

Centro Universitário do Instituto Mauá de Tecnologia
Curso de Bacharelado em Engenharia de Controle e Automação

iStalker - Robô Autoequilibrado Atuado por Duas Rodas

**Felipe de Souza Leal
Guilherme Chinellato
Régis Henrique Munhoz da Cruz**

São Caetano do Sul, SP, 10 de dezembro de 2018

Felipe de Souza Leal
Guilherme Chinellato
Régis Henrique Munhoz da Cruz

iStalker - Robô Autoequilibrado Atuado por Duas Rodas

Trabalho de Conclusão de Curso
Monografia apresentada ao Instituto Mauá de Tecnologia (IMT) como um dos pré-requisitos para a obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Centro Universitário do Instituto Mauá de Tecnologia - IMT

Orientador: Prof. Dr. Rodrigo Alvite Romano

São Caetano do Sul, SP
10 de dezembro de 2018

Felipe de Souza Leal
Guilherme Chinellato
Régis Henrique Munhoz da Cruz

iStalker - Robô Autoequilibrado Atuado por Duas Rodas

Trabalho de Conclusão de Curso

Monografia apresentada ao Instituto Mauá de Tecnologia (IMT) como um dos pré-requisitos para a obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Prof. Dr. Rodrigo Alvite Romano
Orientador

Prof. Ma. Andressa Corrente Martins
Avaliador

Prof. Me. Murilo Zanini de Carvalho
Avaliador

São Caetano do Sul, SP
10 de dezembro de 2018

Aos nossos familiares pelo apoio, paciência, confiança e compreensão nas ausências em decorrência do tempo despendido para elaboração desse trabalho.

Agradecimentos

Agradecemos ao professor orientador Doutor Rodrigo Alvite Romano pela orientação, paciência, conselhos e tempo despendido ao nosso trabalho.

Especialmente aos professores Anderson Harayashiki Moreira, Sérgio Luís Rabelo De Almeida, Gelson Freitas Miori pelas indicações práticas, disponibilidade e pelo auxílio na validação e interpretação mecânica do projeto, no esclarecimento de dúvidas e auxílio nas análises estruturais do projeto.

Ao professor Eduardo Lobo Lustosa Cabral pelas opiniões sinceras e práticas que nos evitou desperdício de tempo e desvio dos resultados desejados.

E aos professores Murilo Zanini de Carvalho e Andressa Corrente Martins pelas aulas, aceite do convite para compor a banca avaliadora e contribuições apresentadas, enriquecendo demasiadamente o trabalho.

*“Se eu vi mais longe,
foi por estar sobre ombros de gigantes.”*
(Sir Issac Newton)

Resumo

O problema do envelhecimento da população gera a necessidade do desenvolvimento de tecnologias que melhorem a mobilidade e a acessibilidade. Soluções baseadas em robôs móveis têm se mostrado atraente para lidar com essa demanda. Neste trabalho descrevem-se a justificativa, as vantagens e desvantagens da escolha por um robô móvel autoequilibrado por duas rodas através da apresentação das diferentes formas de locomoção dos robôs existentes. Com base na revisão da literatura dos robôs autoequilibrados, pode-se escolher a melhor disposição dos componentes, a topologia das partes eletrônicas, ponderar as estratégias de estabilização do robô e integrar todas essas partes de maneira funcional. A arquitetura de software é composta por dois subsistemas baseados em sistemas operacionais de tempo real. Um primeiro implementa a interface com os atuadores (motores) e sensores de movimento, além de realizar as leis de controle necessárias para a locomoção do robô. O segundo provê uma plataforma para a execução de aplicações como, transmissão de vídeo via rádio, conexão com controle Bluetooth e comunicação com a plataforma via rede sem fio. O sistema foi avaliado a partir de ensaios de início de operação para mensurar o tempo necessário para autoequilibrar, testes de impulsos aleatório para simular o comportamento a partir de pertubações e por fim teste de operação completo com todos sistemas em funcionamento, com ênfase para a transmissão de vídeo.

Palavras-chaves: Pêndulo invertido; robô autoequilibrado; sistemas de controle; sistemas dinâmicos; controlador PID em cascata. *RTOS*; sistemas embarcados; eletrônica; *software*.

Abstract

The problem of population ageing triggers the need for the development of technologies that improve mobility and accessibility. Solutions based on mobile robots have proven themselves to be appealing to handle such demand. On this assignment, it is described the justification, the advantages and disadvantages of choosing a mobile robot operated by two wheels through the presentation of different ways of motion of existing robots. From the literature that describes self-balancing robots, it is possible to choose the best component arrangement, the electronic parts topology, evaluating the strategies of robot stabilization and how to merge all these parts in a functional manner. The software architecture is composed by two subsystems based on real time operational systems. The first one establishes the interface with the actuators (motors) and motion sensors, besides carrying out the necessary controlling laws for the robot's locomotion. The second one establishes a platform to execute applications such as video streaming via radio, connection with Bluetooth controlling and wireless communication with the platform. The system has been evaluated from the initial operating rehearsals in order to measure the necessary time to autobalance, tests of random impulses to simulate behaviors from disturbances and, finally, complete operation test with the working systems, with particular emphasis upon video streaming.

Keywords: inverted pendulum. self-balancing robot. control systems. dynamics systems. Cascade PID Control. RTOS. embedded systems. electronics; software.

Listas de ilustrações

Figura 1 – Veículos apresentados pelo PACE em 2018.	1
Figura 2 – XD - Veículo desenvolvido pelo IMT.	2
Figura 3 – EMIEW3.	3
Figura 4 – POD 050 B.	5
Figura 5 – Atlas.	6
Figura 6 – MiP.	7
Figura 7 – Robô desenvolvido por Memarbashi.	8
Figura 8 – JOE, robô desenvolvido por Grasser.	9
Figura 9 – Diagrama de blocos eletrônicos.	9
Figura 10 – Filtro complementar.	10
Figura 11 – Diagrama de PID.	10
Figura 12 – Gerenciador de Tarefas.	11
Figura 13 – Projeto de estrutura mecânica.	13
Figura 14 – Protótipos desenvolvidos pelo grupo em anos anteriores.	14
Figura 15 – Design final da estrutura do projeto.	14
Figura 16 – Projeto da base do robô.	15
Figura 17 – Fixação da estrutura do corpo e dos componentes eletrônicos.	16
Figura 18 – Disposição dos componentes sobre a estrutura.	17
Figura 19 – Estrutura e componentes compõem o conjunto da cabeça.	17
Figura 20 – Modelo dinâmico do pêndulo invertido sobre rodas.	18
Figura 21 – Localização do centro de gravidade l e massa m do modelo.	19
Figura 22 – Comparação entre materiais para impressão 3D.	21
Figura 23 – Comparação entre projeto virtual e real.	22
Figura 24 – Diagrama de Componentes Eletrônicos.	23
Figura 25 – Camadas do sistema.	26
Figura 26 – Fluxo das tarefas.	27
Figura 27 – Orientação do IMU.	28
Figura 28 – Filtro complementar.	29
Figura 29 – Canal A e B do encoder acoplado ao eixo de um motor.	30
Figura 30 – Malha de controle.	31
Figura 31 – Ângulos de Referência da Malha de Controle.	32
Figura 32 – UML da classe PID.	32
Figura 33 – Wifi Monitor.	35
Figura 34 – Diagrama do sistema.	36
Figura 35 – Fluxograma do microprocessador.	37
Figura 36 – Fluxograma do microcontrolador.	38

Figura 37 – iStalker no início de operação.	40
Figura 38 – Ensaio de início de operação.	40
Figura 39 – Ensaio de impulsos aleatórios.	41
Figura 40 – Demonstração prática das aplicações.	42
Figura 41 – Vista explodida da estrutura mecânica do projeto	49
Figura 42 – Dimensões finais da estrutura do iStalker	51
Figura 43 – Simulação de colisão do corpo com obstáculo	53
Figura 44 – Análise da deformação total no teste de impacto	53
Figura 45 – Análise das tensões de Von-Mises	53
Figura 46 – Processo de Montagem do Modelo	55
Figura 47 – Projeto finalizado ja em testes e projeto na Eureka, respectivamente . .	57
Figura 48 – Comandos programados no controle remoto via software embarcado . .	59

Lista de tabelas

Tabela 1 – Especificações gerais do motor 37D x 68L Pololu	20
Tabela 2 – Consumo médio dos componentes eletrônicos em superfície plana e lisa	25
Tabela 3 – Constantes dos controladores.	39
Tabela 4 – Especificações gerais do Projeto	61

Sumário

1	INTRODUÇÃO	1
1.1	Motivação e Justificativa	2
1.2	Objetivos	4
2	REVISÃO DA LITERATURA	5
3	MATERIAIS E MÉTODOS	13
3.1	Projeto Mecânico	13
3.1.1	Design e Estrutura Mecânica do Projeto	13
3.1.2	Base	15
3.1.3	Corpo	15
3.1.4	Cabeça	17
3.1.5	Definição e dimensionamento dos atuadores	18
3.1.6	Definição do Material	21
3.2	Projeto Eletrônico	23
3.2.1	Microcontrolador	23
3.2.2	Microprocessador	24
3.2.3	IMU	24
3.2.4	Encoder	24
3.2.5	Câmera	25
3.2.6	Bateria	25
3.2.7	Conversor DC-DC	25
3.3	Projeto de Software	25
3.3.1	Leitura do Sensor Inercia	28
3.3.2	Filtro Complementar	28
3.3.3	Aquisição do Encoder	29
3.3.4	Processamento da Malha de Controle	30
3.3.5	Atuação dos Motores	34
3.3.6	Interface com Raspberry	34
3.3.7	Sintonização dos controladores	34
3.3.8	Aplicação de Monitoramento	35
3.3.9	Fluxograma e Diagramas	35
4	RESULTADOS	39
4.1	Constantes dos controladores	39
4.2	Ensaios	39

4.2.1	Ensaio de Início de Operação	40
4.2.2	Ensaio de Impulsos Aleatórios	41
4.3	Demonstração prática das aplicações	42
4.4	Discussão dos Resultados	42
5	CONCLUSÕES E TRABALHOS FUTUROS	43
	REFERÊNCIAS	45
	APÊNDICES	47
	APÊNDICE A – VISTA EXPLODIDA DO PROJETO	49
	APÊNDICE B – DIMENSÕES FINAIS DO ISTALKER	51
	APÊNDICE C – TESTE DE IMPACTO	53
	APÊNDICE D – MONTAGEM DO MODELO	55
	APÊNDICE E – PROTÓTIPO FINALIZADO	57
	APÊNDICE F – AÇÕES DO CONTROLE REMOTO	59
	APÊNDICE G – ESPECIFICAÇÕES DO PROJETO	61

1 INTRODUÇÃO

O envelhecimento da população é um dos principais problemas demográficos estudados atualmente, acredita-se que nos próximos 40 anos o número de pessoas acima dos 60 anos irá triplicar, isso atinge mais que questões de sustentabilidade e impactos ambientais [1], aflora preocupações com a qualidade de vida dessa parcela da população.

Diante do cenário apresentado acima, são necessárias medidas para tornar o cotidiano dessas pessoas o mais confortável possível e, saindo do aspecto sócio-político, tecnologias que auxiliam a mobilidade e acessibilidade tornam-se cada vez mais frequentes e, constantemente novos projetos são apresentados.

A *General Motors Co.* possui um programa bienal que tem como objetivo, o desenvolvimento de projetos de engenharia educacionais e colaborativos, o PACE, Parceiros para o Avanço de Educação Colaborativa em Engenharia, do Inglês, *Partners for the Advancement of Collaborative Engineering Education*. Neste programa, universidades do mundo inteiro participam de fóruns e competições divididas em times multinacionais, os quais são responsáveis por apresentar seus projetos no primeiro ano, em um fórum, onde já recebem *feedbacks*, e no ano subsequente, apresentam o protótipo proposto para avaliação e por fim premiações.

O PACE, em 2016, teve como tema o PUMA, Acesso Pessoal de Mobilidade Urbana, do Inglês, *Personal Urban Mobility Access*, o nome do tema é autoexplicativo, cada time tinha como missão criar um veículo que auxiliasse a acessibilidade e mobilidade urbana e tinham como público alvo pessoas com mais de 60 anos, corroborando a problemática exposta anteriormente, 6 veículos foram apresentados no final do projeto em julho 2018, como mostra a Figura 1.

Figura 1 – Veículos apresentados pelo PACE em 2018.



Fonte: <https://pacepartners.org/>

Os projetos de acessibilidade e mobilidade, como os aqui apresentados, em sua grande maioria não levam em consideração que os locais públicos estão cada vez mais lotados e não é comum apresentarem capacidade de mudança de modos de operação para facilitar sua utilização.

A limitação de aplicações também pode ser agregada a lista de elementos faltantes nos projetos em desenvolvimento, em outras palavras, eles atendem a necessidade principal mas não possibilitam a interatividade e nem a adição de periféricos que possam, por exemplo, se comunicar com smartphones, fornecendo informações diversas, como situação da bateria, localização geográfica, meteorologia, aplicações de navegação, entre outras dezenas que somente a integração com telefone celular já possibilitaria.

Pensar em locomoção, acessibilidade e ainda integração de aplicações, conduz o estudo aos robôs móveis, mas há vários tipos desses, e como já mencionado o espaço é uma variável essencial para a escolha. Portanto, a solução estudada neste trabalho será o robô autoequilibrado atuado por duas rodas motorizadas.

1.1 Motivação e Justificativa

Os autores deste trabalho, no final de 2017, foram convidados pelo Instituto Mauá de Tecnologia a adicionar um modo autoequilibrado a um veículo de três rodas sendo duas dianteiras motorizadas e responsáveis pela direção, e uma traseira que no seu modo secundário, citado acima, era recolhida para dentro do veículo, o XD (Figura 2), projeto apresentado no fórum em julho de 2017, realizado em Detroit pela *General Motors Co.*, obtendo 4 premiações.

Figura 2 – XD - Veículo desenvolvido pelo IMT.



A adição do modo autoequilibrado se deu pela necessidade de manobrar o veículo quando este não estivesse sendo utilizado para sua aplicação primária – assistência à pessoas com mobilidade reduzida. Como o veículo já possuía duas rodas atuadas, desenvolver

um novo modo autoequilibrado era uma solução eficiente já que o sistema em questão conseguiria mudar sua direção ocupando somente seu espaço físico em revolução, o que facilitaria a movimentação de seu usuário junto ao veículo em lugares apertados.

Robôs autoequilibrados atuados por duas rodas podem servir como guias em lugares públicos, assim como o robô EMIEW3 (Figura 3), desenvolvido pelo grupo japonês Hitachi, que portado de vários tipos de sensores, é capaz de interagir e se comunicar com pessoas e ainda, trocar dados em tempo real com outros dispositivos utilizando sistemas baseados em nuvem. Estes robôs podem também servir para monitoramento em tempo real de lugares inacessíveis ou perigosos aos usuários.

Figura 3 – EMIEW3.



Fonte: <http://www.hitachi.com.br/>

Além da aplicação supracitada pode-se pontuar o carregamento de ferramentas, bagagens e compras. Em outras palavras, uma variedade de aplicações que aumentam a eficiência das atividades cotidianas não só para pessoas com ou sem mobilidade reduzida, mas também no comércio e na indústria, servindo como assistente nos processos com alta capacidade de se locomover na planta onde for solicitado, ou simplesmente servir como um brinquedo que o usuário pode controlá-lo remotamente e ter acesso à suas imagens em tempo real.

1.2 Objetivos

Projetar e construir um robô autoequilibrado por um sistema embarcado integrado a um sistema operacional de tempo real, possibilitando a implementação de uma plataforma de desenvolvimento e assim valida-la com uma aplicação de monitoramento.

2 REVISÃO DA LITERATURA

Os robôs configuraram um avanço muito importante na indústria. A maioria dos robôs industriais ou manipuladores possuem uma limitação fundamental, a falta de mobilidade, que reduz suas aplicabilidades nos pontos onde estão instalados. Em contrapartida, um robô móvel é capaz de realizar o trabalho para que fora concebido de modo mais eficiente [2].

Há uma multiplicidade de aplicações que os robôs móveis permitem, por exemplo inspecionar um duto de ar em condições impróprias para um ser humano, torna-se uma atividade extremamente mais simples [2] pois pode ser facilmente resolvida por um robô dotado de câmeras e outros tipos de sensores embarcados. É o caso, por exemplo, do robô POD 050 B, desenvolvido pela M-Tecks Robotics (Figura 4).

Figura 4 – POD 050 B.



Fonte: M-Tecks Robotics

Robôs móveis necessitam de mecanismos de locomoção, mas há uma vasta variedade de forma de move-los e selecionar a forma de atua-lo é um aspecto essencial para o projeto. A maioria dos mecanismos de locomoção são inspirados em movimentos de animais, no entanto, a roda atuada por motores foi uma invenção que alcançou uma grande eficiência em terrenos planos, além de ser relativamente simples de implementar mecanicamente, torna-se indiscutivelmente o mecanismo de locomoção mais popular [2].

