Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Angelo Sabbatini Buoro

Controle dos motores e acionamento sem fios de uma pequena embarcação

Rio de Janeiro

2013

Angelo Sabbatini Buoro

Controle dos motores e acionamento sem fios de uma pequena embarcação

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

Orientador: Prof. Dr. Paulo Bulkool Batalheiro

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

Rio de Janeiro

2013

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

B944 Buoro, Angelo Sabbatini.
Controle dos motores e acionamento sem fios de uma pequena embarcação / Angelo Sabbatini Buoro. – 2013.
79f.

Orientador: Paulo Bulkool Batalheiro.
Projeto Final (Graduação) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.
Bibliografia 66-68

1. Engenharia eletrica. 2. Sistemas de comunicação sem fio. 3. MATLAB (Programa de computador). I. Batalheiro, Paulo Bulkool. II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 621.3

Assinatura

Data

Angelo Sabbatini Buoro

Controle dos motores e acionamento sem fios de uma pequena embarcação

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

Aprovado em 17 de abril de 2013.

Banca Examinadora:

_____

Prof.Dr. Paulo Bulkool Batalheiro (Orientador)

Faculdade de Engenharia - UERJ

_____

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

Faculdade de Engenharia - UERJ

_____

Prof. Dr. Frederico Celestino Pontes

Faculdade de Engenharia - UERJ

Rio de Janeiro

2013

AGRADECIMENTOS

Aos meus professores e orientadores, Paulo Bulkool e José Paulo, pelo apoio e idéias para que esse projeto se concretizasse

Ao professor Frederico, pois desde que cursei eletrônica 1 , ele responde todo e qualquer e-mail com dúvidas e assuntos relacionados à eletrônica.

A todo o corpo docente da Faculdade de Engenharia da Universidade do Estado do Rio de Janeiro, por me capacitar como Engenheiro.

Ao professor Lisandro Lovisolo por ceder parte do material utilizado nos testes do projeto.

Aos meus colegas de faculdade, que lutaram junto comigo ao longo desses anos na universidade.

Ao meu pai, por ter me fornecido uma ajuda substancial nos períodos finais do curso.

Ao meu irmão pelas conversas sobre matemática e física.

A todos os meus amigos de infância e adolescência, que sempre entenderam (eu acho) quando não podia atender aos seus convites, pois ou tinha que estudar ou tinha prova no dia seguinte.
Por último e não menos importante, minha amiga, namorada e companheira de longa data, Patrícia Lins, que me apóia incondicionalmente em minhas (algumas vezes sensatas) escolhas desde antes de entrar na universidade, seu pai, João Augusto, por nossas longas conversas sobre a engenharia e a profissão e o resto de sua família por todo apoio que me deram ao longo desses anos.

“Existe uma teoria que diz que, se um dia alguém descobrir exatamente para que serve o Universo e por que ele está aqui, ele desaparecerá instantaneamente e será substituído por algo ainda mais estranho e inexplicável...
Existe uma segunda teoria que diz que isso já aconteceu.”

Douglas Adams

RESUMO

BUORO, Angelo Sabbatini. *Controle dos motores e acionamento sem fios de uma pequena embarcação.* 76f. Projeto Final (Graduação em Engenharia Elétrica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

Neste projeto é desenvolvido o controle e comunicação com uma pequena embarcação sem fio, utilizando um protocolo para comunicação sem fio denominado Zigbee. A pequena embarcação conta com um microcontrolador e um circuito acionador de motor comandado pelo computador em terra utilizando o software Matlab, em que é possível enviar comando e ler variáveis das portas da plataforma microcontrolada. O desempenho do motor foi avaliado experimentalmente. Foi desenvolvida também uma interface gráfica baseada nos recursos disponíveis no Matlab para controle da embarcação.

Palavras-chave: Zigbee, comunicação sem fios, embarcação teleoperada, acionamento de motor, Matlab, Interface gráfica com o usuário (GUI).

ABSTRACT

This project developed the control and communication wireless between a small boat and a PC, using a wireless protocol called Zigbee. The small boat has a microcontroller unit and a motor driver controlled by a computer with the Matlab software, with which is possible to read and write directly to microcontroller I/O ports. The motor´s performance was evalueted experimentally. Also was designed a graphical user interface to Interact remotely with the boat.

Keywords: Zigbee, wireless communication, remotely operated surface vessel, motor driver, matlab, graphical user interface.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Quando é necessário avaliar alguma característica de algum projeto de engenharia que tenha grandes dimensões, onde não seja viável economicamente construir um protótipo em tamanho real para a realização de todos os estudos e previsão de comportamento, ou pela própria limitação dos estudos teóricos de um determinado projeto onde se faça necessário verificar seu comportamento experimental antes da construção, pode-se utilizar um modelo físico, que possua as mesmas proporções do modelo real. A esse tipo de aplicação damos o nome de modelo reduzido ou modelo em escala reduzida. Isto é, um modelo que mantém o formato geométrico e obedece a proporções de escala definida [1].

As principais aplicações desses modelos estão na área de construção naval, que usa diversas escalas para esses modelos, desde pequenos exemplares para ensaios relacionados à ancoragem ou como o navio se comporta amarrado em um porto, em exploração de petróleo *off-shore (*em alto-mar), com o intuito de estudar como o modelo de plataforma se comporta sobre o efeito de ondas e correntezas [2]. Na figura 1 pode-se ver uma mini-plataforma sendo estudada em um tanque do IPT (Instituto de Pesquisa Tecnológica).

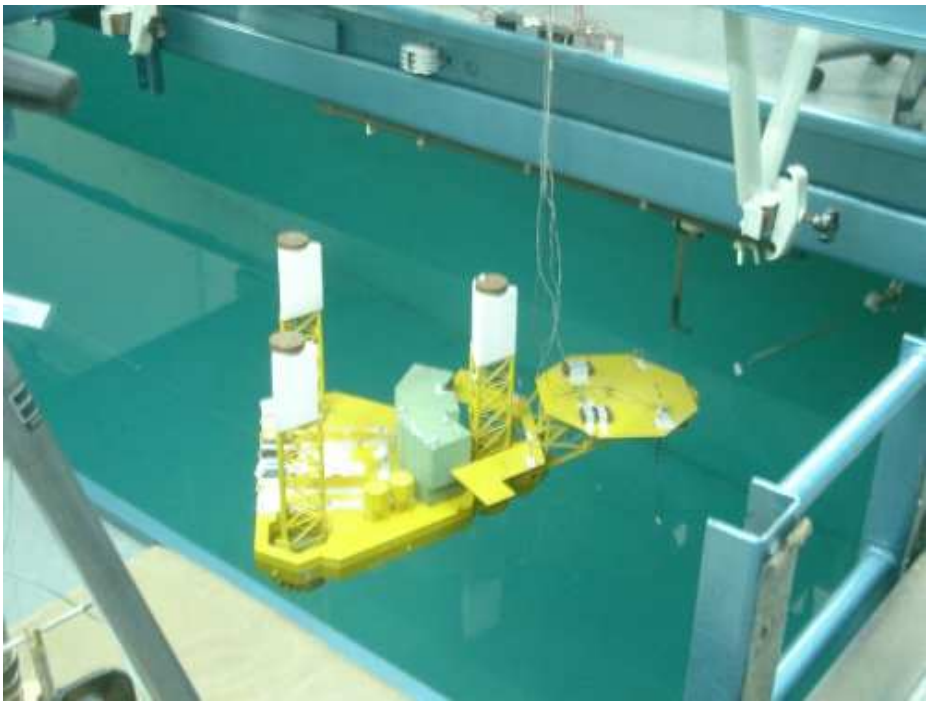

Figura 1: Exemplo de uso de modelo reduzido: modelo de uma plataforma para exploração de petróleo em um tanque.

Outra aplicação para modelos reduzidos que é muito frequente no Brasil é em grandes obras civis, como por exemplo, as usinas hidrelétricas. Os modelos são construídos com os mesmos materiais que serão utilizados na obra real, como a mesma água do rio em que a usina estará instalada, o mesmo solo e o concreto, representando uma estrutura funcional da usina. Estes estudos servem para avaliar a capacidade de escoamento do vertedouro, canal de fuga, capacidade do reservatório, estabilidade da estrutura civil e comportamento hidráulico [3]. Os modelos são usados desde o início do projeto básico, bem como ao longo do projeto executivo. Caso seja detectada alguma falha no projeto inicial, podem ser impostas modificações para adequar o projeto executivo. Na figura 2 podemos observar o modelo reduzido do vertedouro da usina de Itaipu.

Figura 2: Modelo reduzido do vertedouro da usina hidrelétrica de Itaipú.

Os fabricantes de turbinas também utilizam modelos reduzidos para avaliar o rendimento da máquina e se a mesma atenderá as especificações de queda e vazão de água da usina em questão. A figura 3 apresenta um exemplo de turbina em modelo reduzido utilizado para ensaios e teste hidráulicos da UHE Colíder [4].

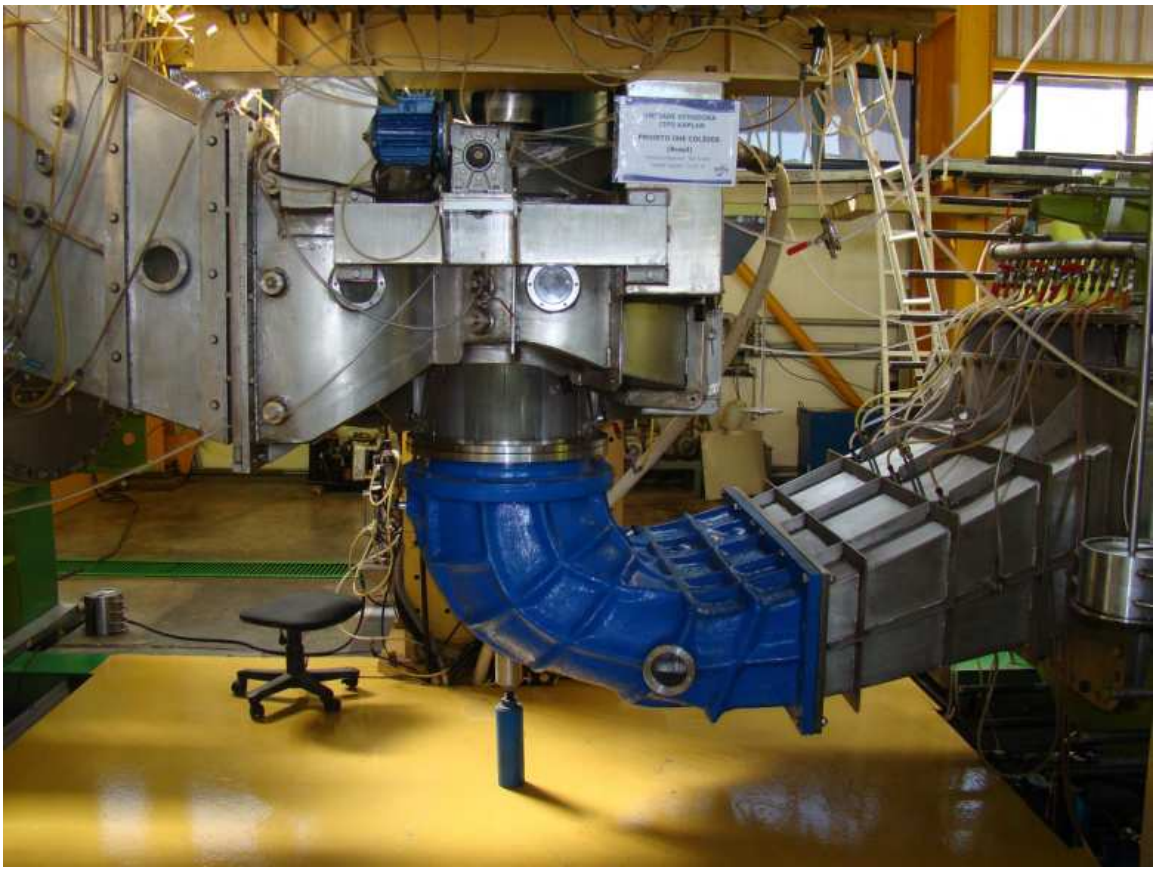

Figura 3: Modelo reduzido de uma turbina da UHE Colíder.

Objetivo

O objetivo deste projeto de graduação é adaptar uma pequena embarcação pelo do uso de um microcontrolador instalado a bordo para receber comandos de um computador em terra, por meio de uma comunicação sem fio, sendo possível o acionamento e o controle dos motores diretamente pelo usuário (controle manual) ou, num projeto futuro, o controle realizado por um *software* de computador que receba informações de posição do barco, controlando-o de acordo com de decisões das estratégias de controle.

Descrição do sistema de comunicação sem fio do barco com o computador

O sistema de comunicação sem fio é amplamente utilizado em sistemas que necessitam estar longe de alguma fonte de alimentação ou demandem baixo consumo e imunidade a ruído. O padrão *Zigbee* foi desenvolvido para ser implementado em ambientes industriais, onde nem sempre o acesso é fácil e haja necessidade de um sistema robusto, seguro e confiável para o controle e a aquisição de dados de sensores instalados ao longo da planta [5].

A figura 4 apresenta um diagrama de blocos básico que ilustra o sistema de comunicação entre a embarcação e o computador.

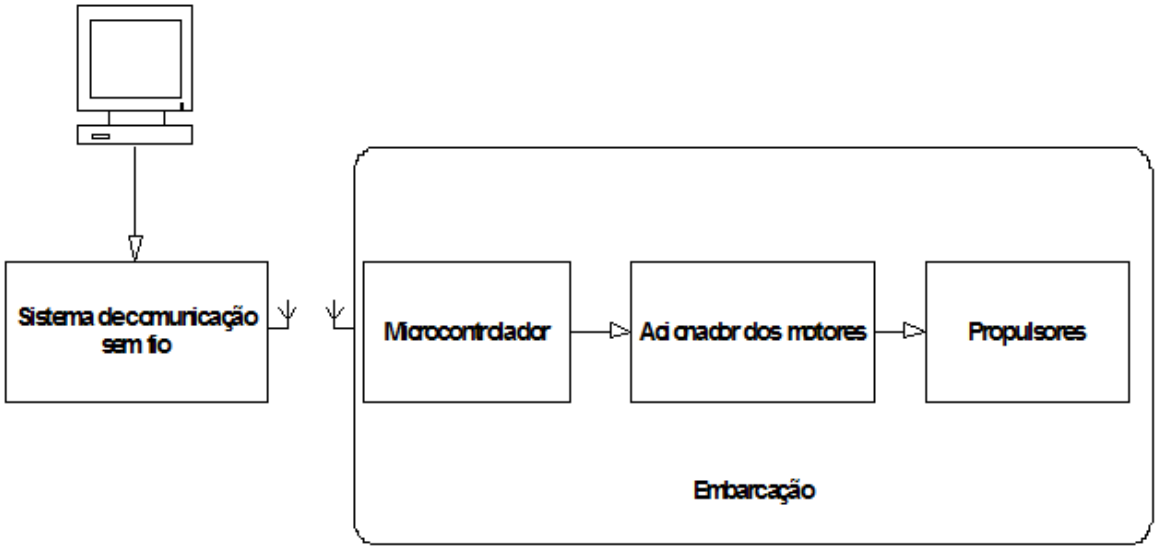

Figura 4: Diagrama de blocos do sistema de comunicação.

Inicialmente o controle será realizado em malha aberta, i.e, a plataforma de controle enviará comando para o barco sem um meio para o retorno da variável de controle (no caso, a posição da embarcação no tanque de teste). A idéia para trabalhos futuros é utilizar um sistema de posicionamento por câmeras, mas ao invés de utilizar uma câmera tradicional e um sistema de aquisição de posição por diferença de luminância e cálculo da geometria do barco (que fora usado anteriormente) [21] será utilizado um conjunto de câmeras especiais que captam o infravermelho refletido por um determinado material. Esse tipo de câmera é muito empregado no estudo da mecânica do movimento de animais, de seres humanos e também indústria cinematográfica e de desenvolvimento de jogos, pois permite capturar movimentos

realizados por um ator, que posteriormente será tratado no computador e servirá para animar personagens, tanto em filmes quanto em jogos.

Na figura 5 mostra um sistema semelhante ao que será utilizado futuramente para a aquisição da posição da embarcação.

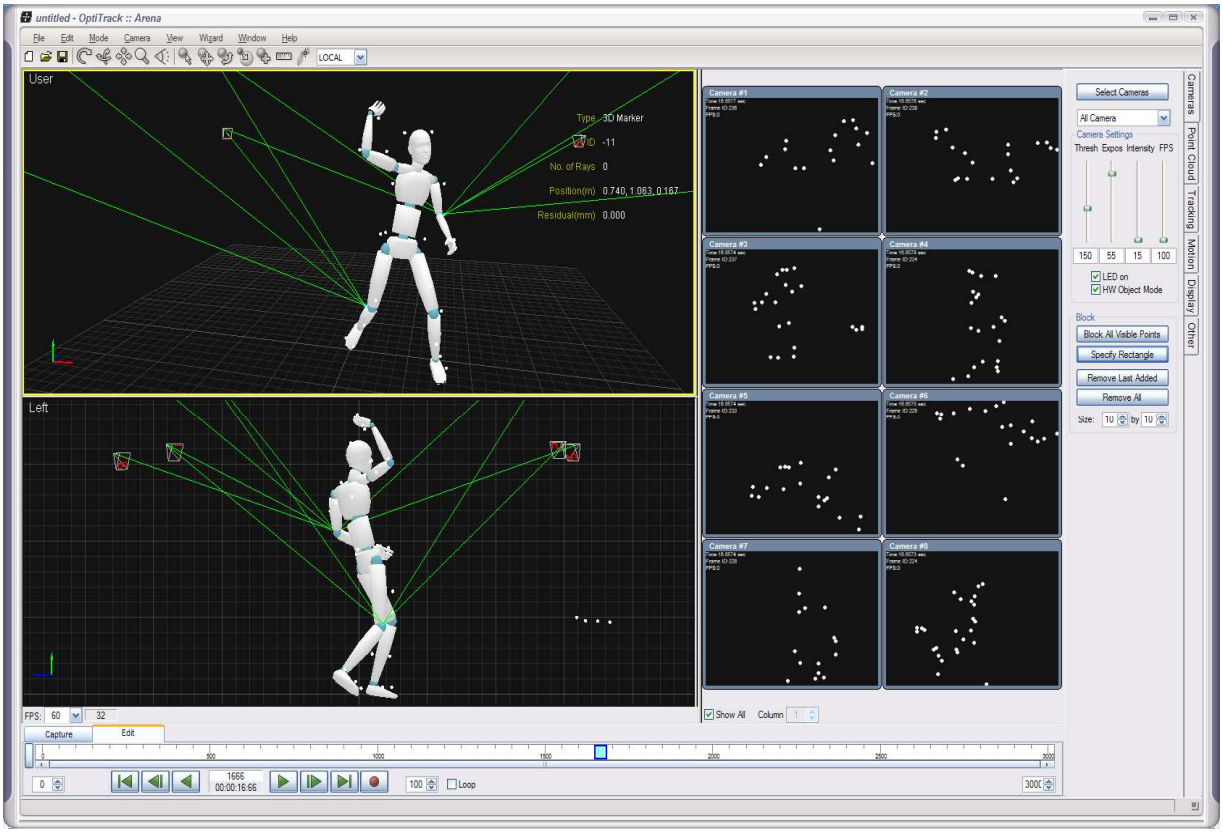

Figura 5: Sistema de captura de posição / movimento.

