CENTRO DE INSTRUÇÃO
ALMIRANTE GRAÇA ARANHA - CIAGA
ESCOLA DE FORMAÇÃO DE OFICIAIS DA
MARINHA MERCANTE – EFOMM

IMPLEMENTAÇÃO DE PILOTO-AUTOMÁTICO EM UM ROBÔ MICROCONTROLADO.

Por: Rafael SADAO Carvalho Horita

Orientador
Prof. André Luís Mourilhe Rocha

Rio de Janeiro
2012

CENTRO DE INSTRUÇÃO
ALMIRANTE GRAÇA ARANHA - CIAGA
ESCOLA DE FORMAÇÃO DE OFICIAIS DA
MARINHA MERCANTE – EFOMM

IMPLEMENTAÇÃO DE PILOTO-AUTOMÁTICO EM UM ROBÔ MICROCONTROLADO.

Apresentação de monografia ao Centro de Instrução Almirante Graça Aranha como condição prévia para a conclusão do Curso de Bacharel em Ciência Náuticas do Curso de Formação de Oficiais de Máquinas (FOMQ) da Marinha Mercante.

Por: Rafael **SADAO** Carvalho Horita

CENTRO DE INSTRUÇÃO
ALMIRANTE GRAÇA ARANHA - CIAGA
ESCOLA DE FORMAÇÃO DE OFICIAIS DA
MARINHA MERCANTE – EFOMM

AVALIAÇÃO

PROFESSOR ORIENTADOR (trabalho escrito):_____

NOTA - _____

BANCA EXAMINADORA (apresentação oral):

_____

Professor (nome e titulação)

_____

Professor (nome e titulação)

_____

Professor (nome e titulação)

_____

NOTA: _____

DATA: _____

NOTA FINAL: _____

AGRADECIMENTOS

Agradeço a Deus, minha família, amigos e professores que me ajudaram a alcançar essa etapa.

DEDICATÓRIA

Dedico este documento aos meus pais,
Carla e Fabio que sempre me apoiaram.

RESUMO

Este projeto tem por objetivo desenvolver um sistema de navegação autônoma enaltecendo o conteúdo sobre veículos controlados remotamente e auto-orientado. Ele implementa o uso de GPS e bússola magnética digital para determinação de rumo e distância até seu alvo destino. Os dispositivos utilizados são de baixo custo e baixo consumo de energia. Os códigos-fontes presentes foram escritos na linguagem Python e C/C++. As partes físicas são compostas pela base e robô microcontrololados, sendo necessário um computador para conectar a base.

Palavras-chave: navegação autônoma, auto-orientado, microcontrolador.

ABSTRACT

This project aims to develop an autonomous navigation system extolling the content on remote controlled vehicles and self-oriented. It implements the use of GPS and digital magnetic compass to determine direction and distance to your target destination. The devices used are low cost and low power consumption. The present source codes were written in Python and C / C + +. The physical parts are made by the base and robot, requiring a computer to connect to the base.

Keywords: autonomous navigation, auto-oriented, microcontroller.

ÍNDICE DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A automação é altamente empregada nas praças de máquinas através de dispositivos como sensores e atuadores, juntamente com controladores. Porém, não é só na rotina do Oficial de Máquinas que tarefas foram automatizadas. Como um simplíssimo aparelho no passadiço, encontra-se o piloto-automático, que automatiza a rotina do Oficial de Náutica.

Um simples piloto-automático recebe informações do receptor do GPS (Global Position System), uma bússola digital e calcula o rumo atual da embarcação e tira sua diferença com o rumo de destino, mudando a posição do leme ou não. Quando a distancia mínima é alcançada o dispositivo além de avisar deverá reduzir a propulsão. Isto é, o destino é alcançado através da convergência do rumo atual para o rumo destino, assim ele é auto-alimentada a cada instante com informações do GPS e bússola pra se localizar e pode calcular sua distancia e rumo ao destino. É um aparelho que trabalha com sensores e atuadores, isto é, ele possui um sistema de controle com entrada de informações de um operador, é um aparelho que trabalha na fronteira do Oficial de Náutica e o Oficial de Máquinas. No entanto, o piloto-automático é como um tripulante obediente e incansável, mas cego, surdo e mudo. Ele não mudará o rumo só porque há um navio vindo. Logo, é confiável enquanto observado por um tripulante.

Outros segmentos também utilizam sistemas de piloto-automático. Na agricultura, as máquinas são equipadas para poder fazer a colheita ou o plantio utilizando o piloto-automático. Na aeronáutica, o sistema é comumente utilizado nas aerovias.

O projeto tem como propósito mostrar de uma forma simples um sistema de piloto-automático. Sendo assim, o projeto é composto por um robô, uma base e o programa de orientação. O robô é microcontrolado com transmissão de dados via radiofreqüência para base que um computador com viabilidade de plotar suas coordenadas num mapa.

2 DETALHAMENTO DO PROJETO

Para iniciar, encontra-se a seguir um Diagrama de Blocos geral do projeto, contendo os componentes a serem desenvolvidos, e logo após uma descrição detalhada do funcionamento tanto do hardware quanto de software de cada componente do projeto.

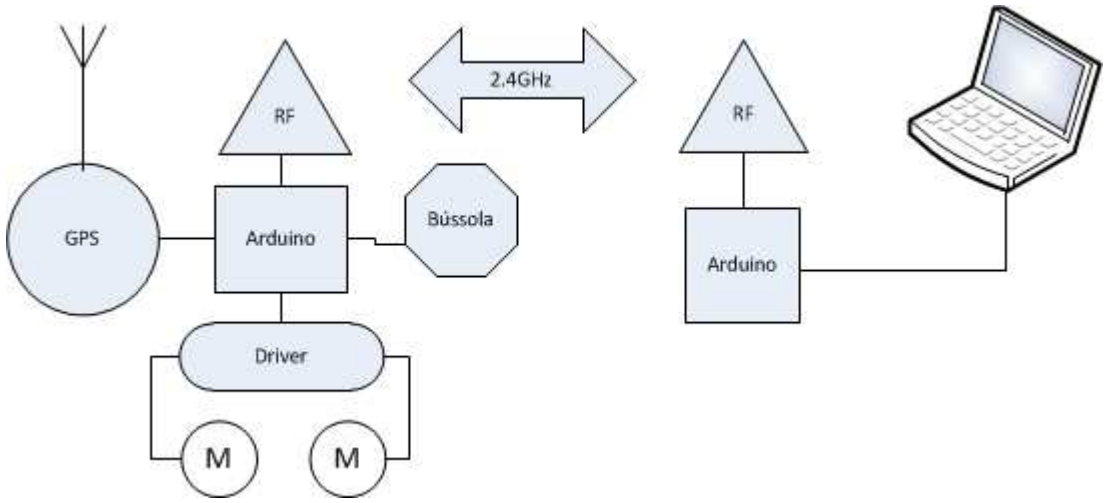

Figura 1 - Diagrama em blocos do projeto

2.1 Robô

O Robô é a unidade onde irá controlar e gerenciar os dados recebidos tanto pelo GPS, tanto pelos transceptores do radiofreqüência (RF). O Robô é composto por quatro dispositivos de hardware sendo eles: Arduino (ATMEL168), Driver dos Motores, dois motores DC, Receptor do GPS, bússola digital, e o módulo de RF (NRF24L01), os quais serão detalhados a seguir.

2.1.1 Arduino

É uma plataforma de computação física livre baseada numa simples placa com microcontrolador (ATMEL©) e um ambiente de desenvolvimento para escrever programas para placa. Tudo orientado por portas de entrada/saída analógico-digitais, utilizando como linguagem C/C++.

Para facilidade de atualizar, ou modificar o código, o Arduino possui gravado um *bootloader* na EEPROM. O compilador e responsável por enviar o arquivo compilado é o Arduino IDE, que utiliza o protocolo STK500.

Naturalmente, o mais interessante é a facilidade que possibilita programar microcontrolador com praticidade e rapidez. O Arduino utilizado é de fabricação com

placa *Single-Sided Serial* (SEVERINO V3.0) devido a facilidade de fabricação e de componentes facilmente encontrados em lojas de componentes eletrônicos. O SEVERINO possui algumas características descritas a seguir.

- Microcontrolador …….ATmega168
- Tensão de trabalho ……..….5V
- Tensão de entrada(recomendado) …...7-12 V
- Tensão de entrada (limites) ……..6-20 V
- Terminais Digitais E/S ……..20 (6 PWM)
- Terminais de Entrada Analógica ……..6
- CC por terminal de E/S ……40 mA
- Memoria Flash …….16 KB (4KB do *bootloader*)
- SRAM ….....1 KB
- EEPROM .....512 B
- Frequência de trabalho .....16 MHz

Os projetos e esquemas de hardwares são distribuídos sob a licença Creative Commons Attribution Share-Alike 2.5, e estão disponíveis em sua página oficial. Arquivos de layout e produção para esta versão estão hospedadas. O código fonte para o IDE e a biblioteca de funções da placa são disponibilizadas sob a licença GPLv2 e hospedadas pelo projeto Google Code. Na figura X encontra-se descritos o *layout* da placa do Arduino.