A locomoção é um complemento da manipulação. Os braços robóticos são fixos e imprimem forças nos objetos para move-los no ambiente de trabalho. Em contra partida, nos robôs com locomoção, a força é aplicada sob o ambiente que é fixo. Os dois tipos de robôs móveis mais estudados são os locomovidos por pernas e os locomovidos por rodas,

ambos compartilham dos mesmos "problemas fundamentais" de estabilidade, características de contato e tipos de ambientes [2]:

- Estabilidade:
 - quantidade e geometria dos pontos de contato;
 - centro de gravidade;
 - estabilidade dinâmica/estática;
 - inclinação do solo.
- Características de contato:
 - ponto/superfície de contato e sua forma;
 - ângulo de contato;
 - atrito.
- Tipo de ambiente:
 - estrutura;
 - meio (água, ar, terrenos macios ou duros).

A locomoção por pernas é caracterizada por uma série de pontos de contato entre o robô e o solo, como exemplifica o robô Atlas produzido pela empresa Boston Dynamics (Figura 5), que apesar de estar sobre duas superfícies de contato, possui mais duas. Suas principais vantagens são a adaptabilidade e manobrabilidade em terrenos irregulares.

Figura 5 – Atlas.



Fonte: Boston Dynamics

As principais desvantagens da locomoção por pernas incluem energia e complexidade mecânica. Além do mais, as vantagens de manobrabilidade somente serão alcançadas se as pernas possuírem graus de liberdade suficientes para imprimir forças em diferentes direções, sendo necessário também ressaltar que sua estabilidade está diretamente ligada a quantidade de pernas do robô [2].

A roda é o mecanismo de locomoção mais popular nos robôs móveis, são eficientes e, comparadas com os mecanismos supracitados, são mais simples de implementar. Sendo assim, o objeto de estudo deste trabalho é um robô do tipo pendulo invertido atuado por rodas (WIP), do Inglês, *Wheeled Inverted Pendulum*, que possui movimentação intrinsecamente "desajeitada" para estabilizar sua postura. Esses tipos de robôs, em sua maioria, são constituídos de um corpo com duas rodas, (Figura 6) para realizar movimentos planos similar ao caminhar do homem. A motorização das rodas é feita de forma independente para realizar o controle de posição e ações similares ao movimento humano [3].

Figura 6 – MiP.



Fonte: <https://wowwee.com/mip>

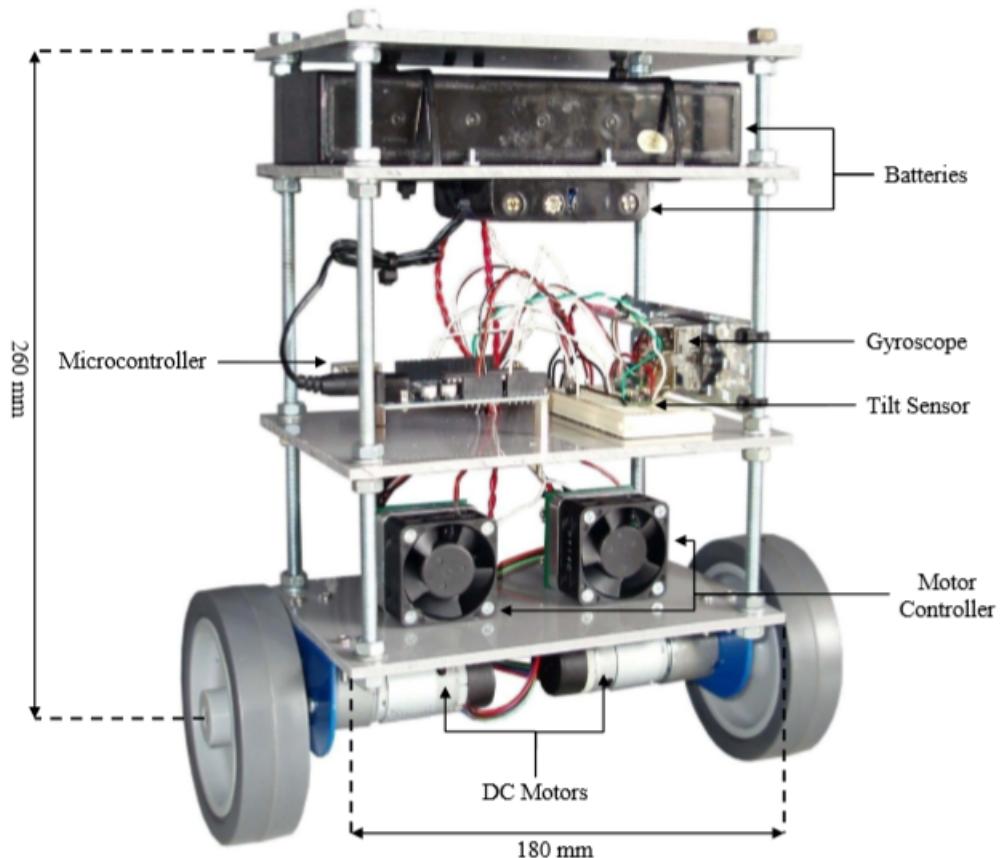
O controle desse tipo de robô é complexo pois trata-se de um sistema sub-atuado, ou seja, possuí menor número de entradas de controle do que graus de liberdade. Nesses tipos de sistemas, as variáveis não atuadas são controladas indiretamente pelas variáveis atuadas por acoplamento dinâmico. Sistemas eletromecânicos sub-atuados geralmente são introduzidos na etapa do projeto a fim de reduzir o custo de fabricação, peso e índice de falha, de modo que o robô realize tarefas complexas utilizando um menor número de atuadores. Não obstante, esses sistemas podem necessitar de novas abordagens para projetar estratégias de controle efetivas [3].

A melhor forma de projetar robôs do tipo WIP é combinando conceitos de *software*, *hardware* e mecânica que vão ao encontro do projeto de sistemas embarcados para realizar

funções específicas e dedicadas, sendo possível integrar sistemas operacionais afim de obter repostas em tempo real e definir prioridades de aplicações [4]. No caso dos WIPs, manter-se equilibrado é a principal prioridade.

A disposição dos componentes que compõem a estrutura mecânica do robô não difere dos trabalhos consultados, é natural entre eles que sigam o modelo teórico do pêndulo invertido sobre um carro. A dissertação [5] mostra que a disposição inicial considera as partes mais pesadas localizadas mais distante do eixo das rodas, os motores acoplados abaixo do primeiro nível da estrutura e os outros componentes distribuídos nos níveis intermediários, como mostra a Figura 5.

Figura 7 – Robô desenvolvido por Memarbashi.



Fonte: (MEMARBASHI, 2010)

No artigo [6], é corroborada a disposição dos componentes eleita pelo trabalho de Memarbashi, tanto que, a extremidade superior do robô é propositalmente mais pesada pela adição de material, como mostra a Figura 8.

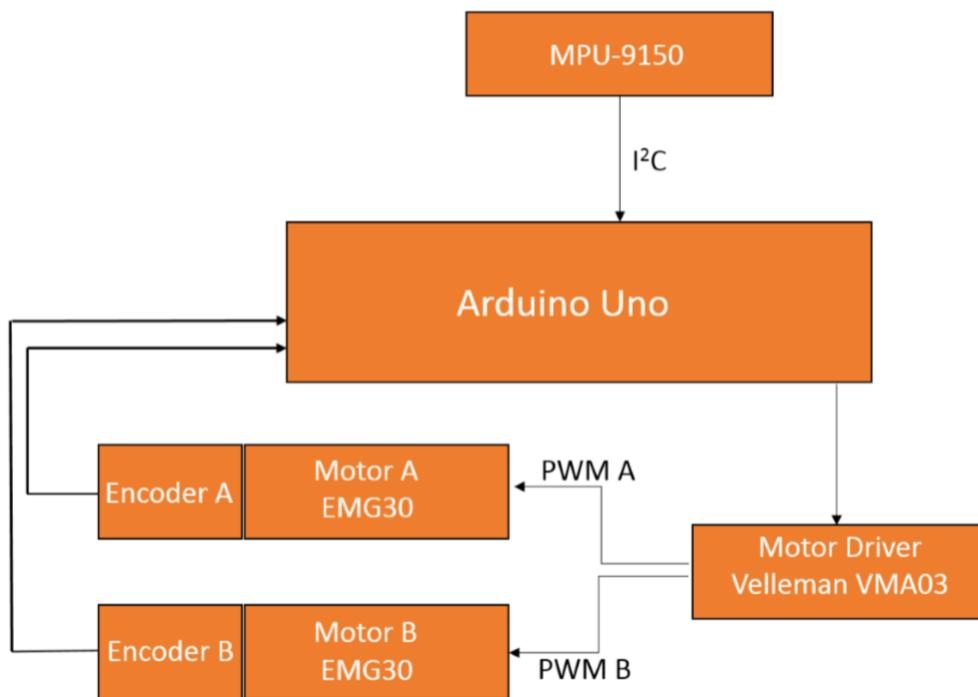
Figura 8 – JOE, robô desenvolvido por Grassser.



Fonte: (Grassser, 2002)

No que se refere à eletrônica, o diagrama de blocos representados pela Figura 9, demonstra a utilização do microcontrolador para aquisição de dados dos sensores e envio de sinais para o *driver*, que por sua vez atua os motores de corrente contínua [7].

Figura 9 – Diagrama de blocos eletrônicos.

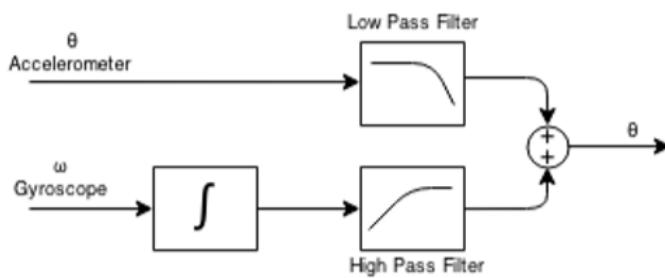


Fonte: (Hellman, 2015)

No artigo [8], se utiliza da mesma hierarquia eletrônica, no entanto, ressalta a importância de implementar a fusão dos dados obtidos pelo giroscópio e acelerômetro para obter uma medida de ângulo precisa e estável.

A fusão dos sensores é realizada implementando um filtro complementar (Figura 10) ao qual aplica um filtro passa-alta ao o giroscópio que, apesar de possuir medida instantânea e precisa, ao decorrer do tempo, acumula erros devido ao efeito integrador. É ainda necessária a presença de um filtro passa-baixa para o acelerômetro já que este é muito sensível às forças aplicadas, produzindo sinais ruidosos [8].

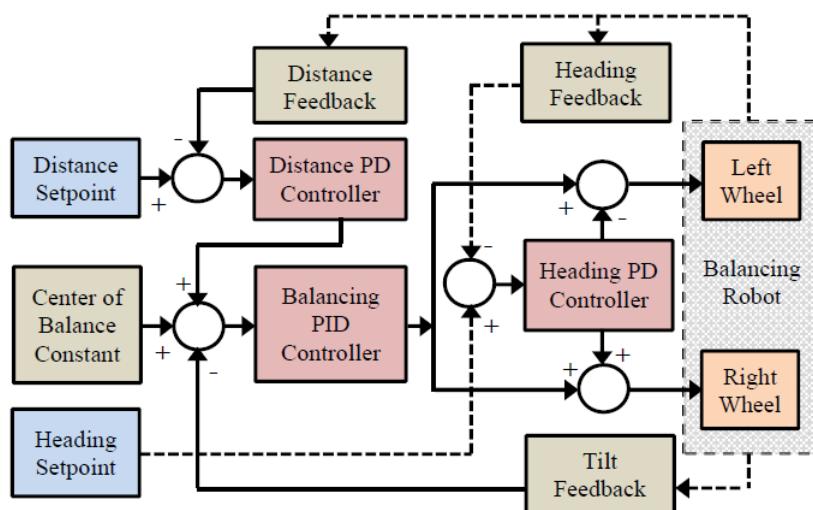
Figura 10 – Filtro complementar.



Fonte: (Jodensvi, 2015)

Estudada a estrutura, a topologia eletrônica e garantida as medidas precisas e estáveis dos sensores, necessita-se elaborar uma estratégia de controle capaz de manter o robô autoequilibrado. No artigo [9], a arquitetura de controle, ilustrada na Figura 11, utiliza de controladores PID em cascata, para equilibrar e movimentar o robô.

Figura 11 – Diagrama de PID.



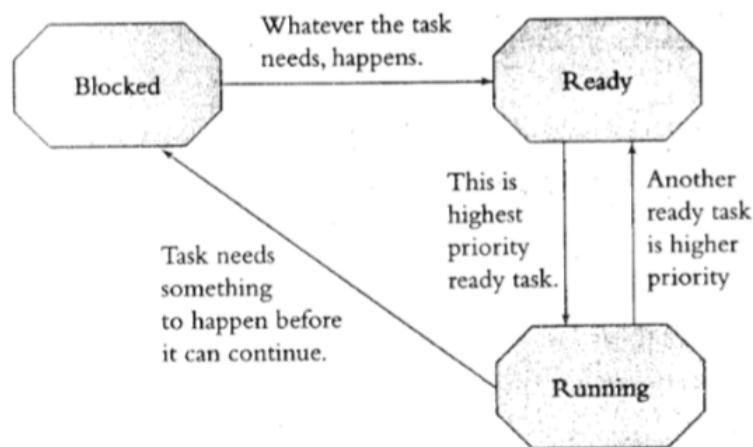
Fonte: (Pratama, 2015)

Para implementação do controlador ilustrado anteriormente, o autor considera o centro de gravidade no centro geométrico do robô quando este está a 90 graus em relação ao solo e então, define manualmente o "setpoint" para o controlador de equilíbrio. A malha de controle mencionada tem como principal função manter o robô equilibrado através da realimentação do sensor, representado no diagrama da Figura 11 por "*Tilt Feedback*". O controle de distância (*Distance PD Controller*) é utilizado como parâmetro para movimentação para frente e para trás, e a saída deste afeta diretamente o erro na entrada do controlador de equilíbrio. Já o rumo do robô é dado pelo controlador de direção, representado no diagrama pelo bloco "*Heading PD Controller*".

Para garantir um tempo de amostragem determinístico a fim de priorizar funções com determinismo como por exemplo, anteferir o cálculo da malha de controle a uma aplicação de comunicação remota sem prejuízo do mesmo, faz-se necessário o uso de um sistema operacional de tempo real, do Inglês, RTOS. Tal sistema garante a possibilidade de aplicação de conceitos de multitarefas, do Inglês *multitasking*, como: *task states*, *task priorities*, *task synchronization* e *message queues*, demonstrados na literatura [4].

O gerenciador de tarefas, do Inglês, *Scheduler*, coordena o estado de cada tarefa, decidindo qual será executada de acordo com a sua prioridade, como ilustra a Figura 12. Se uma tarefa de maior prioridade está pronta para ser executada ("Ready"), o gerenciador bloqueia a tarefa atual que está em execução, "*Running*", e a move para o estado de bloqueio, "*Blocked*". A tarefa "*Ready*" passa então para o estado em execução [10].

Figura 12 – Gerenciador de Tarefas.



Fonte: (Simon, 1999)

3 MATERIAIS E MÉTODOS

Embora todo o projeto tenha sido desenvolvido "simultaneamente", com o propósito de melhorar a organização e apresentação deste documento, dividiu-se o mesmo em três projetos principais: mecânico, eletrônico e de *software*.

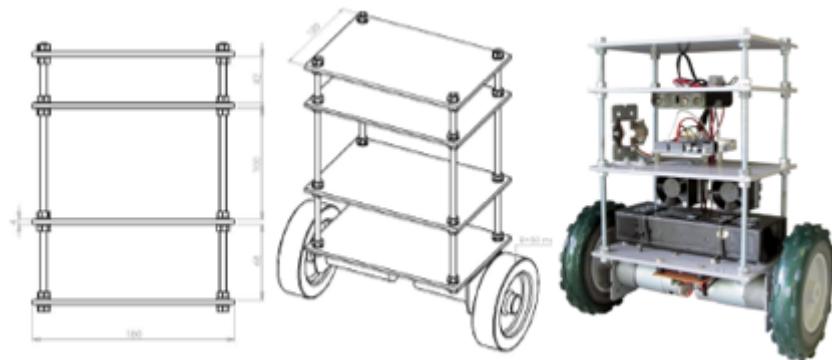
3.1 Projeto Mecânico

Com o intuito de projetar uma estrutura homogênea, com design atraente, montagem simples, e que atendesse aos objetivos do projeto, planejou-se de maneira estratégica a estrutura física do robô.

3.1.1 Design e Estrutura Mecânica do Projeto

Conforme visto na dissertação [5], o autor constrói em sua obra um robô em formato de "prateleiras" (Figura 13) e destaca que alguns fatores como a localização do centro de massa do robô ou mesmo o tipo de rodas pode influenciar diretamente na estabilidade do modelo.

Figura 13 – Projeto de estrutura mecânica.