A figura 6 apresenta um diagrama de blocos básico que ilustra o sistema de comunicação entre a embarcação, o computador e o sistema de aquisição de posição proposto.

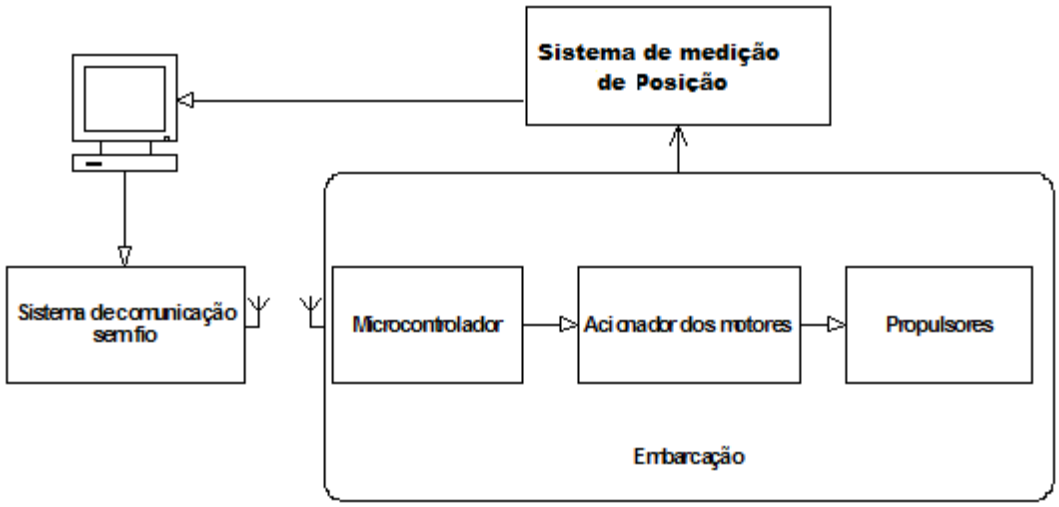

Figura 6: Diagrama de blocos do sistema de comunicação e do sistema de medição.

Organização do texto.

A organização do texto foi estabelecida como segue. No capitulo 1 temos uma breve introdução sobre a plataforma microcontrolada e os sensores que possam vir a ser utilizados na embarcação. O capitulo 2 traz um breve estudo dos sistemas de comunicação inicialmente propostos para comunicação sem fio e um maior detalhamento do sistema de comunicação sem fio selecionado para o projeto da comunicação do computador com a embarcação. O capitulo 3 aborda a comunicação entre a embarcação e o *software* utilizado, o Matlab®. O capitulo 4 apresenta os passos seguidos ao longo do projeto para a instalação de todos os componentes necessários. No capitulo 5 temos os teste e medições realizados nos motores. O capitulo 6 trata da interface gráfica de usuário criada para o controle da embarcação e por fim, no capítulo 7 a conclusões e sugestões para usos futuros do sistema em questão. O apêndice A contará com o programa de interface gráfica de usuário codificado no Matlab utilizado para os testes.

1 PLATAFORMA MICROCONTROLADA, SENSORES E SEUS ACESSÓRIOS

Os microcontroladores (µC, uC ou MCU, do inglês *microcontroller*) são dispositivos eletrônicos que possuem em um único encapsulamento uma unidade central de processamento, memória para armazenamento de um programa a ser executado e periféricos, interface com osciladores internos, relógio de tempo real (em inglês *real time clock*) e portas de entrada e saída, responsáveis pela interação do MCU com o mundo exterior [6]. São muito utilizados para os mais diversos fins, tanto em bens de consumo (geladeiras, microondas, equipamentos de ar-condicionado), como em brinquedos, automóveis e equipamentos hospitalares e industriais.

A função do microcontrolador neste projeto é agregar o *drive* (acionador) dos motores e o sistema de comunicação sem fio.

Futuros sensores poderão ser instalados, tais como acelerômetros, magnetômetros, bússolas e até mesmo câmeras para fotografia. Pela comunicação serial, é possível enviar as informações de todos os sensores remotamente montados para o computador. No caso da câmera, enviar as imagens através do sistema de comunicação sem fio para o computador exibindo-as para o controlador da embarcação localizado remotamente.

Para este projeto, foi escolhida uma plataforma microcontrolada denominada Arduino. O Arduino foi desenvolvido com a finalidade educativa para dar acesso facilitado a pessoas com pouco ou nenhum conhecimento de eletrônica, tais como artistas e designers que gostariam de usar eletrônica em seus trabalhos, mas não tinha o conhecimento técnico necessário para tal [7].

Arduino é um dispositivo de controle de entradas e saídas que utiliza microcontroladores ATMEGA da fabricante ATMEL, cujo ambiente de desenvolvimento IDE (do inglês, *Integrated Development Environment*) é desenvolvido com o código aberto em JAVA, oferecendo assim compatibilidade aos principais sistemas operacionais (Mac OS, Windows e Linux). Sua linguagem de programação é baseada na linguagem *Processing* que por sua vez é baseada nas linguagens C e C++, com algumas bibliotecas padrões dessas linguagens. Conta também com suas próprias bibliotecas (principalmente no que tange as portas de entrada e saída: digitais, analógicas e *moduladas por largura de pulso*) além de inúmeras bibliotecas disponibilizadas pela comunidade para os mais diversos fins, como o controle de telas de cristal liquido e diversos tipos de sensores [8]. Hoje em dia, o Arduino é muito utilizado por “hobbistas” de eletrônica e estudantes da área, além do seu público alvo

original, ajudando a difundir e facilitando o acesso à eletrônica embarcada, auxiliando inúmeros projetos a saírem do papel, vencendo a barreira da falta de conhecimento de eletrônica devido ao emprego de sua interface amigável e as crescentes contribuições da comunidade. O *hardware* Arduino, bem como sua linguagem e interface de programação são distribuídos de acordo com a licença *Creative Commons* [32].

1.1 Por que o Arduino?

Devido à proposta originária do Arduino de facilitar o acesso à tecnologia de microcontroladores, economizando em tempo de desenvolvimento e de aprendizado,em parte por não utilizar a linguagem de montagem do microcontrolador, o *Assembly*. O aceso às portas de entrada e saída do microcontrolador da ATMEGA é facilitado pela placa e conectores do Arduino.

Nas etapas iniciais de pesquisa para o desenvolvimento deste projeto, foram encontradas duas versões alternativas do Arduino que poderiam a ser utilizadas, a placa Arduino UNO e a placa Arduino MEGA.

1.1.1 Diferenças entre as placas Arduino UNO e Arduino MEGA

Nesta sessão apresentamos um comparativo dos itens principais que diferenciam os modelos de Arduino considerados para o projeto, bem como os microcontroladores usados em cada plataforma, portas de entrada e saída digitais, analógicas [9], bem como a quantidades de portas que podem ser usadas com modulação por largura de pulso *(PWM, do inglês Pulse Width Modulation*). É importante salientar que as características elétricas desses Arduinos são as mesmas, ou seja:

- Tensão de operação :5V
- Tensão de entrada (recomendada): 7-12V
- Tensão de entrada (limites): 6-20V
- Corrente contínua por pino (entrada/saída): 40 mA
- Corrente contínua de saída no pino de 3.3V: 50 mA
- Velocidade do Clock: 16 MHz

A tabela 1 apresenta as informações relevantes entre cada modelo e suas imagens mostrando as diferenças físicas entre o Arduino UNO e o Arduino MEGA:

Tabela 1: Diferenças entre as versões propostas de Arduino para o projeto.

		
Modelo	Arduino UNO	Arduino MEGA
Microcontrolador	ATmega328	ATmega2560
Pinos Analógicos	6	16
Pinos Digitais	14	54
Pinos com PWM	6	14
Portas Seriais	1	3
Memória Flash	32KB (0.5K para o *bootloader*)	256KB (8 KB para o *bootloader*)
SRAM	2 KB	8 KB
EEPROM	1 KB	4 KB
Dimensões	75 x 53 x 15 mm	108 x 53 x 15 mm
Peso	30g	40g

Analisando a tabela 1, é possível perceber que há inúmeras vantagens do Arduino MEGA em relação ao Arduino UNO, principalmente na quantidade de portas disponíveis (tanto analógicas quanto digitais, PWM, e entradas seriais para o uso de GPS e outros tipos de sensores). Porém devido ao alto custo de uma placa Arduino MEGA em relação à placa Arduino UNO (no exterior a diferença de preço é o dobro, e no Brasil chega a custar quatro vezes mais) e as próprias necessidades da embarcação (a princípio apenas duas ou três portas de entrada e saída para monitorar a tensão da bateria embarcada e algum sensor de referência de posição, como um GPS), a placa escolhida foi a Arduino UNO. Caso seja necessário o uso de mais portas seriais, como usar um módulo GPS e um módulo de conexão a internet móvel, serão necessárias mais portas seriais, o que o Arduino UNO não possui.

1.2 Sensores

Quando é necessário mensurar alguma grandeza física para monitoração ou controle de alguma atividade, utilizam-se os dispositivos denominados de sensores [10].
Para esse projeto de uma pequena embarcação teleoperada, que a princípio ficará montada dentro do laboratório para testes de estratégias de controle, os sensores que foram inicialmente considerados são os que possibilitam obter algum tipo de referência de posição e/ou aceleração em relação a um determinado eixo de referências. Tais sensores como acelerômetros, bússola, magnetômetro e o sistema de posicionamento global serviriam para obter uma medida de posição do barco para assim poder-se realizar as estratégias de controle necessárias.

Na pesquisa inicial para este projeto, foi elaborado o orçamento para a compra dos sensores que poderiam ser utilizados para as medidas de posição, porém devido à limitação de tempo, recessos e burocracia não foi possível adquiri-los para integrá-los nesse momento ao trabalho. Toda a plataforma estará apta no futuro a receber os sensores, que com simples modificações nos programas, passam a realizar as leituras dos mesmos.

1.2.1 Bússola

A bússola é um dos mais antigos instrumentos de navegação utilizado na história da humanidade. Sua estrutura atual é basicamente a mesma desde a época de sua invenção pelos chineses. Consiste em um pedaço de material ferromagnético, uma agulha magnetizada colocada apoiada em seu centro de gravidade num plano horizontal, onde essa agulha pode girar livremente, podendo assim se alinhar com o campo magnético da Terra e apontando para o norte magnético, próximo ao norte geográfico [11].

A bússola pesquisada que poderá ser utilizada no projeto é uma bússola eletrônica, montada num circuito integrado, que é capaz de medir variações da direção e intensidade do campo magnético da Terra desde 0.10 gauss até 0.75 gauss.

Os módulos de bússola utilizam a comunicação I²C [12], que é conhecida por usar apenas dois fios para a comunicação, possibilitando assim usar vários dispositivos na modalidade mestre-escravo. Neste caso, mestre é o microcontrolador, que gerencia o *clock* e a solicitação de dados dos dispositivos escravos. O mestre solicita a leitura ao escravo, embora ambos possam ler e escrever entre si, mas apenas o mestre pode iniciar a comunicação [12].

Na Figura 7 podemos ver uma bússola eletrônica, da fabricante Honeywell, com seus quatro terminais, dois responsáveis pela alimentação e os dois referentes ao protocolo I²C.

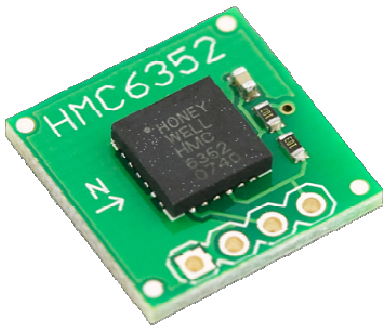

Figura 7: Bússola eletrônica HMC6352 da Honeywell.

1.2.2 Acelerômetro

O acelerômetro é um tipo de sensor que está se popularizando rapidamente nos últimos anos devido à miniaturização dos componentes eletrônicos e consequentemente ao barateamento dos custos de produção. Por isso, tem sido amplamente utilizado dos nos mais diversos dispositivos, que vão desde relógios de pulso (medição de velocidade de corrida), controles de *videogame* (novas maneiras de interação nos jogos), computadores (mecanismos de segurança, como um que desliga os discos rígidos se é detectado uma aceleração brusca, i.e, uma queda, por exemplo) dentre outras aplicações, tanto na indústria quanto nas forças armadas.

O seu funcionamento é baseado na segunda lei de Newton da aceleração da massa e na lei de Hook, lei de força elástica [13]. Enquanto dentro da sua região linear, as molas são regidas pela lei de Hook, onde seu deslocamento é proporcional à força aplicada na mola, ou seja, F=kx, onde k é uma constante da mola que depende de inúmeros fatores como o material que é feito, o numero de espiras, o raio da espira da mola e o raio do material que compõe a mola [14].

O outro princípio físico em que se baseiam os acelerômetros é a segunda lei de Newton, que relaciona a força aplicada na mola com massa do objeto preso a mesma e com a aceleração pela equação F=ma. Quando igualamos as duas equações, temos ma=kx, logo

percebemos que uma aceleração causa um deslocamento da massa de x=(ma)/k, ou ainda, se a massa deslocou x, a equação nos diz que a massa está sob uma aceleração de a=(kx)/m.

Assim, o problema de mensurar a aceleração torna-se um problema de medir o deslocamento de uma massa ao longo de um eixo. Esse seria um acelerômetro de apenas um eixo. Para obter-se aceleração ao longo de outros eixos, deve se entender esse princípio ao longo dos outros eixos que se deseja saber a aceleração. A figura 8 exemplifica um sistema massa mola que pode ser usado para a medição de aceleração em apenas um eixo [14].

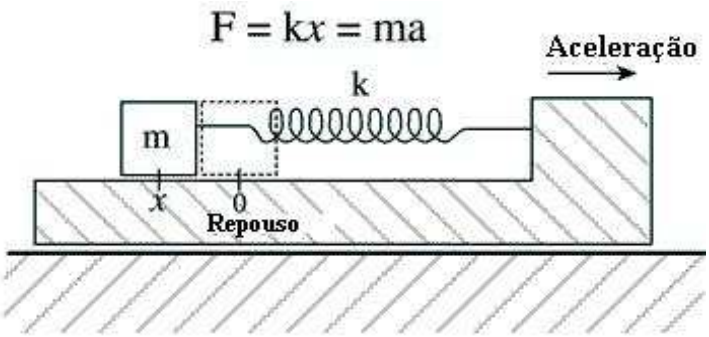

Figura 8: Principio básico de um acelerômetro em um eixo.

O acelerômetro eletrônico analógico tem a característica de ter uma saída referencial para cada eixo, i.e, cada saída indicada por um eixo x, y ou z tem um valor analógico de tensão indicado que varia de acordo com a oscilação detectada pelo mesmo. Essas tensões podem ser medidas usando uma das portas analógicas presentes na nossa plataforma microcontrolada e enviar diretamente pelo sistema de comunicação sem fio para ser medido e interpretador pelo Matlab no computador remoto. Na Figura 9 temos um exemplo de acelerômetro analógico.

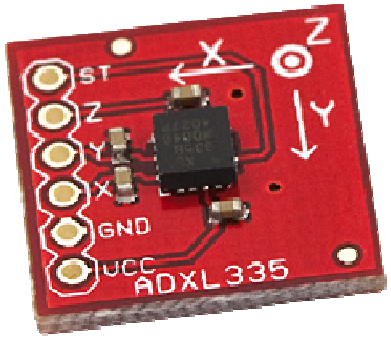

Figura 9: Acelerômetro eletrônico analógico ADXL335.

1.2.3 Giroscópio com acelerômetro

Os giroscópios têm um funcionamento parecido com o dos acelerômetros mas, ao invés de indicar a força que está sendo exercido em uma determinada direção, indicam a velocidade relativa em relação a algum eixo de orientação.

Esse módulo mostrado na Figura 10 possui dois sensores montados em apenas uma placa: são três circuitos integrados, um CI (circuito integrado) responsável pelo posicionamento de *roll* (rolagem, movimento em torno do eixo horizontal,) e *pitch* (inclinação de um ponto em relação ao eixo longitudinal), sendo esse CI o LPR530AL. O segundo circuito integrado, o LY530ALH, é unicamente usado para medir o *yaw* (responsável pelo movimento de guinada, rotação em torno do eixo vertical). A Figura 11 exemplifica a orientação em três dimensões.

O terceiro *chip*, o CI ADXL335, funciona como um acelerômetro puro, como descrito na seção anterior. A diferença é que esse acelerômetro tem uma sensibilidade única, a qual está na faixa de ±3g, enquanto o outro módulo, que possui unicamente o acelerômetro (circuito integrado MMA7361L) tem duas configurações de sensibilidade, ±1,5g (padrão) ou ±6g. Esse módulo todo integrado também não possui regulação de tensão, sendo necessário assim usar uma fonte de alimentação no valor especificado no *data-sheet* para qual pode ser utilizado na saída de 3,3V do Arduino.

