Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Hugo Leonardo Rios de Almeida

Pequena embarcação autônoma microcontrolada

Rio de Janeiro

2022

Hugo Leonardo Rios de Almeida

Pequena embarcação autônoma microcontrolada

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

2022

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

A44 Almeida, Hugo Leonardo Rios de.
Pequena embarcação autônoma microcontrolada / Hugo Leonardo Rios de Almeida. - 2022.
67 f.

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

1. Pequena embarcação autônoma microcontrolada - Monografias. 2. Controle de navegação com obstáculos - Monografias. 3. eletrônica embarcada em embarcação aquática - Monografias. I. Cunha, José Paulo Vilela Soares da. II. Universidade do Estado do Rio de Janeiro. Faculdade de Engenharia. III. Título.

CDU 621.3

Hugo Leonardo Rios de Almeida

Pequena embarcação autônoma microcontrolada

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 11 de maio de 2022

Banca Examinadora:

_____

Prof. Dr. José Paulo Vilela Soares da Cunha (Orientador)

Faculdade de Engenharia - UERJ

_____

Prof. Dr. José Franco Machado do Amaral

Faculdade de Engenharia - UERJ

_____

Prof. Dr. Téo Cerqueira Revoredo

Faculdade de Engenharia - UERJ

Rio de Janeiro

2022

Agradecimentos

Aos meus pais e familiares, que com muito amor e carinho me apoiaram neste e em todos os momentos da minha vida.

À minha namorara Isabela que esteve comigo em todos os momentos dessa jornada, sempre me incentivando e nunca me deixando desanimar.

Ao Prof. Dr. José Paulo Vilela Soares da Cunha que com muita sabedoria e dedicação me orientou neste trabalho acadêmico.

Aos professores doutores José Franco Machado do Amaral e Téo Cerqueira Revoredo que mesmo em momentos de avaliação continuaram a me ensinar muito.

Aos professores e funcionários do Departamento de Eletrônica e Telecomunicações da UERJ por todos os ensinamentos e pela ótima convivência.

Aos amigos do curso de engenharia que tornaram esta jornada mais calorosa e transformaram a biblioteca em um ótimo ambiente de estudos e desenvolvimento.

“Seguir em frente, mesmo que seja um passo de cada vez,
é o que precisamos fazer para um milagre acontecer.”

Autor Desconhecido

Resumo

Este projeto concebe e implementa a automação de uma embarcação de nautimodelismo, modelo *Blast* 38104, com o objetivo de manter o barco em um rumo desejado evitando colisões com obstáculos. Neste projeto foi utilizado um microcontrolador Arduino, sensores de distância, magnetômetro e girômetro para realização das tarefas. O controle foi estruturado em duas malhas: controle de anticolisão (malha externa) e controle de rumo (malha interna). No controle de anticolisão: o microcontrolador recebe as leituras dos sensores de distância e calcula a melhor direção para fuga do obstáculo. Controle de rumo: o microcontrolador recebe a leitura do magnetômetro e girômetro e calcula as saídas para manter o barco no rumo desejado. Para execução das tarefas foi utilizado o controle PD (proporcional derivativo) nas duas malhas. Foi utilizado o protocolo de comunicação I2C para o microcontrolador adquirir as leituras dos sensores. Para a demonstração do funcionamento do sistema proposto, são apresentados alguns resultados experimentais obtidos em uma piscina.

Abstract

This project conceives and implements the automation of a model boating boat, model Blast 38104, with the objective of keeping the boat on a desired course, avoiding collisions with obstacles. In this project, an Arduino microcontroller, distance sensors, magnetometer and gyrometer were used to carry out the tasks. The control was structured in two loops: collision avoidance control (external loop) and heading control (internal loop). In collision avoidance control: the microcontroller receives the readings from the distance sensors and calculates the best direction to escape the obstacle. Heading control: the microcontroller receives the reading from the magnetometer and gyrometer and calculates the outputs to keep the boat on the desired heading. To perform the tasks, the PD (proportional derivative) control was used in the two loops. The I2C communication protocol was used for the microcontroller to acquire the sensor readings. To demonstrate the functioning of the proposed system, some experimental results obtained in a swimming pool are presented.

Lista de Figuras

Lista de Tabelas

Sumário

1 INTRODUÇÃO

Embarcações não tripuladas são veículos aquáticos que navegam sem a necessidade de uma tripulação embarcada para ser operada (Zhang, 2016) (Schultze, 2012). Seus controles podem ser automáticos (Arruda, 2012), através de uma rotina de programação com o trajeto e um ponto de retorno pré-definidos, ou podem ser operados remotamente (Buoro, 2013), mantendo comunicação constante com uma equipe de comando.

Agregando conceitos como sistemas embarcados e internet das coisas, as embarcações não tripuladas tem capacidade para se tornarem relevantes no patrulhamento marinho, assim como os *drones* se tornaram relevantes nos céus.

Suas aplicações são diversas (CUNHA, 2010) e incluem monitoramento de espécies marinhas, estudos climáticos e espionagem de áreas em conflito. Embarcações não tripuladas superam as tripuladas em vários aspectos pois a ausência de tripulação permite embarcações menores que poluem menos o meio ambiente e em caso de naufrágio não colocam em risco vidas humanas (Yan, 2010).

1.1 Objetivo

Desenvolver, construir e testar experimentalmente uma embarcação não tripulada e autônoma a partir de um barco de controle remoto *Blast* 38104 (Figura 1). O barco deve ser capaz de se manter em um rumo especificado e se desviar de obstáculos ao seu redor.

Figura 1 - Blast 38104

O sistema de controle é autônomo e usa sensores a bordo. A embarcação é capaz de se locomover seguindo um rumo especificado (CUNHA, 1992) contornando eventuais obstáculos (Naeem, 2011).

A eletrônica do barco foi desenvolvida tendo em vista a sua eficiência e modularidade, a fim de facilitar a adição de mais módulos à embarcação em projetos futuros.

1.2 Sistema Proposto

A Figura 2 mostra o diagrama do sistema proposto.

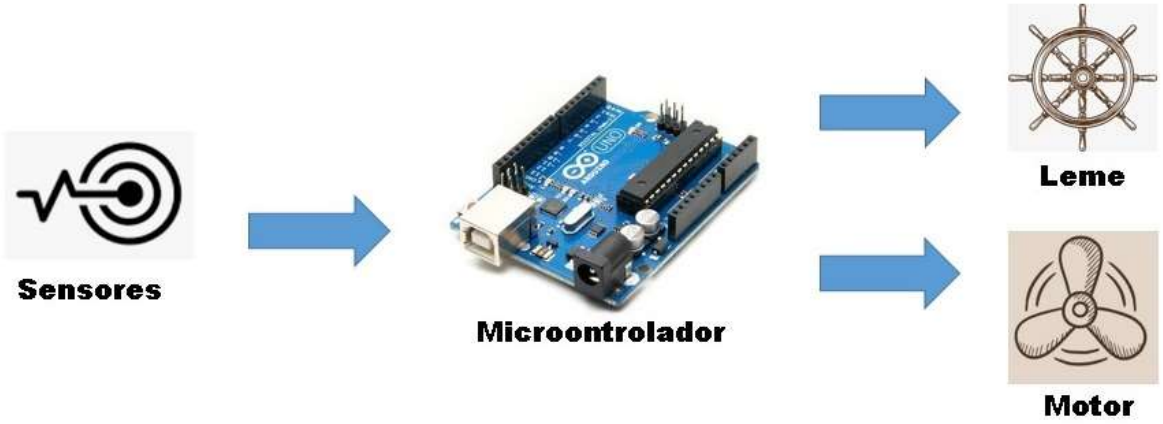


Figura 2 - Diagrama simplificado do projeto

Os sensores magnetômetro e girômetro fornecem as leituras necessárias para o cálculo de rumo da embarcação. O magnetômetro fornece a leitura de posição em relação ao rumo desejado e o girômetro fornece a leitura de velocidade angular. Possuindo as leituras de posição e velocidade é possível implementar um controlador PD (proporcional derivativo) (OGATA, 2004).

Usando os sensores de distância a laser instalados é possível monitorar obstáculos se aproximando do barco e evitar colisões. Sensores de distância foram instalados em direções específicas: a frente(proa), a trás(poupa); e nas laterais; garantem fuga de colisão frontal, traseira e lateral.

Todos os sensores escolhidos utilizam o protocolo de comunicação I^2C (*Inter-Integrated Circuit*) para transferência de dados. Onde o microcontrolador é o mestre, que solicita as medidas, e os demais sensores são escravos e fornecem as leituras pelo barramento.

A embarcação possui 2 controles: controle de rumo, responsável pelo acionamento do leme; e controle de velocidade responsável pelo acionamento do motor.

O controle de rumo (leme) possui duas malhas: uma de cálculo de rumo, utilizando a leitura do magnetômetro e girômetro para estabelecer um rumo a ser seguido pela embarcação; e uma de desvio de obstáculos, utilizando as leituras dos sensores de distância laterais para guiar o leme na direção oposta à do obstáculo. A malha de desvio de obstáculos é dominante em relação à malha de controle de rumo pois a embarcação deve evitar a colisão antes de seguir qualquer rumo.

O controle de velocidade (motor) possui uma malha e dois graus de liberdade, a velocidade e o sentido do giro. O sentido possibilitará o barco fazer manobras indo para frente e para traz. Este controle utiliza as leituras dos sensores de distância localizados na frente e na traseira do barco, o objetivo é evitar interações com objetos frontais e traseiros, melhorando as manobras e a capacidade de esquiva da embarcação.

1.3. Organização do texto

Este texto é organizado como descrito abaixo:

O Capitulo 1 apresenta uma breve introdução e descrição do sistema desenvolvido, bem como suas motivações.

O Capitulo 2 indica todos os sensores e dispositivos eletrônicos utilizados no projeto, além do protocolo de comunicação empregado.

O Capitulo 3 demonstra a modelagem da dinâmica do barco.

O Capitulo 4 explica a estratégia de controle adotada e a influência das posições dos sensores no barco.

O Capitulo 5 descreve os resultados obtidos experimentalmente em uma piscina.

2. ELETRÔNICA EMBARCADA

Neste capítulo são apresentados os componentes eletrônicos e mecânicos da embarcação, alguns já nativos do barco, como o motor de propulsão, a bateria 7,2V e o servomotor do leme, e outros adicionados para automatizar o barco, como o microcontrolador e sensores.

2.1 Motor do barco (*Traxxas 1275 Stinger*)

Motor elétrico conectado mecanicamente à hélice através de um sistema de engrenagens integrantes do leme que possibilitam o giro da hélice junto ao leme, engrenagens similares a um propulsor azimutal. Recebe alimentação diretamente do *driver* do motor, uma vez que o nível de tensão (0 – 7V) e a polaridade atuam sobre sua velocidade e sentido respectivamente.

É um motor *DC* (corrente contínua) *brushed* (com escovas), possui maior ruído e interferência eletromagnética que o modelo sem escovas. Porém os modelos *brushless* (sem escovas) foram avaliados caros para substituição no momento deste trabalho.

Construído com vinte voltas de fio elétrico ao redor da armadura produz menos torque que os modelos com vinte voltas duplas, porém é mais econômico que os outros modelos desta família de motores. Por ser um sistema embarcado, a economia de energia é um fator positivo deste motor.

Nos ensaios, de bancada o motor foi acionado pelo microcontrolador, através do *driver* (*Monster Motor Shield*) e usando uma fonte de alimentação do laboratório. Apresentou pico de 30V no instante de acionamento e funcionou em velocidade proporcional a tensão de alimentação, que variou de 0,6 à 7V. O motor não responde a tensões menores que 0,6V e emite ruído em tensões abaixo de 4V.

O motor foi acionado por sinal *PWM* (*Pulse Width Modulation*) e girou proporcionalmente. Não respondeu para valores de *PWM* menores que 25, gerados pelo Arduino, e apresentou giro máximo em 255. Ensaios de bancada foram realizados e obtiveram os resultados da Tabela 1.

Tabela 1 - Medidas de ensaio do motor em vazio

PWM_{motor}	$V_{motor}(V)$	$I_{motor}(A)$
25	0,6	0,4
50	1,3	1,7
75	2,0	2,6
100	2,7	2,8
125	3,4	3,4
150	4,1	4,0
200	5,5	4,5
255	7,0	5,0

O sinal *PWM* utilizado pelo microcontrolador Arduino apresenta ciclo de trabalho de 100% no valor de 255 e 0% no valor 0.

Em cada ensaio o motor permaneceu ligado por apenas 30 segundos, visando o não aquecimento do sistema e a proteção das engrenagens pois o teste foi realizado fora da água. Durante os testes não foi detectado aquecimento dos componentes.

2.2 Servomotor do leme (Traxxas 2080)

Servomotor facilmente encontrado em modelos de nautimodelismo. Possui 13kg-cm de torque de parada e alimentação entre 4,8V e 7,2V. Um sinal modulado em *PWM* define a posição do leme.

Foi observado que o servomotor responde satisfatoriamente para valores de *PWM* entre 75 e 205, para valores fora dessa faixa o dispositivo não se comporta linearmente.

O valor 140 de *PWM* aciona o servomotor na posição central do leme, para valores acima o leme se move para esquerda, para valores abaixo para direita.