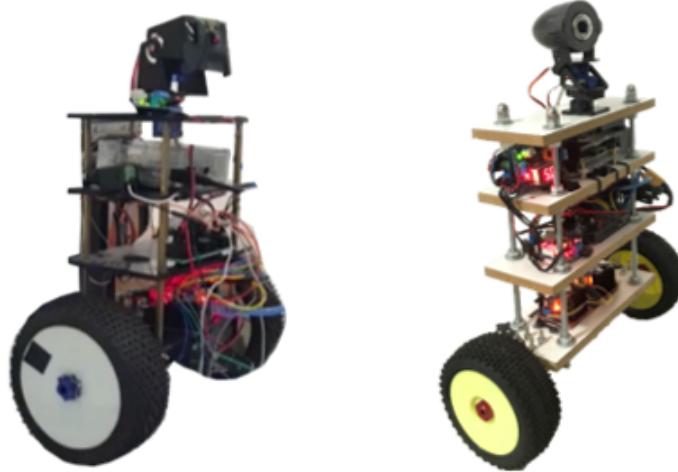


Fonte: (MEMARBASHI, 2010)

Na primeira parte da obra citada, o autor verifica que a bateria concentra a maior massa do protótipo e portanto, quando à realoca na parte superior do modelo, verifica uma significativa melhora no equilíbrio da estrutura, constatando a presença de um controle mais estável e robusto que o anterior. Todavia, quando as rodas foram alteradas com intuito semelhante, observou-se que não houve melhora significativa no controle.

Com base nas observações citadas e em outras obras desenvolvidas anteriormente, foram desenvolvidos dois protótipos, apresentados na Figura 14, para estudo da estrutura e validação de conceitos apresentados em disciplinas de anos anteriores.

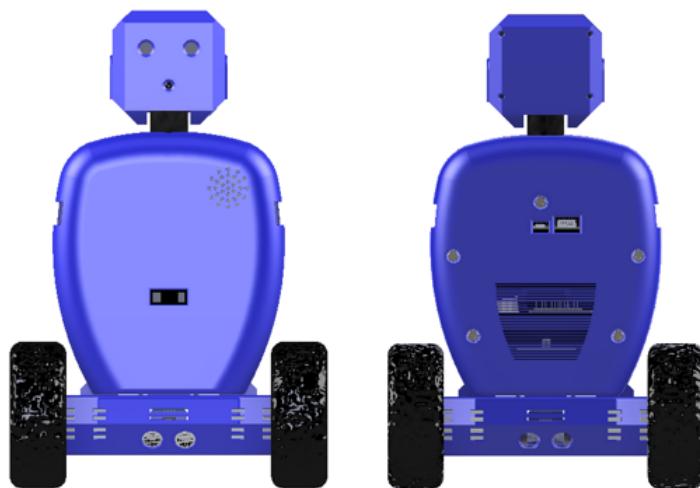
Figura 14 – Protótipos desenvolvidos pelo grupo em anos anteriores.



Embora os protótipos construídos anteriormente tenham alcançado seus objetivos, fez-se necessário o desenvolvimento de um novo projeto com uma aparência mais “amigável”, e com o *hardware* “protegido” para a própria segurança do usuário e do robô.

Após vários esboços buscando alinhar o projeto da estrutura em formato modular com um design externo semelhante ao de um robô humanoide, inicialmente “inspirado” no brinquedo MiP (Figura 6), porém com dimensões superiores para acomodar adequadamente os componentes eletrônicos, esboçou-se o design final do projeto indicado na Figura 15.

Figura 15 – Design final da estrutura do projeto.



Diferente da maioria das referências consultadas ou mesmo dos protótipos desenvolvidos pelo grupo em anos anteriores, a estrutura deste projeto não foi construída em formato de “prateleiras empilhadas” mas sim com um design e estrutura modular, ou seja,

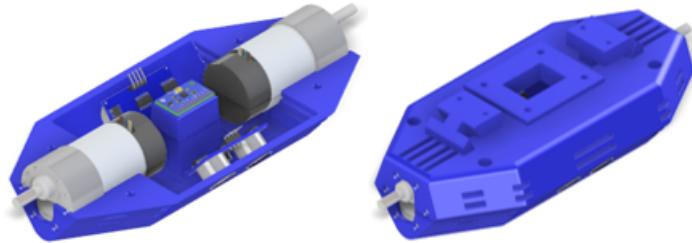
toda a estrutura do robô foi projetada em partes individuais, que são apenas encaixadas e fixadas por parafusos (lembrando o formato de "LEGO") de forma a permitir, além de uma montagem simples, do tipo "*Plug and Play*", uma melhor organização dos componentes.

Para melhor entendimento do planejamento e construção deste projeto, a estrutura foi dividida em partes. O conjunto das estruturas que serão detalhadas a seguir pode ser melhor observado no Apêndice A - Vista explodida do projeto.

3.1.2 Base

Composta por duas partes semelhantes mas não iguais, a Figura 16 representa as peças que compreendem a base, responsável por acoplar o par de motores com *encoders*, o sensor de posição angular e sensores ultrassônicos que, em trabalhos futuros, poderão ser responsáveis por “detectar degraus”, evitando a queda do robô em superfícies elevadas.

Figura 16 – Projeto da base do robô.



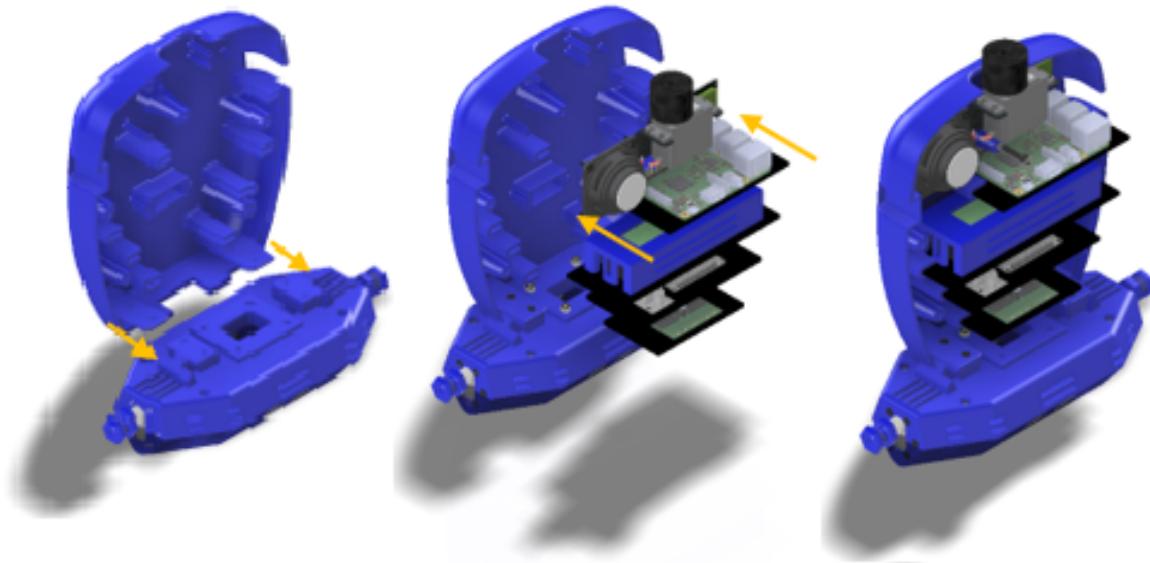
A fim de manter a máxima eficiência do modelo projetado, a placa que engloba o sensor responsável por informar a inclinação do modelo será fixada por meio de parafusos na mesma estrutura da base dos atuadores de forma axial, permitindo a medição exata da inclinação do robô.

3.1.3 Corpo

Assim como todo o projeto do robô, a geometria interna do corpo foi projetada com um "design modular" para que os componentes eletrônicos sejam estratégicamente fixados em placas de acrílico por meio de parafusos e então encaixadas (como "slots" de memória) a estrutura do robô. Este design foi ainda especialmente definido e aplicado a estrutura para facilitar a organização de cabos e favorecer possíveis manutenções dos componentes eletrônicos que possam haver durante ou após a construção do projeto.

Conforme indicado na Figura 17, seguindo a estratégia de estrutura modular, o par de peças que compõem a estrutura citada será fixada na parte superior da base por meio de um encaixe devidamente projetado para acolher tais partes e, em seguida, reforçadas com parafusos para completa fixação.

Figura 17 – Fixação da estrutura do corpo e dos componentes eletrônicos.

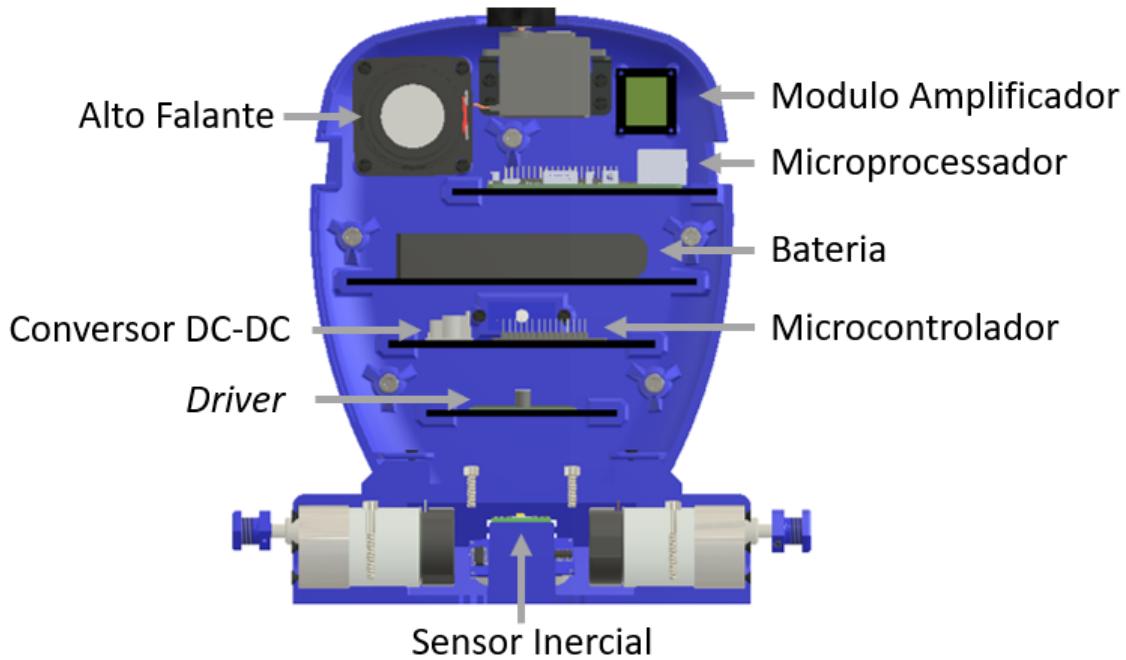


Visando a otimização da interligação dos componentes eletrônicos e a distribuição de forma organizada e homogênea dos elementos a fim de obter uma estrutura com carregamento equilibrado, optou-se por alocá-los de maneira “hierárquica” na estrutura modular.

O *driver*, que atuará exclusivamente nos motores, ficará disposto próximo a base; o microcontrolador que necessita realizar a comunicação entre o microprocessador e o dispositivo de potência, ficará entre estes, na segunda “camada”; já o microprocessador que realizará a comunicação com uma série de componentes localizados na parte superior, ficará mais próximo a esta, na última camada de baixo para cima.

Conforme observado anteriormente, a bateria concentra a maior parte da massa do modelo e portanto a escolha de seu local no modelo se torna imprescindível para localização exata do centro de massa do robô. Dessa forma, para proporcionar maior controle do modelo a fim de torná-lo mais estável, verificou-se que a bateria deve ser alocada na parte superior do modelo. No entanto, para priorizar a conexão dos dispositivos da cabeça, a fonte de energia será alocada abaixo deste componente, conforme demonstrado na Figura 18.

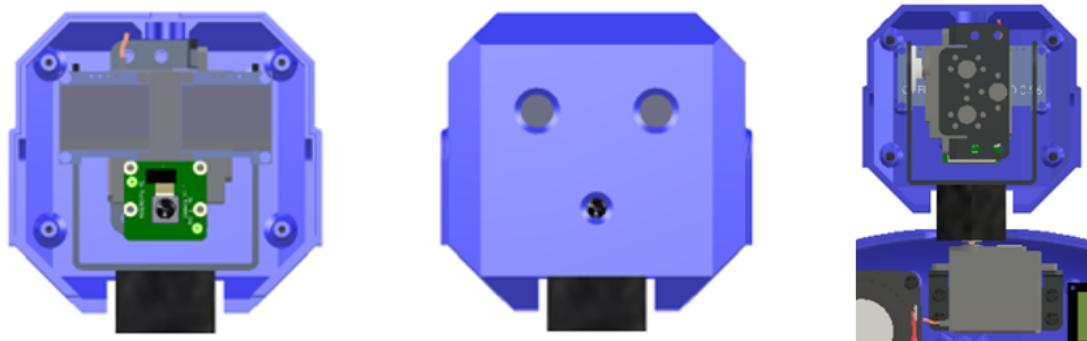
Figura 18 – Disposição dos componentes sobre a estrutura.



3.1.4 Cabeça

Com a ideia de lembrar as características de um típico robô humanoide, projetou-se uma estrutura que, dentre outras funções, acopla uma pequena câmera (Figura 19). Pensando em estender o campo de visão para aplicações de monitoramento de ambientes, utilizou-se de dois pequenos motores para realizar a movimentação da cabeça nos ângulos de arfagem e guinada. Tais motores são também responsáveis por promover a união entre esta estrutura e o corpo anteriormente projetado para receber a peça.

Figura 19 – Estrutura e componentes compõem o conjunto da cabeça.



Uma vez que a geometria e o design da estrutura foram definidos, faz-se necessário estudar o tipo de atuador que será responsável por equilibrar e movimentar o modelo.

3.1.5 Definição e dimensionamento dos atuadores

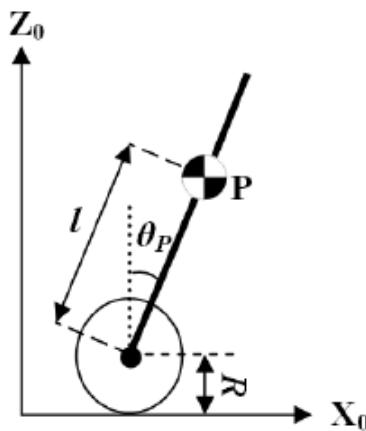
De acordo com as referências estudadas, é possível realizar a construção de robôs autoequilibrados utilizando motores do tipo "brushless" (oferecem maior qualidade de movimento e velocidade), motores "de passo" (possuem maior precisão), ou ainda, pela conveniência, motores "DC com escovas".

Por questões de praticidade e custo, como já havia disponível um par de motores DC com escovas (modelo 37Dx68L) utilizados para concepção do último protótipo realizado, optou-se por estudar a viabilidade de reutilização dos mesmos para atuação da nova estrutura projetada. Portanto, para validar tal hipótese, foi necessário realizar o dimensionamento destes através do cálculo de 2 parâmetros principais: torque e potência no eixo do motor.

Torque motor

Assumindo que o centro de massa (ilustrado na Figura 20) esta localizado no ponto P , isto é, se comporta como se toda a massa do corpo estivesse concentrada sobre ele a uma distância l do eixo das rodas, então a força inicial de atuação presente no modelo é o produto da massa m pela aceleração da gravidade g . O angulo θ_p representa o angulo de *pitch* (rotação em torno do eixo Y) e permite estimar o torque do momento estático necessário para atuar as rodas.

Figura 20 – Modelo dinâmico do pêndulo invertido sobre rodas.

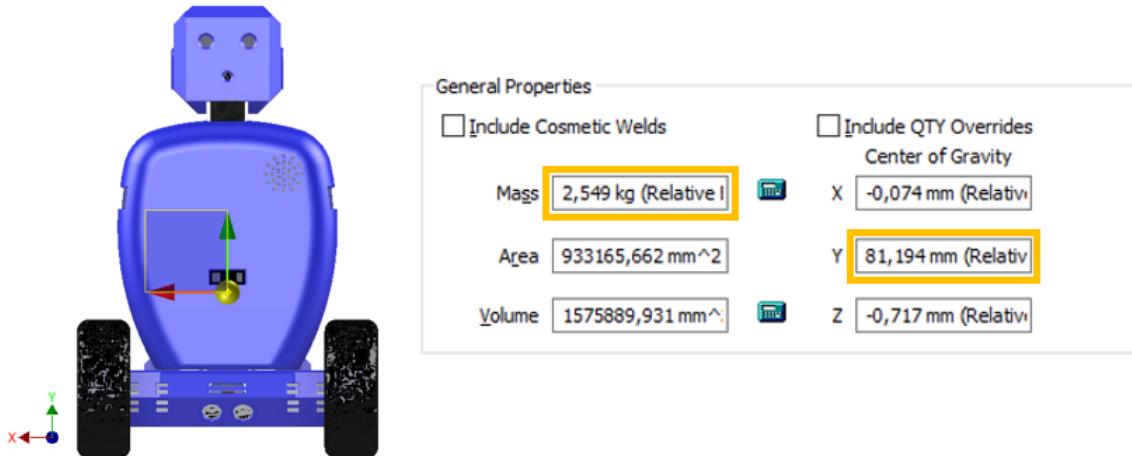


Fonte: (MEMARBASHI, 2010)

Por motivos de "segurança de projeto" e com o intuito de manter a linearidade do sistema, foi estabelecida uma inclinação máxima (representada pelo ângulo Θ_p) na Figura 20) de 10° . Para a aceleração da gravidade local g , adotou-se o valor de $9,81m/s^2$.