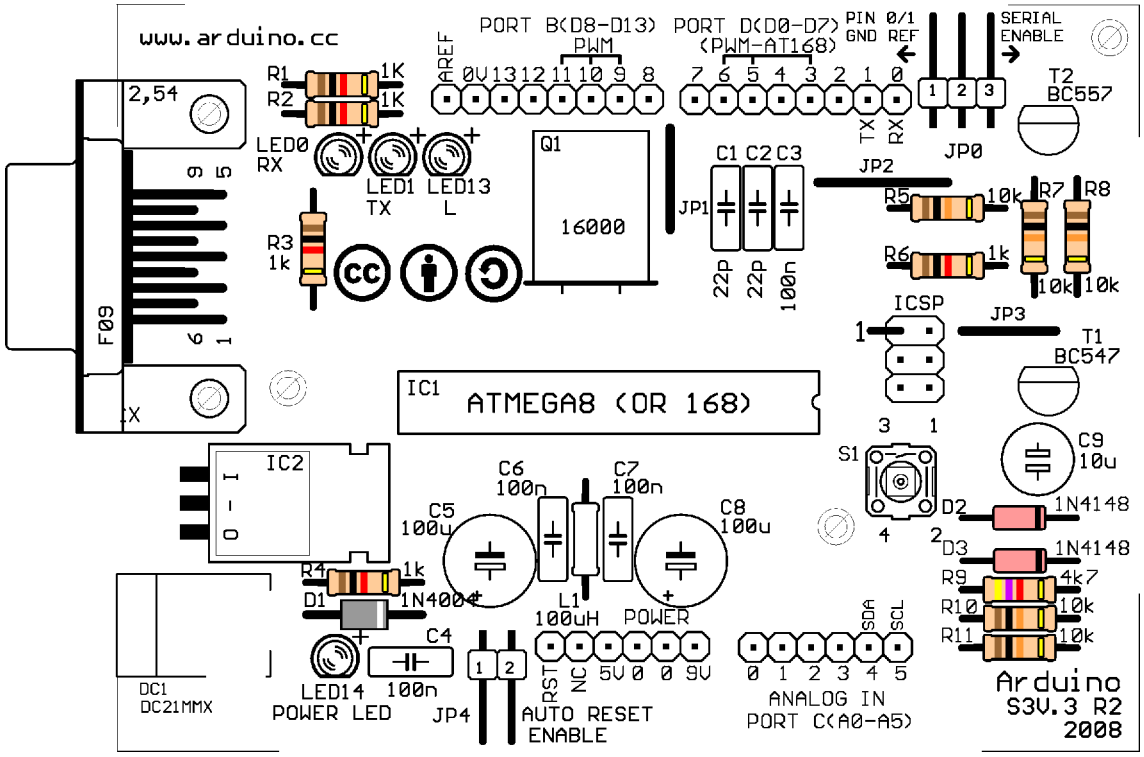

Figura 2- Layout do arduino

2.1.2 Driver (Ponte-H)

Para dar movimento ao robô, é utilizado dois motores de corrente contínua (DC) de 9V. Como o controlador principal é o Arduino, ele não é capaz de fornecer a tensão e corrente para movimentar um motor DC, para o trabalho é utilizado um circuito de pontência, chamado Ponte-H.

Figura 3 - Foto do motor DC 9v

A origem do nome é devido ao chaveamento que pelo Diagrama X é possível, visualmente, observar a letra H. Através do chaveamento em pares, é feito a inversão do sentido do motor DC trocando a passagem de corrente nas bobinas internas do motor.

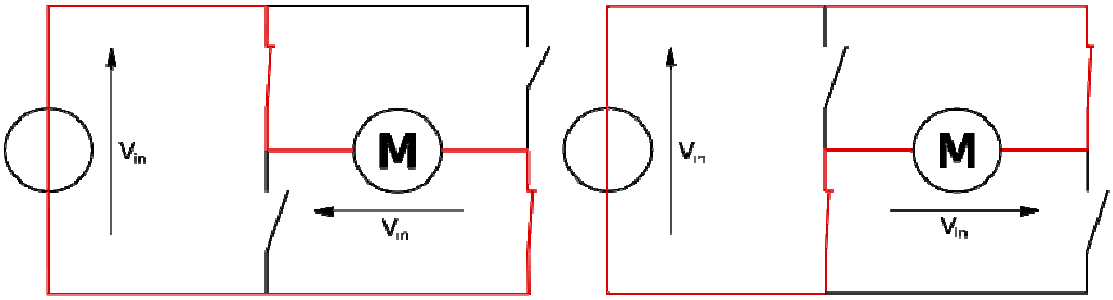

Figura 4 – Ponte-H

Para montar o circuito foi utilizado o circuito integrado L293D com duas pontes e diodos de desacoplagem. O circuito é bastante simples, veja a seuir o Diagrama X.

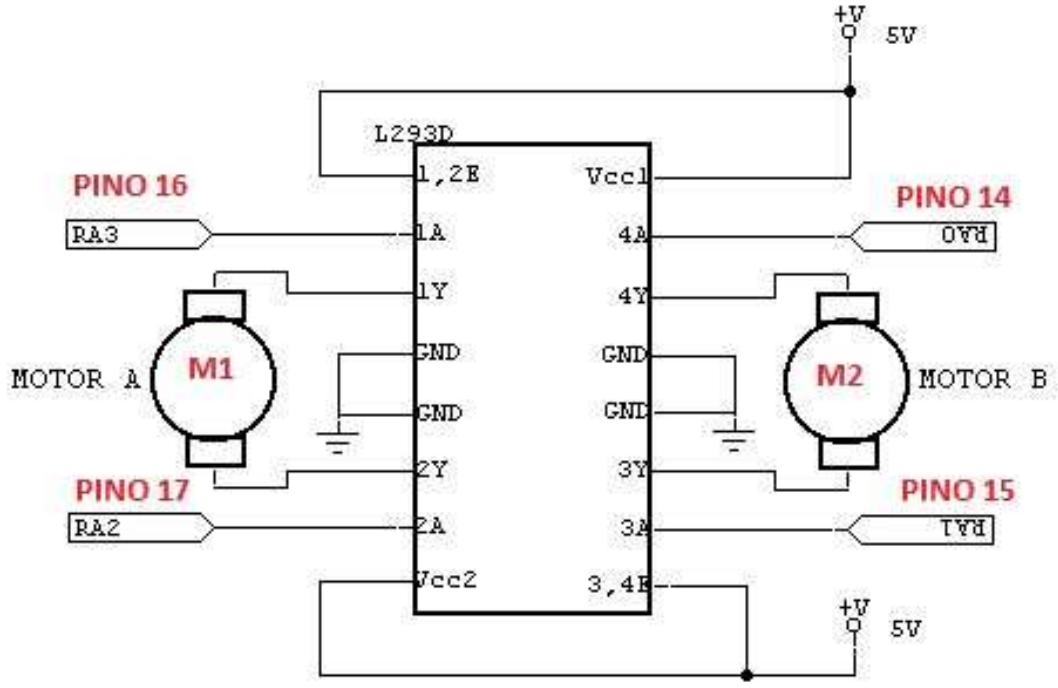

Figura 5 – Diagrama do L293D

Apesar da placa do arduino não apresentar no seu layout os terminais de 14 a 19, eles são os terminais de leitura analógica. Assim, terminal analógico 0 é o terminal 14, ou Pino 14, até o terminal analógico 5, correspondendo ao terminal 19, ou Pino 19. A descrição pode ser vista na figura a seguir retirada do *layout* do Arduino.

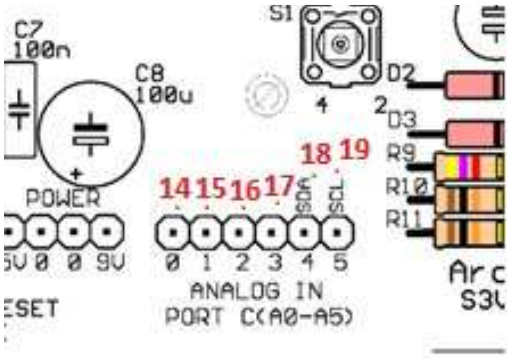

Figura 6 - Terminais analógicos/digitais do arduino

O L239D possui algumas características descritas a seguir:

Supply voltage, V_{CC1} (see Note 1)	36 V
Output supply voltage, V_{CC2}	36 V
Input voltage, V_I	7 V
Output voltage range, V_O	–3 V to V_{CC2} + 3 V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	±2 A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	±1.2 A
Continuous output current, I_O: L293	±1 A
Continuous output current, I_O: L293D	±600 mA
Continuous total dissipation at (or below) 25°C free-air temperature (see Notes 2 and 3)	2075 mW
Continuous total dissipation at 80°C case temperature (see Note 3)	5000 mW
Maximum junction temperature, T_J	150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C
Storage temperature range, T_{stg}	–65°C to 150°C

Figura 7 - Características do driver L293D

O L293D possui um outras aplicações descritas no Diagrama de Blocos a seguir.

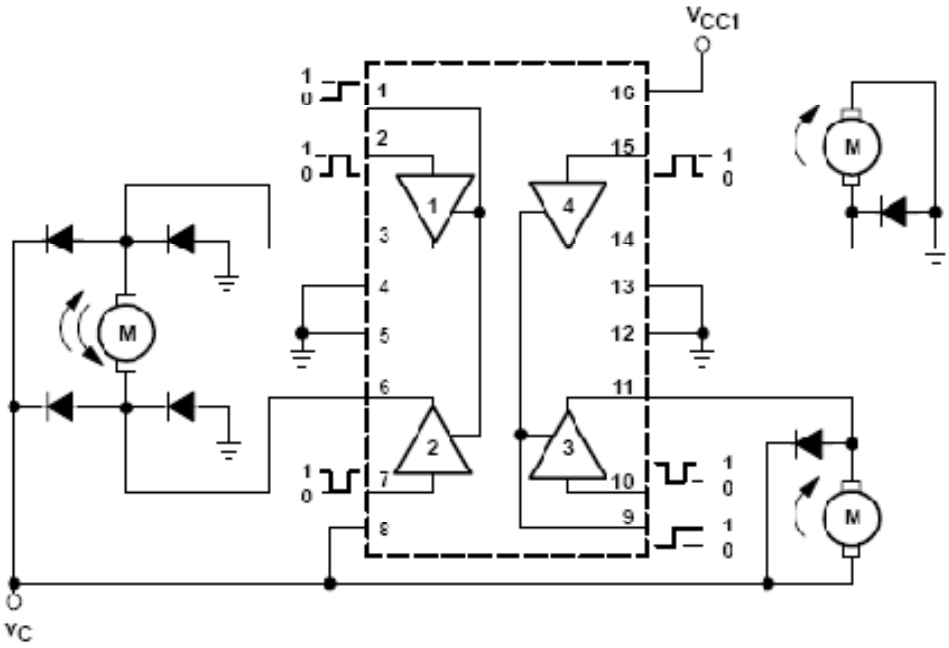

Figura 8 – Usos do driver L293D

2.1.3 Módulo GPS

O módulo GPS é uma das duas partes de sensores que irá dar ao robô a sua orientação para o alcance do destino. Então, é imprescindível que ele seja de fácil interfaceamento com o Arduino e de boa precisão.

Os diversos aparelhos de navegação como: Ecobatímetro, bússola giroscópica, anemômetros, dentre outros – utilizam um protocolo de dados para padronizar as informações. Esse protocolo, ou conjunto de especificações de dados, é o NMEA que carrega as siglas da associação desenvolvedora *National Marine Eletronics Association*.

Os dispositivos que utilizam o NMEA podem entrar duas versões para utilização, NMEA 0183 e NMEA 2000. Dentre as diversas mensagem dentro do pacotes NMEA 0183, a mensagem utilizada no robô é a GGA, ou *Global Positioning System Fixed Data*. A seguir uma figura com os tipo de mensagem.