Para o correto funcionamento do servomotor e visando sua proteção elétrica, foi desenvolvida uma placa reguladora de tensão com saída em 5V (Figura 3) (BOYLESTAD, 1998). O objetivo é reduzir a tensão de alimentação do servomotor e não deixa-lo ligado diretamente à bateria de 7,2V.

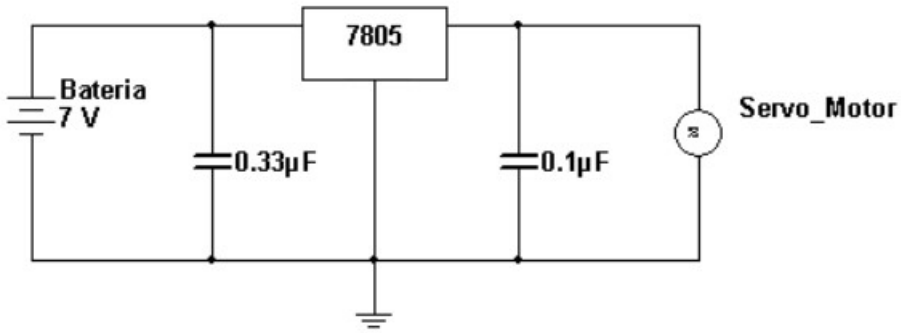


Figura 3 - Esquema elétrico da placa de conversão

2.3 Bateria

Bateria nativa do barco, modelo TRAXXAS 2922 de 7,2V e 3000mAh, composta de 6 células de hidreto metálico de níquel (NiMH) e conector robusto para isolamento de altas correntes. É necessária atenção ao coloca a bateria para recarregar na tomada pois costuma esquentar e precisa ser retirada da tomada imediatamente após completar a carga.

2.4 Mirocontrolador (*Arduino*) – *Arduino MEGA 2560*

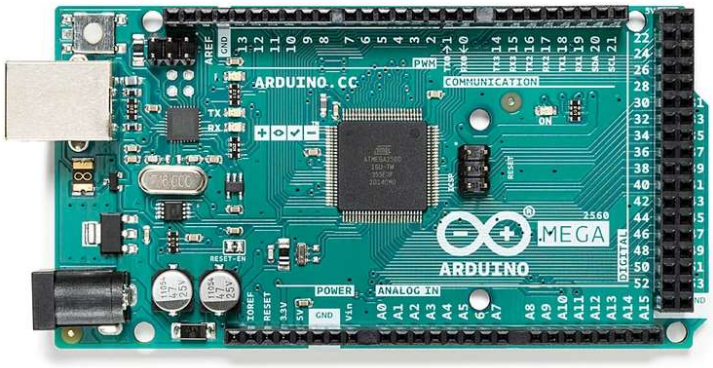


Figura 4 - Arduino MEGA R3

O microcontrolador *Arduino* (Arduino, 2012) (Arduino, 2009) centro de todo o controle da planta, recebe as leituras de todos os sensores instalados, efetua os cálculos de controle e aciona os atuadores para evitar os obstáculos e seguir o rumo desejado.

Possui portas *PWM* (POMILIO 2014) dedicadas além de portas seriais I^2C para facilitar o interfaceamento com os módulos.

Escolhida a plataforma *Arduino* pelo baixo custo e facilidade de implementação, sua capacidade de processamento e memória atendem ao projeto. A alternativa era o *Raspberry PI* porém se mostrou de valor elevado para compra.

Escolhido o modelo Mega-2560 pela capacidade de memória flash, comportando todo o programa de controle que se mostrou extenso, e velocidade de processamento, viabilizando a velocidade necessária para execução da estratégia de controle.

Este modelo possui um regulador de tensão interno, que reduz a tensão de alimentação de 7,2V (tensão entregue pela bateria) para 5V que é a tensão nominal da família Arduino.

Especificações do *Arduino MEGA 2560 R3* (*ATmega2560*):

- Tensão de operação recomendada: 7-12V
- Portas digitais: 54 (dos quais 15 oferecem saída *PWM*)
- Portas analógicas: 16
- Corrente contínua máxima por pino: 40mA
- *Flash memory*: 256Kb (8KB são utilizados pelo carregador de inicialização)
- *SRAM*: 8kB
- *EEPROM*: 4kB
- *Clock*: 16MHz
- Dimensões (CxLxA): 101,6x53,4x10mm
- Peso: 150g

2.5 *Driver* do motor (*Monster Motor Shield*)

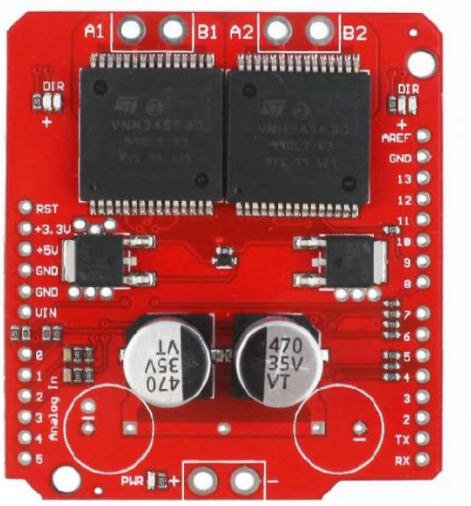


Figura 5 - *Monster Motor Shield*

O *Monster Motor Shield* (Figura 5) é um módulo eletrônico para plataforma *Arduino* e possibilita o controle de sentido e velocidade de até dois motores. Possui duas pontes H que serão ligadas em paralelo para entregar mais corrente elétrica ao motor.

As duas pontes H são acionadas simultaneamente para entregar maior corrente elétrica para o motor (Figura 6), nesta configuração as saídas A1 e B1 foram ligadas em curto-circuito assim como as saídas A2 e B2. Nesta configuração, para a correta sincronia das 2 pontes H, as portas 5 e 6 devem ser unidas e por elas o sinal *PWM* que aciona o motor é transmitido. As portas 4 e 9 devem ser unidas e receberem sinal de nível lógico alto (*high*) pelo microcontrolador, da mesma forma as portas 7 e 8 devem receber sinal de nível lógico baio (*low*). Com esta configuração a sincronia das duas pontes H é garantida e o modulo funciona como uma única ponte H de maior corrente elétrica.

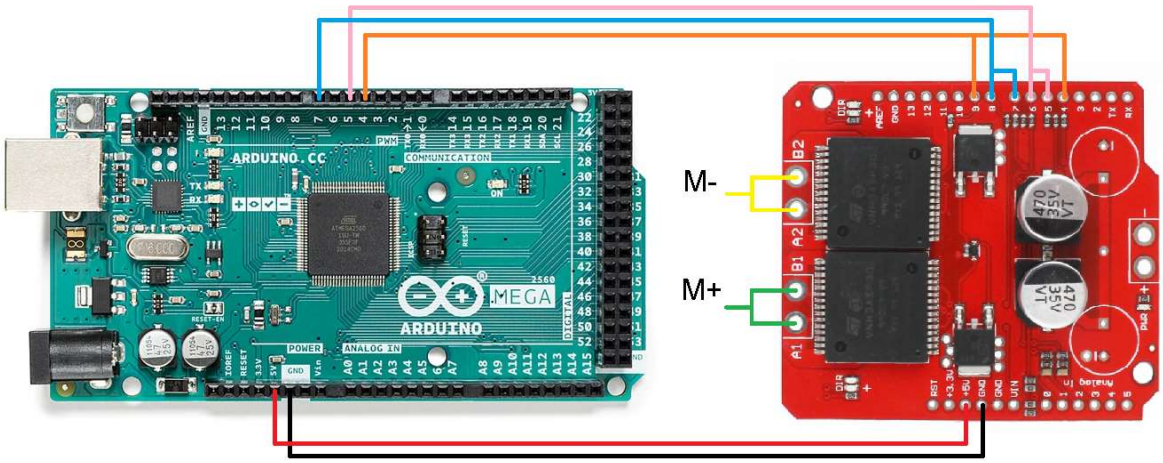


Figura 6 - Ligação eletrônica do microcontrolador e do *driver Monster Motor Shield*

Escolhido pela compatibilidade com o microcontrolador escolhido e pela disponibilidade de ser um item em posse do professor orientador, logo sua compra não foi necessária.

Contém modo de desativação térmica, evitando danos nos componentes por elevadas temperaturas. Possui um pino destinado ao monitoramento de corrente pelo *Arduino* e desligamento por subtensão, pois quando a bateria descarregar irá comprometer o funcionamento correto do sistema.

Especificações do módulo *Monster Motor Shield*:

- Tensão máxima: 5.5 a 16V
- Corrente de pico: até 30A
- Corrente: 14A (por canal)
- Frequência máxima PWM: 20 kHz
- Dimensões (CxLxE): 59x52x12mm
- Peso: 19g

2.6 Módulo magnetômetro(GY-511)

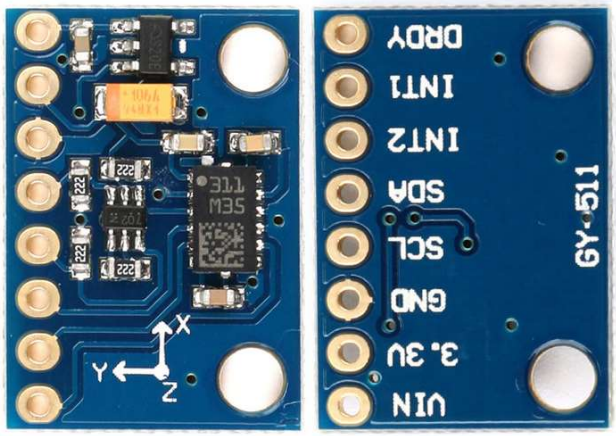


Figura 7 - Módulo magnetômetro GY-511

O Módulo possui o sensor QMC5883L, capaz de detectar o polo norte magnético da terra. Possui leitura em 3 eixos X, Y e Z ou em um ângulo em relação ao norte. Possui baixo consumo e boa precisão.

Sua escolha é baseada no custo mais elevado de seus concorrentes (modelo GY-91) e em sua capacidade de medida satisfatória. Este módulo possui comunicação I^2C, que facilita a comunicação entre o microcontrolador e os sensores do barco.

Especificações do módulo *GY-511*:

- Chip QMC5883L
- Magnetômetro de 3 eixos
- Tensão: 3.3 a 5,5V
- Precisão: 2 graus
- Corrente em modo de medição: 0,1mA
- Frequência máxima PWM: 20 kHz
- Interface de comunicação I^2C
- Dimensões (CxL): 18x13mm

2.7 Módulo acelerômetro e girômetro(*GY-521*)

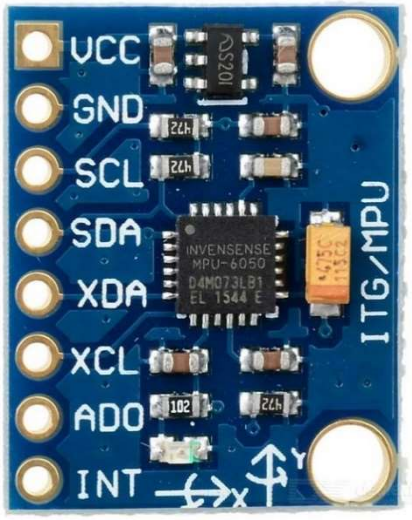


Figura 8 - Módulo acelerômetro e girômetro GY-521

O módulo possui o sensor MPU-6050 que combina 3 eixos de girômetro e 3 eixos de acelerômetro, sendo ao todo 6 graus de liberdade, juntamente com um processador digital de movimento. Possui boa precisão de posição e detecção de movimento.

Escolhido pelo custo e boa resolução de leitura de movimentos. Para neste projeto o mais importante é a leitura de girômetro e esta se mostrou suficiente nos testes de bancada. Experimentos foram realizados girando o módulo e conferindo se sua leitura era compatível com a amplitude e sentido do movimento. Este módulo possui comunicação I^2C, que facilita a comunicação entre o microcontrolador e os sensores do barco.

Especificações do módulo *GY-521*:

- Chip MPU-6050
- Tensão de Operação: 3-5V
- Comunicação: Protocolo padrão I^2C
- Faixa do Girômetro: ±250, 500, 1000, 2000°/s
- Faixa do Acelerômetro: ±2, ±4, ±8, ±16g
- Dimensões: 20 x 16 x 1mm

2.8 Módulo sensor *laser* (VL53L0X)

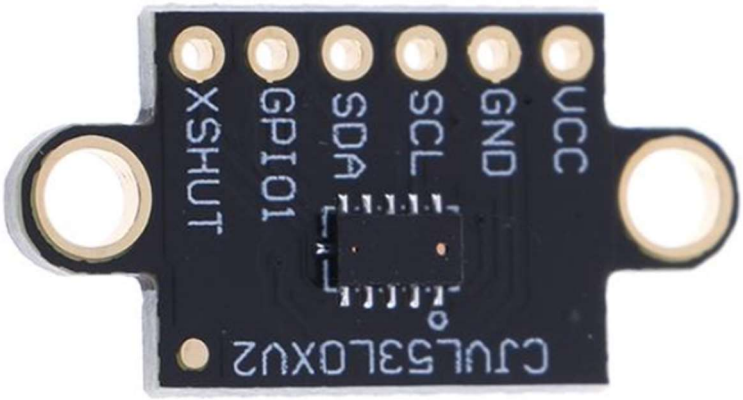


Figura 9 – Módulo sensor laser VL53L0X

O sensor usa uma luz de 940nm para medir a distância para um objeto, o módulo emite um feixe de luz invisível que é refletido no obstáculo e retorna ao sensor. Com base no tempo de leitura e na velocidade da luz, o módulo efetua o cálculo da distância.

