

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIS GUILHERME MACHADO CAMARGO
PEDRO ALBERTO DE BORBA
RICARDO FARAH
STEFAN CAMPANA FUCHS
TELMO FRIESEN

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

MONOGRAFIA

CURITIBA

2013

**LUIS GUILHERME MACHADO CAMARGO
PEDRO ALBERTO DE BORBA
RICARDO FARAH
STEFAN CAMPANA FUCHS
TELMO FRIESEN**

MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR

Monografia apresentado à Unidade Curricular de Oficina de Integração 3 do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para aprovação.

CURITIBA

2013

RESUMO

CAMARGO, L.G.M; BORBA, P.A.; FARAH, R.; FUCHS, S.C.; FRIESEN, T. MAPEAMENTO DE AMBIENTES COM O ROBÔ BELLATOR. 131 f. Monografia – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

O objetivo desse projeto foi o desenvolvimento de um sistema de mapeamento bidimensional de ambientes composto por um robô de sensoriamento, um protocolo de comunicação e uma estação-base. O robô empregado é denominado Bellator e já existia anteriormente à realização desse projeto, sendo constituído de rodas, motores, textitdrivers dos motores, *encoders* ópticos, sensores infra-vermelho de distância e placa de linux embarcado. Ao hardware já existente foram acrescentadas uma *webcam* e uma nova placa de controle embarcada desenvolvida pela equipe e que possui um acelerômetro e um giroscópio, com os quais se pretendeu corrigir eventuais erros de posicionamento do robô. A interface gráfica criada para a estação base permite ao usuário movimentar o robô e visualizar o mapa gerado bem como as imagens captadas pela *webcam*.

Ao final, verificou-se que o mapeamento de ambientes foi possível, embora em alguns casos sejam observadas distorções. Isto se deu devido à imprecisão dos *encoders* ópticos e ao escorregamento das correias e rodas do robô. A correção desses problemas por meio do uso do acelerômetro mostrou-se inalcançável no escopo desse projeto, enquanto a utilização do giroscópio mostrou-se satisfatória.

Palavras-chave: mapeamento de ambientes, robô, sensores infra-vermelho

ABSTRACT

CAMARGO, L.G.M; BORBA, P.A.; FARAH, R.; FUCHS, S.C.; FRIESEN, T. . 131 f. Monografia – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

The aim of this project was the development of a two-dimensional environment mapping system composed of a sensing robot, a communication protocol and a base station. The robot used is denominated Bellator and it already existed prior to this project consisting of wheels, motors, motor drivers, optical encoders, infrared distance sensors and embedded linux board. To the existing hardware a webcam, and a new embedded control board designed by this team and having an accelerometer and a gyroscope were added, with the purpose of correcting eventual robot positioning errors. The graphical interface created for the base station allows the user to move the robot, and view the generated map as well as the images captured by the webcam.

In the end, it was noted that environment mapping was achieved, although some distortion was observed in some cases. This was due to the lack of precision of the optical encoders and also to the slipping of the wheels and belt of the robot. A fix to these problems by means of employing the accelerometer was unachievable in the scope of this project, while the use of the gyroscope was satisfactory.

Keywords: environment mapping, robot, infrared sensors

LISTA DE FIGURAS

FIGURA 1	– Webcam Genius iSlim 1300.	25
FIGURA 2	– Disposição dos itens no robô	26
FIGURA 3	– Placa de desenvolvimento contendo o chip MPU-6050	29
FIGURA 4	– Forma de onda na saída do encoder	30
FIGURA 5	– Curva de resposta do sensor Infra Vermelho IR 2Y0A02F98	31
FIGURA 6	– Diagrama de casos de uso do <i>software</i> da estação base.	49
FIGURA 7	– Diagrama de casos de uso do <i>software</i> para a placa TS-7260.	49
FIGURA 8	– Diagrama de casos de uso do <i>software</i> para a placa de baixo nível.	50
FIGURA 9	– Diagrama de classes da estação base	51
FIGURA 10	– Diagrama de classes do sistema embarcado (placa TS-7260).	55
FIGURA 11	– Diagrama estados da <i>thread</i> principal da conexão da estação base.	63
FIGURA 12	– Diagrama de estados da <i>thread</i> principal da conexão do linux embarcado.	64
FIGURA 13	– Diagrama de estados da <i>thread</i> que envia mensagens (igual para estação base e linux embarcado).	65
FIGURA 14	– Diagrama de estados da <i>thread</i> receptora de mensagens (igual para estação base e linux embarcado).	66
FIGURA 15	– Diagrama de estados da <i>thread</i> que processa mensagens recebidas (igual para estação base e linux embarcado).	67
FIGURA 16	– Diagrama de estados da <i>thread</i> responsável por gerenciar os comandos dos motores (linux embarcado).	68
FIGURA 17	– Diagrama de estados da <i>thread</i> responsável por efetuar a amostragem dos sensores (linux embarcado).	68
FIGURA 18	– Diagrama de estados do visualização de imagens da <i>webcam</i> (estação base).	69
FIGURA 19	– Diagrama de estados do envio de imagens da <i>webcam</i> (linux embarcado).	70
FIGURA 20	– Diagrama de sequência de comando para mudança de velocidade dos motores (representa comando dado pelo usuário).	71
FIGURA 21	– Diagrama de sequência de comando para mudança de velocidade dos motores (representa mensagem chegando no sistema embarcado).	71
FIGURA 22	– Diagrama de sequência da amostragem dos sensores (representa amostras saindo do sistema embarcado).	72
FIGURA 23	– Diagrama de sequência da amostragem dos sensores (representa mensagem chegando na estação base).	72
FIGURA 24	– Diagrama de sequência de comando para ativação da <i>webcam</i> (representa comando dado pelo usuário e o <i>player</i> da <i>libVLC</i> sendo ativado posteriormente).	73
FIGURA 25	– Diagrama de sequência de comando para ativação da <i>webcam</i> (representa mensagem de ativação da <i>webcam</i> chegando no sistema embarcado e estação base sendo posteriormente notificada sobre o novo status).	73
FIGURA 26	– Diagrama de blocos do hardware	74
FIGURA 27	– Diagrama elétrico/eletrônico.	76
FIGURA 28	– Projeto da PCB – lado de cima.	78

FIGURA 29	– Projeto da PCB – lado de baixo.	78
FIGURA 30	– PCB montada.	79
FIGURA 31	– PCB montada.	80
FIGURA 32	– Janela principal da interface gráfica da estação base.	82
FIGURA 33	– Estação base em utilização na prática.	82
FIGURA 34	– Representação básica do robô em movimento circular uniforme (visão superior).	84
FIGURA 35	– Representação de uma roda acoplada a um encoder.	85
FIGURA 36	– Eixos do acelerômetro e giroscópio.	93
FIGURA 37	– Representação do robô e um sensor infra-vermelho detectando um obstáculo (visão superior).	96
FIGURA 38	– <i>Layout</i> do ambiente do primeiro teste.	99
FIGURA 39	– Mapa gerado no primeiro teste a partir dos dados dos encoders usando filtragem por média móvel de 3 períodos para sensores IR.	100
FIGURA 40	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 3 períodos para sensores IR.	100
FIGURA 41	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) sem filtragem por média móvel dos sensores IR.	101
FIGURA 42	– Trilha do robô (em azul) no primeiro teste calculada somente a partir dos dados do acelerômetro e do giroscópio.	101
FIGURA 43	– Gráfico comparativo de velocidades angulares obtidas no primeiro teste.	102
FIGURA 44	– Gráfico comparativo de acelerações obtidas no primeiro teste.	102
FIGURA 45	– Ambiente utilizado no segundo teste.	104
FIGURA 46	– Mapa gerado no segundo teste a partir dos dados dos encoders.	105
FIGURA 47	– Mapa gerado no segundo teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]).	105
FIGURA 48	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 3 períodos para sensores IR.	106
FIGURA 49	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 8 períodos para sensores IR.	106
FIGURA 50	– Trilha do robô (em azul) no segundo teste calculada somente a partir dos dados do acelerômetro e do giroscópio.	107
FIGURA 51	– Gráfico comparativo de velocidades angulares obtidas no segundo teste.	107
FIGURA 52	– Gráfico comparativo de acelerações obtidas no segundo teste.	108
FIGURA 53	– Ambiente utilizado no terceiro teste.	109
FIGURA 54	– Mapa gerado no terceiro teste a partir dos dados dos encoders.	109
FIGURA 55	– Mapa gerado no terceiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]).	110
FIGURA 56	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 5 períodos para sensores IR.	110
FIGURA 57	– Trilha do robô (em azul) no terceiro teste calculada somente a partir dos dados do acelerômetro e do giroscópio.	111
FIGURA 58	– Gráfico comparativo de velocidades angulares obtidas no terceiro teste.	112

FIGURA 59	– Gráfico comparativo de acelerações obtidas no terceiro teste.	112
FIGURA 60	– Ambiente utilizado no quarto teste.	113
FIGURA 61	– Mapa gerado no quarto teste a partir dos dados dos encoders.	114
FIGURA 62	– Mapa gerado no quarto teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.003$ [rad/s]).	114
FIGURA 63	– Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.003$ [rad/s]) e filtragem por média móvel de 4 períodos para sensores IR.	115
FIGURA 64	– Gráfico comparativo de velocidades angulares obtidas no quarto teste.	115
FIGURA 65	– Gráfico comparativo de acelerações obtidas no quarto teste.	116

LISTA DE TABELAS

TABELA 1	- Comparação entre <i>webcams</i> USB.	24
TABELA 2	- Comparação entre tecnologias para odometria.	27
TABELA 3	- Comparação entre acelerômetros/giroscópios para odometria.	28
TABELA 4	- Requisitos mandatórios para escolha do microcontrolador.	32
TABELA 5	- Requisitos desejáveis para escolha do microcontrolador.	32
TABELA 6	- Comparativo entre microcontroladores.	39
TABELA 7	- Comparação entre tecnologias de comunicação sem fios.	40
TABELA 8	- Comparação entre Bibliotecas para desenhos 2D.	40
TABELA 9	- Comparação entre linguagens de programação.	40
TABELA 10	- Comparação entre sistemas operacionais.	41
TABELA 11	- Relação dos entregáveis com seus respectivos responsáveis e prazos ...	43
TABELA 12	- Preços individuais e totais dos componentes do projeto.	46
TABELA 13	- Pacote <i>visual</i>	52
TABELA 14	- Pacote <i>dados</i>	53
TABELA 15	- Pacote <i>comunicacao</i>	54
TABELA 16	- Pacote <i>gui</i>	54
TABELA 17	- Descrição das classes do sistema embarcado (placa TS-7260)	55
TABELA 18	- Medidas do robô.	129

SUMÁRIO

1 INTRODUÇÃO	10
2 ESPECIFICAÇÃO DE OBJETIVOS	13
3 PREMISSAS E RESTRIÇÕES	15
4 DESIGNAÇÃO DO GERENTE E DA EQUIPE	17
5 TRABALHOS CORRELATOS	18
5.0.1 PatrolBot	18
5.0.2 Sistema de mapeamento robótico bidimensional por infravermelho	18
6 ANÁLISE TECNOLÓGICA	20
6.1 VISÃO GERAL DO PROJETO	20
6.2 REQUISITOS	21
6.2.1 Estação base	21
6.2.2 Sistema de comunicação	22
6.2.3 Sistema embarcado	22
6.3 ANÁLISE DE OPÇÕES TECNOLÓGICAS	23
6.3.1 Sistema embarcado	23
6.3.1.1 Imagens instantâneas do ambiente	23
6.3.1.2 Movimentação do robô	25
6.3.1.3 Odometria	26
6.3.1.4 Detecção de obstáculos	30
6.3.1.5 Microcontrolador	32
6.3.1.6 Placa de circuito impresso	33
6.3.2 Sistema de comunicação	34
6.3.3 Estação base	35
6.3.3.1 Biblioteca para desenhos 2D	35
6.3.3.2 Linguagem de programação	36
6.3.3.3 Sistema operacional	37
7 PLANO DO PROJETO	42
7.1 CRONOGRAMA	42
7.2 DELIVERABLES	42
8 ORÇAMENTO DETALHADO	45
9 MODELAGEM UML	47
9.1 REQUISITOS FUNCIONAIS	47
9.2 REQUISITOS NÃO FUNCIONAIS	47
9.3 CASOS DE USO IDENTIFICADOS	48
9.4 DIAGRAMA DE CLASSES DA ESTAÇÃO BASE	50
9.4.1 Descrição das classes da estação base	51
9.4.2 Pacote <i>visual</i>	51
9.4.3 Pacote <i>dados</i>	52
9.4.4 Pacote <i>comunicacao</i>	52
9.4.5 Pacote <i>gui</i>	53
9.5 DIAGRAMA DE CLASSES DO SISTEMA EMBARCADO	54

9.6 PROTOCOLO DE COMUNICAÇÃO	56
9.6.1 Codificação das mensagens	57
9.6.2 Diagramas de estados	61
9.6.3 Diagramas de sequência	70
10 DIAGRAMAS DO HARDWARE	74
10.1 DIAGRAMA DE BLOCOS	74
10.2 DIAGRAMA ELÉTRICO/ELETRÔNICO	75
10.3 PLACA DE CIRCUITO IMPRESSO	77
10.3.1 Guia de montagem do hardware	79
11 INTERFACE GRÁFICA DA ESTAÇÃO BASE	81
12 GERAÇÃO DO MAPA	83
12.1 ENCODERS	84
12.1.1 Deslocamento de cada roda	85
12.1.1.1 Valores práticos	86
12.1.2 Deslocamento do centro de movimento do robô	87
12.1.2.1 Raio do movimento circular uniforme	87
12.1.2.2 Deslocamento linear	88
12.1.2.3 Deslocamento angular	89
12.1.2.4 Casos especiais	89
12.1.2.5 Considerações sobre a notação utilizada	91
12.1.2.6 Velocidade e aceleração	91
12.2 ACELERÔMETRO E GIROSCÓPIO	91
12.3 ALGORITMO DE POSICIONAMENTO	94
12.4 SENSORES INFRA-VERMELHOS	95
12.4.1 Filtragem	97
13 TESTES	98
13.1 PRIMEIRO TESTE	98
13.2 SEGUNDO TESTE	103
13.3 TERCEIRO TESTE	108
13.4 QUARTO TESTE	113
14 CONCLUSÕES	117
15 TRABALHOS FUTUROS	119
Apêndice A – PLANEJAMENTO DE RISCOS	120
Apêndice B – MEDIDAS DO ROBÔ	129
REFERÊNCIAS	130

1 INTRODUÇÃO

O projeto apresentado neste documento trata-se do “Mapeamento de Ambientes com o robô Bellator” e é uma extensão do projeto “Bellator”. Ele teve sua última alteração em 2012 quando foi utilizado por Alexandre Jacques Marin, Júlio Cesar Nardelli Borges e Yuri Antin Wergrzn como plataforma de experimentos para o projeto final de conclusão de curso. O projeto para a disciplina de Oficina de Integração 3 foi desenvolvido com base nesse robô. Na versão anterior dele, estava presente um conjunto de circuitos (com um microcontrolador) que gerenciava as operações de baixo nível. Além disso, estava presente um PC embarcado (executando o sistema Linux), que efetuava as operações de alto nível.

A equipe deste projeto propôs modificar o robô Bellator para efetuar o mapeamento 2D de ambientes controlados como, por exemplo, labirintos construídos para fins de teste do robô. Posteriormente, em trabalhos futuros, ajustes finos poderão ser feitos para o uso em ambientes diversos, como escritórios, salas e quartos.

Na versão anterior do Bellator, estavam sendo utilizadas duas placas de circuito impresso – uma integrada com o microcontrolador e uma para a interface com os sensores – ambas ligadas por cabos entre si. Ao invés de produzir uma terceira placa para sensores adicionais (aspecto explicado mais à frente), o que aumentaria a quantidade de cabos, foi proposto o desenvolvimento de uma nova placa que realizasse a função de interface com todos os sensores e que fosse acoplada ao microcontrolador. Este microcontrolador pode ser usado diretamente na forma encapsulada de circuito integrado (soldado diretamente na nova placa), ou integrado a um kit de desenvolvimento (acoplado como *shield* na nova placa).

O sistema embarcado do robô desenvolvido pela equipe consiste na placa de interface de sensores acoplada ao microcontrolador. Esse sistema realiza as funções de baixo nível, ou seja, leitura de sensores e controle do PWM dos motores. A estação base é um computador, provido de um software que efetua comunicação bidirecional com o robô. A estação é capaz de enviar comandos de movimentação (especificados manualmente pelo teclado) a ele, além de receber imagens da câmera e leituras dos sensores. No software, a partir das leituras dos

sensores, é produzido um mapa em 2D simplificado do ambiente, com os obstáculos que forem detectados à medida que o robô anda, além do caminho estimado percorrido por ele. Protocolos de comunicação são utilizados entre: circuito de baixo nível e o PC embarcado (através de porta serial), e entre PC embarcado e estação base (através de conexão WI-FI). A placa com sistema linux embarcado foi mantida. Ela serve de interface entre a estação-base e a placa de controle embarcada e possui duas entradas USB. Em uma delas foi colocado um dispositivo de comunicação WI-FI, uma vez que a conexão entre a estação base e o robô têm um alcance de até 20 metros e a tecnologia WI-FI se mostra adequada para a comunicação dentro desse limite de distância. À outra porta foi acoplada uma webcam de forma a permitir que o usuário na estação base acompanhe a movimentação do robô à distância.

Um aspecto importante a ser notado é a exatidão e confiabilidade das medições de velocidade. No robô anterior havia somente dois encoders, um para cada roda – a partir dos quais pode ser medida a velocidade e distância percorrida. Há certas desvantagens em utilizar essa abordagem, que são principalmente as questões de exatidão. Por exemplo, caso alguma roda escorregue, gire em falso ou sofra trepidações, as medições podem ser comprometidas – gerando distorções no mapa 2D. Por isso, instalou-se novos sensores na carcaça do robô (acelerômetro e giroscópio) para adicionar maior confiabilidade nas medições do sistema – tendo em vista que esses sensores mensuram o movimento real do robô e não somente o giro das rodas. Dessa forma, teoricamente, poderia se ter uma maior garantia de exatidão nos mapas gerados, levando-se em conta que a velocidade e posição do robô seriam melhor determinados. Especialmente em trabalhos futuros, se o robô for utilizado em ambientes acidentados ou em condições não ideais de terreno, esses sensores podem ser de grande valia – uma vez que nesses ambientes há maior chance das rodas escorregarem, girarem em falso ou trepidarem.

Esta monografia está organizada da seguinte forma: os capítulos 2 e 3 abordam os objetivos a serem explorados e o escopo do projeto. No quarto capítulo é feita a apresentação dos membros da equipe e do gerente. Os capítulos 5 e 6 relatam as pesquisas efetuadas antes de se iniciar o desenvolvimento do projeto. Mais especificamente, o capítulo 5 apresenta abordagens semelhantes encontradas e o capítulo 6 consiste de uma análise detalhada de alternativas tecnológicas possíveis de serem utilizadas e justifica as escolhas tecnológicas feitas. Os capítulos 7 e 8 tratam do planejamento traçado para a realização do projeto, incluindo as tarefas a serem realizadas, os responsáveis por elas, os prazos e os custos totais envolvidos. A seguir é discutido o desenvolvimento do projeto: no capítulo 9 é mostrada a modelagem UML da estação-base e do servidor linux embarcado, a questão dos protocolos de comunicação também é abordada; no capítulo 10 discute-se o hardware desenvolvido, desde seus diagramas até sua confecção em circuito impresso e montagem.

Os capítulos 11 a 13 tratam dos resultados obtidos. No capítulo 11 faz-se a introdução da interface gráfica da estação-base. No capítulo 12 é feita a fundamentação teórica da geração dos mapas a partir da leitura dos encoders ópticos das rodas. No capítulo 13 é feita a apresentação dos três testes efetuados para validação dos resultados obtidos. O capítulo 14 apresenta as conclusões e observações. No capítulo 15 são feitas sugestões de melhorias e de projetos futuros que podem ser desenvolvidos com base no presente trabalho. Por último, o anexo A apresenta o planejamento de riscos contendo eventuais falhas que poderiam ocorrer durante a realização do projeto e o anexo B contém as dimensões físicas do robô.

A realização deste trabalho teve a duração total de 10 semanas entre seu planejamento inicial, desenvolvimento, testes e documentação.

2 ESPECIFICAÇÃO DE OBJETIVOS

OBJETIVOS:

- Implementar um software para comunicação de uma estação base (computador) com o robô, de forma que ela possa enviar comandos de movimentação ao robô, além de receber imagens da câmera e leituras dos sensores. Os comandos de movimentação (mover para frente, para trás, girar para esquerda/direita, parar) serão especificados por um utilizador humano através do teclado da estação base.
- O meio de comunicação entre a estação base e o robô deverá ter alcance máximo de 20 m (se não houverem paredes ou obstáculos entre a estação base e o robô). Para isso a tecnologia WI-FI mostra-se adequada e, portanto, ela será utilizada.
- Inserir uma *webcam* USB no robô, de modo que imagens do ambiente possam ser transmitidas à estação base. O propósito das imagens será unicamente permitir a visualização (pelo usuário da estação base, em tempo real) do ambiente no qual o robô está localizado. A câmera será conectada na porta USB do computador embarcado, e a transmissão de imagens será feita pelo canal Wi-Fi entre a estação base e o robô (o mesmo canal utilizado para a transmissão de dados dos sensores e comandos de movimentação).
- Implementar, no software utilizado na estação base, a geração de uma mapa em 2D com o caminho estimado percorrido pelo robô e os obstáculos detectados pelo mesmo. Os obstáculos serão representados a partir dos pontos em que houve detecção pelos sensores.
- Instalar novos sensores (acelerômetro e giroscópio) para efetuar as medições de velocidade e posicionamento do robô com maior exatidão do que pode ser feito atualmente com os *encoders*. Ambos os sensores serão posicionados na carcaça do robô. Caso discrepâncias de medição entre os *encoders*, acelerômetro e giroscópio sejam detectadas (por exemplo, em caso de escorregamento de rodas), attenuações de erros poderão ser feitas no *software* da estação base.

- Desenvolver uma placa de circuito impresso que realize a função de interface com os sensores e que seja acoplada ao microcontrolador. Este microcontrolador pode ser usado diretamente na forma encapsulada de circuito integrado (sendo soldado diretamente na nova placa) ou integrado a um kit de desenvolvimento (acoplado como *shield* na nova placa).
- Em caso de falha de comunicação entre o robô e a estação base, o robô deverá permanecer parado e aguardando a conexão ser reestabelecida.

3 PREMISSAS E RESTRIÇÕES

PREMISSAS:

- Por ser utilizado o robô Bellator que já provém de trabalhos anteriores, infere-se que não haverá necessidade de haver gastos de tempo com consertos de equipamentos defeituosos ou correções de bugs no código fonte. Parte-se do pressuposto que o robô funciona de acordo com o que foi exposto nos relatórios anteriores.
- O robô é capaz de detectar obstáculos (paredes e objetos fixos de tamanho considerável que sejam maiores que ele) através dos sensores. A distância mínima para detecção é de 20cm e a máxima de 150cm.
- O robô é capaz de locomover-se em terrenos planos, não acidentados e em condições não severas.
- Pressupõe-se que o robô será disponibilizado para a equipe sem custos. Podem ser utilizados os equipamentos e componentes diversos que já estejam disponíveis, com o objetivo de redução de custos.

RESTRIÇÕES:

- O tempo disponível para a equipe é limitado, portanto muita atenção será dada às fases de planejamento e testes iniciais de modo a evitar imprevistos.
- A equipe deverá seguir um calendário previamente estabelecido, tendo o objetivo de evitar atrasos.
- O robô não será capaz de se locomover em terrenos acidentados, em escadas e similares.
- O robô não transportará cargas.
- O robô e a estação base não executarão algoritmos de roteamento ou mapeamento autônomo de ambientes. O controle de movimentação deverá ser feito obrigatoriamente por um

usuário humano junto à estação base. O robô não fará nenhuma movimentação automática em caso de falha de conexão. Ele permanecerá parado aguardando a conexão ser reestabelecida.

- O robô e a estação base não serão capazes de efetuar mapeamento 3D.
- O robô e a estação base não irão armazenar automaticamente fotos ou vídeos dos ambientes explorados.
- O robô e o ponto de acesso WI-FI da estação base devem estar a uma distância máxima de 20 metros um do outro (supondo que não hajam paredes ou obstáculos). Caso contrário, não haverá garantias de que a comunicação entre a estação base e o robô seja funcional.
- Não serão usadas imagens do ambiente para a geração dos mapas.
- Os obstáculos não serão identificados quanto ao tipo ou forma. Serão apenas detectados pela sua presença.

4 DESIGNAÇÃO DO GERENTE E DA EQUIPE

A equipe consiste de cinco integrantes: Pedro Alberto de Borba, Ricardo Farah, Stefan Campana Fuchs e Telmo Friesen e Luis Guilherme Machado Camargo, tendo este último sido o gerente.

5 TRABALHOS CORRELATOS

O mapeamento de ambientes realizado por robôs visa ao desenvolvimento de software e hardware que permitam a construção de um mapa a partir de dados captados por um ou mais sensores. Há diversas tecnologias que podem ser empregadas para alcançar tal objetivo, como o processamento de digital de imagens captadas de uma câmera ou a utilização de sensores de proximidade tais como sensores de ultrassom ou sensores de ondas eletromagnéticas.

Esta última opção mostra-se bastante adequada para a maioria dos projetos, pois garante uma medição satisfatória da distância de objetos próximos ao robô a um custo não muito elevado. Um dos sensores mais populares deste tipo é o sensor de proximidade de infravermelho. Quando integrado ao robô permite a obtenção várias medidas discretas da distância do robô a objetos, um dos elementos básicos que permitem a geração do mapa do ambiente.

5.0.1 PatrolBot

O PatrolBot (BOT, 2013) é um robô configurável desenvolvido com interesses comerciais. Ele pode criar uma planta do interior de construções. Utilizando a tecnologia WI-FI ele pode ser controlado remotamente ou se movimentar de forma autônoma, sendo apenas monitorado pela estação base. Tal comunicação é feita por Wi-Fi.

Ele permite a inclusão de acessórios adicionais tais como uma câmera e microfones que permitirão ver e ouvir o que se passa no ambiente que está sendo mapeado. Há diversos outros periféricos que podem ser incluídos no robô, que também oferece a opção de ser programado, por meio de um kit de desenvolvimento próprio.

5.0.2 Sistema de mapeamento robótico bidimensional por infravermelho

Nesta implementação (SATO, 2013), um telêmetro infravermelho é utilizado para obter a distância de objetos próximos. A câmera infravermelha do Nintendo Wii é utilizada juntamente com sinalizadores (LEDs) para triangular a posição do robô e sua direção.

Como hardware, foram utilizados: um Arduino Mini Pro, um cartão microSD, telêmetro inframovelho, câmera do Wii. Não há comunicação em tempo real com a estação base. Isto significa que os dados são obtidos e armazenados no cartão microSD. Para leitura, o cartão deve ser inserido em um computador e, então, carregado na estação base. Os resultados são visualizados em um arquivo textual simples, no qual a letra "O" simboliza a posição do robô e a letra "X" os obtáculos detectados.

6 ANÁLISE TECNOLÓGICA

Nesta seção está explicitada, primeiramente, uma visão geral do projeto. Em seguida, há uma discussão detalhada a respeito dos requisitos de cada parte fundamental – estação base, sistema de comunicação e sistema embarcado. Por fim há uma enumeração das alternativas tecnológicas pesquisadas e das escolhidas para o preenchimento dos requisitos.

6.1 VISÃO GERAL DO PROJETO

O projeto, de um ponto de vista geral, consiste em um robô controlado manualmente e que seja capaz de efetuar mapeamento em duas dimensões de ambientes. Um usuário humano monitorará e controlará um computador – a estação base – a partir do qual poderão ser enviados comandos de movimentação, via teclado, ao robô. Informações sobre o posicionamento do robô e dos obstáculos detectados por ele serão recebidas na estação base em tempo real. Imagens instantâneas de uma câmera posicionada no robô – aspecto explicado mais à frente – poderão ser visualizadas pelo utilizador.

O sistema de comunicação deverá ter, ao menos, alcance de 20 metros sem fios. Visto que toda a comunicação entre a estação base e o sistema embarcado será feita por um único canal, a velocidade e o tipo de fluxo de transmissão de dados devem ser adequados para, simultaneamente, o envio de comandos de movimentação ao robô, recebimento de dados de leituras de sensores e recebimento de imagens da câmera.

O sistema embarcado é constituído, em suma, pelo robô. Ele deve ser capaz de se mover para frente e para trás e girar para a esquerda e direita. A visualização em tempo real do ambiente pelo utilizador, com o objetivo de facilitar o controle de movimentação manual, poderá ser feita através de imagens instantâneas geradas por uma câmera fixa instalada no robô.

O robô deve ser capaz de obter dados para cálculos (na estação base) de sua velocidade e deslocamento. Um aspecto desejável em relação a isso é a atenuação de erros em decorrência de escorregamento, giros em falso ou trepidação de rodas, visando, dessa forma, a

utilização futura do robô em condições não ideais de terreno. Obstáculos próximos – em uma distância mínima de 30 cm e máxima de 150 cm – devem ser detectados de modo a possibilitar a confecção de um mapa 2D em tempo real na estação base.

6.2 REQUISITOS

6.2.1 Estação base

Esta seção descreve os requisitos da estação base, que foram elaborados de forma a satisfazer os objetivos do projeto.

- O *software* será executado em um computador pessoal.
 - O programa poderá ser executado em computadores pessoais de desempenho médio (de acordo com os padrões atuais). Não haverá necessidade de uma máquina de alto desempenho e custo relativo para executar o *software*.
 - O *software* primariamente será executado em um único sistema operacional, sendo este Linux ou Windows. É desejável que o desenvolvimento (incluindo a escolha das ferramentas) seja feito de forma a simplificar o uso multiplataforma do *software* futuramente.
 - Preferencialmente bibliotecas e ferramentas livres (e gratuitas) deverão ser utilizadas no desenvolvimento do *software*.
- O *software* deve possuir uma interface gráfica.
 - Um utilizador, através da interface gráfica, será capaz de controlar o robô enviando comandos de movimentação especificados pelo teclado.
 - O usuário receberá a imagem em tempo real de uma câmera fixa instalada no robô.
 - Os dados instantâneos de velocidade e posição do robô serão mostrados ao usuário na interface gráfica.
 - Um mapa 2D do caminho percorrido e dos obstáculos detectados pelo robô será gerado, na interface gráfica, à medida em que houver movimentação do mesmo. O caminho percorrido pelo robô será representado visualmente por pontos, gradualmente posicionados no mapa. Os obstáculos serão representados por marcações nas localidades onde houve detecção de objetos por sensores do robô. Todos os pontos representados no mapa serão gerados a partir de amostras obtidas em intervalos de tempo discretos.

- O mapa 2D gerado na interface gráfica poderá ser salvo em um arquivo, podendo ser posteriormente carregado.

6.2.2 Sistema de comunicação

Esta seção descreve os requisitos do sistema de comunicação entre a estação base e o sistema embarcado.

- Distância entre robô e estação base.
 - O sistema de comunicação deve possuir, ao menos, alcance de 20 metros sem fios – de modo que ambientes de tamanho razoável possam ser mapeados.
- Velocidade e direção do fluxo de transmissão de dados.
 - Toda a comunicação entre a estação base e o sistema embarcado será feita por um único canal sem fios e, portanto:
 - A velocidade de transmissão do canal de comunicação deve ser suficiente para, simultaneamente, o envio de comandos de movimentação ao robô, recebimento de dados de leituras de sensores e recebimento de imagens da câmera;
 - O fluxo de dados deve ser bidirecional (*full-duplex*).
- Protocolo de transporte.
 - A tecnologia utilizada para a comunicação deve permitir fácil utilização do protocolo de transporte TCP. Como as leituras de sensores devem obrigatoriamente ser recebidas na estação base na mesma ordem em que forem enviadas pelo robô (e também os comandos de movimentação enviados pela estação base devem chegar ao robô em ordem), o uso desse protocolo de transporte simplificará muito a implementação do protocolo de aplicação ponto a ponto. O TCP possui ainda outro aspecto interessante: além de garantir a ordem de chegada, existem mecanismos de detecção de perdas de pacotes – que efetuam o reenvio destes caso necessário.

6.2.3 Sistema embarcado

Esta seção descreve os requisitos do sistema embarcado (robô).

- Imagens instantâneas do ambiente.

- O robô, através de uma câmera fixa, deverá ser capaz de enviar à estação base imagens instantâneas do ambiente onde ele se encontra.
- Movimentação do robô.
 - O robô deve ser capaz de mover-se para frente, para trás e girar para a esquerda e direita.
- Controle de posicionamento e velocidade.
 - O robô deve ser capaz de obter dados que permitam calcular sua velocidade e posição atual (deslocamento e sentido em relação à posição inicial). Deve ser capaz de enviar os dados à estação base.
- Detecção de obstáculos.
 - O robô deverá ser capaz de detectar obstáculos próximos – com distância de no mínimo 30 cm e no máximo 150 cm – localizados ao seu redor, determinando a distância de cada objeto detectado.

6.3 ANÁLISE DE OPÇÕES TECNOLÓGICAS

Nesta seção está apresentada a análise das opções tecnológicas plausíveis para o atendimento dos requisitos. As alternativas pesquisadas e as escolhidas para cada parte do projeto estão explicitadas a seguir.

6.3.1 Sistema embarcado

Nesta seção serão apresentadas as alternativas pesquisadas para o sistema embarcado, levando-se em conta os requisitos da Seção 6.2.3.

6.3.1.1 Imagens instantâneas do ambiente

Para a obtenção de imagens do ambiente onde o robô se encontra, optou-se por utilizar uma *webcam* USB conectada à placa TS-7260 já presente no robô. Essa placa possuirá o *hardware* de comunicação Wi-Fi do robô, e deve-se relembrar que um único canal sem fios será utilizado (como explicitado nos requisitos). Mostra-se adequada, portanto, a utilização de uma câmera que possa ser conectada por USB à placa, de modo que as imagens possam ser transmitidas por esse canal WI-FI. Abaixo estão listadas as características desejáveis da *webcam*:

- Possuir conexão USB 2.0;
- Ser compatível com Linux;
- Ser capaz de, no mínimo, produzir imagens em resolução VGA (640x480), RGB 24 bits a 30 fps, para que a visualização possa ser feita com qualidade satisfatória.

A compatibilidade com o Linux pode ser garantida com a escolha de uma *webcam* em conformidade com o padrão UVC (*USB Video Class*) e compatível com o *driver* Video4Linux 2 (V4L2), presente nos *kernels* do Linux a partir da versão 2.5. Uma lista de dispositivos que seguem esse padrão está disponível em (TOOLS, 2013). Três câmeras de custo baixo e com disponibilidade no Brasil foram selecionadas a partir da lista, como apresentado na Tabela 1.

Tabela 1: Comparaçāo entre *webcams* USB.

Característica	Genius 2000	FaceCam	Microsoft VX-500	Genius 1300
Conexão USB 2.0	Sim	Sim	Sim	Sim
Compatível com Linux	Sim	Sim	Sim	Sim
Resolução máxima de vídeo	1620X1200	640x480	1280X1024	
RGB 24 bits	Sim	Sim	Sim	
Taxa de amostragem	30 fps	30 fps	30 fps	
Custo	R\$ 82,00	R\$ 79,50	R\$ 34,99	

Percebe-se que as características técnicas de todas as três câmeras são satisfatórias para o preenchimento dos requisitos. Porém, há certas diferenças relacionadas a custo e resolução.

Escolha da equipe: A *webcam* escolhida foi a Genius iSlim 1300, principalmente tendo em vista o seu custo muito reduzido em relação às outras. Além disso, possui resolução muito satisfatória. Em Curitiba, há disponibilidade desse modelo em pronta entrega.



Figura 1: Webcam Genius iSlim 1300.

Fonte: (GENIUS, 2013)

6.3.1.2 Movimentação do robô

Uma vez que o sistema de movimentação do robô, incluindo motores, acionadores, drivers de potência e rodas já estão instalados no robô e atendem aos requisitos, não houve nova pesquisa sobre esses componentes. Um chassi de 40 cm de largura por 50 cm de comprimento, duas rodas de tração e uma roda guia estão presentes atualmente. As rodas de tração estão dispostas na parte posterior do robô, possuindo 20 cm de diâmetro e 4 cm de largura. O chassi está equipado com 2 motores Bosch FPG 12V, 2 baterias Unybatt 12V-7,2 Ampére-hora e duas pontes H L298 (MARIN et al., 2012). A disposição dos itens no robô pode ser vista na figura 2.

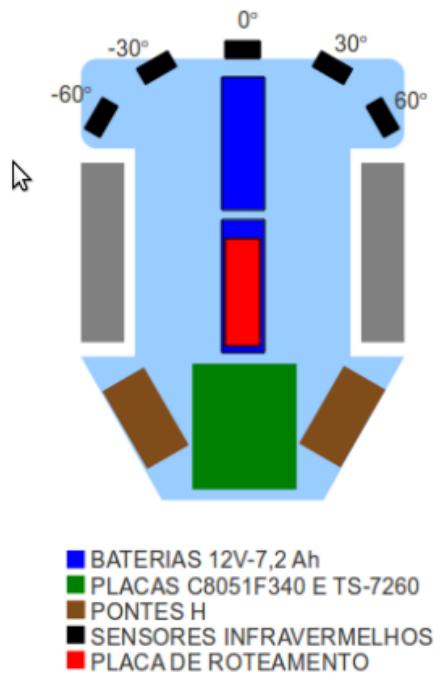


Figura 2: Disposição dos itens no robô

Fonte: (MARIN et al., 2012)

6.3.1.3 Odometria

Para a obtenção da aceleração, velocidade e posição do robô, diversas tecnologias podem ser escolhidas. Abaixo serão descritas as principais opções:

- **Encoder:** Ligado ao eixo da roda do robô, efetua a contagem das rotações realizadas por ela, permitindo assim calcular a distância percorrida. Se dois encoders forem instalados, um em cada roda, a direção do movimento poderá ser obtida a partir de cálculos baseados na contagem de voltas de cada roda.
- **GPS:** Utiliza sinais de satélites para obter as coordenadas geográficas do robô. A direção e o sentido do movimento podem ser obtidos a partir da comparação das leituras atuais com as anteriores.
- **Acelerômetro:** Pode utilizar a tecnologia chamada MEMS para medir a aceleração do componente. A velocidade e deslocamento lineares podem ser obtidos por integração numérica da aceleração.
- **Giroscópio:** Pode utilizar a tecnologia chamada MEMS para medir a aceleração angular do componente. A velocidade angular e ângulo de rotação podem ser obtidos por

integração numérica da aceleração angular.

- **Bússola:** Utiliza os campos magnéticos da terra para obter a orientação geográfica absoluta do robô.

Na Tabela 2 está presente uma comparação entre as tecnologias apresentadas.