Option	Description
GGA	Time, position and fix type data.
GLL	Latitude, longitude, UTC time of position fix and status.
GSA	GPS receiver operating mode, satellites used in the position solution, and DOP values.
GSV	The number of GPS satellites in view satellite ID numbers, elevation, azimuth, and SNR values.
MSS	Signal-to-noise ratio, signal strength, frequency, and bit rate from a radio-beacon receiver.
RMC	Time, date, position, course and speed data.
VTG	Course and speed information relative to the ground.
ZDA	PPS timing message (synchronized to PPS).
150	OK to send message.

Figura 9 - Tipos de mensagens NMEA 0183

Um exemplo de valores dentro do GGA pode ser:

$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,0000*18

O formato dos valores é:

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table 1-4
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude	9.0	meters	
Units	M	meters	
Geoid Separation		meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR> <LF>			End of message termination

Figura 10 - Formato GGA

A sigla GP no escopo do mensagem, significa que sua origem é de um sistema de posicionamento dos Estados Unidos da América, ou GPS. Existem outros sistema como o Russo, chamado GLONASS – com a sigla GL; o Europeu, chamado Galileo – sua sigla é desconhecida ainda, pois o sistema só entrará em funcionamento em 2013. O sistema Norte Americano é o mais utilizado, por que o sistema de posicionamento necessita de uma constelação de satélites distribuídos ao redor do globo terrestre e eqüidistantes entre si. Os Americanos foram os primeiros a alcançar uma constelação satisfatória com 24 satélites. E seu uso pode ser dividido pela necessidade de precisão, sendo o uso militar o de maior precisão – chegando a 1 metro de precisão. O outro tipo de uso é o civil que tem como precisão 15 metros a 100 metros.

O módulo de GPS utilizado é o GP-GS009_Ver2.0, mostrado na figura X a seguir.

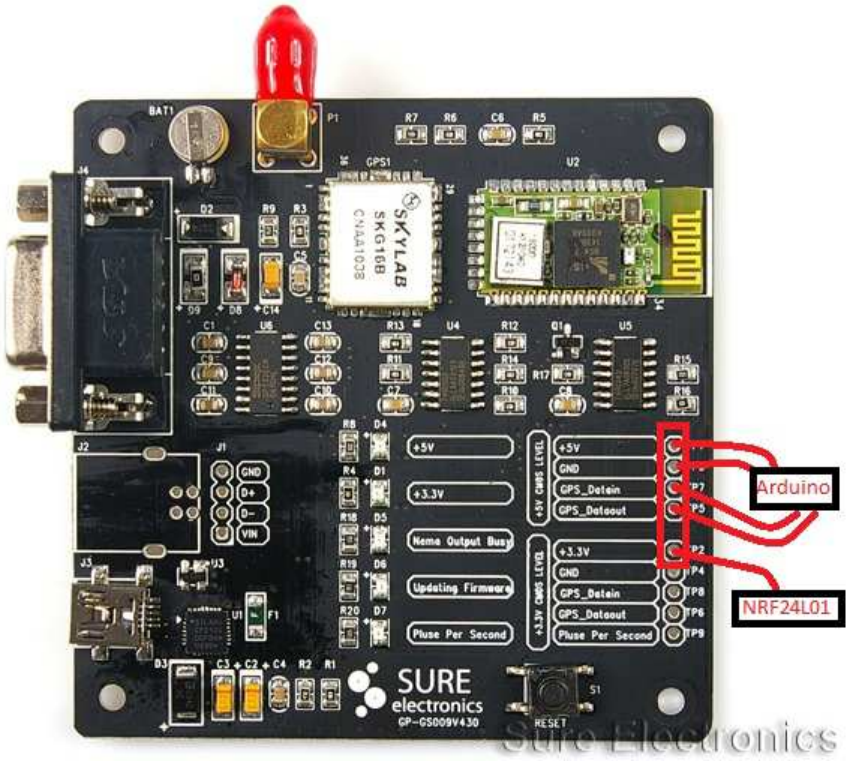

Figura 11 - Módulo GP-GS009

As características do módulo são:

- Baixo consumo de energia
- Receptor de RF de 2.5dB
- Alta sensibilidade -161 dBm (área interna)
- Implementado com padrão NMEA 0183
- Antena com conector SMA
- Alimentação: 5V ou 3V

Possui as seguintes interfaces:

- USB (Porta COM virtual)
- RS232
- Bluetooth 2.0

Todas ligação necessária para o funcionamento do módulo com o arduino é feito pelos terminais como mostrado na figura X.

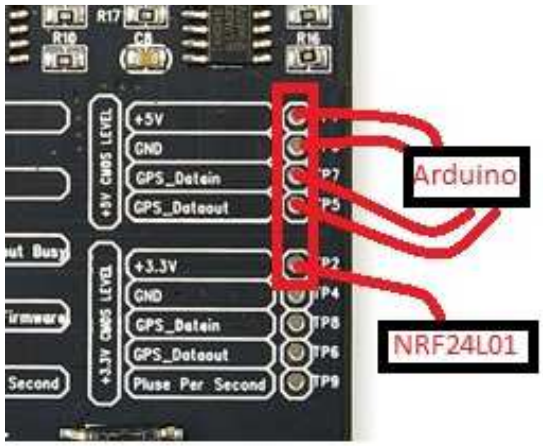

Figura 12 - Ligação módulo GPS

2.1.4 Módulo de Comunicação

Para comunicar o robô sem fios com a base, é utilizado o módulo de rádio freqüência NRF24l01+ da Nordic© com amplificador.

Figura 13 - Módulo NRF24L01+

O uso deste módulo RF é devido sua característica descritas a seguir.

- Banda de Operação 2.4GHz ISM
- Taxa de transferência até 2Mbps
- Baixo consumo de energia em operação
- Tensão de alimentação: 3.6V
- Terminais tolerantes a 5V
- Alcance de transmissão 1000m (área aberta)

O par deste módulo possibilita uma comunicação rápida e de grande capacidade de transmissão dados, assim como a sua transmissão e recepção data através de um único dispositivo. Um módulo ficará no robô e outro na base. O módulo será encarregado de transmitir os dados que o arduino repassar do GPS e da bússola. Sendo

do GPS os valores da mensagem GGA dos campos de latitude, longitude e suas direções, e também repassar o rumo que a bússola digital medir.

A descrição de como o módulo RF é conectado ao arduino é:

- MISO – PINO12 Arduino
- MOSI – PINO11 Arduino
- SCK – PINO13 Arduino
- CE – PINO8 Arduino
- CSN – PINO7 Arduino
- VCC – 3.3V Módulo GPS

2.1.5 Motores DC

É utilizado dois motores para dar movimento ao robô. O sentido da corrente que passa pelos motores dá o tipo de movimento do robô. O sistema empregado é versátil e pode ser utilizado em dois tipos de robô, como um carro, ou um barco, com os movimentos descritos a seguir. Sendo M1 e M2 os motores.

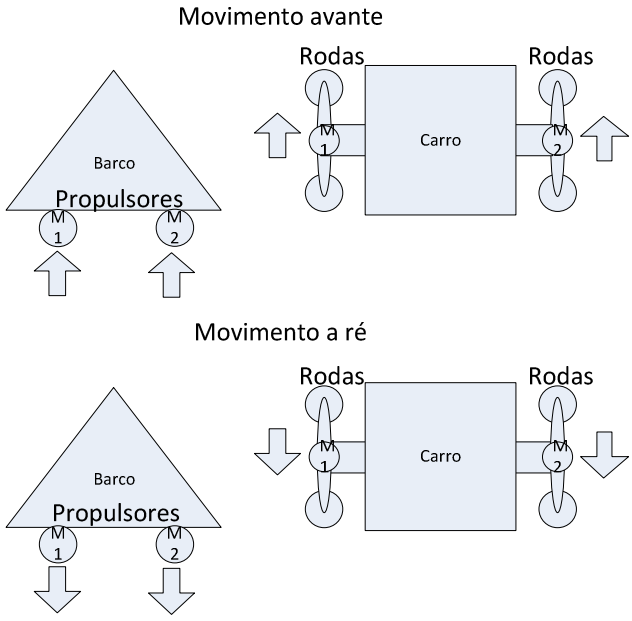

Figura 14 - Movimentos avante e ré

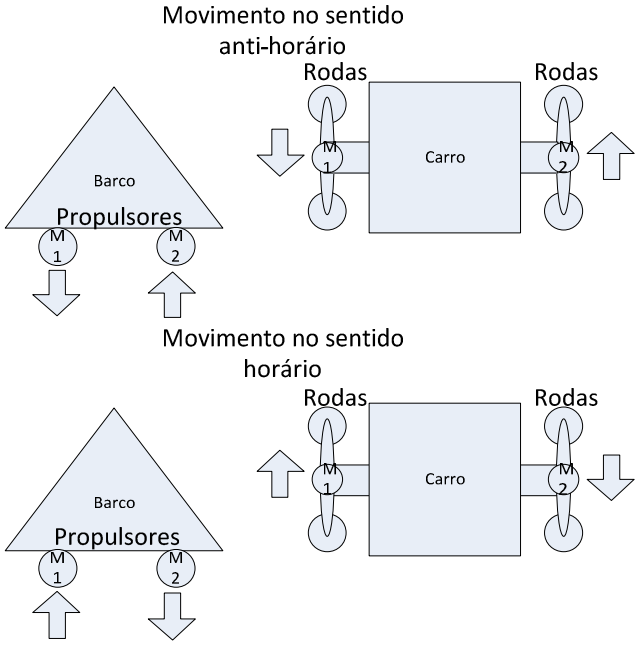

Figura 15 - Movimentos BB e BE

Na Figura 2.1.2, é descrito o Diagrama Esquema para ligação dos motores no Driver Ponte-H

No projeto foi aproveitado o mecanismo de ação de um brinquedo para funcionar através de rodas. Mas é facilmente adaptável a um barco com dois motores DC.

2.1.6 Baterias

As baterias utilizadas são as mesmas utilizadas em algumas células de baterias de notebooks. Elas estão ligadas em série, fornecendo assim 7,4V para o arduino e o driver da ponte-H. O driagrama a seguir mostra como foi interligado.

