Insper

Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI)

Engenharia Mecatrônica

Controle moderno de biciclo autônomo com visão computacional

Breno Alencar Araújo

Orientador: Prof. Dr. Gabriel Pereira das Neves

São Paulo

2025

Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI)

Breno Alencar Araújo

Engenharia Mecatrônica

Controle moderno de biciclo autônomo com visão computacional

Relatório final apresentado ao Programa Institucional de Bolsas de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI)

Orientador: Prof. Dr. Gabriel Pereira das Neves

São Paulo

2025

Resumo

O projeto direciona a modelagem física do sistema usando o método de Euler-Lagrange, a construção do modelo 3D e de um simulador em Simulink, bem como o desenvolvimento de um controle baseado no algoritmo LQR (Linear Quadratic Regulator), sua discretização e implementação em um microcontrolador com acesso ao hardware desenvolvido para aquisição de informações de ângulo, velocidade angular e velocidade angular das rodas, além do controle dos atuadores. Após essas etapas do Processo de Desenvolvimento de Produto, itera-se as matrizes de ajuste do algoritmo para melhor desempenho do controlador e testa-se o envio de setpoints para o microcontrolador. Em paralelo a isso, desenvolve-se o controle para movimentação dos servomotores que movimentam o suporte do microcontrolador com câmera fixado nele. Por fim, integra-se o controle de estabilidade e movimentação, com o controle dos servomotores e a visualização das imagens captadas pela câmera.

Palavras-chave: controle moderno, biciclo, filtro de Kalman, regra de Bryson, LQR,

ESP-32, Matlab, Simulink.

Abstract

The project focuses on the physical modeling of the system, using the Euler-Lagrange method, the construction of the 3D model and a simulator in Simulink, as well as the development of a control system based on the LQR (Linear Quadratic Regulator) algorithm, its discretization, and implementation on a microcontroller with access to the hardware developed for acquiring angle, angular velocity, and wheel angular velocity data, in addition to controlling the actuators. After these stages of the Product Development Process, the algorithm's tuning matrices are iterated for improved controller performance, and the transmission of setpoints to the microcontroller is tested. In parallel, the control system for driving the servomotors, which move the microcontroller support with the attached camera, is developed. Finally, the stability and motion control are integrated with the servomotor control and the visualization of the images captured by the camera.

Keyword: modern control; bicycle; Kalman filter; Bryson rule; LQR, ESP-32, Matlab, Simulink.

Sumário

1. Introdução 6
2. Objetivo 7
3. Metodologia 8
3.1. Matriz de Solução 8
3.2. WBS 12
4.1. Modelagem 3D 13
4.2. Parâmetros estimados 16
4.3. Desenvolvimento da Eletrônica 17
5.1. Construção Mecânica 20
6. Modelagem do Sistema Físico 23
8. Controle Embarcado Discreto 34
9. Conclusões 36
10. Trabalhos Futuros 38
Referências 39
Apêndices 40

1. Introdução

Controles de estabilidade são de extrema importância para uma extensa variedade de aplicações, seja pela necessidade de rejeição de distúrbios, como em um drone sob efeito de ventos fortes, ou para compensar o desbalanceio de um avião após o consumo de combustível, pelo conforto de um passageiro em um veículo, ou pela estabilização de uma câmera por um gimball. Essencialmente, este projeto trata do problema clássico de estabilização do pêndulo invertido, em que a resposta natural da planta ao distúrbio aplicado é a instabilidade, de forma que um controle é necessário para manter a referência. Variações desse sistema são aqueles monoapoiados (Neves, 2017) utilizando rodas de reação. No entanto, neste projeto as rodas fixadas aos dois atuadores apoiarão a estrutura suspensa, assim como em um Segway (Veen, 2018).

Este projeto direciona a modelagem do sistema, a construção da estrutura e dos hardwares necessários para controle e potência (Goes, 2018), além de projetar os controladores de um biciclo acoplado a um suporte pan/tilt móvel com hardware dedicado para visão computacional responsável por fornecer os setpoints para o controlador. As motivações para este estudo incluem a busca por soluções robustas para estabilização de sistemas móveis, como veículos autônomos, o desenvolvimento de algoritmos de processamento embarcado de imagens em núcleo separado e o desenvolvimento de novas abordagens para controle de sistemas mecânicos complexos e desacoplados (Dias, 2010), utilizando hardwares acessíveis e competitivos no mercado, como os da Espressif, muito utilizados em produtos IoT.

