UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Desenvolvimento de plataforma robótica omnidirecional

Autor: Emílio Dolgener Cantú

Orientador: Eduardo Perondi

Porto Alegre, 5 de dezembro de 2017

Sumário

Agradecimentos

Agradeço a todo mundo que ajudou.

Resumo

O presente trabalho apresenta a implementação de uma base robótica omnidirecional holonômica, utilizando 3 omniwheels. Foram implementados um sistema de controle e um método de odometria em malha aberta. Os resultados foram xxxx.

Abstract

This work shows the implementation of a holonomic omnidirectional robotic platform, using 3 omniwheels. Were also implemented a control system and a method for open loop odometry. As a result, xxxx.

Lista de Figuras

Lista de Tabelas

1 Introdução

O interesse na área da robótica se dá pela multi-disciplinaridade do tema, que abrange um espectro de conhecimentos que vai desde mecânica estrutural até a aplicação de teorias de controle sofisticadas. A área ainda se extende por eletrônica, elétrica, computação e até mesmo psicologia. Assim, considera-se que esta área é bastante adequada a um trabalho de conclusão de um curso igualmente abrangente, que é a Engenharia de Controle e Automação.

Em aplicações industriais, a maioria dos robôs utilizados são manipuladores, que realizam tarefas repetitivas – como soldagem ou montagem de peças – com precisão e rapidez adequados a cada aplicação. Estes robôs, no entanto, são em geral fixos, e daí surge o estudo da robótica móvel: como um robô pode se mover sem supervisão e interagir com o mundo real? (Siegwart et al., 2011) Além do interesse acadêmico, existe um significativo interesse comercial, visto que o mercado de robôs móveis, que estava em torno de 4,5 bilhões de dólares americanos em 2014, tende a duplicar até 2020 (Markets and Markets, 2015).

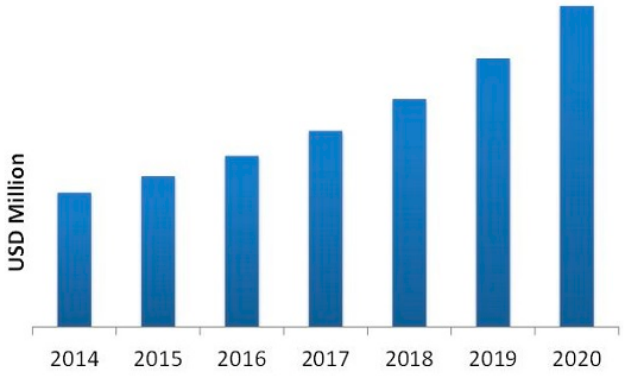

Figura 1: Projeção de crescimento do mercado de robótica móvel até 2020.

Fonte: Markets and Markets (2015)

Segundo Lynch and Park (2017), robôs móveis são divididos em robôs não-holonômicos e robôs omnidirecionais, com diferenças significativas em planejamento de trajetória, controle e modelagem dos dois tipos de robô. Diante da natureza do presente trabalho de conclusão de curso, o escopo foi definido no âmbito dos robôs holonômicos omnidirecionais, que se destacam comercialmente pela habilidade de realizar transporte de cargas pequenas em espaços confinados – como corredores de hospital e depósitos de armazenamento que buscam o aumento da capacidade sem perder agilidade logística nem aumentar o espaço necessário nas instalações. Academicamente, o controle de rodas omnidirecionais apresenta diversos problemas interessantes, vários dos quais serão descritos ao longo do presente trabalho. O desenvolvimento de uma plataforma robótica omnidirecional holonômica se torna útil para futuras aplicações em diversas áreas de investigação em robótica, controle e automação.

O presente trabalho consiste no desenvolvimento (tanto teórico quanto experimental) de uma plataforma robótica que possa se movimentar de maneira autônoma em qualquer direção do plano sem necessidade de reorientação – apresentando holonomicidade. Após uma avaliação na bibliografia sobre os tipos de robôs factíveis de serem construídos no tempo previsto com os recursos financeiros disponíveis, optou-se pela seguinte configuração. A plataforma considerada mais adequada utiliza 3 *omniwheels*, cada uma acionada por um motor elétrico dedicado, conforme o modelo mostrado na Figura 2. Como as rodas são montadas de maneira fixa no chassi, este tipo de robô ainda oferece a vantagem de ser construído com uma estrutura mecânica mais simples, como mencionado por Siciliano and Khatib (2016). As rodas omnidirecionais utilizadas podem ser vistas em detalhe na Figura 3. O sistema de movimentação é controlado por software processado em um computador embarcado a partir dos sinais fornecidos por sensores inerciais e de odometria.

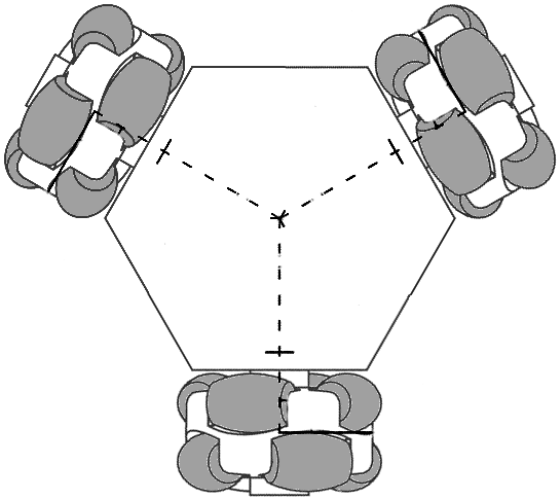

Figura 2: Diagrama de um robô móvel com três rodas omnidirecionais.

Fonte: Adaptado de Ritter (2016)

Este trabalho tem como **objetivo geral** projetar, construir, colocar em operação e testar uma plataforma robótica omnidirecional de baixo custo, mas com características semelhantes às dos sistemas comerciais. Para atingir esse objetivo, se devem alcançar os seguintes **objetivos específicos**:

- Modelagem do robô;
- Especificação e construção de um protótipo;
- Implantação de um algoritmo de controle;
- Implantação de instrumentação e algoritmo de localização;
- Realizar experimentos de seguimento de trajetórias e analisar os resultados obtidos.

O trabalho está organizado da seguinte maneira:

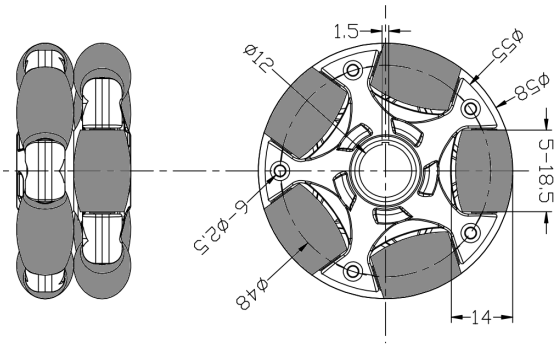

Figura 3: Desenho de uma roda omnidirecional.

Fonte: Adaptado de Nexus (2016)

- Na Seção 2, é apresentada a revisão bibliográfica, versando sobre robótica móvel, técnicas de controle utilizadas na área e um breve resumo sobre métodos de localização, além de uma exposição dos trabalhos mais recentes envolvendo robôs omnidirecionais;
- Na Seção 3, especificação do hardware, estrutura mecânica e montagem do protótipo;
- Na Seção 4, é apresentado o desenvolvimento teórico da modelagem do robô e dos algoritmos de controle e localização;
- Na Seção 5 são implementados os algoritmos descritos na seção anterior;
- Na Seção 6, é desenvolvido e realizado um experimento para avaliar o desempenho do robô projetado;
- Por fim, na Seção 7, serão apresentados as análises e discussões sobre os resultados dos experimentos, a conclusão sobre o trabalho como um todo e algumas propostas para futuros trabalhos.

Além do descrito, o trabalho ainda contém os seguintes apêndices:

- Apêndice A, da descrição dos custos do projeto;
- Apêndice B, com o código fonte da função executada periodicamente para o controle;
- Apêndice C, que mostra as dimensões da plataforma projetada.

2 Revisão Bibliográfica

2.1 Fundamentação Teórica

Para compreender o comportamento de um robô e desenvolver aplicações, é necessário obter o modelo matemático do robô. Para robôs móveis, em contraste com braços robóticos, por exemplo, há um nível de complexidade adicional na estimação da posição do robô, visto que a plataforma móvel não possui nenhuma extremidade fixa em um ponto conhecido. No caso do robô omnidirecional com 3 rodas, se deseja obter um **modelo cinemático** que relacione as velocidades de cada roda à posição do robô no ambiente. Nesse caso, não é ncessário analisar as restrições de movimento adicionadas ao sistema por cada roda, simplificando a obtenção da cinemática direta do TOMR. Tal análise é apresentada por diversos autores, como Siegwart et al. (2011), por exemplo, e será abordada nas próximas seções. Ainda de acordo com estes autores, conforme a velocidade de operação aumenta, se torna importante realizar a análise dos efeitos dinâmicos do sistema. A **modelagem dinâmica** do robô em questão ainda é foco de pesquisa, variando conforme a aplicação desejada, sendo uma das opções apresentada por Kim and Kim (2014). Nesse trabalho, se assumiu o centro de massa no centro geométrico do robô e se contabilizaram diversos parâmetros, como a inércia rotacional do robô e de cada roda, o coeficiente de atrito viscoso das engrenagens redutoras dos motores, os parâmetros elétricos dos motores DC e as características das baterias utilizadas.

Em Lynch and Park (2017) são apresentados diversos métodos de **planejamento de trajetória** para robôs, ou seja, a maneira como o robô (ou um efetuador) se movimenta de um determinado ponto até outro. Há diversas técnicas de se implantação de trajetórias, com diversos níveis de complexidade computacional. Implementações mais simples podem ser uma linha reta de um ponto a outro, com atuação relativamente brusca, causando picos de aceleração, ou mais refinadas, realizado interpolações polinomiais para garantir velocidades e acelerações nulas nos pontos de origem e destino. Também é importante analisar o perfil de velocidade executado pelo robô para garantir uma operação eficiente para as características dinâmicas de cada aplicação.

