limite pré-definido
if (erro_Psi>ang_lim||erro_Psi<(-ang_lim)){

        U1 = (U_Psi)*Duty;     // Sinal para ponte H do motor 1
        if (U1>=0){
                U1=U1+zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,U1);analogWrite(A_neg,0);}
        else {
                U1=U1-zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,0);analogWrite(A_neg,(-U1));}
                U2 = (-U_Psi)*Duty;     // Sinal para ponte H do motor 2
        if (U2>=0){
                U2=U2+zm;
                digitalWrite(HB,h);
                analogWrite(B_pos,U2);analogWrite(B_neg,0);}
        else {
                U2=U2-zm;
                digitalWrite(HB,h);
                analogWrite(B_pos,0);analogWrite(B_neg,(-U2)); }
}
```

Tabela 12 – Acionamento completo dos motores.

Trecho de programa do Arduino

```
else{
        U_d=U_d/5*Duty;
        U_Psi=4*U_Psi/5*Duty;

        U1 = round(U_d+U_Psi);     // Sinal para ponte H do motor 1
        if (U1>=0){
                U1=U1+zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,U1);analogWrite(A_neg,0);}
        else {
                U1=U1-zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,0);analogWrite(A_neg,(-U1));}
                U2 = round(U_d+(-U_Psi));     // Sinal para ponte H do motor 2
        if (U2>=0){
                U2=U2+zm;
                digitalWrite(HB,h);
                analogWrite(B_pos,U2);analogWrite(B_neg,0);}
        else {
                U2=U2-zm;
                digitalWrite(HB,h);
                analogWrite(B_pos,0);analogWrite(B_neg,(-U2)); }
}
```

Por outro lado, também é possível verificar nas Tabela 11 e Tabela 12 que foi necessária a inclusão de mais um termo antes de enviar os comandos para os motores, está inclusão foi realizada para compensar a zona morta dos propulsores, ponto criado no momento da inversão dos motores por conta da tensão mínima necessária para o acionamento desses e para ajudar na melhora dos controladores.

Esta zona morta pode ser demonstrada de acordo a Figura 15, a resposta dos controladores com o compensador implementado pode ser visualizada na Figura 16, pode-se perceber ainda que a tensão enviada para os propulsores vai de 1 até -1 volt de tensão quando o sinal dos controladores passa do plano negativo para o positivo, ou vise e versa.

Este valor de 1 volt foi obtido por meio de medições realizadas utilizando uma fonte controlada de corrente continua, de forma que o acionamento do propulsor foi realizado diretamente pela fonte.

Com esse valor medido, foi possível a obtenção do valor do parâmetro *zm* demonstrado nas tabelas anteriores de forma a eliminar a zona morta, este parâmetro foi definido com o valor de 45, que permitia que a saída PWM do Arduino ficasse entrono do 1 volt, a tensão necessária para o acionamento dos motores.

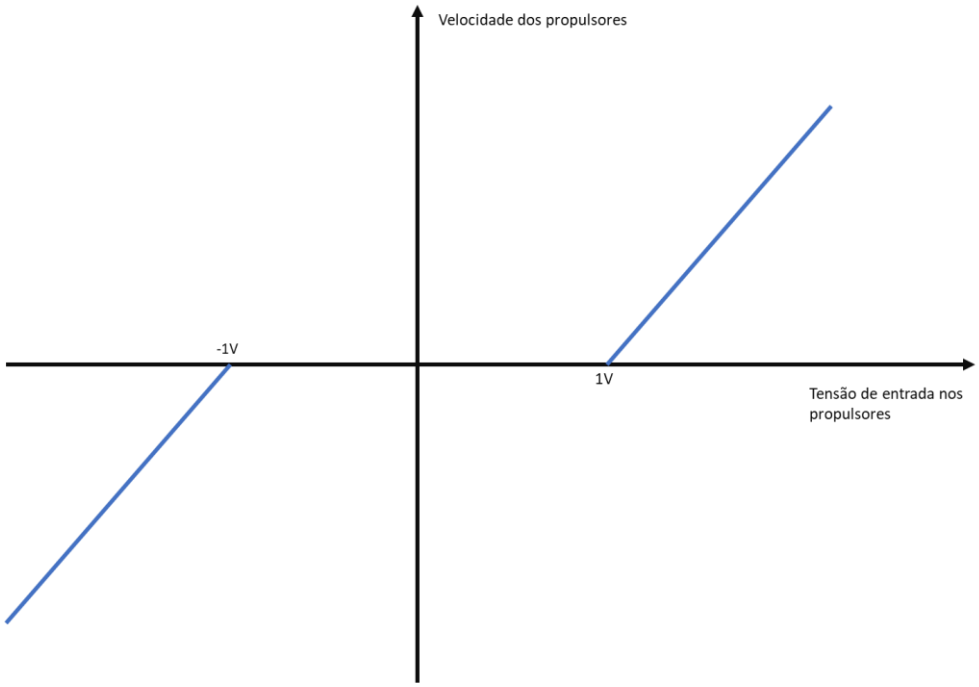


Figura 15 – Zona morta.

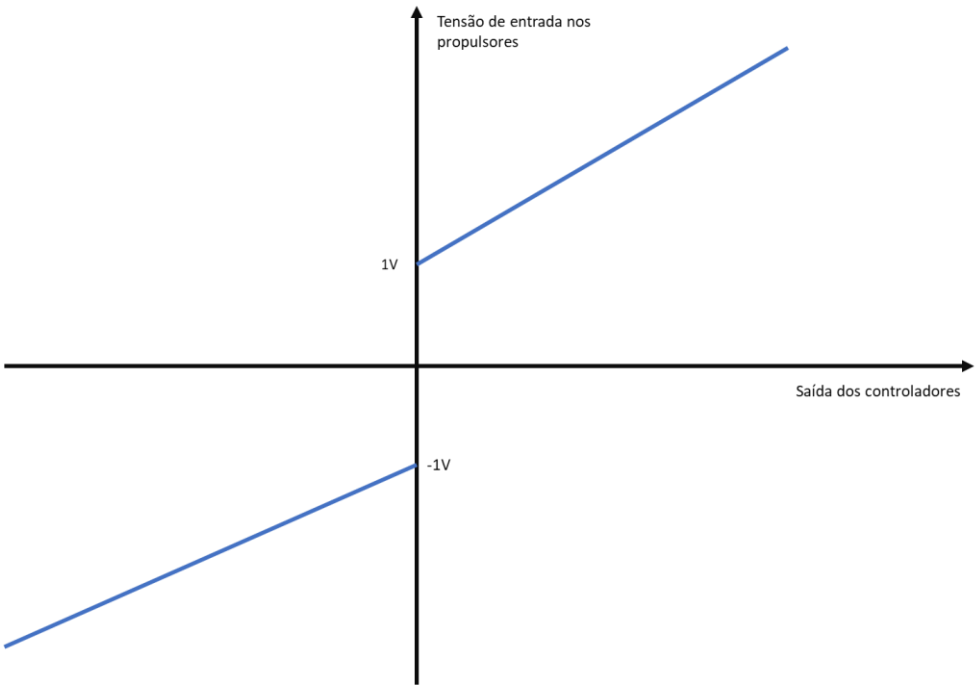


Figura 16 – Resposta dos controladores com o compensador.

5. Resultados experimentais

Neste capítulo é apresentado todo o procedimento experimental necessário para obtenção dos resultados que serão descritos neste trabalho.

Para a realização dos experimentos foram utilizados o sistema de captura de movimentos VICON, composto por três câmeras de precisão (Figura 17), e uma piscina com as dimensões de 3,20m x 1,64m x 0,58m, as câmeras foram posicionadas em três extremidades desta piscina, conforme ilustrado na Figura 18.

Figura 17 - Câmera de precisão do sistema VICON.

Figura 18 – Ambiente dos experimentos.

No Matlab foram utilizados dois *toolboxes*, um para aquisição de dados do sistema VICON e outra para comunicação *ZigBee*, o primeiro *toolbox* utilizado é o *Vicon DataStream SDK* na versão1.8.0 e o segundo é o *Communications System Toolbox Library for the ZigBee Protocol* na versão 18.1.0. Para interface do sistema VICON foi utilizado o *Tracker* na versão 3.5, este *software* é responsável por rastrear a embarcação e o alvo e disponibilizar suas coordenadas no sistema de coordenadas estático.

Depois da fabricação da placa e da montagem desta na embarcação, foram iniciados os testes para calibração do sistema e, em um primeiro momento, foram utilizados LED's em vez dos motores para a verificação do funcionamento da eletrônica montada, conforme ilustrado na Figura 19.

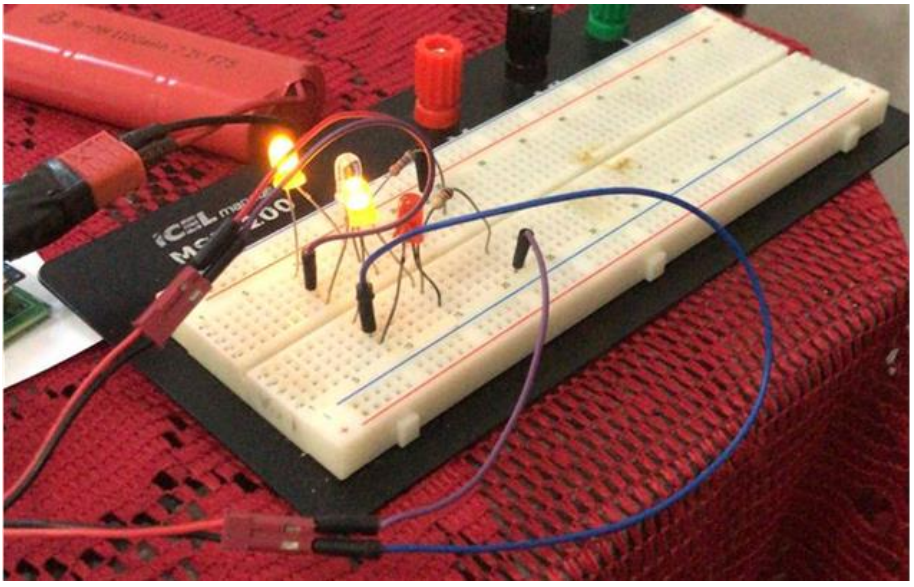


Figura 19 – Testes com os LED’s.

Após a verificação inicial do funcionamento da placa, foi iniciado o teste de sincronismo e configuração da rede *ZigBee*, tendo sido utilizado um programa simples no Arduino para apenas pegar um valor enviado diretamente do XCTU para a saída PWM conectada no LED.

Com a rede configurada e testada, foram implementados os controladores descritos no Capítulo 4 e, ainda com os LED’s conectados ao microcontrolador ao invés dos propulsores, foram realizados os testes iniciais do código descrito anteriormente, tanto a leitura de parâmetros enviados pelo microcomputador como o funcionamento dos controladores do tipo PD.

Subsequentemente ao funcionamento de todos testes anteriores, mas ainda antes de efetuarmos a introdução da piscina no experimento, foram realizados testes com a VICON, que é o sistema de monitoramento de alta precisão utilizado para posicionar a embarcação e o alvo. Nesse momento, foram introduzidos além da VICON, o rastreamento do alvo e os propulsores da embarcação, este teste foi realizado para verificação da resposta dos controladores, como por exemplo, se os propulsores trocariam o sentido da rotação quando alterássemos o alvo de posição.

Somente após a conclusão de todos esses passos é que foram iniciados os testes na água. Inicialmente, foram realizados testes apenas com o controlador de rumo, de forma a verificar seu perfeito funcionamento e calibração deste, por meio do método de aproximações sucessivas. Com a calibração deste controlador de forma satisfatória, foi incluído o controlador de distância. Após, com a realização da

interligação de todos os componentes necessários para o funcionamento completo do sistema, para alcançar os valores finais dos parâmetros utilizados nos controladores do tipo PD, demonstrados nas equações (10) e (11), foram realizados novos testes para a obtenção destes através de aproximações sucessivas.

Os valores dos ganhos relativos aos controladores automáticos obtidos nesses testes estão apresentados na Tabela 13, os valores dos ganhos foram obtidos por aproximações sucessivas, inicialmente foi alcançado um valor para o ganho proporcional que aproximava o barco da resposta ideal e o ajuste fino de sua resposta foi utilizada aumentado o tempo derivativo, para que o controlador ter uma resposta cada vez mais rápida a variação do erro, este procedimento foi realizado para os dois controladores.

Figura 20 – Montagem final da embarcação.

Tabela 13 – Ganhos/Parâmetros dos controladores.

Controlador	Ganho/ Parâmetro	Valor
Rumo	Kc_d (Ganho Proporcional)	3,5
Rumo	Td_d (Tempo Derivativo)	1,4
Distância	Kc_d (Ganho Proporcional)	2