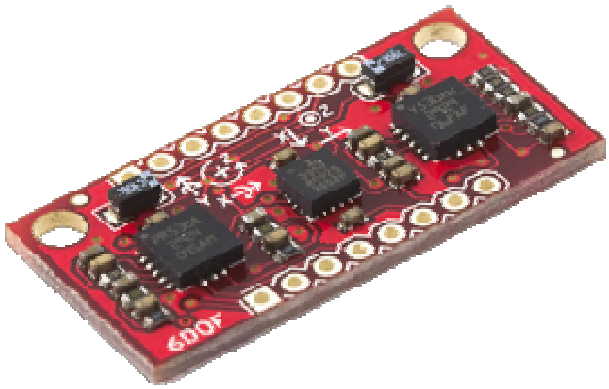

Figura 10: Giroscópio módulo com os 3 chips.

Figura 11: Representação dos eixos de orientação.

Da mesma forma que o módulo puramente de acelerômetro, as saídas desse módulo são analógicas. Seis saídas, três para os eixos do acelerômetro e três para os eixos de orientação do giroscópio.

1.2.4 Sistema de Posicionamento Global

O Sistema de posicionamento global (GPS, do inglês *Global Positioning System*) teve sua origem em 1973 para substituir os antigos meios utilizados na navegação por ondas longas de radio freqüência, que utilizavam essa característica para atravessar condições adversas de clima e da curvatura da terra e substituir o primeiro sistema de navegação por satélite, conhecido como *Transit Satelite* [15], [16].

O *GPS* é de propriedade do governo norte-americano, mas outras nações desenvolveram suas próprias tecnologias de posicionamento global, como a GLONASS da Rússia dentre outras que ainda estão em fase de teste.

O sistema se tornou totalmente operacional a partir do ano de 1995, consistindo de 24 satélites orbitando em torno da Terra. Para um aparelho móvel de GPS funcionar e fornecer corretamente as informações de posição e de hora, ele precisa sincronizar com pelo o menos quatro satélites. O aparelho de *GPS* recebe os sinais dos quatro ou mais satélites, com os quais entra em sincronismo através de um sinal chamado CLK ou *clock*.

Com a informação da posição e das distâncias entre eles, é realizada uma série de cálculos tendo como referência o centro da Terra, realizando a triangulação para estimar a altitude, longitude e latitude do aparelho de GPS. Vale lembrar que a precisão para uso civil do sistema é de metros e para uso militar (restrito aos EUA) é de centímetros. Esse sistema

militar é utilizado para guiar veículos aéreos não tripulados (*UAV* do inglês *Unmanned Aerial Vehicle)* e mísseis de cruzeiro, como o americano *Tomahawk* [15], [16].

A idéia de utilizar um *GPS* no projeto é emular o sistema de posicionamento utilizando em muitos barcos particulares e comerciais para conseguir extrair informações de localização

A Figura 12 mostra um módulo GPS, a figura 12 apresenta o módulo montado em um shield para Arduino, enquanto na figura 13 o módulo *GPS* interligado com ao Arduino.

Figura 12: Módulo GPS.

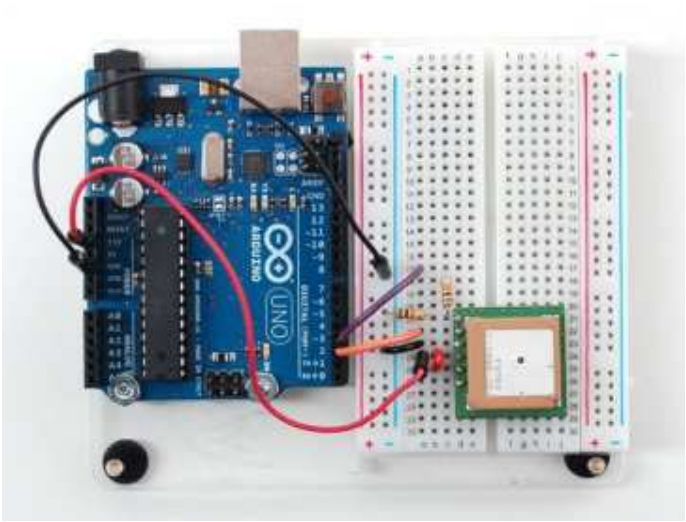

Figura 13: Módulo GPS conectado com o Arduino.

1.3 *Driver* para acionamento e controle dos motores de propulsão do barco

As saídas digitais de circuitos lógicos e de micro controladores não são capazes de acionar diretamente cargas com valores elevados de tensão e corrente. Para possibilitar o acionamento dos motores, far-se-á necessário o uso de circuitos externos para garantir o perfeito funcionamento dos mesmos sem ocasionar nenhum dano ao micro controlador.

Para isso, usaremos um *shield* que funciona como *drive* para os motores, construído em uma Placa de circuito impresso (*PCB*, fo inglês P*rinted Circuit Board*) para ser usada com o Arduino ou qualquer outra plataforma micro controlada. O projeto desse *shield* é baseado no CI L298N. Esse circuito integrado possibilita o controle de motores de passo e o controle de velocidade e direção de rotação para motores *DC*. Optou-se pela compra do drive já pronto ao invés da construção de um drive convencional devido ao seu custo reduzido e também para acelerar o desenvolvimento do projeto. Esse *drive* foi comprado em um momento bem inicial de pesquisa para o projeto, e não sofreu os mesmo atrasos dos sensores. A Figura 15 mostra o drive que será utilizado para controlar os dois motores do barco.

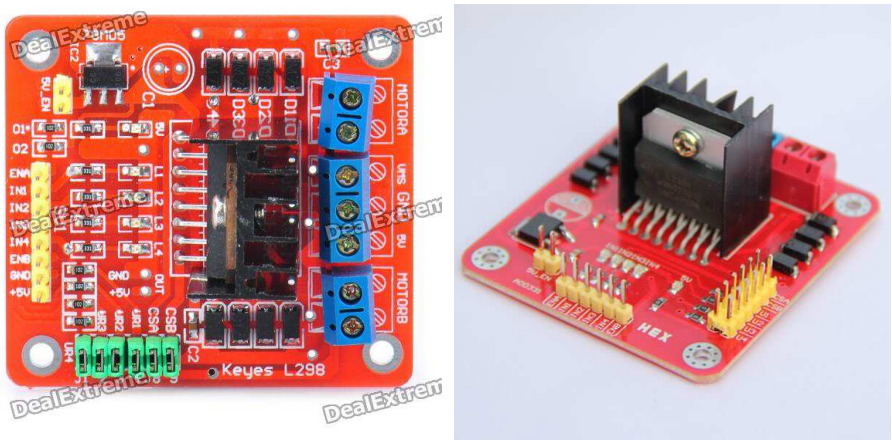

Figura 14: *Driver* para motores DC baseado no circuito integrado L298N.

A figura 15 mostra o diagrama elétrico do CI em questão:

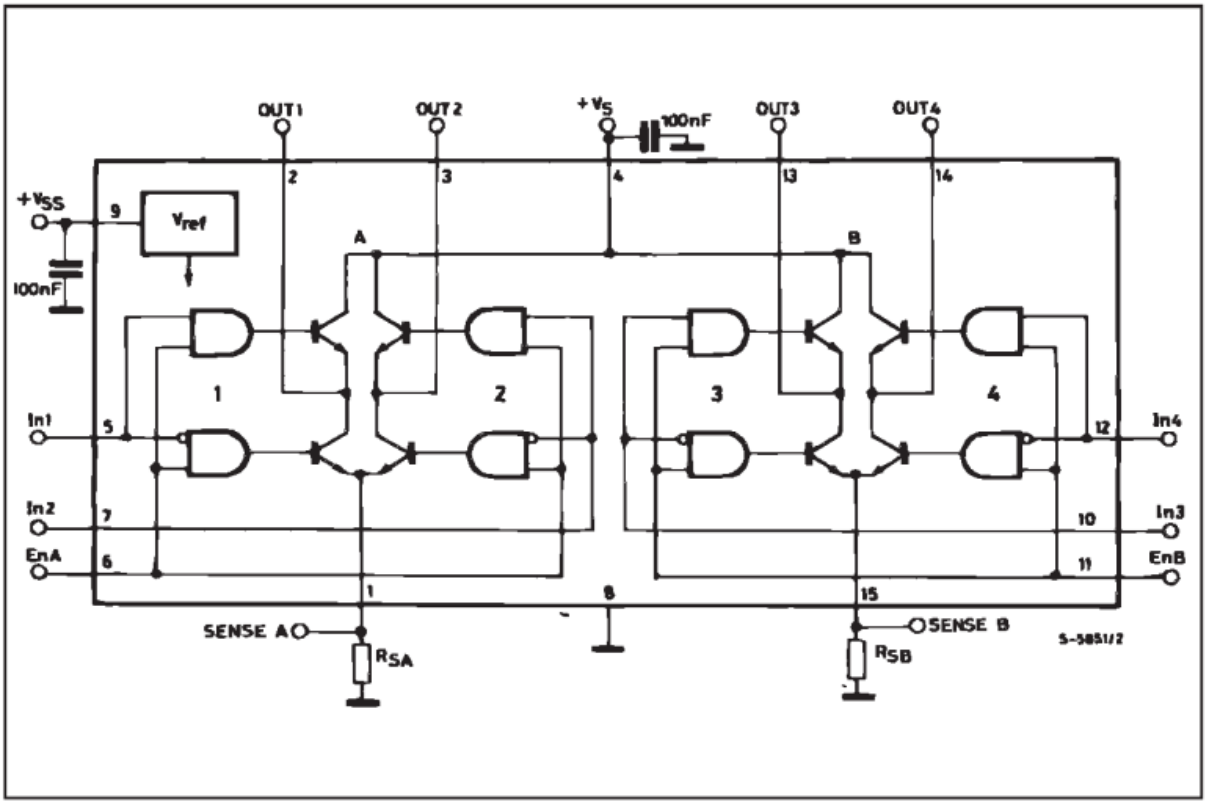

Figura 15: Esquema elétrico do CI L298N.

O capitulo três contará com mais detalhes acerca desse *driver* e os motivos para a escolha.

2 COMUNICAÇÂO SEM FIO

2.1 Introdução

Atualmente, temos diversos tipos de redes utilizadas para troca de arquivos e conexão entre dispositivos e transmissão voz, tal como o *Wi-fi, Bluetooth* e o *3G*. Todos esses métodos são voltados para o usuário final, como pequenas e grandes empresas. Não havia até então uma solução voltada para o controle de atuadores e leitura dos dados de sensores como temos em ambiente automatizado. O padrão *Zigbee* veio para suprir essa necessidade [17], oferecendo a capacidade de controle, através de dispositivos que não possuem uma grande largura de banda para transmissão de volume de dados, mas atuam de forma rápida e consomem pouca energia, possibilitando que fiquem em locais de difícil acesso sem implicar a perda de confiabilidade de uma falta de energia ou esgotamento da mesma.

Foram estudados três tipos de conexão sem fios citados anteriormente exceto 3G, mas apenas dois puderam ser testados, o *Bluetooth e o Zigbee*, pois não tínhamos disponíveis os equipamentos necessários para os testes com o *Wi-Fi.*

O primeiro dispositivo testado foi o *Bluetooth*, através de um módulo *Bluetooth* para microcontrolador e um *Bluetooth* dongle ligado no PC. As figuras 16 e 17 mostram esses dispositivos.

Figura 16: Módulo Bluetooth para uso em microcontroladores.

Figura 17 Bluetooth Dongle para o computador.

O Bluetooth se mostrou difícil de trabalhar. Sua conexão e funcionamento eram intermitentes. A conexão era estabelecida, os dados eram enviados e recebidos, mas uma vez desconectado do programa usado para o controle não era possível restabelecer a conexão. Com esse funcionamento intermitente não seria um dispositivo confiável para o uso.

O Zigbee por outro lado, após serem configuradas as duas unidades, a transmissora no computador e a receptora ligada no microcontrolador, funcionava sem falhas em todas as tentativas de conexão realizadas.

O artigo [18] apresenta comparativo entre as principais características dos principais tipos de conexão sem fio, dentre eles o *Bluetooth*, *Zigbee* e o *Wi-Fi*. Os autores expõem as principais características desses protocolos, tais como banda e consumo de energia. Através dele, conclui-se que o *Zigbee* apresenta o menor consumo e, que apesar da sua largura banda ser a menor dentre os outros, inicialmente para controle de posição e receber informações de tensão oriundas do barco, essa banda se mostrou muito eficiente. Caso no futuro seja necessário implementar uma câmera no barco, o ideal seria a mudança para um protocolo que oferece uma largura de banda maior, como o *Wi-Fi*.

Assim, o Zigbee se mostrou ideal para a aplicação atual do barco, além de ser o que demonstrou a melhor funcionalidade nos teste. O artigo também deixa claro que os autores não concluem qual protocolo é melhor em relação ao outro, pois existem outros fatores práticos que influenciam a aplicação de redes, como confiabilidade e o preço.

A tabela 2 e a figura 18 retiradas do artigo [18] mostram um comparativo em formato de tabela e gráfico comparando o consumo com o consumo de energia de cada protocolo. A

sigla TX refere-se ao dispositivo estar enviando dados e o RX refere-se ao dispositivo estar recebendo dados.

Tabela 2: Tabela comparativa de alguns parâmetros dos protocolos estudados.

Standard	Bluetooth	UWB	ZigBee	Wi-Fi
Chipset	BlueCore2	XS110	CC2430	CX53111
VDD (volt)	1.8	3.3	3.0	3.3
TX (mA)	57	~227.3	24.7	219
RX (mA)	47	~227.3	27	215
Bit rate (Mb/s)	0.72	114	0.25	54

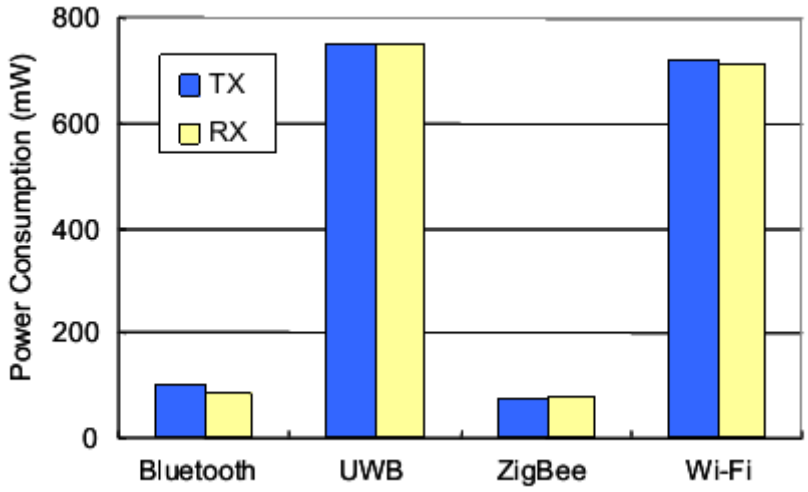

Figura 18: Comparação de potência consumida entre os protocolos.

2.2 **O protocolo Zigbee**

Existem várias versões distintas de módulos *Xbee*, e as diferenças básicas entre os módulos são as seguintes [19]:

Xbee Série 1 : São mais fáceis de utilizar, não precisam de muitas configurações para começar a operar (pode-se otimizá-lo através de um ajuste fino das mesmas). Por serem fáceis de usar, são recomendado para todos que querem começar a trabalhar com o *Xbee*. São recomendados para conexões ponto-a-ponto por não exigirem grandes configurações (o que não ocorre com a da Série 2). Por serem plataformas com características diferentes, não são

capazes de operarem entre si, i.e, o transmissor montado no computador terá que ser série 1 e o receptor montado no Arduino deverá também ser série 1.

Xbee Znet Série 2.5 (Conhecida anteriormente como Série 2 - Fora de linha): Módulos da série 2.5 precisam ser configurados antes de serem utilizados.. Estão atualmente fora de linha, i.e, não são mais vendidos. A desenvolvedora dos módulos oferece um kit de conversão gratuito (um firmware para ser baixado e instalado no módulo). Esse módulo/versão introduziu a característica de redes com três topologias: *mesh* (malha), *star* (estrela) e *cluster tree* (árvore). Podemos ver na figura 19 como são essas topologias.

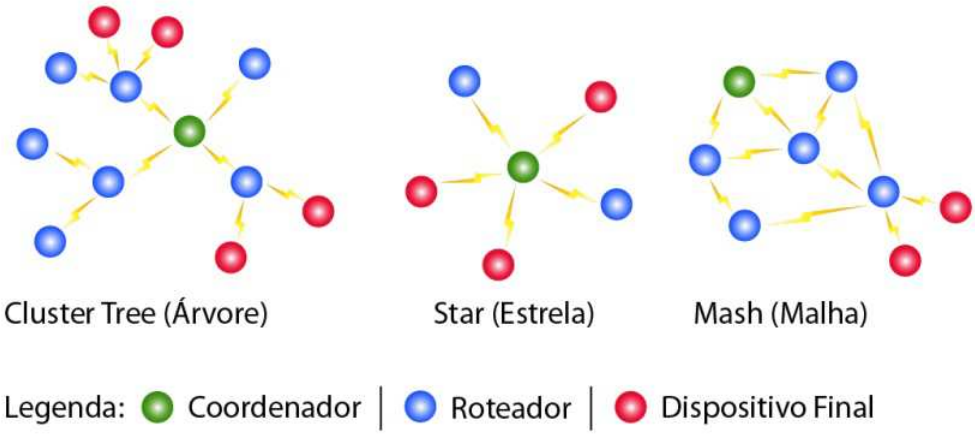

Figura 19: Topologias de rede Zigbee.

Xbee Série ZB (O módulo série 2 atualmente)**:** É basicamente a série 2.5 com um novo *firmware*, o que significa que esses módulos podem funcionar de forma transparente ou usando os comandos API. Esses módulos também podem operar na topologia de Malha, formando redes entre dispositivos finais e outras redes zigbee. Existem na versão Xbee-ZB, Xbee-PRO(S2) e Xbee-PRO(S2B).

Xbee Série 2B: Esse é módulo mais atual, trouxe melhorias, principalmente no consumo e uso de energia. A Série 2B funciona com o firmware ZB, mas devido às mudanças nas características de hardware, eles não são compatíveis e não podem “falar” entre si.