Escolhido no lugar do sensor ultrassônico pela sua velocidade de medida e menor espalhamento do sinal. O sensor ultrassônico em testes experimentais se mostrou pouco útil pois media a superfície da água frequentemente, além de que a propagação física do som possui um ângulo de ataque maior que a da luz. O menor espalhamento do sinal luminoso se mostrou uma vantagem neste projeto.

Seu baixo custo e compatibilidade com a plataforma usada também são pontos positivos. Possui comunicação I^2C igual aos outros sensores utilizados.

Nos ensaios de bancada o módulo foi posicionado em frente a um objeto móvel e este objeto foi aproximado e afastado. Comparando as posições do objedo com as medidas realizadas pelo módulo, o sistema se mostrou eficiente em medir a distância entre um objeto e o barco. Este sensor luminoso apresentou alguma dificuldade em estabilizar medidas em objetos curvilíneos, cilíndricos ou redondos, porém o sensor ultrassônico apresentou piores medidas nestas circunstâncias.

Neste projeto foram utilizados quatro sensores de distância instalados em posições estratégicas do barco (frente, traseira, lado direito e lado esquerdo), como este modelo possui um endereço I^2C padrão, três dos sensores precisaram ter seus endereços trocados. Processo executado com o código da Tabela 2

Tabela 2 - Troca dos endereços dos sensores de distancia

Trecho do programa gravado no Arduino

```
void setID() {
  // todos desligados
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  delay(10);
  // todos ligados
//  digitalWrite(SHT_LOX1, HIGH);
//  digitalWrite(SHT_LOX2, HIGH);
//  digitalWrite(SHT_LOX3, HIGH);
  delay(10);
  // ativando LOX1 e resetando LOX2,LOX3
  digitalWrite(SHT_LOX1, HIGH);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  // iniciando LOX1
  if(!lox1.begin(LOX1_ADDRESS)) {
    Serial.println(F("Failed to boot first VL53L0X"));
    while(1);  }
  delay(10);
  // ativando LOX2
  digitalWrite(SHT_LOX2, HIGH);
  digitalWrite(SHT_LOX3, LOW);
  delay(10);
  //iniciando LOX2
  if(!lox2.begin(LOX2_ADDRESS)) {
    Serial.println(F("Failed to boot second VL53L0X"));
    while(1);  }
  delay(10);
```

```
  // ativando LOX3
  digitalWrite(SHT_LOX3, HIGH);
  delay(10);
//  //iniciando LOX3
  if(!lox3.begin(LOX3_ADDRESS)) {
    Serial.println(F("Failed to boot third VL53L0X"));
    while(1);  }
}
```

Especificações do módulo VL53L0X:

- Tensão de trabalho: 3,3 a 5V DC;
- Alcance: 2cm a 2m;
- Margem de erro: ±3%;
- Comunicação I^2C;
- Dimensões (CxLxE): 15,4x10,7x12mm;
- Peso: 5g;

2.9 Protocolo de Comunicação *I^2C*

Este protocolo é amplamente utilizado em sistemas embarcados baseados em microcontroladores Arduino. O modelo de funcionamento deste protocolo é baseado na interação entre elementos seguindo uma hierarquia mestre-escravo, ou seja, um microcontrolador (mestre) coordena toda a comunicação com os demais dispositivos conectados à rede. O mestre faz requisições aos sensores (escravos) que somente respondem quando solicitados, evitando assim colisão de informações na rede.

A estrutura física que o I^2C usa é um barramento de duas vias, uma para sincronismo de comunicação (SCL) e outra para transmissão das mensagens (SDA). O sinal SLC funciona como um sinal de *clock* sincronizando as mensagens do sinal SDA e evitando suas colisões.

Na Figura 10 ilustra a conexão eletrônica da comunicação *I^2C* entre os sensores e o microcontrolador Arduino.

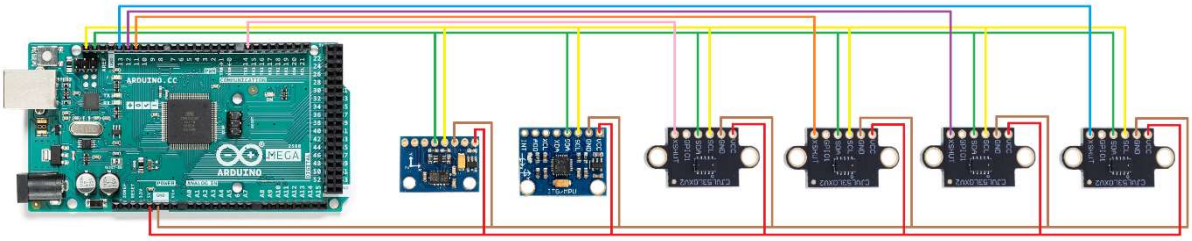


Figura 10 - Diagrama da conexão eletrônica entre os sensores e o microcontrolador pelo barramento I^2C

Este protocolo se mostrou eficiente na comunicação e na construção física, pois possibilitou a redução de fios na interface entre microcontrolador e sensores. No sistema o barramento de alimentação (Vcc e Gnd) e comunicação (SCL e SDA) são compartilhados por todos os sensores.

2.10 Disposição dos sensores

Para o correto funcionamento do sistema os sensores foram instalados em pontos estratégicos no barco. Neste projeto foram instalados dois sensores de distância nas laterais do barco, um na parte frontal e um na traseira (Figura 11).

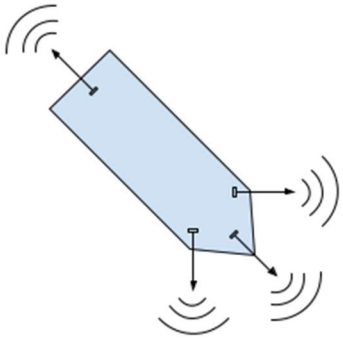


Figura 11 - Diagrama da posição dos sensores de distância no barco

Ambos os sensores foram instalados com um ângulo de 45º graus em relação à frente do barco (Figura 11). Este ângulo auxilia na dinâmica do sistema, uma vez que a inércia do movimento de embarcações aquáticas atrasa a resposta do sistema.

Em testes preliminares os sensores laterais foram instalados com um ângulo de 90º graus em relação à frente do barco, porém os resultados não foram bons pois a embarcação detectava o obstáculo praticamente ao tocá-lo. Nesta construção o barco não possuía tempo hábil de reação, o ângulo de 45º graus se mostrou muito funcional.

3. Modelagem da dinâmica do barco

A modelagem matemática tem por objetivo descrever matematicamente as variáveis físicas inerentes ao sistema (DORF, 2001). O modelo descreve de forma simplificada a posição do barco em relação a um sistema de coordenadas cartesianas (FOSSEN, 2002). Ora o sistema tem como referência as coordenadas da piscina onde foram feitos os testes, ora usa as coordenadas centradas no próprio barco para facilitar o cálculo do ângulo de rumo.

3..1. Sistema de coordenadas

- **Sistema de coordenadas inercial**

  É o sistema de coordenadas estacionário, pode ser representado pelas bordas da piscina.

  O_i – Origem do sistema de coordenadas inercial

  $x_i;\ y_i$ – Eixos do sistema de coordenadas inercial. Compõe o plano paralelo ao solo.

  z_i – Eixo vertical do sistema de coordenadas inercial. É ortogonal ao plano $x_i O_i y_i$. Possui sentido positivo para baixo.

- **Sistema de coordenadas móvel (fixo ao barco)**

  É o sistema em que a origem é fixa no centro de gravidade do veículo.

  O – Origem do sistema de coordenadas móvel.

  x, y, z – Eixos do sistema de coordenadas móvel, onde, x aponta para frente, y aponta para direita e z aponta para baixo.

Neste projeto o balanço da embarcação é desprezado. Os eixos z_i e z são paralelos e são desprezados para facilitar os cálculos. A Figura 12 mostra o diagrama do plano.

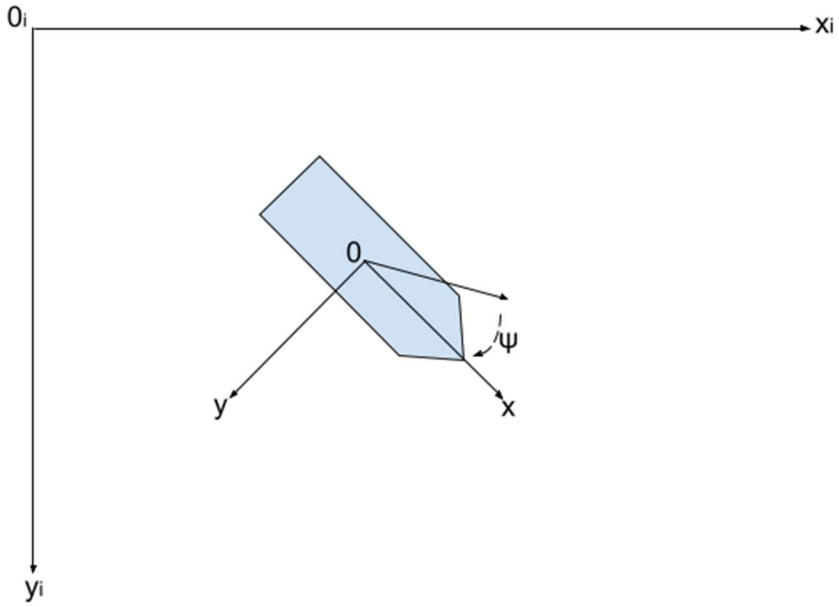


Figura 12 - Sistema de coordenadas adotado

3..2. Modelagem matemática da dinâmica do sistema leme (rumo)

A modelagem matemática de um sistema consiste na obtenção de um conjunto de equações que descrevem o sistema. Usando leis físicas é possível estabelecer um conjunto de equações diferenciais que estabeleçam uma relação entre as entradas e saídas do sistema.

O momento angular do barco é definido como:

$$L = J\,\omega \tag{3.1}$$

Onde:

L – momento angular

J – momento de inercia

ω – velocidade angular

M_l é o momento produzido pela variação do leme e é igual a variação do momento angular do barco:

$$M_l = \frac{dL}{dt} \tag{3.2}$$

Substituindo as equações (3.1) e (3.2)

$$\frac{d(J\omega)}{dt} = M_l \quad \therefore \quad \dot{\omega} = \frac{M_l}{J} \tag{3.3}$$

Onde $\dot{\omega}$ é a aceleração angular do barco.

Para tornar o sistema mais próximo da realidade, é adicionada a parcela E_l que representa o erro:

$$\dot{\omega} = \frac{1}{J}(M_l + E_l) \tag{3.4}$$

A Figura 13 mostra o sistema leme em diagrama de blocos. As relações entre as variáveis são as integrais expressas como $\frac{1}{s}$. A posição angular (Ψ) é a integral da velocidade angular (ω) que por sua vez é a integral da aceleração angular ($\dot{\omega}$).

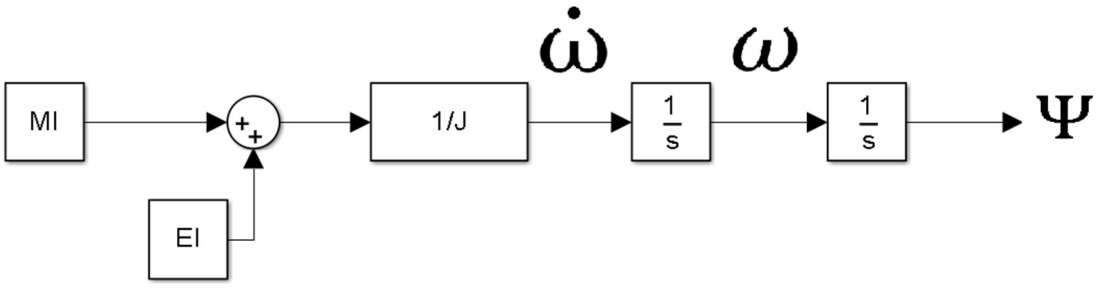


Figura 13 - Diagrama de blocos do sistema leme

Representando o sistema através de equações de estado:

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) + D(t)u(t) \end{cases} \tag{3.5}$$

Onde:

u(t), $\in R^m$, é o sinal de controle do sistema;

$\dot{x}$(t), $\in R^n$ é a derivada do estado x(t);

y(t), $\in R^p$ é o sinal de saída do sistema;

A(t) é a matriz de estado, ordem n x n;

B(t) é a matriz de entrada, ordem n x m;

C(t) é a matriz de saída, ordem p x n;

D(t) é a matriz de transmissão direta, ordem p x m;

Assim, a dinâmica angular do barco fica:

$$x = \begin{bmatrix} \Psi \\ \omega \end{bmatrix}, y = \begin{bmatrix} \Psi \\ \omega \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1/J \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = 0 \tag{3.6}$$

Substituindo (3.6) em (3.5) é encontrada a equação de estado:

$$\begin{bmatrix}\omega \\ \dot{\omega}\end{bmatrix} = \begin{bmatrix}0 & 1 \\ 0 & 0\end{bmatrix}\begin{bmatrix}\Psi \\ \omega\end{bmatrix} + \begin{bmatrix}0 \\ 1/J\end{bmatrix}(M_l + E_L) \tag{3.7}$$

$$\begin{bmatrix}\Psi \\ \omega\end{bmatrix} = \begin{bmatrix}1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\Psi \\ \omega\end{bmatrix} \tag{3.8}$$

3..3. Modelagem matemática da dinâmica do sistema motor (propulsão)

Para definir as equações de controle do motor é necessário utilizar equações de movimento linear. As equações são bem similares, o que facilita o desenvolvimento da estratégia de controle.