Segundo Lynch and Park (2017), aplicar controle com realimentação em robôs omnidirecionais é relativamente simples, visto que esse tipo de plataforma robótica apresenta controlabilidade – ou seja, sempre há um conjunto de velocidades ϕ para as rodas que ocasiona uma certa velocidade v (translacional e rotacional) para o robô. Siegwart et al. (2011) sugerem a utilização de um controlador com realimentação de estados. Nesta estratégia, o algoritmo de geração de trajetória divide o caminho a ser percorrido em diversos pontos, enquanto que o controlador implementado garante que o robô percorra tal trajeto, como se os dois sistemas – planejamento de trajetória e controle –, estivessem operando em camadas hierárquicas.

A maioria dos trabalhos se concentra em controladores que levam em consideração apenas a modelagem cinemática do sistema. No entanto, nos casos em que o modelo do robô exija considerações dinâmicas e não-lineares, controladores mais complexos são utilizados. Dentre outros, Siciliano and Khatib (2016) descrevem o método de controle por torque computado, bastante

popular, no qual o modelo dinâmico inverso é utilizado para linearizar a malha de controle. Ainda, conforme Indiveri (2009), o controle por torque computador configura um sistema de controle centralizado, enquanto a utilização de um controlador PID para cada roda seria um exemplo de aplicação de uma estratégia de controle descentralizado.

A localização do robô no ambiente é essencial para o bom funcionamento de um robô móvel. Conforme Lynch and Park (2017). Uma estratégia para se realizar a **odometria** – a medição da distância percorrida – é atrávés da integração das velocidades das rodas. Como o sensoriamento das rodas é muito comum, através de *encoders* de quadratura, por exemplo, tornando-se a odometria barata e conveniente. No entanto, devido às sucessivas integrações realizadas, erros de estimação tendem a se acumular ao longo do tempo de operação, devido a deslizamentos das rodas e erros numéricos, principalmente. Sensores como acelerômetros e giroscópios também tendem a acumular o mesmo tipo de erro. Assim, conforme Siegwart et al. (2011), se recomenda utilizar métodos de localização absolutos de tempo em tempo, como magnetômetros, GPS e marcadores fixos no ambiente, ou, conforme complementado por Lynch and Park (2017), se pode realizar uma conjugação das leituras de diversos sensores, por meio do método conhecido como **fusão de dados**.

Poucos trabalhos detalham a implantação dos sistemas desenvolvidos. De acordo com Craig (2017), tal fato se deve às constantes mudanças e atualizações tecnológicas em *hardware* e linguagens de programação. Os autores tratam, então, de apresentar os fundamentos básicos da implementação, cabendo a cada pesquisador definir suas soluções a partir do que está disponível. Para Siciliano and Khatib (2016), a escolha da arquitetura utilizada é muito subjetiva, sendo comum tratar o robô como um conjunto de subsistemas. Tal abordagem busca melhorar a modularidade do sistema, facilitar a reutilização de módulos em outros projetos e facilitar testes, validações e manutenção. Arquiteturas mais voltadas à robótica tendem a incluir capacidades de processamento em tempo real, controle de atuadores e sensores, e, por vezes, operam em diferentes camadas de prioridade, com escalas de tempo diferentes para diferentes tarefas.

Craig (2017) ainda apresenta diversos paradigmas para as linguagens de programação utilizadas em robótica, sendo que uma das grandes tendências hoje é a utilização de linguagens gerais já existentes, com a adição de bibliotecas voltadas para o desenvolvimento de aplicações robóticas (ou o desenvolvimento das próprias bibliotecas). A escolha da linguagem a ser utilizada, conforme Siciliano and Khatib (2016), deve ser realizada com cuidado, com o objetivo de manter o desenvolvimento do sistema mais fácil, seguro e flexível. Assim, essa escolha pode ser baseada nas experiências anteriores do desenvolvedor, no tipo de robô e nas tarefas que se deseja executar, sem haver um consenso da comunidade acadêmica sobre qual a melhor abordagem.

2.2 Estado da Arte

A grande maioria dos robôs construídos com ***omniwheels*** utiliza 3 rodas em uma configuração triangular simétrica – como apresentado na Figura 2 –, a exemplo de Ritter (2016), Samani et al. (2007), Williams et al. (2002) e Indiveri (2009), entre outros. Alguns autores, como Krinkin et al.

(2015) e Rojas and Förster (2006) utilizam 4 rodas, sendo que este último desenvolveu algoritmos para que o robô continuasse operando mesmo que um dos motores deixe de funcionar. Em diversos trabalhos existe uma preocupação em relação a possíveis derrapagens das rodas ao aumentar a velocidade de operação. Williams et al. (2002) apresentam um estudo sobre os coeficientes de atrito de rodas omnidirecionais em diversas superfícies e um modelo dinâmico que leva tal efeito em consideração. Samani et al. (2007) utilizam 3 rodas motrizes e outras 3 rodas livres ligadas a *encoders*, para que deslizamentos devido a torque excessivo não afetem a odometria.

Jung and Kim (2001) construíram um robô omnidirecional e um sistema de transmissão mecânica que permite holonomicidade utilizando rodas convencionais. Sugerem ainda modos de operação para o robô caso algum dos motores parem de funcionar, passando a se comportar de maneira similar a um robô diferencial. Nestes casos de falha, no entanto, se perde a capacidade omnidirecional.

Há muitas técnicas de **controle** desenvolvidas para estes sistemas. Assim, é possível que cada autor utilize a que mais convenha às suas necessidades específicas. Ritter (2016) utiliza um controlador PID para cada roda, com parâmetros escolhidos empiricamente. Por sua vez, o robô torna-se difícil de controlar com velocidades acima de 1 m/s, segundo resultados de simulações. Samani et al. (2007) utilizam 3 controladores PID, para controlar posição e orientação do robô, e relatam em detalhes o desenvolvimento de tais controladores. Em contraste, Rojas and Förster (2006) e Indiveri (2009) também utilizam PID, porém para o controle de individual de cada motor, utilizando apenas o modelo cinemático do sistema. Indiveri (2009) também sugere estratégias para evitar saturação dos atuadores.

Tanto Treesatayapun (2011) quanto Oubbati et al. (2005) utilizam redes neurais para ajustar parâmetros dos controladores, sendo que no primeiro se tem uma estrutura de controle baseada em redes neurais enquanto que no segundo se utilizam as redes para calcular os parâmetros de 5 controladores PID, melhorando o desempenho, mesmo levando em consideração de não-linearidades nos modelos dinâmicos utilizados. Oubbati et al. (2005) ainda mencionam que os resultados obtidos não foram os melhores possíveis, devido à dificuldade de se coletar dados de treinamento para as redes.

A maioria das implementações de robôs móveis hoje em dia combinam diversas técnicas de **localização e odometria** para implantar a realimentação necessária aos sistemas de controle. Ginzburg and Nokleby (2013) propõem um sistema de localização para robô omnidirecional baseado em odometria (localização relativa) e triangulação ativa de sensores no ambiente (localização absoluta), com fusão de dados para obter o resultado final. Rojas and Förster (2006) utilizam a leitura dos *encoders* das rodas e uma câmera externa, enquanto que Garcia-Saura (2015) utilizam apenas um giroscópio e um sensor de distância. Röhrig et al. (2010), por outro lado, utilizam medições de distância utilizando sensores laser (aplicados em um AGV).

Carrasco and da Silva (2016) mostram que é possível executar um algoritmo de determinação de atitude a partir de uma IMU utilizando métodos de determinação da atitude triaxial e posterior aplicação de filtros de Kalman para melhorar as estimativas na presença de ruído. A caracterização

do ruído dos sensores inerciais foi obtida utilizando o método da variância de Allan. O sistema foi implantado em uma plataforma Arduino UNO, com razoável precisão. Park et al. (1996) também analisam fusão de dados utilizando um filtro de Kalman indireto para realizar *dead-reckoning* a partir da leitura de *encoders* e um giroscópio. Métodos de localização também são desenvolvidos em outras áreas, como relata Jimenez et al. (2009), que implementam três métodos de localização baseados em INS para trajetórias de pedestres, concluindo que os resultados podem ser melhorados quando há mais qualidade na detecção da orientação. Steinhoff and Schiele (2010) obtiveram resultados similares na mesma área.

Dos trabalhos mencionados, poucos entram em detalhes quanto ao *hardware* utilizado. Oubbati et al. (2005) utilizam um computador embarcado de 2,6 GHz, uma grande evolução em relação a Feng et al. (1989), que utilizavam um computador Motorola 68000 com aproximadamente um milésimo da capacidade computacional daquele. Takemura et al. (2007) e Loh et al. (2003) utilizam computadores externos, envolvendo atrasos na comunicação entre tais computadores e o robô. Carrasco and da Silva (2016) apresentam um sistema que implementa um filtro de Kalman em um microprocessador Arduino UNO, uma alternativa de baixo custo. Diversos trabalhos mais recentes, como Krinkin et al. (2015), utilizam especificamente um **Arduino** para a interface com sensores e atuadores e o computador embarcado **Raspberry Pi** para o processamento – solução esta também adotada neste trabalho.

Para a avaliação dos resultados da plataforma construída, Loh et al. (2003) implementaram 4 tipos de trajetória: translação retilínea , translação curvilínea – ambas sem alteração na orientação –, rotação pura e um caminho combinado de rotação em torno do seu centro e translação retilínia em relação às referências globais. Duas canetas foram montadas no robô, uma no centro e outra na periferia, para avaliar o resultado das movimentações executadas.

3 Especificação e montagem do protótipo

Conforme mencionado nas seções anteriores, o robô construído possui três rodas em uma configuração simétrica. Apesar da falta de redundância – pois se alguma das rodas falhar se perde a holonomicidade –, robôs omnidirecionais com 3 rodas (TOMR) são utilizados com mais frequência por serem mais simples de se implementar, apresentarem custo mais baixo (pois motores e rodas são responsáveis por 53% do custo do projeto, conforme o Apêndice A), e uma certa economia de peso. As rodas utilizadas – mostradas na Figura 3 –, medem 58 mm de diâmetro, com estrutura em plástico e dez roletes emborrachados, e possuem capacidade de carga nominal de 3 kg (Nexus, 2016), suficiente para os fins de demonstração do projeto. Cada roda é acionada por um motor de corrente contínua com caixa de redução, com uma velocidade nominal no eixo de saída de 210 rpm para uma tensão de 6 V. A máxima potência do motor está especificada para uma corrente de 1,1 A, a 110 rpm. Incluso no motor está um *encoder* de quadratura, que permite a leitura da velocidade da roda e da direção de rotação. Com a relação de redução, se tem que para cada revolução da roda se tem 341,2 pulsos do sensor, e portanto cada pulso representa 0.01841 radianos (Banggood.com, 2017).