Antes de verificarmos as características de cada ponto de rede, precisamos conhecer os dois tipos de dispositivos numa rede Zigbee, o FFD e o RFD:

FFD - *Full F**unction D**evice*** (Dispositivos de Funções Completas): São dispositivos mais complexos, que consomem mais energia, pois são responsáveis pela implementação da pilha de protocolo e gerenciamento da rede. Dispositivos finais também podem ser configurados como FFD. Os dispositivos FFD podem se comunicar com quaisquer outros dispositivos presentes na rede.

RFD – ***Reduced F**unction D**evice*** (Dispositivos de Funções reduzidas) : São dispositivos que consomem menos recursos. Eles assumem obrigatoriamente o papel de dispositivo final, ou seja, os atuadores ou sensores de nossa automação.

Visto isso, podemos ver as três classes de dispositivos que controlam a topologia em uma rede Zigbee:

Zigbee Coordenador – São necessariamente dispositivos configurados como FFDs. O coordenador é responsável pela inicialização, distribuição dos endereços e reconhecimento de todos os Nós e também serve como ponte entre outras redes que utilizem outras topologias.

Zigbee Roteador – Também são necessariamente dispositivos configurados como FFDs. Tem a função roteador intermediário entres os nós , sem precisar de um outro coordenador na rede. Um exemplo de uso de um roteador seria o caso de um prédio usando automação com Zigbee, onde os roteadores serviriam para interconectar diferentes andares.

Zigbee Dispositivo Final – Podem ser configurados como FFD ou RFD. É a ponta, o final da rede, onde os sensores ou atuadores estão localizados

3.2.1 Diferenças entre os módulos *Xbee Série* vs. *Xbee Série Pro*:

As Tabelas 3, 4 e 5 apresentam as diferenças entres os módulos de suas correspondentes séries em relação a versão normal e a versão Pro do módulo Zigbee.

Tabela 3: Diferença entres Xbee série 1 e Xbee Série 1 PRO

Xbee Série 1	Xbee Série 1 PRO
Desempenho: - Potência de saída: *1 mW (0 dBm);* - Alcance em ambientes internos/zonas urbanas: *30m;* - Alcance de RF em linha visível para ambientes externos: *100m;* - Sensibilidade do receptor: *-92 dBm;* - Frequência de operação: *ISM 2.4 GHz;* - Taxa de dados de RF: *250.000 bps;* - Taxa de dados da Interface (Data Rate): *Até 250.000 bps;* **Alimentação:** - Tensão de alimentação: *2.8 à 3.4v;* - Corrente de transmissão (típico): *45 mA @ 3.3 V;* - Corrente de Recepção (típico): *50 mA @ 3.3 V;* - Corrente de Power-down/ Sleep: *<10 µA;* **Características físicas:** - Dimensões: 2.438cm x ***2.761cm***; - Peso: 3g; - Temperatura de operação: -40 to 85º C (industrial);	**Desempenho :** - Potência de saída: *60 mW (18 dBm),* - Alcance em ambientes internos/zonas urbanas: *100m;* - Alcance de RF em linha visível para ambientes externos: *1,6Km;* - Sensibilidade do receptor: *-100 dBm (1% PER);* - Frequência de operação: *ISM 2.4 GHz;* - Taxa de dados de RF: *250.000 bps;* - Taxa de dados da Interface (Data Rate): *Até 250.000 bps*; **Alimentação:** - Tensão de alimentação: *2.8 à 3.4v;* - Corrente de transmissão (típico): *20 mA @ 3.3 V;* - Corrente de Recepção (típico): *55 mA @ 3.3 V;* - Corrente de Power-down /Sleep: *<10 µA;* **Características físicas:** - Dimensões: (2.438cm x **3.294cm**); - Peso: *3g;* - Temperatura de operação: -40 to 85º C (industrial);

Tabela 4: Diferenças entre os módulos Xbee Série 2.5 e Xbee Série 2.5 Pro

Xbee Série 2	Xbee Série 2 PRO
Desempenho: - Potência de saída: *2mW(+3dBm com modo Boost ligado), 1,25mW(modo Boost desligado);* - Alcance em ambientes internos/zonas urbanas: *40m;* - Alcance de RF em linha visível para ambientes externos*: 120m;* - Sensibilidade do receptor: -96dBm(boost enable), -95dBm(boost disable); - Frequência de operação: *ISM 2.4 GHz;* - Taxa de dados de RF: *250.000 bps;* - Taxa de dados da Interface (Data Rate): *230400 bps (Taxas de transmissão serial não padrão são suportadas)* **Alimentação:** - Tensão de alimentação: *2.1 à 3.6V;* - Corrente de transmissão (típico): 40mA @ 3.3 V(modo boost ligada) e 30mA @ 3.3V(modo boost desligado); - Corrente de Recepção (típico): *40mA @ 3.3 V(modo boost ligada) e 30mA @ 3.8V(modo boost desligado);* - Corrente em idle(receptor desligado): *15mA;* - Corrente de Power-down/ Sleep: *<1 µA @ 25º C;* **Características físicas:** - Dimensões: *2.438cm x* ***2.761cm***; - Peso: *3g*;	**Desempenho:** - Potência de saída: *63mW (+18dBm);* - Alcance em ambientes internos/zonas urbanas: *100m;* - Alcance de RF em linha visível para ambientes externos: *1,6Km;* - Sensibilidade do receptor: *-102dBm;* - Frequência de operação: *ISM 2.4 GHz;* - Taxa de dados de RF:; *250.000 bps;* - Taxa de dados da Interface (Data Rate): *230400 bps (Taxas de transmissão serial não padrão são suportadas)****;*** **Alimentação:** - Tensão de alimentação: *3 à 3.4V;* - Corrente de transmissão (típico): *295mW @ 3.3 V;* - Corrente de Recepção (típico): *45mA @ 3.3 V;* - Corrente em idle(receptor desligado): *15mA;* - Corrente de Power-down /Sleep: *<1 µA @ 25º C;* **Características físicas:** - Dimensões: *2.438cm x* ***3.294cm****;* - Peso: *3g;*

Tabela 5: Diferenças entre os módulos Xbee Série ZB e. Xbee Série ZB Pro:

Xbee ZB	Xbee-PRO (S2B)
Desempenho : - Potência de saída: *2mW(+3dBm com modo Boost ligado), 1,25mW(modo Boost desligado);* - Alcance em ambientes internos/zonas urbanas: *40m;* - Alcance de RF em linha visível para ambientes externos: *120m;* - Sensibilidade do receptor: -96dBm(boost enable), -95dBm(boost disable); - Freqüência de operação:*ISM 2.4 GHz ;* - Taxa de dados de RF: *250.000 bps;* - Taxa de dados da Interface (Data Rate): *Até 1Mbps* **Alimentação:** - Tensão de alimentação: *2.1 à 3.6V;* - Corrente de transmissão (típico): 40mA @ 3.3 V(modo boost ligada) e 30mA @ 3.3V(modo boost desligado); - Corrente de Recepção (típico): 40mA @ 3.3 V(modo boost ligada) e 30mA @ 3.8V(modo boost desligado); - Corrente em idle(receptor desligado):15mA; - Corrente de Power-down/ Sleep: <1 µA @ 25º C; **Características físicas** - Dimensões: *2.438cm x **2.761cm**;* - Peso: *3g;*	**Desempenho:** - Potência de saída: *63mW (+18dBm);* - Alcance em ambientes internos/zonas urbanas: *90m;* - Alcance de RF em linha visível para ambientes externos: *3.2km;* - Sensibilidade do receptor: *-102dBm*; - Freqüência de operação: *ISM 2.4 GHz;* - Taxa de dados de RF: *250.000 bps;* - Taxa de dados da Interface (Data Rate): *Até 1Mbps (Taxas de transmissão serial não padrão são suportadas);* **Alimentação:** - Tensão de alimentação: *2.7 à 3.6V*; - Corrente de transmissão (típico): *205mW @ 3.3 V;* - Corrente de Recepção (típico): *47mA @ 3.3 V;* - Corrente em idle(receptor desligado): *15mA;* - Corrente de Power-down /Sleep: *3.5 µA típico @ 25º C;* **Características físicas:** - Dimensões: 2.438cm x ***3.294cm***; - Peso: 3g;

2.3 **Considerações acerca dos tipos de antenas disponíveis.**

Existem três tipos de antenas disponíveis para módulos *Xbee*: a antena em chip (Figura 20), antena chicote ou *whip (Figura 21)* e módulo com um conector UFL *pigtail* para a colocação de uma antena externa (Figura 22).

Figura 20: Antena Chip.

Figura 21: Antena whip ou chicote.

Figura 22: Antena com conector Pigtail e UFL.

Ainda existe a antena PCB (do inglês, Printed circuit board), que é o módulo *Xbee* com a antena montada como uma trilha do *PCB*. Podemos ver na Figura 23, como é fisicamente esse modelo de *Xbee* com esse tipo de antena:

Figura 23: Xbee com antena PCB.

As principais diferenças entre os tipos de antena vistos nas figuras 20, 21 e 22 são descritas a seguir:

A antena chicote leva vantagem no alcance em comparação à antena chip, mas apenas para locais abertos. Em locais fechado/urbanos o desempenho é idêntico.

A antena PCB tem desempenho similar da antena chicote (cerca de 5% a menos), ainda assim se saindo melhor que a antena *chip*.

Essas informações foram extraídas do documento oficial da Digi [21] (fabricante destes módulos Xbee) que trata sobre o desempenho das antenas. Segundo esse mesmo documento, é aconselhável o teste de alcance dos dispositivos "*in loco'*, ou seja, já com o dispositivo montado e operando.

2.4 **Xbee *Dongle***

Para realizarmos a comunicação entre o computador e o módulo *Xbee* a fim de configurá-lo, criar, gerenciar, controlar e adquirir dados uma rede Zigbee, é preciso utilizar um adaptador que realize a conversão USB/Serial para podermos ligar o módulo ao computador. Existem duas formas de realizar a interface desses módulos Xbee: utilizando uma placa Arduino que se possa retirar o microcontrolador (ou aterrar o pino de reset, levando assim o microocontrolador a não iniciar) de forma que o módulo Xbee usará o *chip FTDI (*responsável pela conversão de RS-232 serial para TTL ou *USB*) presente na placa Arduino (responsável pela comunicação serial entre o microcontrolador e o computador) para a interface, ou adquirir um conector específico para conexão de Xbee no computador, que pode ser um *Xbee dongle* ou um *Xbee Explorer*.

O *Xbee dongle* é bem parecido fisicamente com um *pendrive,* onde o módulo *Xbee* fica encaixado diretamente na *USB*. Já o *Xbee Explorer* é um adaptador que possui um cabo/extensão *USB*. A figura 24 mostra um Arduino e um *Xbee* montado com um *shield*, como o que será usado neste projeto.

Como o *shield* fica montado por cima do Arduino, não dá para visualizar a ausência do microcontrolador. Na Figura 25 temos um *Xbee doongle* e na Figura 26 um *xbee explorer*.

Figura 24: Arduino com Xbee shield.

Figura 25: Xbee dongle.

Figura 26: Xbee Explorer.

2.5 **Escolha para o projeto**.

Devido a problemas citados anteriormente, apenas tivemos disponíveis para os testes (e os que estão sendo usados no barco) os equipamentos cedidos pelo professor Lisandro Lovisolo, que são dois *Zigbees* Pro série 1, ambos com antena *whip*.

As vantagens do uso desse protocolo em detrimento dos outros é facilidade de uso e o baixo consumo de corrente.

3 EMBARCAÇÃO, COMUNICAÇÃO E O CIRCUITO ACIONADOR DOS MOTORES

31 Embarcação

3.1.1 Desenvolvimento

A embarcação utilizada foi criada e desenvolvida em 2008, pelo aluno Gustavo de Sá Amaral em seu projeto final de graduação [20]. No apêndice A de seu trabalho de conclusão de curso são explicadas de que forma ele chegou à composição do barco, em que fora utilizada uma placa de isopor apenas a principio, mas isso dificultava o controle do barco, pois a placa de isopor possui pouca inércia e, ao serem acionados, os propulsores geravam uma oscilação muito brusca na posição desejada pelo sistema de controle. A solução encontrada foi adicionar uma chapa de madeira para aumentar essa inércia e diminuir esses saltos bruscos.

Outro fator decisivo na construção da estrutura preliminar do barco foi a placa de isopor, formando uma cobertura sobre os propulsores. Essa placa foi necessária, pois os motores dificultaram o sistema empregado a principio para aquisição de posição que foi utilizado. Os propulsores faziam sombra sob a placa e impediam a correta detecção e estimação da geometria da peça que seria utilizada pelo algoritmo de controle para posicionamento da embarcação.

Desde que fosse utilizado o sistema de aquisição de posição proposto na introdução deste trabalho, não seria necessário utilizar essa cobertura de isopor, já que ao invés do sistema de posicionamento por diferença de luminância, esse sistema de câmeras utiliza esferas cobertas com uma superfície reflexiva e câmeras com infravermelho para medir a posição das mesmas em relação a um eixo de referência, bastando assim que as esferas estejam estrategicamente posicionadas nas extremidades do barco a fim de formar o objeto, um “corpo rígido” que será configurado no programa como uma “entidade” com seu nome e dimensões. A precisão da medição de posição desse sistema é de milímetros e além de medir distância e posição, ela é capaz de informar o posicionamento angular nos três eixos em relação à origem do seu sistema de coordenadas.

Com essas configurações prontas e o sistema calibrado, podem-se obter todos os parâmetros necessários para a orientação do barco e assim usar o sistema de posicionamento dinâmico da mesma forma que fora utilizado em [21].

Nas figuras 27 e 28 são apresentadas duas fotos retiradas de [21], mostrando a embarcação completa, com sua cobertura e a segunda foto mostra os propulsores montados na chapa de madeira.

Figura 27: Embarcação completa com sua cobertura de isopor.

Figura 28: Embarcação sem a cobertura de isopor, mostrando a disposição dos propulsores.

3.2 **Propulsores**

Os propulsores utilizados são os mesmos utilizados em [21], três motores elétricos de corrente contínua, modelo PM100-SG, cujas características são apresentadas na tabela 6.

Tabela 6: Características do motor PM100-SG

Tensão (V)		Em vazio		Máximo rendimento					Bloqueio	
		Rotação	Corrente	Rotação	Corrente	Torque	Potência	Rendimento	Torque	Corrente
Operação	Nominal	RPM	A	RPM	A	g.cm	W	%	g.cm	A
12	12	4870	0,03	3901	0,12	23,1	0,923	63,71	116	0,486

Nesse projeto inicial de controle sem fio da embarcação, apenas serão utilizados dois motores por limitação do circuito acionador. Na seção 4.3 será detalhado o circuito de acionamento do motor e os motivos que levaram à escolha dessa solução em detrimento da placa desenvolvida em [21]. Como descrito em [21], não houve preocupação com o consumo de energia, já que a embarcação estaria ligada a partir de um cordão umbilical, que leva aos propulsores no barco os comandos oriundos da placa desenvolvida para controlar os motores através da porta paralela do computador. Este trabalho contará com testes e medições de valores de tensão e corrente de acordo com a velocidade imposta aos motores (através do controle por PWM) para poder dimensionar uma bateria que atenda a autonomia desejada.

A figura 29 mostra a placa que foi desenvolvida para o acionamento dos motores através da porta paralela.

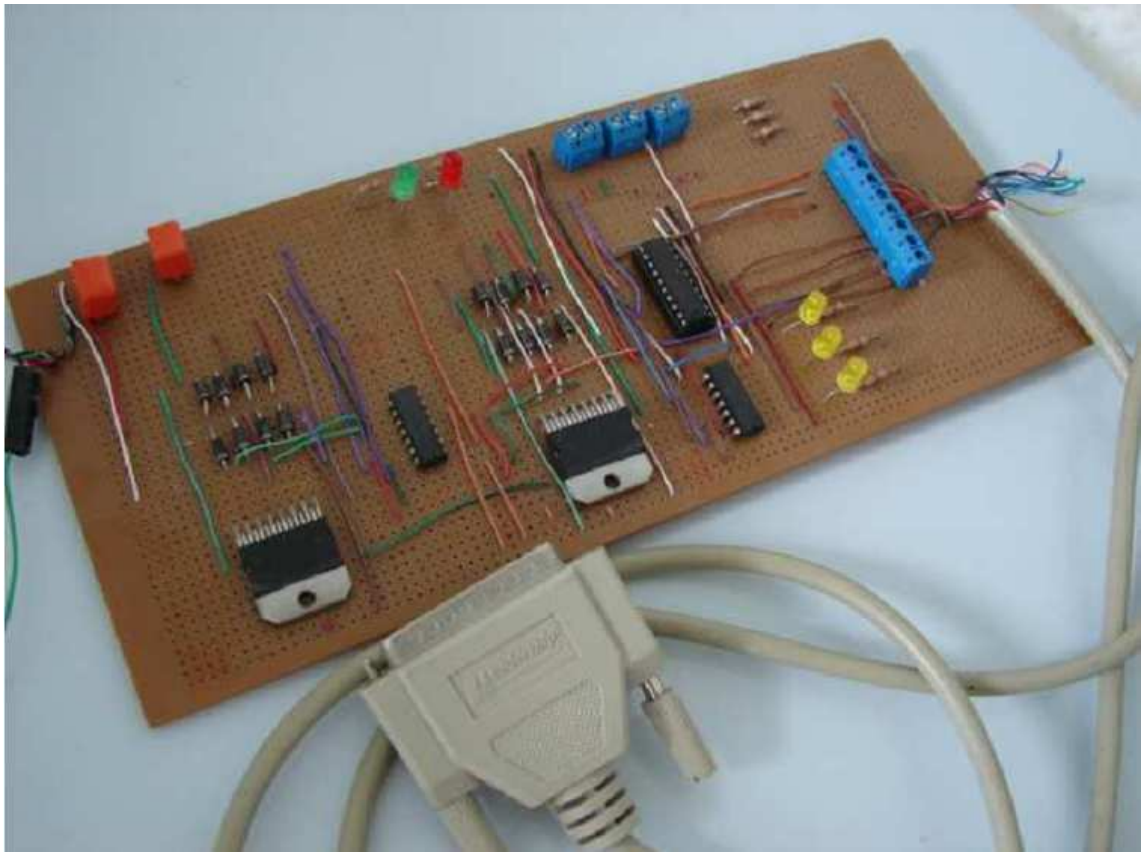

Figura 29: Placa para acionamento dos motores utilizada em [20].