Tabela 2: Comparação entre tecnologias para odometria.

Característica	Encoder	GPS	Acelerômetro	Giroscópio	Bússola
Sujeito a influências externas	Deslizamentos	Não	Não	Não	Ruídos de campos magnéticos diversos
Ambiente de operação	Interno / Externo	Externo	Interno / Externo	Interno / Externo	Interno / Externo
Posicionamento	Relativo	Absoluto	Relativo	Relativo	Absoluto
Acumulo de erro para calculo da posição	Sim	Não	Sim (duas integrações)	Sim (duas integrações)	Sim

Da Tabela 2, vê-se que *encoders* estão sujeitos a erros causados por deslizamentos nas rodas, e que GPS apenas funciona em ambientes externos. A bússola pode ser influenciada por campos magnéticos diferentes do da terra – como por exemplo o gerado pelos motores. O acelerômetro e o giroscópio por sua vez acumulam o erro de duas integrações¹ para obtenção da posição.

Escolha da equipe: Os *encoders* estão sujeito apenas aos erros de deslizamentos e acumulam menos erros na obtenção da posição do que os giroscópios e acelerômetros. Sendo assim eles foram a escolha como principal fonte de dados para odometria. O GPS não foi escolhido pois opera apenas em ambientes internos. A bússola por sua vez poderá sofrer influencias do campo magnético gerado pelos motores do robô. Como a utilização apenas dos *encoders* pode levar a erros no posicionamento devido a deslizamentos, utilizaremos também um acelerômetro e um giroscópio como fonte de dados auxiliar para possibilitar o aumento da exatidão e confiabilidade dos dados obtidos dos *encoders*. Caso discrepâncias consideráveis ocorram entre os dados obtidos pelos sensores, escorregamentos das rodas podem ser detectados e mitigados, atenuando dessa forma erros na determinação do posicionamento.

¹ As acelerações linear e angular devem ser integradas duas vezes numericamente para cálculo da posição: A primeira para determinação das velocidades linear e angular; A segunda para determinação do deslocamento e ângulo de rotação atuais.

Por exemplo, em caso de escorregamento roda, um *encoder* fornece uma medição de velocidade maior do que a que corresponde à realidade do movimento do robô. O acelerômetro e o giroscópio, por sua vez, não sofrem influências do escorregamento das rodas, e tenderão a fornecer uma medida mais próxima à realidade. Discrepâncias nas medições podem ser dessa forma detectadas, e procedimentos de atenuação de erros (como por exemplo, descarte de certas medidas dos *encoders*) poderão ser executados no *software* da estação base.

Os *encoders* que serão utilizados (HEDS-9700) já se encontram acoplados ao robô, logo os motivos para a escolha do modelo não serão analisados. Quanto ao acelerômetro e giroscópio, está apresentado a seguir, na Tabela 3, um comparativo entre as opções de menor custo disponíveis no mercado. Na Tabela estão listados apenas os modelos que possuem placas de desenvolvimento, pois acelerômetros e giroscópios geralmente são vendidos em encapsulamento LGA ou BGA (que são de difícil soldagem).

Tabela 3: Comparação entre acelerômetros/giroscópios para odometria.

Modelo	Fabricante	Acel.	Giro.	Faixa	Interface	Preço
STEVAL-MKI009V1	STMicroelectronics	3x	-	$\pm 2g$ $\pm 6g$	ou I2C / SPI	\$23.94
ATAVRSBIN1	Atmel	1x	-	-	I2C	\$26.25
KIT3803 MMA7660FC	Freescale	3x	-	$\pm 1.5g$	I2C	\$35.0
ATAVRSBIN1	Atmel	-	1x		I2C	\$26.25
MPU-6050	IvenSense	3x	3x	$\pm 2g$ $\pm 4g$; $\pm 250^\circ/\text{seg}$ ou $\pm 500^\circ/\text{seg}$	I2C	\$8.78
MKI086V1	STMicroelectronics	-	1x	$\pm 30^\circ/\text{seg}$	Analog	\$31.50
STEVAL-MKI094V1	STMicroelectronics	-	3x	$\pm 400^\circ/\text{seg}$	Analog	\$31.50
ATAVRSBIN1	Atmel	1x	1x		I2C	\$26.25
DM240316	Zena	3x	3x		RF	\$99.99

MPU-6050

Com base na Tabela 3, o modelo MPU-6050 da IvenSense foi escolhido, principalmente devido ao seu baixo custo: \$8.78. Este modelo possui um acelerômetro e um giroscópio (ambos de 3 eixos), além entradas para uma bússola externa de 3 eixos, tudo integrado a um único chip (EVENSENSE, 2013). A faixa de operação para o acelerômetro é de $\pm 2g$ ou $\pm 4g$ e para o giroscópio é de $\pm 250^\circ/\text{seg}$ ou $\pm 500^\circ/\text{seg}$. A sensibilidade do acelerômetro

é de 16384 LSB/g ou 8192 LSB/g . A sensibilidade do giroscópio é de 131 $LSB/(^{\circ}/seg)$ ou 65.5 $LSB/(^{\circ}/seg)$. A interface de comunicação do módulo suporta o protocolo I2C. O módulo contendo o chip MPU-6050 e alguns componentes necessários para seu funcionamento pode ser visto na figura 3.

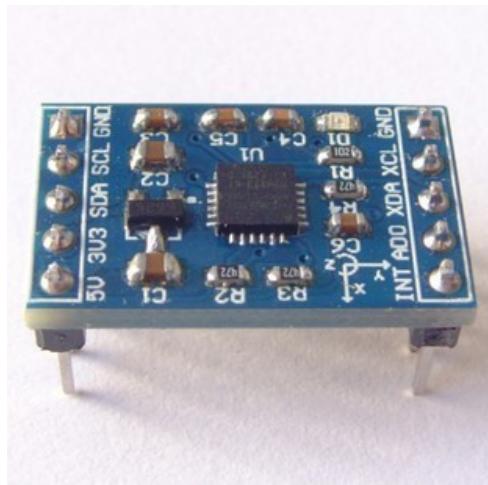


Figura 3: Placa de desenvolvimento contendo o chip MPU-6050

Fonte: (EVENSENSE, 2013)

Encoder Optico HEDS-9700

Como foi explicitado anteriormente, os *encoders* ópticos já existentes no robô serão utilizados. Ele está equipado com duas unidades do modelo HEDS-9700. Quanto ao funcionamento, este modelo gera em sua saída uma onda quadrada à medida em que as rodas são rotacionadas, sendo 1800 pulsos gerados em uma rotação completa. A forma de onda da saída do *encoder* pode ser vista na Figura 4. Pode-se ver na Figura que o *encoder* possui duas saídas, A e B com defasamento ϕ entre elas. O sentido de rotação pode ser determinado pela informação de qual sinal (A ou B) está mais adiantado em fase (AGILENTTECHNOLOGIES, 2002). A leitura e contagem das rotações do encoder serão feitas conforme foi desenvolvido no projeto anterior (descrito em Marin et al. (2012)).

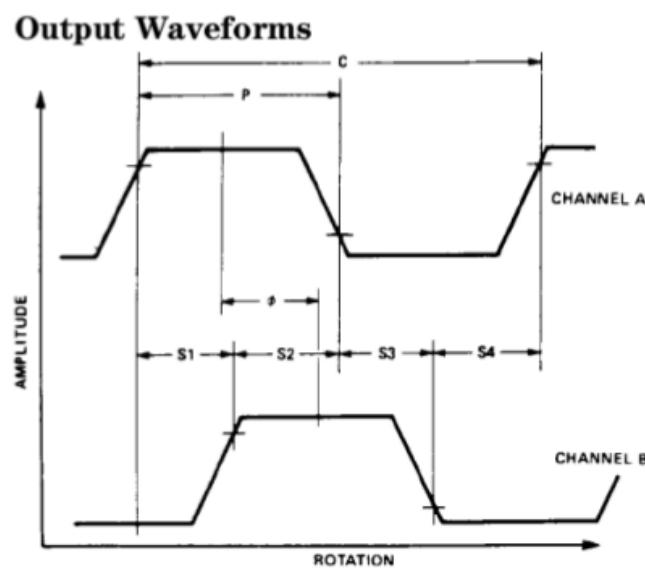


Figura 4: Forma de onda na saída do encoder
Fonte: (AGILENTTECHNOLOGIES, 2002)

6.3.1.4 Detecção de obstáculos

Sensor de proximidade Infra Vermelho IR 2Y0A02F98

A detecção de obstáculos, que é um requisito para o projeto robô, será feita pelos sensores de Infra Vermelho já existentes nele. Estão presentes 5 unidades do modelo IR 2Y0A02F98 (SHARPCORPORATION, 2006). Uma característica interessante deste sensor é que ele sofre pouca influência das cores dos objetos, devido ao método de medição baseado em triangulação. Na figura 5 pode-se verificar o fato. A linha tracejada é a resposta para reflexão em uma superfície cinza e a linha contínua é a resposta para reflexão em uma superfície branca.

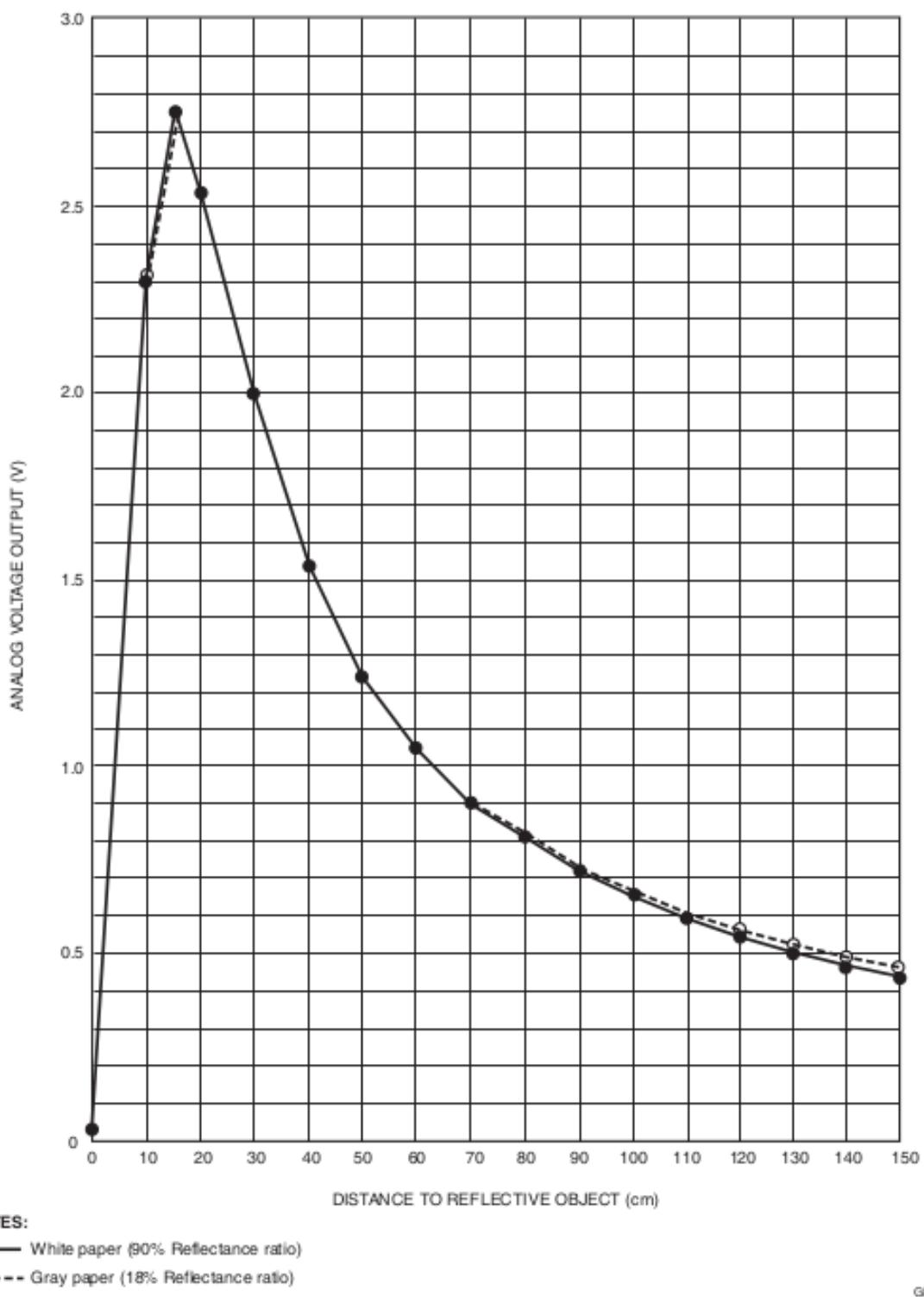


Figura 5: Curva de resposta do sensor Infra Vermelho IR 2Y0A02F98

Fonte: (SHARPCORPORATION, 2006)

6.3.1.5 Microcontrolador

A interface entre sensores e atuadores (sistemas de acionamento dos motores) com a placa TS-7260² será feita por um sistema microcontrolado. Este sistema deve possuir as interfaces adequadas para comunicação com todos os componentes. Na Tabela 4 estão listados os requisitos mandatórios para escolha do microcontrolador, e na Tabela 5 estão listados os requisitos desejáveis (porém não obrigatórios).

Tabela 4: Requisitos mandatórios para escolha do microcontrolador.

Requisito	Justificativa
Interface I2C	Comunicação com acelerômetro e giroscópio
Geração de PWM em 4 canais	Acionamento dos motores pelas pontes H
Interface Serial	Comunicação com a placa TS-7260
Interrupções em 2 canais (com capacidade de processamento de no mínimo 2865 interrupções/segundo em cada canal ³)	Leitura do valor dos <i>encoders</i>
Conversor AD em 5 canais	Leitura dos sensores de IR

Tabela 5: Requisitos desejáveis para escolha do microcontrolador.

Requisito desejável	Justificativa
Desenvolvimento em plataforma livre	Diminuição do custo de softwares para desenvolvimento
2 Interfaces seriais ou JTAG	Utilização para <i>debug</i> ou <i>logs</i>
Solução integrada em um único chip	Redução do tamanho da placa e da quantidade de componentes, diminuindo assim o custo e melhorando a organização e disposição dos mesmos

Na Tabela 6 estão listados diversos microcontroladores que foram pesquisadas para o projeto. Todos os modelos atendem aos requisitos da Tabela 4.

Escolha da equipe: Dentre as opções listadas na Tabela 6, o modelo LPC2103 foi escolhido. A escolha foi feita baseando-se principalmente no custo do microcontrolador. Nota-se, a primeira vista ao efetuar-se uma comparação com o SIM3C146, que essa última opção

²Deve-se ressaltar que os atuadores (sistemas de acionamento dos motores) e a placa TS-7260 já existem no robô (MARIN et al., 2012).

³Valor calculado com base no tamanho das rodas, supondo velocidade máxima de deslocamento de 1m/s.

possui desempenho melhor e custo ligeiramente menor. Porém a escolha do LPC2103 justifica-se pela melhor documentação e mais ampla disponibilidade de recursos para o mesmo. A documentação fornecida pelo fabricante desse microcontrolador é mais completa e, pelo fato do modelo estar há mais tempo no mercado, a quantidade de informações e recursos disponíveis na *internet* é maior.

LPC2103

O LPC2103 é um microcontrolador baseado na arquitetura ARM7TDMI-S da NXP (NXP, 2013). Este microcontrolador pode operar em até $70MHz$ executando a $63MIPS$. O Microcontrolador possui 2 interfaces I2C, 2 interfaces seriais, até 14 saídas de PWM, até 13 canais de interrupções externas, 8 canais de conversão para um conversor analógico digital de 10 bits, 32kbytes de memória FLASH para código e 8kbytes de memória RAM. Ele também suporta *debug* via JTag por meio de um *debugger* externo. O custo desse microcontrolador é de \$6.16 (CORPORATION, 2013). O LPC2103 está disponível em encapsulamento LQFP com 48 pinos.

O microcontrolador escolhido pode também ser programado utilizando o protocolo ISP, por meio de ferramentas livres como o lpc21isp (LPC21ISP, 2013). Para a geração do código hexadecimal utilizado pelo lpc21isp, basta efetuar a compilação de código em C utilizando o GCC (GCC, 2013). Pode-se notar que o LPC2103, além de atender aos requisitos mandatórios da Tabela 4, também atende aos requisitos desejáveis que foram expostos na Tabela 5.

6.3.1.6 Placa de circuito impresso

A placa de circuito impresso será projetada utilizando ferramentas livres, como o gEDA (GEDA, 2013) e PCB (PCB, 2013). Após o projeto da placa os arquivos *Gerber* serão enviados à empresa Stick (IMPRESSOS, 2013) para impressão. A soldagem dos componentes será realizada pela própria equipe.

Deve-se ressaltar que como, em geral, na soldagem de um *chip* com encapsulamento LQFP há complexidade e riscos consideráveis (como por exemplo, queima do *chip* ou da placa), a equipe pesquisou informações sobre *kits* de desenvolvimento que tenham o LPC2103 já soldado na placa. Em Curitiba, existe a empresa eSysTech (ESYSTECH, 2013) que fabrica tais *kits*. Através de contatos da UTFPR, confirmou-se que na universidade há algumas unidades (fabricadas por essa empresa) que podem ser emprestadas à equipe para a realização do projeto, caso haja necessidade.

6.3.2 Sistema de comunicação

Como foi explicitado nos requisitos, o sistema de comunicação ter as seguintes características:

- Alcance de, no mínimo, 20 metros sem fios;
- Velocidade suficiente para, simultaneamente, o envio de comandos de movimentação ao robô, recebimento de dados de leituras de sensores e recebimento de imagens da câmera;
- Fluxo de dados bidirecional (*full-duplex*);
- Utilização simples do protocolo TCP.

Para o cálculo da taxa de transmissão mínima necessária, faz-se uma estimativa inicial. Prevê-se que o envio de imagens da câmera é o que mais utilizará os recursos da conexão. Supondo serem usadas a máxima qualidade e taxa de amostragem suportadas por uma câmera VGA comum (resolução 640x480, RGB 24 bits, 30 FPS), além de uso de compressão JPEG com 90% de qualidade, haverá uso de aproximadamente 2,4 MB/s ou 19,2 MBits/s (RESOLUTION, 2013).

O envio de comandos ao robô, supondo que cada comando tenha 2 KB e hajam 10 comandos por segundo (estimativa exagerada), gastará em torno de 160 KBits/s. O recebimento de informações do robô e leituras dos sensores, supondo que cada pacote tenha 2 KB (com exagero) e haja recebimento de 10 pacotes por segundo, utilizará 160 KBits/s na banda da conexão. Portanto, o valor mínimo desejável da taxa de transferência é de $19,2 + 0,16 + 0,16 = 19,52$ MBits/s.

Na Tabela 7 está presente uma comparação entre diferentes tecnologias de comunicação sem fios, que potencialmente podem satisfazer às necessidades do projeto.

Escolha da equipe: O Wi-Fi é o recurso mais atrativo em todos os aspectos que foram comparados, preenchendo satisfatoriamente os requisitos do sistema de comunicação. Sua velocidade e alcance são suficientes para satisfazer as necessidades do projeto, e o fluxo de dados pode ser *full-duplex*. Notavelmente, o Wi-Fi é o único sistema comparado que oferece a possibilidade (com simplicidade) de uso do protocolo TCP – o que é um requisito importante para o desenvolvimento ágil e satisfatório do projeto.

É importante ressaltar que uma conexão Wi-Fi 802.11g dedicada será utilizada para a comunicação entre o robô e a estação base, de modo que a velocidade de conexão possa ser utilizada com maior eficiência sem interferências de outros utilizadores.

6.3.3 Estação base

As alternativas pesquisadas para a estação base estão apresentadas nesta subseção.

6.3.3.1 Biblioteca para desenhos 2D

Tendo em vista que um dos requisitos é a geração de um mapa em duas dimensões na estação base, deve-se escolher uma biblioteca que permita realizar o desenho de formas geométricas em 2D (via código-fonte) e que possa ser integrada facilmente à interface gráfica. Ela deve também possuir meios simples de obter informações do mouse e teclado, para interatividade com o usuário.

Uma biblioteca interessante disponível em Java que possui o recurso de produzir desenhos dinâmicos com integração a interfaces gráficas é o Processing (PROCESSING, 2013), de código livre. Essa biblioteca foi a principal encontrada que seria capaz de satisfazer as necessidades de desenho do mapa 2D de forma simples. Por possuir inúmeras funções de desenho em alto nível, o trabalho de renderização dos gráficos seria consideravelmente simplificado. Além disso, na biblioteca existem recursos que permitem o recebimento de informações de posicionamento do mouse e de comandos do teclado. Por ser constituído basicamente de um *Applet* Java, o Processing pode facilmente ser integrado a componentes do Swing – biblioteca de interface gráfica (GUI) do Java.

Outra biblioteca para a confecção de desenhos em 2D é o Cairo (CAIRO, 2013), que é *open-source*, disponível nas linguagens C e C++. Ele possui recursos em alto nível para renderização de formas e interação com o usuário, assim como o Processing. Nos aspectos gerais as duas ferramentas são muito semelhantes. A integração do Cairo com a interface gráfica, porém, é dependente na biblioteca externa de GUI utilizada.

Um aspecto importante a notar é que ambas as bibliotecas foram desenvolvidas e otimizadas para terem bom desempenho em máquinas atuais – o que é desejável tendo em vista os requisitos. Na Tabela 8 está presente uma comparação entre as duas bibliotecas.

Escolha da equipe: O Processing foi adotado como solução para desenhos 2D. Deve-se ressaltar que a escolha da biblioteca de desenhos foi feita em conjunto com a escolha de linguagem de programação, tendo em vista a interdependência de ambas. Em razão dos motivos apresentados na próxima subseção – que se constituem de principal argumento para a escolha do Processing – o Java foi adotado. Um ponto interessante do Procesing é a simplicidade de integração a interfaces gráficas (Swing) do Java.

6.3.3.2 Linguagem de programação

Na etapa de avaliação das opções tecnológicas, a escolha de uma boa linguagem de programação que atenda aos requisitos é fundamental. Abaixo está presente uma lista dos aspectos desejáveis da linguagem a ser utilizada neste projeto:

- Deve ser multiplataforma (ao menos compatível com Linux e Windows sem muitas modificações);
- Deve possuir orientação a objetos;
- Deve possuir recursos multiplataforma e de código livre para o desenvolvimento de interfaces gráficas;
- Deve ter a disponibilidade de ferramentas de código livre e multiplataforma para a criação *visual* da interface gráfica, dessa forma agilizando o processo de desenvolvimento;
- Deve possuir recursos, integrados ou em bibliotecas externas de código livre, para o desenvolvimento de desenhos dinâmicos (para a geração do mapa 2D). Os desenhos devem ser facilmente integráveis à interface gráfica.

Abaixo está presente uma descrição de duas linguagens, o C++ e Java, atualmente utilizadas em inúmeras aplicações, e que são potenciais alternativas ao projeto – tendo em vista a experiência e conhecimento dos integrantes a respeito de ambas. A Tabela 9 sumariza os recursos de cada uma.

Java

O Java (JAVA, 2013) é uma linguagem concebida de início como sendo orientada a objetos. A maneira com que é feita a compilação e execução do código permite que, muito facilmente, programas sejam rodados em diferentes plataformas (Linux, Windows, Mac, entre outros). O processo de compilação do código gera os chamados *bytecodes*, que são instruções a serem interpretadas pela *Java Virtual Machine* (JVM). A grande vantagem é que o JVM possui disponibilidade multiplataforma, e a manutenção dele pelos desenvolvedores é frequente.

Há disponibilidade, na API do Java, da biblioteca Swing – que contém recursos completos para a criação de interfaces gráficas (GUI) interativas. Existem ferramentas visuais de código aberto que consideravelmente agilizam o processo de desenvolvimento de interfaces Swing, entre elas o NetBeans (NETBEANS, 2013) e o Eclipse (ECLIPSE, 2013), através de plugins ou extensões.

Para o preenchimento do requisito de confecção de desenhos em 2D com integração à interface gráfica, a biblioteca do Processing (explicada anteriormente na Subseção 6.3.3.1) está disponível nessa linguagem.

C++

O C++ é uma linguagem orientada a objetos, que foi desenvolvida a partir da linguagem C. A compilação de código no C++ deve ser feita especificamente para cada plataforma em que um programa desenvolvido for utilizado. De uma perspectiva prática, certas seções de código frequentemente necessitam de adaptações manuais para cada plataforma e sistema operacional, o que gera retrabalho e gastos de tempo adicionais.

Recursos para desenvolvimento visual de interfaces gráficas estão disponíveis através de bibliotecas e ferramentas externas. O C++ não possui recursos de interface gráfica na própria API e, portanto, deve-se notar que este é um aspecto complicador ao portar programas entre diferentes sistemas.

Para a confecção de desenhos 2D e incorporação dos mesmos à interface gráfica, a biblioteca Cairo (explicada anteriormente na Subseção 6.3.3.1) pode ser utilizada com essa linguagem. A possibilidade de haver integração com a interface, porém, é dependente da biblioteca de GUI utilizada.

Escolha da equipe: O Java foi a linguagem escolhida para o desenvolvimento do *software* da estação base, uma vez que preenche satisfatoriamente os requisitos do projeto. Ressalta-se novamente que a escolha do Java foi feita em conjunto com a escolha da biblioteca do Processing. Notavelmente, no Java há a facilidade em portar, sem adaptações consideráveis na maioria dos casos, programas para diferentes plataformas – processo este que é mais complexo no C++. Com relação ao quesito de desempenho em computadores atuais, a linguagem escolhida é satisfatória, visto que há manutenção constante da implementação das bibliotecas e da máquina virtual do Java pelos desenvolvedores – que buscam, entre outros aspectos, otimizar a linguagem para as tecnologias disponíveis atualmente.

6.3.3.3 Sistema operacional

Estando escolhida a linguagem de programação, o próximo passo é escolher um sistema operacional que seja compatível com ela e que satisfaça os requisitos da seção 6.2.1. Os aspectos desejáveis, portanto, são:

- Deve ser compatível com o Java;

- Deve ser compatível com ferramentas de desenvolvimento *visual* de interface gráfica;
- Deve ser gratuito e de código aberto.

A Tabela 10 apresenta uma comparação entre dois sistemas operacionais, Linux e Windows – sobre os quais a equipe tem considerável experiência e possibilidade de uso nos próprios computadores pessoais.

Escolha da equipe: Ambos os sistemas comparados são compatíveis com o Java e com ferramentas visuais de desenvolvimento de interface gráfica (NetBeans e Eclipse). Porém, o Linux é o único gratuito e de código aberto, e portanto foi o sistema operacional escolhido para o desenvolvimento do projeto da estação base.

Tabela 6: Comparativo entre microcontroladores.

Fonte: Dados obtidos de (CORPORATION, 2013)

uC	STM32F103C6T7A	PIC32MX320F128H	LPC2103
Fabricante	STMicroelectronics	Microchip Technology	NXP Semiconductors
Arquitetura	ARM® Cortex™-M3	MIPS32® M4K™	ARM7
Core	32bits	32-Bit	16/32-Bit
Velocidade	72MHz	80MHz	70MHz
MIPS	90	124.8	63
I2C	1	2	2
PWM	12	5	14
UART	2	2	2
Einterrupt	16	5	13
FLASH	32k	128k	32k
RAM	10k	16k	8k
Adc	10x12b	16x10b	8x10b
JTAG	sim	sim	sim
Custo	\$6.27	\$6.26	\$6.16
uC	MCF52210CAE66	AT32UC3C264C	SIM3C146
Fabricante	Freescale Semiconductor	Atmel	Silicon Laboratories Inc
Arquitetura	Coldfire V2	AVR	ARM® Cortex™-M3
Core	32-Bit	32-Bit	32-Bit
Velocidade	66MHz	66MHz	80MHz
MIPS	75.9	98.34	100
I2C	2	3	2
PWM	4	8	8
UART	3	1	2
Einterrupt	7	7	16
FLASH	64k	64k	64k
RAM	16k	16k	16k
Adc	8x12b	11x12b	28x12b
JTAG	sim	sim	SIM3C146-B-GQ
Custo	\$7.1	\$9.14	\$6.1

Tabela 7: Comparação entre tecnologias de comunicação sem fios.

Característica	802.11g (Wi-Fi)	Rádio Frequência (RF)	Bluetooth
Distância máxima de alcance	50-100 metros	30-100 metros	10 metros
Velocidade de transmissão máxima	54 Mbits/s	2 Mbits/s	1 Mbits/s
Fluxo de dados <i>full-duplex</i>	Sim	Sim	Sim
Possibilidade e simplicidade de uso de TCP	Sim	Não	Não

Tabela 8: Comparação entre Bibliotecas para desenhos 2D.

Característica	Cairo	Processing
Linguagem de programação	C e C++	Java
Integração com interface gráfica	Sim (depende da biblioteca de GUI utilizada)	Sim (com a biblioteca Swing do Java)
Ferramentas de interação com o usuário	Sim	Sim

Tabela 9: Comparação entre linguagens de programação.

Característica	C++	Java
Multiplataforma (Linux e Windows)	Sim (com adaptação)	Sim (sem adaptação)
Orientação a objetos	Sim	Sim
Recursos multi-plataforma e <i>open-source</i> para desenvolvimento de interface gráfica (GUI)	Sim (com bibliotecas externas)	Sim (integrado à API da linguagem)
Ferramentas <i>open-source</i> e multi-plataforma para criação visual de interface gráfica	Sim (ferramentas externas)	Sim (ferramentas externas)
Recursos <i>open-source</i> para desenvolvimento de desenhos dinâmicos, facilmente integráveis à interface gráfica	Sim (biblioteca externa, integração à interface gráfica dependente da GUI utilizada)	Sim (biblioteca externa)

Tabela 10: Comparação entre sistemas operacionais.

Característica	Linux	Windows
Compatível com Java	Sim	Sim
Compatível com ferramentas visuais de desenvolvimento	Sim	Sim
Gratuito e de código aberto	Sim	Não

7 PLANO DO PROJETO

7.1 CRONOGRAMA

O cronograma do projeto está presente em anexo, em um arquivo do OpenProj denominado “Bellator.pod”.

7.2 DELIVERABLES

Na Tabela 11 estão expostos os *deliverables* previstos ao longo do projeto.

Tabela 11: Relação dos entregáveis com seus respectivos responsáveis e prazos

Dia	Auxiliar de Gerenciamento	Deliverables
13/03/2013	Stefan Campana Fuchs	<ol style="list-style-type: none"> 1. Versões iniciais dos diagramas de casos de uso e de classes (estaçao base). 2. Versão inicial do diagrama em blocos (hardware). 3. Explicação inicial de cada bloco (hardware).
27/03/2013	Telmo Friesen	<ol style="list-style-type: none"> 1. Versão inicial do diagrama de casos de uso (software embarcado). 2. Versão inicial do diagrama de fluxo de dados (software embarcado). 3. Versão inicial dos diagramas de estados (sistema de comunicação). 4. Versão inicial da descrição das mensagens e codificações dos comandos (sistema de comunicação). 5. Versão inicial do diagrama elétrico/eletrônico (hardware).

10/04/2013	Ricardo Farah	<ol style="list-style-type: none"> 1. Diagrama de casos de uso e de classes (estação base). 2. Diagrama de casos de uso (software embarcado). 3. Diagrama de fluxo de dados (software embarcado). 4. Diagrama em blocos (hardware). 5. Explicação detalhada de cada bloco (hardware). 6. Diagrama elétrico/eletônico (hardware).
24/04/2013	Stefan Campana Fuchs	<ol style="list-style-type: none"> 1. Projeto da PCB (Printed Circuit Board) (hardware). 2. Lista de componentes para confecção da PCB completa (hardware). 3. Diagramas de estados (sistema de comunicação). 4. Descrição das mensagens e codificações dos comandos (sistema de comunicação). 5. Descrição do uso do Software (Manual do Usuário) 6. Mapa de conexão da PCB com a alimentação e outros elementos (hardware). 7. Guia de montagem (hardware).

8 ORÇAMENTO DETALHADO

⁰Taxas de 60% para importação.

Tabela 12: Preços individuais e totais dos componentes do projeto.

n	quantidade	onde	item	descrição	preço unitario	total	
1	4	digikey	IC REG LDO 5V .95A SOT-223	REGULADOR 5V	\$0,54	\$2,16	
2	3	digikey	IC REG LDO 3.3V .95A SOT-223	REGULADOR 3V3	\$0,54	\$1,62	
3	4	digikey	IC REG LDO 1.8V .95A SOT-223	REGULADOR 1V8	\$0,48	\$1,92	
4	4	digikey	IC BUFF/DVR TRI-ST DUAL 20SOIC	BUFFER P/ PWM	\$0,99	\$3,96	
5	3	digikey	IC ARM7 MCU FLASH 32K 48-LQFP	LPC 2103	\$6,16	\$18,48	
6	3	digikey	IC BUFF/DVR SCHM TRG 6BIT 14SOIC	SCHMITT TRIGGER	\$1,52	\$4,56	
7	25	digikey	RES 47.0K OHM 1/8W 1% 0805	RESISTOR PULL-UP 47K	\$0,01	\$0,23	
8	4	digikey	LED CHIPLED 645NM RED DIFF 0805	LED VERMELHO 1V8 20MA	\$0,09	\$0,36	
9	4	digikey	RES 20K OHM 1/8W 1% 0805 SMD	RESISTOR LED 20K	\$0,04	\$0,16	
10	25	digikey	RES 22.0 OHM 1/8W 1% 0805 SMD	22 HOMS P/ LIMITADOR DE VOLTAGEM	\$0,01	\$0,37	
11	12	digikey	DIODE ZENER DUAL 4.3V SOT-363	DUAL ZENER 4V3	\$0,33	\$4,01	
12	10	digikey	CAP CER 150PF 50V 1% NPO 0805	CAPACITOR 150P FILTRO ENCODERS	\$0,12	\$1,24	
13	10	digikey	RES 332 OHM 1/8W 1% 0805 SMD	RESISTOR 332 FILTRO ENCODERS	\$0,02	\$0,19	
14	10	digikey	CAP CER 33PF 50V 5% NPO 0805	33P CAPACITOR P/ CRISTAL	\$0,05	\$0,53	
15	50	digikey	CAP CER 0.1UF 50V 5% X7R 0805	0.1U PARA MAX3232 E REGULADORES	\$0,06	\$3,16	
16	12	digikey	CAP CER 10UF 10V 10% X5R 0805	10U PARA REGULADOR	\$0,13	\$1,56	
17	4	digikey	IC DRVR/RCVR MULTCH RS232 16SOIC	MAX3232	\$1,65	\$6,60	
18	4	digikey	CRYSTAL 14.7456 MHZ 18PF SMD	14.7456MHZ 10PPM 18pF	\$0,57	\$2,28	
						53,388	
						Frete: \$37,67	
						IOF (\$): \$5,81	
						Imposto (R\$): R\$ 82,61	
						Total (R\$): R\$ 282,15	
1	3	ebay		MPU-6050	Gyro e acelerometro de 3 eixos	\$8,78	\$26,34
1	1	ebay		MPU-6050	Gyro e acelerometro de 3 eixos	\$8,78	\$8,78
							\$35,12
							Frete: \$0,00
							Total (R\$): R\$ 75,34
							IOF (R\$): R\$ 3,56
							Total (R\$): R\$ 78,90
1	1	sparkfun		MPU-6050	Gyro e acelerometro de 3 eixos	\$39,95	\$39,95
							39,95
							Frete: \$37,75
							IOF (\$): \$2,79
							Imposto (R\$): R\$ 106,98
							Total (R\$): R\$ 269,87
1	3	24 de maio		barra de pinos		R\$ 0,55	R\$ 1,65
2	10	24 de maio		resistor		R\$ 0,05	R\$ 0,50
3	10	24 de maio		resistor		R\$ 0,05	R\$ 0,50
4	2	24 de maio		Max 3232		R\$ 7,70	R\$ 15,40
5	10	24 de maio		capacitor ceramico		R\$ 0,10	R\$ 1,00
6	10	24 de maio		diodo zenner		R\$ 0,25	R\$ 2,50
7	2	24 de maio		buffer 74hc244		R\$ 1,30	R\$ 2,60
7	3	24 de maio		Max 232 SMD		R\$ 3,00	R\$ 9,00
8	3	24 de maio		power jack		R\$ 0,80	R\$ 2,40
9	2	24 de maio		diodo		R\$ 0,20	R\$ 0,40
10	4	24 de maio		conector serial		R\$ 1,86	R\$ 7,44
11	1	24 de maio		conectores para fios		R\$ 1,50	R\$ 1,50
12	2	24 de maio		jumper		R\$ 0,76	R\$ 1,52
13	2	24 de maio		pushbutton		R\$ 0,10	R\$ 0,20
							Total (R\$): R\$ 46,61
13	1	24 de maio		pendrive 4Gb		R\$ 15,00	R\$ 15,00
							Total (R\$): R\$ 15,00
1	4	stik		pcb		R\$ 53,47	R\$ 235,25
							235,25
							Frete: R\$ 30,00
							Total (R\$): R\$ 265,25
1	1	mercado livre		webcam		R\$ 34,99	R\$ 34,99
							Total (R\$): R\$ 34,99
							Total (R\$): R\$ 899,28

9 MODELAGEM UML

9.1 REQUISITOS FUNCIONAIS

1. A estação base deve mostrar na interface gráfica um mapa 2D (atualizado automaticamente) representando o robô e os obstáculos detectados por ele. Representado pelo requisito funcional: “**Estação base mostra mapa 2D do robô e dos obstáculos detectados – RF1**”.
2. O usuário pode salvar o mapa 2D no disco rígido. Representado pelo requisito funcional: “**O usuário pode salvar o mapa – RF2**”.
3. O usuário pode carregar o mapa 2D do disco rígido. Representado pelo requisito funcional: “**O usuário pode carregar o mapa – RF3**”.
4. A estação base deve mostrar na interface gráfica a imagem captada pela *webcam* do robô. Representado pelo requisito funcional: “**Estação base mostra a imagem captada pela webcam – RF4**”.
5. O usuário pode movimentar o robô, controlando a velocidade de suas rodas remotamente pelo teclado da estação base. Representado pelo requisito funcional: “**O usuário pode movimentar o robô – RF5**”.
6. A estação base deve ser capaz de estabelecer conexão com o robô, informando o usuário caso a conexão ocorra com sucesso ou não. Representado pelo requisito funcional “**O usuário pode estabelecer a conexão entre o robô e a estação base – RF6**”.

9.2 REQUISITOS NÃO FUNCIONAIS

1. A imagem transmitida pela câmera do robô deve ser colorida. Representado pelo requisito não funcional: “**O robô deve enviar vídeo em imagem colorida para a estação base - RNF1**”.
2. O robô deve transmitir as imagens de sua câmera em tempo real. Representado pelo requisito não funcional: “**O robô deve transmitir os dados de vídeo captados pela**

câmera em tempo real - RNF2”.