Além da utilização dos *encoders* para implementação da odometria, também foi instalada na estrutura uma bússola, para garantir uma medida absoluta da orientação do robô (sem os erros que se acumulam nos métodos de *dead-reckoning*). O modelo utilizado é a HMC5883L, já instalada em uma placa com alguns componentes necessários para seu funcionamento. A precisão do circuito, de acordo com o fabricante, é de 2 graus (Honeywell, 2013). Para complementar a odometria, também foi instalada no robô uma unidade de medidas inerciais MPU6050, que possui acelerômetro e giroscópio em torno dos três eixos utilizados (Invensense, 2013). Os sensores descritos neste parágrafo foram adquiridos e montados à estrutura, porém não foi realizada sua implementação. Os dois periféricos utilizam o protocolo de comunicação I2C (Semiconductors, 2000), também compatível com o computador utilizado.

Foi adquirida também uma bateria NiCd, com capacidade de carga de 2000 mAh e 7,2 V de tensão nominal. Este tipo de bateria se caracteriza por apresentar recarga rápida e boa capacidade de utilização com correntes altas. Ligados à bateria, se tem 2 reguladores de tensão *step down* MP2307, especificados para fornecer corrente constante de até 3 A cada um, suportando picos de até 4 A (MPS, 2008). A tensão de saída dos reguladores foi configurada em 5.1 V (para o computador) e 6 V (para os motores). Como cada motor opera em geral com correntes abaixo de 1 A, o regulador utilizado é adequado, porém apresenta margens de capacidade consideravelmente pequenas. Os *encoders* sãp alimentados pelo próprio computador, que possui saída regulada de 3,3 V capaz de fornecer até 500 mA (Upton and Halfacree, 2014). A bússola e a IMU tem tensão de alimentação de 3,3 V, podendo ser adicionado ao sistema mais um regulador de tensão quando forem eventualmente integradas ao sistema, pois há espaço suficiente no chassi para tal. Neste caso, se recomenda alimentar os *encoders* a partir do mesmo regulador.

O acionamento dos motores se dá por um circuito de pontes H. Há duas destas placas, e cada

uma pode acionar dois motores. Assim, se tem a possibilidade de utilizar mais um motor (ou outro atuador) em trabalhos futuros. Os *drivers* são desenvolvidos em torno do circuito L298N, que pode fornecer 4 A de corrente contínua distribuída entre as cargas (STMicroelectronics, 2000). O chaveamento de cada canal dos *drivers* é feito por meio de modulação de largura de pulso, fornecida pelo computador. Assim como os demais componentes, os *drivers* foram fornecidos integrados a uma placa montada, com terminais para fixação de cabeamento e dissipadores de calor.

Todo o processamento é realizado por um **Raspberry Pi**, um *single board computer*, que utiliza a arquitetura ARM em seu processador, ideal para dispositivos alimentados por baterias por consumir pouca energia e gerar pouco calor. O processador possui quatro núcleos, e um *clock* de 1,2 GHz – poder computacional equivalente há um computador de mesa comum. O RPi utiliza um sistema operacional GNU/Linux, e *software* deve ser desenvolvido para ser executado nesta plataforma. Há ainda 40 pinos de GPIO que podem ser utilizados para conectar sensores, atuadores e diversos componentes, e suporte nativo a I2C (Upton and Halfacree, 2014).

Para unir todos os componentes descritos, se projetou uma estrutura central, como um chassi. Tal estrutura pode ser vista na Figura 4. No centro geométrico da estrutura e na periferia, próximo a uma das rodas, foram feitos dois orifícios que devem acomodar uma caneta hidrográfica cada. Assim, durante a fase de testes, se pode acompanhar graficamente a evolução cinemática do robô. Devido a localização central de uma das canetas, todos os componentes foram instalados na periferia da estrutura. Se tomou ainda o cuidado de instalar os CI de acelerômetro e giroscópio o mais próximo ao centro possível, para que as componentes de aceleração centrípeta dos movimentos com componentes de rotação não influenciassem em demasiado nos resultados. A IMU poderia ter sido colocada no centro geométrico, e este erro poderia ser introduzido no traço da caneta. No entanto, como a odometria e localização dependem muito mais dos sensores montados nos motores do que da IMU, se preferiu manter a caneta no centro, mantendo o MPU6050 o mais próximo possível. A bússola também foi montada relativamente próxima ao centro do robô, se tomando o cuidado de alinhar os eixos dos sistemas de coordenadas dos sensores com os do robô.

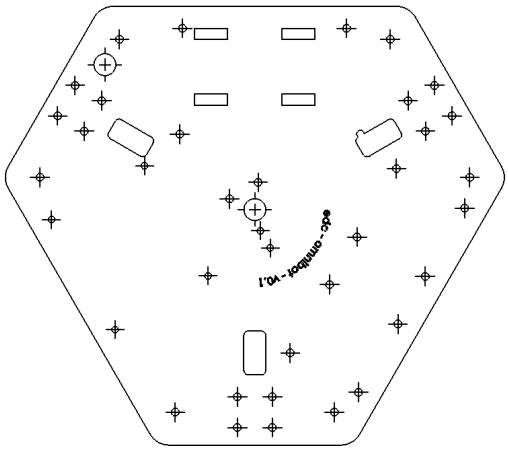

Figura 4: Chassi projetado.

Todos os componentes adquiridos possuem furos para fixação por meio de parafusos com 3 mm de diâmetro. A estrutura foi projetada com furos de 3,5 mm de diâmetro, para compensar possíveis erros de medição (visto que nem todos os componentes apresentaram seus desenhos nas informações técnicas) e possíveis tolerâncias de fabricação. Além dos furos de fixação dos componentes, também foram introduzidos orifícios próximos aos motores, para passagem dos cabos de um lado a outro da placa, e orifícios para fixação da bateria com presilhas plásticas. Na mesma área destinada à fixação da bateria se adicionou furação capaz de receber uma placa Arduino MEGA, caso se deseje utilizar um microcontrolador em trabalhos futuros. Também foram adicionados 6 furos na periferia do chassi, para possibilitar a montagem de outra chapa sobre a dos componentes, caso sejam realizados trabalhos que exijam a expansão da estrutura.

A plataforma projetada foi então fabricada, utilizando chapas de acrílico transparente de 5 mm de espessura. Se cogitou produzir tal estrutura em alumínio, porém se mostrou mais prático utilizar o acrílico. Entre as vantagens do material plástico se podem destacar a fácil obtenção, baixo custo, isolamento elétrico (permitindo montar os componentes eletrônicos diretamente sobre o chassi) e a possibilidade de fabricação utilizando uma máquina de corte a *laser*, que elimina o limite inferior de tamanho de brocas e fresas em relação a fresadora CNC considerada originalmente. A espessura foi escolhida empiricamente, dentro das disponíveis, de maneira relativamente conservadora, e atendeu as necessidades. Na Figura 5 se pode ver a montagem final do protótipo. Os furos sobredimensionados não afetaram o alinhamento das peças.

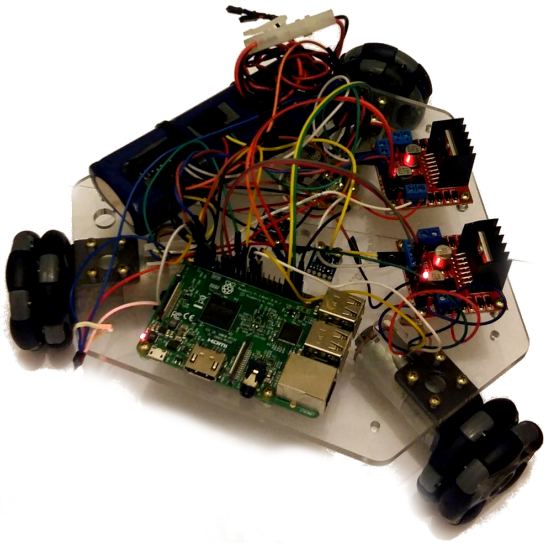

Figura 5: Protótipo montado, sem as canetas.

A bateria foi fixada sobre a estrutura utilizando presilhas plásticas. Ao redor da bateria foram fixados 3 barramentos, para aterramento, alimentação dos *drivers* e alimentação dos *encoders*. Foram instalados um conector para a bateria e outro conector para o caso em que se deseja utilizar uma fonte externa. Além dos fios que alimentam os reguladores de tensão, um par de fios

sobressalente (conectados ao terminal positivo e negativo da fonte ou bateira) foi instalado junto a estes conectores, e pode ser utilizado em trabalhos futuros.

O custo de aquisição dos componentes relatados pode ser visto detalhado no Apêndice A. Cabe ressaltar que todos os itens foram comprados em dobro, para realizar a montagem de dois robôs para futuros trabalhos no LAMECC (Laboratório de Mecatrônia e Controle). Mais detalhes sobre as dimensões do chassi podem ser vistos no diagrama apresentado no Apêndice C.

4 Desenvolvimento Teórico

4.1 Modelagem Cinemática

Primeiramente, se definem dois sistemas de coordenadas. O primeiro, (x_I, y_I), é o sistema de coordenadas global, fixo no ambiente. O segundo, (x_R, y_R), está centrado no próprio robô. Ainda se pode definir o ângulo θ como a orientação do robô – ou seja, o ângulo entre os dois sistemas de coordenadas. Tal relação pode ser vista na Figura 6, e a transformação de um sistema para o outro é descrita na Equação 1, conforme Siegwart et al. (2011) e Ritter (2016).

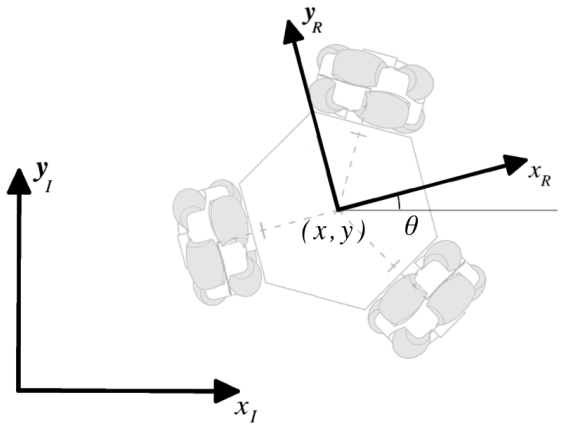

Figura 6: Sistemas de coordenadas global I e relativo ao centro do robô R.