3.3 **Comunicação com o Barco via Matlab**

O Zigbee, dispositivo que será utilizado para a comunicação sem fio entre o barco e o computador, utiliza a comunicação serial para se conectar ao computador e receber/enviar os dados que serão transmitidos ao outro dispositivo Zigbee configurado na mesma rede. Essa comunicação serial se dá entre o PC e módulo Zigbee transmissor (independente da maneira que o Zigbee é ligado ao PC, seja com o Arduino sem o microcontrolador para usar sua interface serial ou usando o Xbee dongle/Explorer) e o entre o módulo Zigbee receptor e o dispositivo que será controlado.

O *software* (programa de computador) que será utilizado para o controle da embarcação é o Matlab®, um *software* de computação numérica para engenharia e cálculos científicos. O nome como é conhecido vem do acrônimo de *Matrix Laboratory* (em português, Laboratório de Matrizes) e foi originalmente desenvolvido para facilitar operações com matrizes em ambientes computacionais [22].

O motivo para a escolha desse *software* foi seu incrível potencial computacional, funções pré-definidas e compatibilidade com interface serial do computador para troca de informações e aquisições de dados com o mundo exterior. Essa escolha visa acelerar o processo de desenvolvimento do projeto, já que muitas ferramentas necessárias para controle já estão disponíveis por padrão, facilitando a tarefa do programador. Essa escolha de programa não inviabiliza o uso de outra linguagem de programação para o controle, mas alongaria demais o tempo para o desenvolvimento do projeto, visto que necessitaríamos desenvolver toda uma interface de troca e aquisição de dados.

3.3.1 **Comunicação Serial do Matlab**

A comunicação serial é a interação de baixo nível mais utilizada para troca de dados entre aparelhos eletrônicos sem a necessidade de conhecimento especifico do funcionamento da porta serial. Na grande maioria dos casos, a comunicação serial se dá normalmente entre um computador e algum outro dispositivo, tais como *modems* para conexão à internet, impressoras (no ambiente doméstico não mais, mas em muitas plataformas comerciais ainda se usam impressoras fiscais que utilizam a porta serial para comunicação), outro computador ou algum instrumento científico, como um osciloscópio, um gerador de sinais ou algum equipamento de instrumentação [23].

3.3.2 **O padrão serial**

A interface serial para realizar a conexão entre dois dispositivos é especificada pela norma TIA/EIA-232C, padronizada pela *EIA (Eletronics Industries Association)*, uma organização privada norte-americana, credenciada pela ANSI *(American National Standards Institute)* para ajudar a desenvolver padrões em equipamentos eletrônicos e para telecomunicações e internet [20], [23], [25].

A norma define os padrões da comunicação serial, como a máxima taxa de transferência de dados, o tamanho máximo do cabo para a conexão dos equipamentos, a nomenclatura, características elétricas e mecânicas dos conectores e a atribuição dos pinos de conexão [24].

A transmissão de dados se dá de duas formas distintas: comunicação síncrona e comunicação assíncrona.

Na comunicação síncrona, todos os bits transmitidos são sincronizados com um único sinal de *clock*. Assim, os dois equipamentos sincronizam entre si e enviam continuamente um dado para permaneceram sincronizados.

Na comunicação assíncrona, cada dispositivo tem seu próprio *clock* interno, resultando assim em bits enviados em momentos arbitrários. A maneira usada para a sincronização dos dados não é o tempo (oriundo do sinal de relógio), mas sim usando o bit inicial da *word* (palavra) enquanto um ou mais bits indicam o fim da *word*. Esse método causa um pequeno inconveniente de ser um pouco mais lento (devido ao envio adicional de bits para o controle) [25].

3.3.3 O pacote de suporte ao Arduino para o Matlab.

Em seu site [27], a Mathworks, responsável pelo desenvolvimento e suporte do Matlab, disponibilizou a partir do ano de 2009 um pacote de suporte para a plataforma microcontrolada Arduino, que oferece ao mesmo tempo suporte ao Matlab e suporte ao Simulink, uma ferramenta para modelagem, simulação e análise de sistemas dinâmicos, onde sua interface utiliza diagramas de blocos para representar os processos ou plantas. Esse pacote inicialmente oferecia suporte apenas à versão com menos portas de entrada e saída e portas seriais do Arduino, o Arduino duemilenove e não à versão MEGA. No período inicial de pesquisa para esse trabalho e ao longo do desenvolvimento, era suportada apenas versão mais básica. No final de 2012, foi lançada uma nova versão que já contava com configuração

padrão da velocidade da porta serial de 115.200bps e já vinha com a programação necessária para suporte as dezenas de portas que o Arduino MEGA disponibiliza. O código já tinha sido analisado e estudado para serem propostas essas modificações para o uso futuro da versão MEGA na embarcação, mas felizmente os desenvolvedores originais disponibilizaram a ferramenta, economizando tempo de pesquisa, teste e resolução de problemas futuros nesse projeto.

Desde o inicio do projeto, as velocidades já tinham sido modificadas tanto nesse pacote de suporte, quanto nos Zigbee e nas portas de comunicação do sistema operacional para não termos problemas com qualquer uma das duas pontas em relação à perda de pacotes e a velocidade não ser suficiente para receber e enviar os dados lidos e escritos através do Matlab.

O pacote de suporte para Arduino usa os comandos disponíveis por padrão nas ferramentas de interface serial do Matlab para criar os objetos necessários e criar atalhos aos comandos para possibilitar o acesso simples às portas, assim como ser possível com apenas uma linha de comando colocar um valor analógico em uma determinada porta ou ler o estado da mesma (no caso de algum sensor ou medidor de temperatura). Em suma, esse pacote visa facilitar o uso do Arduino no Matlab, fazendo com que as linhas de comando necessárias sejam bem próximas a linguagem de programação padrão do Arduino.

O capitulo quatro abordará a instalação e configuração de todos os dispositivos desde o inicio até a configuração para uso desse pacote de suporte ao Arduino.

3.4 **Circuito acionador do motor.**

3.4.1 *Hardware*

Tanto o circuito controlador original usado em [21] quanto o circuito utilizado neste projeto de acionamento sem fio do barco usam placas desenvolvidas com o chip L298, um circuito integrado acionador de motores em ponte completa (*Dual full-bridge driver*). É um circuito integrado composto por duas pontes do tipo “H” que possibilitam controlar em que sentido os motores são acionados [28]. O controle original utilizava um bit (um pino) da porta paralela para selecionar o sentido e outro bit para ligar ou desligar o motor. Esse controle de direção com apenas um bit foi realizado baseado na lógica disponível na sua folha de dados,

como é mostrado na tabela 7 e através do arranjo mostrado na figura 30, onde foi utilizada uma porta inversora para ser ter duas saídas contrárias para o controle do *drive*:

Tabela 7: Configuração utilizada inicialmente segundo sua folha de dados [29].

EnA	In1	In2	Polarização (OUT1-OUT2)
1	1	0	Direta
	0	1	Inversa
0	X	X	Transistores cortados

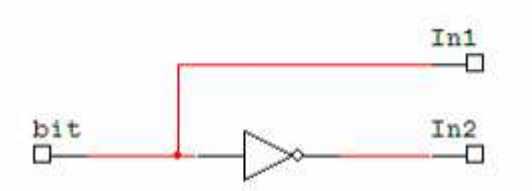

Figura 30: Arranjo para controle de direção do motor.

Pode ser visto ver na figura 29, que o circuito acionador construído originalmente tem dimensões elevadas para ficar a bordo. A solução encontrada foi a compra de um circuito acionador construído com o mesmo chip montado em uma placa de circuito impresso, com componentes em montagem de superfície na placa, reduzindo assim o tamanho para cerca de 1/10 do tamanho da placa original. O circuito acionador adquirido é constituído de apenas um circuito integrado L298, podendo assim controlar até dois motores. Essa escolha foi devido ao baixo custo desse circuito em relação ao custo de componentes e de tempo para desenvolver uma placa menor para o controle. Por se tratar de uma placa construída por processos industriais, ela tem tamanho e peso reduzidos, características críticas nessa aplicação. A figura 31 mostra a placa que foi adquirida:

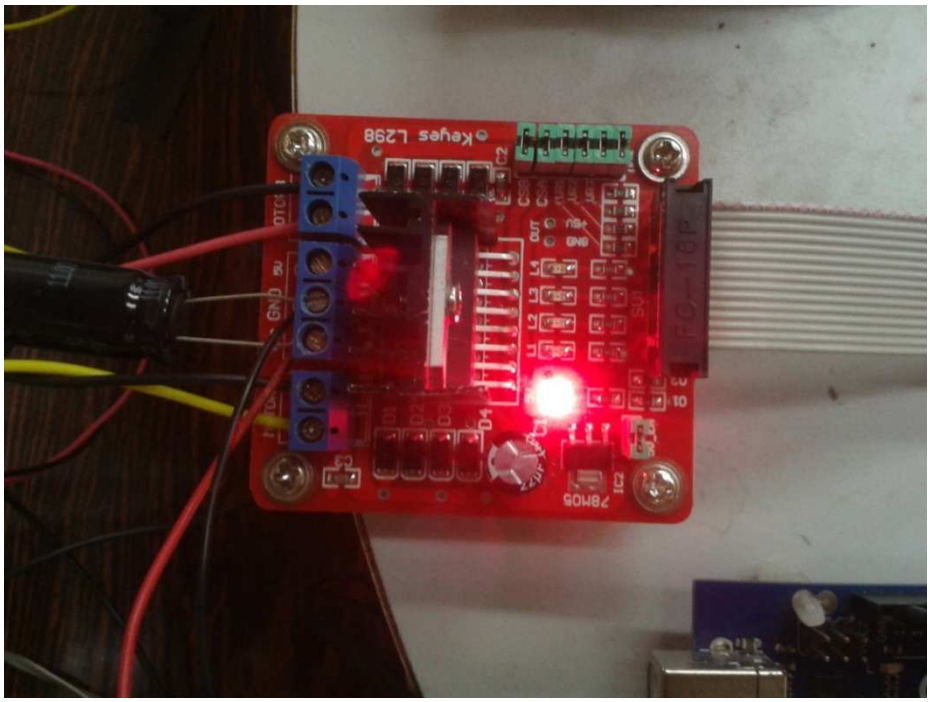

Figura 31: Placa comprada para acionamento dos motores.

A placa originalmente não contava com nenhum dos dois capacitores que são vistos na imagem. Um está posicionado externamente junto com a alimentação (o maior) e o segundo foi soldado diretamente na placa perto do regulador de tensão 7805. Eles foram colocados para diminuir a quantidade de componentes harmônicas presentes nas formas de onda de saída. O maior é de 1000uF e o menor de 230uF.

Ao utilizarmos um microcontrolador, tem-se a possibilidade do controle de velocidade dos propulsores através da saída PWM, o que acrescenta mais possibilidades à embarcação.

A forma de acionamento PWM foi utilizando os pinos de direção como os pinos que recebem o sinal de controle PWM. Isso é possível, pois os valores são configurados no programa de controle (Matlab), bastando programar a direção desejada com o valor que se deseja de velocidade em uma das portas (Int1, por exemplo) e na outra basta colocar o valor zerado. Lembrando que pela tabela 7, se colocarmos os dois valores iguais nos pinos de direção, o propulsor ficará no estado bloqueado.

A figura 32 retirada do osciloscópio mostra a forma de onda aplicada no motor usando a ligação citada acima. A forma de onda em verde é a tensão PWM aplicada nos motores e a forma de onda amarela é a corrente medida utilizando uma ponteira própria para medição desta grandeza. O capitulo cinco, Testes e medições, apresenta as especificações do osciloscópio e das ponteiras de corrente utilizadas nas medições e medições de tensão PWM aplicada nos motores e as formas de onda da corrente.

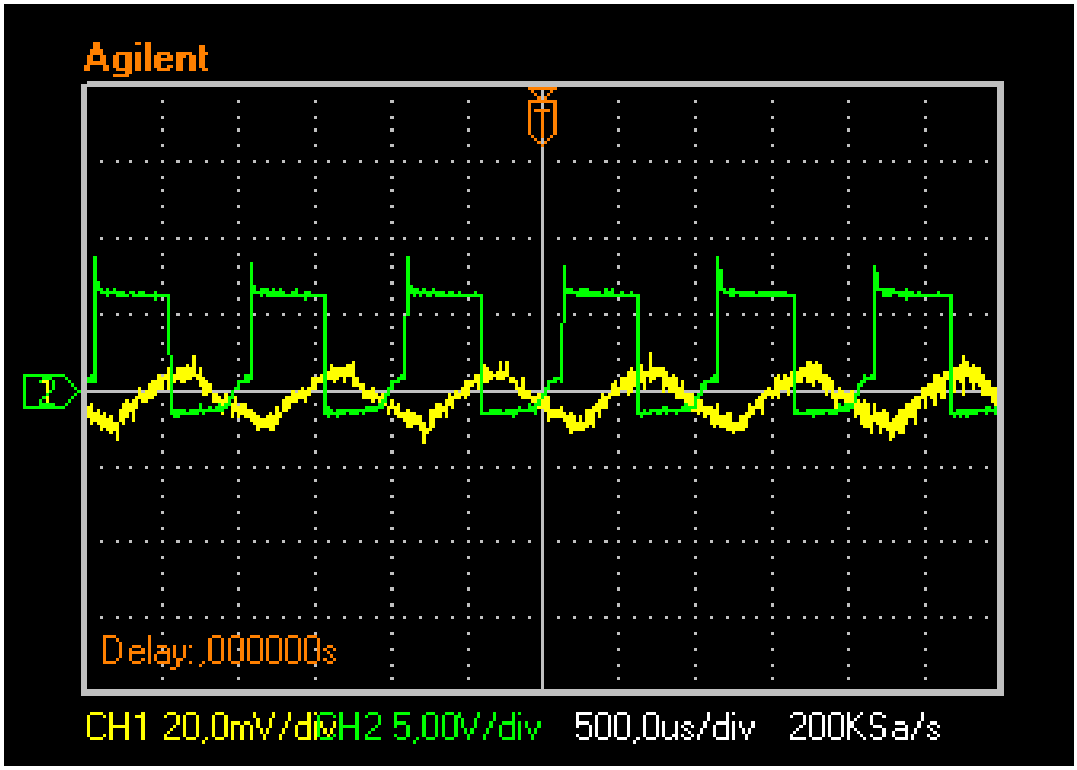

Figura 32: Imagem do osciloscópio mostrando a forma de onda do PWM com o método de acionamento utilizado.

4 INSTALANDO E CONFIGURANDO O ARDUINO, ZIBEE E O MATLAB

4.1 **Arduino**

Esse capítulo tratará da instalação do Arduino, configuração do Zigbee e instalação do pacote de suporte ao Arduino para Matlab no sistema operacional Windows da Microsoft. O Procedimento de instalação é o mesmo para todas as versões (foram testadas o Windows XP e o Windows 7). A versão atual Windows 8 ainda não tem suporte ao Arduino.

4.1.1 Instalando o Arduino.

Para instalar o Arduino no PC, é necessário baixar do *site* oficial [27] o pacote de *drives* que vem junto com a interface de desenvolvimento para Arduino. Todos os programas necessários estarão contidos na mídia que acompanha esse trabalho. Os passos para a instalação são os seguintes:

1. Conecte o Arduino e espere o Windows começar o processo de instalação.
2. Após alguns momentos o processo irá falhar.
3. Clique em menu iniciar e abra o painel de controle
4. No painel de controle, acesso item Sistema e logo após clique em Gerenciador de dispositivos.
5. Procure dentre os itens Portas de Comunicação (COM & LPT) um item chamado Arduino UNO (COMxx)
6. Clique com o botão direito nesse item e escolha “Atualizar Driver”
7. Escolha a opção “Navegar no meu computador em busca dos drivers”
8. Aponte para a subpasta Drivers e escolha o arquivo **"ArduinoUNO.inf"**,
9. O Windows terminará a instalação dos drivers

4.1.2 Instalando a interface de desenvolvimento do Arduino.

Para começar a utilizar a interface de desenvolvimento do Arduino não é necessária a instalação, bastando apenas descompactar o arquivo baixado anteriormente do site oficial e rodar o programa “arduino.exe”. Esse arquivo está contido dentro da pasta “Instaladores” na mídia que acompanha este trabalho.

4.2 **Instalando o suporte ao Arduino no Matlab.**

4.2.1 Instalando o pacote de suporte no Matlab

O pacote de suporte tem o nome ArduinoIO.zip e está localizado dentro da pasta instalador da mídia que acompanha este trabalho. Ao ser descompactado, a pasta conterá três arquivos com a extensão .m que se referem a arquivos de script do Matlab. O arquivo "**arduino.m**" possui as configurações que foram apresentadas e comparadas no capitulo 4. Não é necessário mais mexer nesse arquivo, pois ele já vem configurado. O outro arquivo de interesse é o "**instal_arduino.m**". Esse arquivo é o que configurará o Matlab com os atalhos e funções para a interface serial com o Arduino. Para instalar, basta abrir este *script* com o editor do Matlab e clicar na seta verde "Run install_arduino.m".

4.2.2 Instalando o arquivo server no Arduino.

O Arduino contará com um programa instalado em sua memória que será um servidor para receber e enviar comandos para o computador através da porta serial. A instalação desse programa é feita através de sua IDE. A figura 33 mostra a IDE com o arquivo "**adiosrv.pde**" que está contido na pasta adiosrv da pasta que contém o pacote ArduinoIO para suporte ao Matlab. A sigla que nomeia o arquivo adiosrv significa Servidor de entrada e saída analógico e digital (do inglês, "*Analog and Digital Input and Output Server*"), Sendo especificado nos comentários do programa como um servidor para Matlab das portas de entrada e saída.