9.3 CASOS DE USO IDENTIFICADOS

1. Visualização de mapa 2D na interface gráfica segundo os dados lidos dos sensores do robô. Representado pelo caso de uso: “**Mostrar mapa - UC1**”.
2. Gravação do mapa em um arquivo no disco rígido. Representado pelo caso de uso: “**Salvar mapa - UC2**”.
3. Leitura do mapa de um arquivo do disco rígido. Representado pelo caso de uso: “**Carregar mapa - UC3**”.
4. Leitura de informações dos sensores do robô. Representado pelo caso de uso: “**Ler amostras dos sensores - UC4**”.
5. Visualização de imagens da *webcam* do robô. Representado pelo caso de uso: “**Visualizar imagens da câmera - UC5**”.
6. Alteração pelo usuário da velocidade das rodas do robô. Representado pelo caso de uso: “**Alterar velocidade das rodas - UC6**”.
7. Solicitação de estabelecimento de conexão com o robô. “**Estabelecer conexão - UC7**”.
8. Consulta à documentação do robô pelo usuário. Representado pelo caso de uso: “**Consultar documentação do robô - UC8**”.

Foram produzidos três Diagramas de Casos de Uso (Figuras 6, 7 e 8) com base nos casos de uso apresentados. O primeiro diagrama representa o *software* da estação base, e o segundo e o terceiro representam o sistema embarcado (TS-7260 e placa de baixo nível, respectivamente).

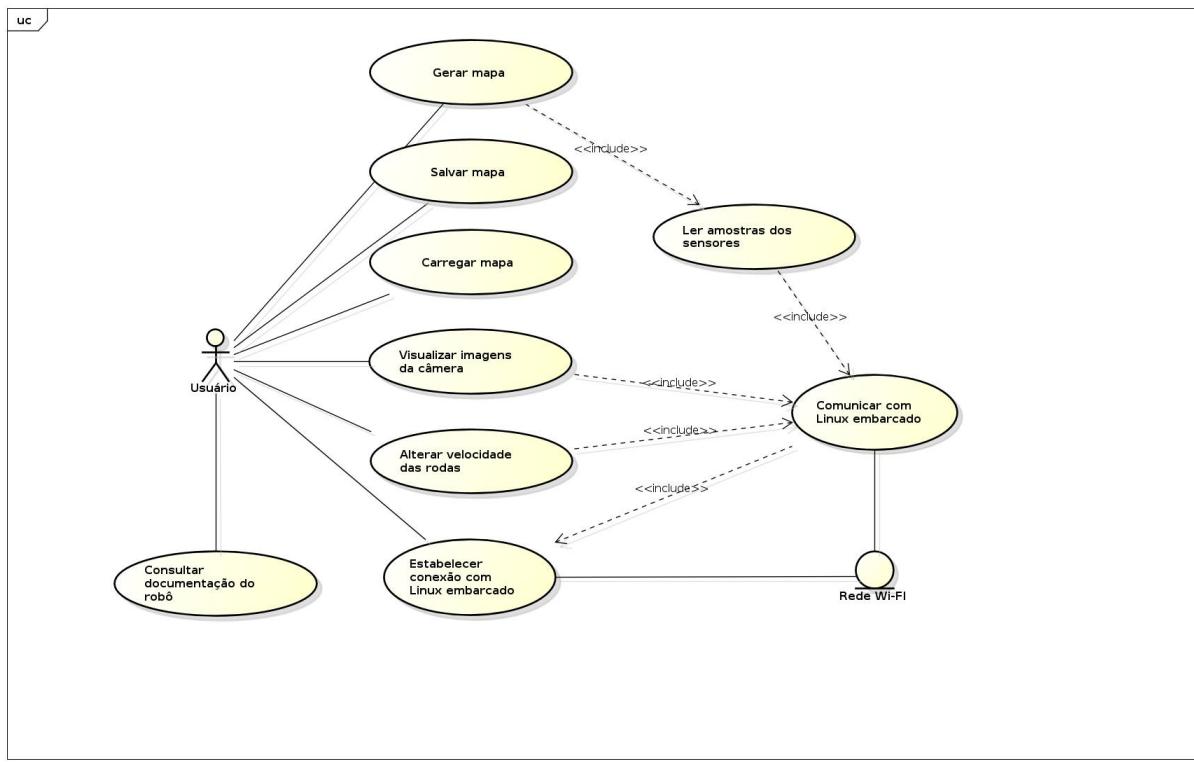


Figura 6: Diagrama de casos de uso do *software* da estação base.

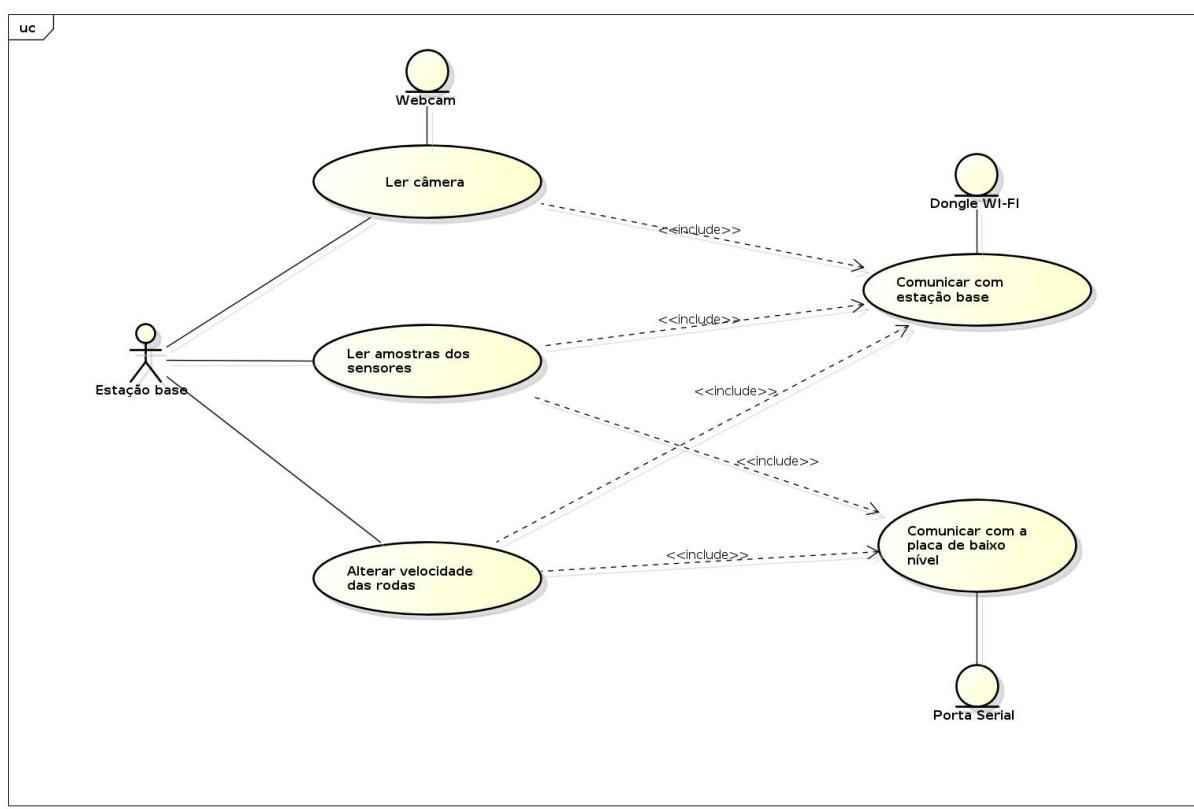


Figura 7: Diagrama de casos de uso do *software* para a placa TS-7260.

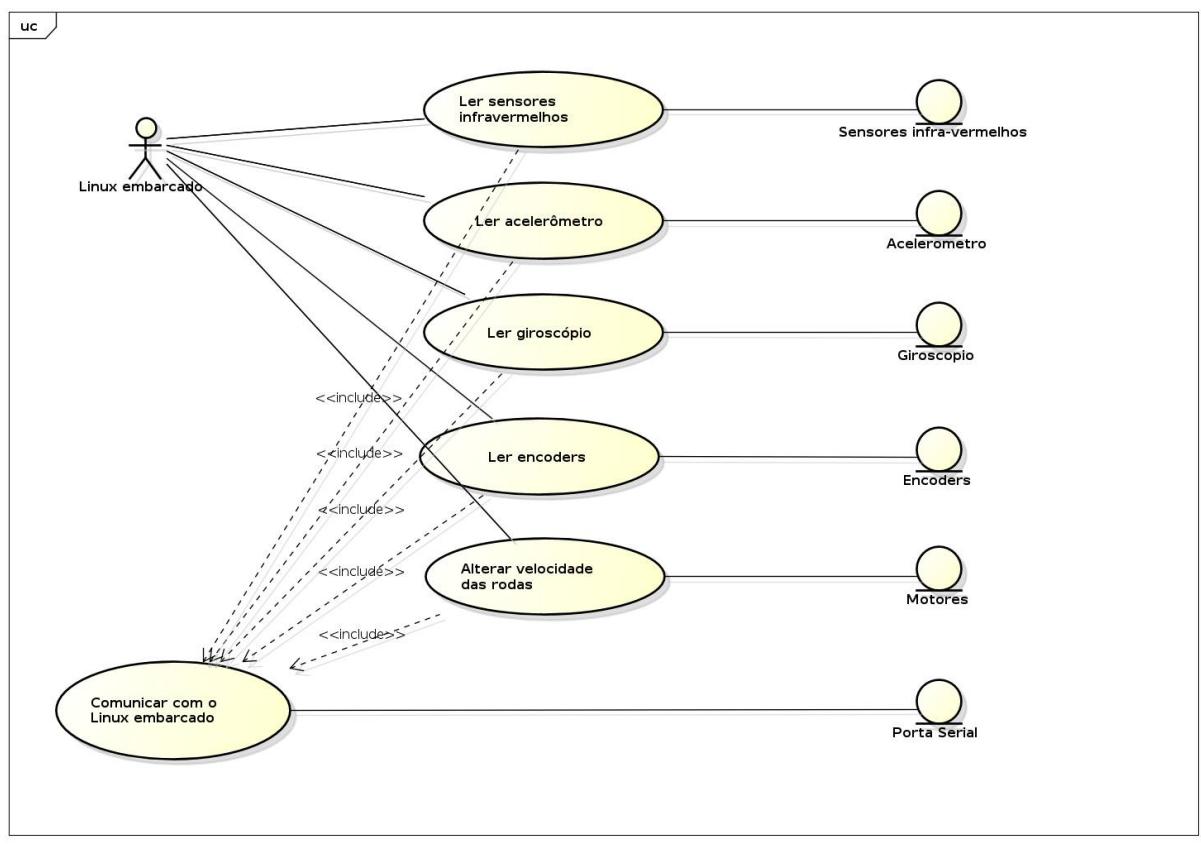


Figura 8: Diagrama de casos de uso do *software* para a placa de baixo nível.

9.4 DIAGRAMA DE CLASSES DA ESTAÇÃO BASE

A Figura 9 mostra o diagrama de classes da estação base.

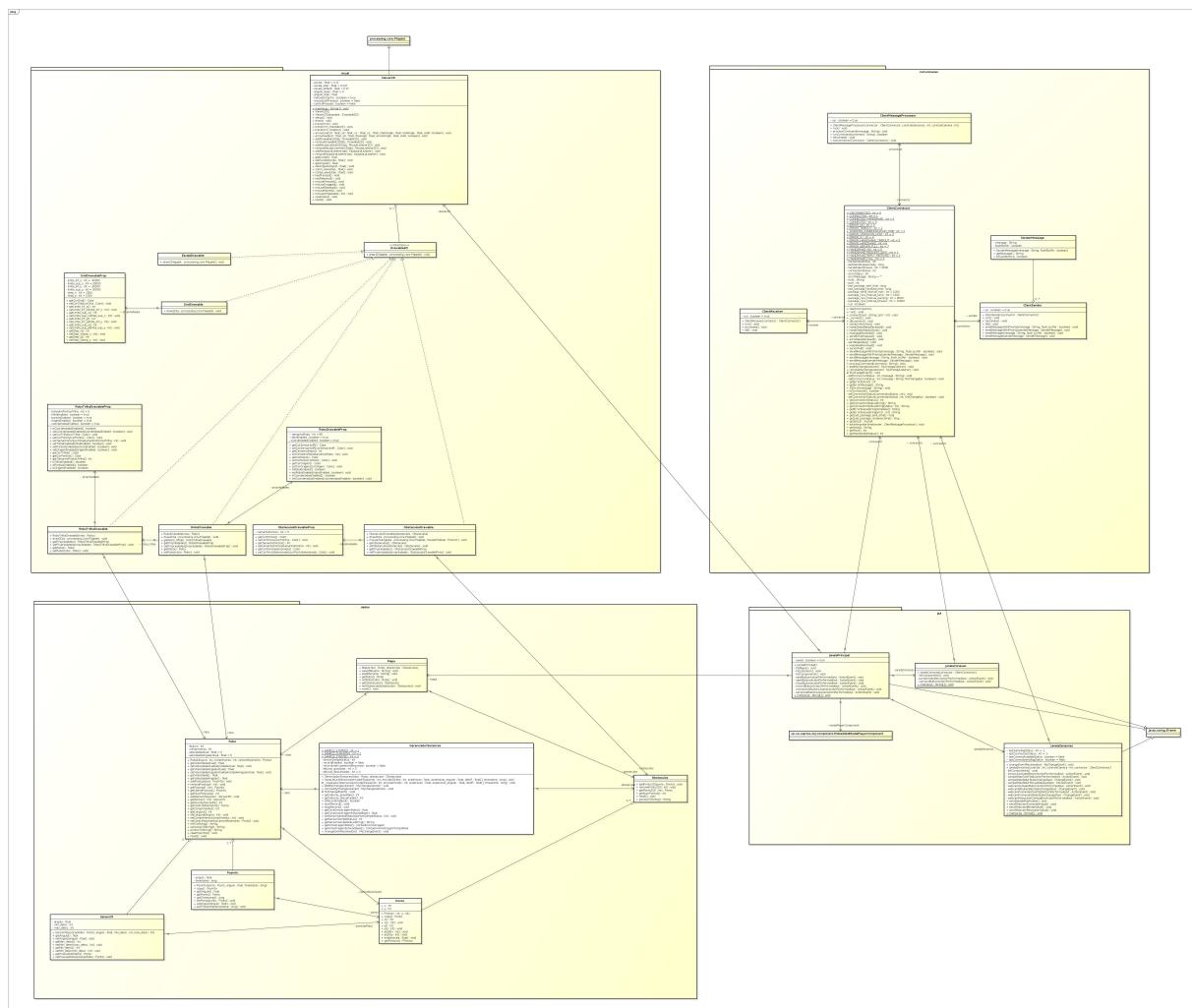


Figura 9: Diagrama de classes da estação base

9.4.1 Descrição das classes da estação base

O *software* da estação base do robô foi dividido em cinco pacotes: *visual*, *dados*, *comunicacao*, e *gui*. A seguir há uma descrição de cada pacote e das suas respectivas classes.

9.4.2 Pacote *visual*

Este pacote consiste de toda a parte visual da estação base e conta com as seguintes classes: Viewer2D, Drawable2D, EscalaDrawable, RoboDrawable, RoboTrilhaDrawable, ObstaclesDrawable, EscalaDrawableProp, RoboDrawableProp, RoboTrilhaDrawableProp e ObstaclesDrawableProp. Na Tabela 13 estão descritas as classes deste pacote.

Tabela 13: Pacote *visual*

Classe	Descrição
Viewer2D	Responsável por exibir os objetos Drawable2D. Possui recursos de pan, zoom e rotate.
Drawable2D	Representa genericamente objetos 2D que podem ser desenhados em um Viewer2D.
EscalaDrawable	Responsável por desenhar uma escala gráfica no mapa.
RoboDrawable	Responsável por desenhar o robô no mapa.
RoboTrilhaDrawable	Responsável por desenhar a trilha percorrida pelo robô no mapa.
ObstaculosDrawable	Responsável por desenhar os pontos de cada obstáculo no mapa.
EscalaDrawableProp	Contém as propriedades visuais de desenho da escala.
RoboDrawableProp	Contém as propriedades visuais de desenho do robô
RoboTrilhaDrawableProp	Contém as propriedades visuais de desenho da trilha do robô.
ObstaculosDrawableProp	Contém as propriedades visuais de desenho dos obstáculos.

9.4.3 Pacote *dados*

Este pacote consiste de toda a parte da estação base que processa e armazena as informações essenciais do robô e do mapa. Conta com as seguintes classes: Mapa, Obstaculos, Robo, ControleSensores, Posinfo, SensorIR e Ponto. Na Tabela 14 estão descritas as classes deste pacote.

9.4.4 Pacote *comunicacao*

Este pacote consiste em toda a parte de comunicação da estação base com o robô e conta com as seguintes classes: ClientMessageProcessor ClientConnection, ClientReceiver, ClientSender e Message. Na Tabela 15 estão descritas as classes deste pacote.

É importante ressaltar que o protocolo TCP requer obrigatoriamente a especificação de um cliente e de um servidor para estabelecimento de uma conexão. Nas implementações desse protocolo em diversas linguagens (como Java e C++) existem tipos de *socket* distintos para cliente e servidor. Na criação de um *socket* de servidor, há obrigatoriamente a atribuição

Tabela 14: Pacote *dados*

Classe	Descrição
Mapa	Responsável por representar o mapa. Armazena as informações essenciais do robô e dos obstáculos detectados.
Obstaculos	Responsável por conter os obstáculos detectados pelo robô.
Robo	Responsável por representar o robô, este contém largura, comprimento e centro de movimento (ponto central entre as duas rodas).
GerenciadorSensores	Responsável por atualizar a posição do robô e dos pontos que representam os obstáculos, de acordo com as leituras feitas pelos sensores.
Posinfo	Responsável por conter as informações de uma posição do robô.
SensorIR	Responsável por representar um sensor IR do robô.
Ponto	Representa um ponto de coordenadas cartesianas (x,y).
GerenciadorCamera	Responsável por gerenciar o status da câmera e o recebimento de imagens.

de uma porta de escuta, na qual o servidor aguarda que um cliente efetue uma requisição de conexão. Não é possível, ao menos nas implementações atuais do TCP, estabelecer conexão entre dois *sockets* de cliente ou entre dois *sockets* de servidor. Como neste projeto, o robô proverá serviços à estação base (envio de imagens da câmera, envio de leituras de sensores, além de prover a possibilidade de comando dos motores) o robô foi escolhido como servidor e a estação base como cliente. Enfatiza-se que o paradigma cliente-servidor não implica de forma alguma que a comunicação seja unidirecional. Pelo contrário, o envio de pacotes pode ser feito bidirecionalmente após uma conexão TCP ser estabelecida, sem nenhuma restrição quanto a isso.

9.4.5 Pacote *gui*

Este pacote consiste em toda a interface gráfica do sistema e conta com as seguintes classes: JanelaConexao, JanelaPrincipal e JanelaSensores. Na Tabela 16 estão descritas as classes deste pacote.

Tabela 15: Pacote *comunicacao*

Classe	Descrição
ClientMessageProcessor	Thread responsável pelo processamento de mensagens recebidas de um host de conexão.
ClientConnector	Thread responsável por efetuar a gerência da conexão do cliente (estaçao base) com o servidor (robô).
ClientReceiver	Thread responsável por receber mensagens de um host de uma conexão.
ClientSender	Thread responsável por enviar mensagens ao host de uma conexão.
SendMessage	Contém uma mensagem que pode ser enviada por um Sender.

Tabela 16: Pacote *gui*

Classe	Descrição
JanelaConexao	Janela com as informações e configurações da conexão com o Bellator.
JanelaPrincipal	Janela principal da interface gráfica da estação base.
JanelaSensores	Janela de configuração dos sensores.

9.5 DIAGRAMA DE CLASSES DO SISTEMA EMBARCADO

A Figura 10 mostra o diagrama de classes do sistema embarcado (placa TS-7260).

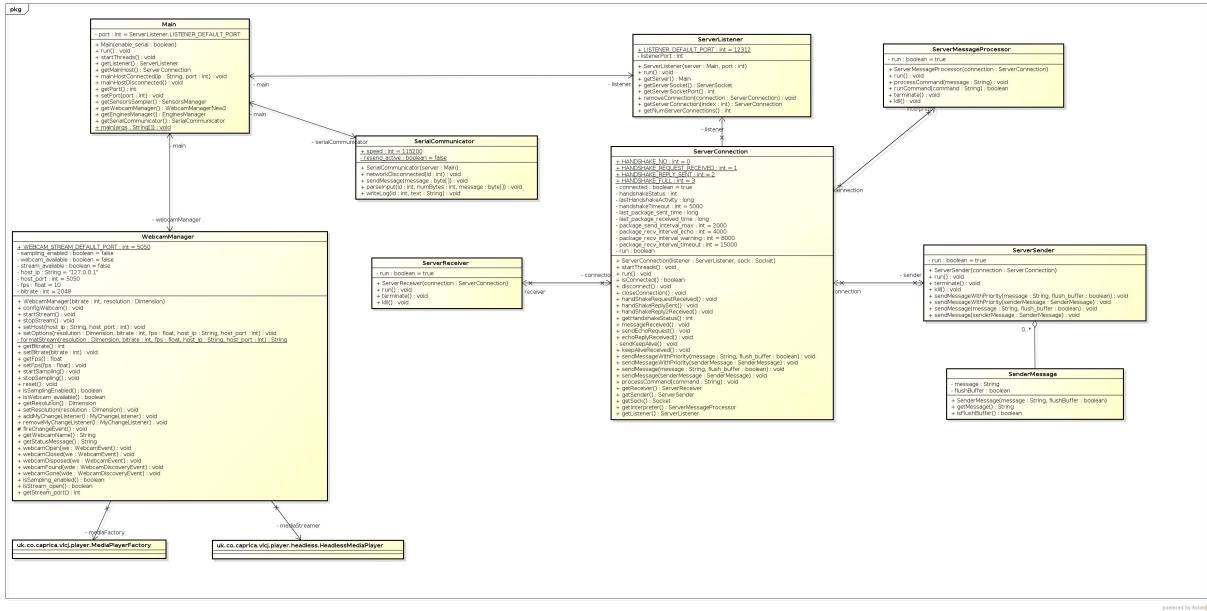


Figura 10: Diagrama de classes do sistema embarcado (placa TS-7260).

Tabela 17: Descrição das classes do sistema embarcado (placa TS-7260)

Classe	Descrição
Main	Classe principal do robô.
SensorsSampler	Thread responsável por requisitar amostras dos sensores da placa de baixo nível em intervalos de tempo previamente programados.
ServerMessageProcessor	Thread responsável por realizar o processamento de mensagens recebidas de um host de conexão.
ServerListener	Thread responsável por escutar requisições de conexão.
ServerSender	Thread responsável por enviar mensagens ao host de uma conexão.
SenderMessage	Contém uma mensagem que pode ser enviada por um Sender.
ServerReceiver	Thread responsável por receber mensagens de um host de uma conexão.
SerialCommunicator	Responsável por gerenciar a comunicação via porta serial entre a TS-7260 e a LPC2103.
WebcamManager	Responsável por gerenciar a abertura e o fechamento da stream de imagens da webcam, além de configurar opções como resolução e taxa de frames.

9.6 PROTOCOLO DE COMUNICAÇÃO

Esta seção detalha o protocolo de comunicação estabelecido entre a estação base, a placa TS-7260 (sistema com linux embarcado) e a placa LPC2103 (sistema embarcado de baixo nível).

O protocolo desenvolvido para a comunicação (via Wi-Fi) entre a estação base e a TS-7260 visa utilizar o TCP como camada de transporte. Como foi explicitado anteriormente na seção 9.4.4, o robô foi escolhido como servidor da conexão, e a estação base como cliente. Para haver confirmação da conexão entre os dois, optou-se por criar um protocolo de *handshake* semelhante ao existente no TCP, de 3 passos: requisição, resposta, confirmação. A requisição é feita pelo cliente no início da conexão, a resposta é dada pelo servidor em seguida. Posteriormente, o cliente envia uma mensagem de confirmação, e a conexão é totalmente estabelecida. O uso do *handshake* contribui para tanto a estação base como o sistema embarcado confirmarem que estão conectados um ao outro e não a um servidor/cliente qualquer.

Para possibilitar que o tráfego de mensagens possa ser feito de forma rápida, reduzindo atrasos, o envio e recebimento de mensagens é feito de forma assíncrona. Essa escolha foi feita tendo em vista que, em uma comunicação totalmente síncrona, um programa (ou thread) é bloqueado ao chamar uma função de recebimento ou envio, até que efetivamente seja completa a transação. Supondo que só houvesse uma thread gerenciando a conexão, o programa não poderia enviar ou receber dados ao mesmo tempo em *full-duplex*, mas somente *half-duplex* (somente enviar ou somente receber).

A solução desenvolvida para possibilitar a comunicação assíncrona foi o uso de 4 threads (tanto na estação base quanto no sistema embarcado) para gerenciar os diversos aspectos envolvidos nela. A primeira *thread* (a gerenciadora principal de conexão) é a responsável por estabelecer e manter a conexão, além de gerenciar os potenciais erros que possam ocorrer (tempo excessivo sem comunicação e fechamento de socket). Para a estação base, o diagrama de estados dessa *thread* está representado na Figura 11, e para o sistema embarcado na Figura 12.

A segunda *thread* tem a função de gerenciar o envio de mensagens. O programa principal, ao necessitar enviar uma mensagem, faz uma requisição a essa *thread* que insere a mensagem em uma fila de envio. O programa principal não fica bloqueado, dessa forma, pois não necessita aguardar a mensagem ser completamente enviada, podendo efetuar outras tarefas. O diagrama de estados dessa *thread* está presente na Figura 13.

A terceira *thread* gerencia o recebimento de mensagens. Seu funcionamento é rela-

tivamente simples: ela possui um loop, no qual aguarda até alguma mensagem ser recebida. Quando ocorre o recebimento de alguma mensagem, ela é encaminhada para a quarta *thread* (cuja explicação está a seguir) que processa o conteúdo dela e executa as operações que são necessárias para cada tipo de mensagem. Dessa forma, novas mensagens podem ser recebidas rapidamente, pois o receptor não fica bloqueado realizando o processamento das informações recebidas. O diagrama de estados dessa terceira *thread* está presente na Figura 14.

A quarta *thread*, como já exposto, é a responsável por processar mensagens recebidas e realizar as operações que são necessárias para cada tipo de mensagem, o que depende da codificação exposta na seção 9.6.1. Ela possui uma fila, na qual são inseridas as mensagens a serem processadas. Dessa forma a *thread* receptora não necessita ficar bloqueada aguardando o término do processamento. O diagrama de estados dessa quarta *thread* está presente na Figura 15.

9.6.1 Codificação das mensagens

- Mensagens do TS-7260 para o LPC2103 (via porta serial)

- SYNC (0xA0)

Quando o microcontrolador LPC2103 recebe esta mensagem, responde com as leituras mais recentes dos encoders, de cada sensor de distância, do acelerômetro e do giroscópio (enviando uma mensagem SENSORS, explicada abaixo).

- ENGINES (0xB0)

(byte) *vel_roda_esquerda*

(byte) *vel_roda_direita*

(byte) *CHECKSUM_H*

(byte) *CHECKSUM_L*

Ao receber este comando, o microcontrolador utiliza os valores para definir o nível de PWM para as rodas do robô. Os valores de velocidade são representados por um byte cada, nos quais o bit mais significativo indica o sentido de rotação da roda (1 para frente e 0 para trás) e os restantes a intensidade do PWM.

Os bytes de checksum são utilizados para verificar se não há dados corrompidos. Os bytes da mensagem são somados (módulo 65536) e o resultado é atribuído aos bytes (high e low) de checksum.

- Mensagens do LPC2103 para a TS-7260 (via porta serial)

- ENGINES_ACK (0xB1)

(byte) *vel_roda_esquerda*

(byte) *vel_roda_direita*

(byte) *CHECKSUM_H*

(byte) *CHECKSUM_L*

Esta mensagem deve ser enviada toda vez que um comando de mudança de velocidade (ENGINES) for recebido na placa de baixo nível. A mensagem é usada no Linux embarcado para verificar se o comando foi corretamente recebido. Caso uma confirmação não seja recebida em certo intervalo de tempo, outro comando ENGINES é enviado para a placa de baixo nível.

O checksum tem função idêntica ao que já foi explicitado na mensagem ENGINES.

- SENSORS (0xC0)

(byte) *encoder_esq_H*, (byte) *encoder_esq_L*,

(byte) *encoder_dir_H*, (byte) *encoder_dir_L*,

(byte) *IR1*, (byte) *IR2*, (byte) *IR3*, (byte) *IR4*, (byte) *IR5*,

(byte) *AX_H*, (byte) *AX_L*,

(byte) *AY_H*, (byte) *AY_L*,

(byte) *AZ_H*, (byte) *AZ_L*,

(byte) *GX_H*, (byte) *GX_L*,

(byte) *GY_H*, (byte) *GY_L*,

(byte) *GZ_H*, (byte) *GZ_L*,

(byte) *TIMESTAMP_H*, (byte) *TIMESTAMP_L*

(byte) *CHECKSUM_H*

(byte) *CHECKSUM_L*

Representa a leitura de todos os sensores (encoders, infra-vermelhos, acelerômetro e giroscópio).

Os 4 primeiros bytes são os valores das leituras dos encoders esquerdo e direito (cada um com um byte alto e um baixo). Os valores das leituras dos encoders representam a diferença entre a contagem atual a contagem anterior.

Nos próximos 5 bytes, as leituras do sensores ópticos são enviadas em sequência. As distâncias que os sensores ópticos são capazes de mensurar são divididos em valores discretos de 0 a 255 (MARIN et al., 2012).

Após isso, os 12 bytes que se seguem representam as leituras do acelerômetro e do giroscópio. Os bytes que começam com ‘A’ representam a leitura de cada um dos eixos do acelerômetro. Aqueles que começam com ‘G’ representam a leitura de

cada um dos eixos do giroscópio.

O timestamp (valor alto e baixo) é um contador de 16 bits que é incrementado entre cada amostra e zera automaticamente depois que chega ao valor máximo (65535), usado para determinar o instante em que foi feita a leitura dos dados. Como a amostragem dos sensores na placa de baixo nível será efetuada em intervalos fixos, a informação do contador do timestamp pode ser utilizada para obter informações de tempo de cada amostra.

O checksum é tem função idêntica ao que já foi explicitado na mensagem ENGINES.

- Mensagens bidirecionais entre estação base e TS-7260 (via WI-Fi):

- ECHO_REQUEST (0x01)**

(byte) *END_CMD*

Requisição de ping.

- ECHO_REPLY (0x02)**

(byte) *END_CMD*

Resposta de ping.

- DISCONNECT (0x0F)**

Solicitação de desconexão.

- Mensagens da estação base para a TS-7260 (via Wi-Fi):

- HANDSHAKE_REQUEST (0x10)**

Solicitação de handshake.

- HANDSHAKE_CONFIRMATION (0x12)**

Confirmação de handshake.

- SENSORS_START (0x20)**

Solicitação de início da amostragem dos sensores.

- SENSORS_STOP (0x21)**

Solicitação de parada da amostragem dos sensores.

- SENSORS_RATE (0x22)**

(float) *Nova taxa de amostragem (comandos SYNC por segundo)*

Solicitação de mudança da taxa de envio de comandos SYNC da TS para a placa de baixo nível.

- WEBCAM_START (0x30)

Solicitação de início da amostragem da webcam.

- WEBCAM_STOP (0x31)

Solicitação de parada da amostragem da webcam.

- WEBCAM_RATE (0x32)

(float) Nova taxa de quadros

Solicitação de mudança da taxa de quadros da webcam.

- WEBCAM_RESOLUTION (0x33)

(int) Largura em pixels

(int) Altura em pixels

Solicitação de mudança da resolução da webcam.

- ENGINES (0xB0)

(float) vel_roda_esquerda

(float) vel_roda_direita

Solicitação de mudança da velocidade dos motores. Para cada roda há um valor de -1 até 1, sendo que -1 é a máxima velocidade para trás, 1 a máxima velocidade para frente e 0 é parada da roda.

- Mensagens da TS-7260 para a estação base (via Wi-Fi):

- HANDSHAKE_REPLY (0x11)

Resposta de handshake.

- SENSORS (0xC0)

(byte) encoder1_H, (byte) encoder1_L,

(byte) encoder2_H, (byte) encoder2_L,

(byte) IR1, (byte) IR2, (byte) IR3, (byte) IR4, (byte) IR5,

(byte) AX_H, (byte) AX_L,

(byte) AY_H, (byte) AY_L,

(byte) AZ_H, (byte) AZ_L,

(byte) GX_H, (byte) GX_L,

(byte) GY_H, (byte) GY_L,

(byte) GZ_H, (byte) GZ_L,

(byte) TIMESTAMP_H, (byte) TIMESTAMP_L

Possui a mesma funcionalidade e parâmetros que a mensagem SENSORS enviada da LPC2103 para a TS-7260, com exceção do timestamp, que é trocado por um

timestamp UNIX em milissegundos (que representa o horário absoluto em que a amostra foi obtida na placa de baixo nível). Essa informação de tempo é utilizada pela estação base para efetuar os cálculos de posicionamento do robô.

- **SENSORS_STATUS (0xC1)**

(boolean) Status da amostragem [on - off]

(float) Taxa de amostragem

Informações de status dos sensores. Usado na interface gráfica para confirmar o recebimento de comandos de mudança de taxa de amostragem e início/parada da amostragem.

- **WEBCAM_STATUS (0x34)**

(float) Taxa de quadros

(int) Largura em pixels

(int) Altura em pixels

(boolean) Status da stream [on - off]

(int) Porta da stream

Informações de status da webcam. Usado na interface gráfica para confirmar o recebimento de comandos relativos à webcam, e para que a estação base tenha conhecimento do status da stream da webcam.

- **ENGINES_STATUS (0xB1)**

(byte) vel_roda_esquerda

(byte) vel_roda_direita

Informações sobre as velocidades programadas dos motores. Usado na interface gráfica para confirmar o recebimento de comandos de movimentação efetuados pelo usuário.

9.6.2 Diagramas de estados

Nesta seção estão expostos os diagramas de estados do protocolo de comunicação.

Um aspecto importante a ressaltar é que, nas *threads* de envio (Figura 13) e de processamento de mensagens (Figura 15), pode haver adição assíncrona de elementos na fila. Ou seja, quando é feita a verificação do número de elementos presentes na fila (como representado nos diagramas), tem-se em vista que elementos podem ter sido adicionados a qualquer instante. Obviamente, no ponto de vista da implementação, existem as seções críticas que devem ser devidamente gerenciadas para evitar condições de disputa e outros problemas de concorrência. Porém, as seções críticas se resumem aos acessos à fila somente, o que reduz consideravelmente

a complexidade do processo.

Na Figura 16 está explicitado o diagrama de estados da *thread* responsável por gerenciar o envio de comandos de velocidade de motores (ENGINES) para a placa de baixo nível. Ela inicialmente aguarda que um comando ENGINES chegue da estação base, e quando isso ocorre, é enviado via serial o comando para a placa de baixo nível. A *thread* aguarda certo tempo para receber um ENGINES_ACK da placa de baixo nível para confirmar que o comando foi corretamente recebido. Caso isso não ocorra dentro do tempo estabelecido (por exemplo, se houver perda de pacotes), outro comando ENGINES é enviado. Isso ocorre até que um ENGINES_ACK correto seja recebido da placa de baixo nível.

Na Figura 17 está exposto o diagrama de estados da *thread* do Linux embarcado que é responsável por realizar a amostragem dos sensores em intervalos fixos de tempo. Ela realiza a amostragem enviando periodicamente – quando programada – comandos SYNC (vide seção 9.6.1) para a placa de baixo nível. Vale ressaltar que para melhor explicar este processo, na Figuras 22 e 23 da próxima seção está exposto um diagrama de sequência que demonstra a amostragem dos sensores.

Nas Figuras 18 e 19 estão presentes os diagramas de estados da captura e recebimento de imagens da webcam. Foi utilizada a biblioteca externa *libVLC* (VIDEOLAN, 2013) – a componente de baixo nível do *player* de mídia VLC – tanto na estação base como no Linux embarcado para efetuar o processo de transmissão e visualização de imagens. No Linux embarcado, quando um comando de início de webcam é dado pelo usuário, uma *stream* HTTP de imagens é aberta pela *libVLC*, e a estação base é posteriormente notificada sobre o fato. Na estação base, quando ocorre a notificação de que a *stream* foi aberta, a componente de *player* da *libVLC* da janela principal é ativada (conectando dessa forma, na *stream* HTTP de imagens).

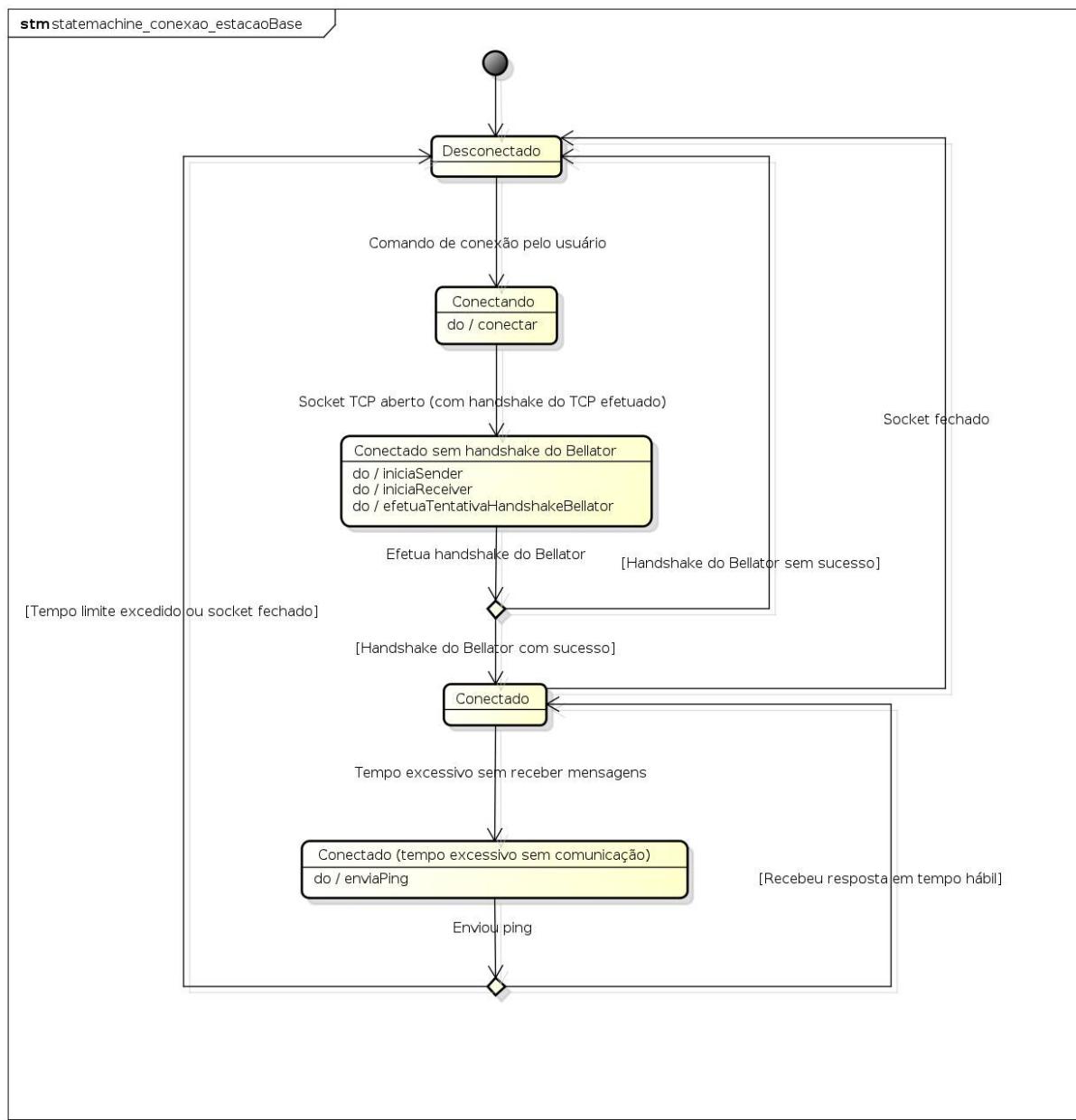


Figura 11: Diagrama estados da *thread* principal da conexão da estação base.