Fonte: Adaptado de Ritter (2016)

$$\begin{pmatrix} x_I \\ y_I \\ \theta \end{pmatrix} = \begin{pmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_R \\ y_R \\ \theta \end{pmatrix} \tag{1}$$

O último termo da Equação 1 também pode ser descrito como q_R, e o vetor de velocidades $[v_x, v_y, \omega_z]^T$, centrados no sistema de coordenadas do robô, é $\dot{q_R}$. Com o objetivo de mapear a velocidade de giro das rodas $\dot{\phi} = [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3]^T$ às velocidades $\dot{q_R}$, se utiliza a modelagem cinemática apresentada por Siegwart et al. (2011), com as referências apresentadas na Figura 7. Na figura, A mesma modelagem é utilizada por Ritter (2016), porém com outra sequência e sentido de giro para as rodas.

Assim, para um robô com 3 rodas dispostas em simetria radial em torno do centro da estrutura, a cinemática direta é dada pela Equação 2. Diversos autores utilizam variações da mesma modelagem (Rojas and Förster (2006), Pin and Killough (1994), entre outros). Nas equações apresentadas, r é o raio de cada roda e R o raio do robô (a distância do centro da roda ao centro da estrutura do robô).

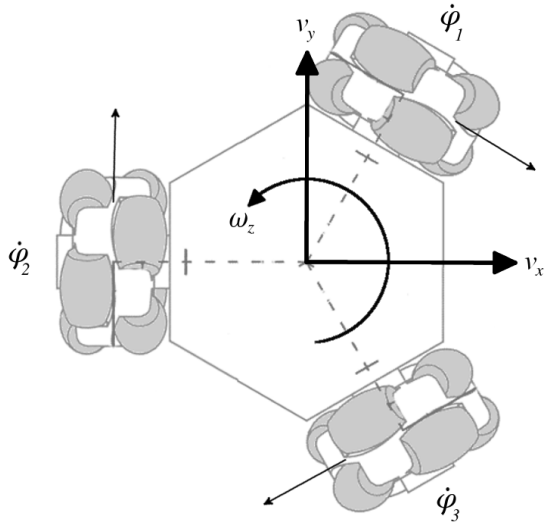

Figura 7: Vista superior do robô, mostrando as convenções adotadas. As grandezas v_x e v_y estão no sistema de coordenadas do robô.

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = \frac{r}{3R} \begin{pmatrix} -\frac{3R}{\sqrt{3}} & 0 & \frac{3R}{\sqrt{3}} \\ R & -2R & R \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{pmatrix}. \tag{2}$$

Também se deseja utilizar a cinemática inversa do modelo, obtida realizando-se a inversão da matriz de transformação apresentada na Equação 2, é dada pela Equação 3. Nota-se que esta inversão é simplificada no caso do robô com 3 rodas, visto que quando há mais rodas é formada uma matriz $3 \times n$, sendo n o número de rodas, e se deve utilizar uma matriz pseudo-inversa, conforme demonstrado por Rojas and Förster (2006).

$$\begin{pmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{pmatrix} = \frac{1}{r} \begin{pmatrix} -\frac{\sqrt{3}}{2} & \frac{1}{2} & R \\ 0 & -1 & R \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & R \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} \tag{3}$$

Como pela classificação de Campion et al. (1996) um TOMR é caracterizado na categoria (3,0), o modelo cinemático das equações 2 e 3 é controlável, estável e descreve a posição, orientação e suas derivadas de forma suficiente.

4.2 Odometria

Durante a operação do robô, se torna necessário calcular a posição da estrutura. Para o cálculo da odometria, se utiliza a metodologia mostrada em Lynch and Park (2017). Se assume que durante

um certo intervalo de tempo Δt se tenha velocidades de rotação constantes nas rodas, o que permite considerar $\dot{\phi}_i.\Delta t = \Delta\phi_i$. Considera-se também que a unidade de tempo deste período é arbitrária, e como se deseja integrar no mesmo intervalo posteriormente, se assume um período unitário $\Delta t = 1$. Este procedimento está descrito na Equação 4, modificada a partir da Equação 2. Na prática, é fácil contar os deslocamentos angulares $\Delta\phi_i$, visto que o número de pulsos por revolução dos *encoders* é determinado.

$$\begin{pmatrix} v_x \\ v_y \\ \omega_z \end{pmatrix} = \frac{r}{3R} \begin{pmatrix} -\frac{3R}{\sqrt{3}} & 0 & \frac{3R}{\sqrt{3}} \\ R & -2R & R \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \Delta\phi_1 \\ \Delta\phi_2 \\ \Delta\phi_3 \end{pmatrix}. \tag{4}$$

De posse das velocidades da plataforma durante o período de tempo unitário Δt – lembrando que v_x, v_y e ω_z estão no sistema de coordenadas centrado no corpo do robô –, se deve avaliar o deslocamento em relação ao centro do robô na posição anterior. Para o caso em que $\omega_z = 0$, numa trajetória retilínea, se tem simplesmente que $\Delta q_R = \dot{q}_R$.

No entanto, quando houve mudança de orientação no período e consequentemente $\omega_z \neq 0$, se deve levar em consideração os desvios de trajetória causados por essa rotação. Assim, se obtem Δq_R de acordo com a Equação 5 (Lynch and Park, 2017).

$$\Delta q_R = \begin{pmatrix} \Delta x_R \\ \Delta y_R \\ \Delta\theta \end{pmatrix} = \begin{pmatrix} (v_x sen(\omega_z)) + v_y(cos(\omega_z) - 1)/\omega_z \\ (v_y sen(\omega_z)) + v_x(1 - cos(\omega_z))/\omega_z \\ \omega_z \end{pmatrix} \tag{5}$$

Sendo k o instante antes do período de tempo analisado, para se obter a nova posição q_I do robô no sistema de coordenadas global se deve utilizar a rotação $R(\theta_k)$ apresentada na Equação 1, e atualizando os valores da última iteração conforme a Equação 6.

$$q_{I(k+1)} = q_{I(k)} + \Delta q_I = q_{I(k)} + R(\theta_k)\Delta q_I \tag{6}$$

4.3 Planejamento de Trajetória

Para o robô desenvolvido, não há a necessidade de implementar algoritmos complexos de planejamento de trajetória (detecção de obstáculos, caminhos de mínima energia, etc.). Serão abordados caminhos "ponto a ponto", que levam de um ponto inicial a um ponto final, ambos em repouso (Lynch and Park, 2017).

Apesar de ser uma trajetória simples, ainda se podem aplicar considerações para uma melhor operação do sistema. Uma dessas considerações é o chamado *time-scaling* da trajetóra, ou seja, a geração de uma função $s(t)$ que suavize o comportamento do robô por meio de restrições em velocidades e acelerações. Na Figura 8 se pode ver uma curva de perfil de velocidade polinomial de quinta ordem, que pode garantir velocidades e acelerações nulas nos pontos de origem e destino.

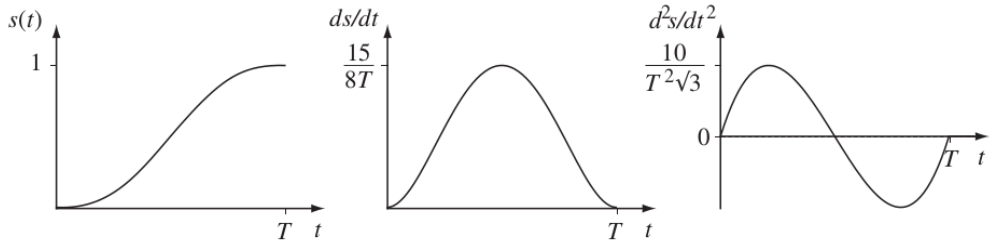

Figura 8: Deslocamento, velocidade e aceleração durante uma trajetória gerada por polinômio de quinta ordem. Aceleração e velocidade são nulas tanto no ponto de origem quanto no ponto de destino.

Fonte: Lynch and Park (2017)

No entanto, a interpolação de um polinômio a cada cálculo de trajetória é um processo que pode envolver um certo custo computacional elevado, e devido à simplicidade dos componentes utilizados, se julgou que o aumento de suavidade na operação não fosse significativo. Portanto, neste trabalho se optou por utilizar um perfil de velocidade trapezoidal, conforme mostrado na Figura 9. Tal perfil é um dos mais comuns em robótica, devido a sua simples implementação. Os limites de aceleração foram definidos na fase de implantação do *software*, de modo a evitar o deslizamento das rodas utilizadas na superfície de testes.

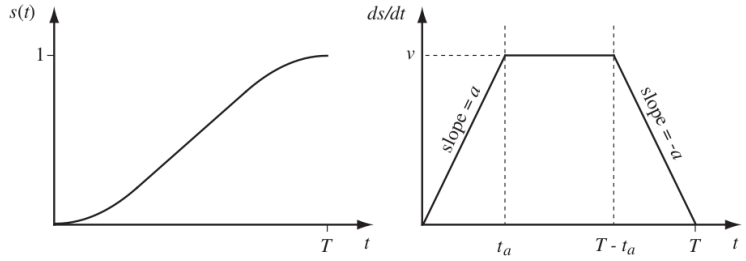

Figura 9: Deslocamento e velocidade durante um deslocamento com perfil de velocidades trapezoidal. Tal perfil foi adotado neste trabalho.

Fonte: Lynch and Park (2017)

Utilizando a curva do perfil de velocidade, se pode fixar setpoints de velocidade para cada ponto.

4.4 Controle

SUBSECTION EM CONSTRUÇÃO

Diversas abordagens podem ser utilizadas para a implantação de controladores em robôs

omnidirecionais. Se pode realizar o controle das coordenadas desejadas, em x, y e z, ou o controle da posição de cada roda, independentemente. No caso, foram realizados três controladores de posição, em X, Y e ω, utilizando a odometria discutida acima. O controlador então decide o set point de velocidade para cada roda, e envia os sinais de comando para um conjunto secundário de controladores de velocidade, 1 para cada roda. Se pode ver um esquema do sistema utilzado na FIgura 10.