Distância	Td_d (Tempo Derivativo)	1,8

Com os controladores ajustados e configurados, foram desenvolvidos três experimentos para comprovar o funcionamento deste sistema, sabendo-se que a diferença entre estes foi apenas o estado do alvo.

5.1. Experimento 1

O primeiro experimento foi realizado utilizando um alvo parado fora da água e com o barco praticamente apontado para este alvo. O resultado deste experimento será demonstrado abaixo, onde a Figura 21 apresenta a evolução do ângulo de rumo, a Figura 22 do controle de distância , e a Figura 23 demostra a trajetória da embarcação e a posição do alvo desejado.

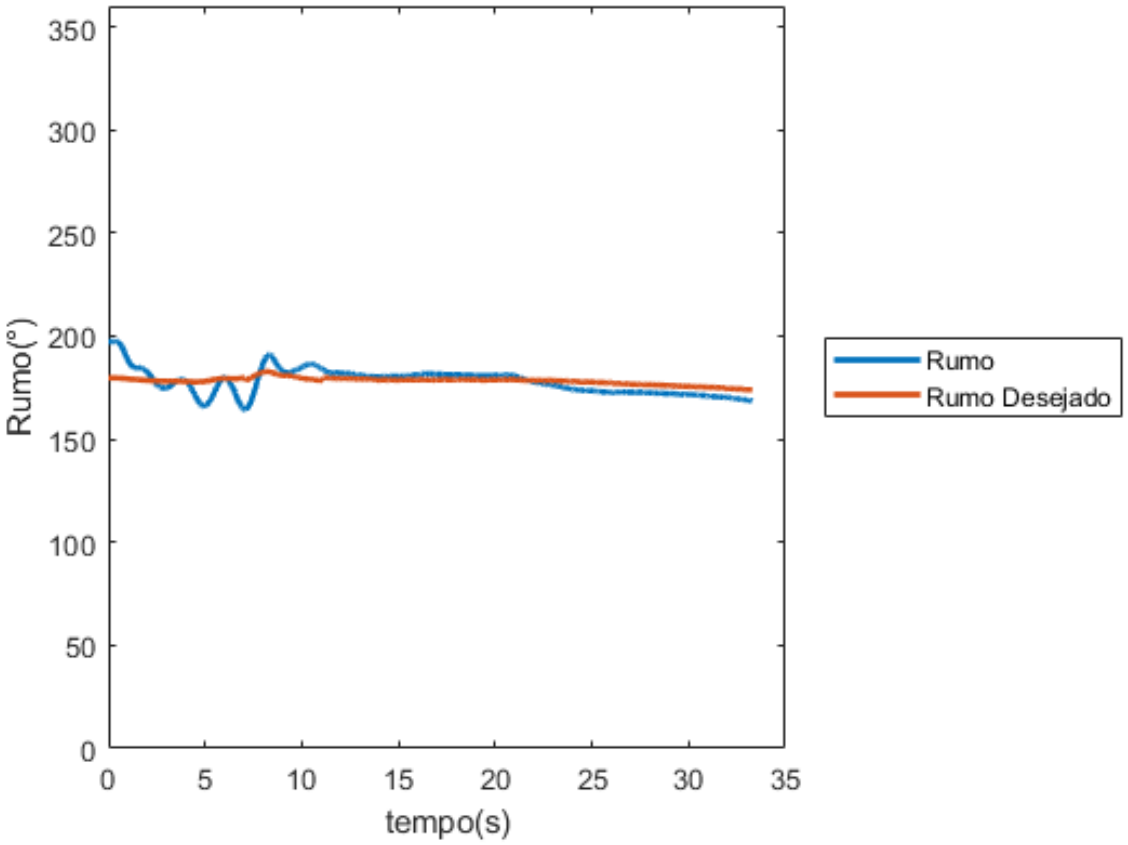


Figura 21 – Ângulo de rumo do barco no experimento 1.

Pode-se verificar na Figura 21 que com 10 segundos a embarcação já se encontrava com o rumo muito próximo do desejado, uma vez que o controlador foi ajustado para ter a resposta mais rápida possível, mas com pouco sobre sinal. Neste experimento, o controlador de distância estava ativo desde o início do experimento, sendo ele um dos responsáveis para que o rumo tenha demorado um pouco mais para estabilizar.

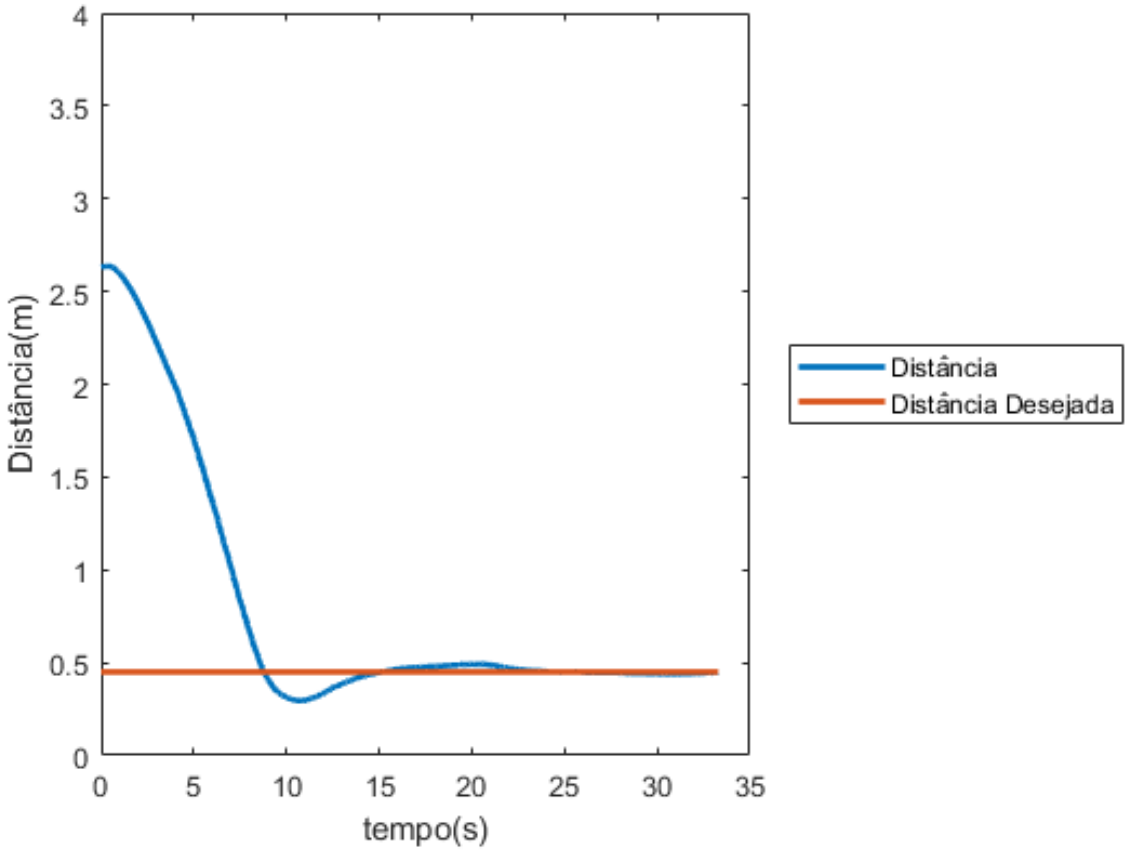


Figura 22 – Distância entre o barco e o alvo no experimento 1.

Podemos verificar também na Figura 22 que este controlador atinge se objetivo 15 segundos após este iniciar seu funcionamento. Por sua vez, este apresenta um pequeno sobre sinal, mas como foi definida uma distância de 45 centímetros entre o centro da embarcação e o centro do alvo como objetivo deste controlador, o sobre sinal que o controlador apresenta é aceitável uma vez que o barco não colide com o alvo.

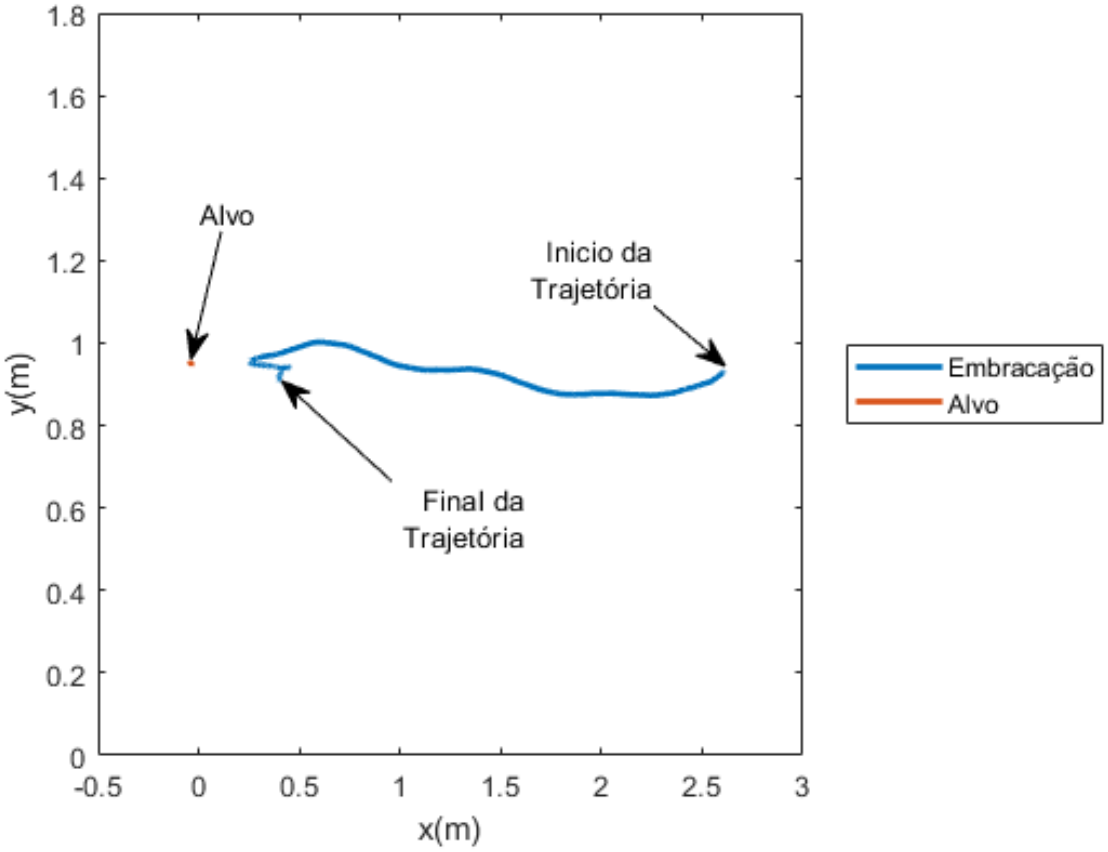


Figura 23 – Trajetória do barco no experimento 1.

Neste primeiro gráfico da trajetória, ver Figura 23, podemos verificar que após a aproximação da embarcação com o alvo, o barco tenta manter sua posição parada em relação ao alvo, que seria a resposta ideal do sistema. A resposta real do sistema nesse caso na Figura 22 é próxima da ideal e é possível verificar a atuação do controlador fazendo com que o posicionamento final da embarcação fique variando muito pouco entorno do ideal.

5.2. Experimento 2

A seguir, foi realizado o segundo experimento utilizando um alvo parado que se encontrava um pouco acima da lâmina d'água, mas desta vez com a embarcação posicionada para o lado oposto do alvo. O resultado obtido no experimento de número 2 está demonstrado a seguir seguindo a mesma disposição do primeiro experimento,

ou seja, a Figura 24 apresenta o controle de rumo, a Figura 25 o resultado do controle de distância, e a trajetória da embarcação e o alvo desejado é descrito na Figura 26.

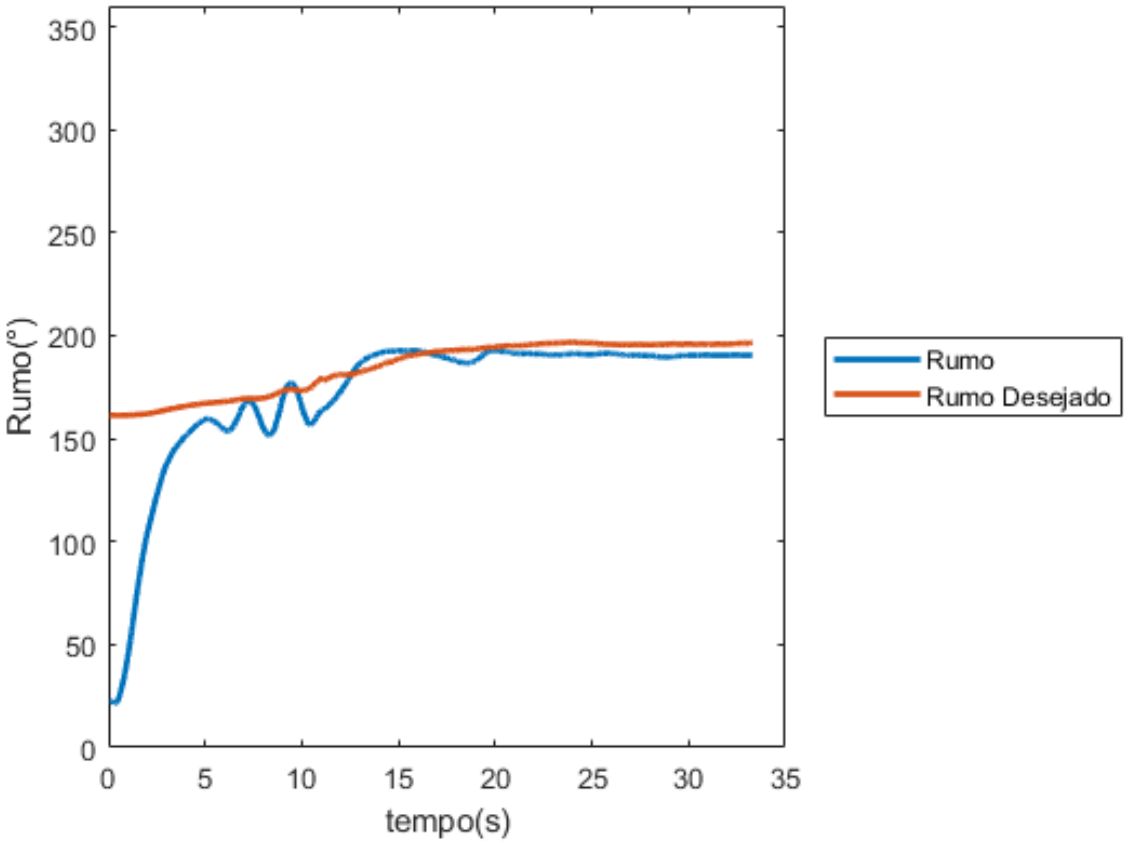


Figura 24 – Ângulo de rumo do barco no experimento 2.

Como pode-se verificar na Figura 24, em aproximadamente 15 segundos a embarcação já se encontrava com o rumo muito próximo do desejado da mesma maneira que no primeiro experimento. Também é possível verificar que o controlador de rumo diminui a velocidade quando passamos do ponto onde apenas o controlador de rumo está ativo, este ponto é causado pelo habilitador do controlador de distância anteriormente citado no Capitulo 4. Outro ponto a salientar é que pode-se perceber que resta um pequeno erro de regime, por volta de 5°.

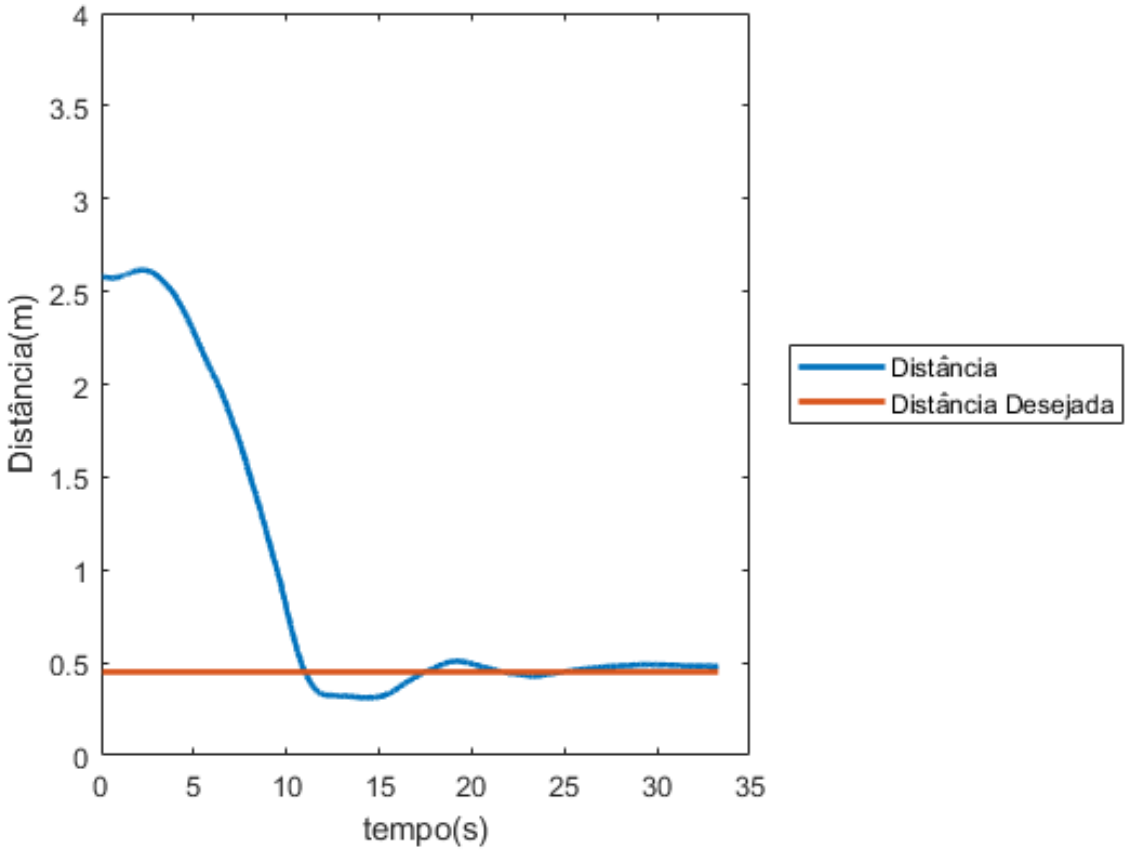


Figura 25 – Distância entre o barco e o alvo no experimento 2.

Na resposta da distância, Figura 25, podemos verificar que esta é muito similar com o experimento anterior, com a diferença de que o barco se afasta enquanto apenas o controlador de rumo está ativo e, assim que o controlador de distância começa a funcionar, o barco se aproxima rapidamente do objetivo.

É possível observar que a resposta desse controlador é possui menor oscilação que a resposta do controlador de rumo. Isto ocorre porque a ação do controlador PD do rumo é 4 vezes maior que a do controlador da distância, o que faz que a resposta seja um pouco mais suave. O erro de regime apresentado neste controlador também é pequeno e fica entorno de 3cm entre a distância desejada e a obtida.

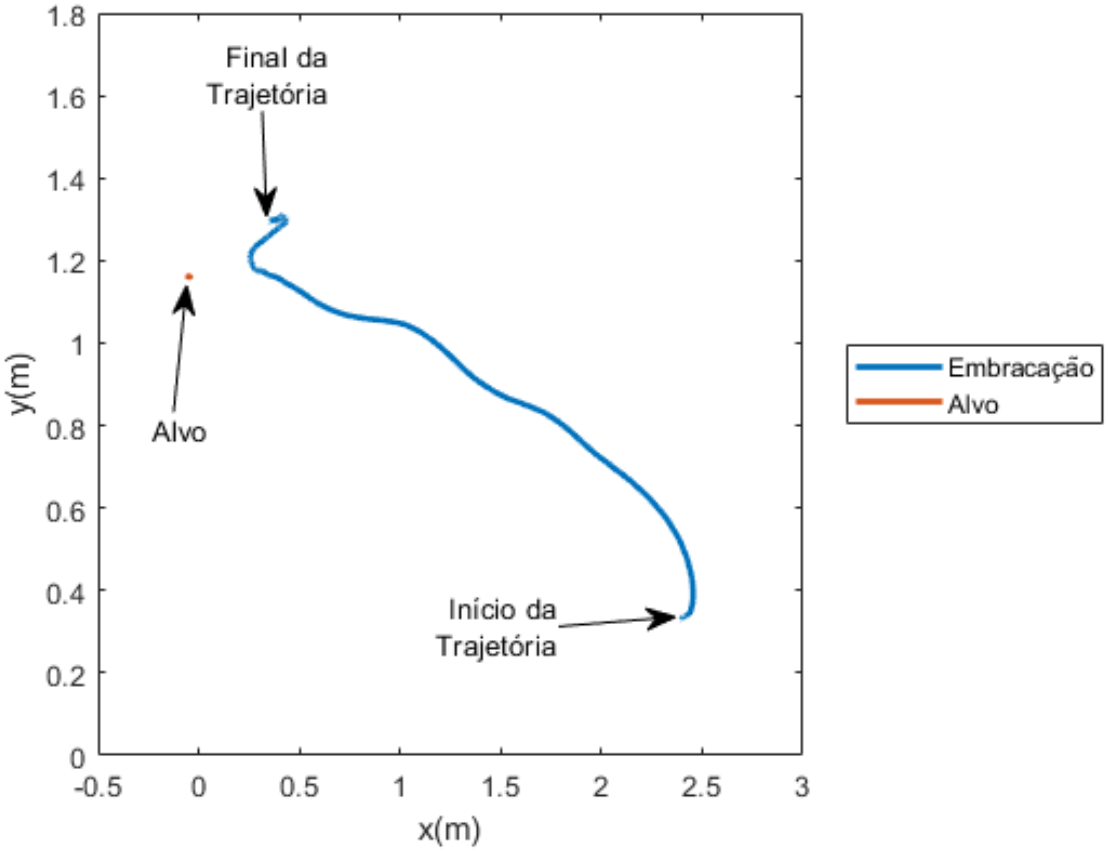


Figura 26 – Trajetória do barco no experimento 2.

No gráfico da trajetória (Figura 26) é possível verificar a aproximação da embarcação até o alvo, bem como a atuação dos controladores mantendo o barco apontando para o alvo, como se ela apresentasse um âncora virtual fixada na proa do veículo não tripulado.

5.3. Experimento 3

Por último, o terceiro experimento tinha como objetivo comprovar a resposta do sistema quando há um alvo com movimento variável, a fim de verificar a possibilidade da utilização do controle proposto em uma situação onde uma embarcação, por exemplo, pudesse seguir outro veículo.

O resultado alcançado nesse último experimento está descrito nas imagens dispostas a seguir de acordo com a seguinte distribuição: a Figura 27 para caracterizar

o controle de rumo, a Figura 28 com o controle de distância, e a trajetória da embarcação, a posição e o movimento do alvo desejado são descritas pela Figura 29.

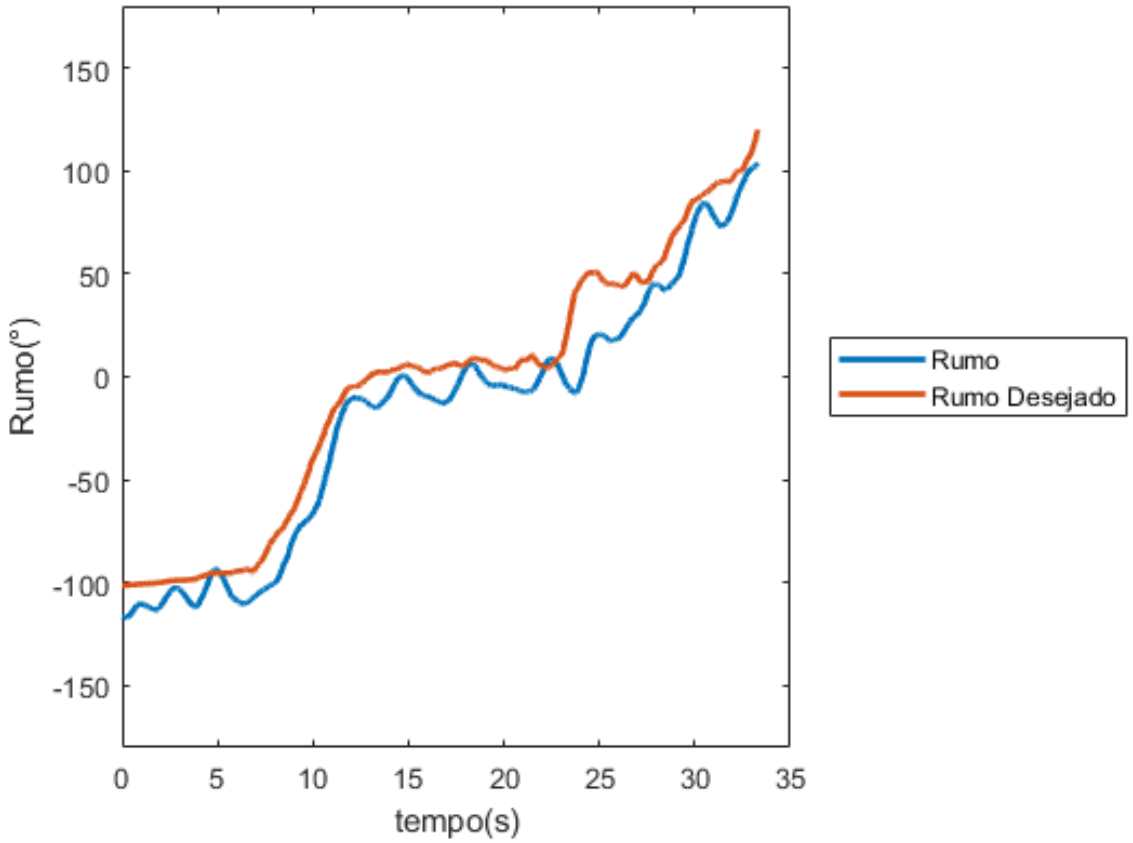


Figura 27 – Ângulo de rumo do barco no experimento 3.

Neste experimento, é exigido muito mais dos controladores uma vez que o alvo se movimenta, fazendo com que o objetivo fique sempre em mudança. No controlador de rumo, é possível observar através da Figura 25 que, desde a aproximação inicial da embarcação ao alvo, este consegue fazer com que o barco fique apontando sempre para o alvo com um atraso de frações de segundos, fazendo com que o barco consiga atingir sua meta.

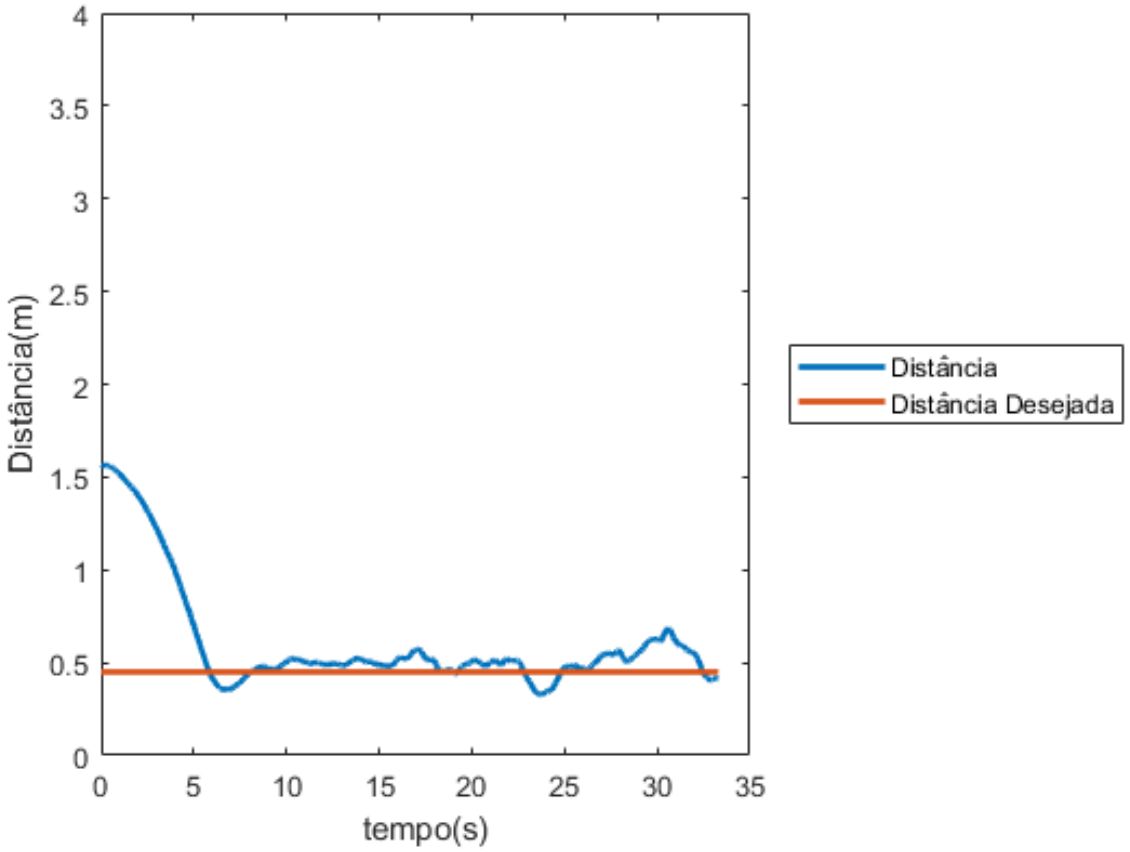


Figura 28 – Distância entre o barco e o alvo no experimento 3.

Na resposta da distância, é possível perceber que o movimento do alvo faz com que aconteça um aumento nos erro de regime que ocorria após a aproximação inicial da embarcação ao alvo, mas o controlador é capaz de manter esse erro muito pequeno, menor do que 5 centímetros, e, sobretudo, com quase nenhum sobressinal, evitando o choque com o alvo.

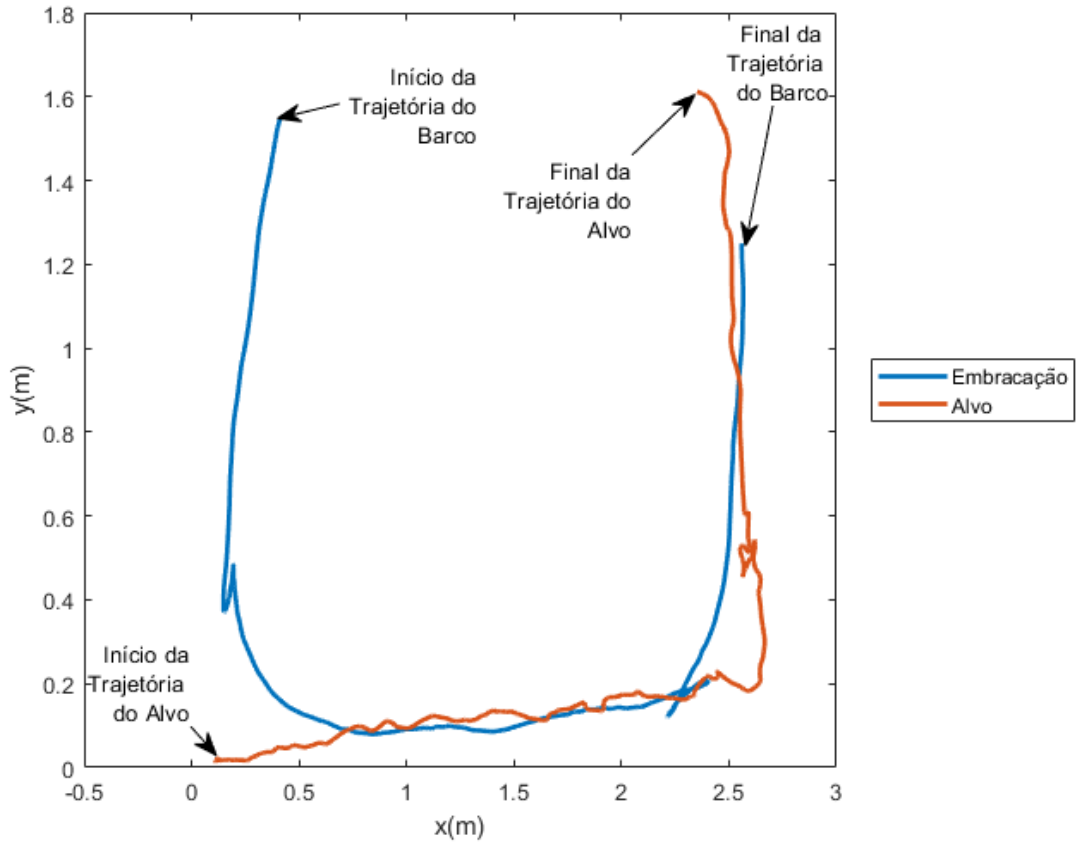


Figura 29 – Trajetória do barco no experimento 3.

Assim, pode-se observar que após a aproximação inicial, os controladores conseguem manter a embarcação com um percurso muito próximo da trajetória realizada pelo alvo. Por último, pode-se analisar que o movimento do alvo é muito mais tortuoso que a resposta da embarcação por se tratar de um movimento manual. Por outro lado, o controlador de rumo faz com que ele aponte para o ponto seguinte antes que a embarcação atinja o ponto onde a oscilação aconteceu, fazendo com que o veículo não tripulado faça o caminho mais curto e suave até o ponto final, e não necessariamente seguindo de forma idêntica a trajetória do alvo.

6. Conclusões