Através da Equação 3.1 e dos dados de estudos realizados no Autodesk Inventor, que podem ser observados na Figura 20, estimou-se o torque mínimo necessário (τ_s) para atuar o modelo.

Figura 21 – Localização do centro de gravidade l e massa m do modelo.



$$\tau_s = F.l.\sin(\Theta_p) \quad (3.1)$$

$$\tau_s = m.g.l.\sin(10^\circ) \quad (3.2)$$

$$\tau_s = 0,3525 N.m \quad (3.3)$$

Como a força para atuação do robô deriva de dois motores, o torque necessário deve ser dividido pela metade. Para tanto, utiliza-se da Equação 3.4 para cálculo do torque efetivo mínimo (τ_{min}) necessário para cada atuador.

$$\tau_{min} = \frac{\tau_s}{2} \quad (3.4)$$

$$\tau_{min} = 0,17625 N.m \quad (3.5)$$

Observando a Tabela 1, de especificações gerais do motor, é possível afirmar que o torque mínimo estimado para cada atuador é perfeitamente atendido. Todavia, é também necessário estimar a rapidez com que o trabalho do atuador será realizado através do cálculo da potência mecânica a fim de garantir uma resposta veloz suficiente para equilibrar o robô.

Tabela 1 – Especificações gerais do motor 37D x 68L Pololu

Modelo	37D x 68L
Peso	190g
Diâmetro do eixo	6mm
Redução	30:1
Velocidade sem carga	350RPM
Tensão	12V
Corrente sem carga	300mA
Corrente Máxima	5A
Torque Máximo	0,7768Nm

Potência motora

Para estimar a potência mínima (P_{min}) dada pela Equação 3.8, calculou-se a velocidade angular ω pela Equação 3.6, adotando-se (com base nas referências e testes realizados nos primeiros protótipos) uma velocidade linear máxima do robô de $0,5m/s$ e raio R das rodas conhecidos de 58,5mm.

$$\omega = \frac{v}{R} \quad (3.6)$$

$$\omega = \frac{0,5}{0,0585} = 8,547 rad/s \quad (3.7)$$

$$P_{min} = \frac{\tau_{min} \cdot 2\pi \cdot \omega}{60} \quad (3.8)$$

$$P_{min} = \frac{0,17625 \cdot 2\pi \cdot 8,547}{60} = 0,1577W \quad (3.9)$$

Através da Equação 3.8 citada anteriormente, e utilizando as informações da Tabela 1, calculou-se a potência do motor escolhido que demonstra, assim como o torque, ser super dimensionada e portanto suficiente para atender o projeto sem a necessidade de aquisição de um novo atuador.

$$P_{motor} = \frac{0,7768 \cdot 2\pi \cdot 350}{60} \quad (3.10)$$

$$P_{motor} = 28,4712kW \quad (3.11)$$

Projetada a estrutura e dimensionado os atuadores, é necessário a obtenção da estrutura física do projeto, para tal, estudou-se a possibilidade de usinar as peças ou injetar material em um molde com o formato da estrutura de cada peça. Devido ao alto

custo das sugestões anteriores, optou-se por realizar a impressão das estruturas através da impressora 3D disponível na instituição. Porém, para realização do processo, foi necessário escolher cuidadosamente o material a ser utilizado.

3.1.6 Definição do Material

O conjunto de atuadores vinculado a base do modelo, assim como outros componentes eletrônicos, podem ter suas temperaturas bastante elevadas quando em uso. Os motores, por exemplo, quando em operação máxima podem aquecer a uma temperatura de até 55°C. Por esse motivo, escolheu-se um material com a maior temperatura de transição vítreia disponível para impressões 3D, o plástico ABS (Acrilonitrila Butadieno Estireno). A Figura 22 mostra, dentre outras comparações, a força suportada, a temperatura de transição vítreia e a qualidade de impressão de cada tipo de material geralmente utilizado neste tipo de impressão.

Figura 22 – Comparaçao entre materiais para impressão 3D.

Printresult	ABS	PLA	EPR PET
Strength	++	-	+
Resolution	+	+	++
Appearance	+	+	++
Layer adhesion	+	+/-	++
Food compliant material	-	++	++
Watertight	+	-	++
Overall score Printresult	+	+/-	++
	NGEN	PLA	ABS
glass temperature	85C	50C	100C
toughness	++	-	+++
printing temperature	220C – 240C	190C – 220C	250C – 260C
printability	+++	+++	++
odor neutral (during print)	+++	+	-
Stability during print	+++	+	++
Surface clarity	+++	+	-
	PETG		

Fonte: <https://3dprinting-blog.com/tag/pet-filament/>

Para garantir o arrefecimento interno do robô, tendo em vista que a estrutura ou mesmo os componentes eletrônicos mais sensíveis podem ser danificados, fez-se necessário o projeto de "pequenos respiros" nas estruturas para ventilação natural (conforme movimentação do robô), principalmente no local onde estão localizados os atuadores (base) e os componentes que mais aquecem. Os detalhes citados podem ser visualizados na Figura 15, que representa o modelo final projetado construído no Autodesk Inventor.

Por último, optou-se por manter as rodas com pneus do tipo "off road" de 117mm pois as mesmas já estavam disponíveis de outro projeto, evitando o aumento de custo do mesmo. Além disso, como observado anteriormente em protótipos anteriores, os "cravos" presentes nos pneus não implicam em grandes distúrbios no controlador, embora a ausência dos mesmos possa trazer maior estabilidade para o robô.

A comparação do modelo virtual do projetado com o modelo físico obtido e as dimensões finais do projeto podem ser observadas, respectivamente, na Figura 23 e no Apêndice B.

Figura 23 – Comparação entre projeto virtual e real.

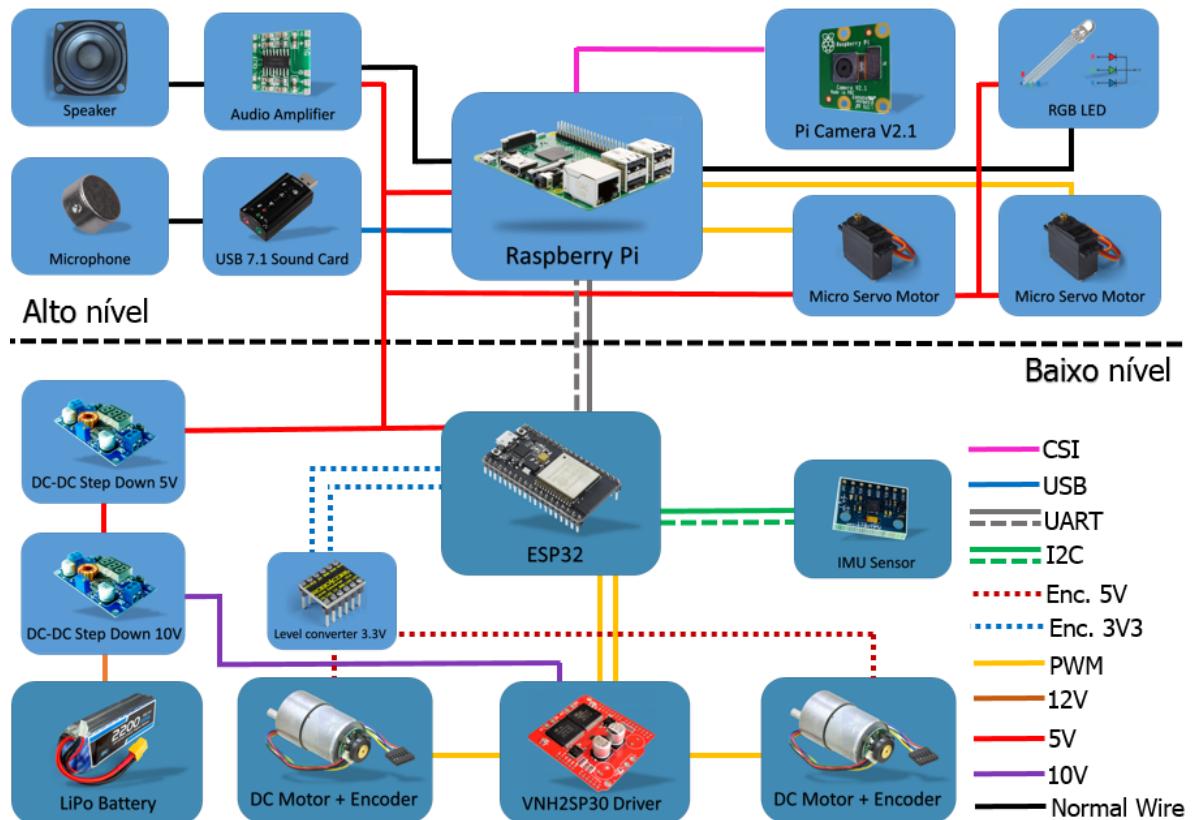


3.2 Projeto Eletrônico

O projeto é classificado entre dois principais grupos, microcontrolador e microprocessador. O primeiro é responsável por interagir com dispositivos de baixo nível, como leitura de sensores e atuação de motores e o segundo grupo dedicado ao desenvolvimento de aplicações em alto nível, como processamento de imagem.

Conforme indicado na revisão da literatura [7], são necessários alguns componentes básicos para autoequilibrar o robô, ilustrados na Figura 24, sem os quais não seria possível a adequação do restante do projeto, vez que estes são essências para definir as primeiras dimensões físicas do modelo.

Figura 24 – Diagrama de Componentes Eletrônicos.



3.2.1 Microcontrolador

Como o sistema de controle necessita de alta velocidade de realimentação em sua malha e uma qualidade robusta e fidedigna do tratamento de sinais para manter o robô equilibrado, utilizou-se de uma placa equipada ESP-WROOM-32, onde está integrado uma CPU Xtensa Dual Core 32 Bits LX6 com memória SRAM de 520 KByte, ROM de 4 MByte e clock ajustável de até 160MHz.

A placa ainda possui 2x DAC de 8-bits, 34 pinos de entrada/saída programáveis como PWM, UART, I2C, SPI e outras configurações, responsável por integrar sensores

e principalmente receber os comandos do microprocessador. O dispositivo é também responsável por realizar, de fato, o controle integral e em tempo real dos atuadores, através da malha projetada e de seus respectivos sinais de entrada e saída, fornecendo maior flexibilidade no desenvolvimento do projeto.

Existem outras opções de placas microcontroladas disponíveis no mercado, como Atmel Arduino Uno, STM32 Discovery, Microchip PIC32, porém a escolha da placa de desenvolvimento Espressif ESP32 se deve ao baixo custo, a quantidade de recursos disponíveis comparado com as outras opções, e ao poder de processamento e memória suficiente para integrar um sistema operacional de tempo real *RTOS*.

3.2.2 Microprocessador

Com a proposta de criar uma plataforma disponível para aplicações, utilizou-se uma placa Raspberry Pi 3 Model B que inclui um adaptador Wi-Fi e Bluetooth 4.1 integrados, facilitando a implementação e comunicação com dispositivos remotos como por exemplo, um controle bluetooth e celular através de uma rede sem fio.

Devido a grande difusão deste microprocessador na comunidade "*Maker*", é possível embarcar uma distribuição Linux no mesmo possibilitando, desta forma, aproveitar o vasto desenvolvimento e suporte da comunidade de software livre para aplicações. A distribuição utilizada é Raspbian, que utiliza a versão 4.14 do kernel do linux.

3.2.3 IMU

Para fornecimento dos sinais de entrada de posição e velocidade angular da estrutura do robô ao microcontrolador utiliza-se um sensor inercial IMU-9250, módulo multi-chip da InvenSense, que possuí facilidade na aquisição dos dados devido a comunicação I2C de 400KHz.

O giroscópio mede a velocidade angular do robô, através de variação do ângulo do eixo, permitindo ser configurado entre uma escala de $\pm 250^\circ/\text{segundo}$ à $\pm 2000^\circ/\text{segundo}$ e o acelerômetro mede o ângulo de inclinação, através da medida de aceleração dinâmica (vibração) e aceleração estática (gravidade) em uma escala de $\pm 2g$ até $\pm 16g$.

3.2.4 Encoder

Para a aquisição de posição e velocidade das rodas, utiliza-se um par de encoders de efeito Hall com dois canais, um canal defasado 90° do outro, e resolução de 1920 pulsos por volta acoplados aos eixos dos motores.

3.2.5 Câmera

Para capturar imagens e realizar aplicações de monitoramento de vídeo, utiliza-se a câmera PiCam, onde possui um sensor Sony IMX219 de 8-megapixel, suportando resoluções de vídeo de 1080p30, 720p60 e VGA. Por ser um módulo dedicado da Raspberry, a interface é realizada através da conexão de uma cabo flat com a porta CSI, *Camera Serial Interface*.

3.2.6 Bateria

Com base na corrente média apresentada por cada componente em uma superfície plana e lisa (indicada na Tabela 2), estimou-se que o robô consome aproximadamente 3100 mAh por hora. Portanto, uma bateria de Lítio-Polímero com capacidade de descarga de 2200 mA por hora e tensão de 12V para suprir a necessidade dos motores, é suficiente para atender as necessidades do projeto por, pelo menos, 40 minutos.

Tabela 2 – Consumo médio dos componentes eletrônicos em superfície plana e lisa

Microcontrolador	300 mAh
Microprocessador	1500 mAh
Motores e <i>Driver</i>	1000 mAh
Sensores e dispositivos	300 mAh
Corrente total estimada:	3100 mAh

3.2.7 Conversor DC-DC

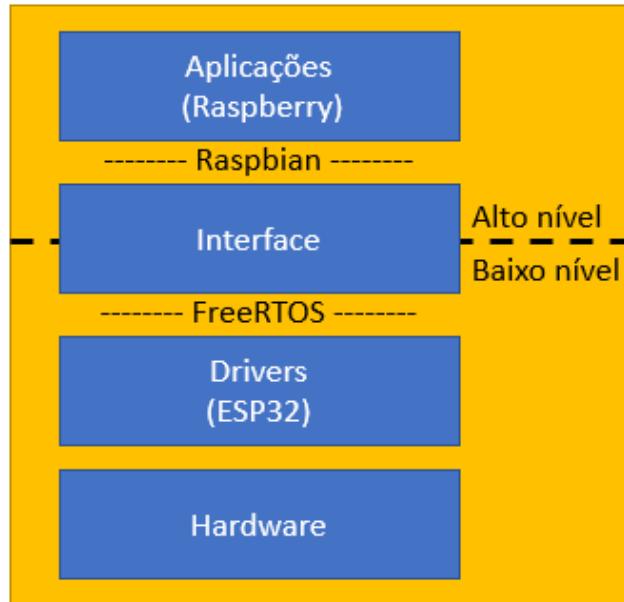
Dois conversores DC-DC reguláveis, foram utilizados. O primeiro foi utilizado para alimentar diretamente os motores DC através dos *drivers*, garantindo ganhos fieis e constantes fornecidos pelo controlador. O segundo conversor, foi utilizado para reduzir a tensão de 12v para 5v necessários para alimentar o microprocessador, servo motores e outros periféricos que poderão ser adicionados futuramente.

3.3 Projeto de Software

A arquitetura de software implementada utiliza-se de um conceito modular, através de programação orientado a objeto, afim de facilitar e abstrair as camadas de software para modificações e/ou manutenção futura.

O desenvolvimento é dividido em dois principais grupos representados na Figura 25 por alto e baixo nível, onde aplicações são implementadas na camada mais alta através de *scripts* em Python e interação com hardware na camada de baixo.

Figura 25 – Camadas do sistema.