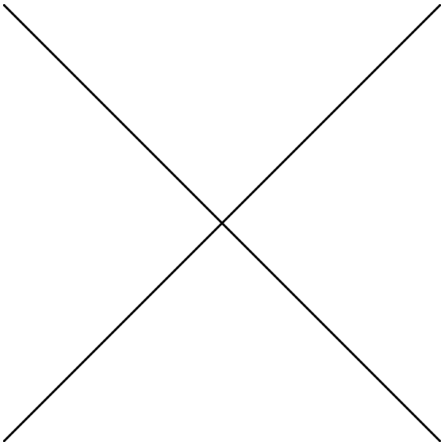

Figura 10: Arquitetura de controle utilizada.

Estabilidade, tipo do controle, proporcional, integral, derivativo, digital, tempo de ciclo, q q eu digo aqui, pô?

Não linearidade do tipo zona morta, como será relatado a seguir. COrreção como na Figura 11.

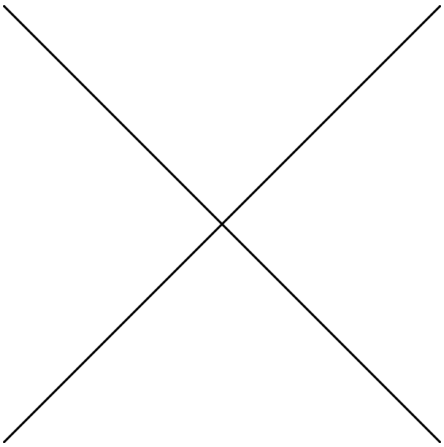

Figura 11: Um tipo de correção para não linearidades do tipo zona-morta.

Lynch and Park (2017):

In most robotic applications, higher control update rates are of limited benefit, given time constants associated with the dynamics of the robot and environment.

We can use feedforward control, which commands motion even when there is no error, in combination with feedback control to limit the accumulation of error:

EQUAÇÃO (11.1) -> EQUAÇÃO (13.11), consigo adaptar pro sinal de controle do motor DC?

This feedforward-feedback controller is the preferred velocity control law.

Loop de velocidade dentro do loop de posição. Posição das rodas ou posição em x e y e ω_z? Practical Bounds on Feedback Gains According to our simple model, we could increase K p and K d without bound to make the real components of the roots more and more negative, achieving arbitrarily fast response. In practice, however, large gains lead to actuator saturation, rapid torque changes (chattering), vibrations of the structure due to unmodeled flexibility in the joints and links, and possibly even instability due to the finite servo rate frequency. Thus there are practical limits on the set of useful gains.

Rojas and Förster (2006): Sugerem que utilizar um controlador para cada roda é melhor do que para cada grau de liberdade. No nosso caso, é tranquilo pois temos apenas 3 rodas, mantendo o mesmo número de controladores. Devido à realimentação externa lenta, utilizam um preditor no robô. Não entram em detalhes. motores: https://www.banggood.com/6V-210RPM-Encoder-Motor-DC-Gear-Motor-with-Mounting-Bracket-and-Wheel-p-1044064.html?p=970719369296201312SG

Given a desired trajectory q d (t), we can adopt the feedforward plus proportional feedback linear controller (11.1) of Chapter 11 to track the trajectory: Lynch and Park (2017)

Samani et al. (2007): Definir os coeficientes dos PIDs é uma novela, pois devemos levar em consideração parâmetros que possuem muita variação, como o coeficiente de atrito do solo, características das baterias, entre outros. Controle deles é bem legal.

4.5 Limitações de Velocidade

É importante ressaltar que toda a cinemática desenvolvida nas subseções anteriores não considera limites de velocidade para os atuadores. Numa aplicação real, entretanto, existe um ponto de saturação no acionamento de cada motor, que deve ser levada em consideração. Na Figura 12 se pode ver o efeito dessas limitações, conforme descrito em Lynch and Park (2017).

Quando não há rotações ($\omega_z = 0$), o limite de velocidade do corpo do robô é descrito pelo hexágono mostrado na porção esquerda da Figura 12: a maior velocidade possível é na direção em que uma das rodas está sendo "arrastada", e as componentes de velocidade das outras rodas se somam. Numa situação em que haja necessidade de rotação, a velocidade angular do robô se torna limitada da maneira mostrada no volume tridimensional à direita da Figura 12, e se torna fácil enxergar que, para realizar um movimento de rotação na maior velocidade ângular possível, não se pode ter movimentos de translação, para que todos os componentes de velocidade das rodas contribuam apenas para a rotação.

Se pode dizer, então, que para aplicações reais nas quais a holonomicidade da plataforma é de

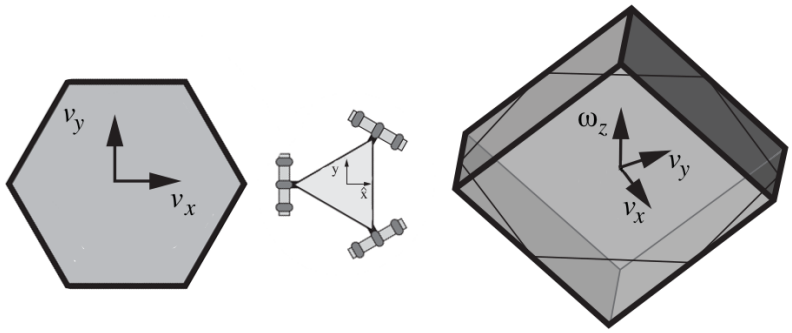

Figura 12: Limites de velocidade translacional e rotacional em função dos limites de saturação dos motores reais.

Fonte: Adaptado de Lynch and Park (2017) para as coordenadas utilizadas.

fato desejável, se deve implantar um sistema de planejamento de trajetória que leve em consideração as limitações de velocidade descritas acima.

5 Implementação dos algoritmos

Os aspectos teóricos desenvolvidos na seção anterior foram implementados em software para a aplicação prática do sistema. A implementação é, em geral, difícil de se encontrar nos trabalhos da bibliografia, e mesmo que todos os autores descrevessem seus algoritmos em detalhes, os rápidos avanços na área tornam obsoletos os recursos utilizados. tá meio ruim isso aqui.

O sistema proposto foi desenvolvido e testado em módulos, conforme as divisões da seção anterior, e se pode enxergar a hierarquia de cada bloco na Figura 13. Seguindo a figura, os subsistemas foram implementados de baixo para cima. *<- mais ou menos, pq eu comecei com a cinemática.*

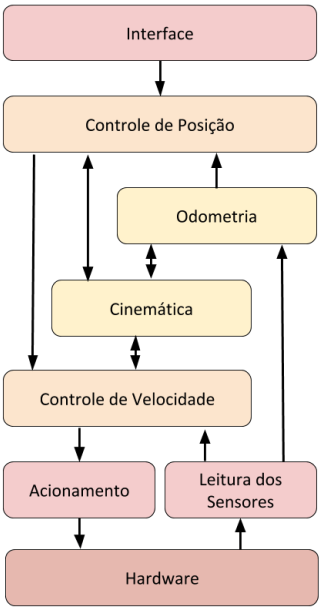

Figura 13: Estrutura e hierarquia dos subsistemas.

Foram omitidos da figura, por simplicidade, os algoritmos de limitação de velocidade, de compensação de zona-morta e de FALTA UM.

O *software* foi implementado em um computador embarcado Raspberry Pi, conforme proposto por Ritter (2016). O **Raspberry Pi** é um *single board computer*, que utiliza a arquitetura ARM em seu processador, ideal para dispositivos alimentados por baterias por consumir pouca energia e gerar pouco calor. O processador possui quatro núcleos, e um *clock* de 1,2 GHz – poder computacional equivalente há um computador de mesa comum. O RPi utiliza um sistema operacional GNU/Linux, e *software* deve ser desenvolvido para ser executado nesta plataforma. Há ainda 40 pinos de GPIO que podem ser utilizados para conectar sensores, atuadores e diversos componentes, e suporte nativo

a I2C (Upton and Halfacree, 2014).

Uma preocupação que se tem ao utilizar um computador se deve ao fato do mesmo não ser caracterizado como um sistema de tempo real. No caso, temporização precisa não é garantida, e a ordem de prioridade de execução de tarefas é gerenciada pelo *kernel VERIFICA*. Assim, funções que seriam executadas imediatamente em um microcontrolador (como rotinas de interrupção), são executadas "assim que possível", o que pode prejudicar o desempenho do sistema.

Com isso em mente, foi implementado um software com a linguagem C++, uma das linguagens suportadas pelas bibliotecas de acesso às portas GPIO que apresenta melhores velocidades de execução. *(citar alguém?)* Com esta escolha de linguagem, também se pode utilizar parcialmente os códigos implementados para o trabalho de Ritter (2016), que também utilizou a mesma linguagem de programação.

A primeira etapa do desenvolvimento foi a de separar o código desenvolvido por Ritter (2016) em duas partes: uma destinada aos comandos de acionamento, que naquele trabalho eram realizados em um simulador, e outra relacionada à cinemática do robô. Esta segunda porção foi encapsulada em uma biblioteca própria, que pode assim ser utilizada por qualquer desenvolvedor de software no âmbito de TOMR. A biblioteca foi então devidamente analisada e testada no computador embarcado. A biblioteca se chama kinematics.h.

Em seguida, foram realizados testes de acionamento e leitura dos sensores dos motores, se baseando na biblioteca PIGPIO *como referenciar a bibioteca?* (Joan, 2017) para a utilização das entradas e saídas físicas do computador. Utilizando o conceito de orientação a objetos, foi criada uma classe que descreve os parâmetros de cada conjunto motor/sensor e as operações a serem realizadas sobre eles. Esta classe foi batizada de RPiInterface, pois realiza a interface entre o processamento da Raspberry Pi com os sensores e atuadores físicos.

A leitura dos sensores de velocidade, no entanto, foi realizada no main.cpp, devido à utilização de uma biblioteca específica de decodificação de encoders utilizada (rotary_encoder.h), que implementa interrupções e *n sei direito pq não rolou deixar dentro da classe.*

As equações implementadas relacionam as matrizes de transformação com a velocidade linear da periferia das rodas, $\dot{\phi}_i r$, em m/s. A leitura dos encoders é realizada em pulsos/μs. O fator de conversão aplicado é (em teoria) 534.18. Ao se utilizar o comando inverseKinematicsWorld, se obtêm as velocidades em m/s e a velocidade angular em rad/s.