O momento linear do barco é definido como:

$$Q = m\,v \tag{3.9}$$

Onde:

Q – momento linear

m – massa do barco

v – velocidade linear

M_m é o momento produzido pela variação de velocidade do motor e é igual a variação do momento linear do barco:

$$M_m = \frac{dQ}{dt} \tag{3.10}$$

Substituindo as equações (3.9) e (3.10)

$$\frac{d(mv)}{dt} = M_m \quad \therefore \quad \dot{v} = \frac{M_m}{m} \tag{3.11}$$

Onde $\dot{v}$ é a aceleração linear do barco.

Para tornar o sistema mais próximo da realidade, é adicionada a parcela E_m que representa o erro:

$$\dot{v} = \frac{1}{m}(M_m + E_m) \tag{3.12}$$

A Figura 14 mostra o sistema motor em diagrama de blocos. Novamente as relações entre as variáveis são funções integrais ($\frac{1}{s}$). A posição linear (d) é a integral da velocidade linear (v) que por sua vez é a integral da aceleração linear ($\dot{v}$).

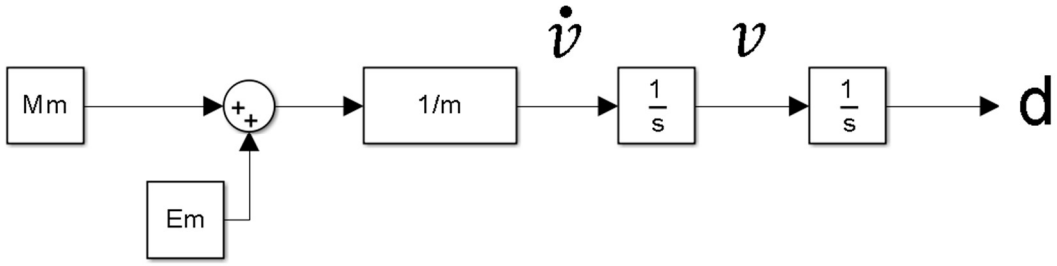


Figura 14 - Diagrama de blocos do sistema motor

Representando o sistema através de equações de estado:

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t) \\ y(t) = C(t)x(t) + D(t)u(t) \end{cases} \tag{3.13}$$

Assim, a dinâmica linear do barco fica:

$$x = \begin{bmatrix} d \\ v \end{bmatrix}, y = \begin{bmatrix} d \\ v \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1/m \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = 0 \tag{3.14}$$

Onde d é uma distância a ser alcançada, frontal ao barco ponto.

Substituindo (3.13) em (3.14) é encontrada a equação de estado:

$$\begin{bmatrix} v \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} (M_m + E_m) \tag{3.15}$$

$$\begin{bmatrix} d \\ v \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ v \end{bmatrix} \tag{3.16}$$

4. Estratégias de Controle

O controle desenvolvido neste projeto utiliza um sensor magnetômetro, um sensor girômetro e quatro sensores de distância distribuídos pelo barco. Para a estratégia de desvio de obstáculos foram utilizados os sensores de distância. Para o desvio de obstáculos nas laterais foram instalados um sensor apontando para a direita e outro para a esquerda. Para evitar obstáculos a frente e atrás foram instalados um sensor apontando para a frente do barco e outro para trás.

Para ambas as estratégias, controle de rumo e controle de erro, foi escolhido o modelo PD(proporcional derivativo) para efetuar o controle. Escolha justificada na próxima seção.

4.1 Controlador PD

Antes de apresentar o controlador PD é necessário apresentar o controlador PID, suas características e o porquê da decisão de desprezar a parcela integradora neste projeto para utilizar o PD.

O controlador PID apresenta atuação robusta e aplicável a diversos sistemas de controle e situações operacionais (Castrucci, 2011), sua facilidade de configuração e ajustes são características decisivas na escolha deste modelo para sistemas industriais e hobistas.

O controlador PID é capaz de tratar as perturbações existentes do sistema e gerar um sinal de saída que compense as interferências externas no sistema, tais interferências são tratadas ao incluir a parcela de erro na equação do controle.

A equação matemática que descreve o controlador PID é a seguinte:

$$u(t) = K_p \mathrm{e}(t) + K_i \int_0^t \mathrm{e}(t)dt + K_d \frac{d\mathrm{e}(t)}{dt} \tag{4.1}$$

Na qual:

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

$\mathrm{e}(t)$ – Sinal de entrada do controlador, sinal de erro;

K_p – Constante de ganho proporcional;

K_i – Constante de ganho integral;

K_d – Constante de ganho derivativo;

A parcela proporcional da equação é responsável pela ação direta de um valor de ganho K_p aplicado ao sinal de erro. Este ganho amplifica o sinal de erro gerando uma reposta maior do sinal do controlador.

$$u(t) = K_p \mathrm{e}(t) \tag{4.2}$$

A parcela integral do controlador que corresponde à constante K_i é responsável por contrapor o erro de regime permanente (*offset*). Assumindo que este erro é pequeno e desprezível, esta parcela da equação de controle é eliminada configurando $K_i = 0$. Erros de regime permanente em ambientes aquáticos são efeitos de correntes marítimas ou ação dos ventos em direção constante, estes fenômenos não são encontrados em uma ambiente de testes pequeno como uma piscina.

A parcela derivativa da equação é responsável por contrapor a velocidade de variação do erro, diminuindo o tempo de resposta do sistema, ou seja, tornando o sistema mais rápido para responder a perturbações.

$$u(t) = K_d \frac{d\mathrm{e}(t)}{dt} \tag{4.3}$$

Logo, a equação que descreve o sistema PD proposto é:

$$u(t) = K_p \mathrm{e}(t) + K_d \frac{d\mathrm{e}(t)}{dt} \tag{4.4}$$

Onde as constantes K_p e K_d são valores atribuídos ao controlador para que o sistema apresente o melhor desempenho, tais valores foram encontrados a partir de aproximações sucessivas e observação experimental.

4.2 Sistema Desenvolvido

O sistema desenvolvido une três estratégias de controle (Figura 15):

- Controle de Rumo – Responsável pela busca incessante pelo ângulo de referência.
- Controle de Desvio de Obstáculos – Responsável pela esquiva de obstáculos laterais ao barco.
- Controle de Velocidade do Motor de Propulsão – Responsável por controlar a velocidade do barco e a fuga de obstáculos frontais e traseiros.

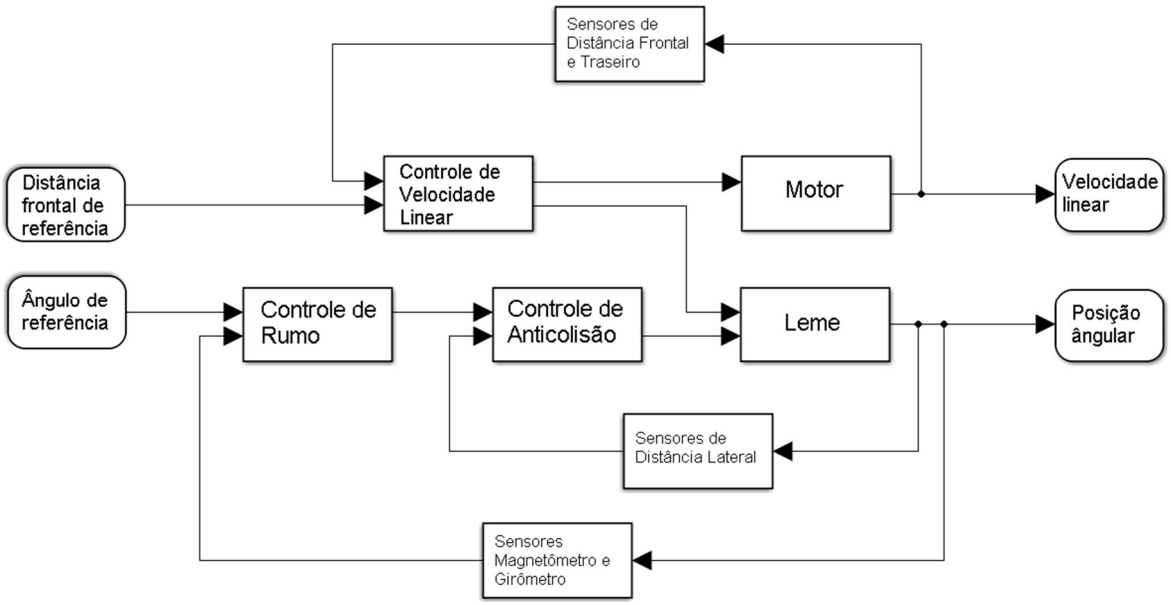


Figura 15 - Diagrama completo do controle da planta

No diagrama as entradas do sistema são respectivamente os valores de referência de distância frontal e ângulo de rumo, As saídas representam a velocidade linear do motor de propulsão e a posição angular do servomotor do leme.

Os controles de Rumo e Desvio de obstáculos formam um controle de duas malhas, sendo este último fazendo parte da malha interna (dominante sobre o output) e o primeiro sendo a malha externa (atuante desde que a malha interna não atue).

A malha de controle de velocidade do motor atua em paralelo às outras duas, atuando sobre o motor do barco mas também interferindo no sentido de direção do leme, este controle está melhor explicado na Seção 4.5.

4.3 Controle de Rumo

O controle de rumo foi desenvolvido com o objetivo de o barco seguir um rumo especificado como ângulo de referência, sendo Ψ a diferença entre este ângulo e a posição angular do barco. Utilizando a leitura do magnetômetro o sistema identifica a posição atual, a diferença entre a posição atual e o ângulo de referência é a variável de entrada da equação de controle. A leitura do girômetro indica a taxa de variação do ângulo (ω) enquanto o barco está em movimento.

Então a equação de controle fica:

$$u(t) = K_p\Psi + K_d\omega \tag{4.5}$$

Tabela 3 - Trecho do código que representa o controle de rumo

Trecho do programa gravado no Arduino

```
void controle_leme(){
  input = (medida_bussola);  // leitura do magnetômetro
    //Calculando as variaveis de erro
    erro_b = - setpoint_b + input;
    d_input_b = (acc_veloc_ang);  //leitura do aceletômetro, velocidade angular
    //Calcula PD
    output = ((kp_b*erro_b) + (kd_b*d_input_b));
    //Calcula limites de output
    if(output > out_max){
      output = out_max;}
    else{
      if(output < out_min){
        output = out_min;}
    }
  }
  //out_leme_pwm = 145 + output;  // 145 é a posição central do leme
}
```

No fim do código é possível notar uma condição para limitar a variável de saída, pois as engrenagens que compõem o sistema leme possuem limitação de movimento. Para não transpor os limites de movimento das engrenagens este código se faz necessário.

4.4 Controle de desvio de obstáculos laterais

Um controle de esquiva de bordas foi desenvolvido para proteger a integridade da embarcação. Neste controle foram utilizados dois sensores, um em cada lado do barco para medir a distância entre a embarcação e a borda (Figura 16).

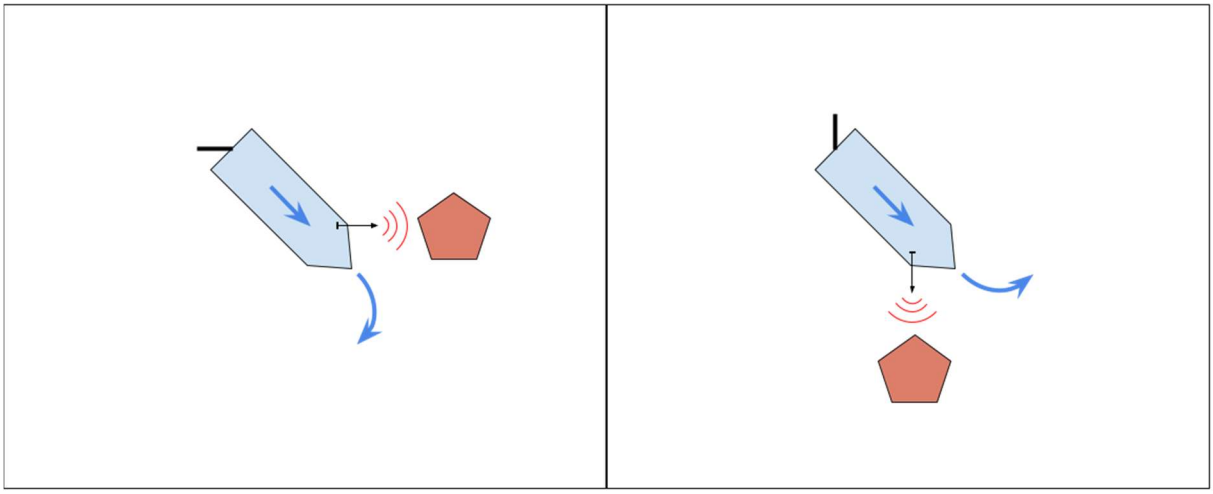


Figura 16 - Ilustração das ações anticolisão: (a) anticolisão à esquerda(bombordo); (b) anticolisão à direita(estibordo);

Este controle entra em operação apenas se um dos sensores laterais obtiver uma leitura de distância menor que 40cm, este valor foi atribuído após observação experimental, para distâncias abaixo deste valor o sistema não tem tempo de responder habilmente ao obstáculo.