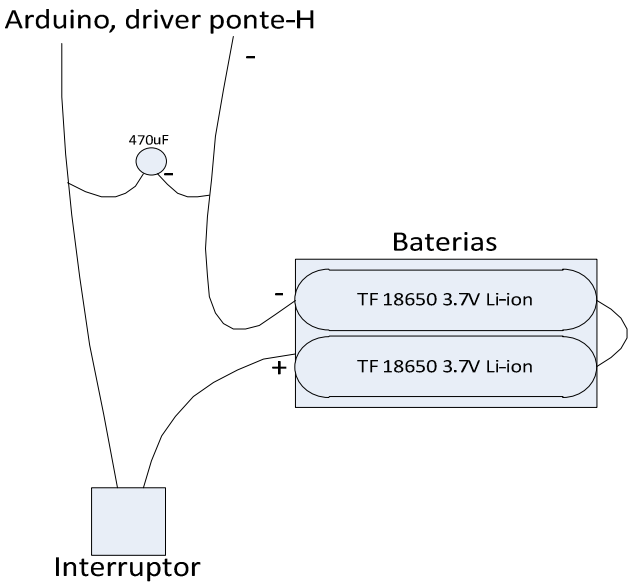

Figura 16 - Alimentação com baterias

É utilizado um capacitor eletrolítico de 470uF para não causar problemas quando o driver ponte-H for acionado. O capacitor tem a função de retificar a queda de tensão que pode ocorrer com o uso dos motores.

2.1.7 Módulo Bússola Digital

Esta é a segunda parte sensorial do robô. A bússola digital promove a experiência do robô saber seus atual rumo para que possa comparar com o rumo de destino. O módulo compatível e baixo custo é o HMC5883L de três eixos magnético. Sua comunicação é conduzida através do barramento bidirecional serial I²C

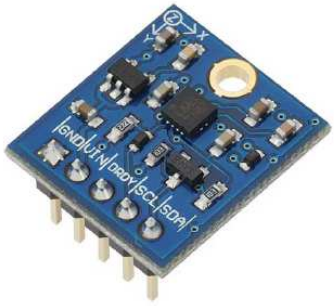

Figura 17 - Módulo bússola digital

O HMC5883L pode ser facilmente conectado ao arduino, descrito a seguir.

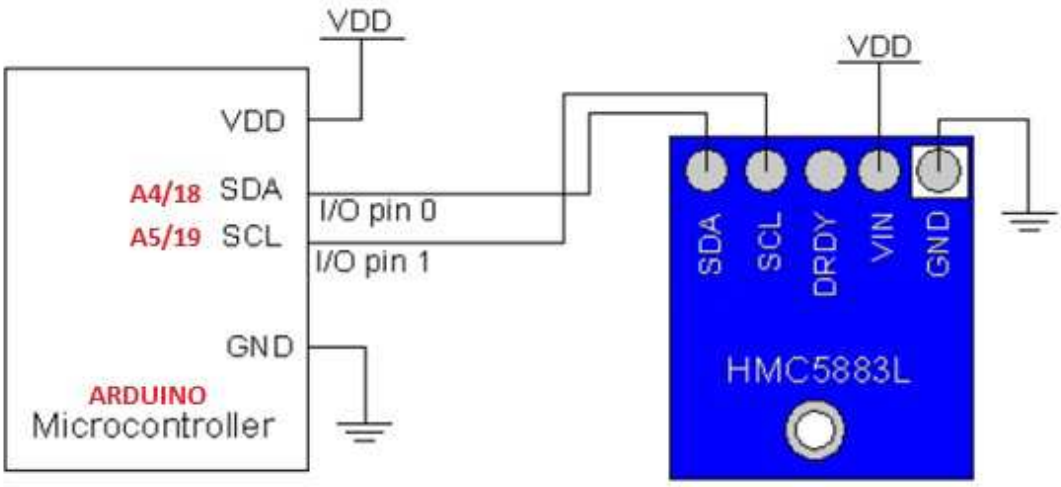

Figura 18 - Interface HMC5883L com arduino

Interface entre HMC5883L e arduino:

- Arduino GND -> HMC5883L GND
- Arduino 5V -> HMC5883L VCC
- Arduino A4 (SDA) -> HMC5883L SDA
- Arduino A5 (SCL) -> HMC5883L SCL

Suas especificações estão descrita a seguir.

Characteristics	Conditions	Min	Typ	Max	Units
Average Current Draw	Idle Mode Measurement Mode	- -	2 100	- -	µA
Field Range	Full Scale (FS) – Total applied field (Typical)	-8		+8	gauss
Mag Dynamic Range	3-bit gain control	±1		±8	gauss
Resolution	VDD=3.0V, GN=2		5		milli-gauss
Linearity	±2.0 gauss input range			0.1	±% FS
Hysteresis	±2.0 gauss input range		±25		Ppm
Cross-Axis Sensitivity	Test Conditions: Cross field =0.5 gauss, Happlied = ±3 gauss		±0.2%		%FS/gauss
Output Rate (ODR)	Continuous Measurement Mode Single Measurement Mode	0.75		75 160	Hz Hz
Measurement Period	From receiving command to data- ready		6		ms
Turn-on Time	Ready for I²C commands		200		µs
Gain Tolerance	All gain/dynamic range settings		±5		%
I²C Address	7-bit address 8-bit read address 8-bit write address		0X1E 0X3D 0X3C		hex hex hex
I²C Rate	Controlled by I²C Master			400	kHz
I²C Hysteresis	Hysteresis of Schmitt trigger inputs on SCL & SDA – Fall (VDDIO=1.8V) Rise (VDDIO=1.8V)		 0.2*VDDIO 0.8*VDDIO		 Volts Volts

Figura 19 - Especificações HMC5883L

2.2 Base

A montagem da base é análogo ao do robô. A base é composta por um Arduino e um módulo RF, NRF24L01+. No entanto, a placa do arduino é uma versão modificada, chamada Iduino.

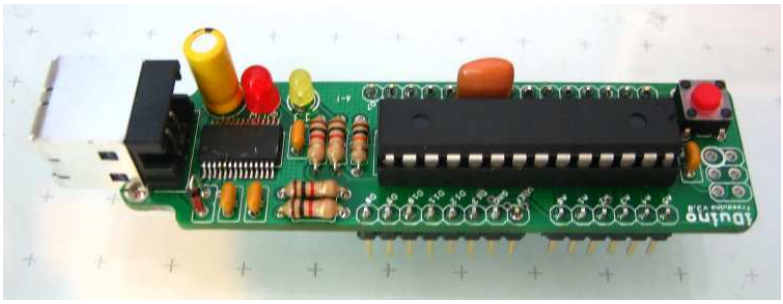

Figura 20 - Iduino

Iduino possui as mesma especificações que o Arduino Diecimila, o mais encontrado para venda. Diferindo apenas que o iduino possui um pino 3.3v e dimensões menores.

A ligação entre os pinos do Iduino e do NRF24L01+ na base é idêntico ao do robô.

- MISO – PINO12 Arduino
- MOSI – PINO11 Arduino
- SCK – PINO13 Arduino
- CE – PINO8 Arduino
- CSN – PINO7 Arduino

2.3 Desenvolvimento do Software

Tanto o robô e a base possuem um programa que foi escrito através do Arduino IDE. No robô deve promove a leitura dos dados do GPS via RS232 9600bps enviar via módulo RF, assim como na leitura da bússola via I²C e enviar via módulo RF. Deverá receber os comandos da base para executar movimentos através do módulo RF. Na base, o programa escrito deve promover a leitura do dado recebido do módulo RF e enviar via RS232 9600bps para o computador, assim como receber os comando do computador e enviar pelo módulo RF. A base trabalha na fronteira do computador e o robô. Como o módulo RF não pode ser conectado a porta serial do computador, é utilizado um arduino.

No computador foi escrito um software com interface gráfica para que se possa conectar a porta serial, ler e escrever na porta. A linguagem escrita é Python e a interface gráfica utiliza o WxPython. O computador é onde é feito os cálculos de distancia e rumo a partir dos pontos fornecido pelo GPS e pelo usuário que entra com o ponto de destino.

O algoritmo recebe o ponto do GPS, recebe ponto de destino e cálculo distância e rumo para destino, caso distância seja superior a 2 metros, depois recebe o rumo atual do robô, fornecido através da bússola digital e subtrai o rumo atual com o rumo de destino, caso seja maior que 10° ou menor -10°, vire para BE ou BB, senão avante com robô.

Para o cálculo de distância ao destino foi utilizada a fórmula de Haversine. O cálculo do rumo, ou rota, foi utilizado o cálculo de círculo máximo.

Para incrementar todas as funções deste projeto, também foram utilizadas bibliotecas de código aberto para facilitar a programação e comunicação com todas as partes do hardware. O código-fonte completo está no final desse trabalho, em anexo.

3 TESTES

3.1 Testes do robô

O teste consiste em compilar e gravar o programa, e depois conseguir enviar os dados do GPS via porta serial do próprio arduino do robô. Apesar de não poder fazer os testes com a bússola digital, engloba o teste do GPS. Os valores dos quatro campos do GGA do NMEA 0183 foram obtidos com sucesso, que são respectivamente: latitude, direção latitudinal, longitude, direção longitudinal.