Na mesma classe se implementou a função de atualização da lei de controle de velocidade de cada motor. Esta função é executada a cada 10ms pela função loop() (mostrada no Apêndice B)

Se implementou um controle proporcional simples, COLOCAR A LEI AQUI, mas se ontou que em trajetórias que necessitam do movimento das trê rodas em velocidades muito distintas se tem problemas. Para se mover em uma reta no eixo x, por exemplo, apenas as rodas 1 e 3 (verificar numeros) precisam girar, e na mesma velocidade, e fica simples de ver que o desempenho do controlador para as duas rodas vai ser similar. Mais sobre o tema é descrito na próxima seção. No entanto, na movimentação no eixo y são necessárias q a roda 1 esteja girando com o triplo da

velocidade das outras duas (verificar), e assim vence a zona morta muito antes. Para solucionar isso, se utilizou uma técnica de eliminação de zona morta, confrome sdkasldkjabsdlkjbladkjbsa.

Após a verificação do sistema de controle de velocidade, foram implementadas da maneira descrita na seção anterior as fórmulas relacionadas à odometria. Os dados de velocidade foram obtidos se acumulando diretamente a contagem de pulsos de em cada encoder, e avaliando a taxa de variação desta contagem e de uma contagem anterior no intervalo de tempo em que a função foi chamada novamente. A precisão da odometria é essencial para o funcionamento do controle de posição, e portanto o sistema foi testado antes do desenvolvimento do outro, com os resultados descritos na seção seguinte.

O controle de posição foi implementado para ser executado dentro da mesma função loop().

Seguir o perfil de velocidade não é muito fácil, visto que dead zone. budibudibuidi Indiveri (2009) trata saturação.

Além de utilizar os comandos fornecidos por Ritter (2016), foram implementados os modos de movimentação citados por Loh et al. (2003): translação retilínea , translação curvilínea – ambas sem alteração na orientação –, rotação pura e um caminho combinado de rotação em torno do seu centro e translação retilínia em relação às referências globais.

Se notou que o acionamento dos motores depende de a O procedimento de opereação pode ser representado pelo diagrama apresentado na Figura 14.

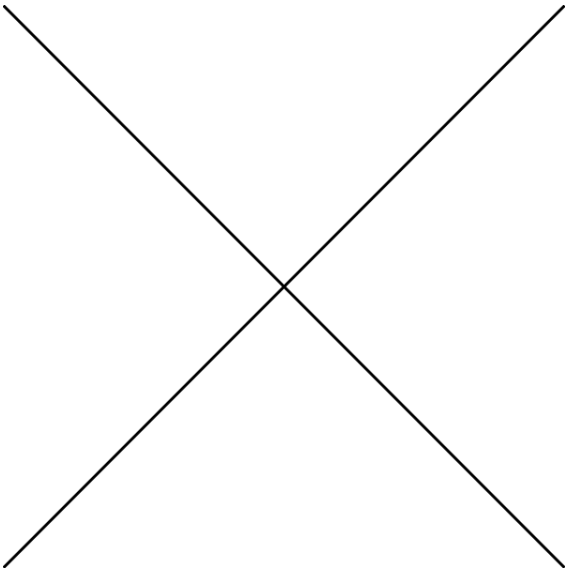

Figura 14: Fluxograma de operação do robô. TROCAR FIGURA

Fazer algum arquivo de log para cada execução?

6 Avaliação experimental

Se deseja avaliar o comportamento dos algoritmos implementados e da modelagem realizada operando o robô em três situações:

- trajetória retilínea, sem nenhum movimento de rotação, em diversas direções e com diversas velocidades de acionamento, conforme mostrado na Figura 15;
- trajetória de rotação pura, analisando a variação de ângulo atingida em função do ângulo desejado e da velocidade desejada. Tal situação é ilustrada na Figura 16;
- trajetória híbrida, realizando tanto uma translação retilínea arbitrária quanto uma rotação sobreposta, como no diagrama da Figura 17.

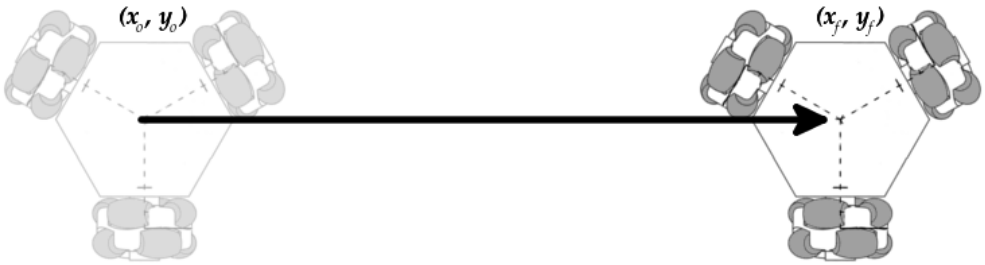

Figura 15: Trajetória retilínea pura.

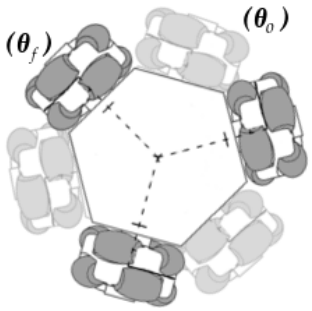

Figura 16: Trajetória de giro.

Se espera com a primeira trajetória, avaliar a robustez dos algoritmos quando se necessita o acionamento combinado de múltiplas rodas. Por exemplo, no eixo X, não há necessidade de movimento em uma das rodas. Em outras direções, se introduz movimentos mais lentos das rodas, o que já se viu que é um problema por causa do atrito viscoso da caixa de redução.

Com a trajetória de giro, em que as três rodas devem operar com a mesma velocidade, se

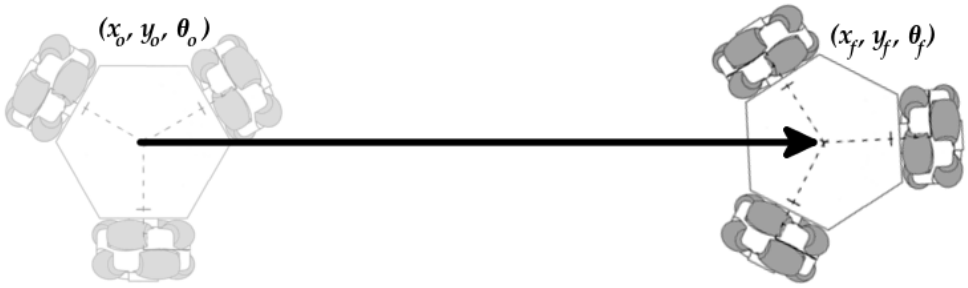

Figura 17: Trajetória combinada.

deseja avaliar alguma coisa relacionada a isso. Na última trajetória, híbrida, se deseja mostrar o funcionamento do algoritmo de limitação de velocidade, e os efeitos deste fator.

Se notou que o acionamento dos motores depende de alguns fatores. Variando a frequência dos PWMs mudou o q?

O protótipo foi acionado sobre um papel enorme, com duas canetas de cores distintas instaladas nos orifícios destinados a tal.

7 Resultados

A implantação do sistema descrito nas seções anteriores é, de certa maneira, hierárquica: para que um subssistema de alto-nível funcione, os que estão abaixo dele devem estar funcionando também. Tal conceito pode ser visto na Figura 13 Portanto, os testes e validações ocorreram de maneira concomitante ao desenvolvimento do software e montagem da estrutura.

Devido ao fato do computador não ser um sistema 100% apropriado para tempo real balbalbal-balbabalalbal a tmeporização dos sensores fic preocupante. ASsim, s emediu as leituras obtidas para cada encoder, e se obteve o seguinte, comforme aFIgura 18.

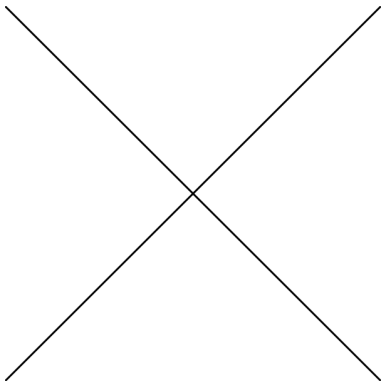

Figura 18: Encoderdoderdoder.

Durante os testes de acionamento dos motores, se percebeu uma não linearidade do tipo "zona-morta", e foi realizado um ensaio para quantificar o problema. Os resultados de tal ensaio podem ser vistos na Figura 19. Foi implementado um controlador proporcional para a velocidade, com ganho bastante baixo, de modo a tornar a resposta do sistema lenta, e foram definidos *setpoints* que propositalmente levassem a saturação dos *drivers* dos atuadores. Na Figura, se pode enxergar a zona morta em torno de t =1.5 s, quando a velocidade da roda se eleva repentinamente, e entre t = 4 s e t = 5 s, quando foi realizado o comando de inversão da velocidade da roda, e para um intervalo de valores de acionamento em torno da origem, não há movimento no eixo. Esta não linearidade é oriunda do atrito estático introduzido pelos componentes mecânicos responsáveis pela relação de redução. *tenho mais um gráfico mostrando q dá pra andar bem devagar se já estiver andando no inicio... temos espaço??*

O ensaio foi realizado simultaneamente com as três rodas, e como todas as leituras apresentaram resultados similares, apenas uma aparece no gráfico.