A estratégia é similar à do controle de rumo, com a modificação que o *setpoint* passa a ser o valor médio entre as medidas dos sensores laterais, com este cálculo é possível garantir a fuga de obstáculos em ambos os lados.

Nesta estratégia, uma variável "*last_input*" armazena a medida do último "*input*", ou seja a última leitura dos sensores. Dessa forma o parâmetro D do controle PD é definido como sendo a diferença entre as variáveis "*input*" e "*last_input*".

Tabela 4 - Trecho do código que representa o controle do desvio de obstáculos

Trecho do programa gravado no Arduino

```
void controle_desvio(){
  input = (distancia_bb - distancia_eb);
    //Calculando as variaveis de erro
    error_c = setpoint_c - input;
    d_input_c = (acc_veloc_lin);  //leitura do aceletômetro, velocidade linear
    //Calcula PD
    output = ((kp_c*error_c) + (kd_c*d_input_c));
    //Calcula limites de output
    if(output > out_max){
      output = out_max;}
    else{
      if(output < out_min){
        output = out_min;}
    }
  }
  //out_leme_pwm = 145 + output;
}
```

4.5 Controle de velocidade e sentido do motor

O controle de velocidade do motor é definido como um sistema PD tendo como ponto de referência (d) um ponto a uma distância de 40cm a frente do barco. Neste trabalho não foi definido um ponto de chegada para que o barco busque exaustivamente seguir o rumo especificado desviando de obstáculos e demonstrando diversas trajetórias possíveis. Usando um ponto distante frontalmente, o barco vai sempre perseguir o rumo especificado, sendo limitado apenas pelas margens da piscina onde foram realizados os testes. Para este controle foram instalados dois sensores de distância, um na frente e outro na traseira do barco (Figura 17).

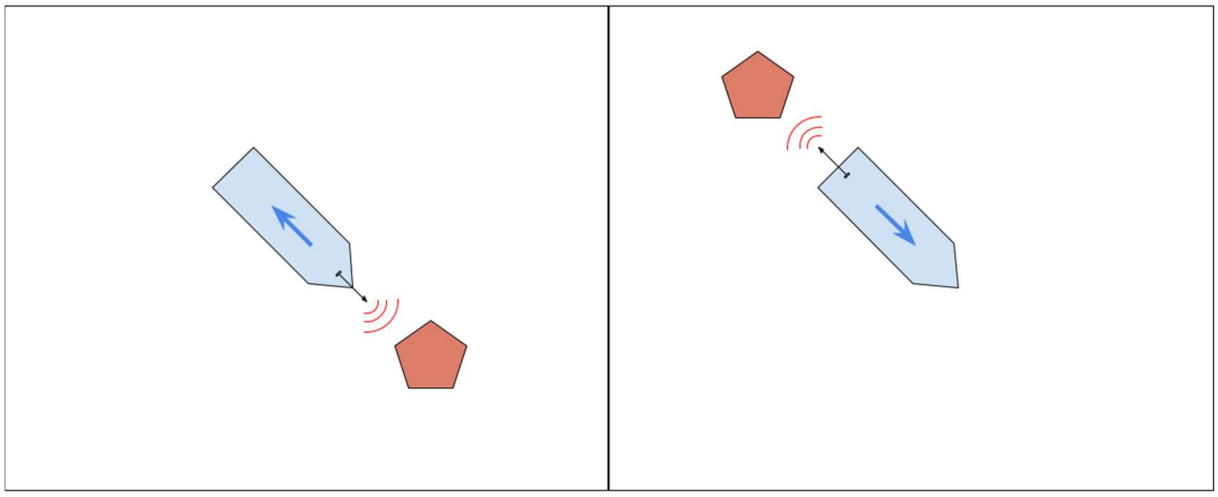


Figura 17 - Diagrama de sentido do motor: (a) sentido negativo; (b) sentido positivo;

A inércia de movimento da embarcação auxilia o sistema para evitar colisões frontais, uma vez q o barco nunca ficará parado em uma determinada posição, estará sempre se movimentando e alterando as leituras dos sensores e assim os parâmetros do controle. O fato de o barco ter dificuldade de parar em uma posição linear fixa faz ele ir além da distância de referência utilizada neste controle, tanto para frente quanto para trás, esta amplitude de movimento linear garante ao barco uma maior variação angular, uma vez que ele apenas varia sua posição angular quando está em movimento linear.

Para a leitura de velocidade (v) foi utilizado novamente o acelerômetro, sua leitura de velocidade linear foi muito útil para o controle do motor.

Então a equação de controle fica:

$$u(t) = K_p d + K_d v \tag{4.6}$$

Diferente do controle do leme que aceita uma variação simples de valores em sua variável de saída, o controle do motor deve tratar variáveis negativas. Quando o controle gerar um resultado para *output* negativo, ou seja, o barco deve ir para trás, o controlador deve inverter as lógicas dos sinais dos *output* tanto do motor quanto do leme. Transformando a saída do motor, que seria negativa, em uma saída positiva com uma *flag* para inverter o sentido através do *driver* do motor. Alterando também a saída do leme, fazendo ele buscar o rumo também no movimento oposto do barco.

Esta estratégia se mostrou eficiente nos testes experimentais pois possibilitou ao barco realizar manobras similares à “baliza” de um carro, e desta forma evitar as bordas da piscina com eficiência.

a

Tabela 5 - Trecho do código que representa o controle de velocidade do motor

Trecho do programa gravado no Arduino

```
void controle_motor(){
  input_m = (distancia_ff – distancia_tt) + 40;
  time_now_m = millis();
  time_change_m = (time_now_m - last_time_m);
  if(time_change_m >= sample_time_m){
    //Calculando as variaveis de erro
    error_m = - setpoint_m + input_m;    d_input_m = (input_m - last_input_m);
    //Calcula PID
    output_m = ((kp_m*error_m) + (kd_m*d_input_m));
    //Calcula limites de output
    if(output_m > out_max_m){
      output_m = out_max_m;}
    else{
      if(output_m < out_min_m){
        output_m = out_min_m;}
    }
     last_input_m = input_m;
     last_time_m = time_now_m;
  }
  //pwm de saida do motor nao pode ser valor negativo
  if(output_m < 0){
    output_m = output_m * (-1);
    sentido_motor = 0.0;}
  else {
    sentido_motor = 1.0;
  }
```

```
  motor_pwm = output_m;
}
```

É possível notar na parte final do código do controle do motor que o sinal da variável é tratado de forma a usar uma outra variável como *flag* para tratar os demais sinais a partir dela.

5. Resultados experimentais

Neste capítulo são apresentados os resultados obtidos. Todos os experimentos foram desenvolvidos em uma piscina desmontável de 3 mil litros, com dimensões de 1,5m x 2m. O volume de água utilizado foi o mínimo necessário para o barco flutuar sem interagir com o fundo da piscina, o nível da água ficou em cerca de 20cm de altura.

A velocidade do motor foi mantida em níveis baixos, apesar do barco ser destinado a grandes velocidades, pois assim o barco precisava de menos espaço para as manobras. Em velocidades medidas o barco alcançava a borda da piscina muito rápido, não dando tempo para realizar a manobra completa. Altas velocidades não foram testadas.

5.1 Experimento 1 – Controle de rumo

Nesta experiência o barco foi colocado em uma posição perpendicular ao rumo desejado. O rumo escolhido é paralelo à maior borda da piscina para que a embarcação passe o máximo tempo seguindo o rumo.

A Figura 18 demonstra a trajetória do barco ao perseguir o rumo desejado. Em todos os testes realizados na piscina, o ângulo de rumo foi perseguido corretamente pela embarcação.

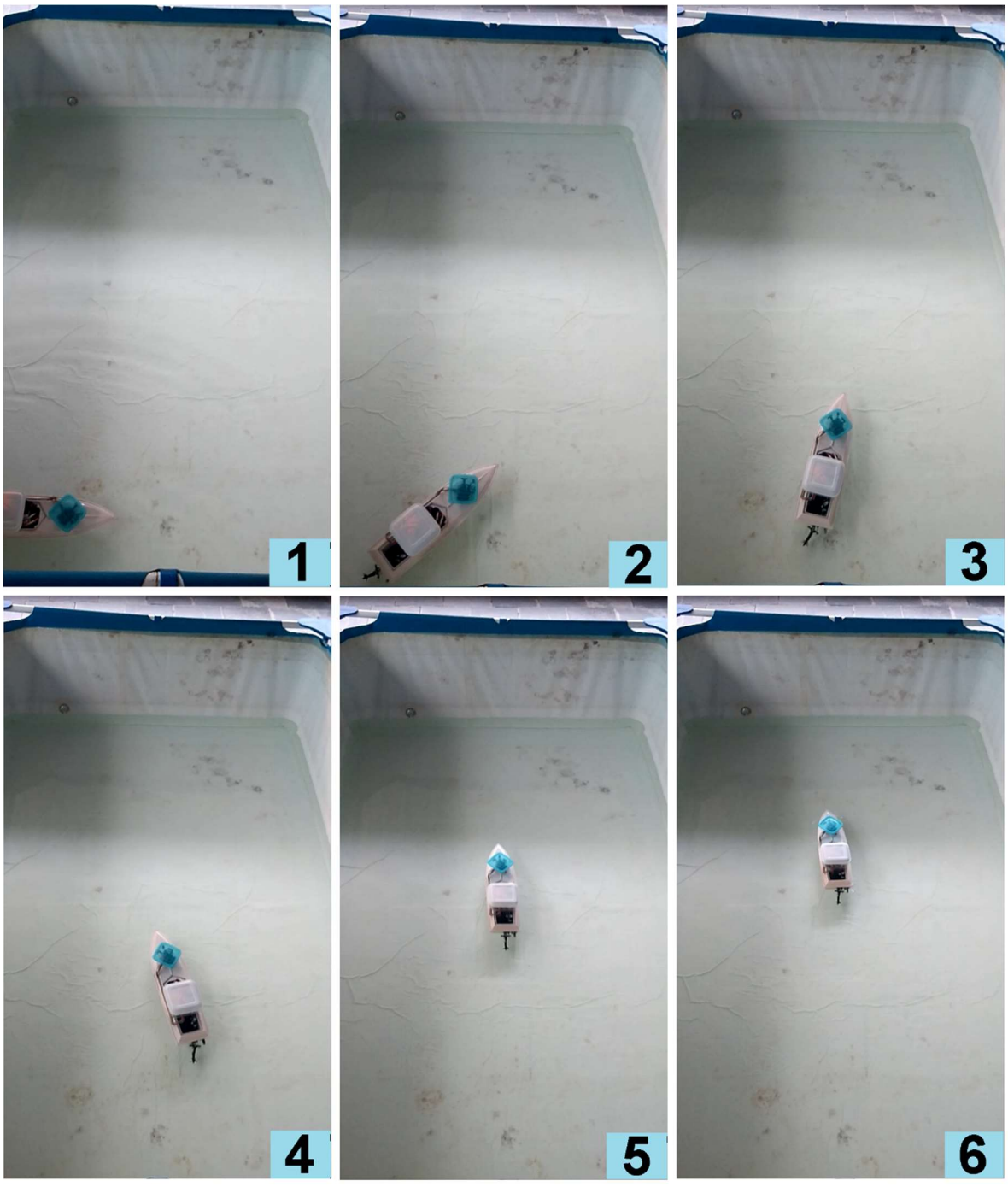


Figura 18 - Experiência do controle de rumo

A Figura 19 demostra as variáveis em ação no controle de rumo. É possível notar a incessante busca pelo rumo através da ação nos atuadores, a velocidade do motor permanece praticamente constante e o leme oscila bastante buscando manter o barco na direção certa.