2. Objetivo

O objetivo do projeto é construir um biciclo e um modelo matemático para desenvolver e testar estratégias de controle. O projeto também integrará imageamento para resposta a estímulos visuais que confiram autonomia ao robô. Para isso, um hardware com sensores, atuadores e microcontroladores será desenvolvido para controle de baixo nível e outro para aquisição e processamento embarcado de imagens. Além disso, um simulador em Simulink será criado para testar as estratégias de controle que serão implementadas.

Com tais objetivos alcançados, restará para a instituição a documentação criada, o modelo da planta para simulação, bem como o construto (CAD e físico), o projeto do controlador e todo artifício utilizado para desenvolvimento do projeto em futuros trabalhos e pesquisas.

3. Metodologia

A metodologia do projeto segue os passos de um processo de desenvolvimento de produto (PDP) de Rozenfeld et al. (2006):

Fonte: elaborado pelo autor (2024)

O planejamento estratégico, no pré-desenvolvimento, se inicia no projeto de pesquisa com os requisitos de projeto, como tamanho aproximado, autonomia de bateria, custo, facilidade de montagem e de aquisição de peças, escalabilidade etc. Durante o projeto informacional são trabalhados esses requisitos, como limitações mecânicas, tempos de assentamento do controlador, tempo de processamento do microcontrolador para tomada de decisões. No projeto conceitual um modelo CAD deverá traduzir esses requisitos para uma planta que possa ser reproduzida. Por fim, durante o projeto detalhado, são desenvolvidos os detalhes a nível de componente e deverá ser produzido o dispositivo, além do controlador responsável pela estabilização da planta e dos algoritmos de visão computacional. Os testes são realizados e se obtém os primeiros resultados, que devem ser sucedidos por iterações até chegar aos requisitos. As etapas em cinza na imagem acima, não farão parte deste projeto.

3.1. Matriz de Solução

Para o desenvolvimento das atividades, foi necessário um estudo das alternativas para se construir o biciclo. Elas foram elencadas em matrizes de solução e uma variante de solução foi escolhida para eletrônica, mecânica e programação.

Tabela 1 - Matriz de solução de eletrônica

Função	Princípios de solução		
	1	2	3
Fabricação	JLCPCB	PCBWay	KKPCB
Montagem	PTH	SMD	Mista
Processamento e controle	ESP32	STM32F411	Teensy
Acionamento para motor	TB6612FNG	DRV8833	A4988
Movimentação	DC	Brushless	-
Aquisição de ângulos	MPU6050	MPU 9250	BMI 160
Carregar bateria	TP4056	DDTCCRUB	134N3P
Regular tensão	TPS54332	AP63205	ME3116AM6G

Fonte: elaborado pelo autor (2024)

A fabricação da PCB foi destinada para a JLCPCB por conta do rápido prazo e menor custo, o fornecedor vieram de anúncios do Mercado Livre e AliExpress utilizando módulos prontos por conta do baixo custo e fácil aquisição de peças, o tipo de placa escolhida foi a SMD por conta do menor espaço requerido, com exceção de alguns componentes. O microcontrolador escolhido foi a ESP32 por conta do baixo custo e as funcionalidades IoT. Para o driver dos motores, escolheu-se o TBGG12FNG por conta das especificações de potência fornecida e baixo custo juntamente com o motor DC para facilitar o acionamento. Para o módulo de carregamento, o DDTCCRUB foi escolhido por conta da configuração das baterias em 2S. O regulador de tensão escolhido foi o ME3116AM6G por conta do baixo custo, da facilidade na montagem no circuito e do fornecimento suficiente de potência para o restante do circuito.