Uma consequência deste efeito foi verificada no acionamento do robô em trajetórias que exigem baixas velocidades de alguma das rodas. Esta situação pode ser vista na Figura 20. Na figura, se pode ver como os motores que devem operar em velocidades menores entram em operação mais tarde, causando desvios de trajetória. *tenho mais um monte de dados parecidos, para varias*

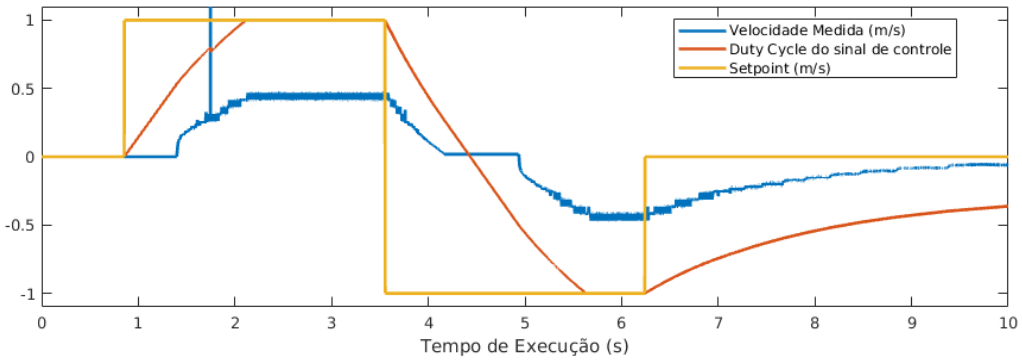

Figura 19: Análise da não-linearidade do tipo “zona-morta” presente nos motores utilizados.

velocidades e vários ganhos

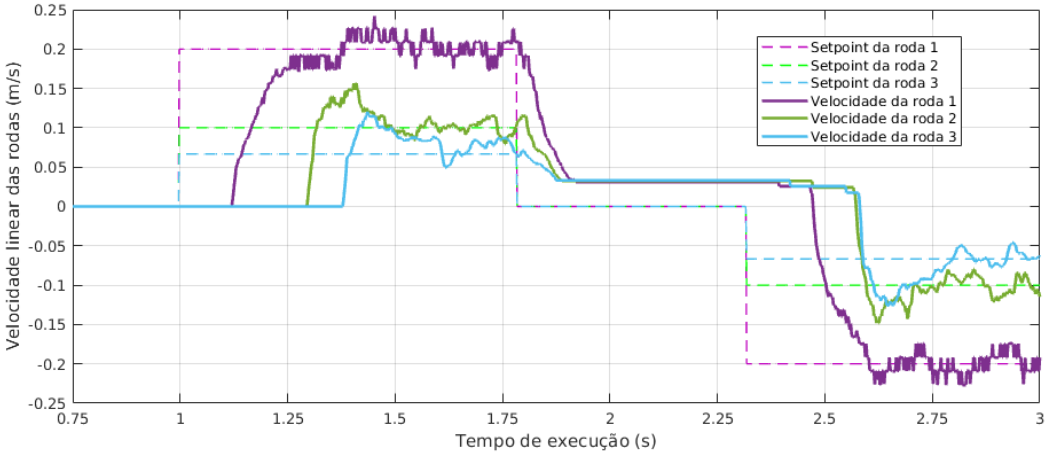

Figura 20: Análise da linearidade do tipo “zona-morta” presente nos motores utilizados.

Cunha (2001) propõe um jeito de se resolver isso. Aplicando esse jeito, se conseguiram melhorar bastante os resultaos, veja na FIgura 21, logo abaix alskdjhalksjdhlaskdjhal.

Desvio padrão do tmepo de execução do loop: 16,68573853

Tempo médio a cada chamada do loop de controle, em μs: 10000,03822

resultados do controle de velocidade

resultados da odometria: Growth of the pose uncertainty for straight-line movement: Note that the uncertainty in y grows much faster than in the direction of movement. This results from the integration of the uncertainty about the robot’s orientation. (Siegwart et al., 2011)

resultados da limitação de velocidade, na Figura 22 sem a limitação, e na Figura 23 com a limitação:

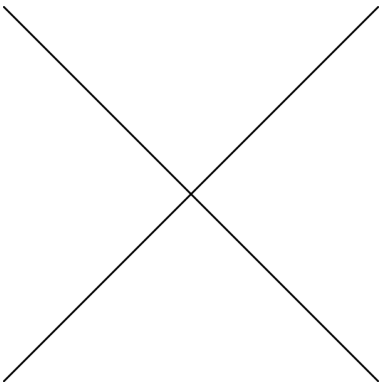

Figura 21: Blabla bla blab balbalbala.

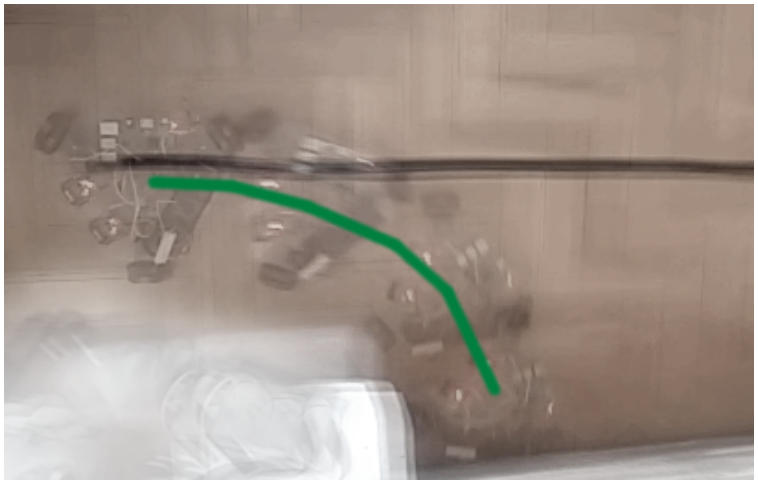

Figura 22: Motores em saturação.

resultados do controle de posição

tempo de duração da bateria

Figura 23: Escalonamento de velocidade ativado.

8 Conclusão e Trabalhos Futuros

Terminar a implementação dos sensores inerciais e da bússola, aplicando algum algoritmo de fusão de sensores a esses dados e a odometria. Cabeamento de 4 vias para a rede I2C da bússola e da IMU.

O chassi podia ser menos espesso, e as peças poderiam ser melhor distribuídas, para utilizar uma área menor da placa. Assim, se poderia instalar mais sensores, ou simplesmente reduzir o tamanho do robô.

Modelagem dinâmica.

Motores melhores.

IMplementar mais trajetórias.

Verificar a temporização do programa, e otimizar. Talvez instalar algum kernel em tempo real, ou utilizar um microprocessador externamente ao comptador para executar os loops de controle.

Caso se deseje utilizar a plataforma projetada para futuros projetos e expansões, talvez seja uma boa ideia organizar melhor os códigos, documentações, verificar se alguma otimização pode ser realizada nas bibliotecas para tornar o software mais robusto, modular e de fácil manutenção. GIthub blaalba.

Láaaaaa no fim, melhorar a interface com o usuário, implementando aplicações práticas para o robô. Exemplos: n sei.

Sensores que utilizam a mesma tensão. Aumentar o número de reguladores ou achar um regulador mais potente para os motores (ou eu posso ligar direto na bateria???). POde ser bacana também adicionar mais uma bateria, exclusivamente para o acionamento dos motores (conforme utilizado em diversos trabalhos, como ACHAR UM TRABALHO).

Os motores representaram 38% do custo total do projeto, e são o componente mais caro de todos. Fica o questionamento.

Processamento na GPU.

É bacana ter a estrutura pronta e bem estudada para saber os seus limites, e a profundidade dos algoritmos que podem ser desenvolvidos para a plataforma. Ao mesmeo tempo, se deve saber onde se deseja chegar com uma certa plataforma, para poder projetá-la para um ifm específico, mas sem esquecer de que ela deve poder ser utilizada para outras coisas também, visto que esse tipo de rrecurso é menos fléxível e mais caro do q desenvover software novo.

De um modo geral, se percebeu a forte interdependência entre todas as partes que compoem um sistema robótico. Se compreendeu que para o desenvolvimento de um projeto em robótica, é necessário que o engenheiro tenha um conhecimento profundo de todas as partes, isoladamente, e que saiba prever a influência de um componente na integração do todo. No caso de um trabalho em equipe, onde cada membro seja responsável por (e especialista em) um tipo específico de

subsistema, ainda assim é necessário entender a influência da sua área nas demais, reforçando o caráter interdisciplinar da área e a importância de uma boa comunicação entre as partes.

Dentre os tipos de subsistema a serem projetados, se destacou a importância de uma boa seleção de hardware, visto que esta parte é bem menos flexível do que o desenvolvimento dos algoritmos de acionamento, por exemplo. No caso em que se deseja alguma modificação de software, basta reprogramar o sistema. Mesmo que tal ato envolva mão-de-obra especializada e tempo de trabalho, não envolve os custos de material, prazos de entrega, disponibilidade comercial e demais requisitos de um sistema físico.

9 Referências

Banggood.com, 2017. Specifications for 6v 210rpm dc motor with encoder and geared reduction. https://www.banggood.com/6V-210RPM-Encoder-Motor-DC-Gear-Motor-with-Mounting-Bracket-and-Wheel-p-1044064.html, acessado em 03/10/2017.

Campion, G., Bastin, G., Dandrea-Novel, B., 1996. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. IEEE transactions on robotics and automation 12 (1), 47–62.

Carrasco, M. M., da Silva, A. L., 2016. Attitude determination for low cost imu and processor board using the methods of triad, kalman filter and allan variance. Revista Brasileira de Iniciação Científica 3 (2), 26–41.

Craig, J., 2017. Introduction to Robotics: Mechanics and Control. Pearson.
URL https://books.google.com.br/books?id=JblZuwAACAAJ

Cunha, M. A. B., 2001. Controle em cascata de um atuador hidráulico: contribuições teóricas e experimentais. Ph.D. thesis, UFSC, Florianópolis, SC, tese de doutorado em Controle, Automação e Informática Industrial.

Feng, D., Friedman, M. B., Krogh, B. H., 1989. The servo-control system for an omnidirectional mobile robot. In: Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on. IEEE, pp. 1566–1571.

Garcia-Saura, C., 2015. Self-calibration of a differential wheeled robot using only a gyroscope and a distance sensor. arXiv preprint arXiv:1509.02154.

Ginzburg, S., Nokleby, S., 2013. Indoor localization of an omni-directional wheeled mobile robot. Transactions of the Canadian Society for Mechanical Engineering. Canada 37, 1043–1056.

Honeywell, 2013. Hmc5883l 3-axis digital compass ic. Acessado em 17/11/2017.
URL https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

Indiveri, G., 2009. Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control (1), 164–171.

Invensense, 2013. Mpu6050. Acessado em 18/11/2017.
URL https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

Jimenez, A. R., Seco, F., Prieto, C., Guevara, J., 2009. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In: Intelligent Signal Processing, 2009. WISP 2009. IEEE International Symposium on. IEEE, pp. 37–42.