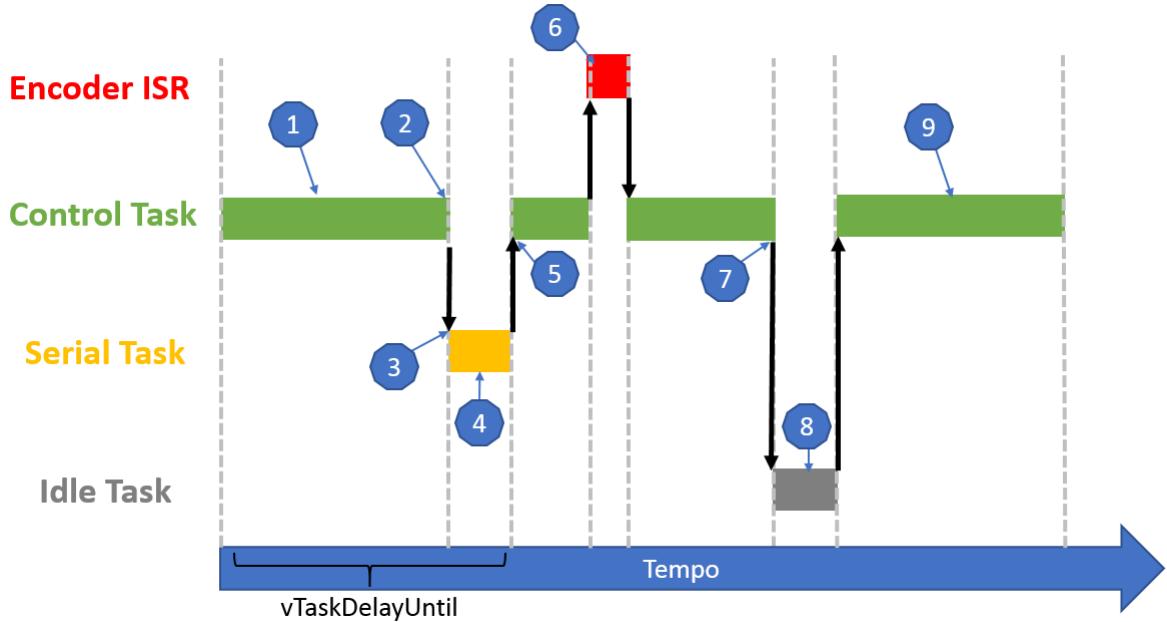


Para o microcontrolador, necessita-se de determinismo na execução da rotina de controle, assim se utiliza um sistema operacional de tempo real, FreeRTOS, onde sua API, *Application Programming Interface*, para implementação está descrita em [11], com destaque para as rotinas:

- `xTaskCreate`: Cria nova tarefa com prioridade determinada;
- `vTaskDelayUntil`: Atrasa uma tarefa até um período especificado;
- `xQueueCreate`: Cria uma fila;
- `xQueueSend`: Envia uma mensagem para a fila;
- `xQueueReceive`: Recebe uma mensagem da fila.

Desta forma, duas *Tasks* são instanciadas, uma rotina do controle com a maior prioridade de execução e tempo de ciclo pré-determinado por `vTaskDelayUntil` e outra com menor prioridade apenas para comunicação de eventos com dispositivos externos via serial, além de um *Queue* para troca de informação entre as *Tasks*. O tempo de execução médio e a divisão de cada *Tasks* pode ser observado na Figura 26.

Figura 26 – Fluxo das tarefas.



Referindo-se aos números no diagrama acima:

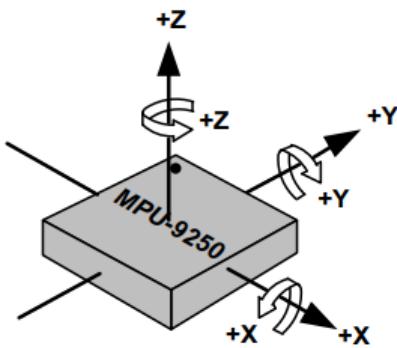
- Em (1), *Control Task* é executada;
- Em (2), o kernel suspende *Control Task*, pois foi executada e está bloqueada até o tempo definido em vTaskDelayUntil expirar;
- Em (3), *Serial Task* assume a execução;
- Em (4), *Serial Task* processa dados carregados na serial;
- Em (5), o kernel suspende *Serial Task*, pois o tempo definido em vTaskDelayUntil expirou, assim *Control Task* assume por possuir maior prioridade;
- Em (6), rotina de interrupção é executada;
- Em (7), o kernel suspende *Control Task*, pois foi executada e como não existe dados carregados na serial, move para estado de espera até vTaskDelayUntil expirar novamente;
- Em (8), *Idle Task* é criada automaticamente pelo RTOS para garantir que sempre exista uma tarefa executando;
- Em (9), reassume *Control Task*.

Pode-se categorizar o desenvolvimento dos módulos através dos seguintes tópicos:

3.3.1 Leitura do Sensor Inercia

A aquisição de dados de inclinação, *Pitch*, é realizada por comunicação *I₂C* onde o microcontrolador é o “Mestre” e o módulo IMU-MPU9250, o “Escravo”. Como mencionado anteriormente, o módulo IMU possui acelerômetros e giroscópios integrados, um para cada eixo (X, Y e Z), conforme indicado na Figura 27.

Figura 27 – Orientação do IMU.



Fonte: (MPU9250 - Product Specification, 2016)

É necessário configurar o sensor conforme mencionado no documento de especificação do produto [12], ler individualmente seis registros de 8-bits e converter para três registros de 16-bits (X, Y, Z), vide mapa de registros [13].

Utilizando trigonometria é possível calcular, por meio da Equação 3.12, o ângulo do vetor resultante do acelerômetro em relação ao solo.

$$\text{AccelerometerAngle} = \arctan\left(\frac{-\text{accVector}X}{\sqrt{(\text{accVector}Y^2 + \text{accVector}Z^2)}}\right) \quad (3.12)$$

Para o ângulo medido do giroscópio, obtido pela equação 3.13, aplica-se um fator de sensibilidade, por exemplo de 2000 graus por segundo.

$$\text{GyroAngle} = \text{gyroVector}X * \text{Sensitivity} \quad (3.13)$$

3.3.2 Filtro Complementar

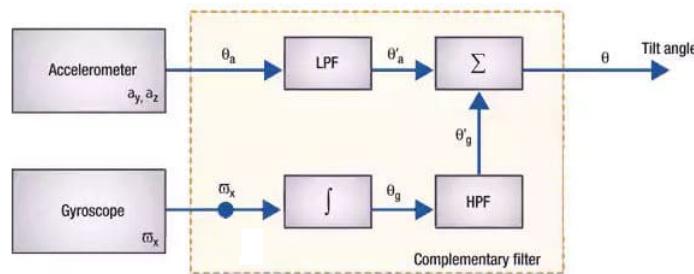
Após obter a leitura individual de cada sensor do IMU, é necessário realizar a fusão dos dados para obter um valor mais confiável e sem ruídos. Assim, foram analisados alguns métodos para tal, como filtro de Kalman e filtro complementar (Figura 28). Definiu-se a utilização do filtro complementar pela facilidade de aplicação, onde existe um fator de

filtro passa-alto e passa-baixo para cada termo, realizando o somatório em cada iteração, conforme Equação (3.14).

$$\theta(t) = \alpha(\theta(t - 1) + gyroAngle * sampleTime) + (1 - \alpha)accelerometerAngle \quad (3.14)$$

- θ - Ângulo *Pitch* do pêndulo
- α - Fator do filtro complementar (Valor utilizado = 0.98)

Figura 28 – Filtro complementar.



Fonte: (Grasser, 2002)

3.3.3 Aquisição do Encoder

A leitura do encoder é realizada através da contagem de interrupções geradas a cada borda de subida nos pinos de interrupção externa do microcontrolador. Assim uma entrada digital, *ENCODERA1 PIN*, está conectada ao canal A e outra entrada digital, *ENCODERB1 PIN*, ao canal B.

Destaca-se abaixo, um modelo de implementação para a rotina de interrupção. Nas linhas 2 e 3, são definidos os pinos de entradas digitais como *Pull-Up*, afim de evitar contagem de pulsos indevidos. A função *attachInterrupt* é utilizada para registrar uma função a ser executada quando ocorrer uma transição de borda de subida na entrada *ENCODERA1 PIN*.

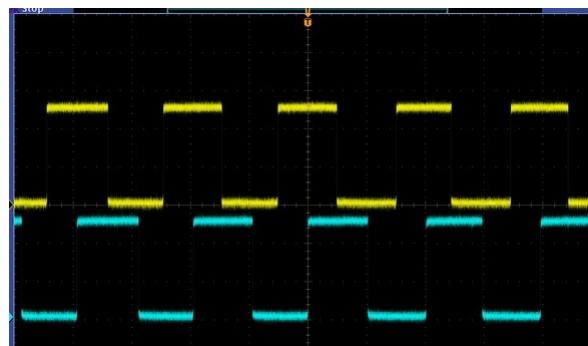
Seguindo boas práticas de implementação, rotinas de interrupção devem ser curtas e realizar suas tarefas rapidamente, nota-se que *encoderISR1*, apenas atualiza o atributo *ticks*, nas linhas 11 e 13.

```

1 /* Interrupt Init */
2 pinMode(ENCODERA1_PIN, INPUT_PULLUP);
3 pinMode(ENCODERB1_PIN, INPUT_PULLUP);
4 attachInterrupt(digitalPinToInterrupt(ENCODERA1_PIN), encoderISR1, RISING);
5
6 /* Interrupt Service Routine */
7 void IRAM_ATTR encoderISR1()
8 {
9     portENTER_CRITICAL_ISR(&mux1);
10    if (digitalRead(ENCODERB1_PIN))
11    { encoder1.ticks++; }
12    else
13    { encoder1.ticks--; }
14    portEXIT_CRITICAL_ISR(&mux1);
15 }
```

Analizando a defasagem de 90° de cada canal do encoder, pode-se verificar o sentido de rotação do motor, assim incrementando o atributo *ticks* caso o canal B estiver em nível alto e decrementando se o nível estiver baixo. Considerando que uma volta completa do eixo equivale a 1920 *ticks*.

Figura 29 – Canal A e B do encoder acoplado ao eixo de um motor.



3.3.4 Processamento da Malha de Controle

Por ser uma rotina crítica, o processamento da malha de controle é realizado através de uma tarefa com maior prioridade no sistema, *Control Task*. Esta tarefa, utiliza controladores PID, realizando o processamento com termos proporcional, integral e derivativo, conforme demonstrado na Equação 3.15:

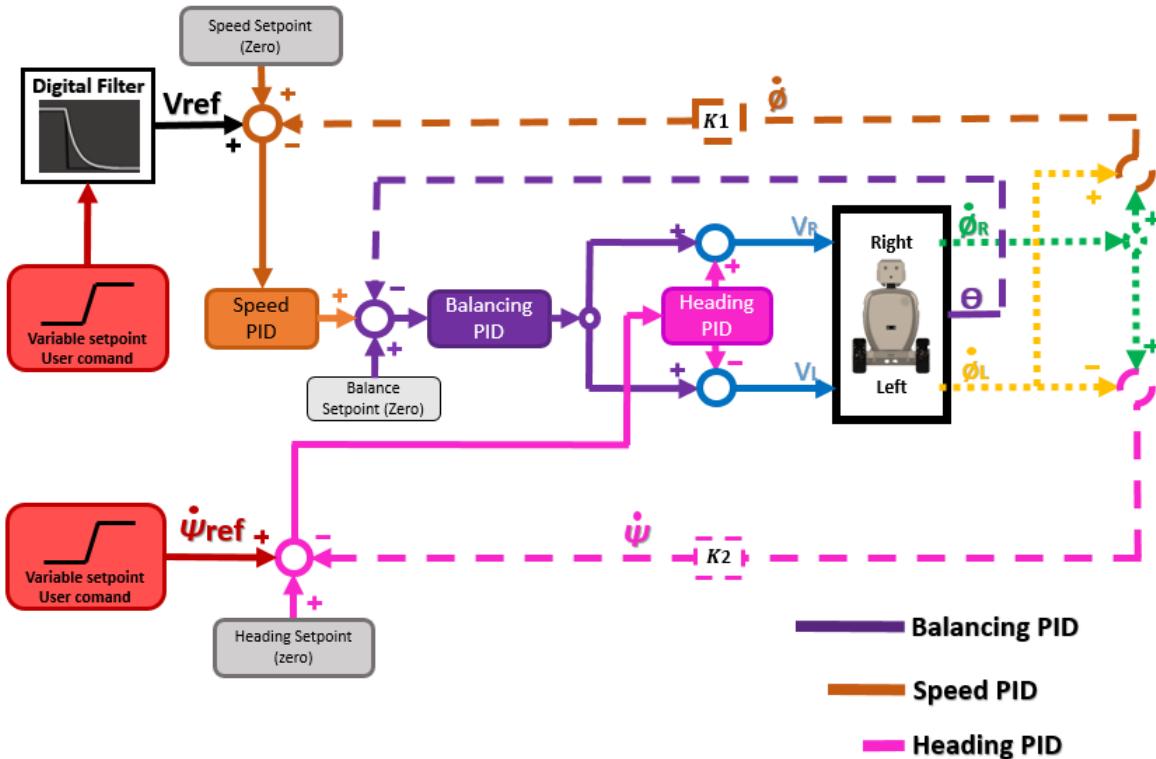
$$y(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{di(t)}{dt} \quad (3.15)$$

- K_p - Constante do termo proporcional;
- K_i - Constante do termo integral;

- K_d - Constante do termo derivativo;
- e - Erro entre setpoint e entrada;
- de - Diferença do erro atual com o último;
- di - Derivada da entrada atual com a última.

São instanciados três objetos, *BalancingPID*, *SpeedPID* e *HeadingPID*. Assim, através na Figura 30, é possível estabelecer a estratégia de controle por uma malha de PID em cascata [14].

Figura 30 – Malha de controle.



O *BalancingPID* tem a função de manter o robô equilibrado calculando o erro entre o *setpoint* e a realimentação do sensor IMU. O controle de velocidade utiliza a derivada da posição medida pelo *encoder* do motor como entrada e a saída é utilizada como referência, *setpoint*, do *BalancingPID*. Para os movimentos de rotação, é utilizado o *HeadingPID*, onde é aplicado diretamente na entrada da planta. Salientando que as constantes K_1 e K_2 calculadas pelas Equações 3.16 e 3.17, respectivamente.

$$K_1 = \text{Constante de velocidade linear}$$

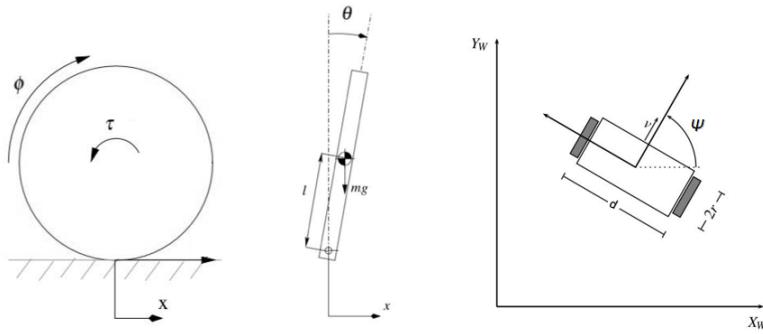
$$K_1 = \frac{0.5}{r} \quad (3.16)$$

K_2 = Constante de velocidade angular

$$K_2 = 0.5 * \frac{2r}{d} * \frac{360}{TicksperTurn} \quad (3.17)$$

Onde os ângulos apresentados na malha, θ - Pitch, ϕ - Velocidade, Ψ - Yaw, fazem referência a posição apresentada pelo robô em cada instante, conforme indicado na Figura 31.

Figura 31 – Ângulos de Referência da Malha de Controle.



Fonte: (Velazquez,2016) - Adaptado.

Para ilustrar a implementação dos objetos PID, nota-se no diagrama UML da Figura 32, que a classe possui um método principal para computar os dados e métodos adicionais para definir e obter as constantes de cada controlador:

Figura 32 – UML da classe PID.

PID	
- lastError: float	
- lastInput: float	
- lastTime: unsigned long	
- setpoint: float	
- Cp: float	
- Ci: float	
- Cd: float	
- Kp: float	
- Ki: float	
- Kd: float	
- windup: float	
+ compute(float): float	
+ setSetpoint(float): void	
+ getSetpoint(void): float	
+ setTunings(float): void	
+ getTunings(float): void	

Algoritmo implementado para o método *compute* da classe PID em linguagem C++, onde o parâmetro *input*, é a variável controlada, utilizada como entrada da malha:

```
1 float PID::compute(float input)
2 {
3     unsigned long now = millis();
4     float dt, error, de, di, output, outputSat;
5
6     /* Calculate delta time in seconds */
7     dt = (float)(now - lastTime) / 1000.0f;
8
9     /* Calculate error and delta error */
10    error = setpoint - input;
11    de = error - lastError;
12    di = input - lastInput;
13
14    /* Proportional Term */
15    Cp = error*Kp;
16
17    /* Integral Term */
18    Ci += error*Ki*dt;
19
20    /* Derivative term */
21    Cd = 0;
22    if(dt>0){
23        Cd = (di*Kd)/dt;
24    }
25
26    /* Sum terms: pTerm+iTerm-dTerm */
27    output = Cp + Ci - Cd;
28
29    /* Saturation - in order to Integral term do not reach very large
       values */
30    if(output > windup){
31        outputSat = windup;
32    } else if (output < -windup){
33        outputSat = -windup;
34    } else{
35        outputSat = output;
36    }
37
38    /* Save for the next iteration */
39    lastError = error;
40    lastInput = input;
41    lastTime = now;
42
43    return outputSat;
44 }
```

3.3.5 Atuação dos Motores

Os motores são acionados através de saídas PWM que estão conectadas com o *driver*, onde a porcentagem da razão cíclica, *duty cycle*, é definida pela diferença percentual da saída do controlador em relação ao setpoint, onde o sentido de rotação é definido pelo valor da saída dado pelas equações a seguir:

- Anti-horário:

$$-100\% \leq x \leq -1\% \quad (3.18)$$

- Parado:

$$x = 0\% \quad (3.19)$$

- Horário:

$$1\% \leq x \leq 100\% \quad (3.20)$$

3.3.6 Interface com Raspberry

A interface com o Raspberry é realizada através de uma tarefa específica com menor prioridade do que a tarefa de controle, *Serial Task*. Sendo somente responsável por receber, analisar e notificar de acordo com o evento recebido pela serial.