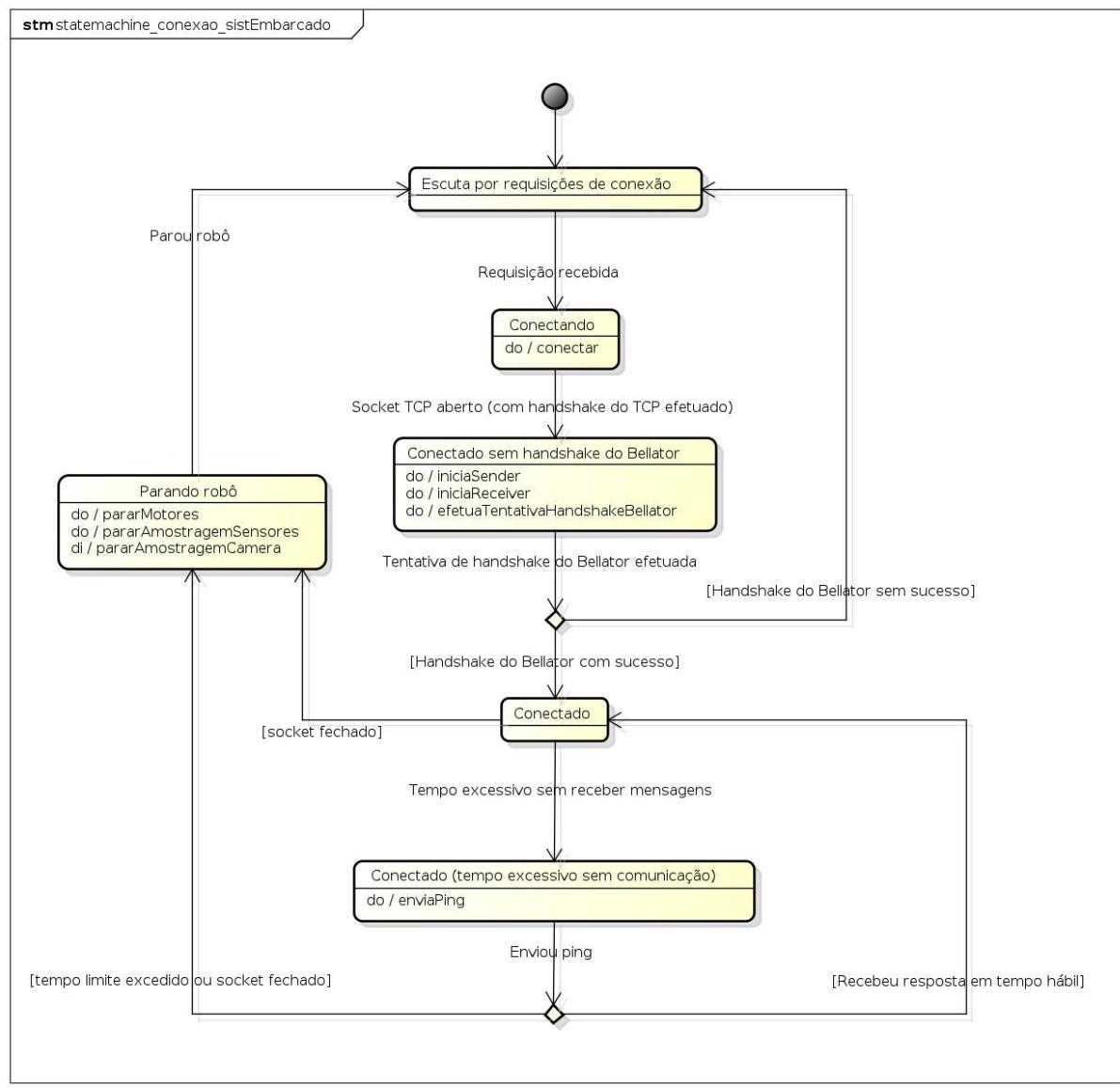


Figura 12: Diagrama de estados da *thread* principal da conexão do linux embarcado.

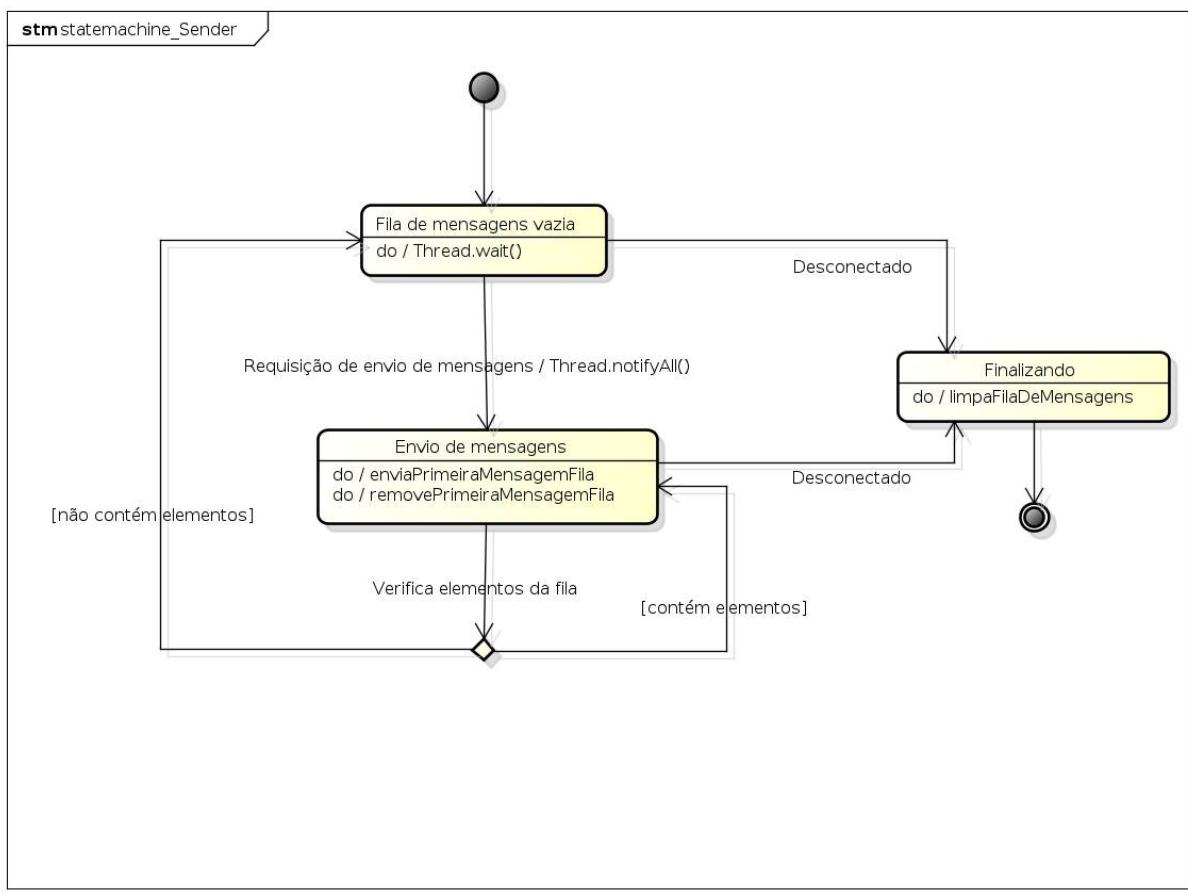
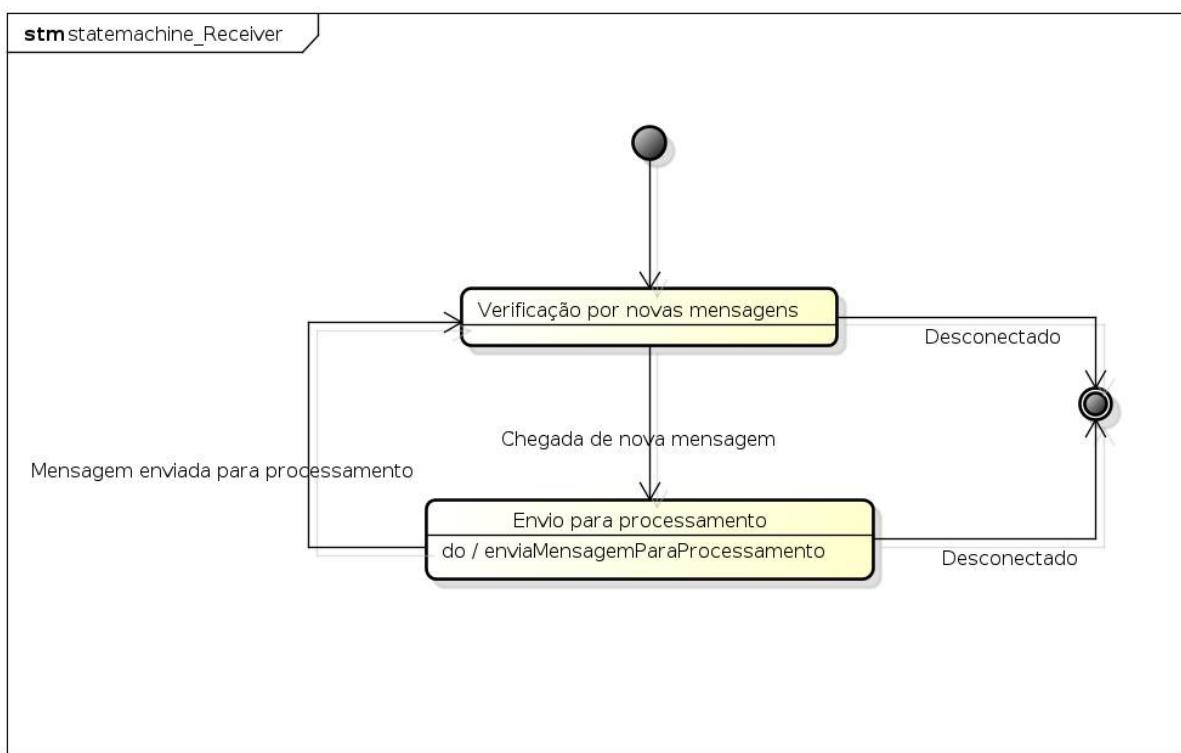
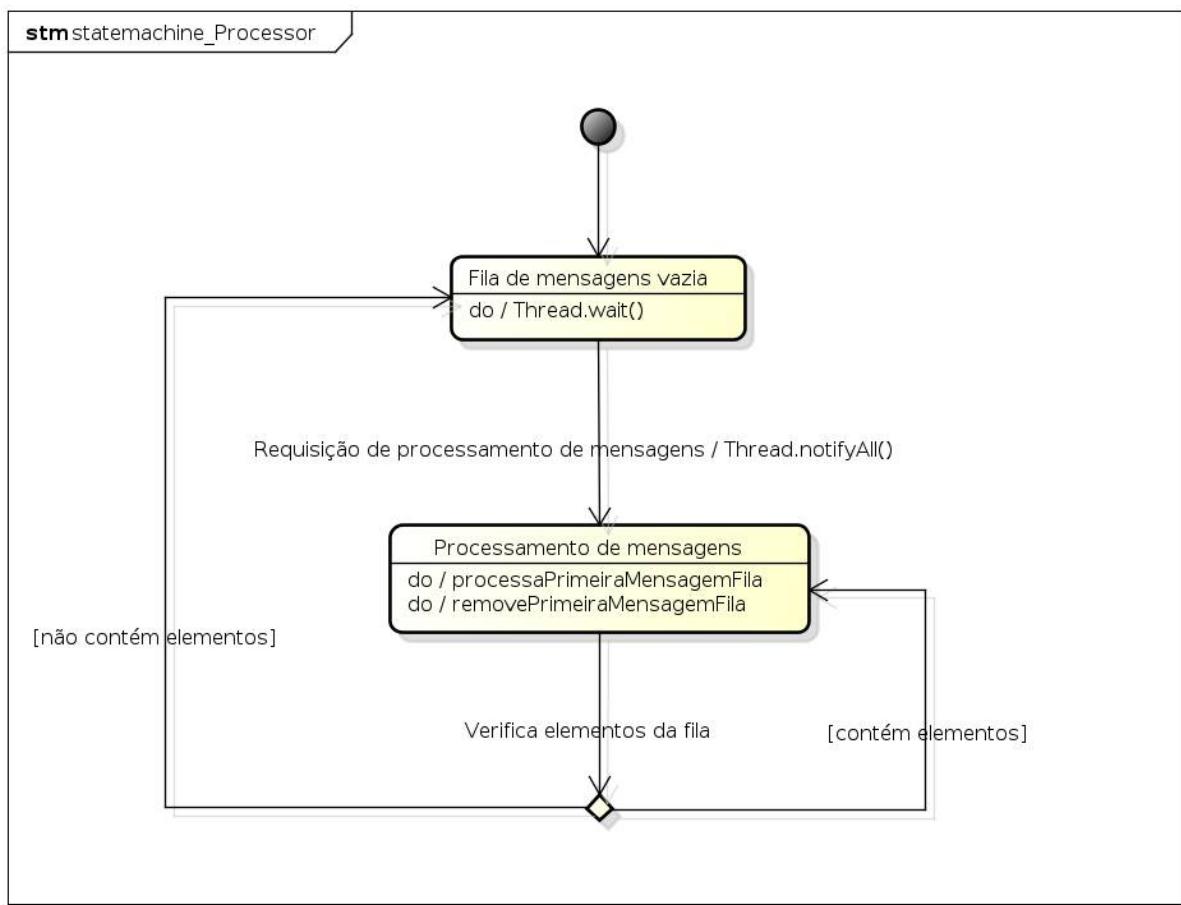


Figura 13: Diagrama de estados da *thread* que envia mensagens (igual para estação base e linux embarcado).



powered by Astah

Figura 14: Diagrama de estados da *thread* receptora de mensagens (igual para estação base e linux embarcado).



powered by Astah

Figura 15: Diagrama de estados da *thread* que processa mensagens recebidas (igual para estação base e linux embarcado).

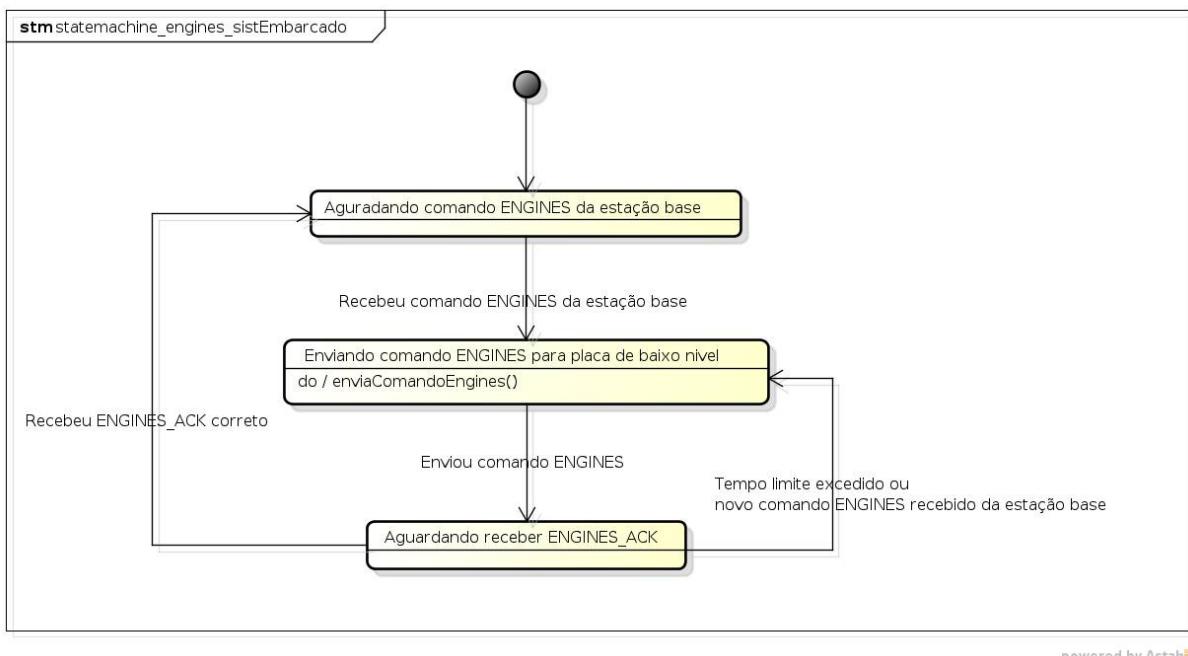


Figura 16: Diagrama de estados da *thread* responsável por gerenciar os comandos dos motores (linux embarcado).

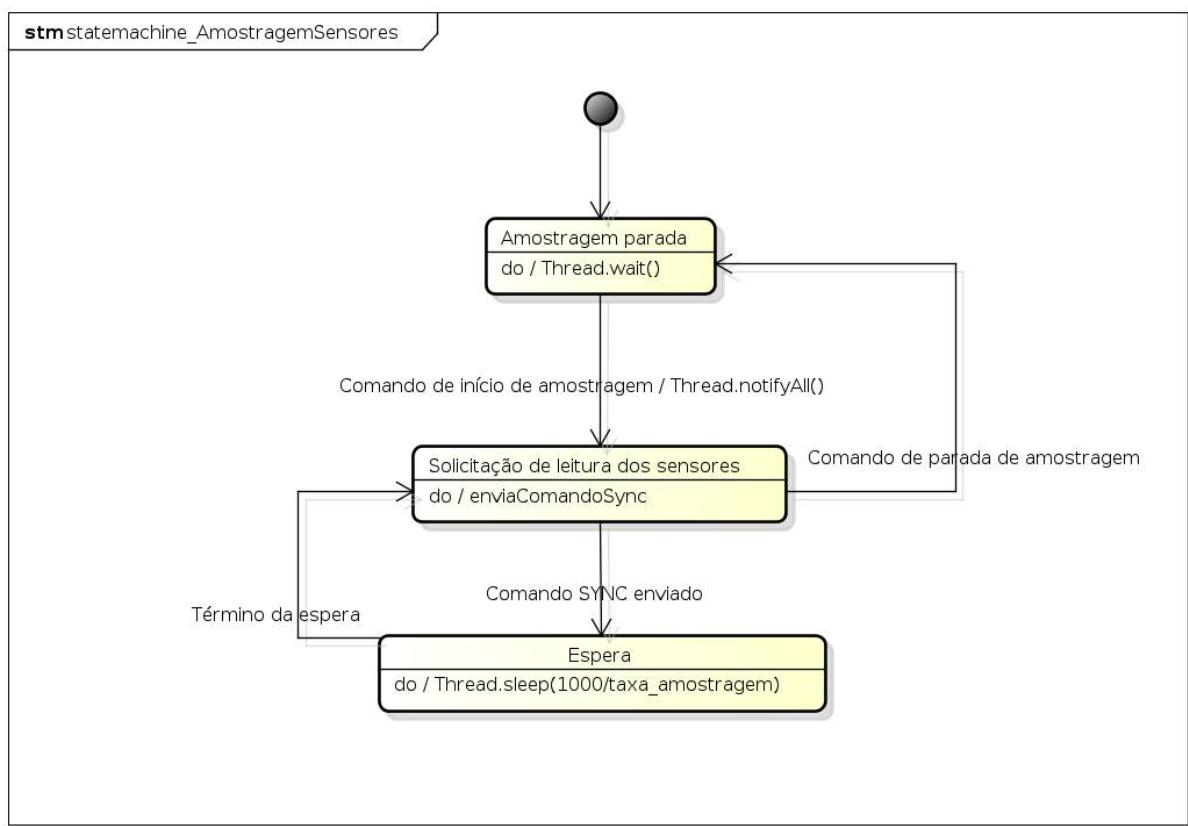
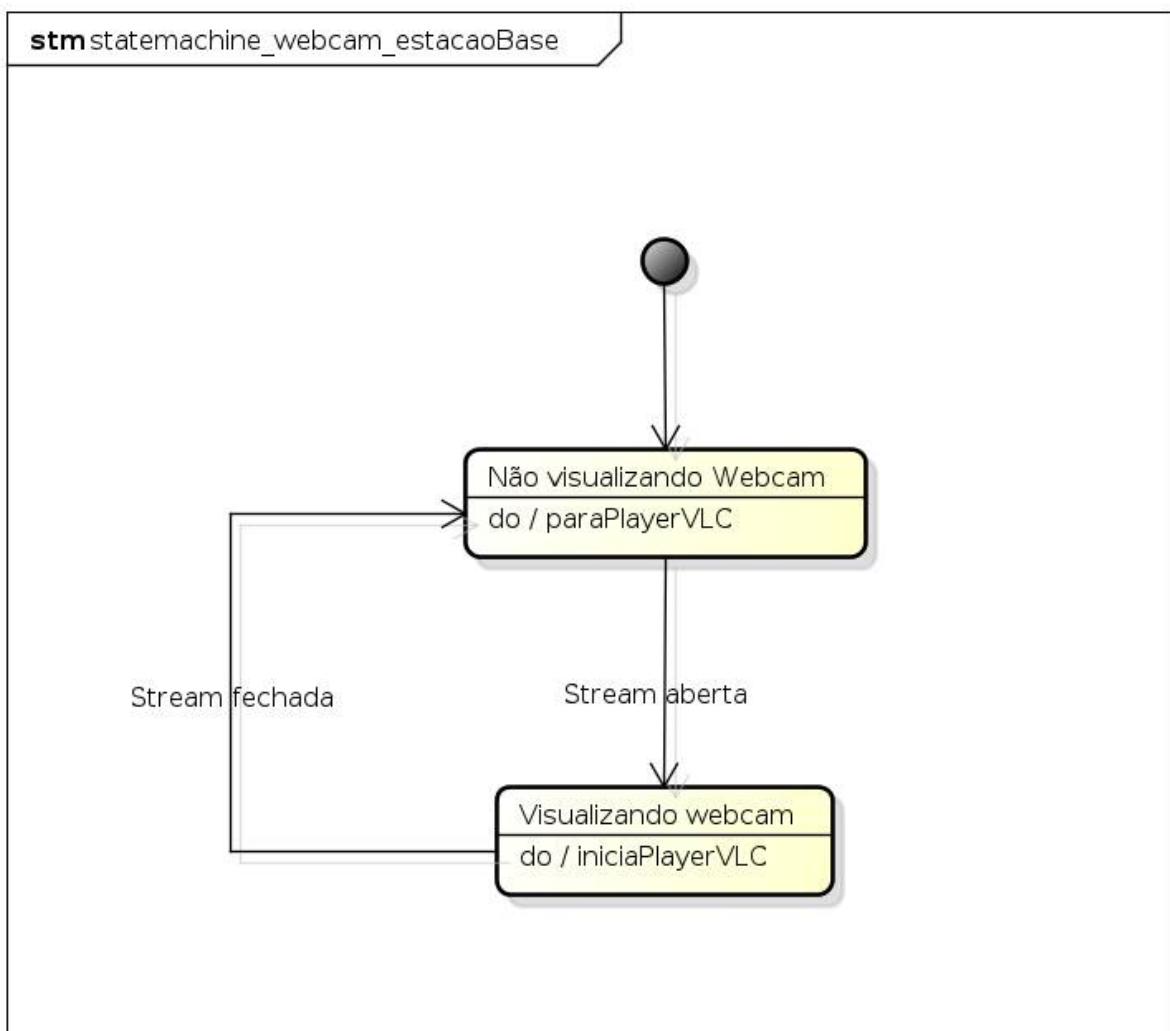
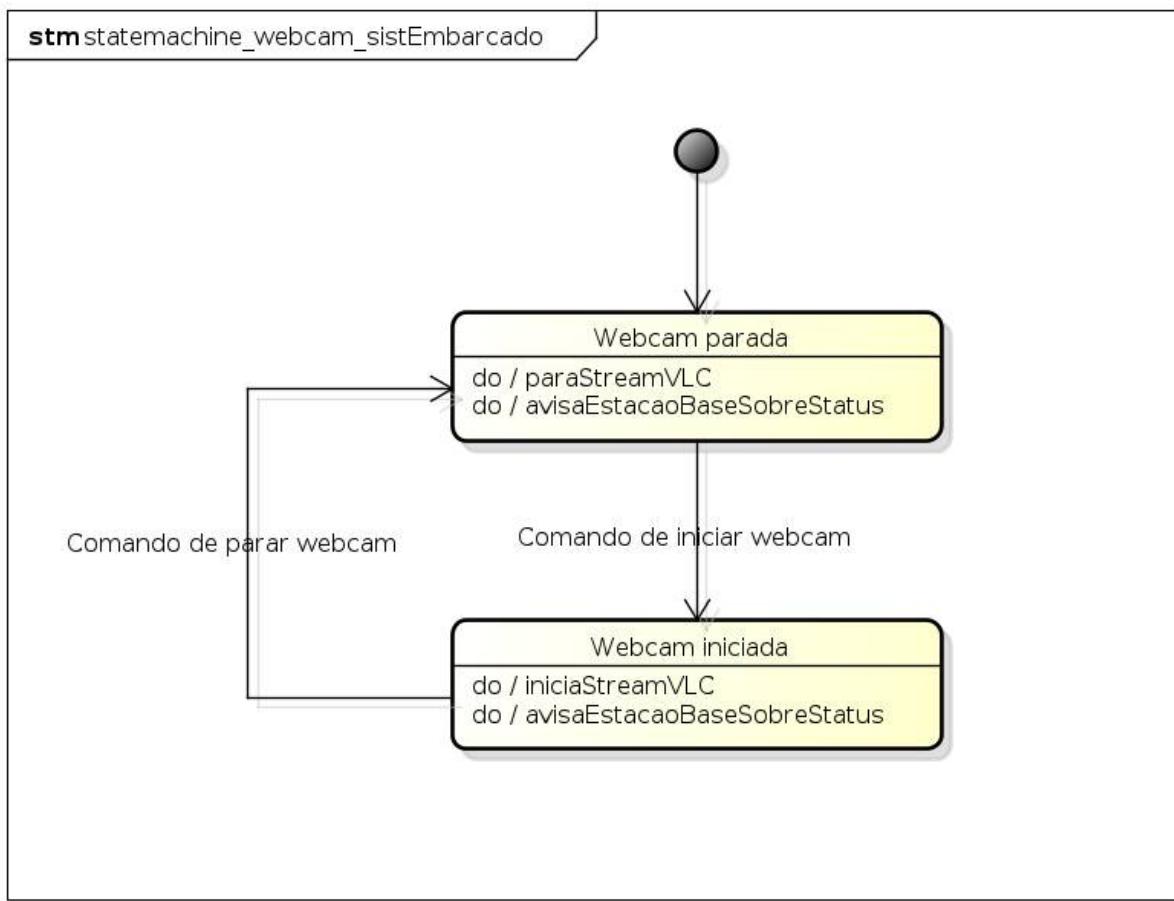


Figura 17: Diagrama de estados da *thread* responsável por efetuar a amostragem dos sensores (linux embarcado).



powered by Astah

Figura 18: Diagrama de estados do visualização de imagens da *webcam* (estaçao base).



powered by Astah

Figura 19: Diagrama de estados do envio de imagens da *webcam* (linux embarcado).

9.6.3 Diagramas de sequênciа

Nessa seção os diagramas de sequência de comandos dos motores, de mensagens de amostras dos sensores e da ativação da webcam. Os diagramas das Figuras 20 e 21 representam como um comando de mudança de velocidade das rodas dado pelo usuário chega até a placa de baixo nível. Os das Figuras 22 e 23 demonstram a sequência dos dados de leituras dos sensores que saem da placa de baixo nível e chegam até o usuário. Os diagramas das Figuras 24 e 25 demonstram um comando de ativação da webcam dado pelo usuário, como ele chega até o Linux embarcado e como posteriormente o usuário recebe as imagens da webcam.

Vale ressaltar que as chamadas assíncronas, ou seja, que não bloqueiam a execução da *thread* chamadora, foram representadas também nestes diagramas.

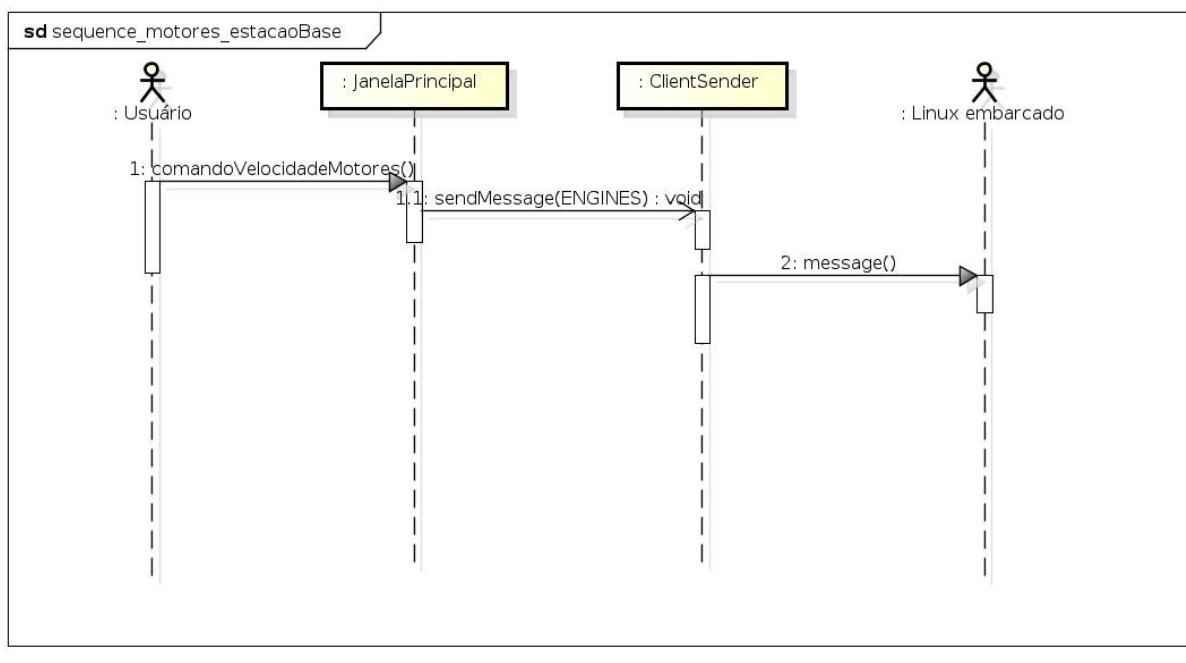


Figura 20: Diagrama de sequência de comando para mudança de velocidade dos motores (representa comando dado pelo usuário).

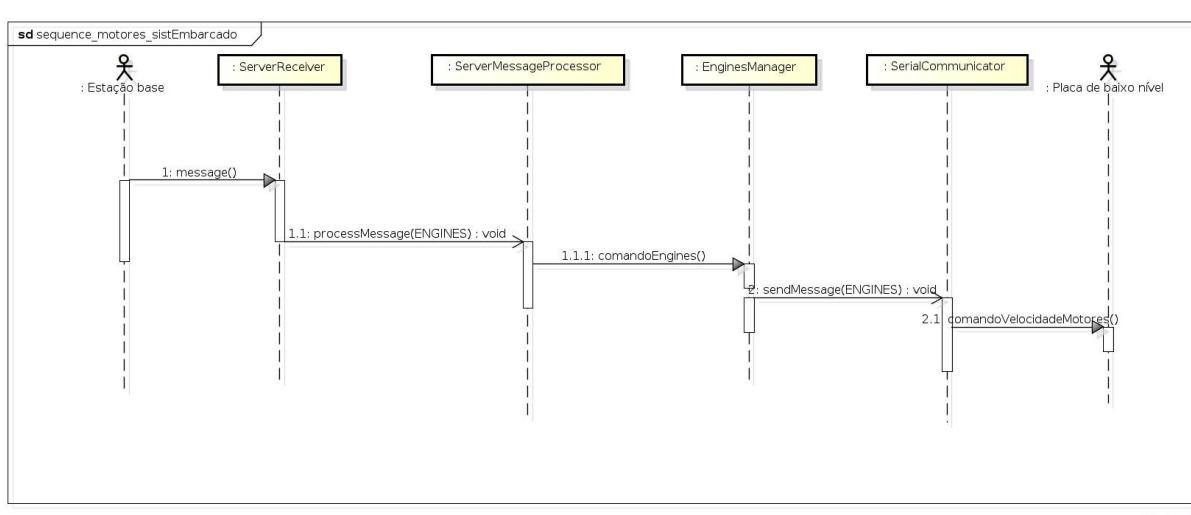
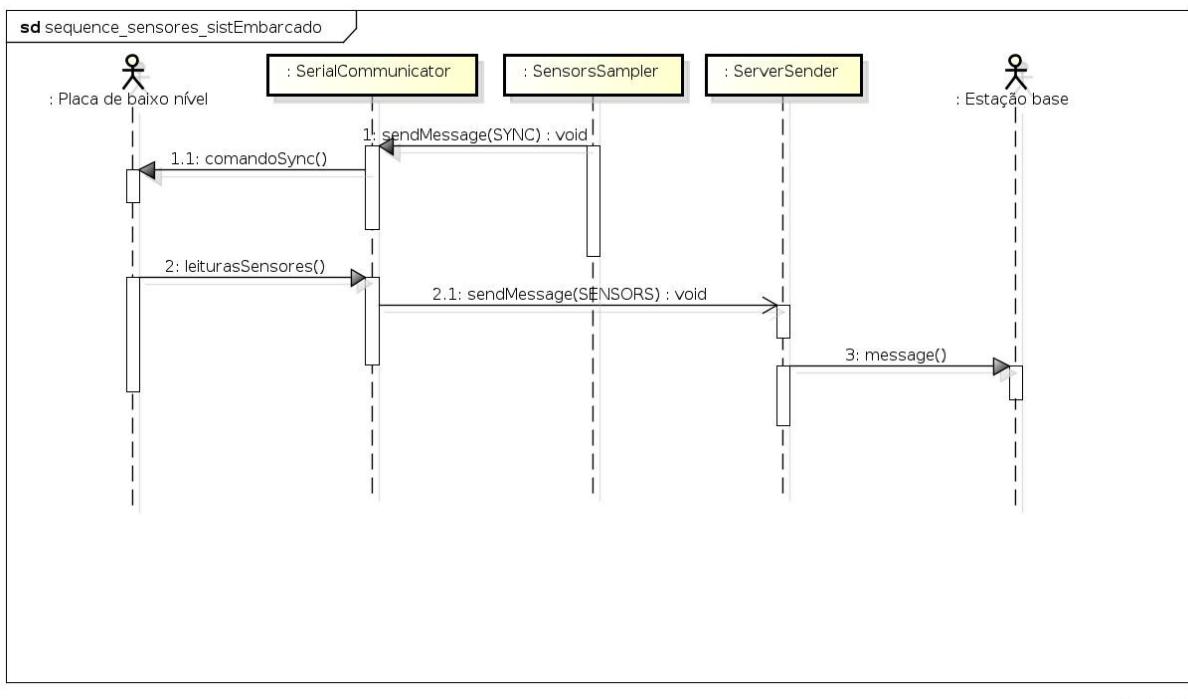
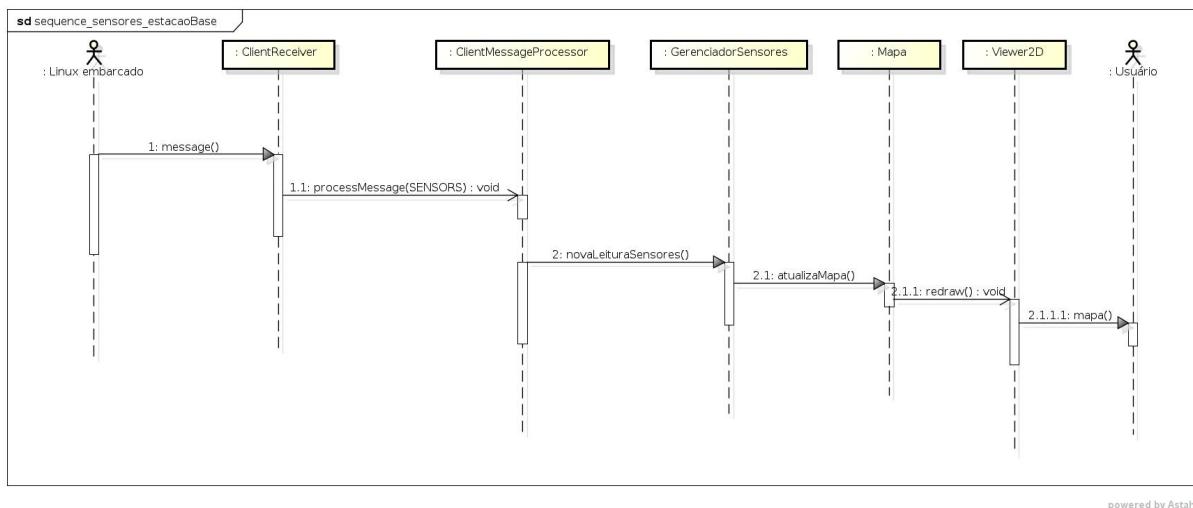


Figura 21: Diagrama de sequência de comando para mudança de velocidade dos motores (representa mensagem chegando no sistema embarcado).



powered by Astah

Figura 22: Diagrama de sequência da amostragem dos sensores (representa amostras saindo do sistema embarcado).



powered by Astah

Figura 23: Diagrama de sequência da amostragem dos sensores (representa mensagem chegando na estação base).