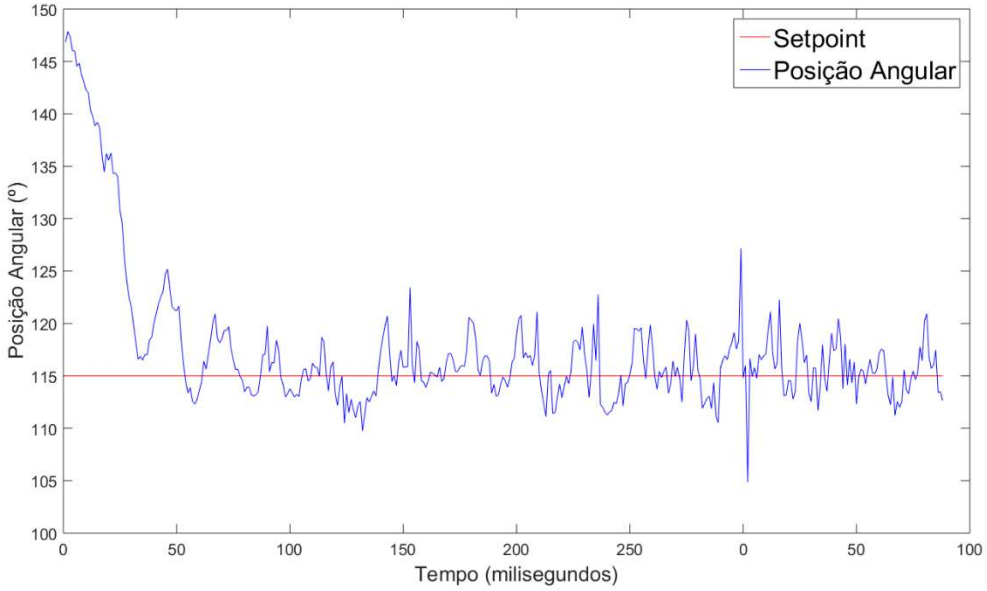


Figura 19 - Resultado do controle de rumo

5.2 Experimento 2 – Controle de desvio de obstáculos laterais

Neste experimento foi adicionado um obstáculo para enfatizar o movimento de fuga do barco em relação ao objeto. A embarcação responde da mesma maneira ao detectar a borda da piscina. No caso de detectar a borda da piscina o barco tende a seguir a borda até o fim pois ele ainda busca o rumo do controle principal.

Para objetos móveis o experimento demonstra eficácia no contorno do obstáculo. Este trabalho não realizou testes com objetos indo ao encontro direto com o barco, apenas obstáculos fixos ou móveis com movimentos laterais à embarcação.

Na Figura 20 é possível notar que durante o período em que o barco esta esquivando do objeto, o controle de rumo não comanda a direção da embarcação, apenas a estratégia de fuga funciona neste momento. Após o movimento de fuga, quando o barco não detecta mais nenhum objeto em sua proximidade o controle de rumo volta guiar a embarcação para o rumo correto.

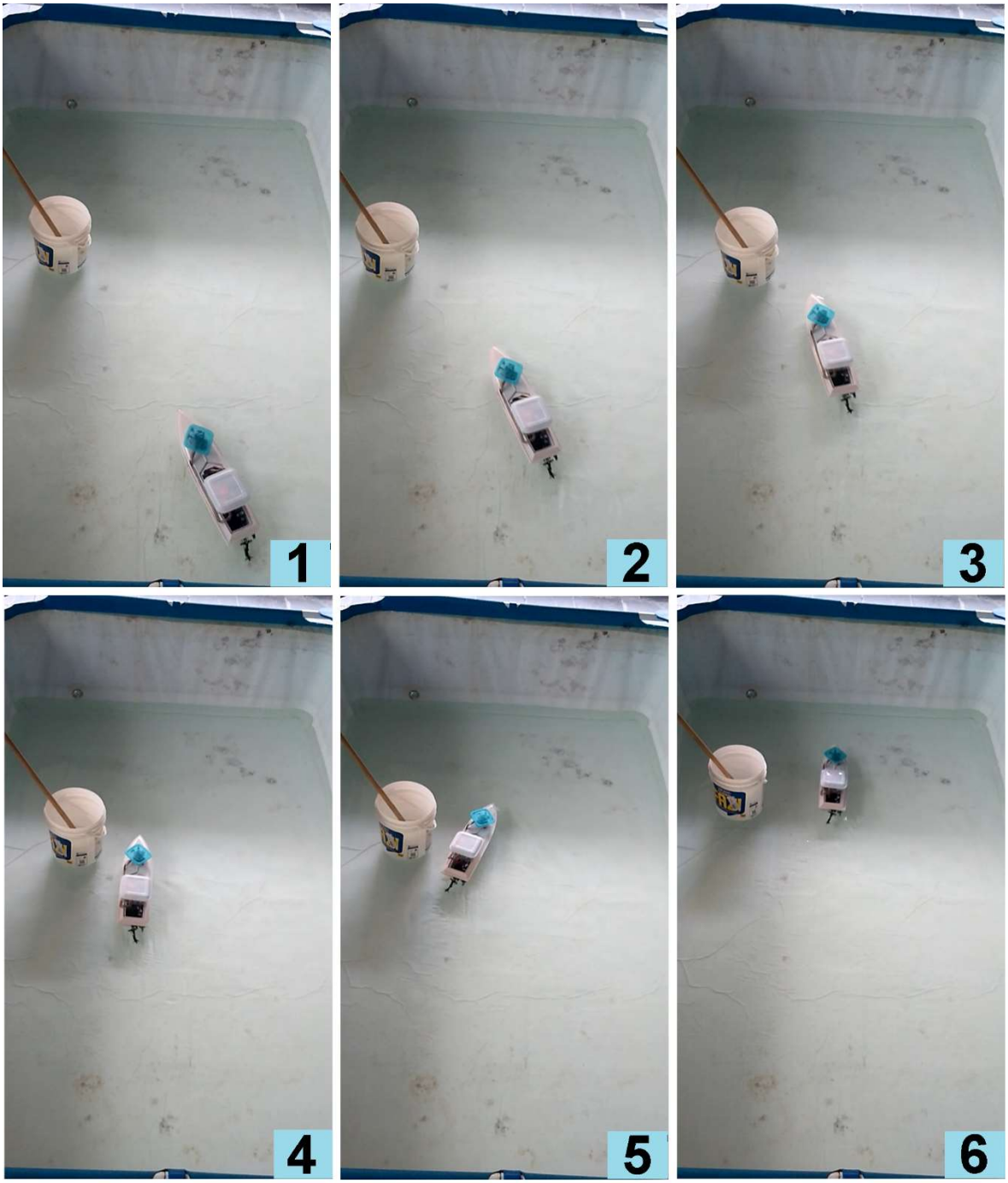


Figura 20 - Experiência do controle de desvio de obstáculo lateral

A Figura 21 demostra as variáveis em ação no controle de desvio de obstáculos laterais. É possível notar que a variável que comanda o leme alcança valores máximos ou mínimos em alguns pontos do percurso, neste momento é quando o objeto é identificado próximo à embarcação. Após o sistema se afastar do objeto, o leme volta ao controle de rumo.

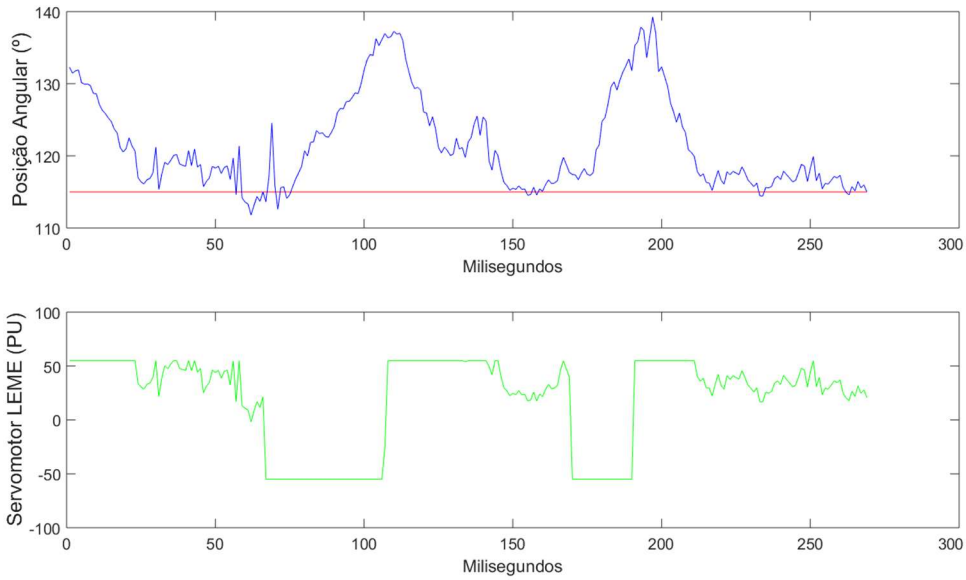


Figura 21 - Resultado do controle de desvio de obstáculos laterais

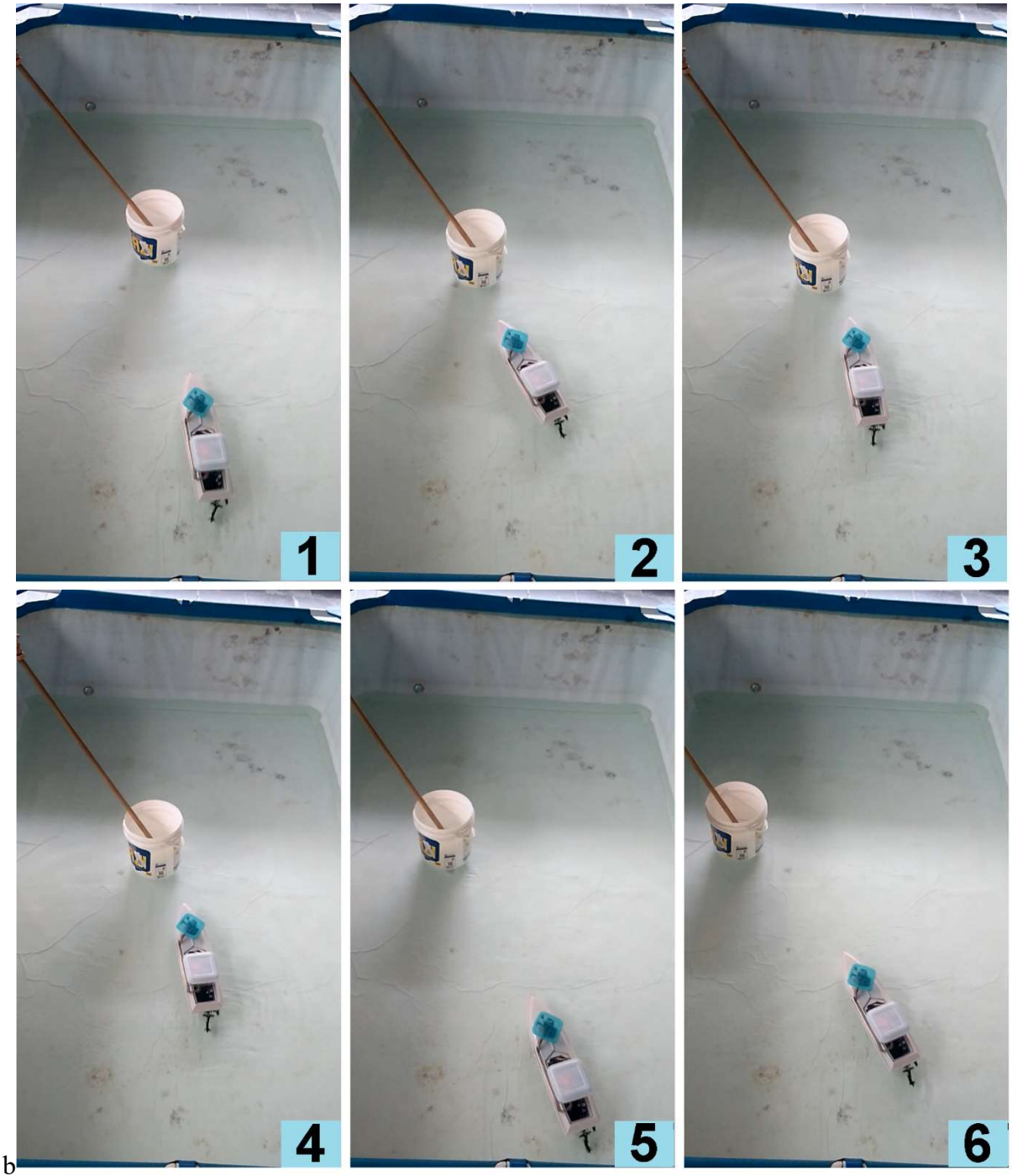
A Figura 21 demostra também uma ligeira fuga do rumo enquanto o sistema esquiva de um obstáculo e a retomada do rumo após o objeto não ser mais detectado.

5.3 Experimento 3 – controle de desvio de obstáculos frontal e traseiro

Neste experimento o objeto foi posicionado na frente do barco. A embarcação responde invertendo o sentido do motor, e invertendo também a direção do leme, fazendo a embarcação perseguir o rumo mesmo enquanto desvia do obstáculo frontal. A inversão do sinal de controle do leme possibilita o alinhamento com o rumo enquanto o barco vai para trás.

Em testes livres na piscina, o barco às vezes alcança o canto, momento em que a inversão dos sinais auxilia o movimento do barco em fazer um movimento similar à baliza de um carro. A barco esquiva da quina da piscina com movimentos para frente (quando não detecta a borda) e para trás (quando detecta a borda) invertendo a direção do leme e evitando o canto.