Exemplo de eventos:

- Obter eventos para acelerar e rotacionar (exemplo: via botão analógico do controle de Playstation);
- Desligar malha de controle (exemplo: botão *Reset*, vide Apêndice E);
- Definir *setpoint* dos controladores;
- Definir constantes dos controladores;
- Definir máxima inclinação;
- Obter informações em tempo real de inclinação, velocidade e esforço de controle.

3.3.7 Sintonização dos controladores

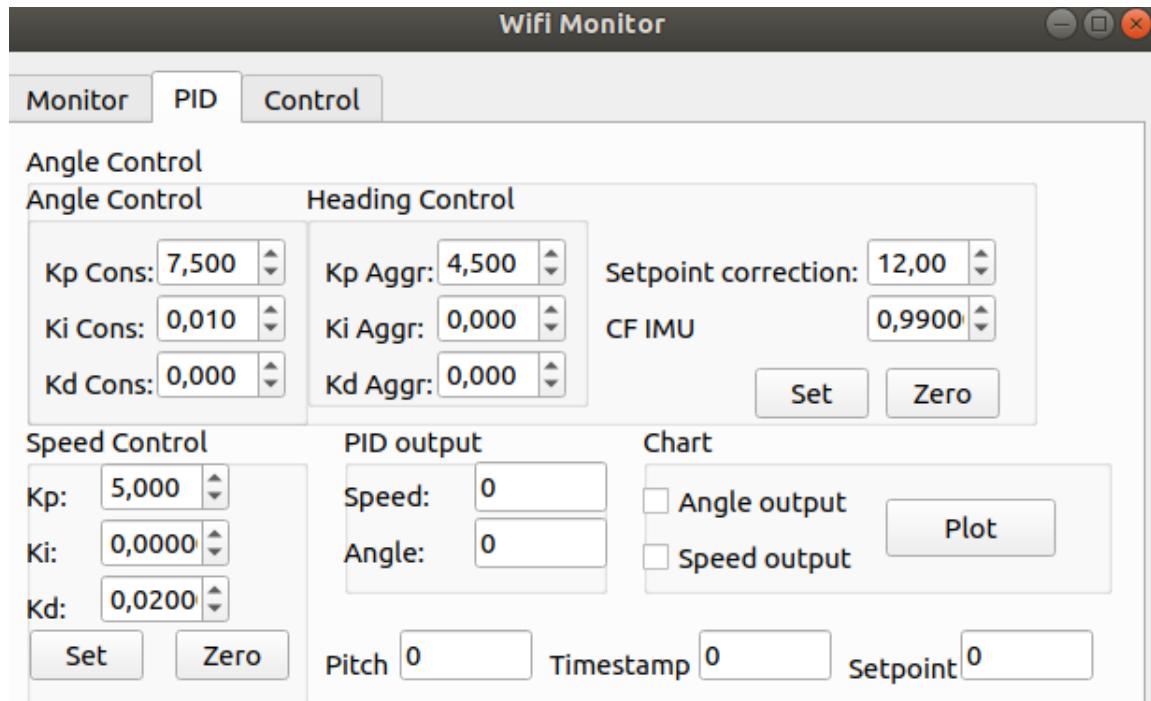
A sintonização dos controladores é realizada individualmente de forma empírica. Com uma interface gráfica desenvolvida em QT, envia-se os valores das constantes através de conexão UDP/IP para o Raspberry que redireciona para o ESP32.

Primeiramente, as constantes K_p , K_i e K_d do *BalancingPID* são ajustadas para zero. Em seguida, aumenta-se o ganho do termo proporcional até que o sinal de saída faça

o robô oscilar sem perder a estabilidade. Posteriormente, mantém-se os valores encontrados com o *BalancingPID* e inicia-se o ajuste do *SpeedPID*. Por fim, defini-se as constantes do *HeadingPID*.

A Figura 33, ilustra a interface gráfica desenvolvida para sintonizar o robô em tempo real.

Figura 33 – Wifi Monitor.



3.3.8 Aplicação de Monitoramento

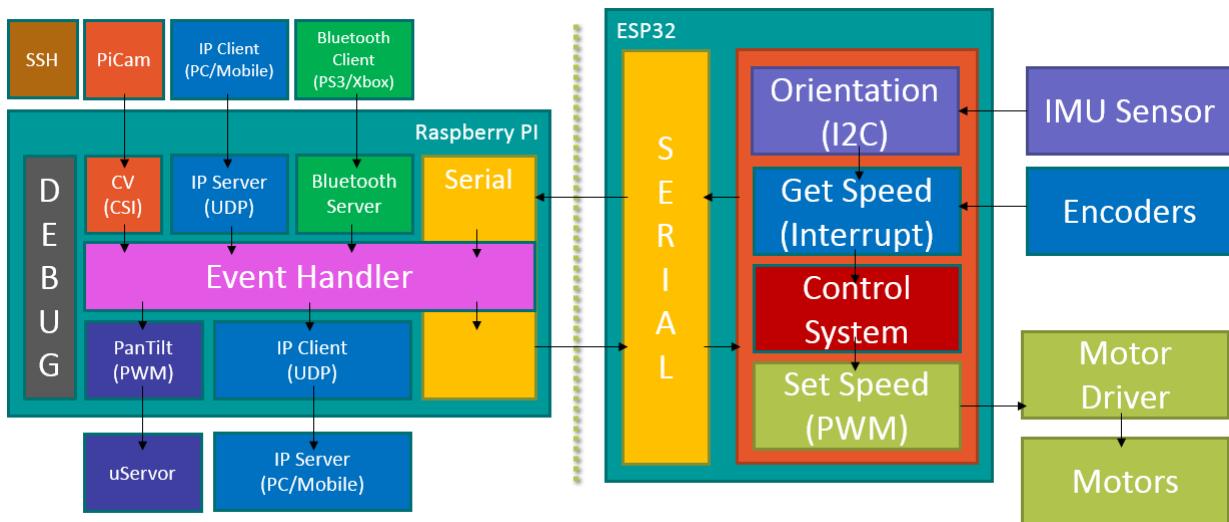
Com o Raspberry rodando a distribuição Raspbian, é possível desenvolver de forma prática aplicações utilizando *scripts* em Python. Uma aplicação de monitoramento é facilmente integrada através da biblioteca *PiCamera*.

Com a câmera conectada na porta CSI do Raspberry, basta instanciar o dispositivo e iniciar o visualização através da saída de vídeo analógica. Um transmissor de rádio FPV pode ser utilizado para realizar o *streaming* de vídeo para um monitor remoto.

3.3.9 Fluxograma e Diagramas

No diagrama da Figura 34, evidênciase a interação entre cada módulo do microcontrolador com o microprocessador.

Figura 34 – Diagrama do sistema.



Destaca-se os módulos na camada de alto nível:

- *Event Handler*: Gerenciar os eventos da camada de alto nível;
- *Bluetooth Server*: Receber comandos de dispositivos Bluetooth, como controle de Playstation;
- *Client/Server UDP/IP*: Enviar e receber comandos de dispositivos IP através do protocolo UDP;
- *Computer Vision (CV)*: Realizar o processamento de imagem, como detectar cor de objetos;
- *PanTilt*: Atuar micro servos para aumentar o campo de visão da cabeça do robô;
- *Serial*: Interfacear a comunicação com o microcontrolador.

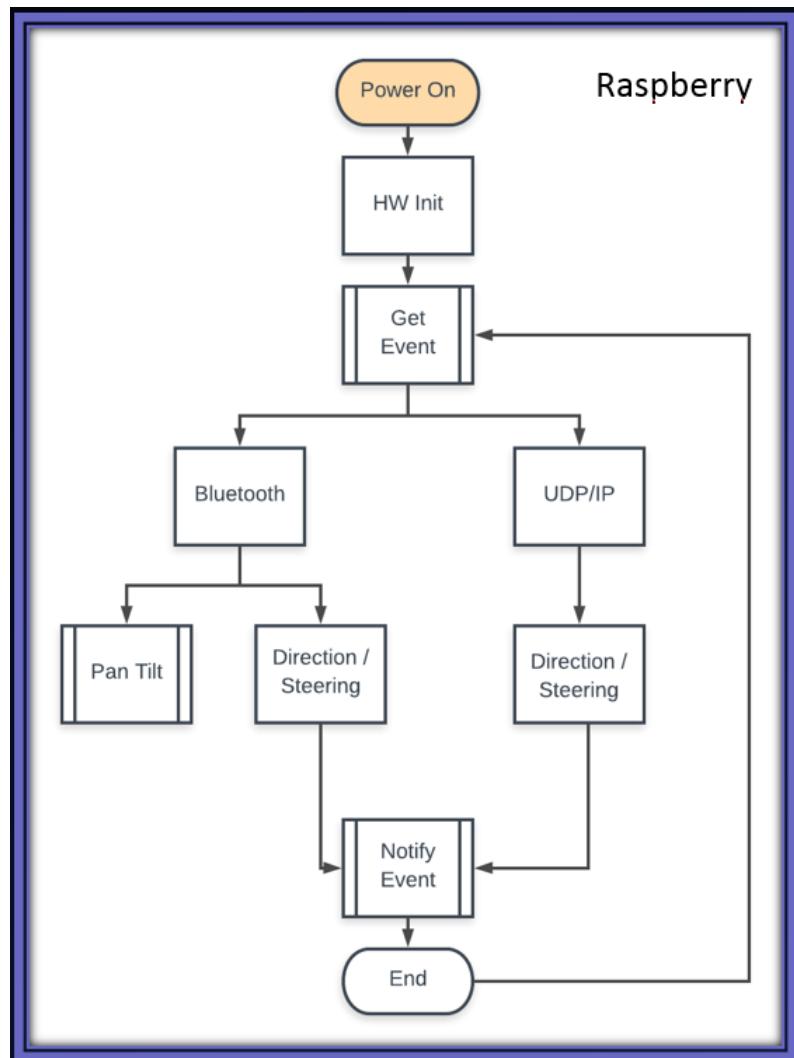
Destaca-se os módulos na camada de baixo nível:

- *Serial*: Comunicar com o microprocessador.
(Optou-se pela comunicação serial para essa finalidade, pois o microcontrolador é energizado pela própria saída USB do microprocessador);
- *Orientation*: Calcular a inclinação através da leitura do sensor IMU;
- *Get Speed*: Obter a posição e velocidade através da leitura do encoder;
- *Control System*: Processar a malha de controle;
- *Set Speed*: Atuar o *driver* dos motores.

No fluxograma do microprocessador (Figura 35), nota-se que é implementado uma hierarquia orientada a eventos, onde qualquer evento externo é notificado ao módulo principal *EventHandler* e esse redireciona para o respectivo módulo, por exemplo, o controle remoto do *Playstation* envia o evento de acelerar o robô para o Raspberry, o *EventHandler* detecta que o evento ocorreu via Bluetooth, realiza o tratamento e redireciona para o microcontrolador via serial, e assim sucessivamente.

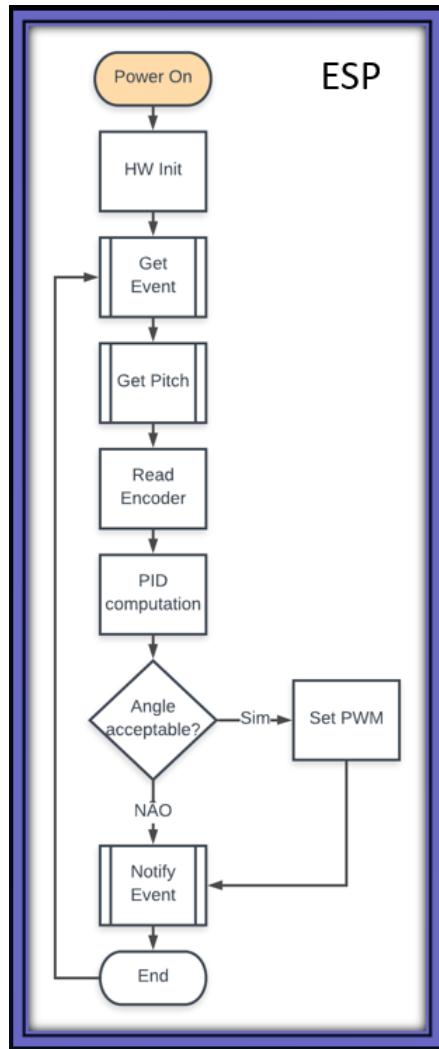
É importante salientar, que a comunicação entre os módulos é realizada através de *Message Queues*, pois os módulos são executados pela suas respectivas *tasks*, assim garantindo o aspecto de *Thread-Safe* a aplicação.

Figura 35 – Fluxograma do microprocessador.



Pode-se verificar no fluxograma do microcontrolador (Figura 36), a tarefa de controle, sendo responsável por verificar a possível ocorrência de eventos para mover o robô em para uma direção, obter a inclinação através da fusão do acelerômetro e giroscópio, calcular a posição e velocidade atual, processar a malha de controle e atuar os motores através do *driver*.

Figura 36 – Fluxograma do microcontrolador.



O código completo e documentação de como utilizar cada módulo está descrito no repositório do GitHub: <https://github.com/gchinellato/Toy-Robot>

4 RESULTADOS

Integrados os periféricos de hardware, definida a arquitetura do software e elencada as prioridades de suas aplicações, foram realizados ensaios onde o funcionamento do iStalker foi desafiado em terrenos irregulares, fazendo-se necessário o ajuste manual do novo ângulo de referência da malha de controle para cada ensaio. Outro problema encontrado foi a interferência de sinais de redes sem fio fazendo com que as aplicações de comunicação remota fossem comprometidas pela perda de pacotes de dados, obrigando a procura de ambientes com menor número de redes. Vencidos esses desafios, o sistema de controle embarcado, e a comunicação com a aplicação de mudança das constantes do robô obtiveram resultados satisfatórios e os objetivos principais e específicos foram alcançados, como será aqui demonstrado.

Ademais dois suportes temporários foram instalados na base com a função de manter o robô inclinado, fora do ângulo de operação, quando o mesmo estivesse desligado ou fossem desativados os atuadores e sistema de controle já que foi demostrado pela análise estrutural de elementos finitos que o material utilizado para estrutura do protótipo do iStalker era de fato frágil, e qualquer erro de operação poderia danificá-lo.

4.1 Constantes dos controladores

Como mencionado anteriormente, as constantes dos controladores foram ajustadas de forma empírica, sendo os valores que apresentaram os melhores resultados e utilizados para os ensaios, constam na Tabela 3.

Tabela 3 – Constantes dos controladores.

Controlador	Kp	Ki	Kd
Speed PID	5.50	0.00	0.20
Balancing PID	7.50	0.01	0.00
Heading PID	4.50	0.00	0.00

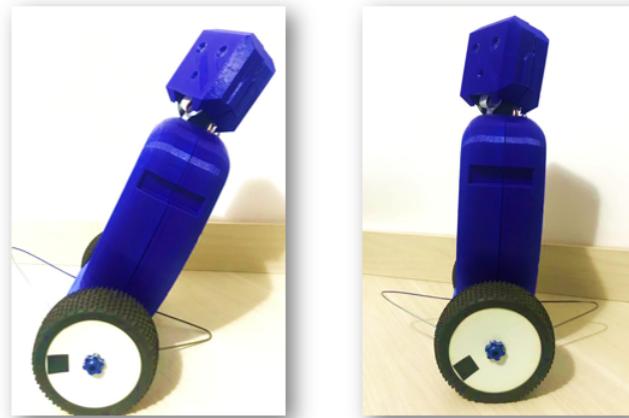
4.2 Ensaios

Os ensaios foram realizados em um ambiente fechado com superfície plana e bateria totalmente carregada. Antes de obter os dados que serão apresentados, foram repetidos diversas vezes, pelo menos três, com o propósito de constatar a duração da bateria, como fora dimensionada e a fim de obter resultados confiáveis.