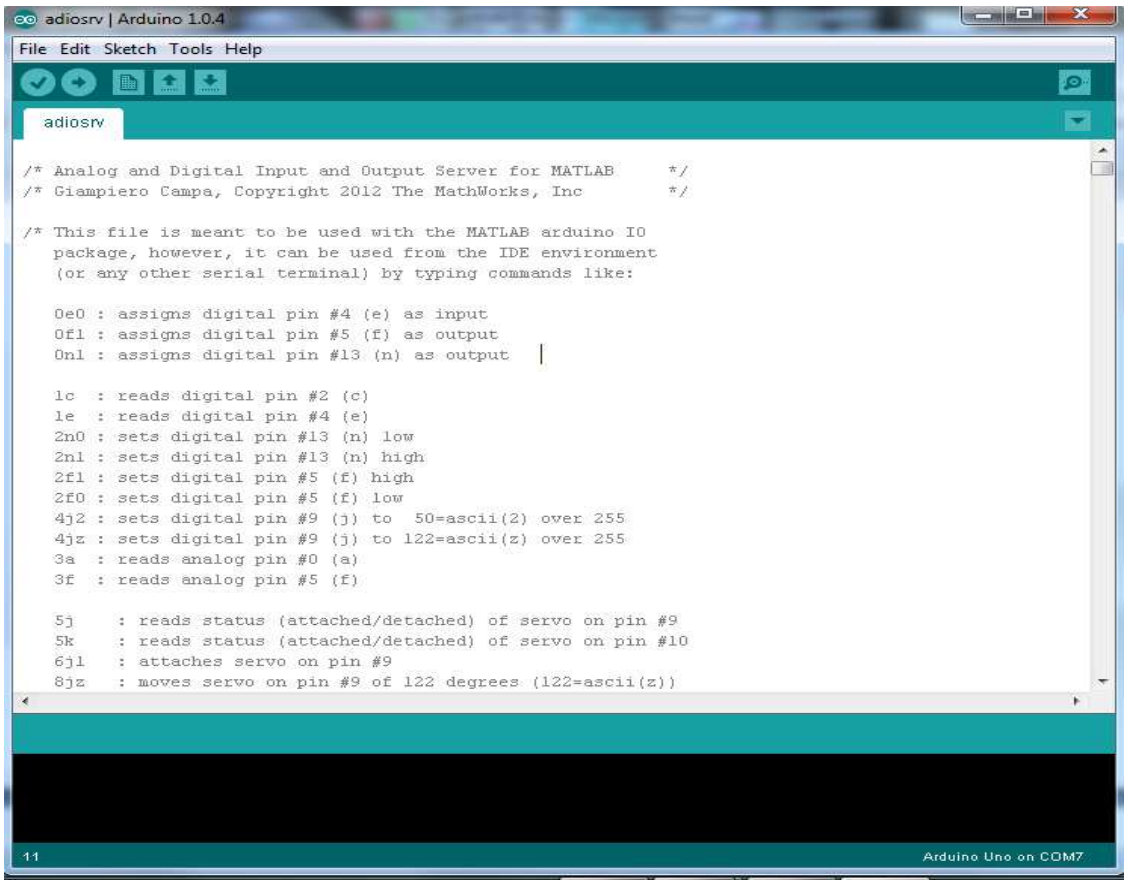

Figura 33: IDE do Arduino com o programa servidor carregado.

Após o código ser compilado e gravado na eeprom do Arduino, o mesmo estará pronto para conectar-se com o Matlab e enviar a receber comandos para suas portas analógicas e digitais.

4.3 **Configurando o Zigbee**

4.3.1 Ligação Física.

Como não foi possível adquirir a tempo o USB dongle para utilizar o módulo Xbee no computador, foi utilizada a conversão FTDI – USB presente na placa de desenvolvimento Arduino. Para tal, é necessário retirar o microcontrolador e deixar apenas o *shield* com o módulo Xbee instalado. O shield Xbee possui dois jumpers que necessitam estar na posição USB para serem configurados no PC. Quando o módulo for usado no Arduino que estiver montado no barco, ambos os jumper nesse *shield* precisam estar na posição XBEE. O módulo

que enviará os comandos, montado na placa Arduino que ficará ligado ao computador continuará na posição USB. A figura 34 mostra essa ligação feita. Não é possível visualizar a ausência do microcontrolador, pois o shield está montado em cima de onde se localiza o chip.

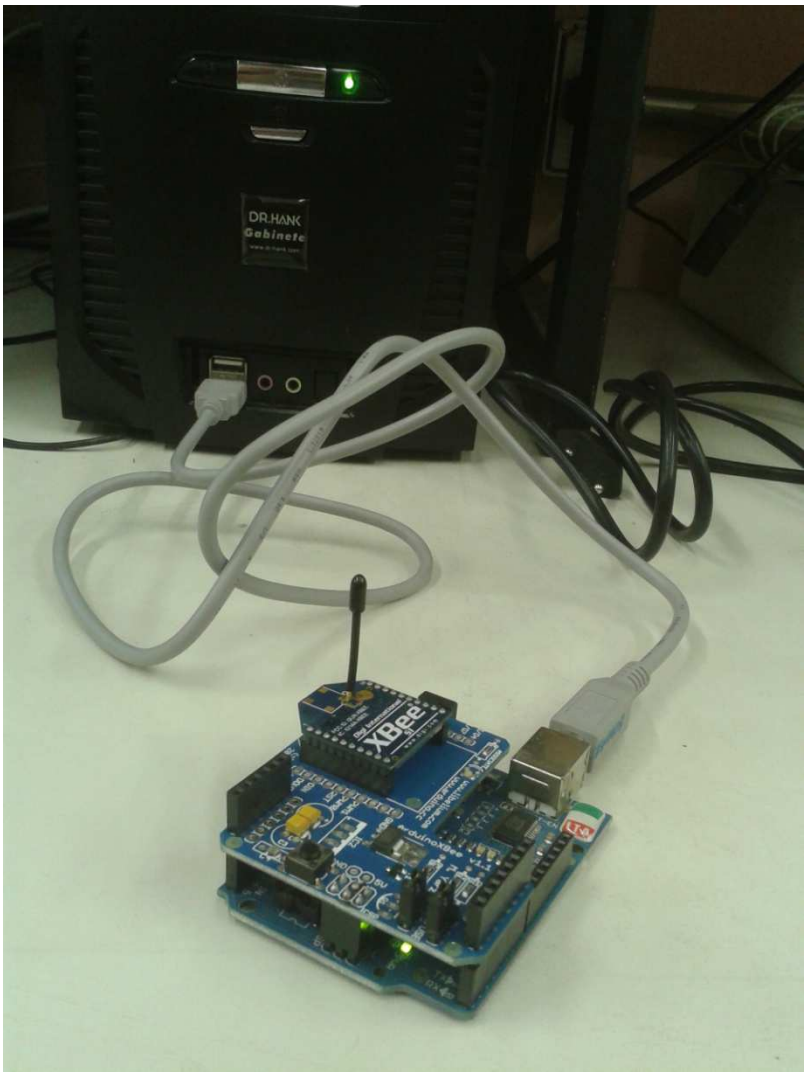

Figura 34: Xbee ligado a uma placa Arduino sem o microcontrolador.

4.3.2 Configuração do Xbee

Para configurar os parâmetros de interesse do Xbee, é necessário instalar o programa X-CTU, um programa desenvolvido pela Digi [20]. Esse programa tem como função alterar os parâmetros padrões do *firmware* Xbee.

Depois de instalado, ao iniciar-se o programa com o módulo Xbee conectado ao computador, o programa apresentará a tela mostrada na figura 35.

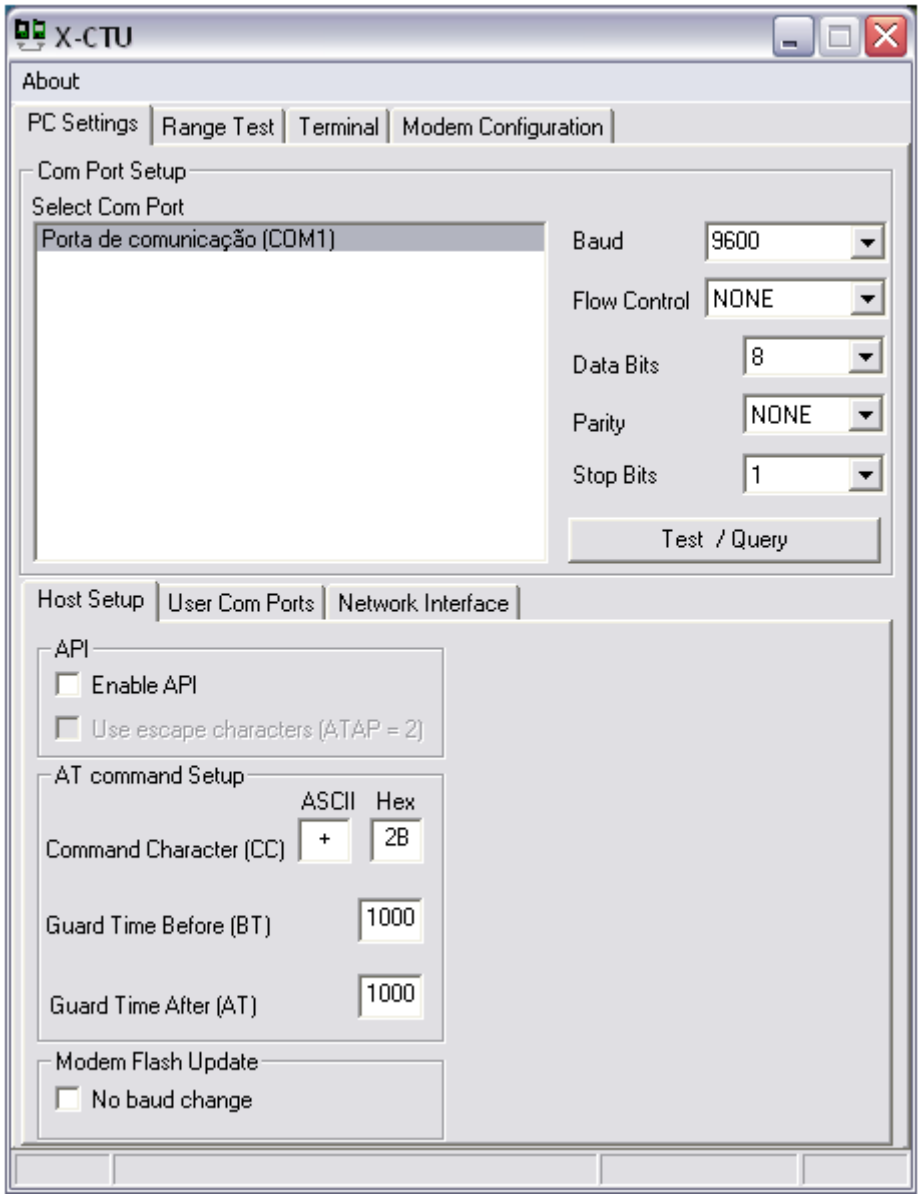

Figura 35: Tela inicial do programa para configurar o Xbee.

As configurações abaixo são realizadas conectando o *shield* desejado (seja o transmissor base ou a receptora remota) como mostra a figura 35. Ambos os módulos precisam ser configurados antes de serem utilizados.

4.3.2.1 Configuração do módulo transmissor base.

As configurações são feitas na aba *modem configuration*, como mostrado na figura 36:

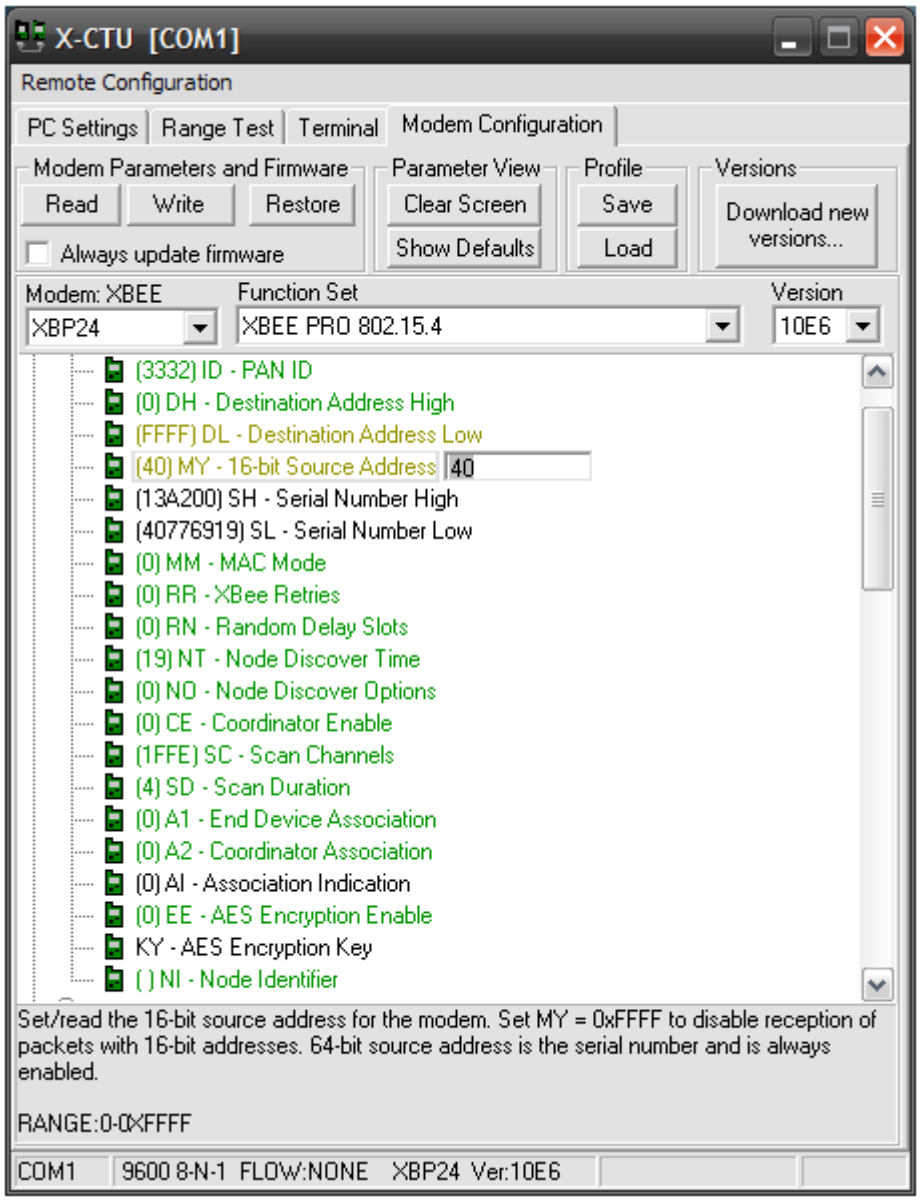

Figura 36: Aba *modem configuration.*

A topologia básica Zigbee usada será nesse projeto é a ponto-a-ponto, a única disponível para os módulos Zigbees da série 1. As outras topologias só estão incluídas a partir da série 2.

Os parâmetros que precisam ser modificados e seus respectivos valores para o Xbee base são:

1. DL – *Destination Address Low* com o valor Hexa de FFFF
2. MY – 16 *Bit Source Address* – Um valor hexadecimal de 16 Bits. Esse valor corresponde ao “nome” ou “endereço” do módulo base a que cada Xbee remoto se referenciará.
3. PAN ID – Deve ser um valor hexa de 32 bits para identificar a rede.
4. Interface Data Rate – A velocidade da comunicação serial que será usada. Deverá ser configurada para 115,200 bps.

A figura 37 não mostra na imagem essa opção, mas nota-se que existe uma barra de rolagem e esta opção está logo após as exibidas.

4.3.2.2 – Configuração do módulo transmissor remoto.

No dispositivo remoto, os seguintes valores devem ser modificados:

1. DL – *Destination Address Low* com o valor Hexa configurado no campo MY – 16 *Bit Source Address* configurado anteriormente no modulo base.
2. PAN ID – Deve ser um valor hexa de 32 bits para identificar a rede. Precisa ser o mesmo colocado no base.
3. MY – 16 *Bit Source Address* – Valor que identifica o modulo na rede. Será o valor subseqüente ao valor configurado para o módulo base.
4. Interface Data Rate – A velocidade da comunicação serial que será usada. Deverá ser configurada para 115,200 bps

4.4 **Conexão com o Matlab.**

Com todos os softwares instalados e hardwares configurados, para estabelecer a conexão com o Matlab, o comando utilizado na janela *command* do editor é a= arduino(‘comX’) , onde X será a porta que o Arduino foi instalado no sistema operacional.

O pacote de suporte ao Arduino para o Matlab, além dos programas de configuração, possui um tutorial em arquivo que explica e exemplifica os principais usos do Arduino com o Matlab, como leitura e escrita de portas analógicas e digitais. Esse arquivo chamado “ArduinoIO.ppt” está contido na mídia que acompanha este trabalho.

5 **TESTES E MEDIÇÕES**

5.1 **Metodologia**

Foram realizadas quatro medições das formas de onda PWM aplicadas aos motores e das respectivas formas das ondas de corrente, cujos valores PWM aplicados variam de 105 até 255. Esses valores podem variar entre 0 até 255, representados com oito bits (em valores binários, $2^8 = 256$ que é valor que o ATMEGA, microcontrolador usando no Arduino permite representar nos seus conversores analógico-digitais).

O valor inicial escolhido foi de 105, pois valores menores aplicados as saídas de PWM não conseguiam vencer a inércia do motor, ou seja, ele continuava parado. Esse valor variou um pouco do motor lateral superior para o motor traseiro (e também variará do motor que não foi utilizado), pois a falta de lubrificação se dá de maneira desigual e ocasionou essa disparidade de valores para tirarem os motores do repouso. Outro fator determinante para valores elevados para iniciar a movimentação do motor são as hélices colocadas nos propulsores. Essas hélices são feitas de chumbo, o que torna necessário valores maiores de PWM. Uma das propostas seria a mudança dessas hélices para plástico, o que reduziria a necessidade de valores elevados de PWM para iniciar o movimento dos propulsores.

Todos os testes foram feitos em vazio, i.e, a embarcação estava suspensa e os motores estavam girando sem carga, ou seja, sem água, sendo a única carga a hélice de chumbo coloca na extremidade da haste ligada nos propulsores.

O osciloscópio utilizado foi da fabricante Agilent Technologies modelo DSO3102A, que permite ser ligado a um computador através de um cabo USB e, utilizando um software próprio, é possível obter as formas de ondas, sendo os valores medidos diretamente salvos no computador em formato de imagem e valores em formato de tabela. A figura 37 mostra o osciloscópio usado.