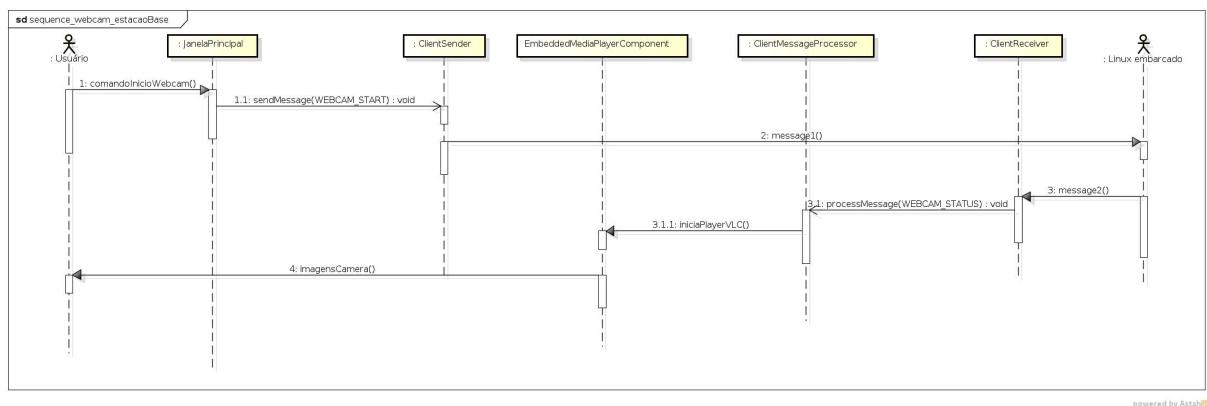


Figura 24: Diagrama de sequência de comando para ativação da webcam (representa comando dado pelo usuário e o *player* da *libVLC* sendo ativado posteriormente).

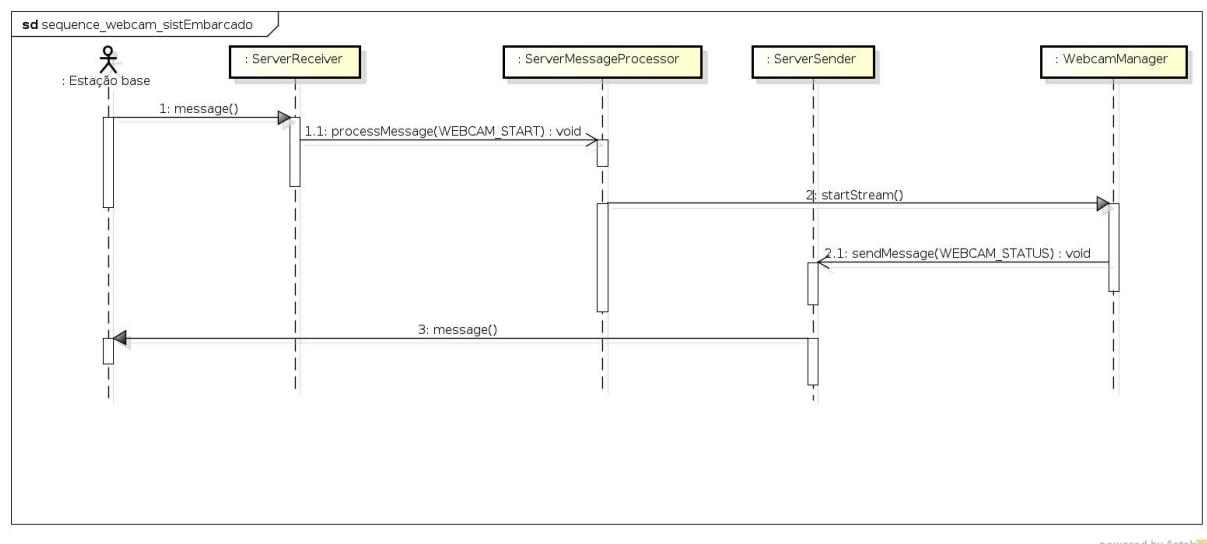


Figura 25: Diagrama de sequência de comando para ativação da webcam (representa mensagem de ativação da webcam chegando no sistema embarcado e estação base sendo posteriormente notificada sobre o novo status).

10 DIAGRAMAS DO HARDWARE

10.1 DIAGRAMA DE BLOCOS

Na figura 26 mostra-se o diagrama de blocos do sistema embarcado e suas conexões com o restante do robô. A seguir está também uma descrição para cada um dos blocos da placa de circuito impresso do sistema embarcado.

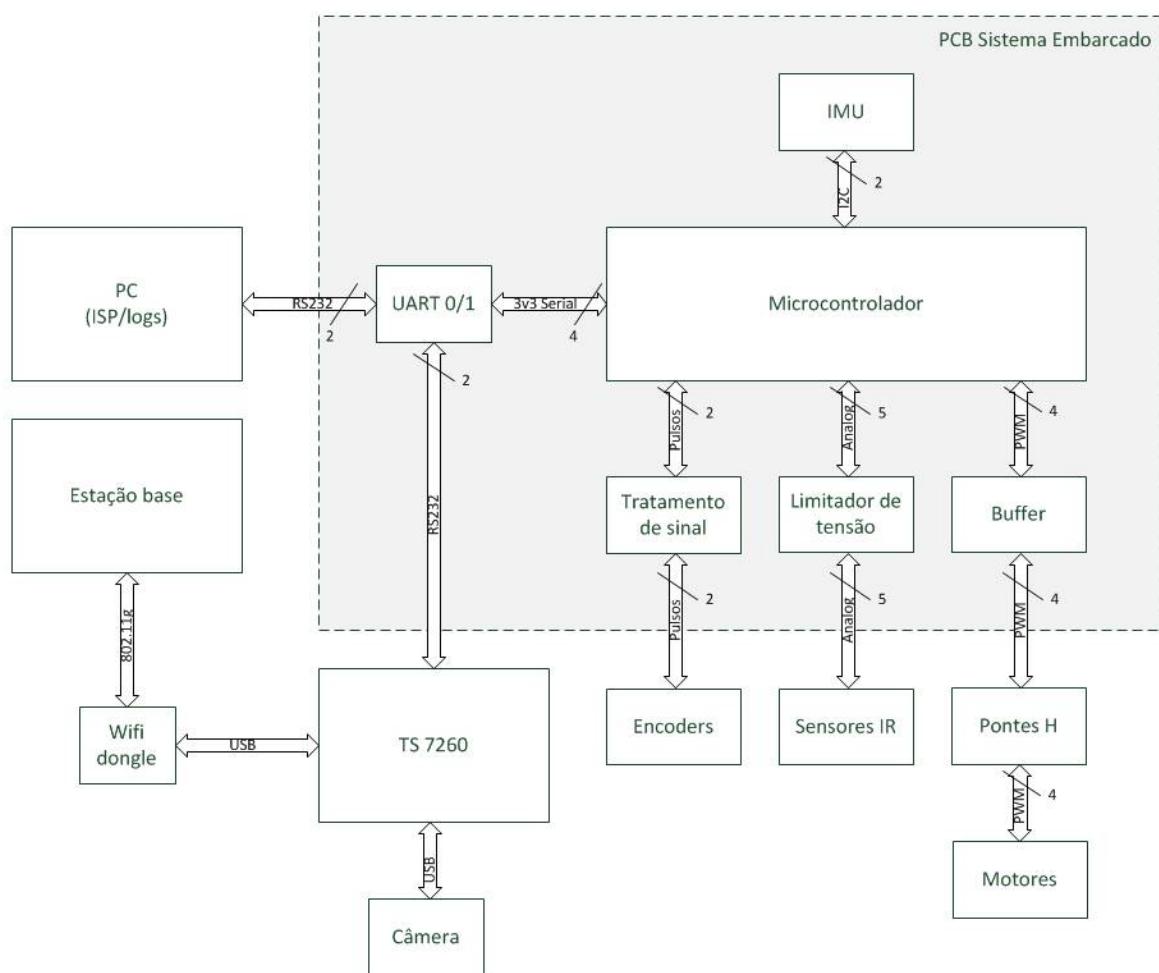


Figura 26: Diagrama de blocos do hardware

1. Microcontrolador: Este bloco fará a leitura dos sensores: encoders, infra-vermelhos, ace-

lerômetro e giroscópios. Além disso possui a implementação do protocolo de comunicação para interação com o linux embarcado da placa TS-7260.

2. UART 0/1: Responsável por ajustar os níveis de tensão para comunicação serial no padrão RS-232 com a placa TS-7260.
3. Buffer: Responsável por fornecer corrente e elevar os níveis de tensão de saída do microcontrolador de 3,3V para 5,0V. Esse buffer é conectado às pontes H já existentes no robô.
4. IMU (Inertial Measurement Unit): possui o acelerômetro e o giroscópio e se comunicará com o microcontrolador por meio do protocolo I2C.
5. Limitador de tensão: Necessário pois os sinais de saída dos sensores de infravermelho que já existem no robô não estão limitados em 5V, podendo a saída ultrapassar 5,0V e danificar o microcontrolador.
6. Tratamento de sinal: Composto por um filtro RC passa baixas e um Schmitt trigger para remover qualquer falha que possa ocorrer na geração dos pulsos no encoder.

10.2 DIAGRAMA ELÉTRICO/ELETRÔNICO

Na figura 27 mostra-se o diagrama de elétrico eletrônico do sistema embarcado. Cada bloco da figura 26 corresponde a alguns componentes do diagrama elétrico eletrônico. A seguir detalha-se um pouco mais cada bloco.

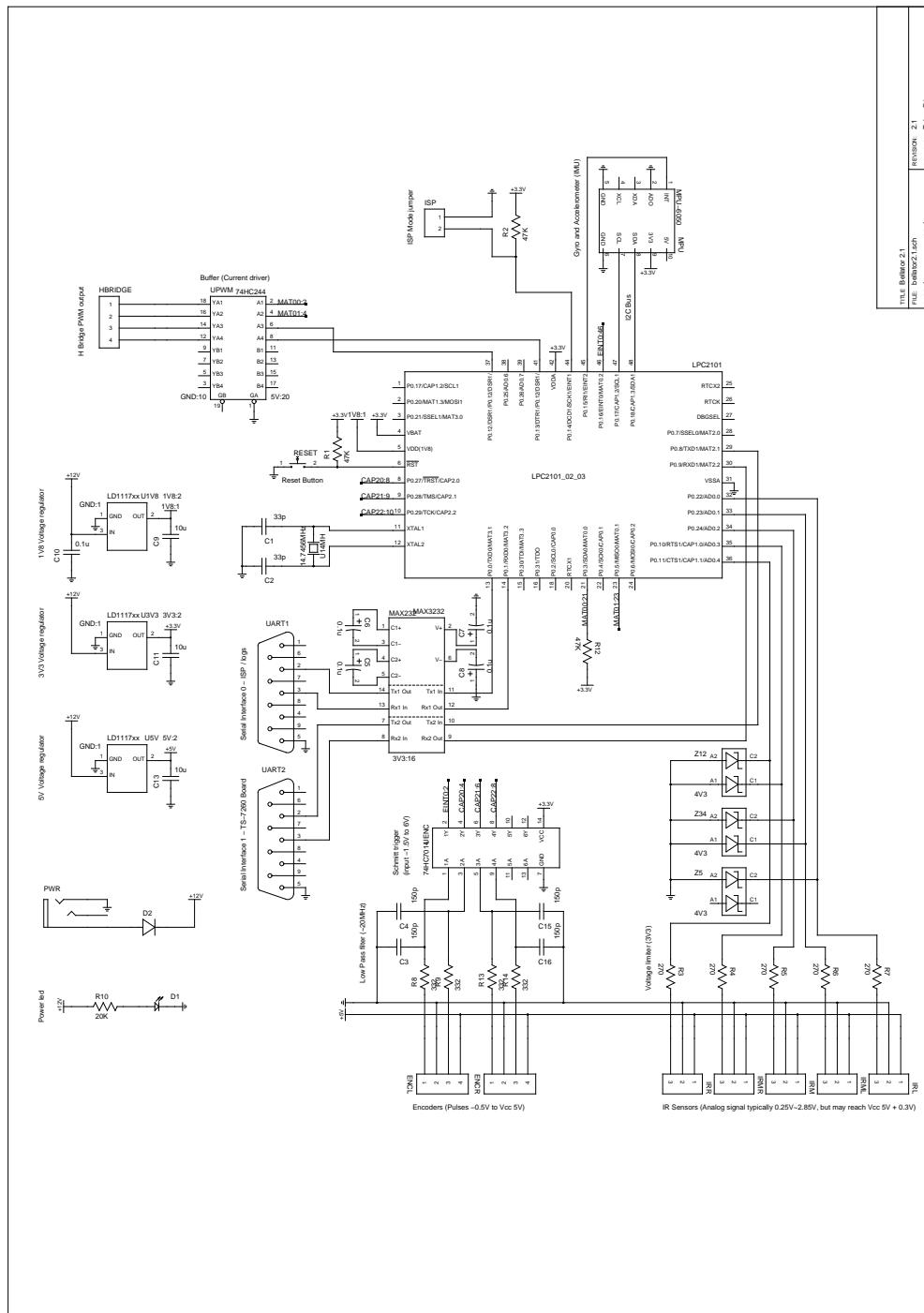


Figura 27: Diagrama elétrico/eletrônico.

1. Microcontrolador: Composto pelo microcontrolador **LCP2103** da NXP que possui arquitetura ARM. Ele dispõe de duas interfaces seriais, conversor analógico digital com 8 canais, interface i2c, entradas de captura e interrupção, saídas de PWM entre outras funções que não serão utilizadas nesse projeto.

2. UART 0/1: Constituído por um chip max3232 que opera em níveis de tensão CMOS e que gera os níveis adequados para o padrão RS-232 utilizando alguns capacitores.
3. Buffer: Constituído por um chip 74HC244, é responsável por fornecer corrente e elevar os níveis de tensão de saída do microcontrolador de 3,3V para 5,0V.
4. IMU: Composto pela placa de desenvolvimento MPU-6050, que possui um chip com o mesmo nome, MPU-6050, e circuitos RC auxiliares necessários para o funcionamento do MPU-6050.
5. Limitador de tensão: Constituído de um resistor com baixo valor, 270 ohms, e um diodo Zener polarizado reversamente e com tensão de ruptura de 4.3V. Quando a tensão de entrada ultrapassar 4.3V o diodo passa a conduzir e mantém a tensão de 4.3V no resistor. O datasheet do microcontrolador sugere que se mantenha a impedância da carga menor que 40kohms, logo a adição de um resistor de 270 ohms pode ser desconsiderado com relação ao erro que possa causar na leitura do conversor. Um resistor de 270 ohms leva a uma corrente de 3.7mA quando a saída do sensor for 5.3V, que é o valor máximo previsto no datasheet.
6. Tratamento de sinal: Composto por um filtro RC passa baixas e um chip 74HC7014. As frequências acima de 20MHz são atenuadas no sinal do encoder. Esse valor foi calculado com base na forma de onda da saída especificada no datasheet do encoder. Para tanto utilizam-se resistores de 332 ohms e capacitores de 150pF.

10.3 PLACA DE CIRCUITO IMPRESSO

O projeto da placa de circuito impresso (PCB) do sistema embarcado de baixo nível está explicitado nas Figuras 28 e 29. Na Figura 30 está presente uma foto da placa pronta com os componentes soldados.

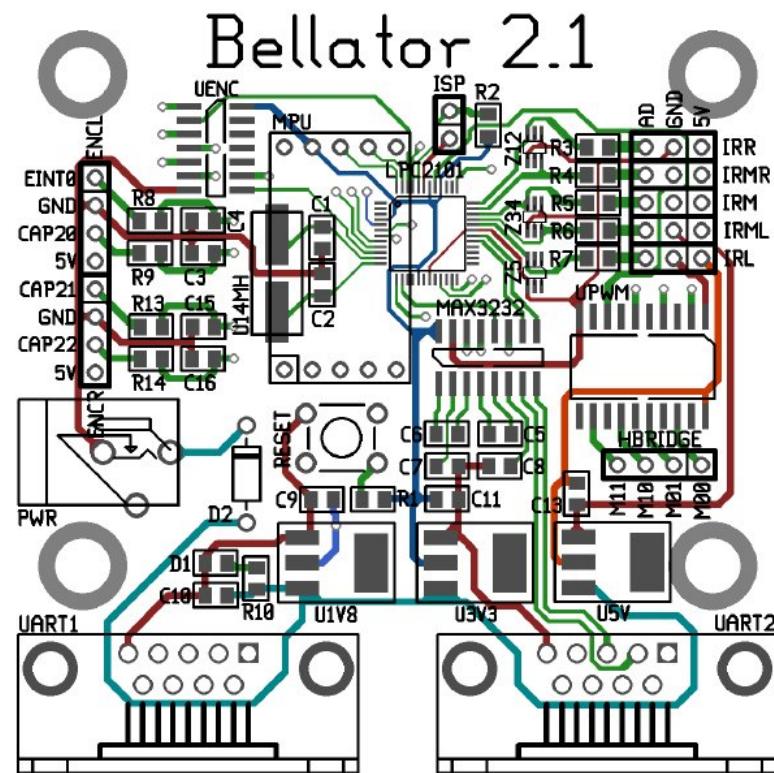


Figura 28: Projeto da PCB – lado de cima.

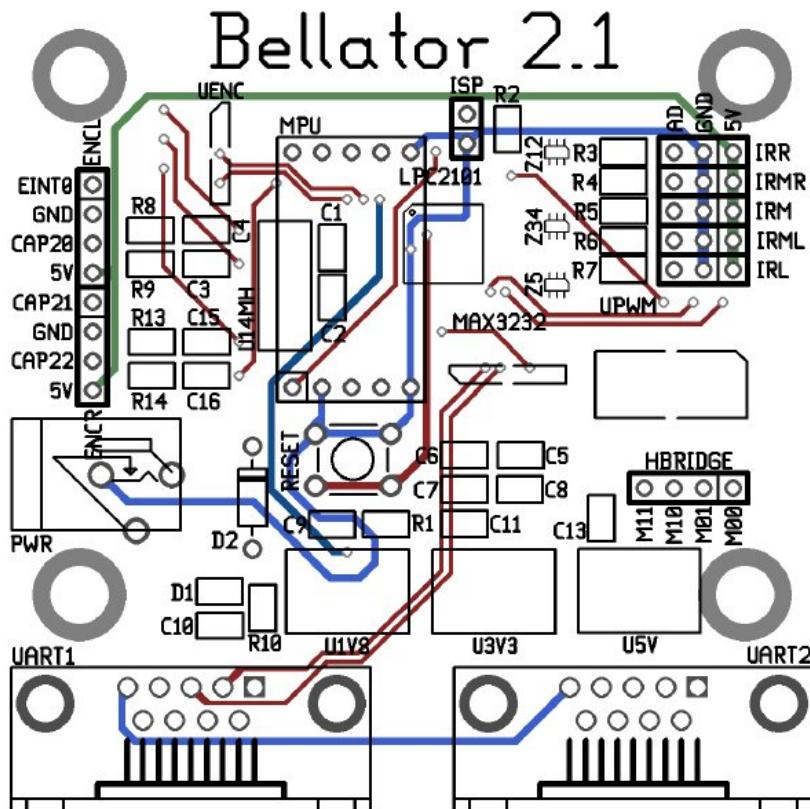


Figura 29: Projeto da PCB – lado de baixo.



Figura 30: PCB montada.

10.3.1 Guia de montagem do hardware

Os passos a seguir, juntamente com a Figura 31 indicam como a placa embarcada deve ser montada.

1. Conexão com sensores IR. A conexão deve ser feita de acordo com a indicação da placa, respeitando os pinos de alimentação e sinal.
2. Conexão com os encoders. A conexão deve ser feita de acordo com a indicação da placa, respeitando os pinos de alimentação e sinal.
3. Conexão para drivers de potência dos motores. Cada pino é um sinal de PWM.
4. Conexão Serial RS232 com a placa TS-7260.
5. Conexão Serial RS232 para envio de logs ao computador ou programação do microcontrolador. (não necessita obrigatoriamente estar conectado durante operação normal)
6. Jumper para seleção do modo ISP para gravação do microcontrolador. (Modo ISP selecionado quando com o jumper durante o boot ou reset da placa).
7. Conexão da alimentação. 12V, negativo na parte externa do conector jack.

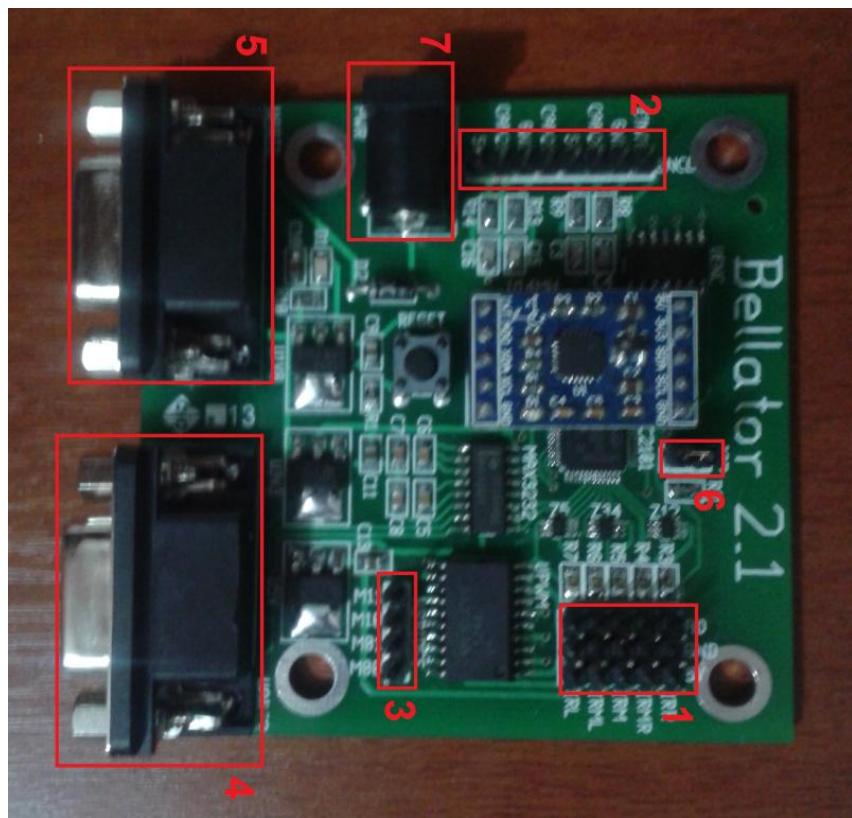


Figura 31: PCB montada.

11 INTERFACE GRÁFICA DA ESTAÇÃO BASE

A interface gráfica desenvolvida para a estação base está representada nas Figuras 32 e 33. O usuário tem acesso na janela principal aos recursos básicos, como imagem da webcam, controles de movimentação do robô e visualização do mapa. A parte de visualização 2D do mapa (a maior área da janela principal) foi desenvolvida a partir da biblioteca do Processing. O visualizador 2D (classe *Viewer2D* apresentada anteriormente) possui recursos de arraste, *zoom* e rotação, desenvolvidos a partir do zero pela equipe. O usuário pode salvar e carregar os mapas criados com o robô através dos botões do canto superior esquerdo da janela. A conexão com o robô pode ser feita rapidamente com o botão “Conexão”, e o controle de gravação do mapa e recebimento de dados dos sensores e da webcam pode ser feito a utilizando-se dos dois botões logo a seguir. Com o botão “Reposicionar”, o usuário pode efetuar o reposicionamento do robô no mapa.

O botão “Avançado” abre uma janela que possui recursos de configuração e utilização avançada, o que inclui alteração dos parâmetros de ajuste fino, como média móvel dos sensores IR e limiares de aceleração e velocidade angular. Além disso, está presente um recurso bastante útil que é o de salvar em um arquivo amostras dos sensores da maneira que são recebidas (sem nenhum tratamento adicional). O programa e o algoritmo de criação do mapa podem ser alterados, e essas amostras podem ser posteriormente carregadas para efetuar testes das alterações feitas. Vale ressaltar que, no decorrer do projeto, esse foi um recurso que melhorou o rendimento da equipe, pois dessa forma testes práticos com o robô não necessitavam ser repetidos continuamente para testar modificações no programa.

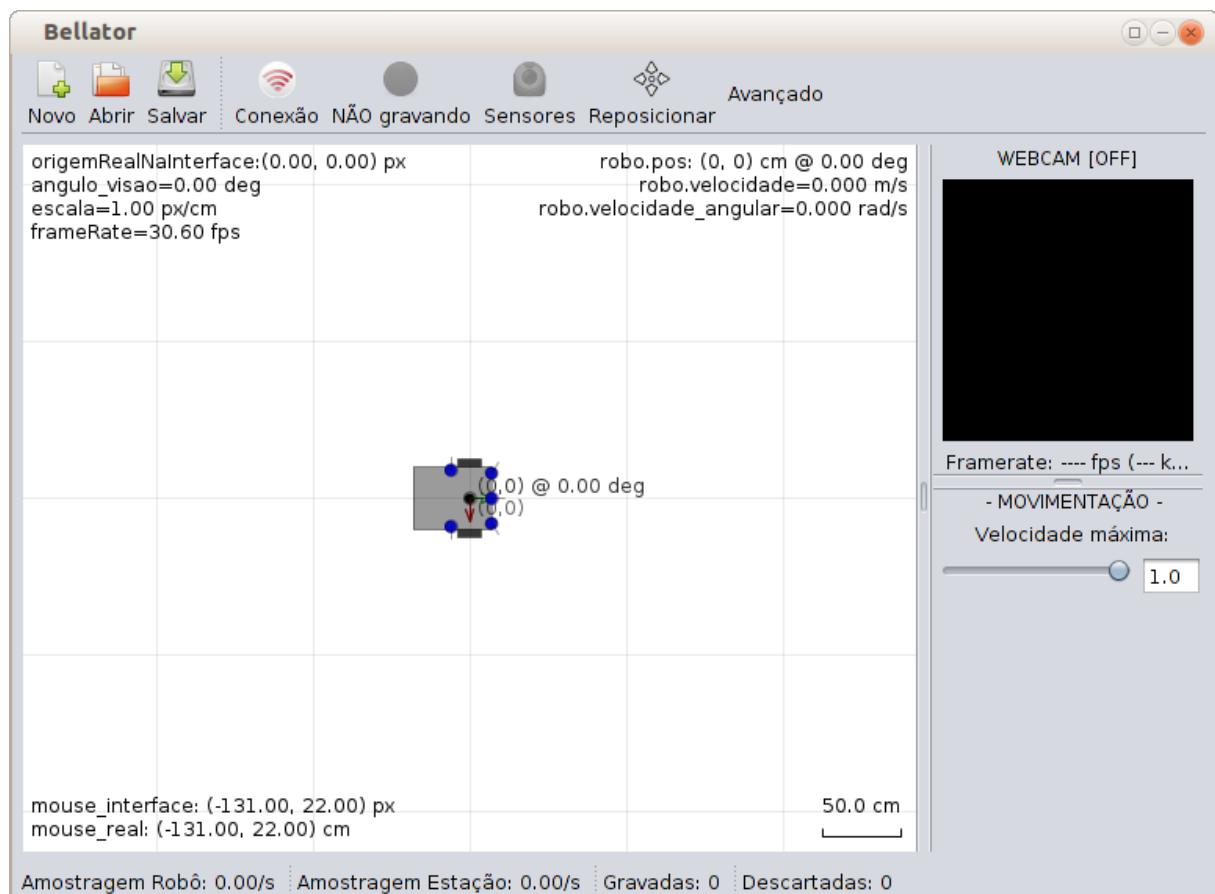


Figura 32: Janela principal da interface gráfica da estação base.

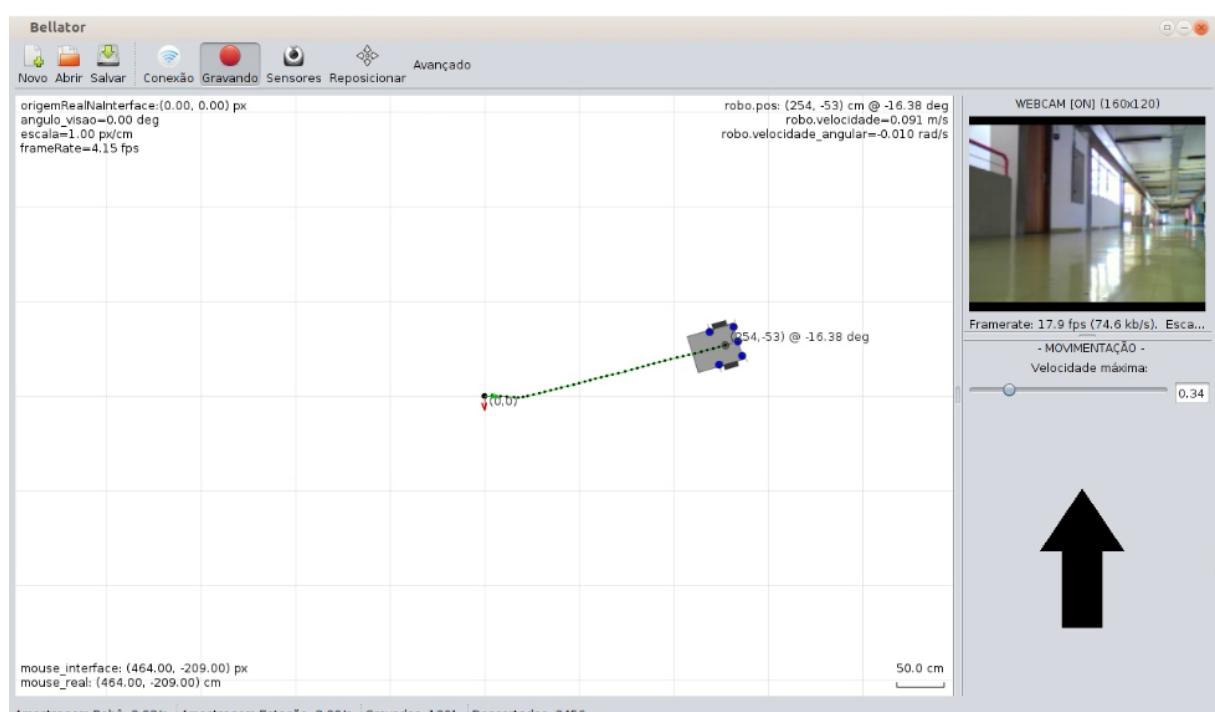


Figura 33: Estação base em utilização na prática.

12 GERAÇÃO DO MAPA

A posição do robô no mapa em cada instante é representada por um ponto no plano cartesiano e por um ângulo, que indica para qual sentido o robô está orientado. Esse ponto no plano indica onde está o centro de movimento do robô, que é o ponto médio entre as duas rodas.

Neste projeto, a determinação do deslocamento do robô é determinada primariamente pelos encoders presentes cada roda. O acelerômetro e o giroscópio são utilizados para aumentar a confiabilidade dos cálculos de deslocamento, principalmente em caso de escorregamento das rodas.

Na próxima seção será explicada a teoria da determinação do deslocamento, velocidade e aceleração do centro de movimento do robô a partir das leituras dos encoders em cada instante. Na seção 12.2 será explicitada a forma como as leituras do acelerômetro e giroscópio serão utilizadas para aumentar a exatidão das medições.

Na Figura 34 está presente um esquema básico do robô visto de cima e virado com a frente para a direita. Na figura estão presentes os nomes das variáveis que são utilizadas nos cálculos posteriores. As medidas R , R_D e R_E representam os raios de um movimento circular uniforme descrito pelo robô, supondo que a roda esquerda esteja se deslocando mais do que a direita. Esse aspecto será melhor explicado nas seções seguintes.

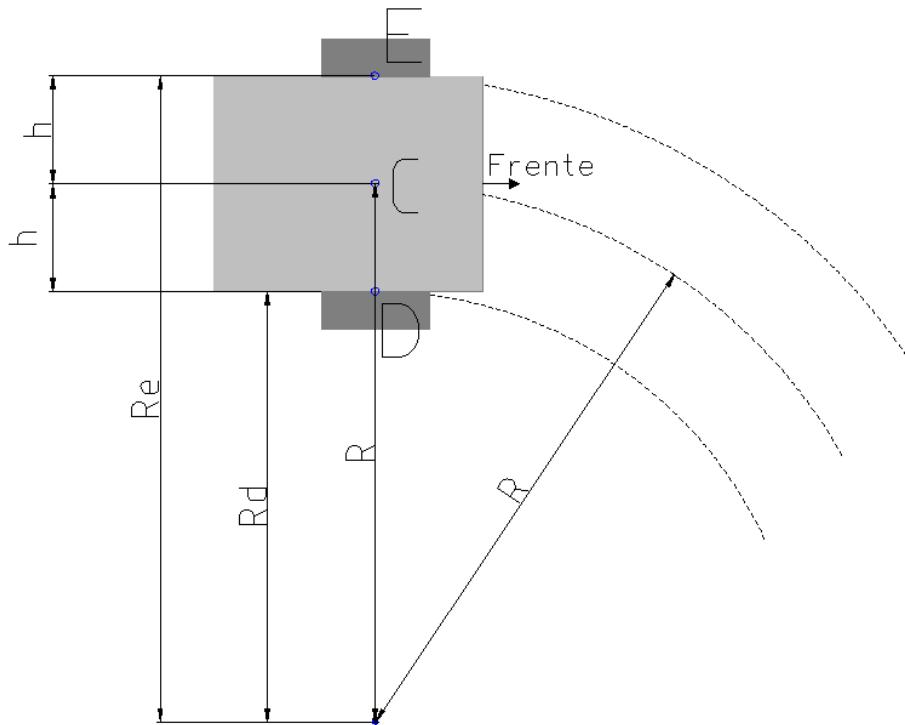


Figura 34: Representação básica do robô em movimento circular uniforme (visão superior).

Um aspecto importante a ressaltar é que o sistema de coordenadas do Processing (a biblioteca usada para desenhar o mapa 2D na interface gráfica) possui o eixo Y invertido. Portanto, os ângulos crescem no sentido horário, e não no anti-horário como seria o convencional. Essa convenção do Processing (ângulos que crescem em sentido horário) é utilizada integralmente no escopo deste projeto.

12.1 ENCODERS

Cada encoder fornece uma medida de contagem de pulsos por volta a cada intervalo de amostragem, e isso permite calcular o deslocamento de cada roda. A partir dessas informações, dois dados importantes podem ser determinados a respeito do deslocamento do centro de movimento do robô: o deslocamento linear (distância absoluta percorrida) e o angular (variação do ângulo de orientação do robô).

Na Figura 35 está presente um representação básica de uma roda, acoplada a um encoder. Os números da figura são utilizados como índices nos cálculos explicitados posteriormente.

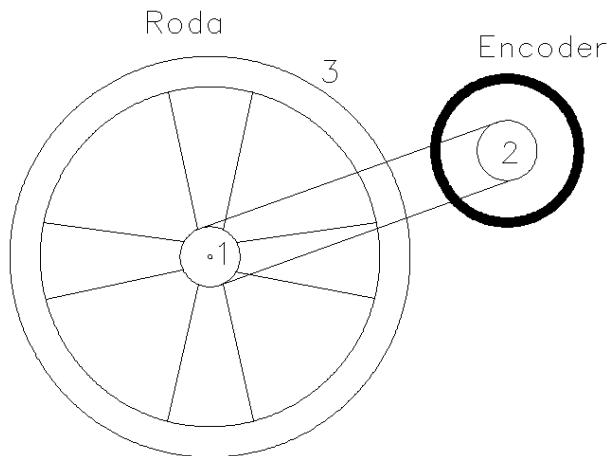


Figura 35: Representação de uma roda acoplada a um encoder.

12.1.1 Deslocamento de cada roda

Um princípio importante utilizado nos cálculos é a relação entre o deslocamento Δx ao redor de uma circunferência (de raio R) e a variação do ângulo $\Delta\theta$:

$$\Delta x = R \cdot \Delta\theta \quad (1)$$

Nos cálculos a seguir, faz-se uso do índice 1 para o eixo da roda, 2 para o eixo do encoder e 3 para a própria roda, de acordo com a figura 35. Para determinar a distância percorrida pela roda, deve-se considerar a circunferência do eixo da roda (C_1), a circunferência do eixo do encoder (C_2) e a circunferência da roda (C_3).

O acoplamento do eixo do encoder com o eixo da roda é feita por uma correia de borracha, e portanto considera-se que o deslocamento (Δx_1) na superfície do eixo da roda é igual ao deslocamento (Δx_2) na superfície do eixo do encoder. Pode-se, com isso, calcular:

$$\Delta x_1 = \Delta x_2 \rightarrow \Delta\theta_1 R_1 = \Delta\theta_2 R_2 \rightarrow \Delta\theta_1 \frac{C_1}{2\pi} = \Delta\theta_2 \frac{C_2}{2\pi}$$

$$\Delta\theta_1 = \frac{C_2}{C_1} \cdot \Delta\theta_2 \quad (3)$$

Calculando-se a relação entre a contagem de pulsos do encoder (E) e o ângulo de rotação ($\Delta\theta_2$) do eixo do encoder, levando-se em conta que há uma contagem de PV pulsos por volta:

$$\Delta\theta_2 = \frac{2\pi}{PV} \cdot E \text{ [rad]} \quad (4)$$

Substituindo-se a equação 4 na 3, tem-se que:

$$\Delta\theta_1 = \frac{C_2}{C_1} \frac{2\pi}{PV} \cdot E \text{ [rad]} \quad (5)$$

Calculando-se a relação entre a variação do ângulo do eixo da roda ($\Delta\theta_1$) e o deslocamento da roda (Δx_3):

$$\Delta\theta_1 = \Delta\theta_3 \rightarrow \Delta\theta_1 = \frac{\Delta x_3}{R_3} \rightarrow \Delta\theta_1 = \Delta x_3 \frac{2\pi}{C_3} \rightarrow \Delta x_3 = \Delta\theta_1 \frac{C_3}{2\pi}$$

Substituindo-se o valor de ($\Delta\theta_1$) da equação 5, obtém-se:

$$\Delta x_3 = \frac{C_2}{C_1} \frac{2\pi}{PV} \cdot E \cdot \frac{C_3}{2\pi}$$

$$\boxed{\Delta x_3 = \frac{C_2 C_3}{PV \cdot C_1} \cdot E} \quad (9)$$

Que é o valor do deslocamento da roda em função da contagem de pulsos do encoder.

12.1.1.1 Valores práticos

Os discos do encoders instalados atualmente no robô possuem nominalmente 1800 pulsos por volta. Porém, eles estão visivelmente danificados, o que gera falhas na geração de pulsos. Na prática, os valores reais foram determinados a partir de 10 rotações dos discos, calculando-se o valor médio. A roda esquerda gera aproximadamente $PV = 1708$ pulsos, e a direita $PV = 1627$ pulsos por volta.

As medidas das circunferências C_1 (eixo da roda), C_2 (eixo do encoder) e C_3 (roda) do robô estão presentes no Apêndice B.