Na Figura 22 é possível notar o momento em que o barco muda o sentido do motor para se afastar do objeto logo a sua frente sem perder a referência do ângulo de rumo.



b

Figura 22 - Experiência do controle de desvio de obstáculo frontal

Na Figura 23 é possível notar o momento em que o sinal que controla o sentido do motor (em rosa) muda seu valor, este é o momento em que o obstáculo frontal foi detectado. É possível notar que mesmo tendo o sentido do motor invertido, o rumo continua sendo seguido, exatamente como o previsto, a embarcação persegue o rumo não importando o sentido do motor.

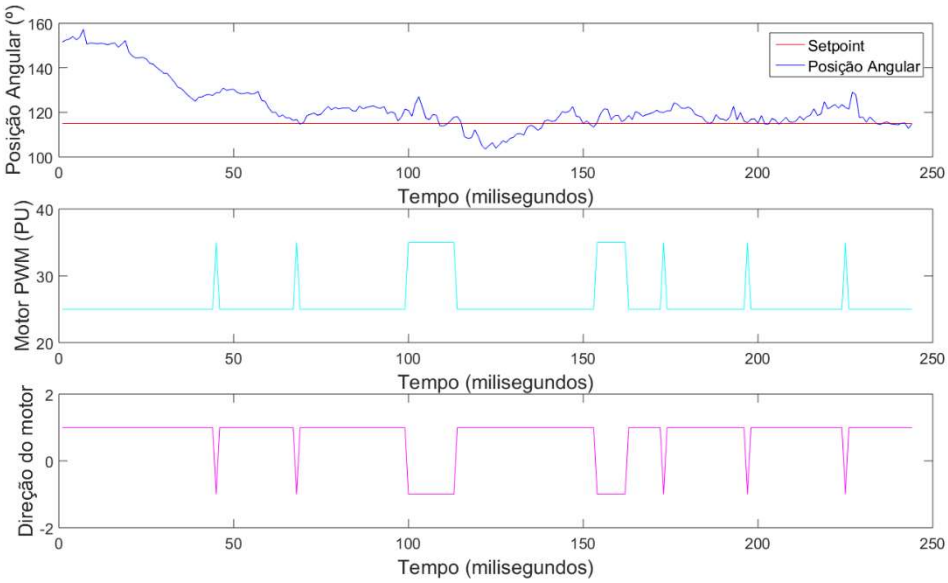


Figura 23 - Resultado do controle de desvio de obstáculo frontal

O segundo sinal indica o valor *PWM* de acionamento do motor e demostra valores baixos que saturam rapidamente. Este sinal é limitado para evitar altas velocidades do motor. É possível notar que ao limite de velocidade é maior para o sentido de ré, pois o barco apresenta maior resistência de movimento nesta direção.

O terceiro sinal indica o sentido do motor, sinal este que fica negativo quando o barco detecta um obstáculo à frente. É possível notar que mesmo com o sentido de movimento invertido, o barco ainda persegue o ângulo de rumo.

6. Conclusão

Este projeto acadêmico teve como objetivo a construção funcional de uma pequena embarcação autônoma de baixo custo capaz de seguir um rumo e desviar de obstáculos. A proposta foi desenvolvida utilizando um barco de nautimodelismo, um microcontrolador e diversos sensores embarcados.

O objetivo de seguimento de rumo foi alcançado: com a instalação do sensor magnetômetro, que mede o campo magnético da terra e serviu para ler a posição angular do barco; a instalação de um sensor girômetro para a leitura da velocidade angular do barco; e o desenvolvimento de uma estratégia de controle para acionar os atuadores do barco.

O objetivo de desvio de obstáculos laterais foi alcançado com a instalação de dois sensores de distância e uma estratégia de controle anticolisão programada para ler os sensores e atuar de maneira dominante, em relação ao controle de rumo, sobre os atuadores do barco.

O objetivo de desvio de obstáculos frontal e traseiro foi alcançado com a instalação de dois sensores de distância, frontal e traseiro, e uma estratégia de controle que evite obstáculos e ao mesmo tempo mantenha o barco em movimento para seguir o rumo desejado.

O desempenho foi avaliado experimentalmente em uma piscina de 3 mil litros, motivo pelo qual a velocidade do barco foi limitada para melhor demonstração da trajetória. Em uma piscina ou lago de maior superfície é esperado um melhor desempenho do sistema, possibilitando uma maior velocidade do barco, maior trajetória e maior número de obstáculos.

Para projetos futuros utilizando este sistema, este trabalho recomenda a adição de um sistema de comunicação com um computador, um módulo *WIFI* por exemplo, para efetuar comunicação com um computador em terra e possibilitar o envio de comandos e melhor transferência de dados. Testes em ambientes aquáticos maiores também são indicados pois exigiriam um controlador PID completo.

Referências Bibliográficas

ARDUINO. **Arduino Build Process**, 2012. Disponível em: <https://arduino.github.io/arduino-cli/0.21/sketch-build-process/>

ARDUINO. **Arduino – Arduino Board Mega**, 2009. Disponível em: <https://docs.arduino.cc/hardware/mega-2560>

ARRUDA, José Wilton Oliveira; 2012 - **Controle Automático de Rumo de uma Embarcação de superfície não tripulada**, 2012. Projeto de Graduação em Engenharia Eletrônica – UERJ.

BOYLESTAD, R.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 6. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora S.A., 1998.

BUORO, Angelo Sabbatini; 2013 - **Controle dos motores e acionamento sem fios de uma pequena embarcação**, 2013. Projeto de Graduação em Engenharia Eletrônica - UERJ.

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

CUNHA, J. P. V. S. **Embarcações Não Tripuladas para Monitoração Ambiental e Defesa**. Rio de Janeiro: FAPRJ, 2010.

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.

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

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

NAEEM, 2011 - **COLREGs-based collision avoidance strategies for unmanned surface vehicles** - Mechatronics - International Federation of Automatic Control. Published by Elsevier Ltd – 2011.

OGATA, K. - **Engenharia de Controle Moderno**, Prentice-Hall, 4ª. ed., 2004.

POMILIO, José Antenor, 2014 - **Eletrônica de potência**, livro disponível em: http://www.dsce.fee.unicamp.br/~antenor/elpot.html

SCHULTZE, Hendrik Jürgen , 2012 - **Projeto e Construção de uma Embarcação Teleoperada**, 2012. Projeto de Graduação em Engenharia Eletrônica - UERJ.

YAN, Ru-jian at al. 2010 - **Development and Missions of Unmanned Surface Vehicle. Journal of Marine Science and Application**. v. 9, issue 4, p. 451-45

ZHANG, Youmin, 2016 - **Unmanned surface vehicles: An overview of developments and challenges** - Annual Reviews in Control - International Federation of Automatic Control. Published by Elsevier Ltd - 2016.

Apêndice A - Programa Gravado no Microcontrolador

Tabela 6 - Programa gravado no Arduino

Programa gravado no Arduino

```
#include <Wire.h>
#include <LSM303.h> // Magnetometro
#include <MPU6050_tockn.h> // acelerometro + giroscopio
#include "Adafruit_VL53L0X.h" // laser

#define LOX1_ADDRESS 0x30
#define LOX2_ADDRESS 0x31
#define LOX3_ADDRESS 0x33
#define LOX4_ADDRESS 0x34
#define SHT_LOX1 11
#define SHT_LOX2 12
#define SHT_LOX3 13
#define SHT_LOX3 14

MPU6050 mpu6050(Wire);
LSM303 compass;

int i = 0;  //variavel para repeticao FOR

const int leme_pin = 10; //pino para acionar o leme

const int motor_pin = 5; //pino para acionar o Motor

double medida_bussola, ang;
int sensor1,sensor2,sensor3;
Adafruit_VL53L0X lox1 = Adafruit_VL53L0X(); // objetos para sensor de distancia vl53l0x
Adafruit_VL53L0X lox2 = Adafruit_VL53L0X();
Adafruit_VL53L0X lox3 = Adafruit_VL53L0X();
```

```
Adafruit_VL53L0X lox4 = Adafruit_VL53L0X();

VL53L0X_RangingMeasurementData_t measure1; // variaveis que armazenam as medidas de distancia
VL53L0X_RangingMeasurementData_t measure2;
VL53L0X_RangingMeasurementData_t measure3;
VL53L0X_RangingMeasurementData_t measure4;

double distancia_bb; //VARIÁVEL DISTANCIA BOMBORDO
double distancia_eb; //VARIÁVEL DISTANCIA ESTIBORDO
double distancia_ff; //VARIÁVEL DISTANCIA FRENTE
double distancia_tt; //VARIÁVEL DISTANCIA FRENTE

//--------------------------------------------
// variaveis PID LEME
unsigned long time_now=0, last_time=0;
double last_input=0;
int time_change=0, sample_time = 60;  //1 sec
double input, output;

//  variaveis PID - colisao
double kp_c=7.0, ki_c=0.0, kd_c=0.8, error_c;
double i_term_c=0, d_input_c=0;
double setpoint_c=0.0;

//  variaveis PID - bussola
double kp_b=7.0, ki_b=0.0, kd_b=0.3, erro_b;
double i_term_b=0, d_input_b=0;
double setpoint_b = 115;
//--------------------------------------------
// variaveis PID MOTOR
unsigned long time_now_m=0, last_time_m=0;
double last_input_m=0;
```

```
int time_change_m=0, sample_time_m = 60;  //1 sec
double input_m, output_m;

//  variaveis PID - motor
double kp_m=6.0, ki_m=0.0, kd_m=0.7, error_m;
double i_term_m=0, d_input_m=0;
double setpoint_m=40.0;

//  variaveis para limitar o movimento do MOTOR
double out_min_m=-35.0, out_max_m=25.0;
int motor_pwm = 2;  //Velocidade do motor
double sentido_motor = 20.000000;  // 1-frente 0-traz
//---------------------------------------------
int controle = 0;

//  variaveis para limitar o movimento do leme
double out_min=-55.0, out_max=55.0;
double out_leme_pwm=145.0;

double accX, accY, accZ, gyroX, gyroY, gyroZ, accAngleX, accAngleY, gyroAngleX,
gyroAngleY, gyroAngleZ, angleX, angleY, angleZ;

//=============================================================================
=============================================================================
void setID() {
  // todos desligados
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  delay(10);
  // todos ligados
//  digitalWrite(SHT_LOX1, HIGH);
//  digitalWrite(SHT_LOX2, HIGH);
```

```
//  digitalWrite(SHT_LOX3, HIGH);
  delay(10);
  // ativando LOX1 e resetando LOX2,LOX3
  digitalWrite(SHT_LOX1, HIGH);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  // iniciando LOX1
  if(!lox1.begin(LOX1_ADDRESS)) {
    Serial.println(F("Failed to boot first VL53L0X"));
    while(1);  }
  delay(10);
  // ativando LOX2
  digitalWrite(SHT_LOX2, HIGH);
  digitalWrite(SHT_LOX3, LOW);
  delay(10);
  //iniciando LOX2
  if(!lox2.begin(LOX2_ADDRESS)) {
    Serial.println(F("Failed to boot second VL53L0X"));
    while(1);  }
  delay(10);
  // ativando LOX3
  digitalWrite(SHT_LOX3, HIGH);
  delay(10);
//  //iniciando LOX3
  if(!lox3.begin(LOX3_ADDRESS)) {
    Serial.println(F("Failed to boot third VL53L0X"));
    while(1);  }
}
//=========================================================================
=========================================================================
void read_acelerometer() {
accX = mpu6050.getAccX();
accY = mpu6050.getAccY();
accZ = mpu6050.getAccZ();
```

```
  gyroX = mpu6050.getGyroX();
  gyroY = mpu6050.getGyroY();
  gyroZ = mpu6050.getGyroZ();
accAngleX = mpu6050.getAccAngleX();
accAngleY = mpu6050.getAccAngleY();
  gyroAngleX = mpu6050.getGyroAngleX();
  gyroAngleY = mpu6050.getGyroAngleY();
  gyroAngleZ = mpu6050.getGyroAngleZ();
angleX = mpu6050.getAngleX();
angleY = mpu6050.getAngleY();
angleZ = mpu6050.getAngleZ();
}
//====================================================================
====================================================================
void imprimir() {
Serial.print("Buss: ");
Serial.print(medida_bussola);
  Serial.print("\t Sensor 1: ");
  Serial.print(sensor1);
  Serial.print("\t Sensor 2: ");
  Serial.println(sensor2);
Serial.print("accX : ");Serial.print(accX);
Serial.print("\taccY : ");Serial.print(accY);
Serial.print("\taccZ : ");Serial.println(accZ);
  Serial.print("gyroX : ");Serial.print(gyroX);
  Serial.print("\tgyroY : ");Serial.print(gyroY);
  Serial.print("\tgyroZ : ");Serial.println(gyroZ);
Serial.print("accAngleX : ");Serial.print(accAngleX);
Serial.print("\taccAngleY : ");Serial.println(accAngleY);
  Serial.print("gyroAngleX : ");Serial.print(gyroAngleX);
  Serial.print("\tgyroAngleY : ");Serial.print(gyroAngleY);
  Serial.print("\tgyroAngleZ : ");Serial.println(gyroAngleZ);
Serial.print("angleX : ");Serial.print(angleX);
```

```
Serial.print("\tangleY : ");Serial.print(angleY);
Serial.print("\tangleZ : ");Serial.println(angleZ);
  Serial.println();
  Serial.println();
}
//================================================================
================================================================
void controle_leme(){
  input = (medida_bussola);  // leitura do magnetômetro
    //Calculando as variaveis de erro
    erro_b = - setpoint_b + input;
    d_input_b = (acc_veloc_ang);  //leitura do aceletômetro, velocidade angular
    //Calcula PD
    output = ((kp_b*erro_b) + (kd_b*d_input_b));
    //Calcula limites de output
    if(output > out_max){
      output = out_max;}
    else{
      if(output < out_min){
        output = out_min;}
    }
  }
  //out_leme_pwm = 145 + output;  // 145 é a posição central do leme
}  //out_leme_pwm = 145 + output;
}
//================================================================
================================================================
void controle_desvio(){
  input = (distancia_bb - distancia_eb);
    //Calculando as variaveis de erro
```

```
   error_c = setpoint_c - input;
   d_input_c = (acc_veloc_lin);  //leitura do aceletômetro, velocidade linear
   //Calcula PD
   output = ((kp_c*error_c) + (kd_c*d_input_c));
   //Calcula limites de output
   if(output > out_max){
     output = out_max;}
   else{
     if(output < out_min){
       output = out_min;}
   }
  }
  //out_leme_pwm = 145 + output;
}//=====================================================================
=====================================================================
void controle_motor(){
  input_m = (distancia_ff – distancia_tt);
   //Calculando as variaveis de erro
   error_m = - setpoint_m + input_m;
   d_input_m = acc_X;  //leitura do acelerometro
   //Calcula PID
   output_m = ((kp_m*error_m) + (kd_m*d_input_m));
   //Calcula limites de output
   if(output_m > out_max_m){
     output_m = out_max_m;}
   else{
     if(output_m < out_min_m){
       output_m = out_min_m;}
   }
```

```
    last_input_m = input_m;
    last_time_m = time_now_m;
  }
  //pwm de saida do motor nao pode ser valor negativo
  if(output_m < 0){
    output_m = output_m * (-1);
    sentido_motor = 0.0;}
  else {
    sentido_motor = 20.0;
  }
  motor_pwm = output_m;
}//##############################################################################
##########################################################################
void setup() {
  delay(10000);
  Serial.begin(9600);
  // inicia o leme
  pinMode(leme_pin, OUTPUT); // configura pino como saída
  //inicia Motor
  pinMode(motor_pin,OUTPUT);
  pinMode( 4,OUTPUT);
  pinMode( 7,OUTPUT);
  //pinMode(11,OUTPUT);
  digitalWrite(4,LOW);  //indica a direcao de giro do motor
  digitalWrite(7,HIGH);  //indica a direcao de giro do motor
  for (i = 0; i <= motor_pwm; i++){
    analogWrite(motor_pin,i);
    delay(80);
    }
  // inicia os sensores de distancia
  pinMode(SHT_LOX1, OUTPUT);
```

```
  pinMode(SHT_LOX2, OUTPUT);
  pinMode(SHT_LOX3, OUTPUT);
  delay(200);
  digitalWrite(SHT_LOX1, LOW);
  digitalWrite(SHT_LOX2, LOW);
  digitalWrite(SHT_LOX3, LOW);
  setID();
delay(200);
  // inicia a bussola
  Wire.begin();
  compass.init();
  compass.enableDefault();
  delay(200);
  compass.m_min = (LSM303::vector<int16_t>){-32767, -32767, -32767};
  compass.m_max = (LSM303::vector<int16_t>){+32767, +32767, +32767};
delay(200);
  // inicia acelerometro, giroscopio
//  mpu6050.begin();
//  mpu6050.calcGyroOffsets(true);
  last_time=millis();
}

//#############################################################################
#############################################################################
void loop() {
  //Acionamento do Motor
  //analogWrite(motor_pin,motor_pwm);

  mpu6050.update();

  // leitura da bussola
  compass.read();
  medida_bussola = compass.heading();
```

```
  //ang = cos(compass.heading() * 3.1416/180);

  //  leitura dos sensores de distancia
  lox1.rangingTest(&measure1, false); // 'true' para debug do pino
  lox2.rangingTest(&measure2, false); // 'true' para debug do pino
  lox3.rangingTest(&measure3, false); // 'true' para debug do pino
  distancia_eb = (measure1.RangeMilliMeter)/10;
  distancia_bb = (measure2.RangeMilliMeter)/10;
  distancia_ff = (measure3.RangeMilliMeter)/10;
//  printdistancia(distancia_bb, distancia_eb);

  // leitura dos dados do acelerometo e giroscopio
  //read_acelerometer();

//_____________________________________________________ CONDICOES
LEME
  if((distancia_bb < 40) or (distancia_eb < 40)){
    controle = 1;}
  else{
    controle = 2;}

//_____________________________________________________ controle LEME
switch (controle) {
  case 1:
    controle_desvio ();
  break;
  case 2:
    controle_leme ();
  break;
}
//_____________________________________________________ fim controle
LEME
```