Para a implementação do sistema proposto neste projeto, foi necessária a construção de uma placa de circuito impresso que foi instalada no veículo não tripulado, ou seja, a embarcação, e a implementação de um controlador automático para que este pudesse comandar o deslocamento do barco. De forma a minimizar os erros, um sistema de monitoramento de alta precisão foi utilizado para posicionar o alvo e a embarcação no plano.

Após os experimentos realizados, foi possível concluir com os resultados demonstrados no Capítulo 5, que os controladores do tipo PD escolhidos para desenvolver o controle da embarcação tiveram um resultado satisfatório, principalmente o controlador de rumo, que apresentou um resultado mais preciso, com uma resposta rápida e sem sobressinal (“*overshoot*”).

Por outro lado, na resposta do controlador de distância, foi possível verificar um erro de regime apresentado na resposta do sistema, graças a retirada da ação integral do controlador PID e a resposta do motor na zona morta, além disso, de acordo com os gráficos de resposta desse controlador, fica explicito que as perturbações externas ao sistema, como movimentos residuais da água, afetam de forma mais direta no posicionamento final da embarcação, fazendo com que este controlador não consiga ficar estabilizado por muito tempo.

Para possibilitar o melhor controle do veículo não tripulado, algumas ações são sugeridas. A primeira delas é a troca das saídas PWMs selecionadas no Arduino, para que as quatro saídas apresentem a mesma frequência, uma vez que durante os experimentos foi possível verificar que mesmo que pouca, essa diferença nas frequências é capaz de alterar o resultado esperado.

A segunda ação sugerida é a implementação de um controlador diferente no controle de distância, capaz de responder mais rápido e com menor erro de regime, tentando, assim, permitir que o alvo não se distancie muito da embarcação.

Por fim, além das recomendações supracitadas, também é possível a inclusão de um segundo veículo não tripulado, para que o primeiro veículo siga uma trajetória ou alvo, e o segundo siga a primeira embarcação, o que demonstra a possibilidade de

utilização deste tipo de sistema no controle de uma frota e não apenas de um barco isolado.

Referências

AMARAL, G. de S. *Sistema de Posicionamento Dinâmico para um Pequeno Veículo Flutuante*. Projeto (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2008.

BUORO, A. S. *Controle dos motores e acionamento sem fios de uma pequena embarcação*. Projeto (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

CASTRUCCI, P. de L.; BITTAR, A.; SALES, R.M. *Controle Automático*. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 2011.

CUNHA, J. P.V.S. *Projeto e Estudos de Simulação de um Sistema de Controle a Estrutura Variável de um Veículo Submarino de Operação Remota*, Tese de Mestrado em Engenharia, Programa de Pós-Graduação de Engenharia Elétrica, COPPE/UFRJ, Rio de Janeiro, 1992.

DIGI INTERNATIONAL, *DIGI ZigBee RF Modules User Guide*, Product Manual XBee, Xbee-Pro2, Xbee-ProS2, 2018.

DORF, R.C; BISHOP, R.H. *Sistemas de Controle Modernos*. Tradução de Prof. Bernardo Severo. 8ª ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 2001.

FOSSEN, T. I. *Nonlinear Modeling and Control of Underwater Vehicles.* Tese (PhD), Norwegian University of Science and Technology, Trondheim, 1991.

FOSSEN, T. I. *Marine Control Systems: Guidance, Navigation, and Control of Ships, Rigs and Underwater Vehicles*. Norway: Marine Cybernetics AS, 2002.

GUERRA. P. H. L. *Carros autônomos: como funcionam e quais seus benefícios.* 2017. Disponível em: <https://educacaoautomotiva.com/2017/07/09/carros-autonomos-como-funciona-beneficios/>. Acesso em 04.jun.2019.

LEE, J.; Su Y; Shen C., *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*, The 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON), Taiwan, 2007.

LORENZETTI. L. *Disney Cruises Is Expanding Its Fleet with Massive New Ships.* 2016. Disponível em: < https://fortune.com/2016/03/03/disney-cruise-expansion/ >. Acesso em 09.nov.2019.

NEVES, J. T; BASTOS, M. P. *Development of a system of monitoring the positioning of a mobile robot for collection of variables in non-structured via ZigBee.* Journal of Engineering and Technology for Industrial Applications. ed. 08.vol: 02, Amazonas, 2016.

ROSÁRIO, R. V. de C. *Controle a Estrutura Variável de um Barco Empurrando uma Carga Flutuante Subatuada*. Tese (Mestrado em Ciências), Programa de Pós-Graduação de Engenharia Eletrônica – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2017.

ROSÁRIO, R. V. de C; CUNHA, J. P.V.S. *Experimentos de Rastreamento de Trajetória de uma Embarcação de Superfície Utilizando Linearização por Realimentação e Controle a Estrutura Variável*. XXI Congresso Brasileiro de Automática, Vitória, Espirito Santo, Outubro 2016.

OGATA, K. *Engenharia de Controle Moderno*. Tradução de Prof. Bernardo Severo. 3ª ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora S.A., 2000.

RAPPAPORT, T. S., *Comunicação sem fio princípios e práticas*, 2ª Edição, Tradução Daniel Vieira, Pearson Education do Brasil Ltda, 2009

RAMOS, J. de S. B., *Sistema digital de instrumentação sem fio de uma mola helicoidal TI-NI usando tecnologia ZigBee, usb e Linux*, Dissertação de mestrado, Universidade Federal de Pernambuco, 2010.

SOKAL, G. J. *Posicionamento Dinâmico Utilizando Controle a Estrutura Variável e Servo visão***.** 121 p. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Julho 2010.

SOUSA, L. R. de., *Acionamento dos motores CC de uma embarcação tele operada*. Projeto (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2016.

TENNINA, S.; et al. *IEEE 802.15.4 and ZigBee as Enabling Technologies for Low-Power Wireless Systems with Quality-of-Service Constraints*, Springer, 2013.

VICON. *System Reference*. 1.4. ed. [S.l.], 2006.

WOLF, A. S.; et al. *VEÍCULO TERRESTRE NÃO TRIPULADO CONTROLADO VIA REDE WI-FI.* Revista destaques acadêmicos - CETEC/UNIVATES, vol. 7, n. 4, Rio Grande do Sul, 2015.

ZIGBEE. *ZigBee specification: ZigBee document 053474r13*. [S.I.], 2006.

Apêndice A – Código Arduino

Tabela 14 - Programa do Arduino completo.

Programa do Arduino

//Autor: Victor Elias de Sousa da Silva
//Engenharia Elétrica - Sistemas Eletrônicos
//Data: 06 de Novembro de 2019
//Professor Orientador: Professor José Paulo Vilela Soares da Cunha
 	
int x_barco1=0;
int y_barco1=0;
int x_ref1=0;
int y_ref1=0;
float x_barco=0;
float y_barco=0;
float x_ref=0;
float y_ref=0;
float teta=0;
float d=0;
bool h=LOW; //valor para habilitadores dos motores
int test=0; // valor adicionado para teste de habilitador
float d_ant=0;
float Psi_ant=0;
float U_d=0;
float U_Psi=0;
float PD_d=0;
float PD_Psi=0;
int U1=0;
int U2=0;
float erro_d=0;
float erro_Psi=0;
int Psi_Ref1=0;
float Psi_Ref=0;
const float Freq = 30 ; // Frequencia entre quadros do ZigBee
const float d_Ref=450; // Tamanho do barco para evitar a singularidade do calculo do ângulo
const float zm=45; // Constante para compensar a zona morta do motor
const float Duty = 1; // seleção do Duty-Cycle para o PWM valor máximo de 255
const int HA = 7; // Habilita ponte H para o motor A