12.1.2 Deslocamento do centro de movimento do robô

Como já explicitado anteriormente, o centro de movimento do robô considerado é o ponto médio entre as duas rodas. As duas variáveis para determinar em cada instante de tempo são o deslocamento linear (distância absoluta percorrida) e o angular (variação do ângulo de orientação do robô). Considerando-se que em cada instante o robô descreve um movimento circular uniforme (MCU), o raio da trajetória deve ser determinado para que os cálculos de posicionamento do robô possam ser feitos. Este raio depende do deslocamento das rodas em cada instante, e é uma importante variável que será estudada na próxima subseção.

Nos cálculos que serão explicitados a seguir, utiliza-se o índice E para a roda esquerda, C para o centro de movimento do robô e D para a roda direita.

Uma relação importante a notar a princípio é que a variação do ângulo de orientação em cada instante é igual nos três pontos: E (roda esquerda), C (centro de movimento) e D (roda direita), visto que todos estão fixos em relação à carcaça do robô. Parte-se da seguinte relação fundamental, portanto:

$$\Delta\theta_E = \Delta\theta_D = \Delta\theta_C \quad (12)$$

12.1.2.1 Raio do movimento circular uniforme

Para determinar o raio (R) descrito pelo centro de movimento do robô em sua trajetória instantânea em MCU, usa-se os dois primeiros termos da igualdade da equação 12:

$$\Delta\theta_E = \Delta\theta_D \rightarrow \frac{\Delta x_E}{R_E} = \frac{\Delta x_D}{R_D}$$

Sabe-se, pela Figura 34, que:

$$R_E = R + h, \quad R_D = R - h$$

Portanto:

$$\frac{\Delta x_E}{R+h} = \frac{\Delta x_D}{R-h} \rightarrow \frac{\Delta x_E}{\Delta x_D} = \frac{R+h}{R-h} \rightarrow \frac{\Delta x_E(R-h)}{\Delta x_D} = R+h$$

$$\frac{\Delta x_E \cdot R - \Delta x_E \cdot h}{\Delta x_D} = R + h \rightarrow \frac{\Delta x_E}{\Delta x_D} \cdot R - \frac{\Delta x_E}{\Delta x_D} \cdot h = R + h \rightarrow \frac{\Delta x_E}{\Delta x_D} \cdot R - R = \frac{\Delta x_E}{\Delta x_D} \cdot h + h$$

$$R \left(\frac{\Delta x_E}{\Delta x_D} - 1 \right) = h \left(\frac{\Delta x_E}{\Delta x_D} + 1 \right) \rightarrow R = h \cdot \frac{\left(\frac{\Delta x_E}{\Delta x_D} + 1 \right)}{\left(\frac{\Delta x_E}{\Delta x_D} - 1 \right)} \rightarrow R = h \cdot \frac{\frac{\Delta x_E + \Delta x_D}{\Delta x_D}}{\frac{\Delta x_E - \Delta x_D}{\Delta x_D}}$$

$$R = h \cdot \frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D}$$

(18)

Vê-se que o raio R do movimento circular uniforme em cada instante depende do deslocamento de cada roda (Δx_E e Δx_D) e da distância h entre as rodas e o centro de movimento do robô.

12.1.2.2 Deslocamento linear

Para calcular o deslocamento linear do centro de movimento do robô, usa-se os dois últimos termos da equação 12. Vale ressaltar que poderiam ser escolhidos também o primeiro e o último termos, pois o resultado obtido seria idêntico. Tem-se que:

$$\Delta \theta_D = \Delta \theta_C \rightarrow \frac{\Delta x_D}{R_D} = \frac{\Delta x_C}{R}$$

Mas:

$$R_D = R - h$$

Portanto:

$$\frac{\Delta x_D}{R - h} = \frac{\Delta x_C}{R} \rightarrow x_D = x_C \cdot \frac{R - h}{R}$$

Substituindo-se o valor de R da equação 18:

$$\Delta x_D = \Delta x_C \left[\frac{h \cdot \frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D} - h}{h \cdot \frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D}} \right]$$

Dividindo-se o numerador e denominador do segundo termo por h :

$$\Delta x_D = \Delta x_C \left[\frac{\frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D} - 1}{\frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D}} \right]$$

Multiplicando-se o numerador e denominador do segundo termo por $\frac{\Delta x_E - \Delta x_D}{\Delta x_E + \Delta x_D}$:

$$\begin{aligned}\Delta x_D &= \Delta x_C \left[1 - \frac{\Delta x_E - \Delta x_D}{\Delta x_E + \Delta x_D} \right] \rightarrow \Delta x_D = \Delta x_C \left[\frac{(\Delta x_E + \Delta x_D) - (\Delta x_E - \Delta x_D)}{\Delta x_E + \Delta x_D} \right] \\ \Delta x_D &= \Delta x_C \cdot \left[\frac{2\Delta x_D}{\Delta x_E + \Delta x_D} \right] \rightarrow \Delta x_C = \frac{\Delta x_D}{\left(\frac{2\Delta x_D}{\Delta x_E + \Delta x_D} \right)} \\ \boxed{\Delta x_C = \frac{\Delta x_E + \Delta x_D}{2}} \end{aligned} \quad (25)$$

Nota-se que o deslocamento linear do centro de movimento do robô (Δx_C) é uma média simples dos dois deslocamentos lineares das rodas. Vê-se que ele não depende do raio de deslocamento nem da distância entre as duas rodas.

12.1.2.3 Deslocamento angular

O deslocamento angular ($\Delta\theta_c$) do centro de movimento do robô em cada instante pode ser calculado a partir do deslocamento linear e do raio do movimento circular uniforme. Usando-se a relação da equação 1, tem-se que:

$$\boxed{\Delta\theta_c = \frac{\Delta x_C}{R}} \quad (29)$$

Onde Δx_C é o deslocamento linear do robô (equação 25) e R é o raio do movimento circular uniforme (equação 18).

12.1.2.4 Casos especiais

Há dois casos especiais que devem ser considerados no cálculo do deslocamento (a partir dos dados dos encoders) do centro de movimento do robô. O primeiro é quando as duas rodas têm deslocamento igual ($\Delta x_E = \Delta x_D$). O raio do movimento circular (equação 18) neste caso é:

$$R = h \cdot \frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D} = h \cdot \frac{2 \cdot \Delta x_E}{0} \rightarrow \infty \quad (32)$$

O raio tende a infinito, o que implica que o deslocamento angular (equação 29) seja:

$$\Delta\theta_c = \frac{\Delta x_C}{R} = \frac{\Delta x_C}{\infty} \rightarrow 0 \quad (33)$$

Já o deslocamento linear (equação 25) é:

$$\Delta x_C = \frac{\Delta x_E + \Delta x_D}{2} = \frac{2\Delta x_E}{2} = \Delta x_E \quad (34)$$

Isso corresponde à realidade, uma vez que quando as rodas têm deslocamentos iguais o robô está se deslocando sem fazer curvas. Não há deslocamento angular, portanto, e o deslocamento linear do centro de movimento é igual ao deslocamento de cada roda.

O segundo caso especial ocorre quando os deslocamento têm módulos iguais, porém sentidos contrários (ou seja, $\Delta x_E = -\Delta x_D$). O raio nesse caso é:

$$R = h \cdot \frac{\Delta x_E + \Delta x_D}{\Delta x_E - \Delta x_D} = h \cdot \frac{\Delta x_E - \Delta x_E}{\Delta x_E + \Delta x_E} = 0 \quad (35)$$

O deslocamento linear (equação 25) é:

$$\Delta x_C = \frac{\Delta x_E + \Delta x_D}{2} = \frac{\Delta x_E - \Delta x_E}{2} = 0 \quad (36)$$

Esse valor corresponde à realidade, pois quando as rodas se deslocam em sentidos contrários, e na mesma quantidade, o centro de movimento do robô não se desloca linearmente, mas apenas muda seu ângulo. Usando-se a equação 29, tenta-se calcular o valor do deslocamento angular:

$$\Delta\theta_c = \frac{\Delta x_C}{R} = \frac{0}{0} \quad (37)$$

O valor obtido é indeterminado quando usa-se essa equação. Porém, analisando-se a natureza deste caso especial, pode ser notado que há um movimento circular cujo centro é o ponto médio entre as rodas (que é o centro de movimento do robô). O raio do MCU é a distância h entre uma roda e o centro do robô, e o deslocamento ao longo da circunferência é o deslocamento de qualquer uma das rodas. Portanto, a equação 1 pode ser utilizada, isolando-se a variável θ , para determinar o deslocamento angular do robô:

$$\Delta\theta_c = \frac{\Delta x_E}{h} \quad (38)$$

12.1.2.5 Considerações sobre a notação utilizada

A notação utilizada no decorrer do texto para representar o estado de uma variável em certo instante discreto de tempo é um número subscrito entre parênteses. Por exemplo, a posição \vec{P} do robô em um instante de tempo n é representado por: $\vec{P}_{(n)}$.

Outro aspecto presente no decorrer do texto, que vale ser ressaltado para que haja melhor entendimento das demonstrações, é o fato de que $\Delta x_{(n)}$ é o deslocamento absoluto do robô no intervalo de tempo discreto de $(n - 1)$ até n .

12.1.2.6 Velocidade e aceleração

A velocidade e aceleração lineares do centro de movimento do robô podem ser calculadas por derivação numérica do deslocamento e velocidade em cada intervalo de tempo. Em cada intervalo discreto n :

$$v_{c(n)} = \frac{\Delta x_{c(n)}}{t_{(n)} - t_{(n-1)}} \quad (39)$$

$$a_{c(n)} = \frac{v_{c(n)} - v_{c(n-1)}}{t_{(n)} - t_{(n-1)}} \quad (40)$$

A velocidade e aceleração angulares podem ser também obtidas por derivação numérica, considerando-se que o raio do movimento circular uniforme do robô é constante dentro do intervalo considerado. Em cada intervalo discreto n :

$$\omega_{c(n)} = \frac{\Delta \theta_{c(n)}}{t_{(n)} - t_{(n-1)}} \quad (41)$$

$$\alpha_{c(n)} = \frac{\omega_{c(n)} - \omega_{c(n-1)}}{t_{(n)} - t_{(n-1)}} \quad (42)$$

12.2 ACELERÔMETRO E GIROSCÓPIO

O acelerômetro e o giroscópio são utilizados para aumentar a confiabilidade dos dados de deslocamento do robô em caso de escorregamento das rodas.

Como definido na seção 9.6.1, é recebido um valor de 2 bytes para cada eixo do acelerômetro. A faixa de funcionamento do acelerômetro está configurada em ± 2 g, logo a sensi-

bilidade pelo *datasheet* é de 16384 LSB/g. Portanto, o valor de aceleração pode ser obtido pela fórmula:

$$a = \frac{valorMedido}{16384} \cdot g \text{ [m/s}^2\text{]} \quad (43)$$

Onde g é a aceleração da gravidade (9,80665 [m/s²]).

Para cada eixo do giroscópio, também é obtido um valor de 2 bytes. A faixa de funcionamento do giroscópio está configurada em ± 250 graus/s, logo a sensibilidade pelo *datasheet* é de 131 LSB/(graus/s). Portanto, o valor de velocidade angular pode ser obtida pela fórmula:

$$\omega = \frac{valorMedido}{131} \text{ [graus/s]} = \frac{\pi}{180} \cdot \frac{valorMedido}{131} \text{ [rad/s]} \quad (44)$$

A princípio, apenas o eixo Y do acelerômetro (voltado para trás do robô) e o eixo Z do giroscópio (positionado no sentido baixo/cima, e com ângulos crescentes no sentido anti-horário) serão utilizados para mapeamento, visto que os dados poderão ser dessa forma comparados facilmente com os dados obtidos pelos encoders. Na Figura 36 está explicitada a disposição dos eixos do acelerômetro e giroscópio, extraída do *datasheet* do MPU-6050. Para que uma comparação de acelerações (encoders *vs* acelerômetro) e velocidades angulares (encoders *vs.* giroscópio) seja realizada, o circuito integrado deve ser posicionado no centro de movimento do robô (ponto médio entre as rodas).

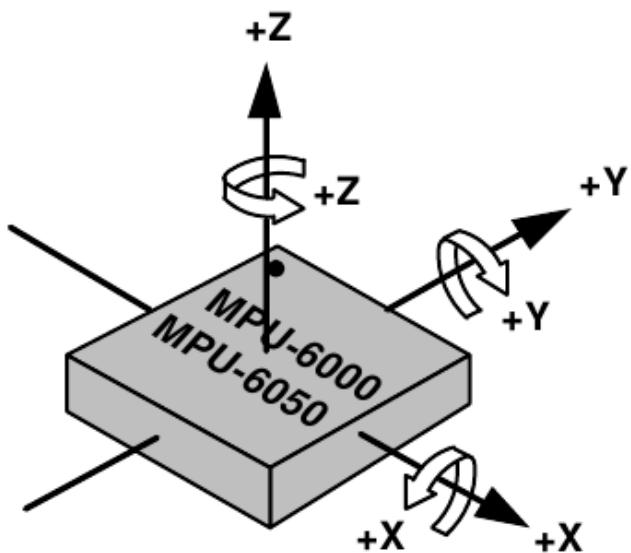


Figura 36: Eixos do acelerômetro e giroscópio.

Fonte: Datasheet do MPU-6050

Na prática, observou-se que o acelerômetro e o giroscópio possuem um *offset* inicial, mesmo com o robô estando parado. O valor desse *offset* deve ser descontado de todas as medidas obtidas para que os valores finais correspondam à realidade. Outro aspecto a ressaltar é que como o eixo Y do acelerômetro é voltado para trás do robô, os valores devem ser utilizado com sinal invertido. O eixo Z do giroscópio também deve ser utilizado com o sinal trocado, pois (como explicitado no início do capítulo) o sistema de coordenadas utilizado neste projeto implica em que os ângulos crescem em sentido horário.

Para determinar a velocidade e deslocamento lineares a partir dos dados do acelerômetro, efetua-se uma integração numérica (simples e dupla) da aceleração linear em cada intervalo discreto n :

$$v_{(n)} = v_{(n-1)} + a_{(n)} \cdot (t_{(n)} - t_{(n-1)}) \quad (45)$$

$$\Delta x_{(n)} = v_{(n)} \cdot (t_{(n)} - t_{(n-1)}) \quad (46)$$

Para determinar o deslocamento angular a partir dos dados do giroscópio, efetua-se uma integração numérica (simples) da velocidade angular em cada intervalo discreto n :

$$\Delta\theta_{(n)} = \Delta\theta_{(n-1)} + \omega_{(n)} \cdot (t_{(n)} - t_{(n-1)}) \quad (47)$$

12.3 ALGORITMO DE POSICIONAMENTO

O algoritmo proposto para determinação da posição do robô em cada instante de tempo, utilizando os vários sensores (encoders, acelerômetro e giroscópio), está explicitado nesta seção. Para cada amostra dos sensores que é recebida o algoritmo efetua os seguintes passos:

1. A partir das leituras dos encoders (Δx_E e Δx_D), usando as equações 25 e 29, calcular deslocamento linear (Δx_C , em metros) e angular ($\Delta\theta_c$, em rad) do centro de movimento do robô.
2. Derivar duas vezes o deslocamento linear (Δx_C), usando as equações 39 e 40 para obter aceleração linear (a_C , em m/s^2), e derivar uma vez o deslocamento angular (θ_C), usando a equação 41, para obter a velocidade angular (ω_C , em rad/s).
3. Comparar a aceleração linear (a_C) e a velocidade angular (ω_C), obtidas com os encoders, com as leituras do acelerômetro (a_a) e giroscópio (ω_g). Caso as diferenças dos valores passem de limiares L_a e L_ω (determinados experimentalmente), é provável que um escorregamento de rodas ou falha de leitura tenha ocorrido. Especificar com base nas comparações pesos p_{a_c} e p_{a_a} para a aceleração linear (encoders vs. acelerômetro) e pesos p_{ω_c} e p_{ω_g} para velocidade angular (encoders vs. giroscópio), dando mais prioridade ao acelerômetro e giroscópio caso escorregamentos sejam detectados. Em termos formais, tem-se que:

$$p_{a_c} = \begin{cases} 0, & |a_C - a_a| > L_a \\ 1, & \text{caso contrário} \end{cases}$$

$$p_{a_a} = \begin{cases} 1, & |a_C - a_a| > L_a \\ 0, & \text{caso contrário} \end{cases}$$

$$p_{\omega_c} = \begin{cases} 0, & |\omega_C - \omega_a| > L_\omega \\ 1, & \text{caso contrário} \end{cases}$$

$$p_{\omega_g} = \begin{cases} 1, & |\omega_C - \omega_a| > L_\omega \\ 0, & \text{caso contrário} \end{cases}$$

4. Calcular com base nos pesos obtidos anteriormente a aceleração linear final (a) e a velocidade angular final (ω), de acordo com as equações 48 e 49.

$$a = p_{ac} \cdot a_c + p_{a_a} \cdot a_a \quad (48)$$

$$\omega = p_{\omega_c} \cdot \omega_c + p_{\omega_g} \cdot \omega_g \quad (49)$$

5. Usando-se as equações 45 e 46, integrar duplamente a aceleração linear (a) para obter o deslocamento linear (Δx) do robô, e a partir da equação 47 integrar uma vez a velocidade angular (ω) para obter o deslocamento angular ($\Delta \theta$) do robô.
6. A partir das equações 50 e 51 explicitadas abaixo, calcular a nova posição ($\vec{P}_{(n)}$) do robô, e a partir da equação 52 calcular o novo ângulo $\theta_{(n)}$ do robô. Decompondo-se o vetor posição (\vec{P}) em suas componentes P_x e P_y , tem-se que no intervalo discreto n :

$$P_{x(n)} = P_{x(n-1)} + \Delta x \cdot \cos(\theta_{(n-1)}) \quad (50)$$

$$P_{y(n)} = P_{y(n-1)} + \Delta x \cdot \sin(\theta_{(n-1)}) \quad (51)$$

$$\theta_{(n)} = \theta_{(n-1)} + \Delta \theta \quad (52)$$

Um ponto importante a ressaltar é que os pesos calculados no passo 3 são utilizados (a princípio) como sendo 0 ou 1. Ou seja, se escorregamentos forem detectados nas rodas os pesos p_{a_a} e p_{ω_g} devem valer 1, e p_{ac} e p_{ω_c} devem valer 0. Dessa forma os dados do acelerômetro e do giroscópio são utilizados (nesses momentos específicos) em detrimento dos dados dos encoders.

12.4 SENsores INFRA-VERMELHOS

Os sensores infra-vermelhos são usados para detectar obstáculos próximos ao robô (de 30 a 150 cm de distância). Na Figura 37 está explicitada uma representação de um sensor infra-vermelho detectando um obstáculo. Na figura estão presentes os nomes das variáveis utilizadas nos cálculos. Como já explicado no começo do capítulo, o sistema de coordenadas utilizado nesse projeto possui o eixo Y invertido. Portanto, os ângulos crescem no sentido horário e não no anti-horário como seria o convencional. Por essa razão, o ângulo “-theta(n)” da figura – que nesse caso específico está em sentido anti-horário em relação ao eixo X (horizontal) do mapa – é negativo. O ângulo “thetaIR” é medido com com relação ao eixo X do robô (e não do mapa), e na figura ele é positivo pois está em sentido horário com relação a esse eixo.

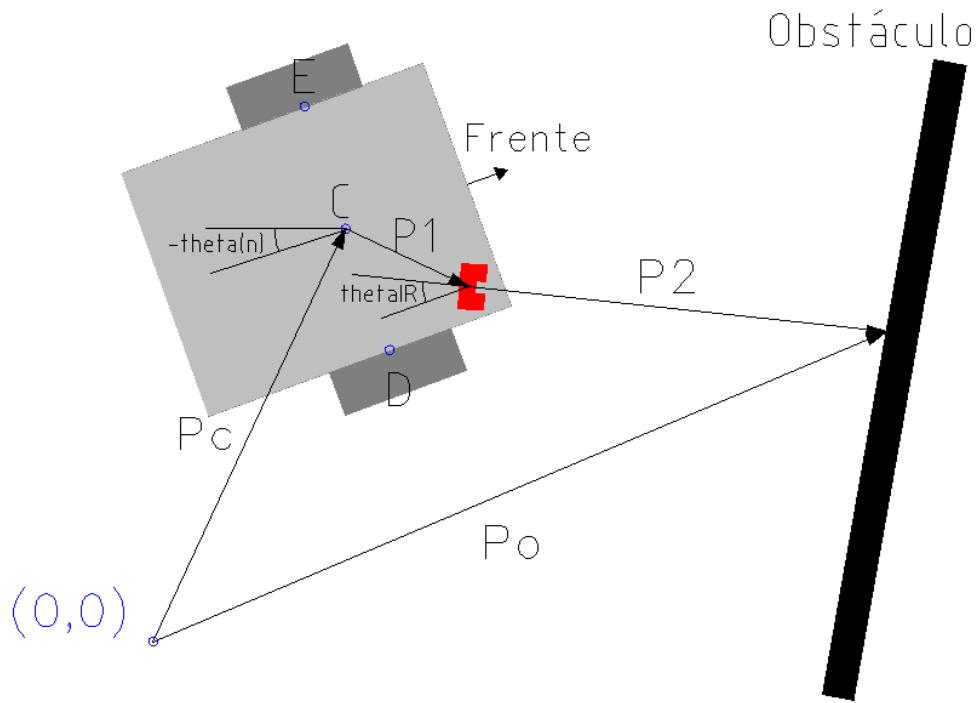


Figura 37: Representação do robô e um sensor infra-vermelho detectando um obstáculo (visão superior).

Para obter a posição em que um obstáculo detectado está no mapa, primeiramente deve-se obter a distância entre o obstáculo e o sensor infra-vermelho. Como explicitado em (MARIN et al., 2012), o valor x recebido em 1 byte do sensor pode ser convertido para a distância d em centímetros pela fórmula (obtida por interpolação polinomial):

$$d = 3,6404 \cdot 10^{-7}x^4 - 2,4435 \cdot 10^{-4}x^3 + 6,0732 \cdot 10^{-2}x^2 - 6,8962x + 339,361 \quad (53)$$

Sendo \vec{P}_C o vetor que sai da origem e vai até o centro de movimento do robô, \vec{P}_1 o vetor que vai do centro do robô até o sensor, e \vec{P}_2 o vetor que vai do sensor até o ponto do obstáculo detectado, faz-se a seguinte soma vetorial para encontrar o vetor \vec{P}_O , que é a posição do obstáculo no mapa:

$$\vec{P}_O = \vec{P}_C + \vec{P}_1 + \vec{P}_2 \quad (54)$$

O vetor \vec{P}_C pode ser facilmente determinado de acordo com a equação abaixo. Sendo $\vec{P}_{(n)}$ a última posição do robô (intervalo discreto n):

$$\vec{P}_C = \vec{P}_{(n)} \quad (55)$$

O vetor \vec{P}_1 é o vetor que vai do centro de movimento do robô até o sensor infra-vermelho. Sendo \vec{P}_{IR} que vai do centro do robô até o sensor (este vetor é especificado nas configurações iniciais, supondo que o robô não esteja rotacionado), e $\theta_{(n)}$ o ângulo em que o robô está orientado em sua última posição (instante discreto n):

$$|\vec{P}_1| = |\vec{P}_{IR}| \quad (56)$$

$$\phi \left\{ \vec{P}_1 \right\} = \phi \left\{ \vec{P}_{IR} \right\} + \theta_{(n)} \quad (57)$$

O vetor \vec{P}_2 é o vetor que vai do sensor infra-vermelho até o obstáculo. Sendo d a distância detectada pelo sensor, θ_{IR} o ângulo em que o sensor infra-vermelho está posicionado relativamente ao robô (este ângulo é especificado nas configurações iniciais, supondo que o robô não esteja rotacionado), e tendo $\theta_{(n)}$ o mesmo significado que na equação anterior:

$$|\vec{P}_2| = d \quad (58)$$

$$\phi \left\{ \vec{P}_2 \right\} = \theta_{IR} + \theta_{(n)} \quad (59)$$

12.4.1 Filtragem

Percebeu-se, nos testes realizados com o robô, que os sensores infra-vermelhos utilizados na prática provêm, esporadicamente, leituras discrepantes e que não correspondem à realidade. Portanto, certos métodos de filtragem digital foram utilizados na estação base. O primeiro filtro é um passa-faixa, que deixar passar apenas medidas que estejam entre 30cm e 150cm, que são as distâncias mínima e máxima que podem ser detectadas com exatidão pelos sensores (segundo o *datasheet* do dispositivo). O segundo filtro é uma média móvel. Esse filtro pode ser ativado pelo usuário através da interface gráfica, na qual pode-se também configurar o número de períodos da média. Em alguns ambientes testados a média móvel mostrou-se adequada para melhorar a qualidade do mapa.

13 TESTES

Nesta seção serão expostos os testes realizados na prática com o robô. Durante os testes foi possível chegar ao ponto de utilizar com sucesso o giroscópio para correção de erros de escorregamentos e de falhas de leituras dos encoders, como explicado a seguir.

Um ponto que vale ser ressaltado, para melhor compreensão dos mapas, é que o robô sempre sai da origem (ponto marcado por $(0,0)$ em cada mapa) e chega na posição em que está o seu ícone (■).

13.1 PRIMEIRO TESTE

O primeiro teste bem-sucedido foi realizado utilizando com o *layout* do ambiente da Figura 38. O robô fez um trajeto em sentido anti-horário no ambiente retangular, voltando à posição inicial $(0,0)$ ao final do caminho. Os encoders sofreram imprecisões nas medições em certos momentos do trajeto. No gráfico da Figura 43 está explicitada uma comparação entre as velocidades angulares obtidas com os encoders e com o eixo Z do giroscópio. Vê-se que o giroscópio proveu medidas razoavelmente confiáveis no decorrer do tempo.

Na Figura 39 está presente o mapa produzido a partir desse teste somente utilizando os dados dos encoders, e na Figura 40 está presente um mapa que foi produzido utilizando as leituras do giroscópio para correção de erros (o limiar utilizado foi $L_\omega = 0.025$ [rad/s]). Vê-se que o trajeto do robô pôde ser determinado com maior exatidão no segundo mapa, usando-se o giroscópio. A posição final do robô ficou próxima da origem, e o mapa ficou mais semelhante à forma retangular que corresponde à realidade.

O acelerômetro, por outro lado, não se provou eficaz para o fim de correção de erros, ao menos não da forma como os dados estão sendo atualmente obtidos e tratados. A obtenção de dados é feita a partir de amostras instantâneas a 3.89 Hz. Na Figura 44 está explicitada uma comparação (produzida a partir desse mesmo teste) das acelerações lineares obtidas com os encoders e com o eixo Y do acelerômetro. Vê-se que há grandes diferenças entre as acelerações,

e que o acelerômetro provê medidas bastante discrepantes. Na Figura 42 está presente um mapa que foi gerado somente a partir das amostras do acelerômetro e giroscópio, exposto para fins de comparação. A trilha em azul representa o caminho que o robô supostamente percorreu de acordo com esses sensores. É visível que os resultados obtidos com o acelerômetro estão longe da realidade.

Um aspecto que vale ressaltar é que nos dois primeiros mapas foi utilizada uma filtragem de média móvel de 3 períodos para os sensores IR. No mapa da Figura 41, exposto para comparação, não foi utilizada a filtragem por média móvel.

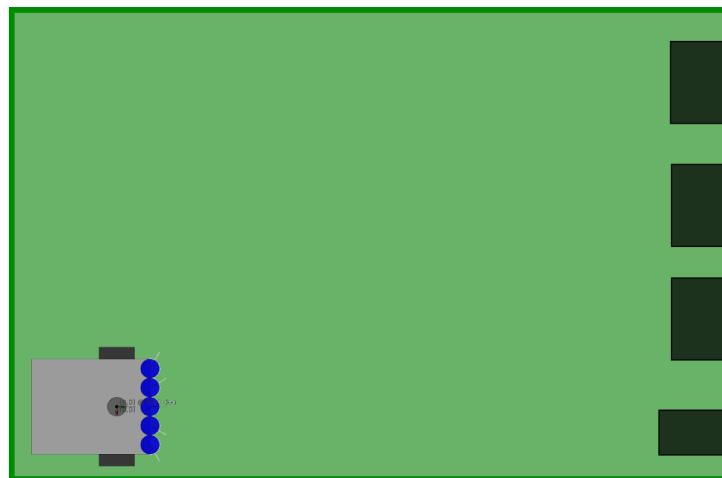


Figura 38: *Layout* do ambiente do primeiro teste.

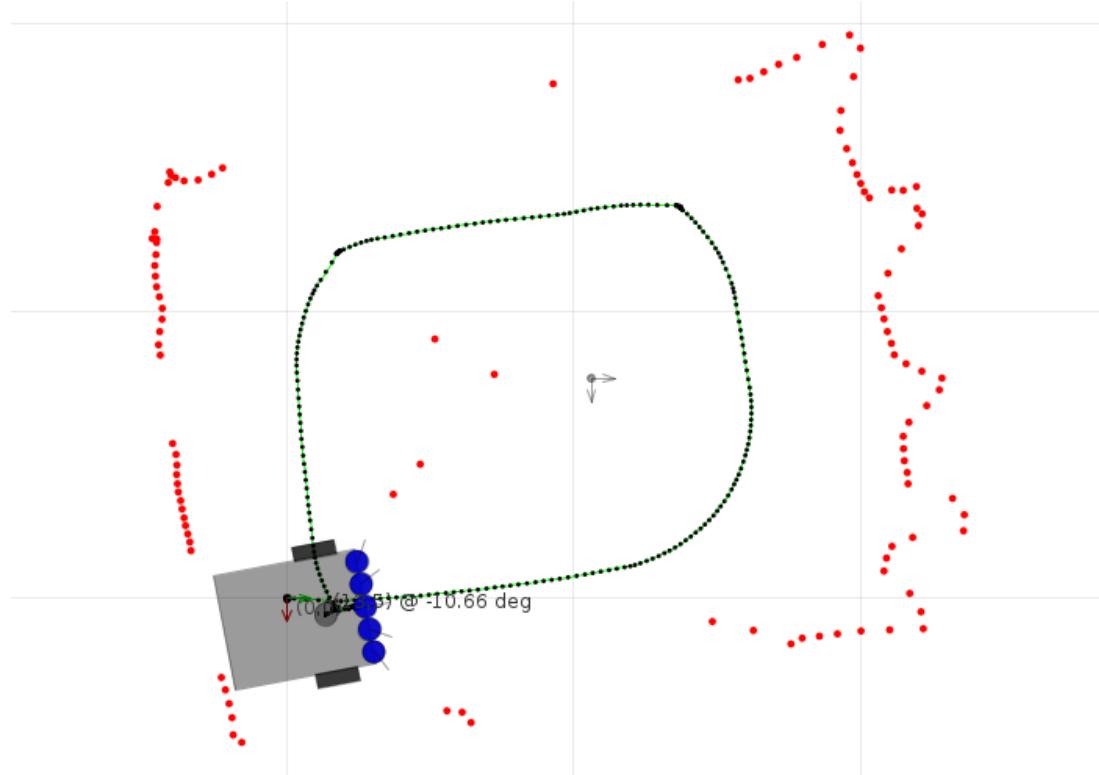


Figura 39: Mapa gerado no primeiro teste a partir dos dados dos encoders usando filtragem por média móvel de 3 períodos para sensores IR.

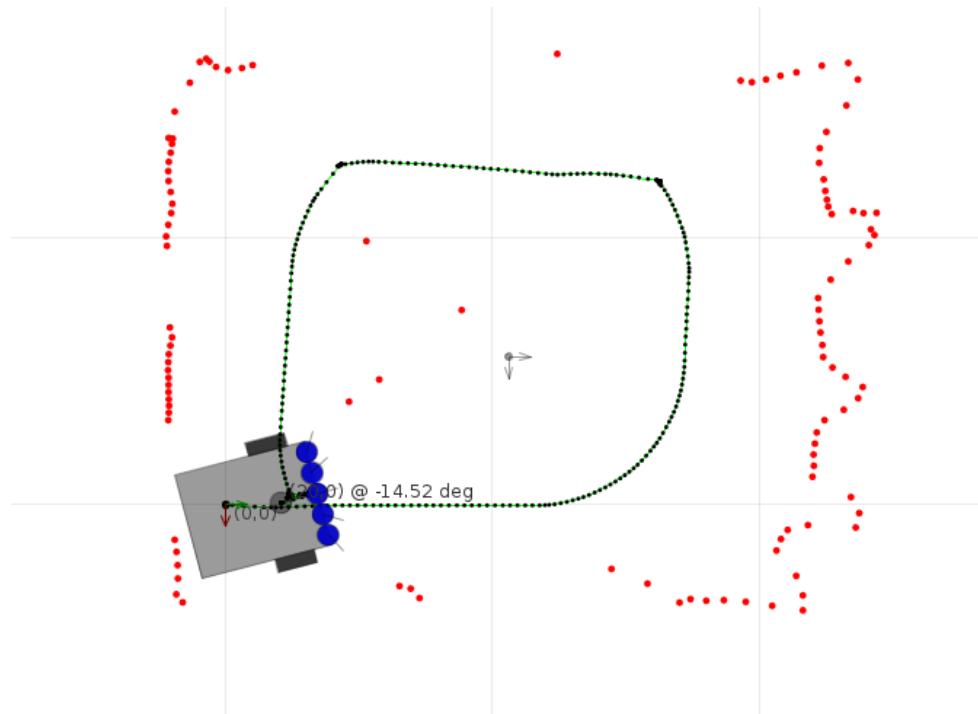


Figura 40: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 3 períodos para sensores IR.

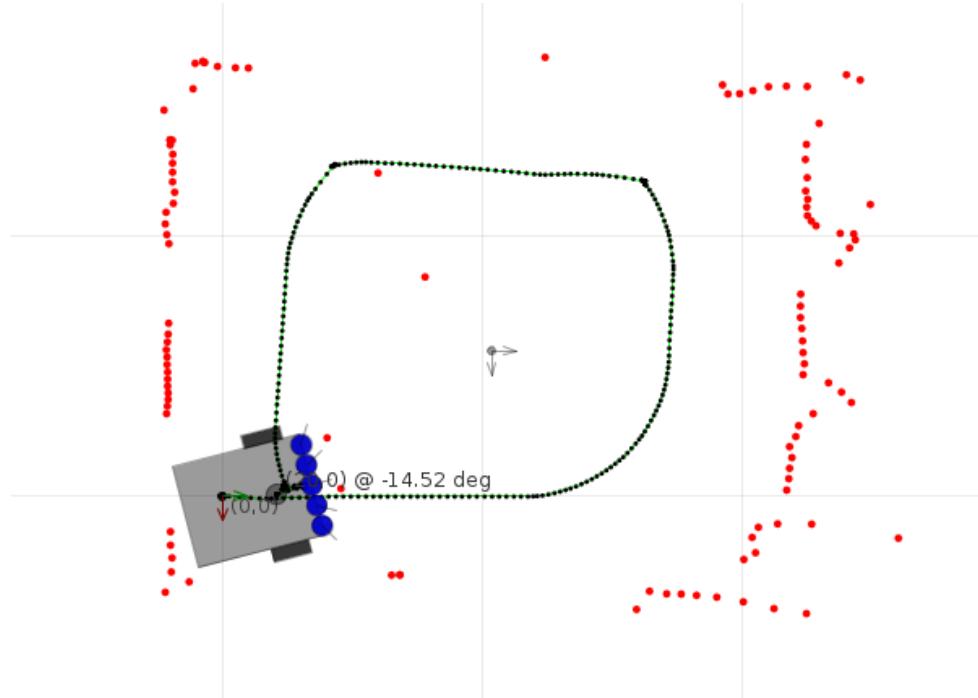


Figura 41: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) sem filtragem por média móvel dos sensores IR.

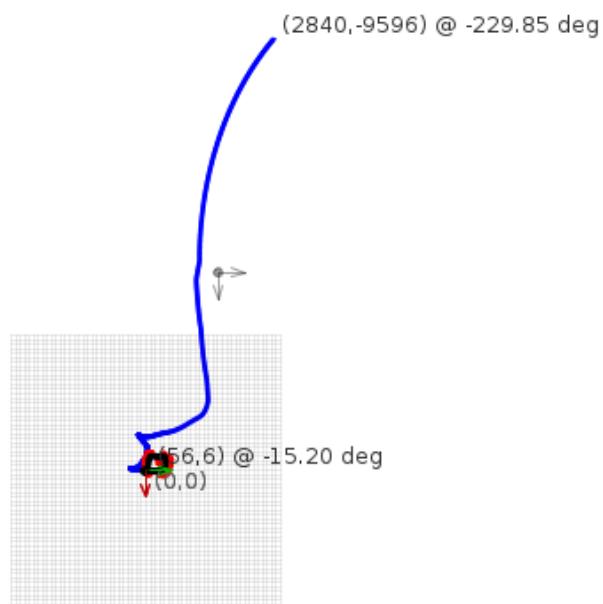


Figura 42: Trilha do robô (em azul) no primeiro teste calculada somente a partir dos dados do acelerômetro e do giroscópio.

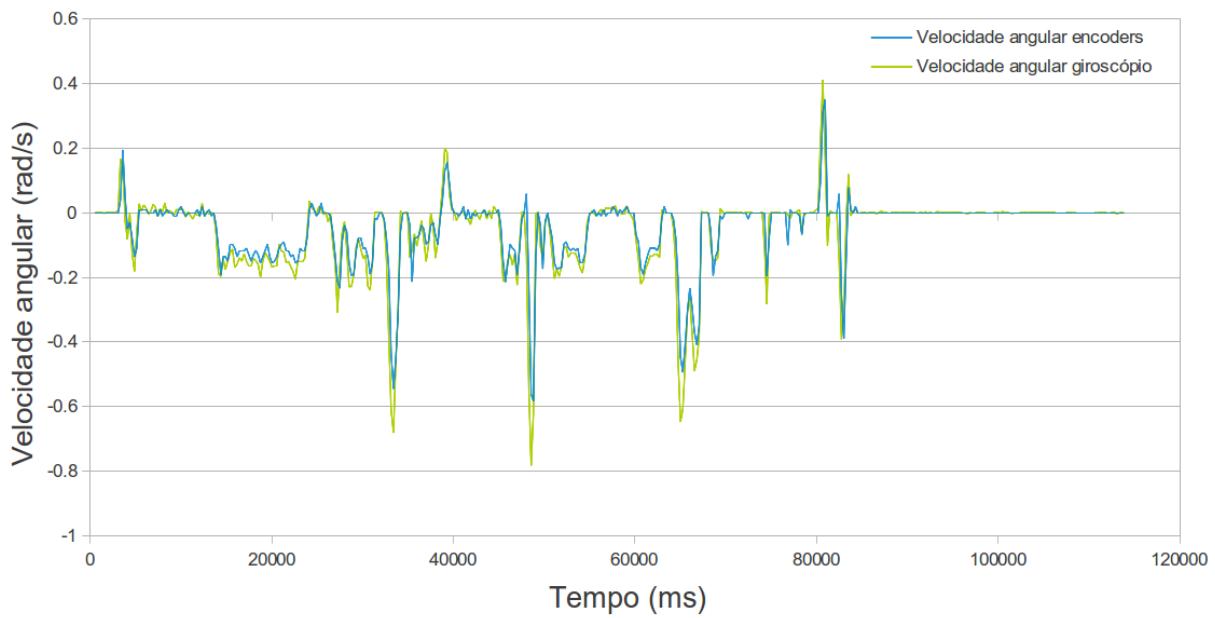


Figura 43: Gráfico comparativo de velocidades angulares obtidas no primeiro teste.