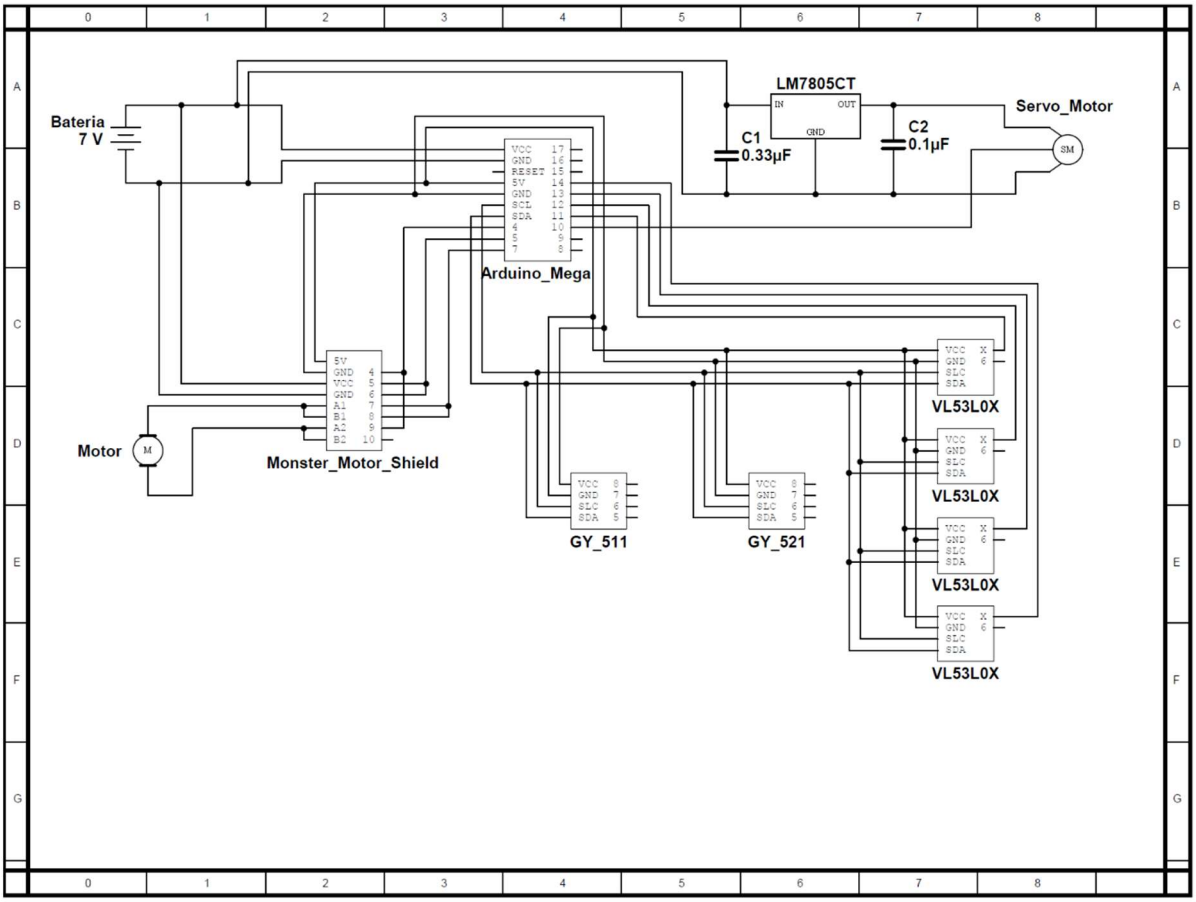
```
//out_leme_pwm=140.0;

//_____________________________________________________________ controle
MOTOR
controle_motor ();

//trata o sentido do motor
if (sentido_motor == 20.0){
  digitalWrite(4,LOW);  //indica a direcao de giro do motor
  digitalWrite(7,HIGH);  //indica a direcao de giro do motor
  out_leme_pwm = 145 + output;
}
else{
  digitalWrite(4,HIGH);  //indica a direcao de giro do motor
  digitalWrite(7,LOW);  //indica a direcao de giro do motor
  out_leme_pwm = 145 - output;
}
analogWrite(motor_pin,motor_pwm);
analogWrite(leme_pin, out_leme_pwm); // aciona o leme com o valor analógico

//delay(10);
}
```

Apêndice B - Diagrama das Interconexões dos Circuitos Embarcados



Apêndice C – Montagem final do barco

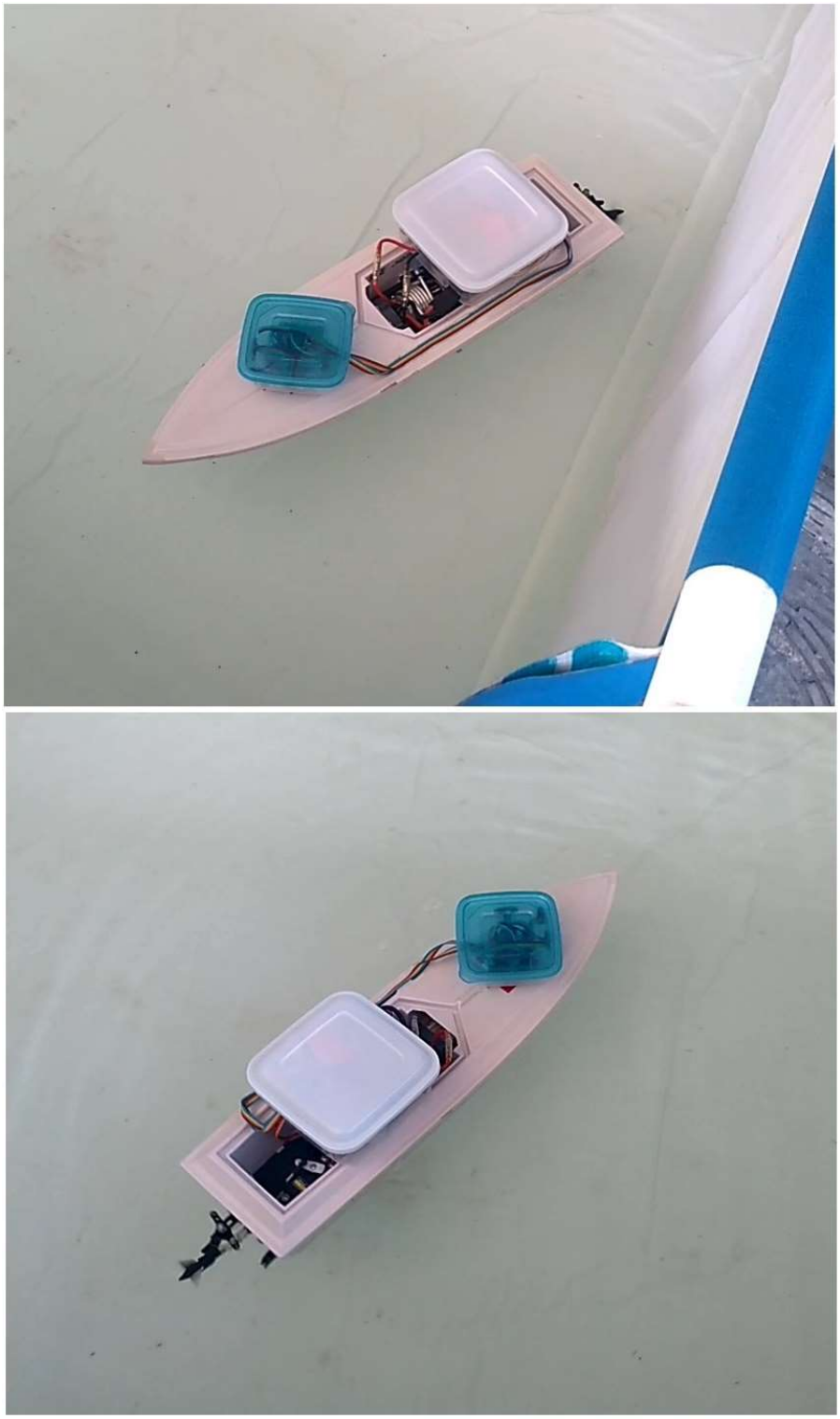


Figura 24- Montagem final do barco, proa e popa.