```
 const int A_pos = 5; // Porta saída positiva para o motor A
 const int A_neg = 6; // Porta saída negativa para o motor A
 const int HB = 8; // Habilita ponte H para o motor B
 const int B_pos = 11; // Porta saída positiva para o motor B
 const int B_neg = 10; // Porta saída negativa para o motor B
 const float ang_lim=90; // angulo limite para controle de rumo
 int Kc_d1=0; // Ganho Proporcional para a distancia
 int Kc_Psi1=0; // Ganho Proporcional para o rumo
 int Td_d1=0;// Tempo derivativo para a distancia
 int Td_Psi1=0; // Tempo derivativo para o rumo
 float Kc_d=0; // Ganho Proporcional para a distancia
 float Kc_Psi=0; // Ganho Proporcional para o rumo
 float Td_d=0;// Tempo derivativo para a distancia
 float Td_Psi=0; // Tempo derivativo para o rumo
 const float pi = 3.1415926 ; // Constante pi
  #include <math.h>
  #include <XBee.h>
  XBee xbee = XBee();
  //=============================================Endereço Xbee Coordenador
  uint8_t payload[] = {0,0,0,0,0,0,0,0,0,0};// Array payload dados a serem enviados
  //=== SH + SL endereço do XBEE destino
  XBeeAddress64 addr64 = XBeeAddress64(0x0013a200,0x4098bf1e);
  ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));
  ZBTxStatusResponse txStatus = ZBTxStatusResponse();
  XBeeResponse response = XBeeResponse();//= cria objeto de respota reusavel
  ZBRxResponse rx = ZBRxResponse();
  ModemStatusResponse msr = ModemStatusResponse();
//=========================================================SETUP