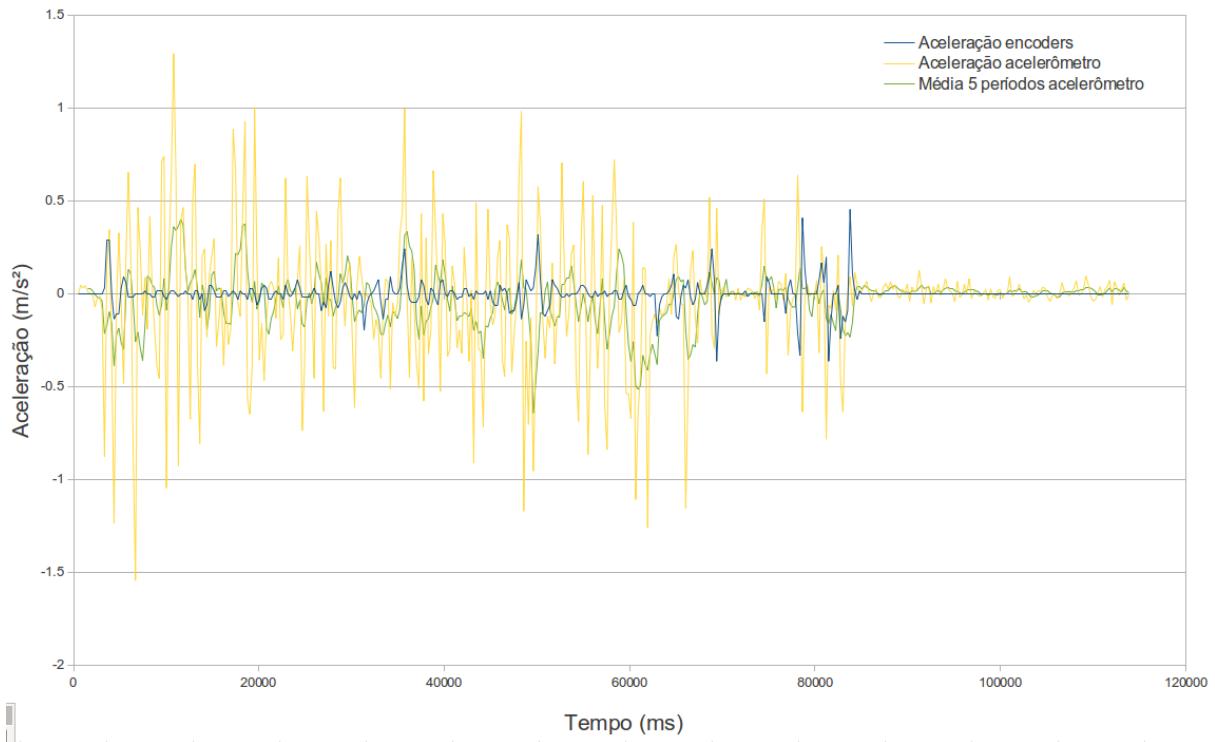


Figura 44: Gráfico comparativo de acelerações obtidas no primeiro teste.

13.2 SEGUNDO TESTE

O segundo teste foi realizado em um corredor de 2,25m de largura, no ambiente da Figura 45. A mochila que pode ser vista na foto foi posicionada no centro do corredor para que a detecção dela fosse feita pelo robô. Na Figura 46, está presente o mapa produzido apenas com dados dos encoders. Vê-se que eles sofreram erros de medição nas curvas. Na Figura 47, pode-se ver um mapa produzido com correções desses erros pelo giroscópio, usando um limiar $L_\omega = 0.0025$ [rad/s].

Nos mapas das Figuras 46 e 47, não foi utilizado nenhum tipo de filtragem (além do passa-faixa de 30 a 150cm) para os sensores infra-vermelhos. O uso de média móvel de 3 períodos (Figura 48) não resultou em diferenças muito consideráveis, e a de 8 períodos (Figura 49) gerou resultados visivelmente piores.

Percebe-se nesse teste, novamente, que o giroscópio foi confiável nas medições de velocidade angular. Já o acelerômetro, assim como no teste anterior, não pôde ser utilizado para correção de erros (pelos mesmos motivos já apresentados). A Figura 50 expõe a trilha do robô gerada apenas com os dados do acelerômetro e do giroscópio. Percebe-se que os resultados obtidos com o acelerômetro saem bastante da realidade.



Figura 45: Ambiente utilizado no segundo teste.

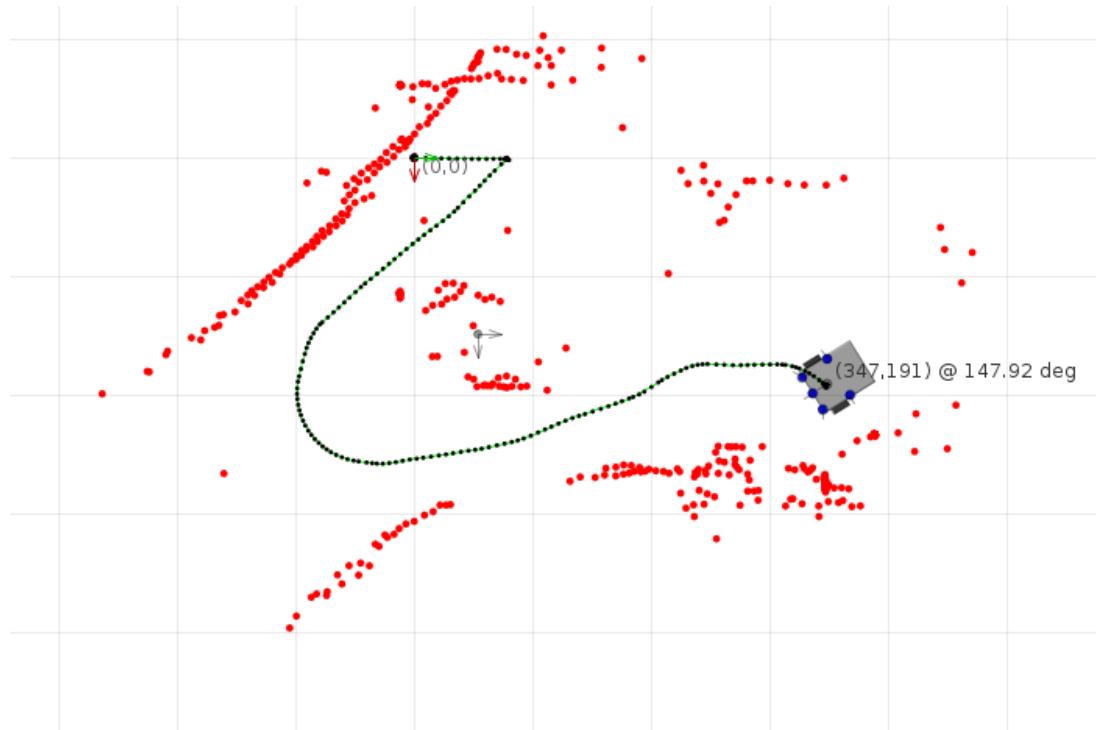


Figura 46: Mapa gerado no segundo teste a partir dos dados dos encoders.

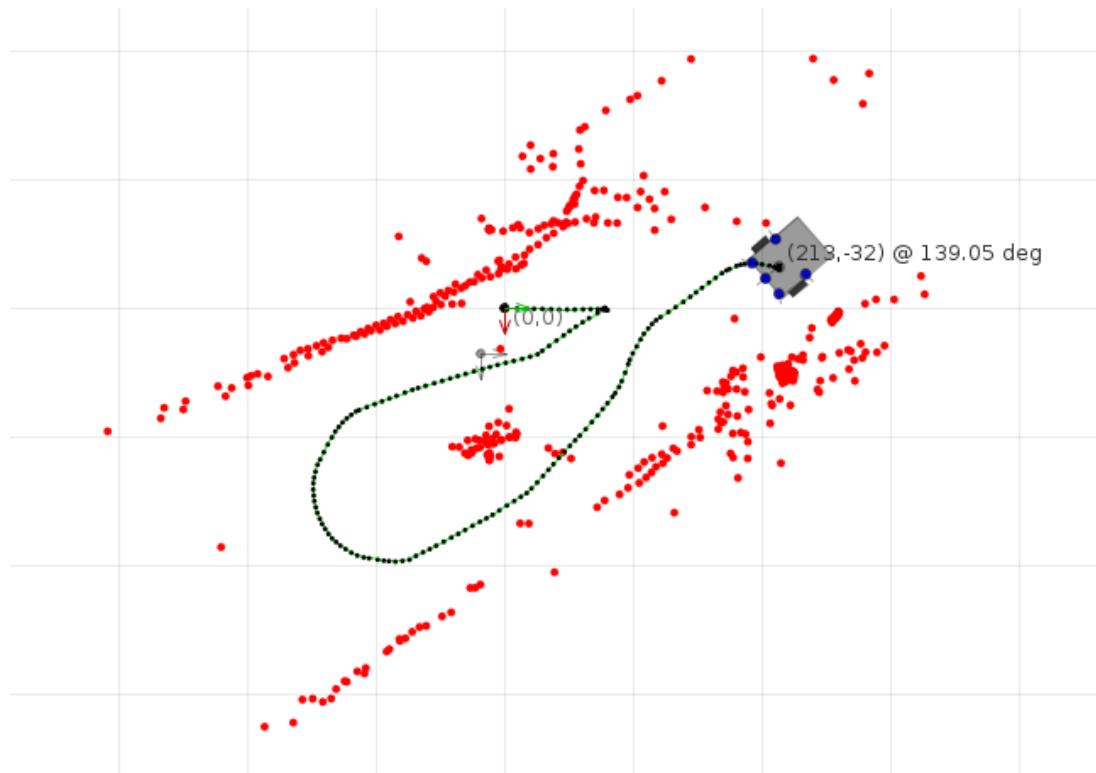


Figura 47: Mapa gerado no segundo teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]).

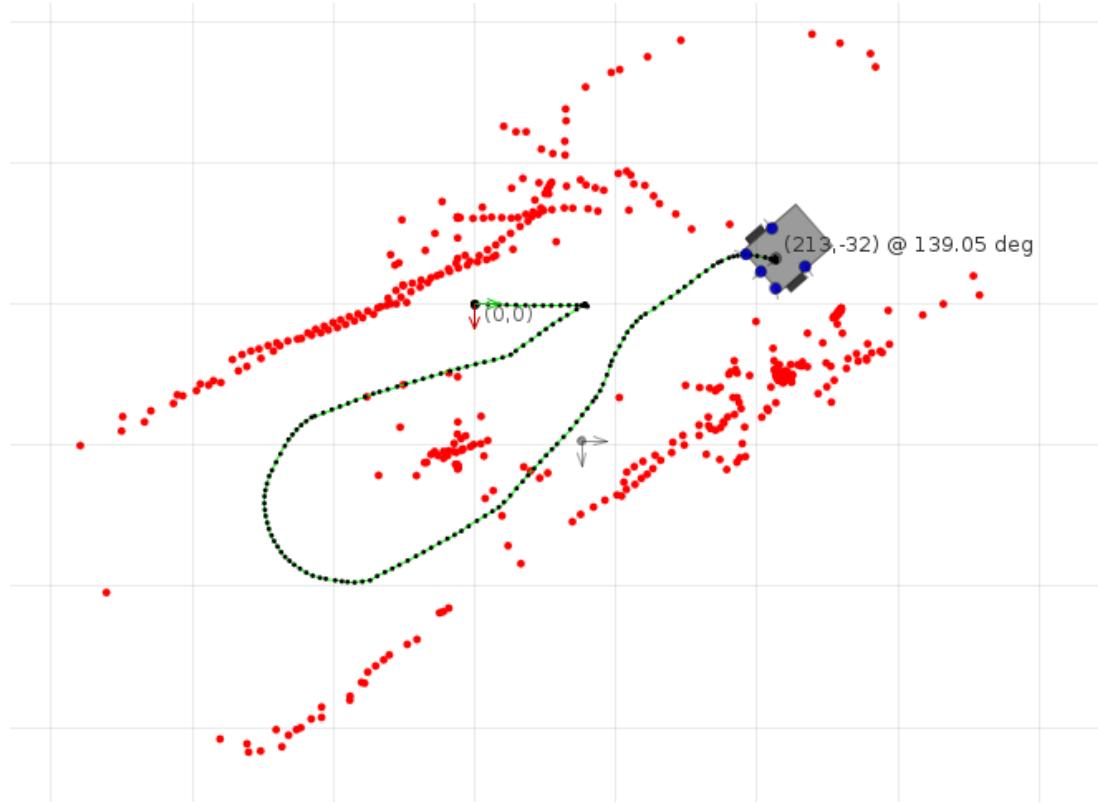


Figura 48: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 3 períodos para sensores IR.

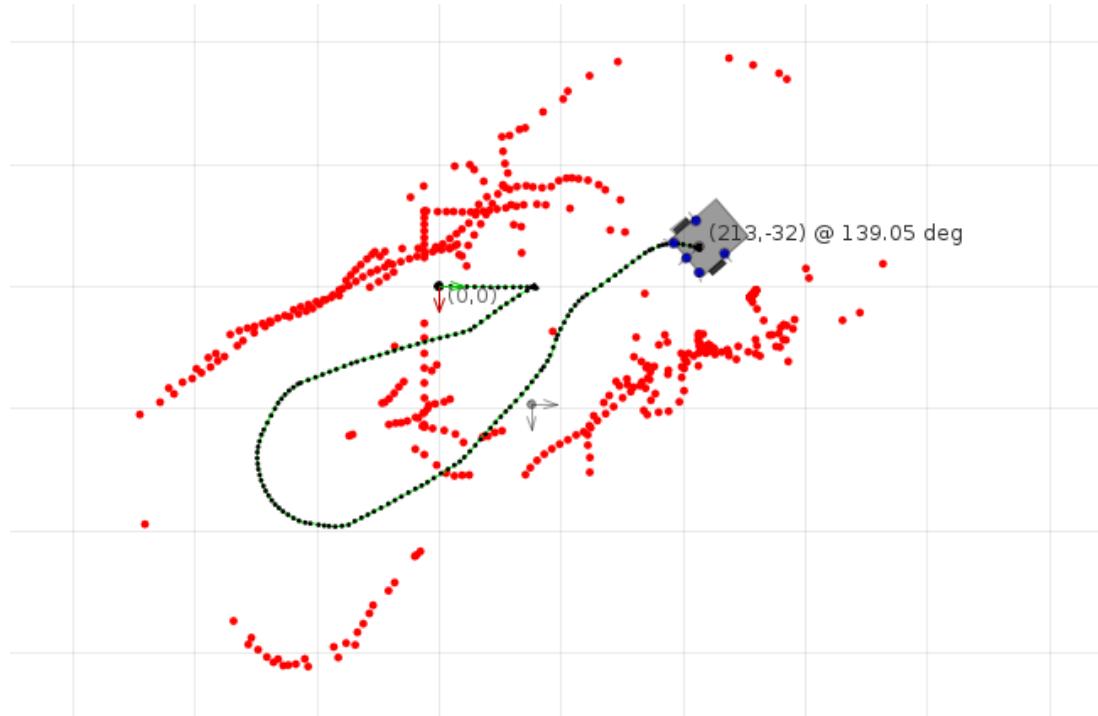


Figura 49: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 8 períodos para sensores IR.

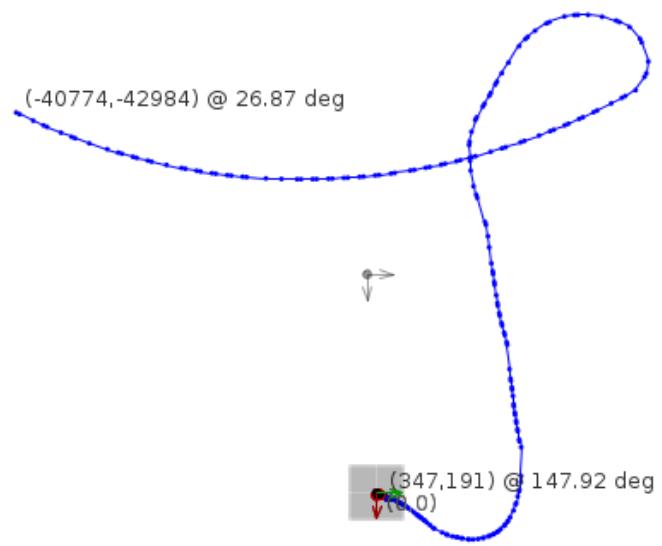


Figura 50: Trilha do robô (em azul) no segundo teste calculada somente a partir dos dados do acelerômetro e do giroscópio.

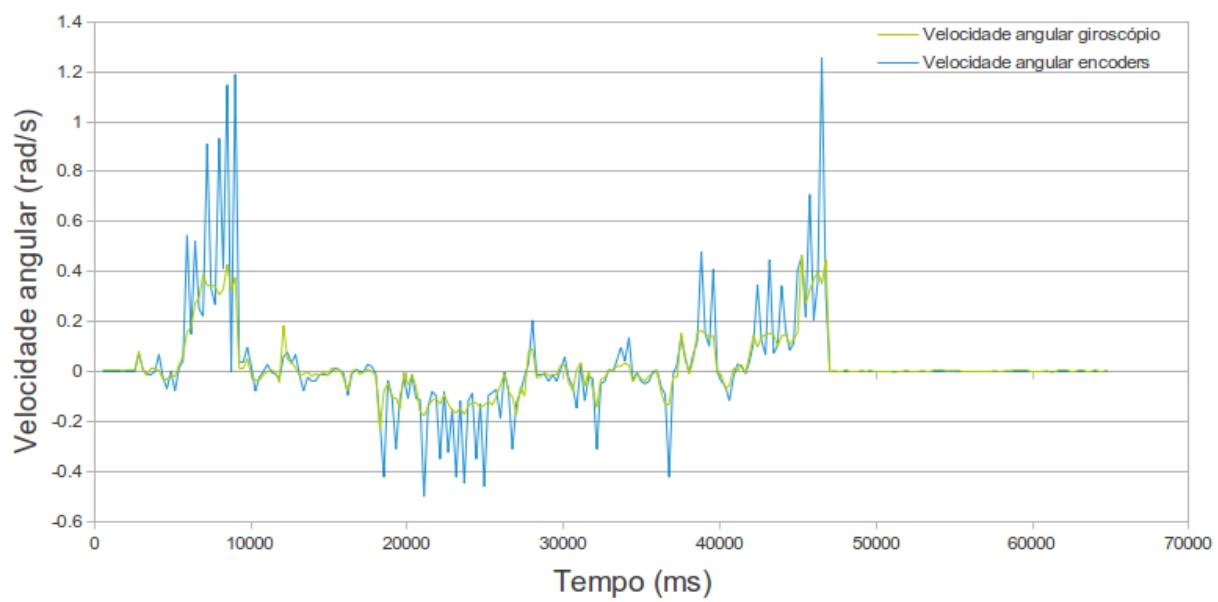


Figura 51: Gráfico comparativo de velocidades angulares obtidas no segundo teste.

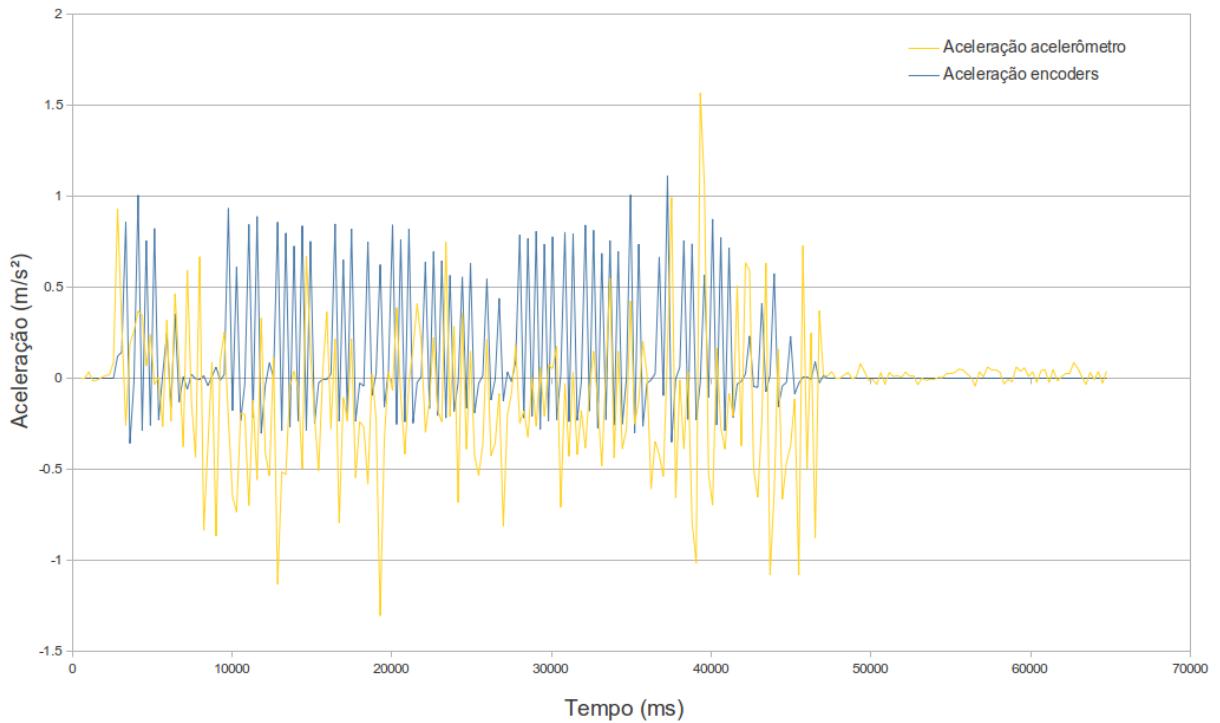


Figura 52: Gráfico comparativo de acelerações obtidas no segundo teste.

13.3 TERCEIRO TESTE

O terceiro teste foi realizado partindo-se de um corredor de 2,25m de largura (à direita na Figura), saindo dele e voltando ao ponto de partida novamente. O ambiente está representado na Figura 53. Na Figura 54, está presente o mapa produzido apenas com dados dos encoders. Vê-se que eles sofreram erros de medição nas curvas, assim como no teste anterior. Na Figura 55, pode-se ver um mapa produzido com correções desses erros pelo giroscópio, usando um limiar $L_\omega = 0.0025$ [rad/s].

Nos mapas das Figuras 54 e 55, não foi utilizado nenhum tipo de filtragem (além do passa-faixa de 30 a 150cm) para os sensores infra-vermelhos. O uso de média móvel de 5 períodos (Figura 56), a melhor configuração encontrada, resultou em leves melhorias na qualidade do mapa.

Percebe-se nesse teste, novamente, que o giroscópio foi confiável nas medições de velocidade angular. Já o acelerômetro, assim como nos testes anteriores, não pôde ser utilizado para correção de erros (pelos mesmos motivos já apresentados). A Figura 57 expõe a trilha do robô gerada apenas com os dados do acelerômetro e do giroscópio. Percebe-se novamente que os resultados obtidos com o acelerômetro fogem bastante da realidade.



Figura 53: Ambiente utilizado no terceiro teste.

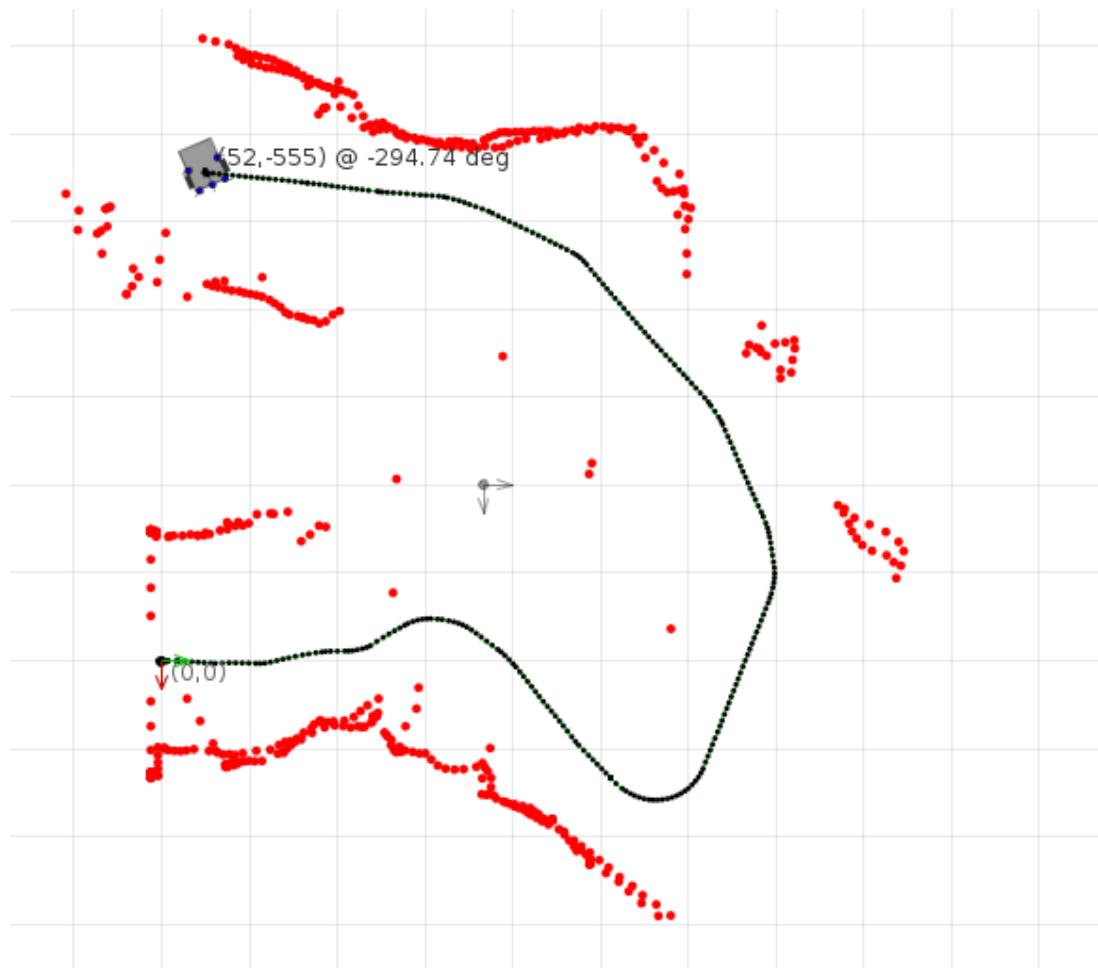


Figura 54: Mapa gerado no terceiro teste a partir dos dados dos encoders.

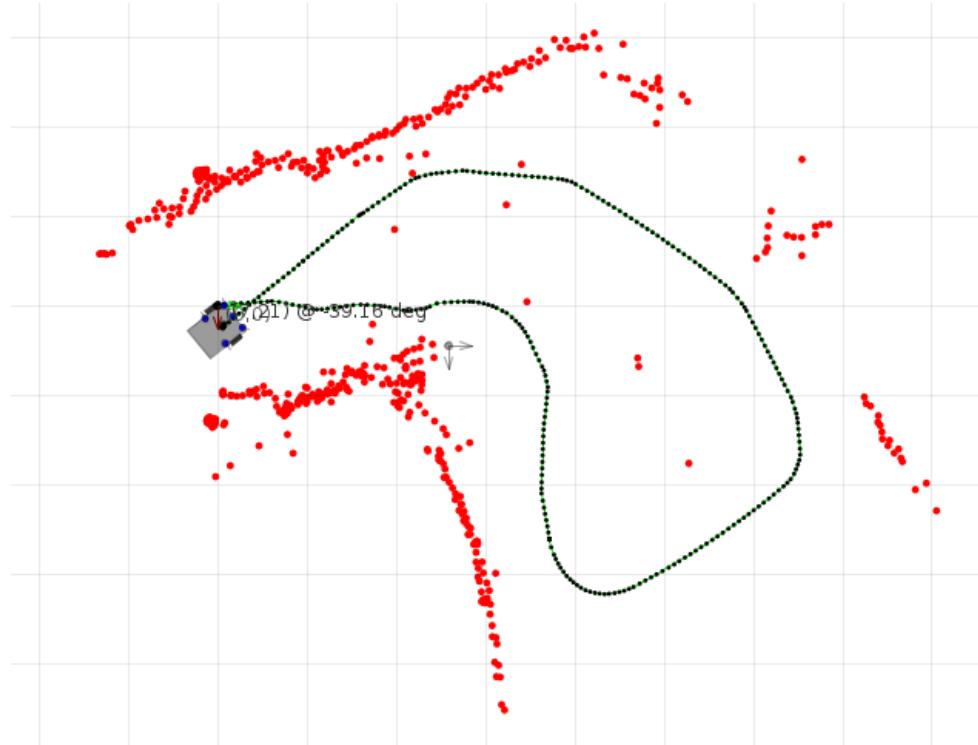


Figura 55: Mapa gerado no terceiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]).

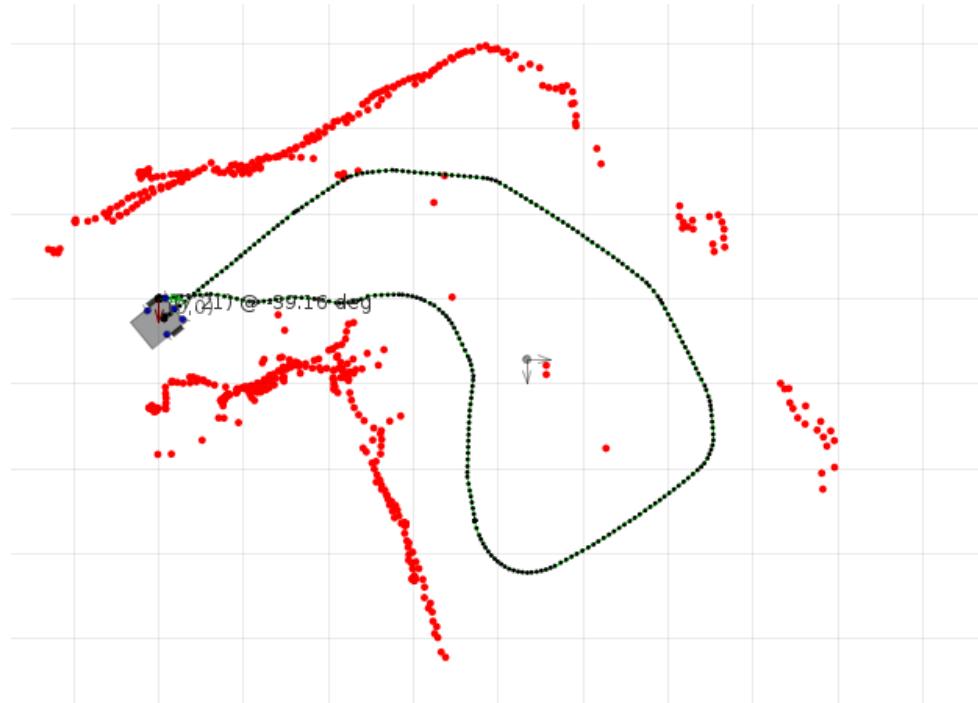


Figura 56: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.0025$ [rad/s]) e filtragem por média móvel de 5 períodos para sensores IR.



Figura 57: Trilha do robô (em azul) no terceiro teste calculada somente a partir dos dados do acelerômetro e do giroscópio.

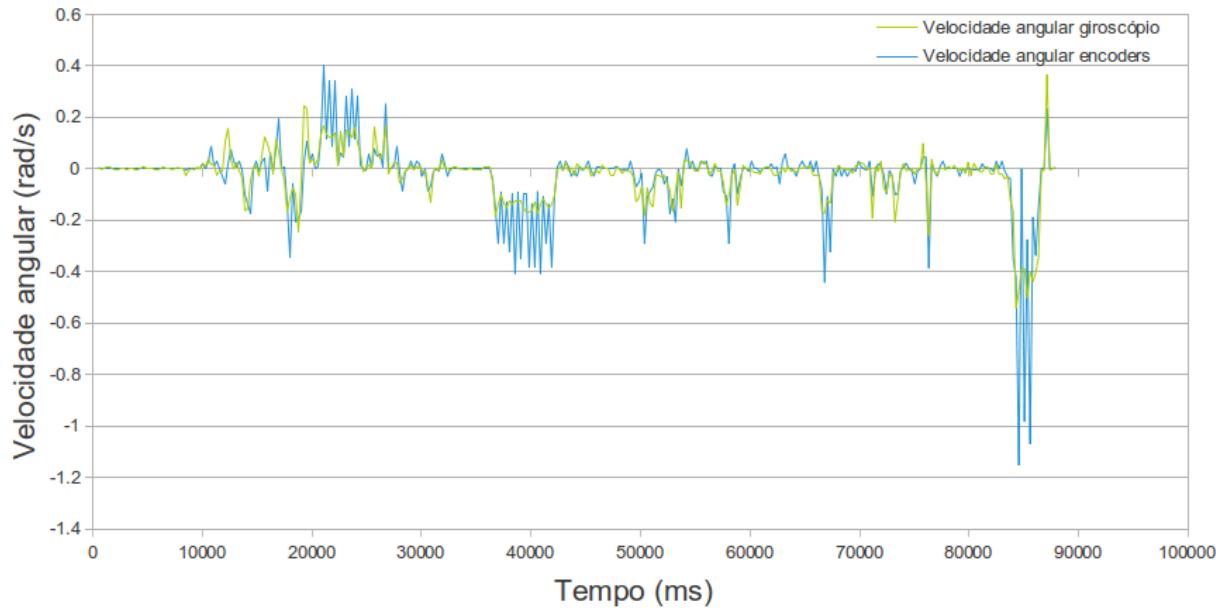


Figura 58: Gráfico comparativo de velocidades angulares obtidas no terceiro teste.

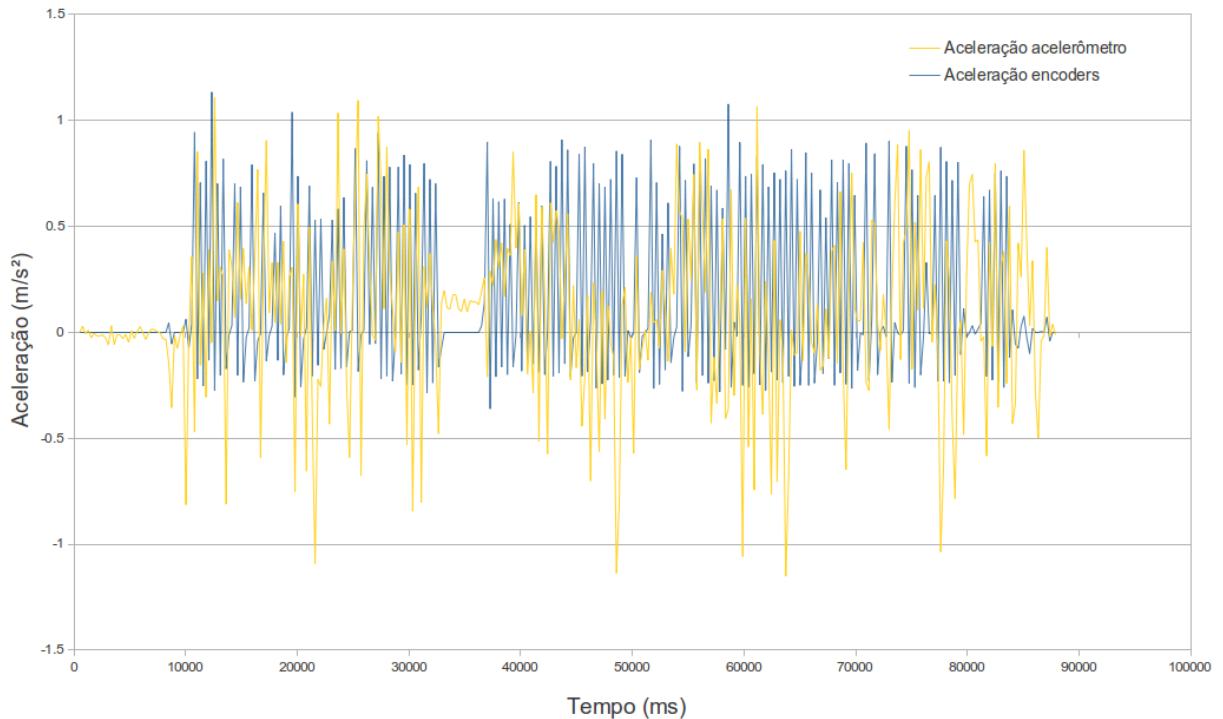


Figura 59: Gráfico comparativo de acelerações obtidas no terceiro teste.

13.4 QUARTO TESTE

O quarto teste foi realizado na extensão do corredor principal do terceiro andar dos blocos A, B, C e D da UTFPR campus Curitiba. O ambiente está representado na Figura 60. Um mapa produzido apenas com dados dos encoders está explicitado na Figura 61. Vê-se que os encoders sofreram erros de medição, pois o mapa aparece de forma curvada. Na Figura 62, pode-se ver um mapa produzido com correções desses erros pelo giroscópio, usando um limiar $L_\omega = 0.003$ [rad/s].

Nos mapas das Figuras 61 e 62, não foi utilizado nenhum tipo de filtragem (além do passa-faixa de 30 a 150cm) para os sensores infra-vermelhos. O uso de média móvel de 4 períodos (Figura 63), a melhor configuração encontrada, resultou em leves melhorias na qualidade do mapa.

Percebe-se nesse teste, novamente, que o giroscópio foi confiável nas medições de velocidade angular. Já o acelerômetro, assim como nos testes anteriores, não pôde ser utilizado para correção de erros pelos mesmos motivos já apresentados.



Figura 60: Ambiente utilizado no quarto teste.