Figura 37: Osciloscópio usado nas medições.

Para a medição da corrente, foi utilizada uma ponteira de medição de corrente, da Agilent Technologies modelo 1146A, utilizando a escala 100mv/A.. A medição foi realizada envolvendo com a ponta do instrumento um dos fios que ligam o motor. A figura 38 mostra a ponteira utilizada nos teste.

Figura 38: Ponteira para medição de corrente utilizada nos teste.

A fonte de tensão utilizada para alimentar o drive do motor é da ICEL, modelo PS-6100 e é mostrada na figura 39. A tensão usada foi de 7,2V. Esse valor foi escolhido pois é um valor múltiplo da tensão de alimentação de pilhas recarregáveis(6 pilhas). Nos testes na água, esse valor mostrou-se muito baixo para a velocidade do motor e conclui-se que o valor ideal serial o valor de 9,6 V (8 pilhas).

Figura 39: Fonte de tensão utilizada para alimentar o *driver* dos motores para os testes.

5.2 **Medições**

5.2.1 Valor PWM de 105, *duty cicle* de 42%.

Esse foi o primeiro valor testado em que foi percebida a movimentação do motor. A figura 40 mostra a forma de onda da tensão PWM aplicada no motor em verde e a forma de onda da corrente em amarelo.

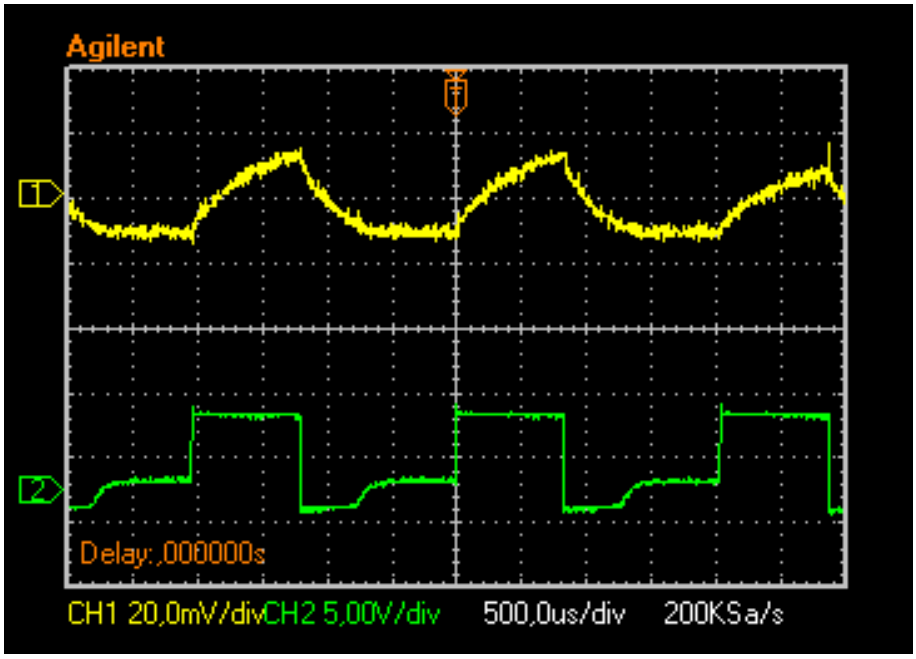

Figura 40: Formas de onda de tensão e corrente para um acionamento em PWM no valor de 105 (duty cicle de 42%).

5.2.2 Valor PWM de 165, *duty cicle* de 65%.

A figura 41 mostra a forma de onda do PWM aplicada ao motor em verde e a corrente em amarelo.

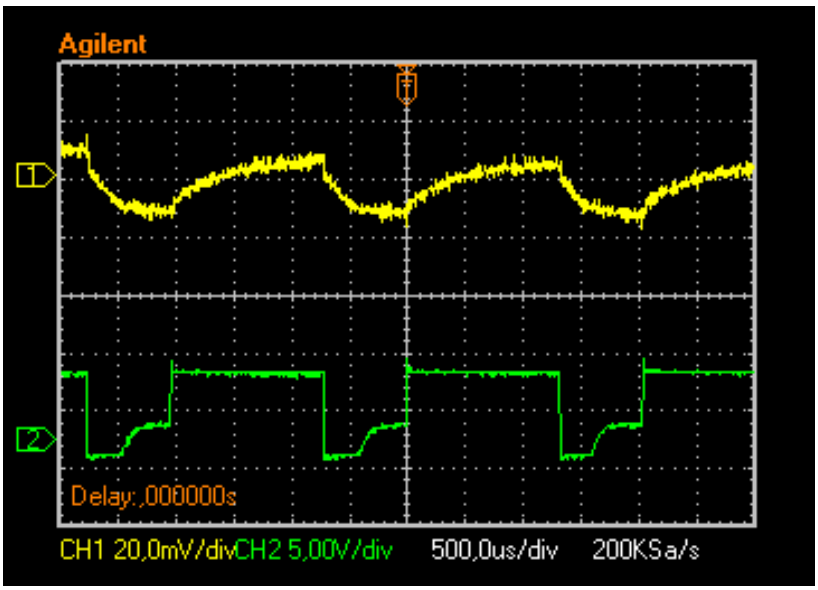

Figura 41: Formas de onda de tensão e corrente para um acionamento em PWM no valor de 165 (duty cicle de 65%).

5.2.3 Valor PWM de 210, *duty cicle* de 82%.

A figura 42 mostra a forma de onda do PWM aplicada ao motor em verde e a corrente em amarelo.

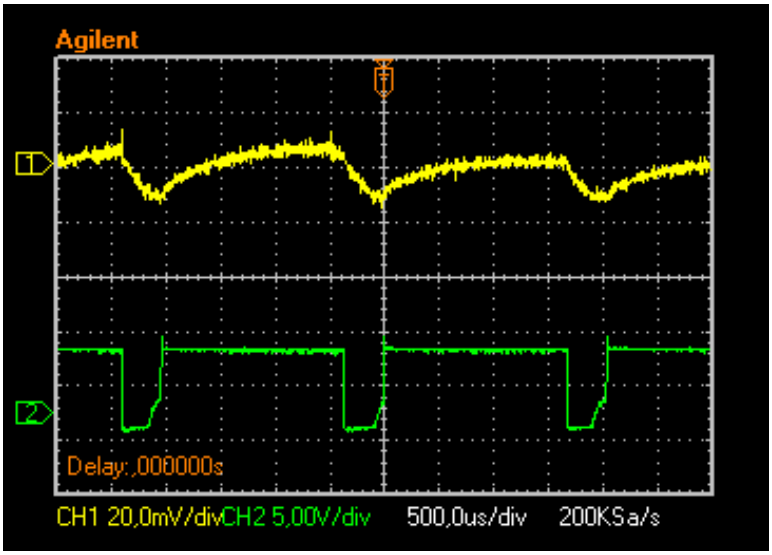

Figura 42: Formas de onda de tensão e corrente para um acionamento em PWM no valor de 210 (duty cicle de 82%).

5.2.4 Valor PWM de 255, *duty cicle* de 100%.

A figura 43 mostra a forma de onda do PWM aplicada ao motor em verde e a corrente em amarelo.

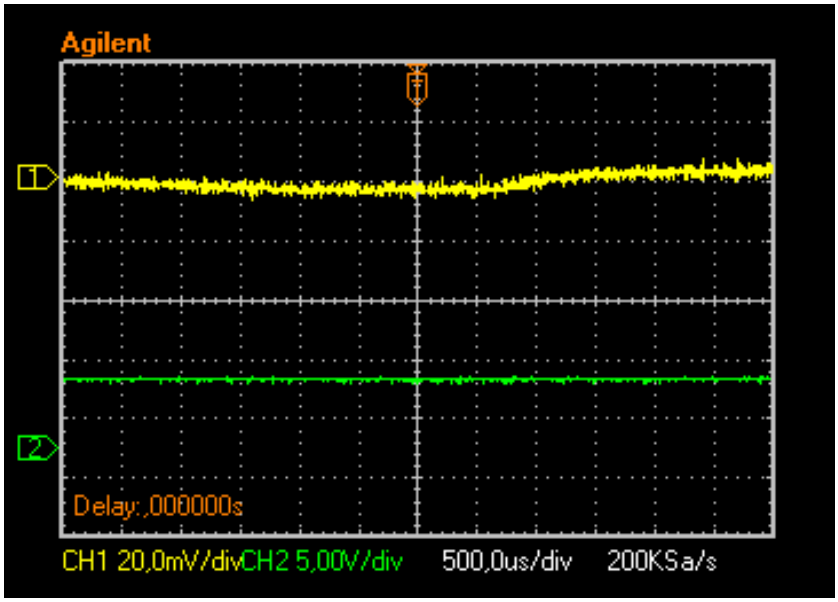

Figura 43: Formas de onda de tensão e corrente para um acionamento em PWM no valor de 255 (duty cicle de 100%).

5.2.5 Valor PWM de 0, *duty cicle* de 0%.

A figura 44 mostra a forma de onda do PWM aplicada ao motor em verde e a corrente em amarelo.

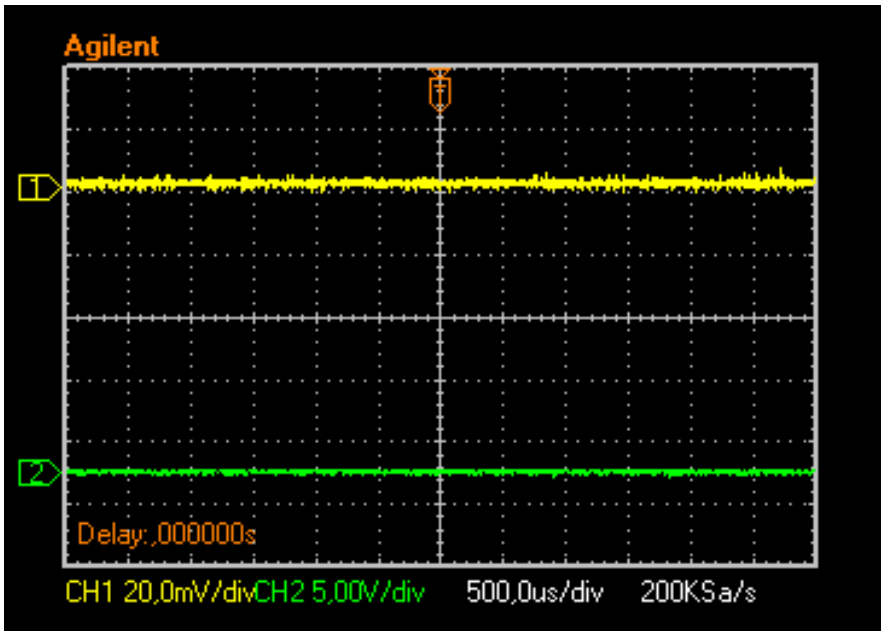

Figura 44: Formas de onda de tensão e corrente para um acionamento em PWM no valor de 0 (duty cicle de 0%).

A ponteira de correte foi utilizada na escala 100mV/A. A tensão média medida no segundo canal usando a ponteira de corrente não ultrapassou no osciloscópio o valor de 10mV. Realizando a conversão, obtemos um valor médio para um motor funcionando por volta de 100mA. A corrente média para dois motores foi medida através do display da fonte de alimentação utilizada. Para os dois motores funcionando, a corrente média medida foi 185mA. Essa diferença é devido ao aspecto já citado anteriormente que difere de cada motor, como lubrificação e limpeza.

6 **INTERFACE GRÁFICA DE USUÁRIO E CONTROLE DO BARCO**

6.1 **GUI**

O Matlab possui sua própria linguagem de programação orientada a objetos para a criação de interfaces gráficas (do inglês GUI, graphical user interface). O intuito dessas interfaces gráficas é dar uma resposta visual ao operador do programa que esteja rodando e facilitar o uso através de botões, janelas, barras de rolagem e o desenho dos gráficos que respondem aos comandos do usuário [31].

Neste projeto a interface gráfica foi criada com a intenção de realizar o controle do barco de forma interativa. Ao invés de serem digitados comandos para ligar e selecionar a velocidade de cada motor e direção, basta simplesmente clicar com o botão do mouse no comando desejado na GUI. A figura 45 mostra a *GUI* que foi criada para o controle do barco.

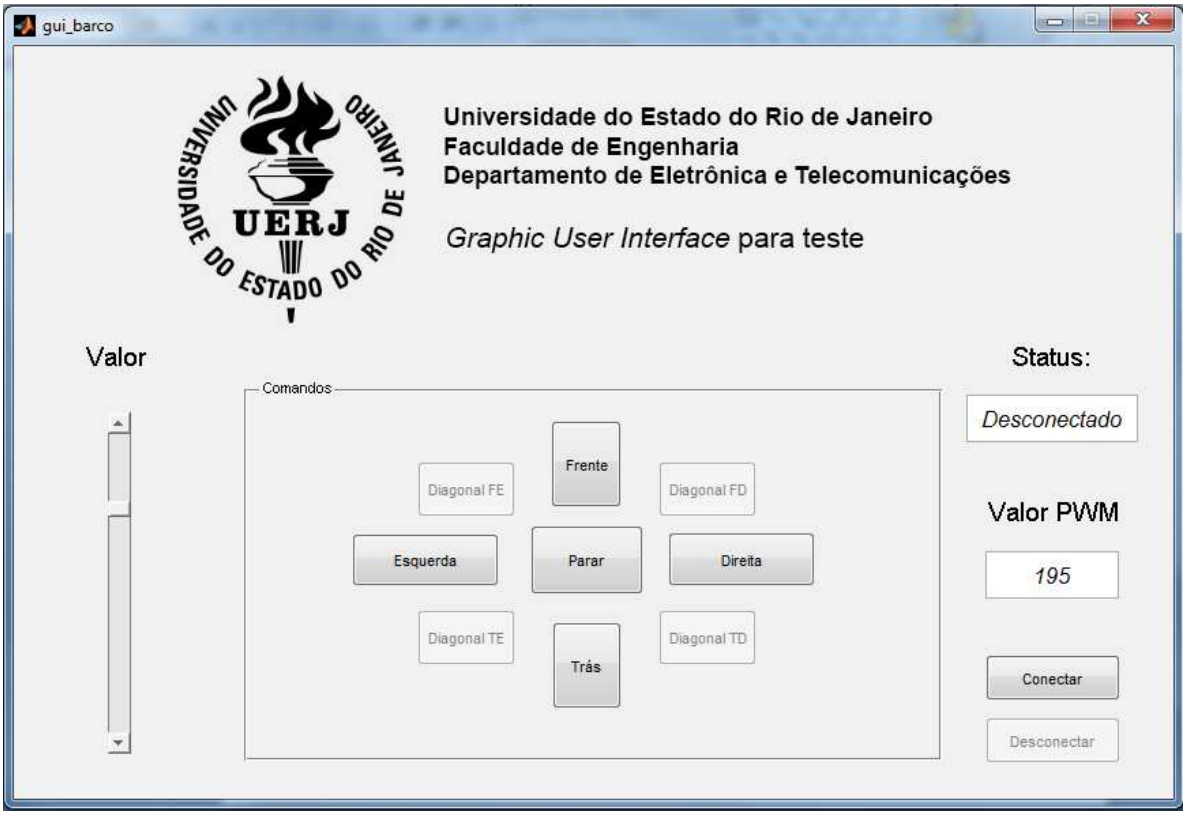

Figura 45: GUI criada para o controle do barco.

Como é possível ver na figura 45, existe uma barra de rolagem que possíblita selecionar o valor de PWM que será atribuído a um motor do barco. Esse valor escolhido é indicado pela *textBox* (um dos muitos componentes com o que é possível se criar a *GUI*). Também foi criado um indicador de *status*, que mostra se o barco está atualmente conectado ou desconectado do computador em terra. Os botões conectar e desconectar alternam sua disponibilidade dependendo do status atual da embarcação.

Além desses indicadores visuais, a GUI permite que se crie atalhos no teclado para que quando um determinada tecla for pressionada no teclado, a GUI executará uma série de comandos. Para a demonstração da embarcação, foram configuradas as teclas W, A, S e D para executarem as funções que estão disponíveis na interface principal da GUI, que são respectivamente mover o barco para frente, para esquerda, para trás e para direita. Existe uma função tempo em que a GUI possui uma função para executar uma ação ao se pressionar uma tecla, existe uma função para executar uma ação ao se soltar uma tecla qualquer do teclado. Foi configurado para ao se soltar qualquer tecla do teclado, a GUI executará a função PARAR.

Todo o código da GUI esta no apêndice A deste trabalho e os arquivos da *GUI* na mídia que acompanha este trabalho.

6.2 **Uso de um *Joystick***

Para uma demonstração alternativa do controle do barco, foi utilizado um programa de computador chamado JoytoKey, que é capaz de mapear qualquer tecla do teclado para um controle instalado no computador. Com isso, foi possível movimentar o barco com *joystick*, tornando mais intuitivo a movimentação do barco. O programa é *freeware (*gratuito e com livre distribuição) e esta disponível na mídia que acompanha este trabalho assim como um vídeo demonstrando o uso de um controle para direcionar a embarcação e os arquivos da interface gráfica.

7 CONCLUSÕES E PROPOSTAS FUTURAS

7.1 Conclusões

O protocolo Zigbee se mostrou o mais eficiente e com o melhor desempenho e confiabilidade, considerando a facilidade e a estabilidade de conexão em relação ao Bluetooth. Infelizmente não foi possível testar o Wi-Fi por não ter-se a disposição os módulos e, apesar do estudo comparativo mostrar que possui o maior consumo de energia dentre os quarto protocolos estudados, seria bom ter uma alternativa de conexão, apesar de que não seria ideal para um sistema móvel com alimentação por bateria.

A embarcação foi testada na piscina disponível no LEPAT (Laboratório de Eletrônica de potência e automação) com sucesso. A comunicação sem fio entre o barco e o computador funcionou e o acionamento foi realizado com o uso de uma interface gráfica no Matlab e utilizando um *joystick* para a demonstração

7.2 Propostas para trabalhos futuros

Com as medições de corrente realizadas no Capitulo 6, será possível dimensionar a bateria para uma autonomia desejada para o uso dentro do laboratório e em campo.