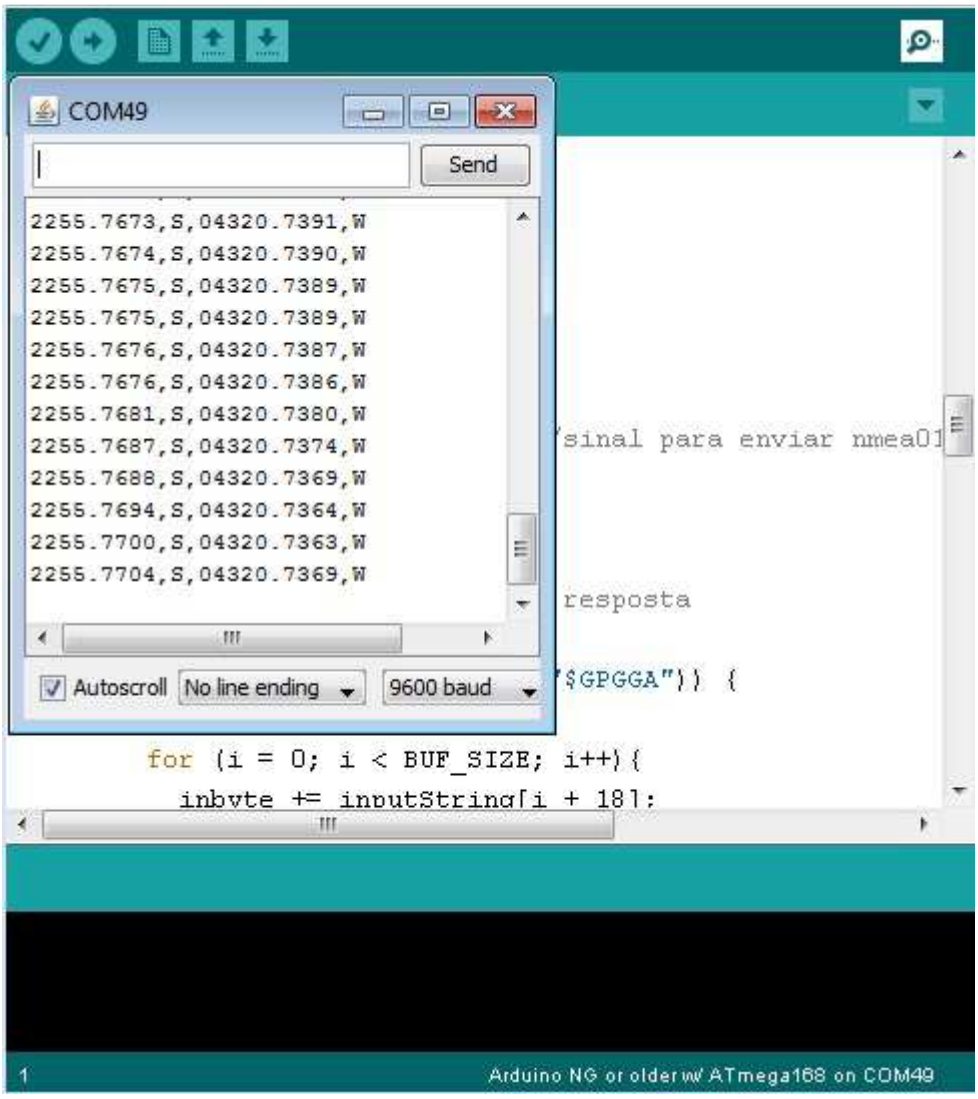

Figura 21 - Teste do robô

3.2 Teste da base

O teste da base consiste em receber os mesmos dados de latitude e longitude enviados do robô para a base. Esse teste engloba o teste dos módulos de RF.

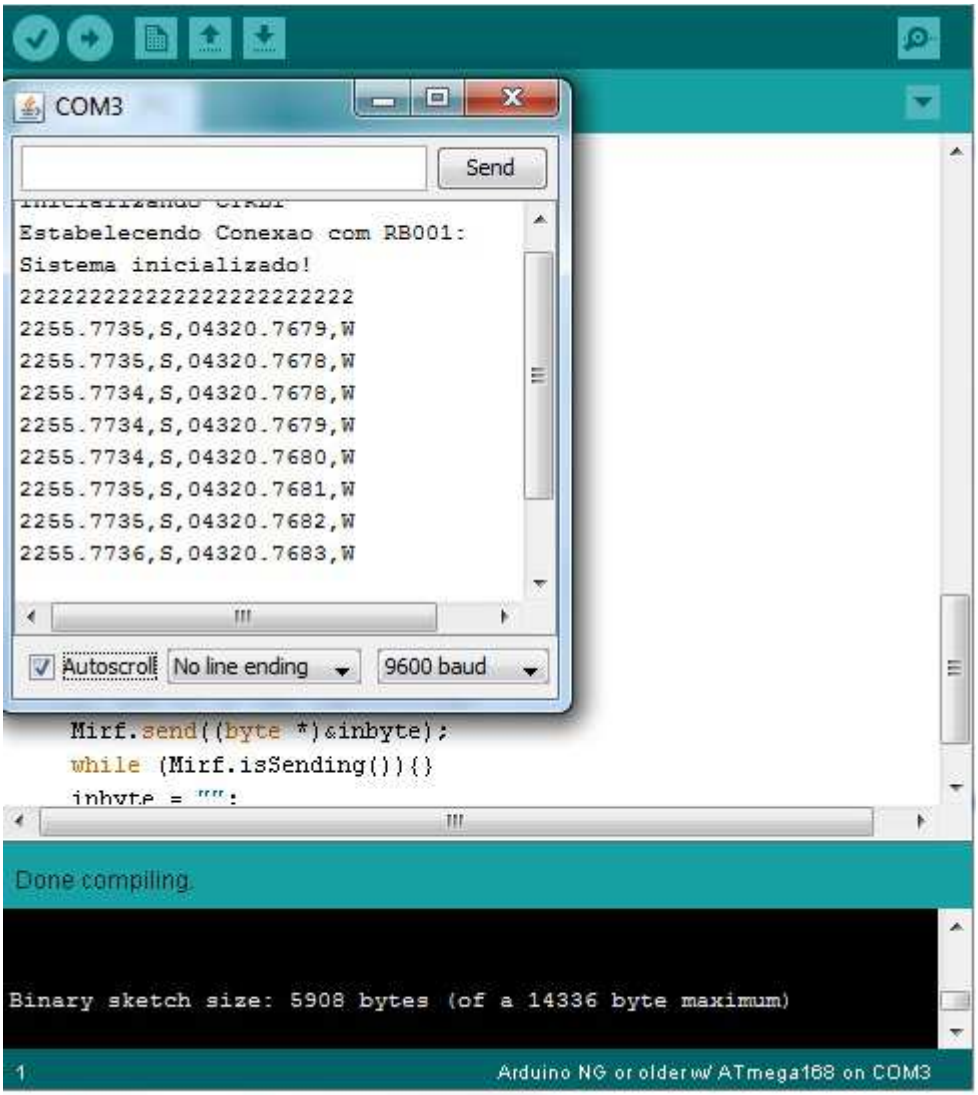

Figura 22 - Teste da base

3.3 Teste da interface gráfica

Esse teste engloba, essencialmente, todo o projeto. É necessário seguir uns passos para uso da interface gráfica. Os passos serão descrito a seguir.

3.3.1 Conexão da interface

Por ordem, o primeiro passo é se conectar a uma porta de comunicação. Bastando pressionar o botão Conectar.

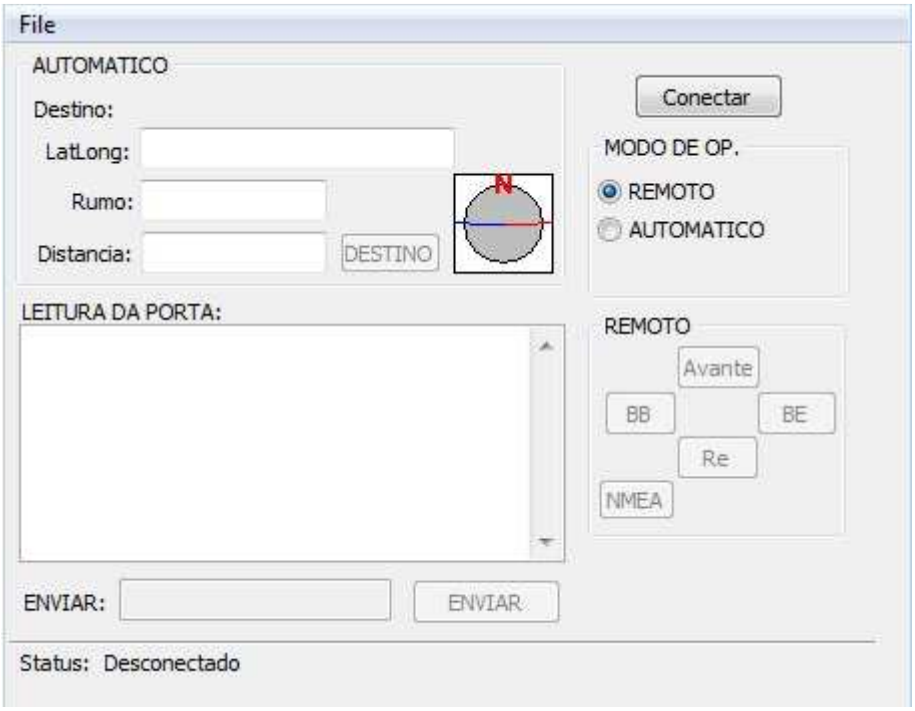

Figura 23 - Primeiro passo da interface gráfica

Em seguida, uma janela de escolha deverá aparecer, bastando para conectar pressionar duas vezes sobre a porta que a base está conectada.

Figura 24 - Segundo passo da interface gráfica

Após abrir a porta e começar a receber os dados do robô, a interface exibirá os dados numa área de texto.

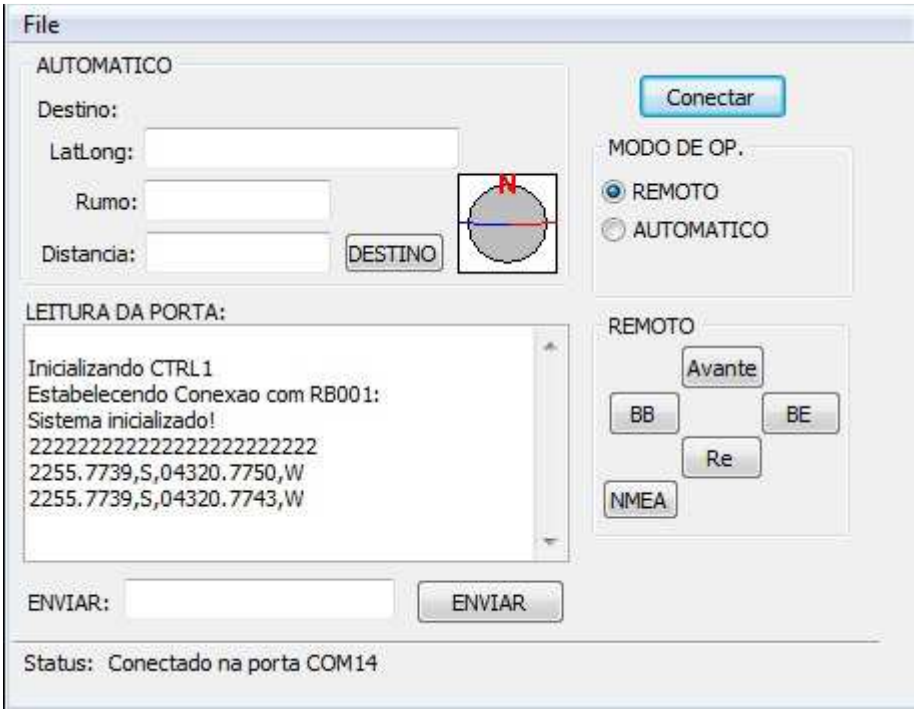

Figura 25 - Terceiro passo da interface gráfica

3.3.2 Enviando comandos

Caso o robô esteja num ponto para onde deverá retornar mais tarde, é necessário que se pressione o botão DESTINO. Isso preencherá o campo de destino, possibilitando o cálculo de rumo e distância até aquele ponto. O rumo e a distância são atualizados a cada novo ponto recebido do GPS. Uma imagem ao lado do botão DESTINO mostra em vermelho o rumo para o destino e o traço azul mostra o rumo atual do robô.