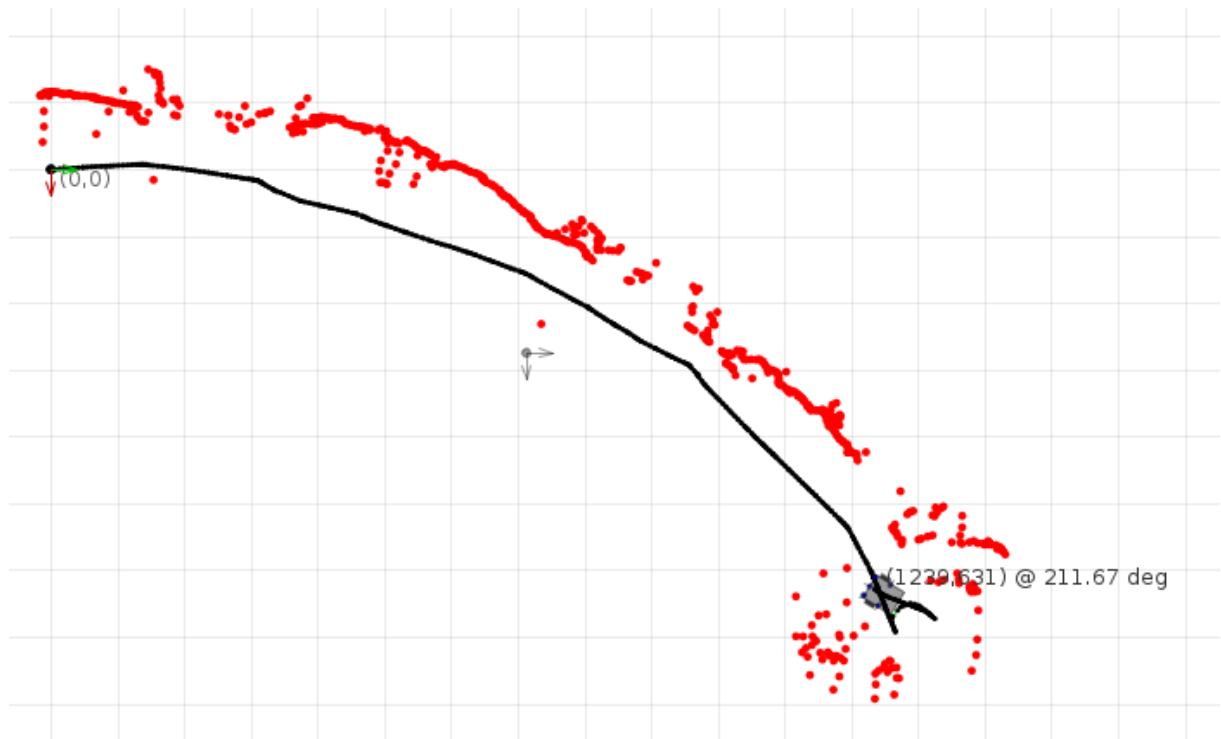


Figura 61: Mapa gerado no quarto teste a partir dos dados dos encoders.

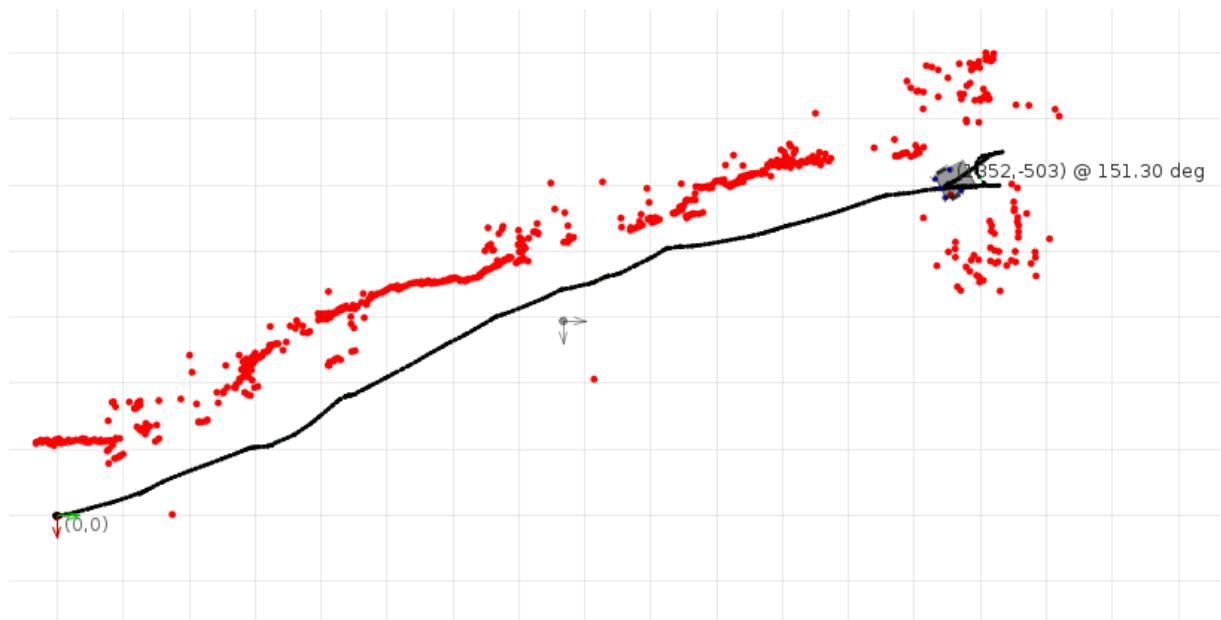


Figura 62: Mapa gerado no quarto teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.003$ [rad/s]).

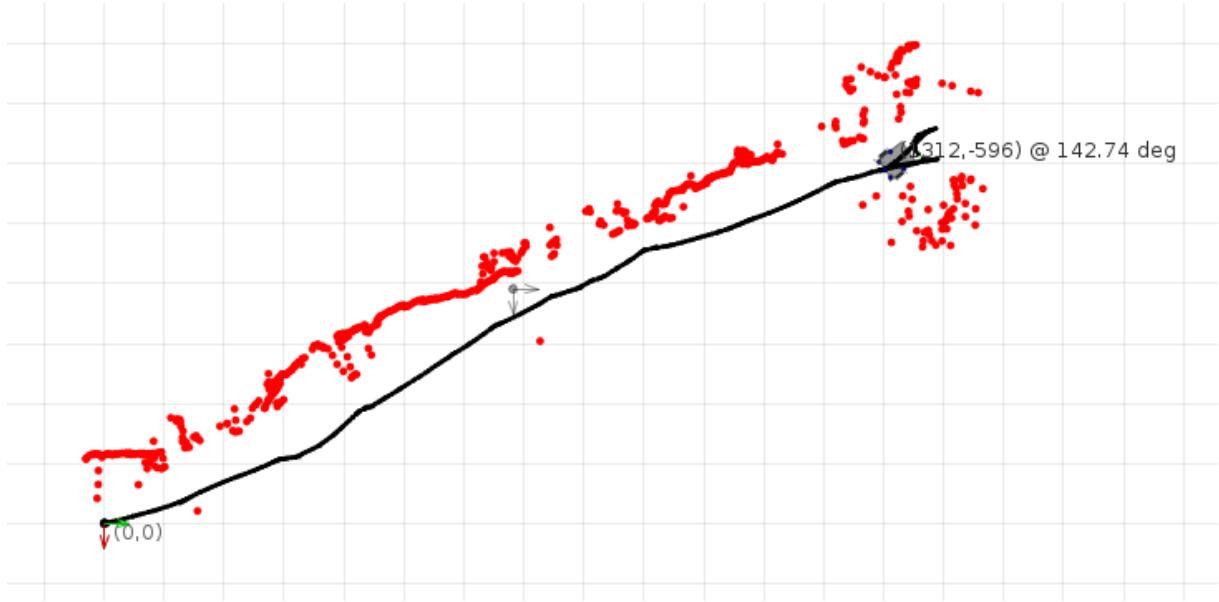


Figura 63: Mapa gerado no primeiro teste a partir dos dados dos encoders, com correção de erros pelo giroscópio ($L_\omega = 0.003$ [rad/s]) e filtragem por média móvel de 4 períodos para sensores IR.

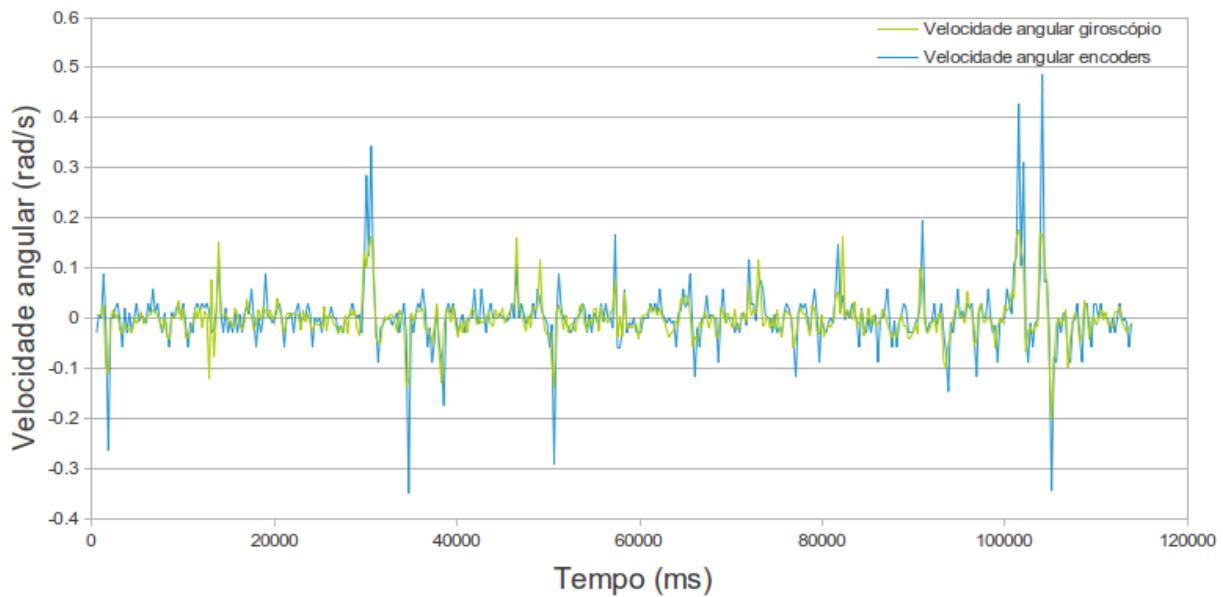


Figura 64: Gráfico comparativo de velocidades angulares obtidas no quarto teste.

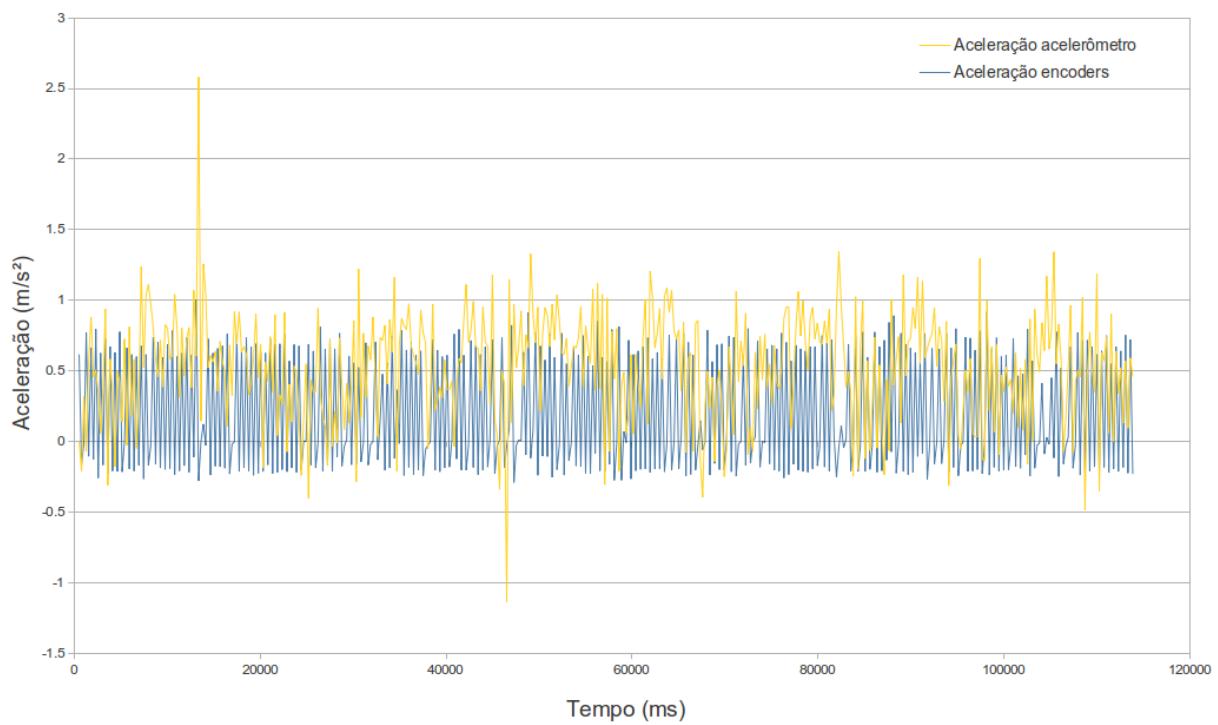


Figura 65: Gráfico comparativo de acelerações obtidas no quarto teste.

14 CONCLUSÕES

Pôde ser visto nos testes apresentados que os encoders sofreram falhas nas medidas, sobretudo na realização de curvas. Outro aspecto percebido também pela equipe é que erros de medição ocorrem frequentemente quando o robô inicia o movimento de uma roda parada, ou quando inverte o sentido de rotação abruptamente. Uma das possíveis causas dos erros podem ser os discos dos encoders, que estão visivelmente danificados em certos pontos. Eles deveriam gerar, nominalmente, 1800 pulsos por volta, mas geram na prática aproximadamente 1600 pulsos. Outra causa provável é que a correia utilizada para fazer o acoplamento do eixo da roda com o eixo do encoder é de borracha, e frequentemente está sujeita a escorregamentos. A equipe efetuou tentativas de reduzir esse problema utilizando uma polia maior (e com mais aderência) no eixo dos encoders, e aumentando a aderência do eixo das rodas aplicando a ele uma fita de borracha. Porém, ainda sim os problemas persistiram.

Uma alternativa viável seria utilizar outro disco para os encoders, com menos pulsos por volta e que sejam menos suscetíveis a falhas de leitura. Além disso, para acoplar o encoder às rodas, uma solução pode ser a utilização de uma correia dentada, que é menos sujeita a escorregamentos que uma correia de borracha. Outra opção seria acoplar os discos dos encoders diretamente ao eixo das rodas, dessa maneira eliminando o problema das correias.

Um aspecto que foi bem-sucedido nos testes foi a utilização do giroscópio para a correção de erros dos encoders. Sobretudo o giroscópio foi de grande valia para determinar com maior exatidão a orientação do robô na realização de curvas. Um aspecto interessante que pode ser trabalhado em projetos futuros é a determinação do limiar L_ω para cada caso específico. Nos testes aqui apresentados, o valor foi determinado empiricamente em cada teste, mas seria interessante que houvesse um trabalho no desenvolvimento de um algoritmo para determinar dinamicamente o valor de L_ω , adequando-o a cada caso. Possivelmente o filtro de Kalman seja outra opção para corrigir os erros dinamicamente. Porém, conhecidamente, a complexidade de utilização dele na prática é considerável.

Percebeu-se que o acelerômetro não foi eficaz para o fim de correção de erros de odo-

metria, ao menos não da forma como os dados estão sendo atualmente obtidos e tratados. A obtenção de dados é feita a partir de amostras instantâneas a 3.89 Hz. No escopo do projeto (tendo em vista as limitações de tempo) não foi possível trabalhar mais à fundo com os dados do acelerômetro. Porém, deixa-se a possibilidade para trabalhos futuros. Como alternativa de tratamento desses dados, possivelmente possa ser utilizada uma taxa de amostragem maior para o acelerômetro, utilizando um processo de filtragem para reduzir os ruídos da aceleração.

Do ponto de vista dos objetivos, percebe-se que a maioria deles foram alcançados. Foi implementado com sucesso um *software* com interface gráfica para geração e visualização de mapas, visualização de imagens da *webcam* do robô e controle do robô por meio do teclado. A comunicação foi implementada satisfatoriamente, tanto entre a estação base e a placa TS-7260 (via Wi-Fi) quanto entre a TS e a placa de baixo nível, via porta serial. Uma *webcam* USB foi instalada e configurada com sucesso no robô para a transmissão de imagens ao usuário. O giroscópio foi utilizado satisfatoriamente na prática para corrigir erros de leituras dos encoders e escorregamento das rodas. Uma placa de circuito impresso, de tamanho reduzido, foi desenvolvida de forma a integrar as funções de baixo nível do robô: interface com os encoders, sensores infra-vermelhos, acelerômetro e giroscópio, e geração de PWM para os motores.

Quanto aos riscos, observou-se que três deles ocorreram. O primeiro foi a “Não entrega de componentes ou entrega fora do prazo”. Uma unidade MPU-6050 (acelerômetro e giroscópio) encomendada no site eBay não foi entregue da maneira especificada e foi devolvida. Como a encomenda desse item havia sido adiantada, realizou-se uma nova encomenda no site Sparkfun, que chegou no prazo estabelecido não causando atraso no projeto. O segundo risco foi “Taxação dos componentes comprados no exterior”, que gerou um aumento no preço dos componentes, mas não de forma que ultrapassasse o orçamento especificado. O terceiro e último risco foi “Problemas Técnicos com o Hardware”. Como relatado, a imprecisão da leitura dos encoders ópticos interferiu na correta determinação da posição do robô. Como solução, buscou-se aumentar a aderência das rodas e correias do robô a fim de diminuir os escorregamentos e falsas leituras, além de utilizar o giroscópio para correção do posicionamento.

Como última consideração, enfatiza-se que este projeto não chegou ao seu fim, mas pelo contrário, um novo caminho foi aberto para pesquisas mais profundadas no que tange ao mapeamento 2D de ambientes. A possibilidade está aberta a novas equipes para utilizarem a plataforma de mapeamento desenvolvida para o Bellator – o que envolve tanto o *hardware* quanto o *software*. Apesar deste trabalho ter sido consideravelmente extenso e demandar muito tempo e energia de todos os membros da equipe, pode-se dizer que com ele o ponta-pé inicial foi dado, e que ainda existem muitas possibilidades a serem exploradas.

15 TRABALHOS FUTUROS

Há inúmeros modos de se prosseguir com o projeto do robô Bellator, pois muitas melhorias podem ser efetuadas a fim de se atingir resultados mais precisos em termos de mapeamentos de ambientes.

No que diz respeito ao sensoriamento, é importante melhorar a confiabilidades dos dados dos encoders. Uma das formas de se fazer isso é utilizando um *timing belt*, que é uma roda dentada acoplada a um sensor. Dessa forma, problemas de escorregamento da correia em relação à polias do robô seriam eliminados.

Quanto ao uso do acelerômetro, a técnica do filtro de Kalman poderia ser explorada se forma a se realizar a fusão de dados dos três sensores do robô: encoder, acelerômetro e giroscópio. Se desenvolvida corretamente, em tese, a utilização desta técnica permitiria a compensação automática de erros dos sensores o que permitiria uma estimativa muito melhor do posicionamento do robô. As dificuldades de se utilizar esta técnica são: correta determinação da margem de erro de cada sensor, criação de um modelo matemático da movimentação do robô e complexidade do entendimento e implementação do filtro.

Em relação à câmera, há dois aspectos a serem melhorados: a redução do atraso do sinal de vídeo (que é da ordem de segundos) e a interação do usuário com a câmera. O primeiro problema poderia ser resolvido com a substituição da *webcam* atual por outra que possua menos latência, ou pelo desenvolvimentos de software que faça a leitura dos dados brutos da câmera e faça o envio instantâneo desses dados à estação base, procurando reduzir ou eliminar a utilização de *buffers* de vídeo. Em termos de interatividade, pode-se construir um dispositivo de rotação que permita ao usuário girar a câmera 360 graus em torno do eixo vertical, possibilitando maior flexibilidade e autonomia durante a utilização do robô.

APÊNDICE A – PLANEJAMENTO DE RISCOS

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Desistência de Membros da Equipe ou Redução do número de integrantes da equipe por força maior	Nº Identificação 01
Descrição do risco: Possibilidade da desistência de membros da equipe, seja de um único membro até todos os cinco.	

2º Etapa: Avaliação do Risco

Impacto:	<input checked="" type="radio"/> 5 (alta) <input type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: Caso o risco venha a ocorrer, o cronograma e a qualidade do projeto serão afetados. Toda a carga de trabalho deverá ser redividida e o cronograma refeito. Dependendo da quantidade de trabalho restante, o escopo do projeto terá que ser reavaliado. Caso muitos membros desistam e a carga de trabalho remanescente seja muito grande para completar o projeto, caberá aos membros documentar o que foi feito e tentar terminar o que for possível para que uma outra equipe de continuidade em outra ocasião.	
Probabilidade:	<input checked="" type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input checked="" type="radio"/> 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: Os integrantes da equipe terão, na medida do possível, suas tarefas e cronogramas definidos de forma que não haja sobrecarga de trabalho (levando em conta, conjuntamente, as outras matérias e/ou compromissos). A motivação da equipe será também levada bastante em consideração.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: O gerente fará reuniões constantes do grupo para análise da contribuição e desempenho de cada integrante. O gerente também monitorará o andamento do projeto, efetuando ajustes no ritmo de trabalho (caso necessário) para que não haja sobrecarga de trabalho para um único membro da equipe. Todos na equipe deverão se comunicar frequentemente.	
Mitigar: Caso haja desistência de algum membro da equipe, a carga de trabalho deve ser reajustada entre os remanescentes. O cronograma deve ser modificado para que as atividades restantes sejam acomodadas dentro do possível para cada membro. Caso não haja possibilidade de realizar todas as atividades no tempo disponível, deve-se reanalisar o escopo do projeto e direcionar esforços nas áreas mais importantes. Caso muitos membros desistam, consequentemente impossibilitando a completude do trabalho, caberá aos remanescentes documentar o que foi realizado até o momento para que outras equipes, caso venham a dar continuidade ao projeto, possam obter máximo de informações possíveis.	
Impacto reavaliado: 3	Probabilidade reavaliada: 2

Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo. Data: 31/01/2013

Respostas incluídas na WBS/Cronograma

Registros adicionais:
Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Problemas Técnicos com o <i>Hardware</i>	Nº Identificação 02
Descrição do risco: Possibilidade do hardware apresentar problemas	

2º Etapa: Avaliação do Risco

Impacto:	<input type="radio"/> 5 (alta) <input checked="" type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: Dependendo do tipo de problema apresentado pelo <i>hardware</i> , reparos poderão ser realizados sem grandes alterações no cronograma, qualidade ou custo do projeto. Caso o problema seja de difícil identificação ou se for requerida reposição de uma peça, o fato deverá ser levado em consideração para que sejam alocados os recursos necessários para a realização do reparo.	
Probabilidade:	<input type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input checked="" type="radio"/> 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: Problemas com peças de <i>hardware</i> são comuns em diversos aparelhos eletrônicos. Como o projeto apresenta vários componentes eletrônicos diversos, há considerável probabilidade de ocorrerem falhas.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: Manter um estoque de peças reserva para a eventualidade de alguma falhar. Manter contato com professores que possam auxiliar caso o problema seja de difícil identificação ou conserto. Compra de peças de qualidade. Cautela ao manusear o hardware. Manter documentação com alterações e testes no hardware para facilitar identificação caso problemas ocorram. Manter vídeos do hardware funcionando caso ele venha a falhar nas demonstrações do projeto.	
Mitigar: Caso algum problema técnico venha acontecer, os membros da equipe devem primeiro tentar identificar o problema e consertá-lo. Caso não seja possível identificar ou consertar de maneira rápida, serão alocados os recursos necessários para que os reparos sejam feitos pelos membros da equipe ou por um terceiro especializado.	
Impacto reavaliado: 3	Probabilidade reavaliada: 2
Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.	Data: 31/01/2013
	<input type="checkbox"/> Respostas incluídas na WBS/Cronograma
	Registros adicionais: Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Possibilidade do <i>software</i> apresentar problemas.	Nº Identificação 03
Descrição do risco: Problemas decorrentes da programação ou dificuldades técnicas relacionadas ao programa desenvolvido	

2º Etapa: Avaliação do Risco

Impacto:	<input type="radio"/> 5 (alta) <input checked="" type="radio"/> 4 (média/alta) ● 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: Dependendo do problema apresentado pelo <i>software</i> , o mesmo poderá ser reparado sem grandes alterações no cronograma, qualidade ou custo do projeto. Caso o problema seja de difícil identificação e conserto, o fato deverá ser informado ao gerente para que sejam alocados os recursos necessários (tempo, dinheiro, etc...) para que o conserto seja realizado.	
Probabilidade:	<input type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) ● 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: Problemas com <i>software</i> são relativamente comuns de ocorrerem. Como muitas vezes não é possível realizar testes extensivos para garantir a corretude do programa, há probabilidade mediana de serem encontrados <i>bugs</i> ao ser utilizado o <i>software</i> .	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: Manter um repositório de versões do software, com adições, remoções, testes e correções documentadas. Realizar backup do software entre os membros da equipe e em repositórios online.	
Mitigar: Caso algum problema técnico venha acontecer, os membros da equipe devem primeiro tentar identificar o problema e repará-lo. Caso não seja possível identificar ou consertar de maneira rápida, serão alocados os recursos necessários para que o conserto seja feito pelos membros da equipe ou um terceiro especializado. Em caso de urgência, utilizar a versão funcional mais atual. Ter em vídeo o software detalhado e funcionando também é uma medida pertinente no caso de haver falhas na apresentação do projeto.	
Impacto reavaliado: 2	Probabilidade reavaliada: 2
Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo	Data: 31/01/2013
	<input type="checkbox"/> Respostas incluídas na WBS/Cronograma
	Registros adicionais: Verso ou Anexos

Formulário sugerido por Gasnier, 2000 Editora IMAN e alterado por Wille

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Não entrega de componentes ou entrega fora do prazo.	Nº Identificação 04
Descrição do risco: Alguns dos componentes a serem utilizados para o desenvolvimento do projeto deverão ser comprados no exterior por serem inexistentes no Brasil ou muito mais caros de se comprar por aqui. Importar componentes gera incerteza em relação ao prazo de entrega.	

2º Etapa: Avaliação do Risco

Impacto:	<input checked="" type="radio"/> 5 (alta) <input type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique:	Sem os componentes não é possível montar o <i>hardware</i> do projeto.
Probabilidade:	<input type="radio"/> 5 (alto) <input checked="" type="radio"/> 4 (médio/alto) <input type="radio"/> 3 (média) <input checked="" type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique:	Lojas internacionais raramente deixam de enviar o produto comprado. Se houver algum problema de entrega, é mais provável que ocorra devido às transportadoras ou demora na liberação do produto pelas alfândegas brasileiras.

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: Comprar de lojas confiáveis no exterior, que apresentem boa reputação e de preferência que já tenham sido utilizadas com sucesso por membros da equipe no passado.	
Comprar componentes com bastante antecedência à sua utilização para evitar que possíveis atrasos na entrega sejam danosos ao projeto.	
Mitigar: Pesquisar lojas nacionais que ofereçam os mesmos componentes caso haja atrasos ou problemas na entrega, de forma a minimizar os prejuízos ao cronograma do projeto.	
Impacto reavaliado: 4	Probabilidade reavaliada: 2

Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.

Data: 31/01/2013

Respostas incluídas na WBS/Cronograma

Registros adicionais:
Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Taxação dos componentes comprados no exterior	N° Identificação 05
Descrição do risco: Para compras de componentes em sites do exterior existe o risco de, na alfândega brasileira, ocorrer cobrança de imposto de importação e ICMS, o que pode encarecer o custo desses componentes em até 100%. Além disso, podem haver atrasos na entrega por conta da cobrança das taxas.	

2º Etapa: Avaliação do Risco

Impacto:	<input checked="" type="radio"/> 5 (alta) <input type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input checked="" type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: A compra de componentes no exterior muito provavelmente não ultrapassará o valor limite do orçamento, mesmo com as taxas de importação. Os atrasos por conta de taxação normalmente não são muito expressivos.	
Probabilidade:	<input checked="" type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input type="radio"/> 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: A maioria das encomendas remetidas por empresas internacionais ao Brasil é taxada.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Mitigar: Reservar dinheiro para possível pagamento de impostos, e efetuar o pagamento o mais rapidamente possível, caso seja solicitado, de forma que o produto fique retido o menor tempo possível na alfândega. Dessa forma o impacto no cronograma será minimizado.	

Impacto reavaliado: 2	Probabilidade reavaliada: 5	
Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.	Data: 31/01/2013	<input type="checkbox"/> Respostas incluídas na WBS/Cronograma

Formulário sugerido por Gasnier, 2000 Editora IMAN e alterado por Wille

Registros adicionais:
Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Necessidade de mudança de escopo	Nº Identificação 06
Descrição do risco: Caso ocorram muitos problemas, previstos ou não, e isso impossibilite a conclusão total do projeto, o escopo deverá ser diminuído para o que será possível concluir o projeto com utilizando os recursos (humanos, tempo, dinheiro, etc...) disponíveis.	

2º Etapa: Avaliação do Risco

Impacto:	<input type="radio"/> 5 (alta) <input type="radio"/> 4 (média/alta) <input checked="" type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: A diminuição de escopo é um risco que pode ocorrer caso os recursos disponíveis não sejam suficientes para concluir o projeto em sua totalidade. O escopo deverá ser refeito considerando as limitações.	
Probabilidade:	<input type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input type="radio"/> 3 (média) <input checked="" type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: A diminuição de escopo é um último caso a ser analisado quando os recursos (humanos, tempo, dinheiro, etc...) disponíveis impossibilitem a completude do projeto, ou quando existe recomendação proposta pelos <i>sponsors</i> do projeto.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: Manter um contato constante com os <i>sponsors</i> do projeto e manter um planejamento eficiente para que os recursos permitam que o projeto seja realizado em sua totalidade.	
Mitigar: Caso o escopo seja reajustado, a equipe deverá reportar o fato aos <i>sponsors</i> do projeto. Deve ser informado o novo escopo e as dificuldades que impediram o escopo anterior de ser realizado.	

Impacto reavaliado: 3	Probabilidade reavaliada: 3	
Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.	Data: 31/01/2013	<input type="checkbox"/> Respostas incluídas na WBS/Cronograma

Registros adicionais:
 Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Impossibilidade de uso do robô em certos horários.	Nº Identificação 07
Descrição do risco: O robô é patrimônio da universidade, e, portanto o uso dele é restrito aos dias e horários em que há disponibilidade tanto de equipamento quanto de laboratório para a equipe.	

2º Etapa: Avaliação do Risco

Impacto:	<input type="radio"/> 5 (alta) <input checked="" type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: O impacto de não ser possível manusear diretamente o robô é expressivo, uma vez que há previsão de que muitos testes tenham que ser feitos, e o tempo gasto com o desenvolvimento do <i>hardware</i> provavelmente será considerável. O acesso ao robô é imprescindível para o bom andamento do projeto.	
Probabilidade:	<input type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input type="radio"/> 3 (média) <input checked="" type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: Há modesta probabilidade de não ser possível ter acesso direto ao robô, uma vez que o robô é patrimônio da universidade e a sua utilização deve ser previamente autorizada pelos <i>sponsors</i> .	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade):	
Prevenir: Marcar previamente os horários para ter acesso garantido ao robô. Organizar o cronograma de acordo com os horários de disponibilidade do robô.	
Mitigar: Focar o trabalho da equipe em áreas do projeto que não necessitem acesso direto ao robô (por exemplo, partes específicas do <i>software</i> ou projetos teóricos de circuitos).	
Impacto reavaliado: 4	Probabilidade reavaliada: 1
Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.	Data: 31/01/2013
	<input type="checkbox"/> Respostas incluídas na WBS/Cronograma
	Registros adicionais: Verso ou Anexos

Formulário sugerido por Gasnier, 2000 Editora IMAN e alterado por Wille

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Problemas no planejamento e confecção da placa de circuito impresso.	Nº Identificação 08
Descrição do risco: Problemas na elaboração do modelo do circuito ou dificuldades com os fabricantes da placa.	

2º Etapa: Avaliação do Risco

Impacto:	<input type="radio"/> 5 (alta) <input checked="" type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: dificuldades com a placa de circuito impresso tem impacto direto no desenvolvimento do hardware e realização dos testes, uma vez que há dependência de que a placa esteja disponível e funcionando.	
Probabilidade:	<input type="radio"/> 5 (alto) <input type="radio"/> 4 (médio/alto) <input checked="" type="radio"/> 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: A equipe possui um integrante na equipe que tem considerável experiência com desenvolvimento e confecção de placas de circuito impresso. Portanto, a probabilidade de haverem dificuldades do tipo é media.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão

Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade): Prevenir: Buscar documentação a respeito dos softwares para modelagem de circuitos impressos. Antes de confeccionar a placa de circuito impresso, realizar simulações, quando possível, dos circuitos desenvolvidos.
Mitigar: Buscar auxílio com pessoas mais especializadas na área (por exemplo, professores da universidade). Realizar testes e simulações, se possível, do circuito desenvolvido em busca de erros de projeto e fabricação.
Impacto reavaliado: 4 Probabilidade reavaliada: 1

Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo.

Data: 31/01/2013

Respostas incluídas na WBS/Cronograma

Registros adicionais:
Verso ou Anexos

Planejamento de Riscos

Projeto: Mapeamento de ambientes com o robô Bellator.

1º Etapa: Identificação do Risco

Denominação do risco: Falhas de comunicação entre os membros da equipe.	Nº Identificação 09
Descrição do risco: Por fatores como falta de tempo, problemas pessoais ou falta de comprometimento, as informações importantes a respeito do projeto podem não ser propriamente propagadas entre todos os membros da equipe.	

2º Etapa: Avaliação do Risco

Impacto:	<input checked="" type="radio"/> 5 (alta) <input checked="" type="radio"/> 4 (média/alta) <input type="radio"/> 3 (médio) <input type="radio"/> 2 (médio/baixo) <input type="radio"/> 1 (baixo)
Explique: O impacto caso ocorra um problema desse tipo considerável, uma vez que o andamento do projeto pode ser muito prejudicado. Leva-se em conta também que falhas de comunicação podem acarretar na desmotivação da equipe.	
Probabilidade:	<input checked="" type="radio"/> 5 (alto) <input checked="" type="radio"/> 4 (médio/alto) <input checked="" type="radio"/> 3 (média) <input type="radio"/> 2 (média/baixa) <input type="radio"/> 1 (baixa)
Explique: Tendo em vista a experiência prática de equipes anteriores, há uma probabilidade expressiva de ocorrer falhas de comunicação. Porém, uma vez que há vários canais de comunicação disponíveis e todos os membros possuem acesso à <i>internet</i> , essa probabilidade é reduzida.	

3º Etapa: Desenvolvimento da Resposta ao Risco

Ações, Responsáveis e Datas de Conclusão	
Estratégias e Ações para eliminar ou reduzir este risco (minimizar impacto e/ou probabilidade): Prevenir: Manter vários canais de comunicação (e controle de versões) entre os membros da equipe: <i>e-mail</i> , telefone, <i>github</i> , <i>Dropbox</i> . O gerente irá agendar ao menos uma reunião presencial por semana para troca de informações e motivação da equipe.	
Mitigar: Agendamento de reuniões presenciais com maior frequência com todos os membros da equipe. Uso mais frequente dos meios de comunicação disponíveis. Estímulo maior ao uso dos sistemas de controle de versões.	
Impacto reavaliado: 3	Probabilidade reavaliada: 1

Elaborado por: Luís, Pedro, Ricardo, Stefan e Telmo | Data: 31/01/2013

Respostas incluídas na WBS/Cronograma

Registros adicionais:
Verso ou Anexos

APÊNDICE B – MEDIDAS DO ROBÔ

Tabela 18: Medidas do robô.

Medida	Valor
Comprimento da carcaça	50 cm
Largura da carcaça	40 cm
Distância entre a parte da frente do robô e os eixos das rodas	14 cm
Largura de cada roda	4 cm
Circunferência de cada roda	64 cm
Circunferência do eixo de cada roda	7,5 cm
Circunferência do eixo de cada encoder	22 cm

REFERÊNCIAS

- AGILENTTECHNOLOGIES. **Small Optical Encoder Modules HEDS-9700 Series.** 2002. Disponível em:
<http://www.digchip.com/datasheets/parts/datasheet/021/QEDS-9871-pdf.php>. Acesso em: 28 de Fevereiro de 2013.
- BOT, P. 2013. Disponível em:
<http://www.mobilerobots.com/researchrobots/researchpatrolbot.aspx>.
- CAIRO. 2013. Disponível em: <http://www.caiographics.org/>.
- CORPORATION, D. **DigiKey Corporation.** 2013. Disponível em:
<http://www.digikey.com>. Acesso em: 18 de Fevereiro de 2013.
- ECLIPSE. 2013. Disponível em: <http://eclipse.org/>.
- ESYSTECH, S. **eSysTech Embedded Systems.** 2013. Disponível em:
<http://www.esystech.com.br>. Acesso em: 18 de Fevereiro de 2013.
- EVENSENSE. 2013. Disponível em:
<http://www.invensense.com/mems/gyro/mpu6050.html>.
- GCC. 2013. Disponível em: <http://gcc.gnu.org/>. Acesso em: 18 de Fevereiro de 2013.
- GEDA. 2013. Disponível em: <http://www.gpledapcb.com>.
- GENIUS. 2013. Disponível em:
<http://geniusnet.com/wSite/ct?xItem=16764&ctNode=161>.
- IMPRESSOS, S. C. **Stick Circuitos Impressos.** 2013. Disponível em:
<http://www.stick.ind.br>. Acesso em: 18 de Fevereiro de 2013.
- JAVA. 2013. Disponível em: <http://www.java.com>.
- LPC21ISP. 2013. Disponível em: <http://sourceforge.net/projects/lpc21isp/>. Acesso em: 18 de Fevereiro de 2013.
- MARIN, A. J.; BORGES, J. C. N.; WERGRZN, Y. A. **Desenvolvimento de robô móvel e análise qualitativa de algoritmos de navegação fuzzy.** Curitiba, 2012.
- NETBEANS. 2013. Disponível em: [http://netbeans.org/](http://netbeans.org).
- NXP. **LPC2103 Overview.** 2013. Disponível em:
<http://www.nxp.com/products/microcontrollers/arm7/LPC2103FBD48.html>. Acesso em: 28 de Fevereiro de 2013.
- PCB. 2013. Disponível em: <http://pcb.gpledapcb.com>.

- PROCESSING. 2013. Disponível em: <www.processing.org>.
- RESOLUTION, M. C. digital camera. 2013. Disponível em:
<<http://web.forret.com/tools/megapixel.asp?width=640&height=480>>.
- SATO, B. N. W. K. 2 dimensional infrared robotic mapping system. 2013. Disponível em:
<<http://www-sens.sys.es.osaka-u.ac.jp/research/thesis/10/tech-rep/brad/thesis.pdf>>.
- SHARPCORPORATION. **GP2Y0A02F98YK Datasheet**. 2006. Disponível em:
<http://www.mindsensors.com/index.php?module=documents&JAS_DocumentManager_op=downloadFile&JAS_File_id=335>. Acesso em: 1 de Agosto de 2011.
- TOOLS, L. U. driver . 2013. Disponível em: <<http://www.ideasonboard.org/uvc>>.
- VIDEOLAN. 2013. Disponível em: <<http://www.videolan.org/vlc/>>.