void setup() {
  // put your setup code here, to run once:


  TCCR0B = TCCR0B & B11111000 | B00000010;  // Troca a frequência do PWM das portas D5 e
D6 para 7812.50 Hz

  TCCR1B = TCCR1B & B11111000 | B00000010;  // Troca a frequência do PWM das portas D9 e
D10 para 3921.26 Hz
```

```
  TCCR2B = TCCR2B & B11111000 | B00000010;  // Troca a frequência do PWM das portas D3 e D11 para 3921.26 Hz

    Serial.begin(9600); // Inicia porta serial à 9600 bps:
    xbee.begin(Serial);

    pinMode(A_pos,OUTPUT);
    pinMode(A_neg,OUTPUT);
    pinMode(B_pos,OUTPUT);
    pinMode(B_neg,OUTPUT);
    pinMode(HA,OUTPUT);
    pinMode(HB,OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:



 xbee.readPacket(); //=== Ler continuamente algum pacote ZigBee (ZB)

 if (xbee.getResponse().isAvailable()) //==recebeu algo
 {
   if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)//== É um pacote ZB (API ID 0x90 - received packet)
     {
       xbee.getResponse().getZBRxResponse(rx); //==Aloca os dados recebidos na calsse rx
        if (rx.getOption() == ZB_PACKET_ACKNOWLEDGED)
        {

          d_ant = erro_d; //armarzena o valor do erro da distancia para calculo da velocidade

          Psi_ant = erro_Psi; //armarzena o valor do erro do rumo para calculo da velocidade



          x_barco1 = combina(rx.getData(0),rx.getData(1));//armazena a coordenada x do barco
          x_barco = x_barco1/10; //desloca a virgula uma casa decimal para a direita para aumentar a sensibilidade do sistema
```

```
        y_barco1 = combina(rx.getData(2),rx.getData(3));// armazena a coordenada y do barco
        y_barco = y_barco1/10; //desloca a virgula uma casa decimal para a direita para aumentar a sensibilidade do sistema
        Psi_Ref1 = combina(rx.getData(4),rx.getData(5));//armazena o rumo atual
        Psi_Ref = Psi_Ref1/10; //desloca a virgula uma casa decimal para a direita para aumentar a sensibilidade do sistema
        x_ref1 = combina(rx.getData(6),rx.getData(7));//armazena a coordenada x do destino
        x_ref = x_ref1/10; //desloca a virgula uma casa decimal para a direita para aumentar a sensibilidade do sistema
        y_ref1 = combina(rx.getData(8),rx.getData(9));// armazena a coordenada y do destino
        y_ref = y_ref1/10; //desloca a virgula uma casa decimal para a direita para aumentar a sensibilidade do sistema
        //Aloca dados do controlador da distancia
        Kc_d1 = combina(rx.getData(10),rx.getData(11));// armazena o ganho proporcional para distancia
        Kc_d = Kc_d1/100; //desloca a virgula duas casas decimais para a direita.
        Td_d1 = combina(rx.getData(12),rx.getData(13));// armazena o tempo derivativo para distancia
        Td_d = Td_d1/100; //desloca a virgula duas casas decimais para a direita.
        //Aloca dados do controlador do rumo
        Kc_Psi1 = combina(rx.getData(14),rx.getData(15));// armazena o ganho proporcional para o rumo
        Kc_Psi = Kc_Psi1/100; //desloca a virgula duas casas decimais para a direita.
        Td_Psi1 = combina(rx.getData(16),rx.getData(17));// armazena o tempo derivativo para o rumo
        Td_Psi = Td_Psi1/100; //desloca a virgula duas casas decimais para a direita.