Mesmo que o robô não esteja no ponto para deverá retornar, impossibilitando o uso do botão DESTINO, pode-se inserir diretamente no campo LatLong as coordenadas do destino. Essas coordenadas devem obedecer ao formato (latitude, direção latitudinal, longitude, direção longitudinal) a unidade é a mesma da mensagem GGA - ou seja, graus, minutos e minutos decimais.

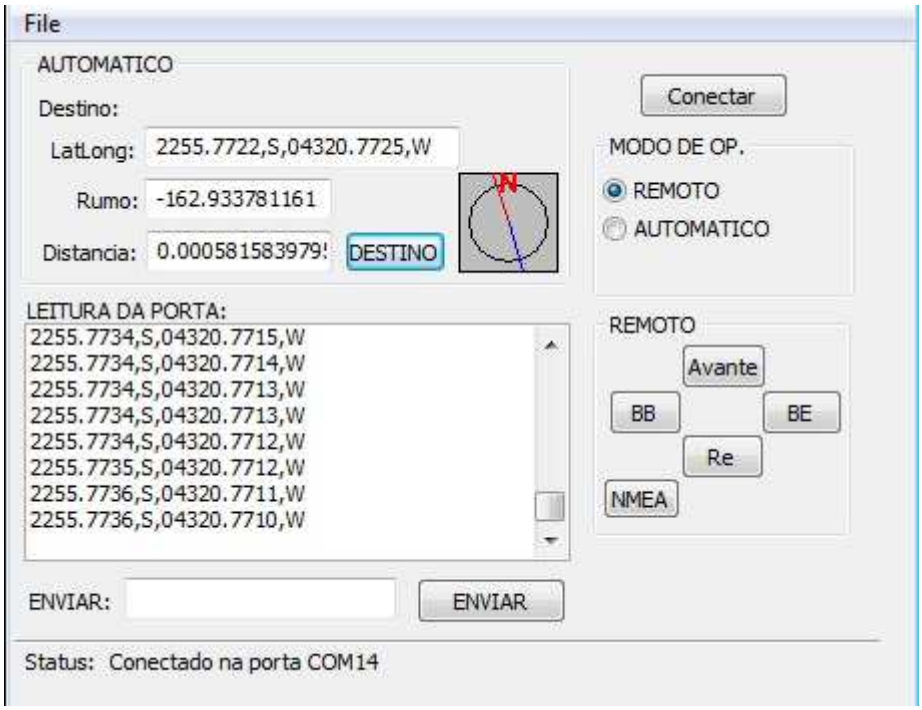

Figura 26 - Quarto passo da interface gráfica

Existe a opção de retorno automático para o ponto de destino, porém devido à ausência da bússola não é possível se obter o rumo atual do robô com a precisão necessária para executar os movimentos que façam convergir para o ponto de destino.

O botão NMEA apenas faz um teste na comunicação entre da base e robô. Os botões Avante, Re, BE, BB servem para mover o robô. Os comandos para mover o robô são caracteres únicos. Sendo, respectivamente, w, s, d, a, os caracteres enviados do computador para o robô.

Existe uma opção de aumentar o tempo de ativação do driver ponte-H. O objetivo seria tornar os movimento mais rápido e maior alcance. Para enviar esse comando deve usar o campo ENVIAR, a letra e aumenta para quinhentos milissegundos o tempo de ação dos motores, a letra q reduz para cem milissegundos o tempo de ação dos motores.

4 CONCLUSÃO

Este documento apresentou essencialmente o projeto para elaboração de um sistema de piloto-automático. Demonstraram-se as etapas que foram desenvolvidas uma a uma com as devidas especificações, circuitos, pinagens e diagrama de blocos, também foram abordados os tipos de testes para seqüência do projeto, buscando sempre a robustez do projeto para obter consistência e qualidade.

O uso da plataforma Arduino deve-se ao fato do projeto necessitar de custo barato, modularidade e de fácil interface com outros componentes. Acelerando o processo de execução do projeto. Apesar de o Arduino apresentar alguns problemas em projetos grandes, neste projeto é compatível e de baixo custo.

Devido a ausência do módulo que possui a bússola, houve alguns imprevistos de como se obter o rumo atual do robô. Pois sem o rumo atual não é possível se determinar automaticamente qual deve ser o movimento que aproxime o robô do destino, ou alvo. Outro imprevisto foi a ocorrência de baixa tensão quando os motores eram acionados, mas um dimensionamento de um capacitor entre os terminais da bateria foi o suficiente.

O baixo consumo por parte dos dispositivos, exceto driver ponte-H, promoveu ao projeto longa autonomia de trabalho para o robô. Aumentando a possibilidade de melhorias no projeto como a implementação de uma câmera serial, um sendo de distância através de ultra-som.

Assim, concebe-se que este projeto atingiu, fundamentalmente, os objetivos propostos e declarados, salvo alguns imprevistos e alterações. Exorto que todo desenvolvimento serviu como um grande aprendizado e experiência, onde o enquadramento do projeto nos requisitos, como problemas funcionais do projeto, prazos, dimensionamentos aumentam a capacidade de lidar com a elaboração de pesquisa e estudo, e o mais importante entender que planejar um projeto desse porte requer muito estudo e conhecimento das tecnologias existentes.

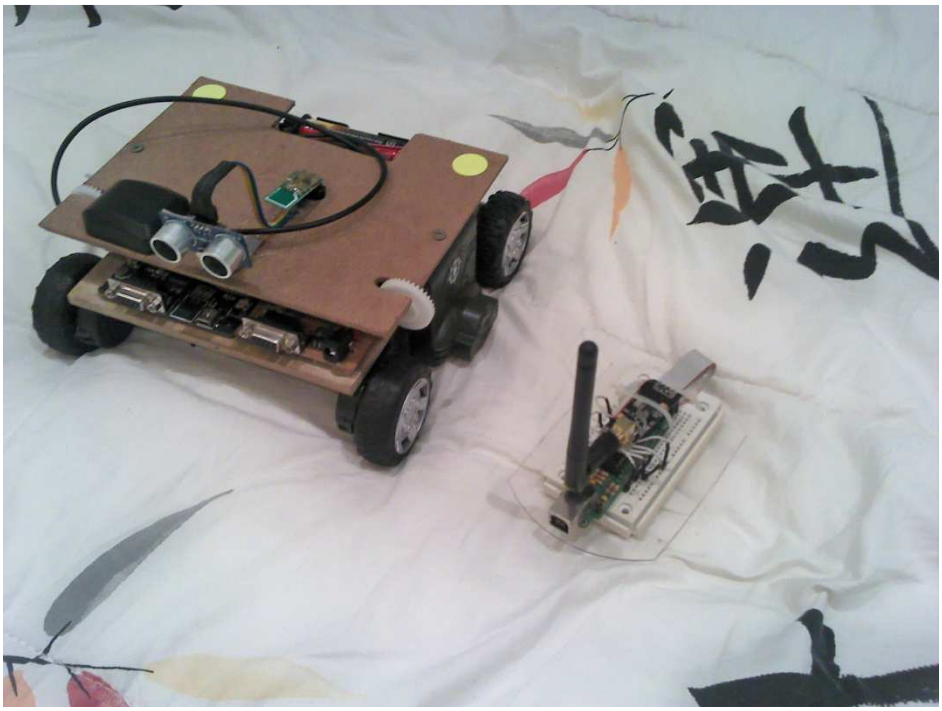

Figura 27 - Artefato Final

5 REFERÊNCIAS

[1] Arduino and nrf24l01 interface. Disponível em:

< https://github.com/aaronds/arduino-nrf24l01 >

Acesso em 5 de Março de 2012.

[2] Arduino Home Page. Disponível em:

< http://arduino.cc/ >

Acesso em 4 de Março de 2012.

[3] Comunicação I²C . Disponível em:

< http://pt.wikipedia.org/wiki/I²C >

Acesso em 17 de Abril de 2012.

[4] DC motors. Disponível em:

< http://en.wikipedia.org/wiki/DC_motor >

Acesso em 9 de Fevereiro de 2012.

[5] Digital Compass. Disponível em:

< http://www.elechouse.com/elechouse/ >

Acesso em 19 de Março de 2012.

[6] Driver H-bridge. Disponível em:

< http://www.electroons.com/electroons/dc_motor_control.html >

Acesso em 9 de Fevereiro de 2012.

[7] Freeduino Project Page. Disponível em:

< http://freeduino.org/ >

Acesso em 29 de Janeiro de 2012.

[8] GPS Accuracy. Disponível em:

< http://www.gps.gov/systems/gps/performance/accuracy/ >

Acesso em 25 de Abril de 2012.

[9] GUI library in Python. Disponível em:

< http://pythoncard.sourceforge.net/ >

Acesso em 20 de Dezembro de 2011.

[10] H bridge. Disponível em:

< http://en.wikipedia.org/wiki/H_bridge >

Acesso em 2012 de Fevereiro de 9.

[11] Iduino Page. Disponível em:

< http://spiffie.org/kits/iduino/ >

Acesso em 9 de Fevereiro de 2012.

[12] Itead Studio - NRF24L01+ with PA and LNA. Disponível em:

< http://iteadstudio.com/ >

Acesso em 4 de Março de 2012.

[13] LatLong Calculator. Disponível em:

< http://www.movable-type.co.uk/scripts/latlong.html >

Acesso em 26 de Abril de 2012.

[14] Library for serial port in Python. Disponível em:

< http://pyserial.sourceforge.net/ >

Acesso em 20 de Dezembro de 2011.

[15] National Marine Eletronics Association . Disponível em:

< http://www.nmea.org/ >

Acesso em 4 de Junho de 2012.

[16] Noções de Latitude e Longitude para GPS. Disponível em:

< http://www.cienciaviva.pt/latlong/anterior/gps.asp >

Acesso em 25 de Abril de 2012.

[17] Python Home Page. Disponível em:

< http://www.python.org/ >

Acesso em 20 de Fevereiro de 2012.