Joan, 2017. Reference for pigpio c interface. Acessado em 27/11/2017.
URL http://abyz.me.uk/rpi/pigpio/cif.html

Jung, M.-J., Kim, J.-H., 2001. Fault tolerant control strategy for omnikity-iii. In: Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Vol. 4. IEEE, pp. 3370–3375.

Kim, H., Kim, B. K., 2014. Online minimum-energy trajectory planning and control on a straight-line path for three-wheeled omnidirectional mobile robots. IEEE Transactions on industrial electronics 61 (9), 4771–4779.

Krinkin, K., Stotskaya, E., Stotskiy, Y., 2015. Design and implementation raspberry pi-based omni-wheel mobile robot. In: Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT), 2015. IEEE, pp. 39–45.

Loh, W., Low, K. H., Leow, Y., 2003. Mechatronics design and kinematic modelling of a singularityless omni-directional wheeled mobile robot. In: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. Vol. 3. IEEE, pp. 3237–3242.

Lynch, K., Park, F., 2017. Modern Robotics: Mechanics, Planning, and Control. Cambridge University Press.
URL https://books.google.com.br/books?id=86G0nQAACAAJ

Markets, Markets, 2015. Mobile robots market by environment (aerial, ground, and marine), component (hardware and software), application (professional service and personal service), and geography (north america, europe, asia-pacific, and the row) - global forecast to 2020. http://www.marketsandmarkets.com/Market-Reports/mobile-robots-market-43703276.html, acessado em 27/09/2017.

MPS, 2008. Synchronous rectified step-down converter. Acessado em 30/11/2017.
URL https://cdn.solarbotics.com/products/datasheets/mp2307.pdf

Nexus, 2016. 58mm plastic omniwheel. Acessado em 8/10/2017.
URL http://www.nexusrobot.com/product/58mm-plastic-omni-wheel-for-lego-nxt-and-servo-motor-14135.html

Oubbati, M., Schanz, M., Buchheim, T., Levi, P., 2005. Velocity control of an omnidirectional robocup player with recurrent neural networks. In: RoboCup. Springer, pp. 691–701.

Park, K., Chung, D., Chung, H., Lee, J. G., 1996. Dead reckoning navigation of a mobile robot using an indirect kalman filter. In: Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on. IEEE, pp. 132–138.

Pin, F. G., Killough, S. M., 1994. A new family of omnidirectional and holonomic wheeled platforms for mobile robots. IEEE transactions on robotics and automation 10 (4), 480–489.

Ritter, G. A., 2016. Modelo genérico de plataforma robótica omnidirecional em código aberto. Ph.D. thesis, UNISC, Santa Cruz do Sul, RS, trabalho de Conclusão de Curso de Engenharia de Computação.

Röhrig, C., Heß, D., Kirsch, C., Künemund, F., 2010. Localization of an omnidirectional transport robot using ieee 802.15. 4a ranging and laser range finder. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE, pp. 3798–3803.

Rojas, R., Förster, A. G., 2006. Holonomic control of a robot with an omnidirectional drive. KI-Künstliche Intelligenz 20 (2), 12–17.

Samani, H. A., Abdollahi, A., Ostadi, H., DaneshPanah, M., 2007. Comprehensive omni-directional soccer player robots. International Journal of Advanced Robotic Systems.

Semiconductors, P., 2000. The i2c-bus specification. Philips Semiconductors 9397 (750), 00954.

Siciliano, B., Khatib, O., 2016. Springer handbook of robotics. Springer.

Siegwart, R., Nourbakhsh, I., Scaramuzza, D., 2011. Introduction to Autonomous Mobile Robots. Intelligent robotics and autonomous agents. MIT Press.
URL https://books.google.com.br/books?id=4of6AQAAQBAJ

Steinhoff, U., Schiele, B., 2010. Dead reckoning from the pocket-an experimental study. In: Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on. IEEE, pp. 162–170.

STMicroelectronics, 2000. Dual full-bridge driver. Acessado em 30/11/2017.
URL http://www.st.com/content/ccc/resource/technical/document/datasheet/82/cc/3f/39/0a/29/4d/f0/CD00000240.pdf/files/CD00000240.pdf/jcr:content/translations/en.CD00000240.pdf

Takemura, Y., Sanada, A., Ichinose, T., Nakano, Y., Nassiraei, A. A., Azeura, K., Kitazumi, Y., Ogawa, Y., Godler, I., Ishii, K., et al., 2007. Development of "hibikino-musashi" omni-directional mobile robot. In: International Congress Series. Vol. 1301. Elsevier, pp. 201–205.

Treesatayapun, C., 2011. A discrete-time stable controller for an omni-directional mobile robot based on an approximated model. Control Engineering Practice 19 (2), 194–203.

Upton, E., Halfacree, G., 2014. Raspberry Pi user guide. John Wiley & Sons.

Williams, R. L., Carter, B. E., Gallina, P., Rosati, G., 2002. Dynamic model with slip for wheeled omnidirectional robots. IEEE transactions on Robotics and Automation 18 (3), 285–293.

Apêndices

A Custo dos componentes utilizados

Há no mercado uma variada gama de componentes a serem utilizados em projetos robóticos. Para o projeto em questão, se utilizaram componentes que mostrassem um preço de mercado competitivo e grande disponibilidade. Dessa forma, se pode manter o projeto viável, mesmo que por vezes sacrificando um possível incremento de desempenho que se daria ao utilizar um componente mais robusto, por exemplo.

Na Tabela 1 se pode ver a lista de componentes adquirida e os respectivos custos. Nota-se que no caso das 3 rodas há – integrado ao valor apresentado – as taxas de importação e conversão de moedas, visto que esses componentes foram importados dos Estados Unidos. Também é importante mencionar que o chassi foi produzido sem custo utilizando a estrutura da universidade.

Tabela 1: Custo dos componentes utilizados no projeto.

Item:	**Valor por unidade:**	**Quantidade:**	**Valor total:**
Omniwheel	R$ 46,76	3	R$ 140,29
Motor c/ encoder	R$ 119,00	3	R$ 357,00
Driver	R$ 15,00	2	R$ 30,00
Raspberry Pi	R$ 148,00	1	R$ 148,00
microSD 16GB	R$ 34,00	1	R$ 34,00
IMU MPU6050	R$ 9,00	1	R$ 9,00
Magnetômetro HMC5883	R$ 12,80	1	R$ 12,80
Placa de Acrílico	R$ 25,00	1	R$ 25,00
Bateria	R$ 120,00	1	R$ 120,00
Reguladores de Tensão	R$ 4,98	3	R$ 14,97
Parafusos	R$ 14,00	X	R$ 14,00
Diversos	R$ 30,00	X	R$ 30,00
		Custo Total:	R$ 935,06

B Função do loop de controle

```
     void loop (void)
blabalbalablbalab
blablablbbablaba
balablababalbalbabla
blabalbalbalbablabla
lbablablablablblablblablbla
```

C Dimensões da estrutura mecânica

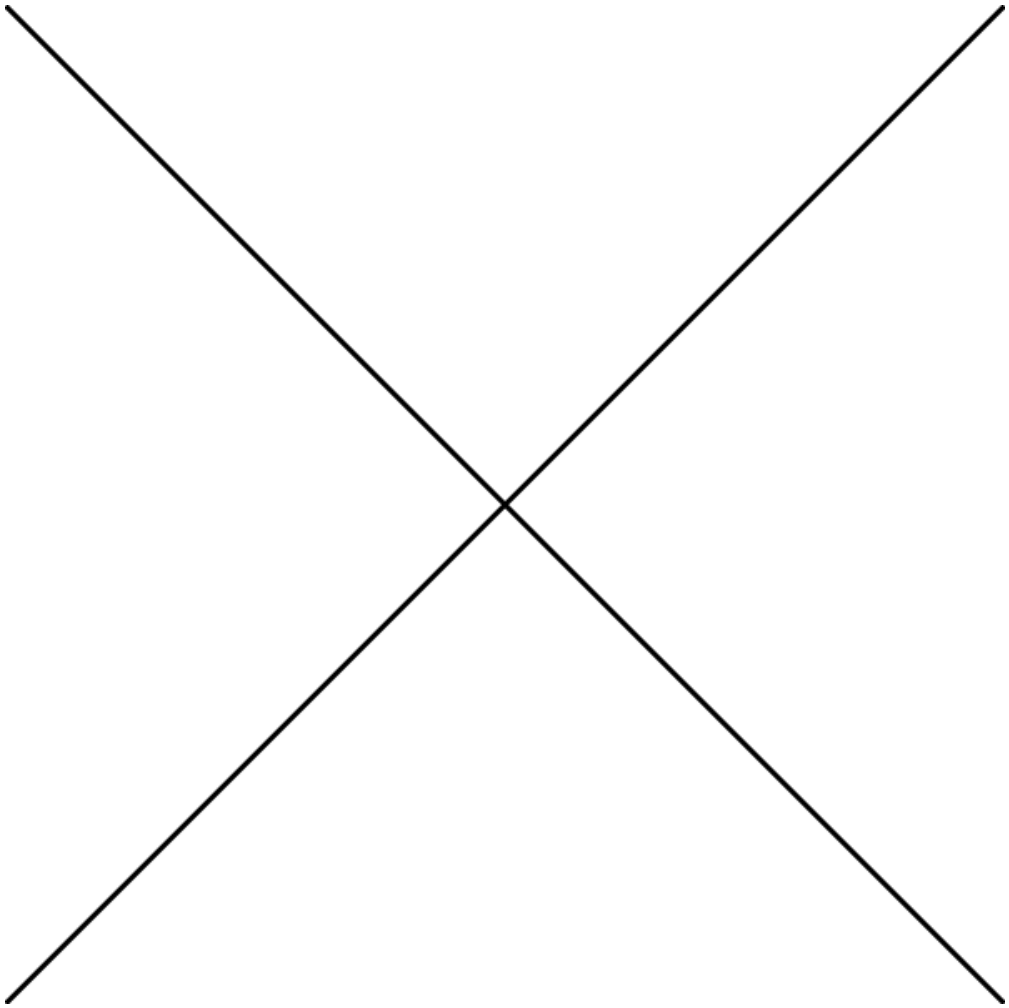

Figura 24: Dimensões dos elementos do chassi. TROCAR FIGURA