        h=HIGH; // Habilita os motores
        test=200000; // valor de vezes que o loop roda sem receber dados com os motores habilitados (10000 aproximadamente 3s)

        }
    }
}

         test--;

         d = sqrt(((x_ref-x_barco)*(x_ref-x_barco))+((y_ref-y_barco)*(y_ref-y_barco)));

         teta = ((atan2((y_ref-y_barco),(x_ref-x_barco)))*180)/pi;

         erro_d=d-d_Ref; //Erro na distancia
```

```
        erro_Psi=Psi_Ref-teta; //Erro na orientação

        if (erro_Psi>180)       //garante que o barco sempre gire em relação ao menor ângulo de diferença
             erro_Psi=erro_Psi-360;
        else if (erro_Psi<(-180))
             erro_Psi=erro_Psi+360;
        else erro_Psi=erro_Psi;

          PD_d = PD_barco(Kc_d, Td_d, Freq, erro_d, d_ant);

          if (PD_d>(255.0-zm))
             U_d=(255.0-zm);
          else if(PD_d<(-255.0+zm))
             U_d=(-255+zm);
          else U_d=PD_d;


          PD_Psi = PD_barco(Kc_Psi, Td_Psi, Freq, erro_Psi, Psi_ant);

          if (PD_Psi>(255.0-zm))
             U_Psi=(255.0-zm);
          else if(PD_Psi<(-255.0+zm))
             U_Psi=(-255+zm);
          else U_Psi=PD_Psi;


//verifica se a diferenfa do rumo e o angulo da trajetória é menor que o angulo limite pré-definido
          if (erro_Psi>ang_lim||erro_Psi<(-ang_lim)){

             U1 = (U_Psi)*Duty;     // Sinal para ponte H do motor 1
             if (U1>=0){
                U1=U1+zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,U1);analogWrite(A_neg,0);}
             else {
                U1=U1-zm;
                digitalWrite(HA,h);
                analogWrite(A_pos,0);analogWrite(A_neg,(-U1));}
```

```
            U2 = (-U_Psi)*Duty;     // Sinal para ponte H do motor 2
            if (U2>=0){
               U2=U2+zm;
               digitalWrite(HB,h);
               analogWrite(B_pos,U2);analogWrite(B_neg,0);}
            else {
               U2=U2-zm;
               digitalWrite(HB,h);
               analogWrite(B_pos,0);analogWrite(B_neg,(-U2)); }
         }

         else{
            U_d=U_d/5*Duty;
            U_Psi=4*U_Psi/5*Duty;

            U1 = round(U_d+U_Psi);     // Sinal para ponte H do motor 1
            if (U1>=0){
               U1=U1+zm;
               digitalWrite(HA,h);
               analogWrite(A_pos,U1);analogWrite(A_neg,0);}
            else {
               U1=U1-zm;
               digitalWrite(HA,h);
               analogWrite(A_pos,0);analogWrite(A_neg,(-U1));}
            U2 = round(U_d+(-U_Psi));     // Sinal para ponte H do motor 2
            if (U2>=0){
               U2=U2+zm;
               digitalWrite(HB,h);
               analogWrite(B_pos,U2);analogWrite(B_neg,0);}
            else {
               U2=U2-zm;
               digitalWrite(HB,h);
               analogWrite(B_pos,0);analogWrite(B_neg,(-U2)); }
         }

         if (test==0){
            h=LOW;
         }

}
```

```
//================================================ função que extrai 10 bits do Payload
int combina(unsigned int x_high, unsigned int x_low)
{
  int combined;
  combined = x_high;
  combined = combined*256;
  combined |= x_low;
  return combined;
}
//================================================implementação para o PD
float PD_barco(float Kc, float Td, float T, float erro, float erro_ant)
{
float controle;
float veloc;
float P;
float D;

P= Kc*erro;

veloc= (erro-erro_ant)*T;

D= Kc*Td*veloc;

controle = P + D;

return controle;

}
```

Apêndice B – Códigos Matlab

Tabela 15 – Programa Principal do Matlab.

Programa do Matlab – Programa Principal

```
% Configuração e disponibilização da porta serial para comunicação
ZigBee:

delete(instrfindall); %Solved unavailable port error
MyPort =
serial('COM6');%'baudrate',9600,'databits',8,'parity','none','stopbits',1
,'readasyncmode','continuous');

Kc_d=4; % valor do ganho proporcional para a distância
Td_d=1.2; % 1 valor do tempo derivativo para a distância
Kc_Psi=-8; % 7 valor do ganho proporcional para o rumo
Td_Psi=1.5; % valor do tempo derivativo para o rumo

% calculo para poder enviar paramentros dos controladores como inteiros

Kc_d=Kc_d*100;
Td_d=Td_d*100;
Kc_Psi=Kc_Psi*100;
Td_Psi=Td_Psi*100;


fopen(MyPort);

% Frequencia e periodo de amostragem:

fs = 30; % Frequencia de amostragem (Hz).

h = 1/fs; % Periodo de amostragem (s).

% Numero de frames a serem aquisitados no experimento:

F_final = 3000;

% Inicializacao do contador de frames:

F = 0;

% Limpa dados armazenados:

 dados_armazenados = [];

% Inicializacao Vicon:

 Configuracao_Vicon;

% Tolerância do contador de voltas
tolerancia_contador_voltas = 3;
```

```
% Número de voltas
n = 0;
 % 90°
noventa = pi / 2;
% 270°
duzentos_e_setenta = 3 * pi / 2;
% 360°
trezentos_e_sessenta = 2 * pi;

% Inicializa o relógio
tic;
while F < F_final
    %%
    % Obter quadro
    while MyClient.GetFrame().Result.Value ~= Result.Success
    end% while
    % Obter número do quadro
    ObterNumeroFrame = MyClient.GetFrameNumber();
    % Tempo
    tempo_vicon = double(ObterNumeroFrame.FrameNumber) * h;
    % Tempo no primeiro quadro
    if F == 0
        tempo_0 = tempo_vicon;
    end
    % Tempo começando do zero
    tempo = tempo_vicon - tempo_0;
    % Inicializa
    dados = zeros(1, 10);
    % Armazena o tempo
    dados(1) = tempo;
    % Conta o número de objetos
    SubjectCount = MyClient.GetSubjectCount().SubjectCount;
    % Percorre todos os objetos
    for SubjectIndex = 1:SubjectCount;
        % Obtem o nome do objeto
        SubjectName = MyClient.GetSubjectName(SubjectIndex).SubjectName;
        % Conta o número de segmentos
        SegmentCount =
MyClient.GetSegmentCount(SubjectName).SegmentCount;
        % Percorre todos os segmentos
        for SegmentIndex = 1:SegmentCount;
            % Obtem o nome do segmento
            SegmentName = MyClient.GetSegmentName(SubjectName,
SegmentIndex).SegmentName;
            % Obtem as posições lineares do objeto
            TranslacaoSegmentosGlobais =
MyClient.GetSegmentGlobalTranslation(SubjectName, SegmentName);
            % Posição em x convertida para o sistema de coordenadas com

            if strcmp(SegmentName, 'BarcoFV')%incluir entre aspas o nome
do objeto que será controlado

                % Posição em x convertida para o sistema de coordenadas
com
                % visão superior
                x_obj1 = TranslacaoSegmentosGlobais.Translation(2)*10;% /
1000;
                % Arredondar o valor
                x_obj1= round(x_obj1);
```

```
                % Posição em y convertida para o sistema de coordenadas
com
                % visão superior
                y_obj1 = TranslacaoSegmentosGlobais.Translation(1)*10;% /
1000;
                % Arredondar o valor
                y_obj1= round(y_obj1);

                % Obtem as posições angulares do objeto
                RotacaoEulerSegmentosGlobais =
MyClient.GetSegmentGlobalRotationEulerXYZ(SubjectName, SegmentName);
                % ângulo de rumo convertido para o sistema de coordenadas
com
                % visão superior
                psi_obj1_vicon = noventa -
RotacaoEulerSegmentosGlobais.Rotation(3);
                %transformar de radianos para graus
                psi_obj1_vicon= ((psi_obj1_vicon*180)/pi);

                psi_obj1_vicon=(psi_obj1_vicon)*10; %corrigi o rumo do
braco para apontar para a frente.

                % Arredondar o valor
                psi_obj1_vicon= round(psi_obj1_vicon);

                        dados = [1,...
                   x_obj1,         ...
                   y_obj1,         ...
                   psi_obj1_vicon];

                    dados_aux =  [1,...
                   x_obj1/10000,         ...
                   y_obj1/10000,         ...
                   psi_obj1_vicon/10];

            end


            if strcmp(SegmentName, 'disco')% Incluir entre aspas o nome
do objeto que será seguido
                % disco
                % Posição em x convertida para o sistema de coordenadas
com
                % visão superior
                x_obj2 = TranslacaoSegmentosGlobais.Translation(2)*10;% /
1000;

                % Arredondar o valor
                x_obj2= round(x_obj2);

                % Posição em y convertida para o sistema de coordenadas
com
                % visão superior
                y_obj2 = TranslacaoSegmentosGlobais.Translation(1)*10;% /
1000;

                % Arredondar o valor
                y_obj2= round(y_obj2);
```

```
                % Obtem as posições angulares do objeto
                RotacaoEulerSegmentosGlobais =
MyClient.GetSegmentGlobalRotationEulerXYZ(SubjectName, SegmentName);
                % ângulo de rumo convertido para o sistema de coordenadas
com
                % visão superior
                psi_obj2_vicon = noventa -
RotacaoEulerSegmentosGlobais.Rotation(3);
                %transformar de radianos para graus
                psi_obj2_vicon= ((psi_obj2_vicon*180)/pi)*10;

                % Arredondar o valor
                psi_obj2_vicon= round(psi_obj2_vicon);

            % end

          dados = [dados,2,...
                   x_obj2,        ...
                   y_obj2,        ...
                   psi_obj2_vicon];

          dados_aux =  [dados_aux,2,...
                   x_obj2/10000,        ...
                   y_obj2/10000,        ...
                   psi_obj2_vicon/10];

             end
         end
        end

    % Armazena os dados atuais
    dados_armazenados(F + 1, :) = dados_aux%;

    %%
    %envio pelo ZigBee
    teste = communication_xbee(x_obj1, y_obj1, psi_obj1_vicon , x_obj2,
y_obj2, Kc_d, Td_d, Kc_Psi, Td_Psi, MyPort);

% Incrementa um quadro
F = F + 1;
end

%deliga os motores

Kc_d=0;  % valor do ganho proporcional para a distância
Td_d=0; % valor do tempo derivativo para a distância
Kc_Psi=0; % valor do ganho proporcional para o rumo
Td_Psi=0; % valor do tempo derivativo para o rumo

    teste = communication_xbee(x_obj1, y_obj1, psi_obj1_vicon , x_obj2,
y_obj2, Kc_d, Td_d, Kc_Psi, Td_Psi, MyPort);


fclose(MyPort); %Disconnect port

% Finaliza o relógio
toc
```

Tabela 16 – Programa da comunicação *ZigBee* no Matlab.

Programa do Matlab – Programa da comunicação *ZigBee*