[18] Sparkfun Studio. Disponível em:

< http://www.sparkfun.com/ >

Acesso em 4 de Março de 2012.

[19] Sure Eletrocnics Studio. Disponível em:

< http://www.sureelectronics.net/ >

Acesso em 3 de Março de 2012.

[20] Wx gadged for python. Disponível em:

< http://wxpython.org/ >

Acesso em 21 de Dezembro de 2012.

6 APÊNDICE

6.1Código-fonte Robô

Arquivo Robo.pde

```
/*
  Código-fonte do robô
*/
//Parametros e declarações
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
#define BUF_SIZE 24
#define SPEED 128
int motor1Pin1 = 18;
int motor1Pin2 = 19;
int motor2Pin1 = 16;
int motor2Pin2 = 17;
int entrada = 0;
String inputString = "";
String inbyte = "";
boolean stringComplete = false;
byte data[BUF_SIZE];
//Função de inicialização do robô
void setup(){
  //Ajustes de portas E/S
  pinMode(motor1Pin1, OUTPUT);
  pinMode(motor1Pin2, OUTPUT);
  pinMode(motor2Pin1, OUTPUT);
  pinMode(motor2Pin2, OUTPUT);
  pinMode(speedPin, OUTPUT);
  digitalWrite(speedPin, LOW);
  pinMode(fLight, OUTPUT);
  pinMode(ledSinc, OUTPUT);
  inputString.reserve(200);
  inbyte.reserve(BUF_SIZE);
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"RB001"); //Endereço local (ROBÔ)
  Mirf.setTADDR((byte *)"CTRL1"); //Enrereço destino (BASE)
  Mirf.payload = BUF_SIZE;
  Mirf.config();
  Serial.println("Inicializando Veiculo RB001 ... ");
  Serial.println("Esperando Controle CTRL1 ! ");
  Mirf.send((byte *)"RB001_iniciado!"); //Quando robô iniciar, envia esta mensagem para base
  while(Mirf.isSending()){}
}
```

```
//Função de repetição de rotina padrão
void loop(){
  if(Mirf.dataReady()){ //Escuta base pela RF
    Mirf.getData(data);
    entrada = data[0];
    if (entrada == 'a'){ //Caso o recebido pela RF for 'a'
      esquerda();
      passo();
    } else if (entrada == 'd'){
      direita();
      passo();
    } else if (entrada == 'w'){
      avante();
      passo();
    } else if (entrada == 's'){
      recuar();
      passo();
    } else if (entrada == 'r'){ //Teste de resposta de comunição
       Mirf.send((byte *)"OK");
       while(Mirf.isSending()){}
    } else { parar(); }
  } else { //caso não tenha recebido nada pela RF, enviar LatLong
    if (stringComplete) {
      if (inputString.startsWith("$GPGGA")) { //Parse do NMEA GGA
        int i = 0;
        for (i = 0; i < BUF_SIZE; i++){ //Parse do GGA LatLong
          inbyte += inputString[i + 18];
          data[i] = inbyte[i];
        }
        Mirf.send(data);  //Envia LatLong para CTRL1 (BASE)
        while(Mirf.isSending()){}
        Serial.println(inbyte);
        inbyte = "";
      }
      inputString = "";
      stringComplete = false;
    }
  }
}
//Função de virar a direita
void direita(){
 digitalWrite(motor1Pin1, 1);
 digitalWrite(motor1Pin2, 0);
 digitalWrite(motor2Pin1, 0);
 digitalWrite(motor2Pin2, 1);
}
//Função de virar a esquerda
void esquerda(){
 digitalWrite(motor1Pin1, 0);
 digitalWrite(motor1Pin2, 1);
```

```
 digitalWrite(motor2Pin1, 1);
 digitalWrite(motor2Pin2, 0);
}
//Função de avançar
void avante(){
 digitalWrite(motor1Pin1, 0);
 digitalWrite(motor1Pin2, 1);
 digitalWrite(motor2Pin1, 0);
 digitalWrite(motor2Pin2, 1);
}
//Função de ré
void recuar(){
 digitalWrite(motor1Pin1, 1);
 digitalWrite(motor1Pin2, 0);
 digitalWrite(motor2Pin1, 1);
 digitalWrite(motor2Pin2, 0);
}
//Função para liberar os motores
void parar(){
 digitalWrite(motor1Pin1, 0);
 digitalWrite(motor1Pin2, 0);
 digitalWrite(motor2Pin1, 0);
 digitalWrite(motor2Pin2, 0);
}
//Função para travar os motores, simulando um freio
void freiar(){
 digitalWrite(motor1Pin1, 1);
 digitalWrite(motor1Pin2, 1);
 digitalWrite(motor2Pin1, 1);
 digitalWrite(motor2Pin2, 1);
}
/*
  Rotina padrão após o acionamento
  dos motores
*/
void passo(){
 analogWrite(speedPin, SPEED);
 delay(100);
 parar();
}
/*
  Evento gerado quando um dado é recebido
  pela porta serial RX
*/
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      stringComplete = true;
    }
```

```
  }
}
```

6.2 Código-fonte Base

Arquivo Base.pde

```
/*
  Código-fonte base
*/
//Parametros e declarações
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
#define BUF_SIZE 24
int i = 0;
String inbyte = "";
byte data[BUF_SIZE];
//Função de inicialização da base
void setup(){
  inbyte.reserve(BUF_SIZE);
  Serial.begin(9600);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte *)"CTRL1"); //Endereço local (BASE)
  Mirf.setTADDR((byte *)"RB001"); //Endereço destino (ROBÔ)
  Mirf.payload = BUF_SIZE;
  Mirf.config();
  Serial.println("Inicializando CTRL1");
  Serial.println("Estabelecendo Conexao com RB001:");
  Serial.println("Sistema inicializado!");
}
//Função repetitiva de rotina padrão
void loop(){
  if (Mirf.dataReady()) { //Escuta o robô via RF
    Mirf.getData(data);
    for (i = 0; i < BUF_SIZE; i++){
      inbyte += char(data[i]);
    }
    Serial.println(inbyte); //Envia para computador via serial TX
    inbyte = "";
  }
}
/*
  Função de interrupção da rotina padrão
  para receber da porta serial RX (Escuta o computador)
  Envia para robô via RF
*/
void serialEvent() {
```

```
  while (Serial.available()) {
    char inChar = char(Serial.read());
    Mirf.send((byte *)&inChar);
    while (Mirf.isSending()){}
    inbyte = "";
  }
}
```