4.2.1 Ensaio de Início de Operação

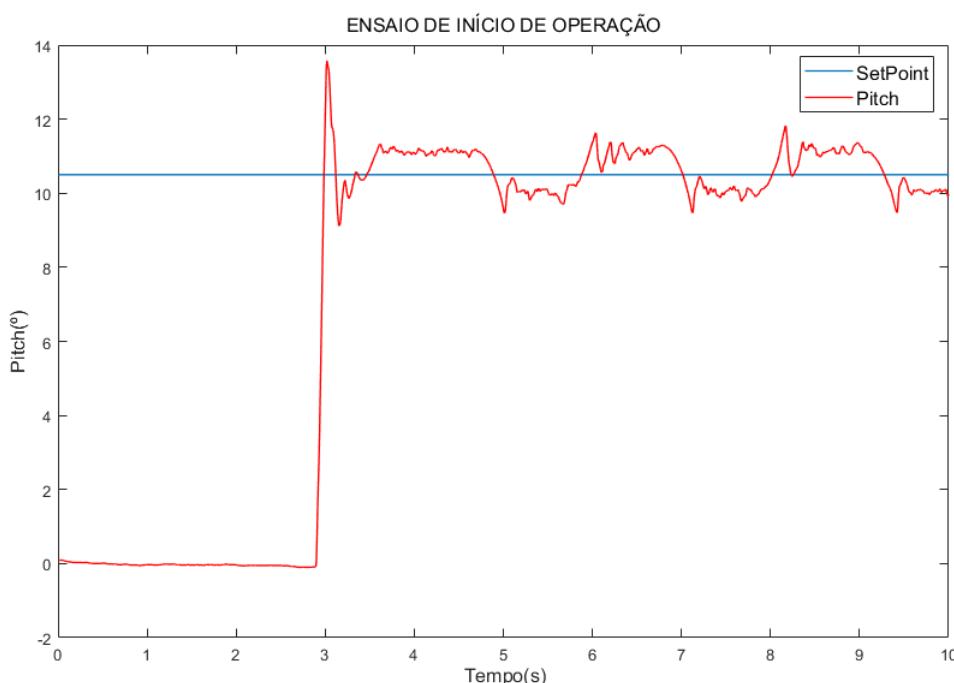
Constata-se que o iStalker iniciando de seu modo seguro apoiado em seus suportes temporários, se autoequilibra em menos 0,5 segundo (Figura 37).

Figura 37 – iStalker no início de operação.



Uma vez em equilíbrio o gráfico ilustrado pela Figura 38, demonstra que em operação sem nenhuma perturbação externa, a variação do ângulo de inclinação, do Inglês, *pitch*, não supera 2 graus em torno do seu ângulo de referência, *setpoint*. No mesmo gráfico supracitado, é importante ressaltar que o *pitch* não é medido até que se inicie a operação em 3 segundos.

Figura 38 – Ensaio de início de operação.



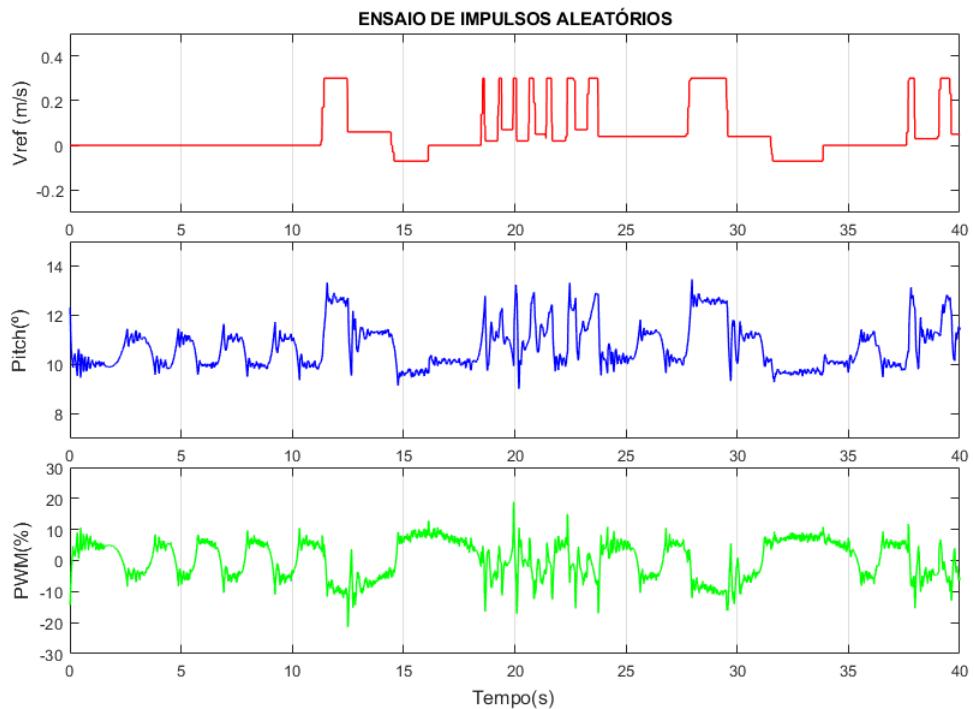
4.2.2 Ensaio de Impulsos Aleatórios

Julga-se o ensaio de início de operação como evidência que o sistema de controle embarcado no iStalker segue as especificações aqui pré-estabelecidas, não obstante, o primeiro ensaio não sofria de nenhuma perturbação externa, sendo assim, realiza-se um ensaio com entradas aleatórias, sendo algumas agressivas, como leves inclinações manuais no robô, e novamente se nota o comportamento totalmente satisfatório, como mostram os gráficos contidos na Figura 39.

O comportamento do controlador é facilmente notado no instante próximo a 12 segundos, quando se aplica uma entrada de velocidade (V_{ref}) aproximadamente 0,3m/s, fazendo o robo atingir quase 14 graus, 3,5 graus a mais que seu *setpoint*, neste mesmo instante pode-se notar que o PWM, saída do controlador, muda de sinal equilibrando o robô, a mudança de sinal do PWM significa a inversão de rotação das rodas, inclinando-o para o lado oposto.

Há outras amostras interessantes de serem apresentadas, como o intervalo de tempo 25 segundos a aproximadamente 27 segundos, onde não há variação na entrada, porém, nota-se mudança no *pitch* e a resposta de forma contrária no gráfico do PWM, isso mostra uma perturbação causada manualmente o que não impediu o funcionamento do controlador mantendo o equilíbrio.

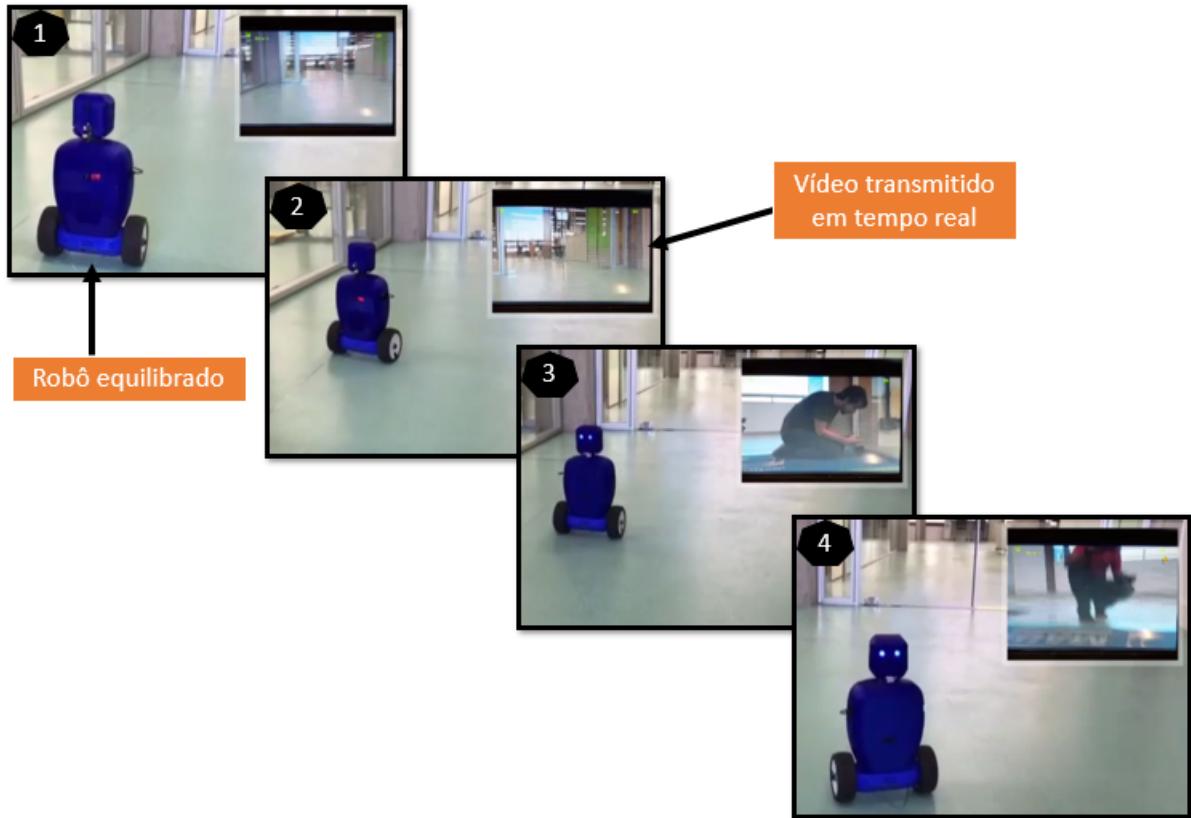
Figura 39 – Ensaio de impulsos aleatórios.



4.3 Demonstração prática das aplicações

Nota-se na sequência de imagens (Figura 40), o comportamento das aplicações em execução, onde o robô se autoequilibra enquanto é movimentado por um controle Bluetooth de Playstation, também verifica-se que a transmissão de vídeo é realizada em tempo real sem perda de imagem e sem sofrer interferências.

Figura 40 – Demonstração prática das aplicações.



4.4 Discussão dos Resultados

Os ensaios revelam que para as especificações pré-determinadas o sistema de controle embarcado apresenta resultados positivos, apesar das constantes dos controladores PID em cascata terem sido estimadas empiricamente. O cuidado com o tempo de amostragem determinístico, a aquisição e tratamento dos dados dos sensores através do filtro complementar e a praticidade de possuir um sistema operacional multitarefas integrado ao iStalker faz com que suas constantes e seus dados possam ser acessados remotamente, foram determinantes para os resultados positivos apresentados.

5 Conclusões e Trabalhos Futuros

Desenvolvidos os projetos e realizados os ensaios, deduz-se que todos os objetivos foram alcançados, todavia alguns pontos podem ser destacados e outros poderiam ser realizados com uma outra abordagem, afim de obter mais dados para análise.

Quanto a estrutura do robô, apesar de ser um protótipo, o material selecionado, alinhado a qualidade da impressão 3D, não atende perfeitamente o projeto em escala industrial, devido ao alto custo e fragilidade. Uma primeira solução seria a injeção de um plástico mais resistente para fabricação de toda a estrutura.

O desenvolvimento do sistema embarcado integrando o *FreeRTOS* (Sistema operacional de tempo real) apresentou-se essencial para o atendimento das expectativas do projeto, tanto por tornar possível tempos de amostragem determinísticos, quando ao se tratar dos sistemas de controle, até a integração de outras aplicações, como transmissão de áudio, vídeo e reconhecimento de cores. Aplicações estas que tornaram o iStalker interessante e atrativo, possibilitando a idealização de outras aplicações. Dentre elas, a que mais se destaca é a construção de uma plataforma de desenvolvimento voltada a aprendizado, onde se pode alterar a interface gráfica de comunicação remota criada no presente trabalho afim de torna-la mais amigável e intuitiva, fazendo com que estudantes possam notar a diferença da resposta do sistema dinâmico, no caso do robô, quando variasse suas constantes, ou ainda mais, implementar sua própria arquitetura de controle.

Apesar do objetivo principal ser projetar um sistema para equilibrar o iStalker, e que este propósito tenha sido atendido, indubitavelmente se conclui que a dinâmica do robô, mesmo com desenvolvimento do sistema de controle embarcado, em algumas condições específicas é instável. Assim, faz-se necessário para as aplicações que tenha que desloca-lo, a limitação de sua velocidade a valores reduzidos para não perder estabilidade.

Pensando em melhorar a movimentação do robô poderia se tentar estratégias de caracterização, modelagem e controle do mesmo, como é visto em outros trabalhos, cujo o foco é somente o sistema controlado, onde em sua maioria se desenvolve um controlador linear quadrático LQR, mesmo assim não é garantido para o iStalker um funcionamento muito melhor.

A comunicação remota via rede sem fio é bastante suscetível à interferências e, como faz parte da aplicação do iStalker monitorar e alterar suas constantes, pensar em outros tipos de comunicação que perdessem menos pacotes de dados, como a transmissão de vídeo via rádio (que não apresentou nenhum problema), se torna imprescindível. A aplicação via rede sem fio com perda de dados e interferência, e o *Bluetooth* com alcance limitado, em alguns casos, comprometeram a utilização do robô.

A vida útil da bateria, apesar de ser suficiente para demonstração do trabalho e realização de ensaios, é de fato pequena quando utilizada a aplicação de monitoramento. Porém, se o iStalker for utilizado para fins de entretenimento, a autonomia da fonte é satisfatória comparada com outros brinquedos, como por exemplo os "quadcopteros" comerciais que, sem efeito do vento, duram aproximadamente trinta minutos. Mesmo que outros projetos com finalidades semelhantes tenham seu tempo de operação abreviado pela mesma limitação, alguns componentes como o microprocessador (devido a seu consumo) e os motores, que além de pesados são super dimensionados, poderiam ser substituídos de modo que o tempo de operação do robô fosse prolongado.

Referências

- [1] BUARQUE, D. Envelhecimento da população mundial preocupa pesquisadores. *G1 Mundo*, v. 29, 2011.
- [2] SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011.
- [3] LI, Z.; YANG, C.; FAN, L. *Advanced control of wheeled inverted pendulum systems*. [S.l.]: Springer Science & Business Media, 2012.
- [4] BARR, M.; MASSA, A. *Programming embedded systems: with C and GNU development tools*. [S.l.]: " O'Reilly Media, Inc.", 2006.
- [5] MEMARBASHI, H. R. *Design and parametric control of co-axes driven two-wheeled balancing robot: a thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in Mechatronics at Massey University, School of Engineering and Advanced Technology, Albany, New Zealand*. Tese (Doutorado) — Massey University, 2010.
- [6] GRASSER, F. et al. Joe: a mobile, inverted pendulum. *IEEE Transactions on industrial electronics*, IEEE, v. 49, n. 1, p. 107–114, 2002.
- [7] HELLMAN, H.; SUNNERMAN, H. *Two-Wheeled Self-Balancing Robot: Design and control based on the concept of an inverted pendulum*. 2015.
- [8] JODENSKI, L. et al. One robot to roll them all. 2015.
- [9] PRATAMA, D.; BINUGROHO, E. H.; ARDILLA, F. Movement control of two wheels balancing robot using cascaded pid controller. In: IEEE. *Electronics Symposium (IES), 2015 International*. [S.l.], 2015. p. 94–99.
- [10] SIMON, D. E. *An embedded software primer*. [S.l.]: Addison-Wesley Professional, 1999. v. 1.
- [11] BARRY, R. Mastering the freertos real time kernel-a hands on tutorial guide. *Real Time Engineers Ltd*, 2016.
- [12] INVENSENSE. *MPU9250 - Product Specification*. [S.l.], 2016. Rev. 1.1.
- [13] INVENSENSE. *MPU9250 - Register Map and Descriptions*. [S.l.], 2015. Rev. 1.6.
- [14] VELAZQUEZ, M. et al. Velocity and motion control of a self-balancing vehicle based on a cascade control strategy. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 13, n. 3, p. 106, 2016.