```
function [APIframe] = communication_xbee(x1, y1, rumo1, x2, y2, Kc_d,
Td_d, Kc_Psi, Td_Psi, MyPort)

x_barco=comp2bits16(x1);
y_barco=comp2bits16(y1);
rumo_barco=comp2bits16(rumo1);
x_ref=comp2bits16(x2);
y_ref=comp2bits16(y2);
Kc_1=comp2bits16(Kc_d);
Td_1=comp2bits16(Td_d);
Kc_2=comp2bits16(Kc_Psi);
Td_2=comp2bits16(Td_Psi);

x_barco=bin2dec(x_barco);
x_barco=dec2hex(x_barco,4);
[x_barco_h,x_barco_l]=div8bits(x_barco);

y_barco=bin2dec(y_barco);
y_barco=dec2hex(y_barco,4);
[y_barco_h,y_barco_l]=div8bits(y_barco);

rumo_barco=bin2dec(rumo_barco);
rumo_barco=dec2hex(rumo_barco,4);
[rumo_barco_h,rumo_barco_l]=div8bits(rumo_barco);

x_ref=bin2dec(x_ref);
x_ref=dec2hex(x_ref,4);
[x_ref_h,x_ref_l]=div8bits(x_ref);

y_ref=bin2dec(y_ref);
y_ref=dec2hex(y_ref,4);
[y_ref_h,y_ref_l]=div8bits(y_ref);

Kc_1=bin2dec(Kc_1);
Kc_1=dec2hex(Kc_1,4);
[Kc_1_h,Kc_1_l]=div8bits(Kc_1);

Td_1=bin2dec(Td_1);
Td_1=dec2hex(Td_1,4);
[Td_1_h,Td_1_l]=div8bits(Td_1);

Kc_2=bin2dec(Kc_2);
Kc_2=dec2hex(Kc_2,4);
[Kc_2_h,Kc_2_l]=div8bits(Kc_2);

Td_2=bin2dec(Td_2);
Td_2=dec2hex(Td_2,4);
[Td_2_h,Td_2_l]=div8bits(Td_2);

APIframe(1,:) = '7E'; % Frame Class Name for reading only
APIframe(2,:) = '00';  % length 1
APIframe(3,:) = '20'; % length 2
APIframe(4:5,:) = ['10';'01'];   % type & ID
```

```
APIframe(6:13,:) = ['00'; '13'; 'A2'; '00'; '40'; '63'; 'D2'; '2F'];
%address
APIframe(14:15,:) = ['FF'; 'FE']; %address 16BIT
APIframe(16,:) = '00';
APIframe(17,:) = '00';
APIframe(18:35,:) = [ x_barco_h ; x_barco_l ; y_barco_h ; y_barco_l ;
rumo_barco_h ; rumo_barco_l ;
                      x_ref_h ; x_ref_l ; y_ref_h ; y_ref_l; Kc_1_h;
Kc_1_l; Td_1_h; Td_1_l; Kc_2_h;
                      Kc_2_l; Td_2_h; Td_2_l];% APIframe;
checksumframe = hex2dec(APIframe);
checksum = dec2hex(hex2dec('FF') - (mod((sum(checksumframe(4:35))),
256)));
APIframe(36,:) = checksum;
APIframe = hex2dec(APIframe);
fwrite(MyPort, APIframe, 'uint8')
%fclose(MyPort); %Disconnect port
end
function [a] = comp2bits16 (b)
    if b<0
        a=bitcmp(b,'int16');
        c=65535-(abs(b))-a;
        a=a+c;
        a=a+1;
        a=dec2bin(a,16);
    else
        a=dec2bin(b,16);
    end
end
function [bh,bl] = div8bits (a)
b=hex2dec(a);

bh=bitshift(b,-8);

bl=bitshift(bh,8);
bl=b-bl;

b=dec2bin(b,16);

bh=dec2bin(bh,8);

bl=dec2bin(bl,8);

bh=bin2dec(bh);
bh=dec2hex(bh,2);