6.3 Código-fonte Interface Gráfica

Arquivo InterfaceGraficaRobo.py

```
from PythonCard import model, dialog
import os, sys, time
import wx
import serial
import math
from threading import Thread
global gearth
ser = serial.Serial()
ser.baudrate = 9600

class InterfaceGraficaRobo(model.Background):

    bParse = False
    bEnvia = False
    bDestino = False
    bAuto = False
    num = 0
    pi = math.pi
    dEarth = 6371
    latitude = 0
    longitude = 0
    teta = -90
    omega = 90
    orientacaoAtual = 0, 0
    orientacaoAnterior = 0, 0
    orientacaoDestino = 0, 0
    pontoAtual = "0" #lat,dir lat,long,dir long
    pontoAnterior = "0"
    pontoDestino = "0"
    dir = 'r'

    def calc_rumo_distancia(self, PtD1, PtD2):
     #transforma pontos string em graus e decimais de graus em um float

        lato = math.radians(float(PtD1[:2]) + float(PtD1[2:9])/ 60)
        longo = math.radians(float(PtD1[12:15]) + float(PtD1[15:22])/ 60)

        latd = math.radians(float(PtD2[:2]) + float(PtD2[2:9])/ 60)
```

```
        longd = math.radians(float(PtD2[12:15]) + float(PtD2[15:22])/ 60)

        if PtD1[10:11] == "S": #atribui o -1 caso seja sul ou oeste
            lato = -1*lato
        if PtD1[23:24] == "W":
            longo = -1*longo
        if PtD2[10:11] == "S":
            latd = -1*latd
        if PtD2[23:24] == "W":
            longd = -1*longd

        dLat =  lato - latd
        dLong = longo - longd

        #calculo de distancia com a formula de haversine
        a =
math.sin(dLat/2)*math.sin(dLat/2)+math.sin(dLong/2)*math.sin(dLong/2)*math.cos(lato)*mat
h.cos(latd)
        c = 2*math.atan2(math.sqrt(a),math.sqrt(1-a))
        dist = self.dEarth*c

        #rumo com circulo perfeito
        y = math.sin(dLong)*math.cos(latd)
        x = math.cos(lato)*math.sin(latd)-math.sin(lato)*math.cos(latd)*math.cos(dLong)
        rumo = math.degrees(math.atan2(y, x))
        return rumo, dist

    def on_initialize(self, event):
        self.bitmapCanvasTest(self.teta, self.omega)

    def bitmapCanvasTest(self, teta, omega):
        canvas = self.components.BitmapCanvasD
        canvas.backgroundColor = 'white'
        canvas.foregroundColor = 'black'
        canvas.drawRectangle((0, 0), (50, 50))
        #canvas.foregroundColor = 'gray'
        canvas.fillColor = 'gray'
        canvas.drawCircle((25,25), 20)
        canvas.foregroundColor = 'red'
        canvas.drawText('N', (20, -3))
        canvas.foregroundColor = 'red'
        canvas.drawLine((25, 25), (-15*math.degrees(math.sin(math.radians(teta))), -
15*math.degrees(math.cos(math.radians(teta))))) #raio 15
        canvas.foregroundColor = 'blue'
        canvas.drawLine((25, 25), (-15*math.degrees(math.sin(math.radians(omega))), -
15*math.degrees(math.cos(math.radians(omega))))) #raio 15

    def on_ButtonConectar_mouseClick(self, event):
        if self.components.txtStatus.text == 'Desconectado':
            result = dialog.singleChoiceDialog(self, "Selecionar porta:", "Conectar", ['COM2',
'COM3', 'COM4', 'COM5', 'COM6', 'COM7', 'COM8', 'COM9', 'COM10', 'COM11', 'COM12',
```

```
'COM13', 'COM14', 'COM15', 'COM16', 'COM17', 'COM18', 'COM19', 'COM20', 'COM21',
'COM22', 'COM23', 'COM24', 'COM25'])
        ser.port = result.selection
        ser.timeout = 1
        if ser.portstr == result.selection:
            self.components.txtStatus.text = 'Conectado na porta ' + result.selection
            self.components.TextFieldEnviar.enabled = True
            self.components.ButtonEnviar.enabled = True
            self.components.ButtonAvante.enabled = True
            self.components.ButtonRe.enabled = True
            self.components.ButtonBB.enabled = True
            self.components.ButtonBE.enabled = True
            self.components.ButtonNMEA.enabled = True
            self.components.ButtonDestino.enabled = True

            ser.open()
            self.trataThread()
        else:
            ser.close()
    else:
        ser.close()
        self.components.txtStatus.text = 'Desconectado'
        self.components.TextFieldEnviar.enabled = False
        self.components.ButtonEnviar.enabled = False
        self.components.ButtonAvante.enabled = False
        self.components.ButtonRe.enabled = False
        self.components.ButtonBB.enabled = False
        self.components.ButtonBE.enabled = False
        self.components.ButtonNMEA.enabled = False
        self.components.ButtonDestino.enabled = False

  def on_ButtonEnviar_mouseClick(self, event):
    msg = self.components.TextFieldEnviar.text
    ser.write(msg)
    self.components.TextFieldEnviar.text = ""

  def on_ButtonDestino_mouseClick(self, event):
    self.components.TextFieldLatD.text = self.pontoAtual
    self.pontoDestino = self.pontoAtual

  def on_RadioGroupOP_select(self, event):
    if self.components.RadioGroupOP.stringSelection == 'AUTOMATICO':
        self.bAuto = True
    if self.components.RadioGroupOP.stringSelection == 'REMOTO':
        self.bAuto = False

  def on_ButtonAvante_mouseClick(self, event):
    ser.write('w')

  def on_ButtonRe_mouseClick(self, event):
    ser.write('s')
```

```
  def on_ButtonBB_mouseClick(self, event):
    ser.write('a')

  def on_ButtonBE_mouseClick(self, event):
    ser.write('d')

  def on_ButtonNMEA_mouseClick(self, event):
    ser.write('r')

  def trataThread(self):
    t = Thread(target = self.recebendo)
    t.start()

  def paraThread(self):
    self._stop()

  def recebendo(self):
    while True:
      if ser.isOpen() and ser.inWaiting():
        msg = ser.readline()
        self.components.TextAreaRecebe.appendText(msg)
        if msg.rfind(",") == 22:
          #print msg[:9]
          #print msg[10:11]
          #print msg[12:22]
          #print msg[23:24]
          self.pontoAtual = msg
          if (self.pontoAtual[23:24] == "W"):
            #Se bDestino for verdadeiro ira acionar os motores para mover
            if self.components.TextFieldLatD.text[23:24]=="W" and self.pontoAnterior[0]!=0:
              self.orientacaoDestino = self.calc_rumo_distancia(self.pontoAnterior,
self.components.TextFieldLatD.text)
              self.orientacaoAtual = self.calc_rumo_distancia(self.pontoAnterior,
self.pontoAtual)
              self.components.TextFieldDistanciaD.text = str(self.orientacaoDestino[1])
              self.components.TextFieldRumoD.text = str(self.orientacaoDestino[0])

              delta = self.orientacaoAnterior[1] - self.orientacaoAtual[1]
              if delta > 0.0005:
                self.orientacaoAnterior = self.orientacaoAtual
                self.pontoAnterior = self.pontoAtual

              if self.orientacaoAnterior[0] != 0:
                #DISPLAY DE ORIENTACAO DESTINO
                self.bitmapCanvasTest(self.orientacaoAnterior[0], self.orientacaoDestino[0])
                if self.bAuto == True:
                  print self.bAuto
                  if self.orientacaoDestino[1] > 0.0008:
                    deltaS = self.orientacaoDestino[1] - self.orientacaoAnterior[1]
                    deltaA = self.orientacaoDestino[0] - self.orientacaoAnterior[0]
```

```
if deltaA > 180:
   deltaA = deltaA - 360
if deltaA < -180:
   deltaA = deltaA + 360

angulofim = 0.5
#Afastando
if deltaS > 0.001:
   ser.write('q')
   #se rumoFinal - RumoAnterior > 0
   if deltaA > angulofim:
      print '2xBB'
      ser.write('d')
      time.sleep(0.2)
      ser.write('d')
      time.sleep(0.2)
      ser.write('e')
      ser.write('s')
      time.sleep(0.2)
   if deltaA < -angulofim:
      print '2xBE'
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('e')
      ser.write('s')
      time.sleep(0.2)

   if deltaA < angulofim and deltaA > -angulofim:
      print 'AVANTE'
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)
      ser.write('a')
      time.sleep(0.2)

#Aproximando
if deltaS < 0.001:
   ser.write('q')
```

```
                        #se rumoFinal - RumoAnterior > 0
                        if deltaA > angulofim:
                            print 'BB'
                            ser.write('d')
                            time.sleep(0.2)
                            ser.write('d')
                            time.sleep(0.2)
                            ser.write('e')
                            ser.write('s')
                            time.sleep(0.2)

                        if deltaA < -angulofim:
                            print 'BE'
                            ser.write('a')
                            time.sleep(0.2)
                            ser.write('a')
                            time.sleep(0.2)
                            ser.write('e')
                            ser.write('s')
                            time.sleep(0.2)

                        if deltaA < angulofim and deltaA > -angulofim:
                            print 'AVANTE'
                            ser.write('s')
                            time.sleep(0.2)
                            ser.write('s')
                            time.sleep(0.2)

                    else:
                        print 'AVANTE Chegou'

                    time.sleep(1)
                else:
                    self.orientacaoAnterior = self.orientacaoAtual
            else:
                self.pontoAnterior = self.pontoAtual
if __name__ == '__main__':
    app = model.Application(InterfaceGraficaRobo)
    app.MainLoop()
```

Arquivo InterfaceGraficaRobo.rsrc.py

```
{'application':{'type':'Application',
        'name':'InterfaceGraficaRobo',
    'backgrounds': [
    {'type':'Background',
        'name':'bgTemplate',
        'title':u'Standard Template with File->Exit menu',
        'size':(458, 378),
```

```
        'style':['wx.SYSTEM_MENU', 'wx.MINIMIZE_BOX', 'wx.STAY_ON_TOP', 'wx.CLOSE_BOX',
'wx.CAPTION'],

       'menubar': {'type':'MenuBar',
        'menus': [
           {'type':'Menu',
           'name':'menuFile',
           'label':'&File',
           'items': [
               {'type':'MenuItem',
                'name':'menuFileExit',
                'label':'E&xit',
                'command':'exit',
               },
            ]
           },
        ]
    },
       'components': [

{'type':'BitmapCanvas',
   'name':'BitmapCanvasD',
   'position':(224, 64),
   'size':(50, 50),
   'backgroundColor':(255, 255, 255, 255),
   },
{'type':'TextField',
   'name':'TextFieldDistanciaD',
   'position':(68, 93),
   'size':(93, -1),
   'editable':False,
   },
{'type':'StaticText',
   'name':'StaticTextDistanciaD',
   'position':(17, 97),
   'text':'Distancia:',
   },
{'type':'Button',
   'name':'ButtonDestino',
   'position':(167, 93),
   'size':(51, 21),
   'enabled':False,
   'label':'DESTINO',
   },
{'type':'Button',
   'name':'ButtonNMEA',
   'position':(296, 216),
   'size':(39, 21),
   'enabled':False,
   'label':'NMEA',
   },
```

```
{'type':'Button',
   'name':'ButtonEnviar',
   'position':(203, 266),
   'enabled':False,
   'label':'ENVIAR',
   },
{'type':'TextField',
   'name':'TextFieldEnviar',
   'position':(57, 266),
   'size':(136, -1),
   'enabled':False,
   },
{'type':'StaticText',
   'name':'StaticText411',
   'position':(10, 271),
   'text':'ENVIAR:',
   },
{'type':'StaticText',
   'name':'StaticText41',
   'position':(9, 126),
   'text':'LEITURA DA PORTA:',
   },
{'type':'TextField',
   'name':'TextFieldRumoD',
   'position':(67, 67),
   'size':(94, -1),
   'editable':False,
   },
{'type':'TextField',
   'name':'TextFieldLatD',
   'position':(67, 41),
   'size':(158, -1),
   },
{'type':'StaticText',
   'name':'StaticText3311',
   'position':(34, 71),
   'text':'Rumo:',
   },
{'type':'StaticText',
   'name':'StaticText33',
   'position':(21, 46),
   'text':'LatLong:',
   },

{'type':'StaticText',
   'name':'StaticText321',
   'position':(15, 25),
   'text':'Destino:',
   },
{'type':'TextArea',
   'name':'TextAreaRecebe',
```

```
    'position':(7, 139),
    'size':(273, 119),
    'text':'\n',
    },
 {'type':'StaticText',
    'name':'txtStatus',
    'position':(50, 301),
    'size':(384, -1),
    'text':'Desconectado',
    },
 {'type':'StaticText',
    'name':'StaticText1',
    'position':(8, 301),
    'size':(34, -1),
    'text':'Status:',
    },
 {'type':'StaticLine',
    'name':'StaticLine1',
    'position':(2, 297),
    'size':(434, -1),
    'layout':'horizontal',
    },
 {'type':'Button',
    'name':'ButtonConectar',
    'position':(314, 14),
    'label':'Conectar',
    },
 {'type':'Button',
    'name':'ButtonAvante',
    'position':(335, 149),
    'size':(43, -1),
    'enabled':False,
    'label':'Avante',
    },
 {'type':'Button',
    'name':'ButtonRe',
    'position':(335, 194),
    'size':(42, -1),
    'enabled':False,
    'label':'Re',
    },
 {'type':'Button',
    'name':'ButtonBB',
    'position':(299, 172),
    'size':(37, -1),
    'enabled':False,
    'label':'BB',
    },
 {'type':'Button',
    'name':'ButtonBE',
    'position':(375, 172),
```

```
    'size':(41, -1),
    'enabled':False,
    'label':'BE',
    },
{'type':'StaticBox',
    'name':'StaticBox1',
    'position':(291, 133),
    'size':(129, 113),
    'label':'REMOTO',
    },
{'type':'RadioGroup',
    'name':'RadioGroupOP',
    'position':(291, 44),
    'size':(131, 83),
    'items':['REMOTO', 'AUTOMATICO'],
    'label':'MODO DE OP.',
    'layout':'vertical',
    'max':1,
    'stringSelection':'REMOTO',
    },
{'type':'StaticBox',
    'name':'StaticBox2',
    'position':(5, 3),
    'size':(275, 118),
    'label':'AUTOMATICO',
    },
] # end components
} # end background
] # end backgrounds
} }
```