Apêndices

APÊNDICE A – Vista explodida do projeto

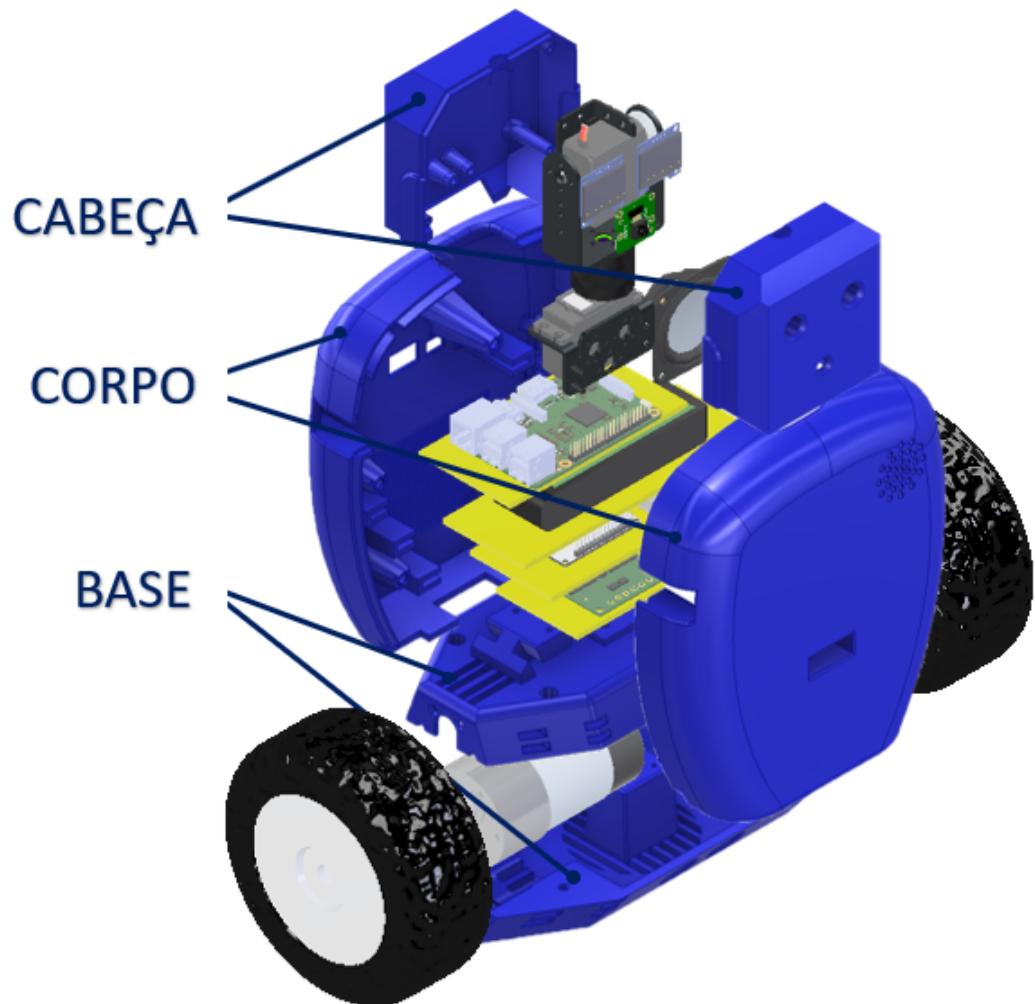


Figura 41 – Vista explodida da estrutura mecânica do projeto

APÊNDICE B – Dimensões finais do iStalker

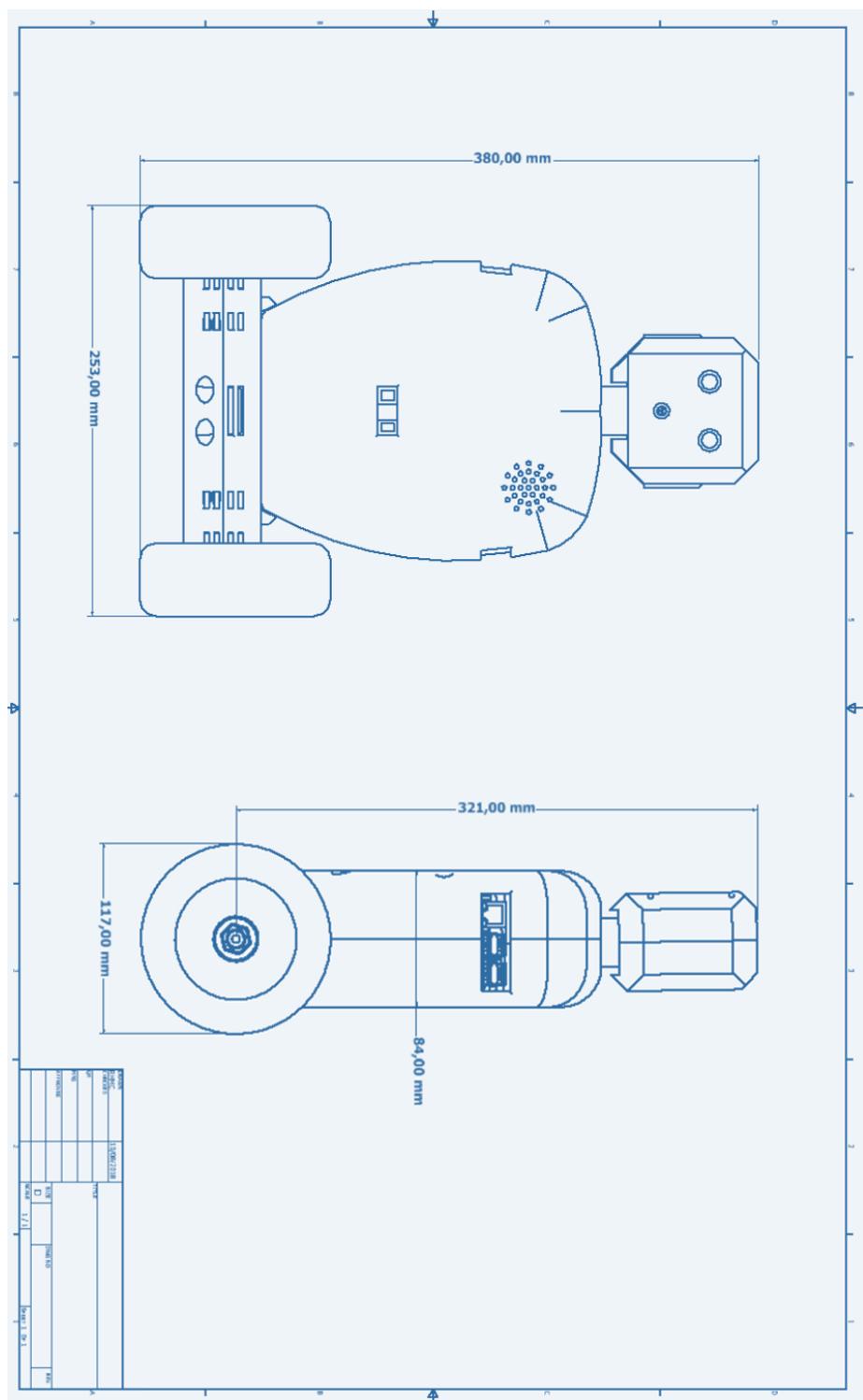


Figura 42 – Dimensões finais da estrutura do iStalker

APÊNDICE C – Teste de Impacto

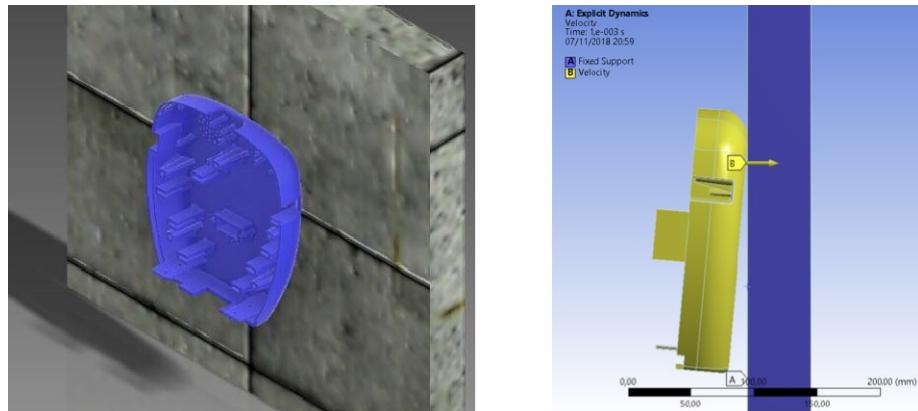


Figura 43 – Simulação de colisão do corpo com obstáculo

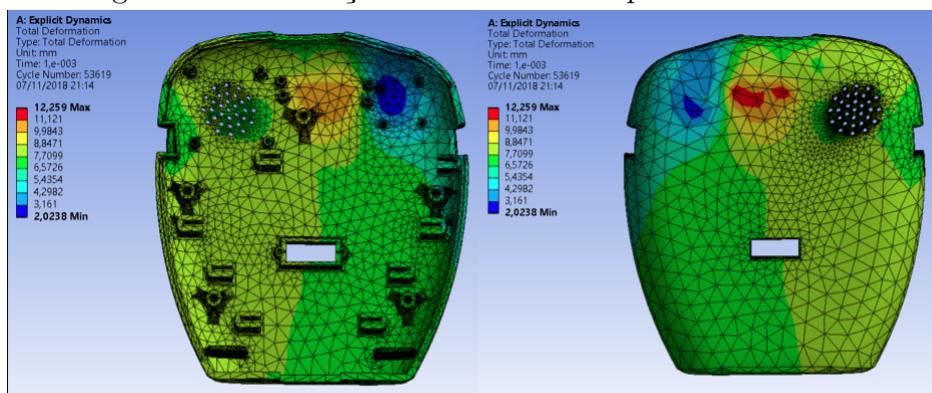


Figura 44 – Análise da deformação total no teste de impacto

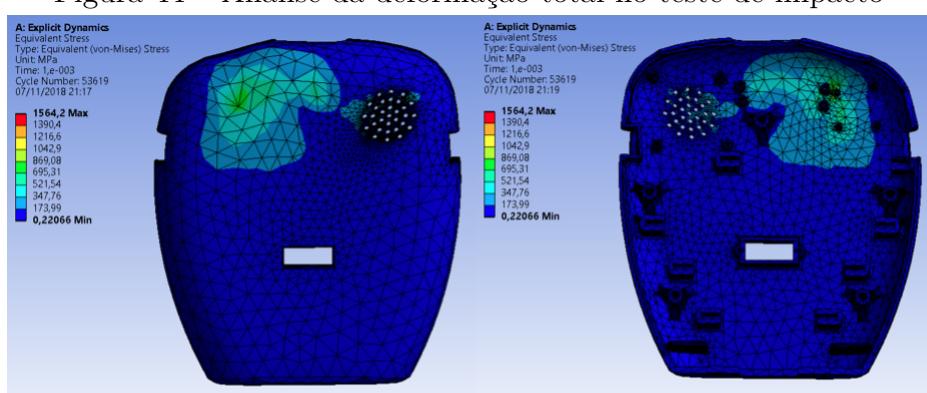


Figura 45 – Análise das tensões de Von-Mises

Simulações realizadas com auxílio do software ANSYS Workbench

APÊNDICE D – Montagem do Modelo

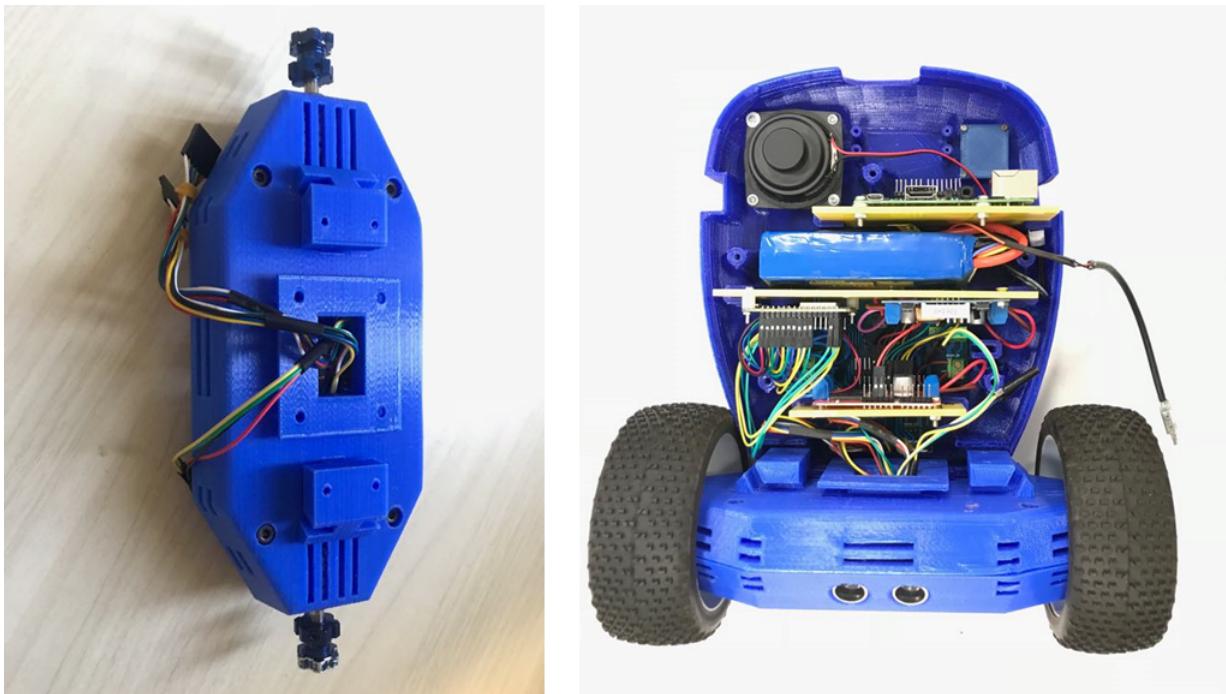


Figura 46 – Processo de Montagem do Modelo

APÊNDICE E – Protótipo finalizado



Figura 47 – Projeto finalizado ja em testes e projeto na Eureka, respectivamente

APÊNDICE F – Ações do Controle Remoto

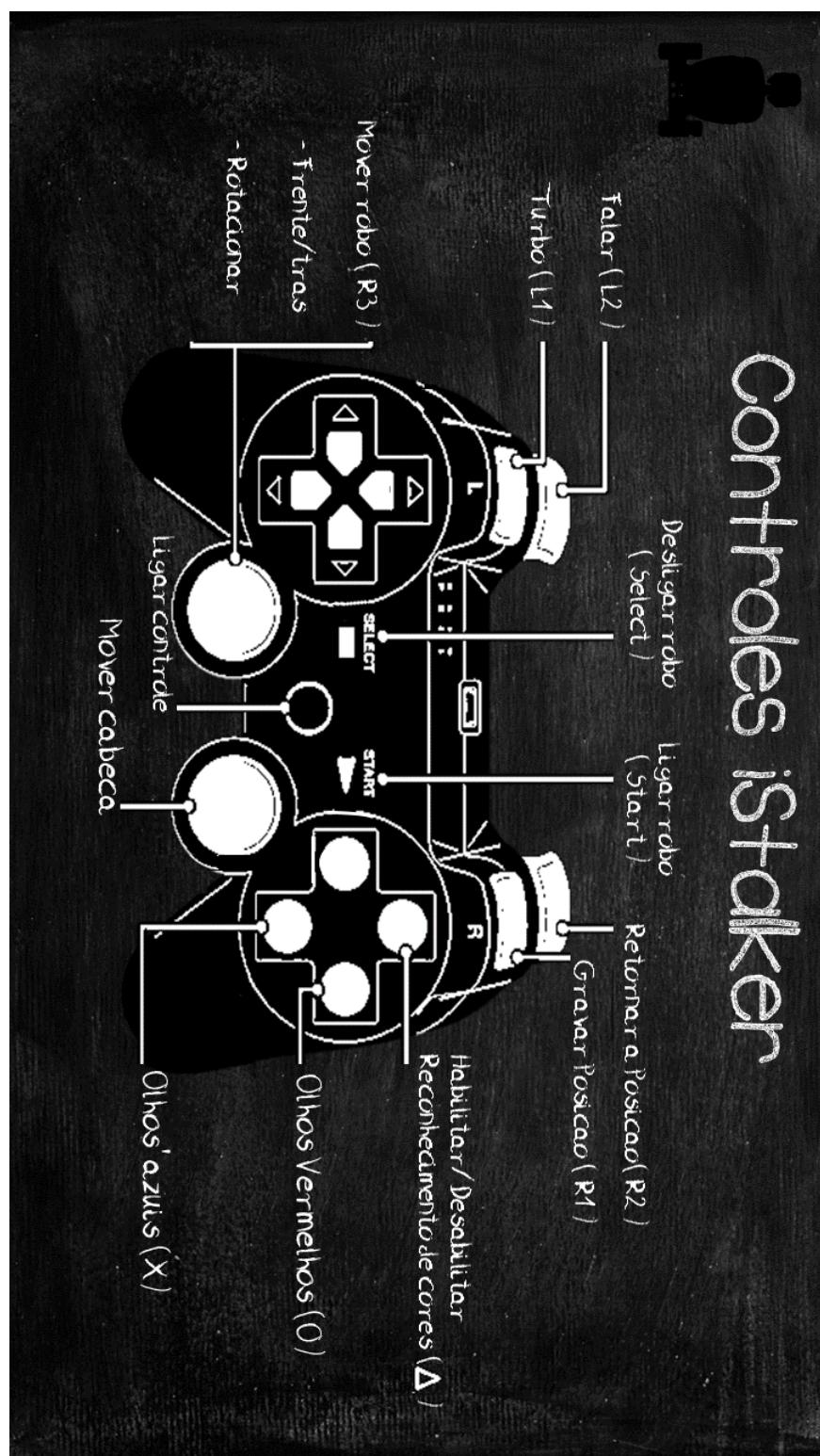


Figura 48 – Comandos programados no controle remoto via software embarcado

APÊNDICE G – Especificações do Projeto

Tabela 4 – Especificações gerais do Projeto

Acionamento	Motor DC Pololu 12V
Roda	117 mm
Sensores	Sensor Inercial (IMU), Câmera, Sensores Ultrassônicos, Encoders
Bateria	LiPo 3s - 12V 2200 mAh
Autonomia	30 minutos
Velocidade Máxima	0,5 m/s
Peso	2,54 Kg
Dimensões	380x253x117mm