Tabela 2 - Matriz de solução de mecânica

Função	Princípios de solução		
	1	2	3
Suportes horizontais	Em acrílico	Em aço	Em madeira
Suportes verticais	Standoff em latão	Impressos	Em acrílico
Redução	Planetária	Por eixos paralelos	Parafuso sem-fim
Pneu	Borracha lisa	Borracha macia	Borracha dura

Fonte: elaborado pelo autor (2024)

Os suportes em acrílico foram escolhidos por conta da resistência, baixo peso em relação ao de aço e acabamento após corte. Para os suportes verticais, serão usados os standoff em latão por conta da melhor fixação. A redução escolhida foi a por eixos paralelos por conta do baixo custo. O pneu de borracha macia foi escolhido por reduzir vibrações, imperfeições do solo e minimizar derrapagens. O suporte do motor escolhido foi o em L por conta da melhor fixação.

Tabela 3 - Matriz de solução de programação

Função	Princípios de solução		
	1	2	3
Modelagem 3D	SolidWorks	Catia	Siemens NX
Modelagem matemática	Octave	MapleSim	MATLAB
Simulação	Scilab	Simulink	LabVIEW
Arquitetura de firmware	Via interrupção	Via threads	-
IDE	ESP-IDF	Arduino IDE	PlatformIO

Fonte: elaborado pelo autor (2024)

Para a modelagem utilizou–se o SolidWorks 2020 por maior perícia, a modelagem matemática foi desenvolvida no MATLAB pela disponibilidade e integração com o ambiente de simulação Simulink. A arquitetura de firmware foi desenvolvida em threads para viabilizar o processamento em tempo real. Para isso, optou-se por utilizar a ESP-IDF por facilitar a melhor configuração do microcontrolador.

3.2. WBS

Com cada uma das soluções definidas, os entregáveis para a realização de cada uma das soluções foram divididos em cinco grandes áreas, são elas: mecânica, eletrônica, modelagem física, controle e visão computacional.

Tabela 4 - Work Breakdown Structure

Categoria	Tarefas
Mecânica	Desenvolver base para os suportes dos motores. Desenvolver contorno da placa. Desenvolver fixação entre base e placa. Desenvolver suporte para baterias e para o suporte pan/tilt. Especificar porcas, parafusos e espaçadores. Assinalar materiais. Montagem completa. Extração dos parâmetros físicos aproximados.
Eletrônica	Desenvolver esquemático da IMU. Desenvolver layout da IMU. Desenvolver esquemático do driver de potência. Desenvolver layout do driver de potência. Desenvolver esquemático da MCU. Desenvolver layout da MCU. Desenvolver esquemático do carregador de baterias. Desenvolver layout do carregador de baterias. Desenvolver esquemático da placa da câmera. Desenvolver layout da placa da câmera. Desenvolver esquemático para acionamento dos servomotores. Desenvolver layout para acionamento dos servomotores.
Modelagem Física	Equacionar cinemática. Equacionar energias com o método de Lagrange. Equacionar eletromecânica. Obter sistema não linear. Linearizar sistema. Encontrar matrizes A e B.
Controle	Projetar primeiro LQR discreto. Embarcar controlador no microcontrolador. Enviar e receber comandos via WIFI.
Visão Computacional	Desenvolver algoritmo de detecção de objeto. Desenvolver algoritmo para tomada de decisão.

Fonte: elaborado pelo autor (2024)

4. Desenvolvimento

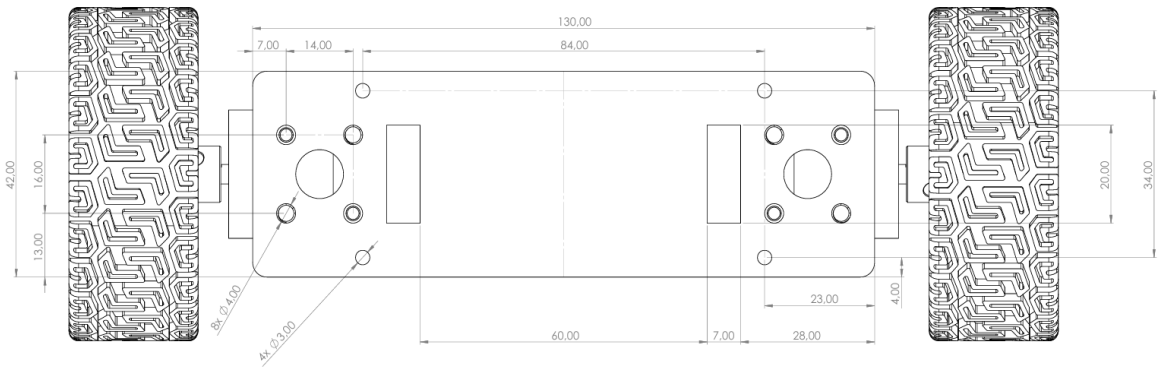
Seguindo os entregáveis do WBS e o cronograma, após as definições no planejamento estratégico, as atividades se desenvolveram até a montagem da estrutura mecânica e projeto da placa de circuito.