bl=bin2dec(bl);
bl=dec2hex(bl,2);
end
```

Apêndice C – Montagem eletrônica

Neste apêndice são descritos toda a montagem realizada na embarcação utilizada nos experimentos.

A. Placa de circuito impresso.

A placa de circuito impresso utilizada na embarcação foi desenhada e projetada de acordo com o projeto inicial demonstrado na Figura 30, já nas Figura 31 e Figura 32 são apresentados as vistas da placa fabricada.

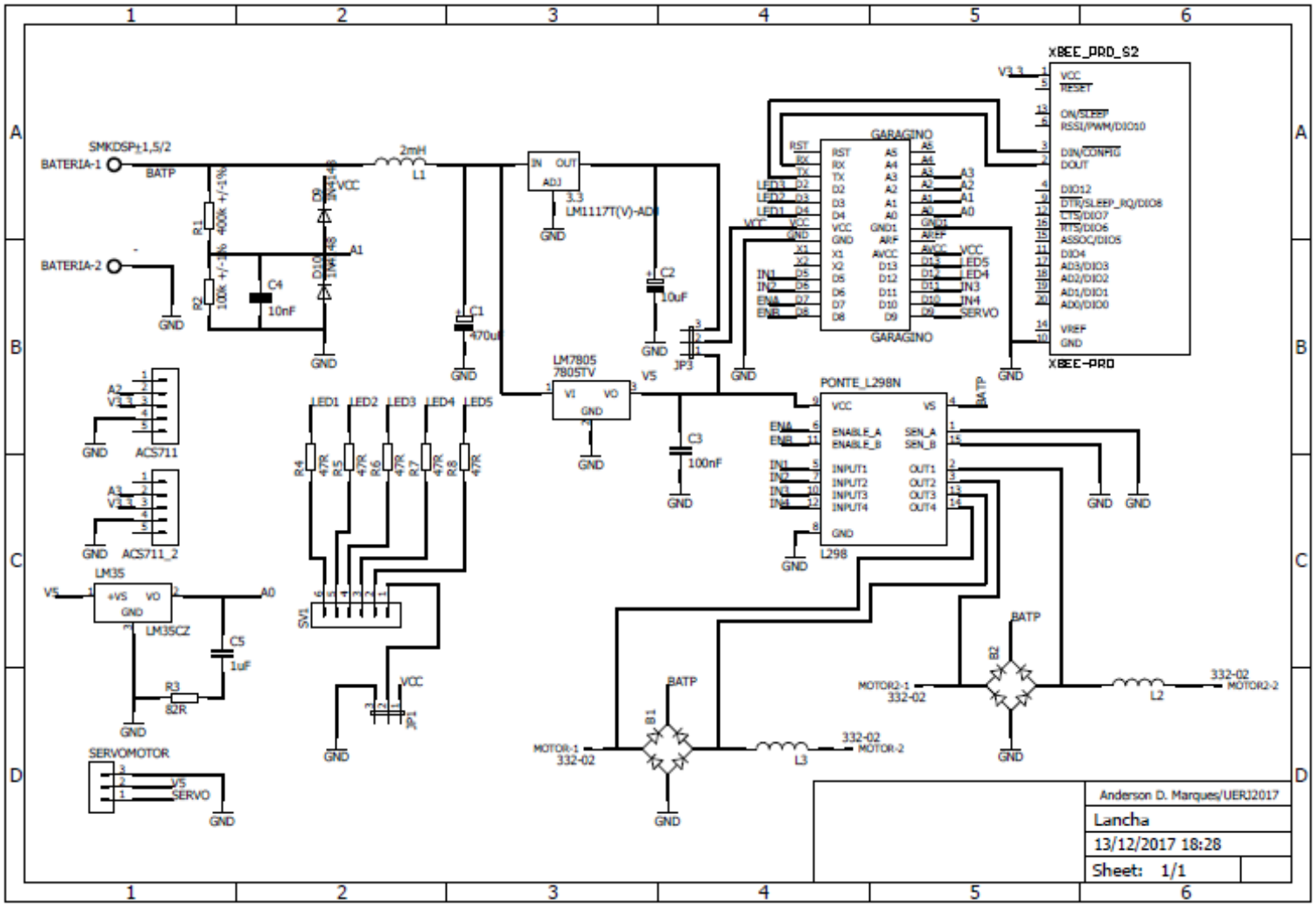


Figura 30 – Projeto da placa.

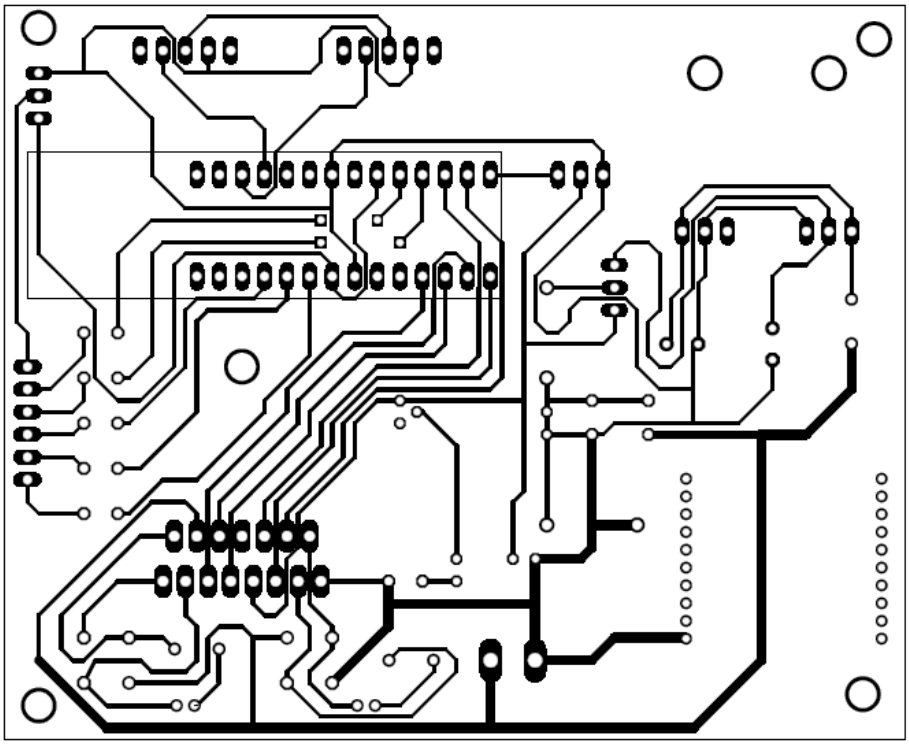


Figura 31 – Vista inferior da placa da embarcação.

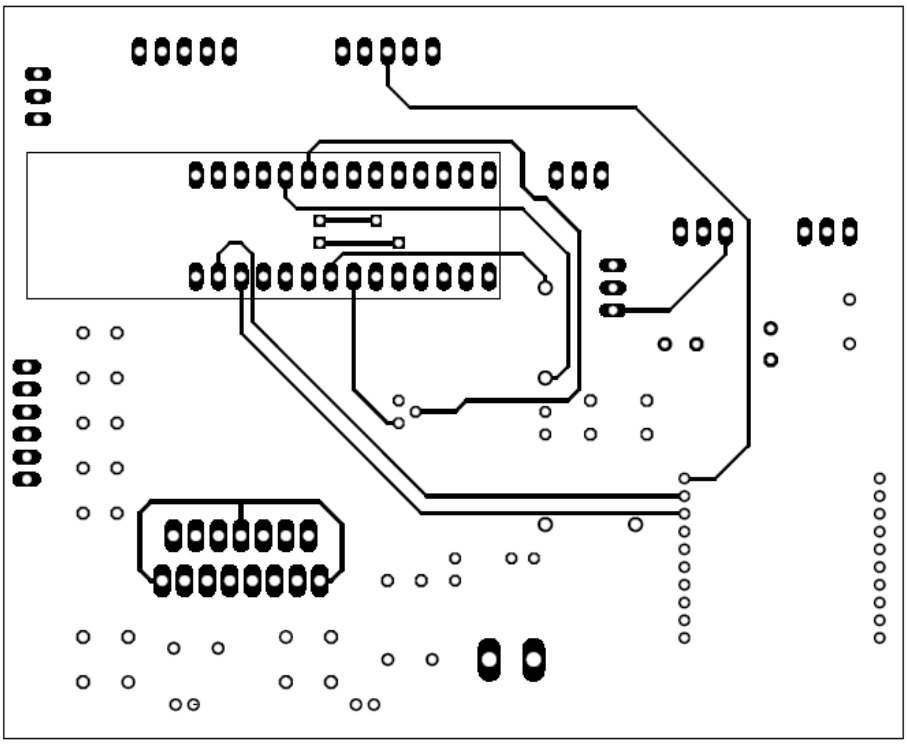


Figura 32 – Vista superior da placa da embarcação.

Após a fabricação da placa foram necessárias duas mudanças nos trilhos da desta para que coincidisse com o diagrama demonstrada na Figura 30. Estas mudanças consistem na realização da ligação de dois pontos que ficaram desconectados e na retirada de um curto entre os terminais do capacitor C4, estas alterações estão demonstradas nas Figura 33 e Figura 34.

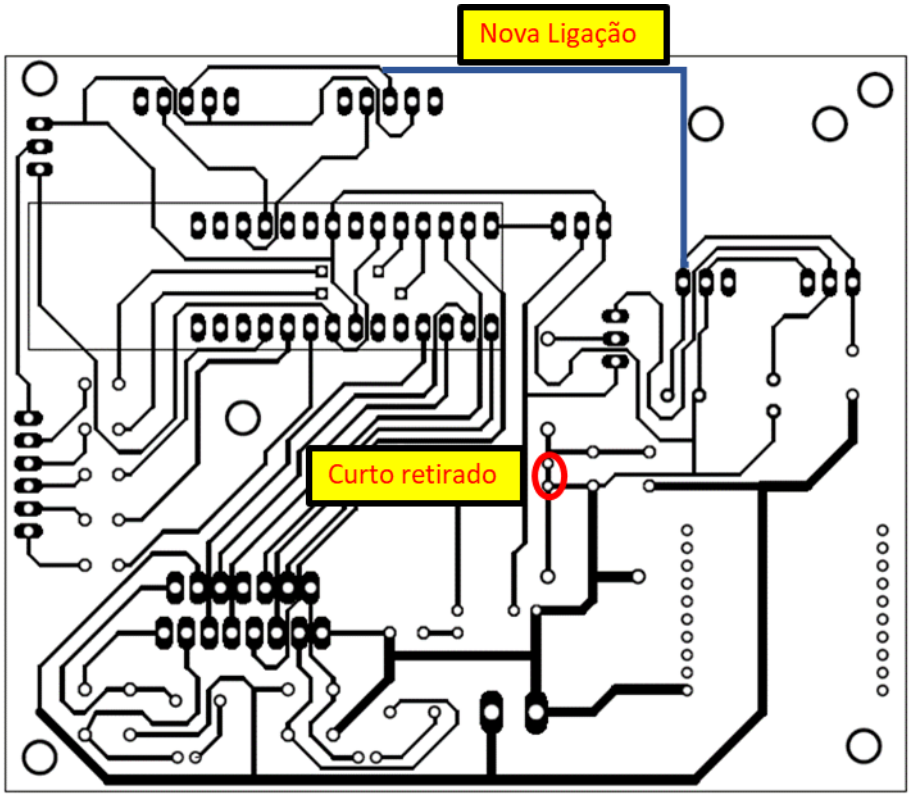


Figura 33 – Alterações realizadas.

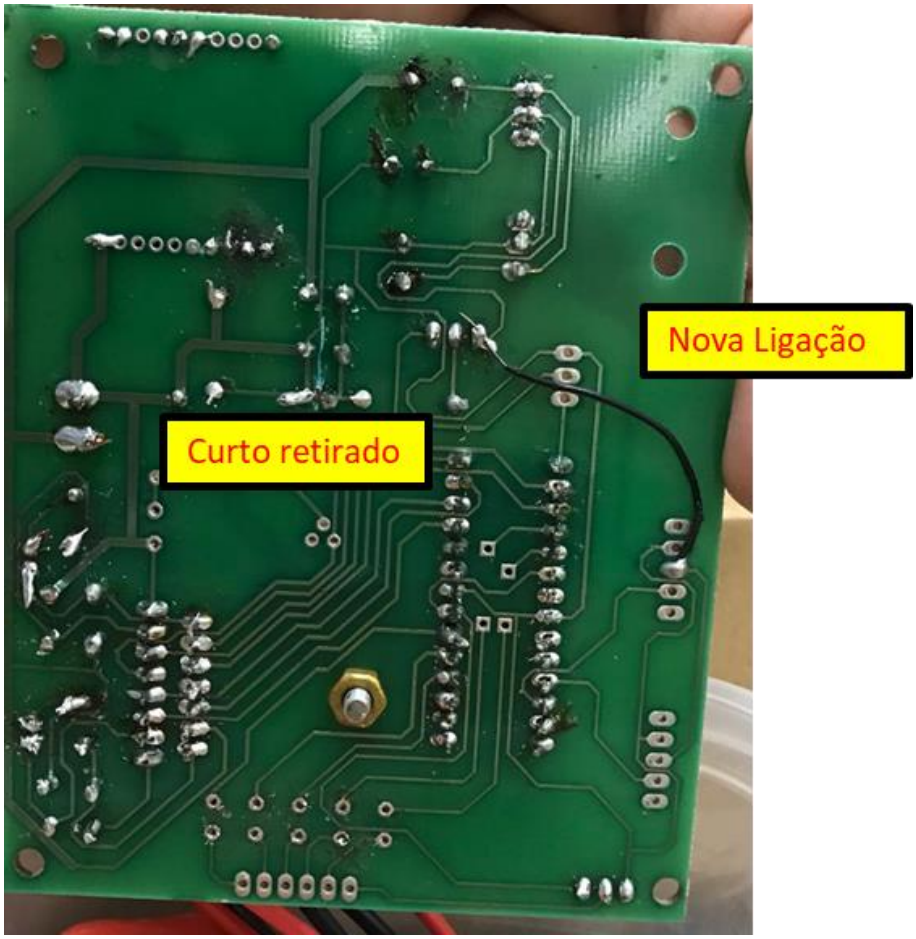
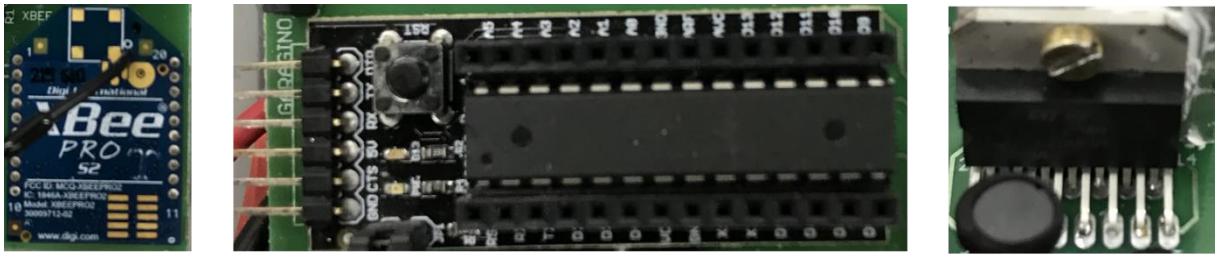


Figura 34 – Alterações implementadas na placa.

Os principais componentes utilizados na placa são o modulo XBee, o Garagino e a ponte L298N (Figura 35), a montagem final da placa pode ser verificada na Figura 36.



(a) (b) (c)

Figura 35 – Componentes. a) Xbee. b) Garadino. c) L298N.

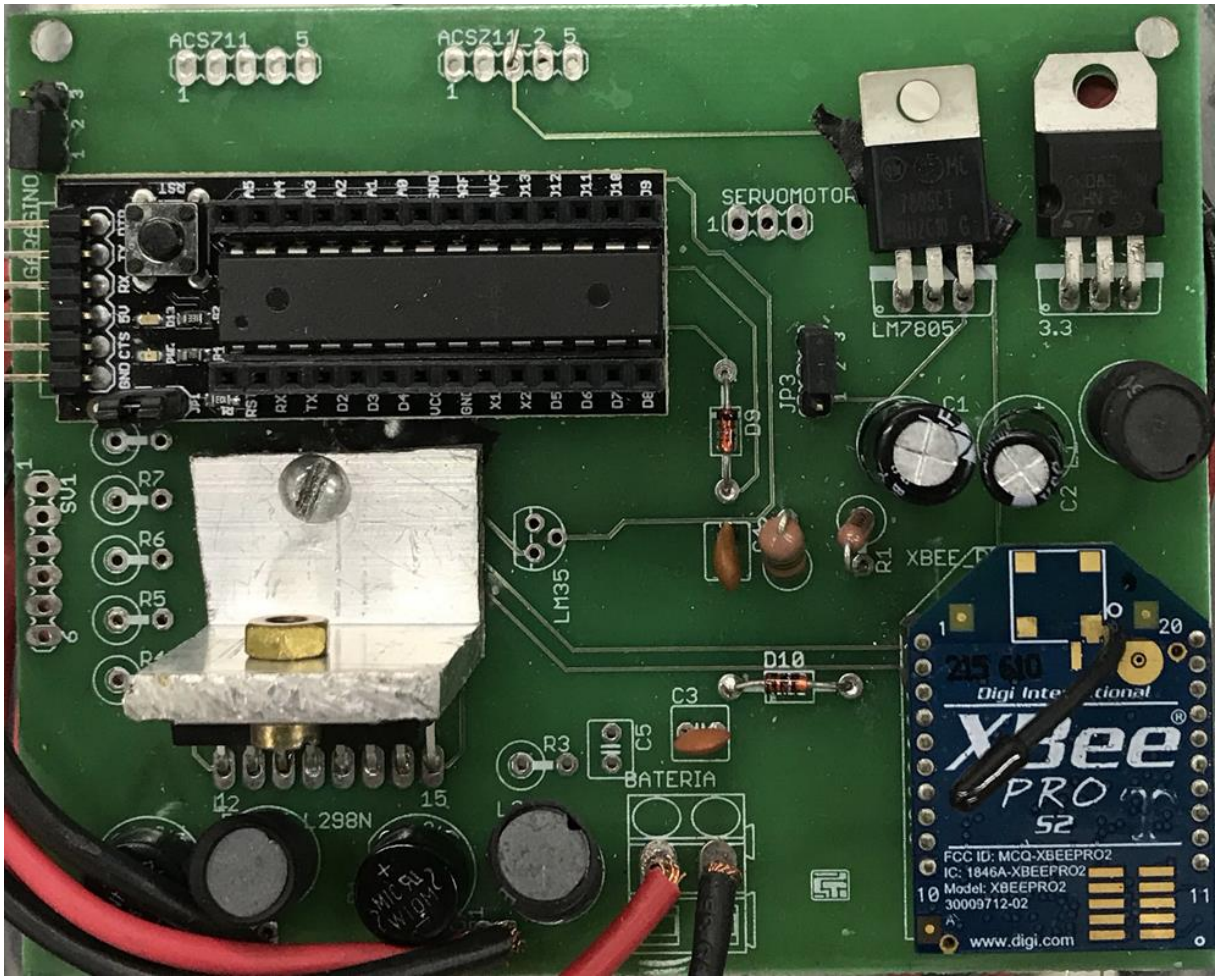


Figura 36 – Montagem final da placa.