Os sensores apresentados podem ser implementados no barco e com o uso do Matlab é e uma interface gráfica, podem ser exibir gráficos e indicadores visuais dos valores obtidos diretamente da embarcação, como orientação, inclinação, velocidade e posição.

Para o acionamento dos 3 motores, seria necessário desenvolver um circuito acionador como o utilizado em [21] , porém exigindo maior esforço do projetista em utilizar o menor espaço possível para a alocação dos componentes, levando em consideração que os CI L298N precisam de dissipadores de calor. Esse circuito poderia ser projetado com a ajuda de um programa para projeto em computador (CAD, do inglês, Computer Aided Design), como por exemplo, o EAGLE.

Quando o barco estiver preparado para o uso em ambientes externos, seria conveniente o uso de uma câmera para fotografar o ambiente em que ele se encontra o que vai de encontro com a proposta do projeto do DETEL de embarcações tele-operadas para uso em monitoramento ambiental e defesa.

REFERÊNCIAS

[1] Modelo Reduzido. Página consultada em 20 de Janeiro 2013, Disponível em: en.wikipedia.org/wiki/Similitude_(model)

[2] Modelo Reduzido. Página consultada em 20 de Janeiro 2013, Disponível em: http://www.comciencia.br/comciencia/?section=3&noticia=604

[3] Modelo reduzido de hidroelétricas e turbinas. Página consultada em 20 de Janeiro 2013, Disponível em: pt.wikipedia.org/wiki/Laboratório_de_Hidráulica_Experimental_e_Recursos_Hídricos

[4] Informações sobre a usina de Colider. Página consultada em 20 de Janeiro 2013, Disponível em: http://www.copel.com/uhecolider/

[5] Informações sobre Zigbee. Página consultada em 14 de Novembro 2012. Disponível em: http://www.rogercom.com/ZigBee/ZigBee.htm

[6] Microntroladores. Página consultada em 14 de Novembro 2012, Disponível em: http://en.wikipedia.org/wiki/Microcontroller

[7] Sobre Arduino e causa do seu sucesso. Página consultada em 16 de Novembro 2012, Disponível em: http://www.newtoncbraga.com.br/index.php/microcontroladores/105-atmel/986-a-onda-do-arduino-col001.html

[8] Introdução ao Arduino. Página consultada em 16 de Novembro 2012, Disponível em: http://www.sabereletronica.com.br/secoes/leitura/1307

[9] Lista com as versões disponíveis da plataforma Arduino e acessórios. Página consultada em 16 de Novembro 2012, Disponível em: http://arduino.cc/en/Main/Hardware

[10] Sensores. Página consultada em 22 de Novembro 2012, Disponível em: http://en.wikipedia.org/wiki/Sensor

[11] Bússolas. Página consultada em 22 de Novembro 2012, Disponível em: http://en.wikipedia.org/wiki/Compass

[12] Protocolo I²C. Página consultada em 22 de Novembro 2012, Disponível em: http://en.wikipedia.org/wiki/I²C

[13] David Halliday, Robert Resnick, Jearl Walker - Fundamentos de Física - Mecânica - Vol.1 – Editora LTC – 2012

[14] Molas. Página consultada em 25 de Fevereiro 2013, Disponível em: http://www.ebah.com.br/content/ABAAABgTAAH/molas

[15] Como o GPS funciona. Página consultada em 22 de Novembro 2012, Disponível em: http://www.gpstesouro.com/Outros/Como%20funciona%20o%20GPS.htm

[16] Como funcionam os receptores de GPS. Página consultada em 22 de Novembro 2012, Disponível em: http://informatica.hsw.uol.com.br/receptores-gps.htm

[17] Site oficial do protocolo Zigbee. Página consultada em 22 de Novembro 2012, Disponível em: http://www.zigbee.org/

[18] A Comparative Study of Wireless Protocols Bluetooth, UWB, ZigBee, and Wi-Fi - Jin-Shyan Lee, Yu-Wei Su, and Chung-Chou Shen - Industrial Technology Research Institute (ITRI) - Taiwan

[19] Site em português sobre Zigbee. Página consultada em 15 de Junho 2012, Disponível em: http://www.rogercom.com/ZigBee/ZigBee.htm

[20] Site com mais informações sobre os módulos, *whitepapers* e *data-sheets* dos módulos Xbee. Página consultada em 15 de Junho 2012, Disponível em: http://www.digi.com

[21] Gustavo de Sá Amaral,Sistema de Posicionamento Dinâmico para um Pequeno Veículo Flutuante, Projeto de Graduação – DETEL - 2008.UERJ. Disponível em http://www.lee.eng.uerj.br/~jpaulo/PG/2008/PG-Posicionamento-Dinamico-2008.pdf

[22] Electronics and Circuit Analysis using MATLAB, Attia; John Okyere

[23] Porta serial no Matlab. Página consultada em 05 de Janeiro 2013, Disponível em: http://www.mathworks.com/help/matlab/matlab_external/overview-of-the-serial-port.html

[24] EIA - Electronic Industries Alliance. Página consultada em 05 de Janeiro 2013, Disponível em: **http://en.wikipedia.org/wiki/Electronic_Industries_Alliance**

[25] TIA/EIA-232-F, *Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange*

[26] Diferença entre bit rate e baud rate.Página consultada em 30 de Janeiro 2013, Disponível em: http://www.paulotrentin.com.br/eletronica/diferenca-entre-bit-rate-e-baud-rate/

[27] Site oficial *Matlab*. Página consultada em 20 de Março 2013, Disponível em: **http://www.mathworks.com/**

[28] *Power Electronics: Circuits, Devices and Application* - Muhammad Rashid

[29] *Datasheet* do CI L298. Página consultada em 20 de Março 2013, Disponível em: **http://www.datasheetcatalog.com/datasheets_pdf/L/2/9/8/L298N.shtml**

[30] Site oficial Arduino. Página consultada em 20 de Março 2013, Disponível em: **http://www.arduino.cc/**

[31] Matlab 6 – Curso Completo – Hanselman , Littlefield 2006

[32] Creative Commons. Página consultada em 16 de abril de 2013, Disponível em: http://creativecommons.org/

APÊNDICE A: Interface Gráfica GUI para o Matlab.

Tabela 8: Programa codificado no Matlab usado para gerar a *GUI.*

```
% Universidade do Estado do Rio de Janeiro
% FEN - Faculdade de Engenharia
% DETEL - Deparatmento de Telecomunicações e Eletrônic

% Angelo Sabbatini Buoro
% Rascunho: Interface Gráfica para controle do Barco
%
% Start of initialization code - DO NOT EDIT
function varargout = gui_barco(varargin)
%GUI_BARCO M-file for gui_barco.fig
%      GUI_BARCO, by itself, creates a new GUI_BARCO or raises the
existing
%      singleton*.
%
%      H = GUI_BARCO returns the handle to a new GUI_BARCO or the handle
to
%      the existing singleton*.
%
%      GUI_BARCO('Property','Value',...) creates a new GUI_BARCO using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to gui_barco_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      GUI_BARCO('CALLBACK') and GUI_BARCO('CALLBACK',hObject,...) call
the
%      local function named CALLBACK in GUI_BARCO.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui_barco

% Last Modified by GUIDE v2.5 13-Apr-2013 22:32:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_barco_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_barco_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End of initialization code - DO NOT EDIT

% --- Executes just before gui_barco is made visible.
function gui_barco_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for gui_barco
handles.output = hObject;
% Update handles structure
set(handles.pushbutton13,'Enable','off') % Torna o botão 'Desconectar'
desligado.
set(handles.pushbutton8,'Enable','off') % Torna o botão  8 desligado.
set(handles.pushbutton9,'Enable','off') % Torna o botão  9 desligado.
set(handles.pushbutton10,'Enable','off') % Torna o botão 10 desconectar
desligado.
set(handles.pushbutton11,'Enable','off') % Torna o botão 11 desconectar
desligado.
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = gui_barco_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;




% Configuração das funções da GUI começam aqui.

    % Barra de Rolagem para seleção do valor PWM

        % Configuração do slider
            function slider2_Callback(hObject, eventdata, handles)
            global b;global f;global h;             % Cria as variáveis
que serão utilizadas pelo slider.
            b = get(handles.slider2,'Value');       % Captura o valor
atual do slider
            h = floor(b);                           % Truca e arredonda
para baixo o valor do slider. É necessário pois o valor PWM precisa ser
inteiro
            %set(handles.text7,'string',h);         % Mostra o valor do
slider no campo configurado, caixa text7.
            set(handles.edit3,'string',h);
            guidata(hObject, handles);              %Código da GUI.

            function slider2_CreateFcn(hObject, eventdata, handles)
            if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
                set(hObject,'BackgroundColor',[.9 .9 .9]);  % Formatação
do slider.

            end
```

```
% %   Botões para os comandos de direção do Barco

        %  Frente
            function pushbutton1_Callback(hObject, eventdata, handles)
            global a;global f;                          %Cria as variáveis
que serão utilizadas
            a.analogWrite(11,0);a.analogWrite(10,0);    %Interrompe o motor
traseiro
            a.analogWrite(6,0);a.analogWrite(5,0);      %Interrompe o motor
lateral
            a.digitalWrite(4,0);                        %Desliga o motor
lateral
            a.digitalWrite(12,1);                       %Liga apenas o
motor traseiro
            f = get(handles.slider2,'Value');           %Lê o valor
colocado no slider.
            F = floor(f)                                %Trunca e arredonda
para baixo o valor do slider.
            a.analogWrite(10,F);a.analogWrite(11,0);    %Movimenta o barco
para frente com o valor PWM do slider


        %  Direita
            function pushbutton2_Callback(hObject, eventdata, handles)
            global a;global f;                          %Cria as variáveis
que serão utilizadas
            a.analogWrite(11,0);a.analogWrite(10,0);    %Interrompe o motor
traseiro
            a.analogWrite(6,0);a.analogWrite(5,0);      %Interrompe o motor
lateral
            a.digitalWrite(12,0);                       %Desliga o motor
traseiro
            a.digitalWrite(4,1);                        %Liga apenas o
motor lateral
            f = get(handles.slider2,'Value');           %Lê o valor
colocado no slider.
            F = floor(f)                                %Trunca e arredonda
para baixo o valor do slider.
            a.analogWrite(6,0);a.analogWrite(5,F);      %Movimenta o barco
para direita com o valor PWM do slider


        %  Esquerda
            function pushbutton3_Callback(hObject, eventdata, handles)
            global a;global f;                          %Cria as variáveis
que serão utilizadas
            a.analogWrite(11,0);a.analogWrite(10,0);    %Interrompe o motor
traseiro
            a.analogWrite(6,0);a.analogWrite(5,0);      %Interrompe o motor
lateral
            a.digitalWrite(12,0);                       %Desliga o motor
traseiro
            a.digitalWrite(4,1);                        %Liga apenas o
motor lateral
            f = get(handles.slider2,'Value');           %Lê o valor
colocado no slider
            F = floor(f)                                %Trunca e arredonda
```

```
para baixo o valor do slider.
            a.analogWrite(6,F);a.analogWrite(5,0);     %Movimenta o barco
para esquerda com o valor PWM do slider



        %  Trás
            function pushbutton4_Callback(hObject, eventdata, handles)
            global a;global f;                          %Cria as variáveis
que serão utilizadas
            a.analogWrite(11,0);a.analogWrite(10,0);   %Interrompe o motor
traseiro
            a.analogWrite(6,0);a.analogWrite(5,0);     %Interrompe o motor
lateral
            a.digitalWrite(4,0);                        %Desliga o motor
lateral
            a.digitalWrite(12,1);                       %Liga apenas o
motor traseiro
            f = get(handles.slider2,'Value');          %Lê o valor
colocado no slider
            F = floor(f)                                %Trunca e arredonda
para baixo o valor do slider.
            a.analogWrite(10,0);a.analogWrite(11,F);   %Movimenta o barco
para frente com o valor PWM do slider



        %  Parar
            function pushbutton6_Callback(hObject, eventdata, handles)
            global a;global f;                          %Cria as variáveis
que serão utilizadas
            a.analogWrite(11,0);a.analogWrite(10,0);   %Interrompe o motor
traseiro
            a.analogWrite(6,0);a.analogWrite(5,0);     %Interrompe o motor
lateral
            a.digitalWrite(12,0);                       %Desliga o motor
lateral
            a.digitalWrite(4,0);                        %Desliga o motor
lateral


 % Ainda não implementado.Desativado por padrão.

        %  Diagonal Frente-esquerda
            function pushbutton8_Callback(hObject, eventdata, handles)
            global a;
            a.analogWrite(11,0);a.analogWrite(10,0);
            a.analogWrite(6,0);a.analogWrite(5,0);

        %  Diagonal Frente-direita
            function pushbutton9_Callback(hObject, eventdata, handles)
            global a;
            a.analogWrite(11,0);a.analogWrite(10,0);
            a.analogWrite(6,0);a.analogWrite(5,0);

        %  Diagonal Trás-direita
            function pushbutton10_Callback(hObject, eventdata, handles)
            global a;
            a.analogWrite(11,0);a.analogWrite(10,0);
            a.analogWrite(6,0);a.analogWrite(5,0);
```

```
        %  Diagonal Trás-esquerda
            function pushbutton11_Callback(hObject, eventdata, handles)
            global a;
            a.analogWrite(11,0);a.analogWrite(10,0);
            a.analogWrite(6,0);a.analogWrite(5,0);


% %   Fim dos botões para os comandos de direção do Barco



% Função para acionar o barco pelas teclas W(frente),A(esquerda),S(trás) e
D(direita).
            function figure1_KeyPressFcn(hObject, eventdata, handles)

            switch eventdata.Key

                % Todos os comandos são iguais ao dos botões da GUI.

               case 'w'
                    global a;
                    global f;
                    a.analogWrite(11,0);a.analogWrite(10,0);
                    a.analogWrite(6,0);a.analogWrite(5,0);
                    a.digitalWrite(4,0);
                    a.digitalWrite(12,1);
                    f = get(handles.slider2,'Value');
                    F = floor(f)
                    a.analogWrite(10,F);a.analogWrite(11,0);

                case 's'
                    global a;
                    global f;
                    a.analogWrite(11,0);a.analogWrite(10,0);
                    a.analogWrite(6,0);a.analogWrite(5,0);
                    a.digitalWrite(4,0);
                    a.digitalWrite(12,1);
                    f = get(handles.slider2,'Value');
                    F = floor(f)
                    a.analogWrite(10,0);a.analogWrite(11,F);

                case 'a'
                    global a;
                    global f;

a.analogWrite(11,0);a.analogWrite(10,0);a.digitalWrite(12,0);
                    a.digitalWrite(4,1);
                    f = get(handles.slider2,'Value');
                    F = floor(f)
                    a.analogWrite(6,F);a.analogWrite(5,0);

               case 'd'
                    global a;
                    global f;

a.analogWrite(11,0);a.analogWrite(10,0);a.digitalWrite(12,0);
                    a.digitalWrite(4,1);
                    f = get(handles.slider2,'Value');
                    F = floor(f)
```

```
                    a.analogWrite(6,0);a.analogWrite(5,F);

               case 'r'
                    global a;
                    global f;
                    a.analogWrite(11,0);a.analogWrite(10,0);
                    a.analogWrite(6,0);a.analogWrite(5,0);
                    a.digitalWrite(12,0);
                    a.digitalWrite(4,0);

          end



% Função para parar o barco quando soltar uma tecla.
          function figure1_KeyReleaseFcn(hObject, eventdata, handles)
          global a;
          global f;
          a.analogWrite(11,0);a.analogWrite(10,0);
          a.analogWrite(6,0);a.analogWrite(5,0);
          a.digitalWrite(12,0);
          a.digitalWrite(4,0);


% % Mostra a imagem para cabeçalho
          function axes6_CreateFcn(hObject, eventdata, handles)
          axes(hObject);
          imshow('titulo.bmp') % Carrega a imagem titulo.bmp.


% Conecta o Barco
          function pushbutton12_Callback(hObject, eventdata, handles)
          global a;
          a=arduino('com4');      % Valor precisa ser alterado
dependendo da porta em que o Arduino esteja instalado no computador.
          a.pinMode(12,'output'); % Configura a porta 12 como saida.
Essa porta liga ou desliga o motor traseiro.
          a.pinMode(11,'output'); % Configura a porta 11 como saida.
Essa porta controla a direção e velocidade com PWM.
          a.pinMode(10,'output'); % Configura a porta 10 como saida.
Essa porta controla a direção e velocidade com PWM.
          a.pinMode(4,'output');  % Configura a porta 4 como saida. Essa
porta liga ou desliga o motor lateral superior.
          a.pinMode(5,'output');  % Configura a porta 5 como saida. Essa
porta controla a direção e velocidade com PWM.
          a.pinMode(6,'output');  % Configura a porta 6 como saida. Essa
porta controla a direção e velocidade com PWM.
          a.digitalWrite(4,1);    % Ativa o motor lateral superior.
          a.digitalWrite(12,1);   % Ativa o motor central.
%           if a ~= 0
%             msg1='Conectado'
%             set(handles.edit2,'String',msg1);
%           end
          set(handles.pushbutton12,'Enable','off') % Torna o botão
Conectar desligado.
          set(handles.pushbutton13,'Enable','on') % Torna o botão
Desconectar ligado.
          msg1='Conectado';
          set(handles.edit2,'String',msg1); % TROCAR A ORDEM DEPOIS. SÓ
```

```
MUDAR DEPOIS DE CONECTADO


% Desconecta o Barco
            function pushbutton13_Callback(hObject, eventdata, handles)
            global a;
            a.analogWrite(11,0);a.analogWrite(10,0);
            a.analogWrite(6,0);a.analogWrite(5,0);
            a.digitalWrite(12,0);
            a.digitalWrite(4,0);
            delete(a);              % Deleta o objeto criado para o uso do
Arduino no Matlab
            set(handles.pushbutton12,'Enable','on') % Torna o botão
Conectar ligado.
            set(handles.pushbutton13,'Enable','off') % Torna o botão
Desconectar desligado.
            msg2='Desconectado';
            set(handles.edit2,'String',msg2);




% Function criada para o texto PWM
            function text7_CreateFcn(hObject, eventdata, handles)


% Textbox para configurar o estatus

function edit2_Callback(hObject, eventdata, handles)



% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%  Função para a caixa de edição.
function edit3_Callback(hObject, eventdata, handles)

function edit3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```