4.1. Modelagem 3D

Visando a aquisição dos parâmetros estimados para a modelagem e controle do biciclo, um modelo tridimensional foi criado no software de modelagem SolidWorks 2020, cujos documentos podem ser encontrados em anexo e no repositório do GitHub. É na modelagem que são desenhadas as peças que serão suporte para os atuadores (motores), para a placa de circuito impresso, para o suporte da bateria e para o suporte pan/tilt da câmera. Escolheu-se chapas de acrílico de 3 mm para a fabricação. A montagem final é útil para verificar se o modelo é de fácil montagem e, por fim, quais são os parâmetros resultantes da geometria do construto.

Iniciando pela base, ela precisa ter pelo menos a largura dos suportes dos motores e mais uma sobra para o suporte da placa de circuito, que será a base para o restante do dispositivo. Tais suportes foram adquiridos juntamente com os motores com encorders e caixas de redução, os cubos de roda e as rodas. Esses componentes (motor e caixa de redução) podem ser usados para a modelagem e futura construção do robô pois possuem dimensões muito semelhantes entre os modelos, mudando apenas algumas características eletromecânicas estudadas posteriormente para o controle.

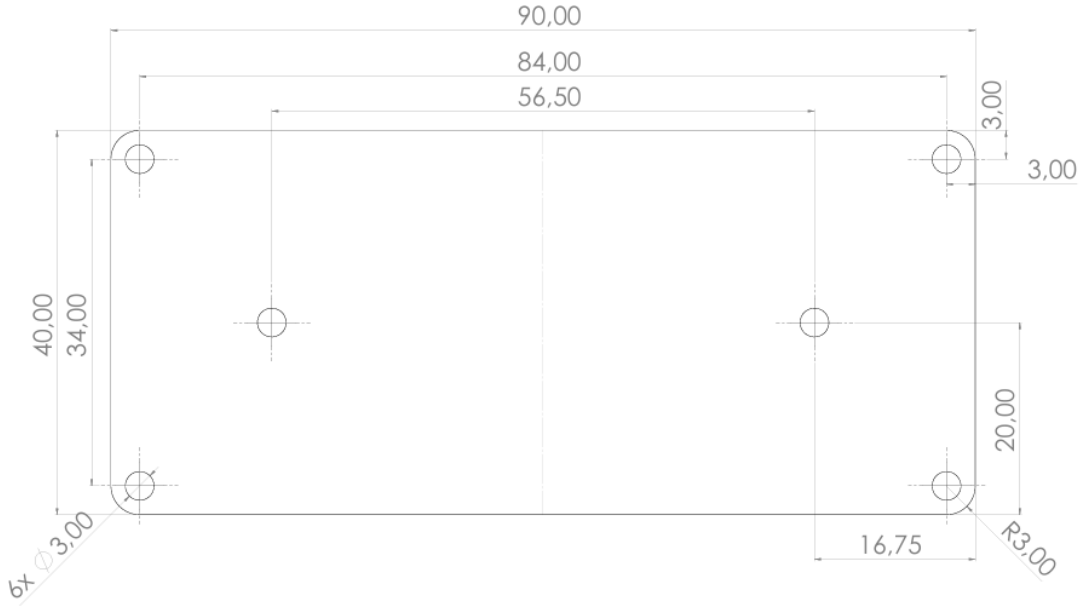
Figura 1 - Desenho da base com as rodas



Fonte: elaborado pelo autor (2025)

Para a placa de circuito, o requisito pede que tenha dimensão máxima de 90x40mm. Os dois furos centrais são utilizados para a fixação dos espaçadores que suportarão o restante do robô. Os quatro furos próximos aos cantos arredondados serão para fixação da placa na base. Esquemáticos e detalhes do layout serão abordados na seção de Construção Eletrônica.

Figura 2 - Desenho das bordas da PCB com furos

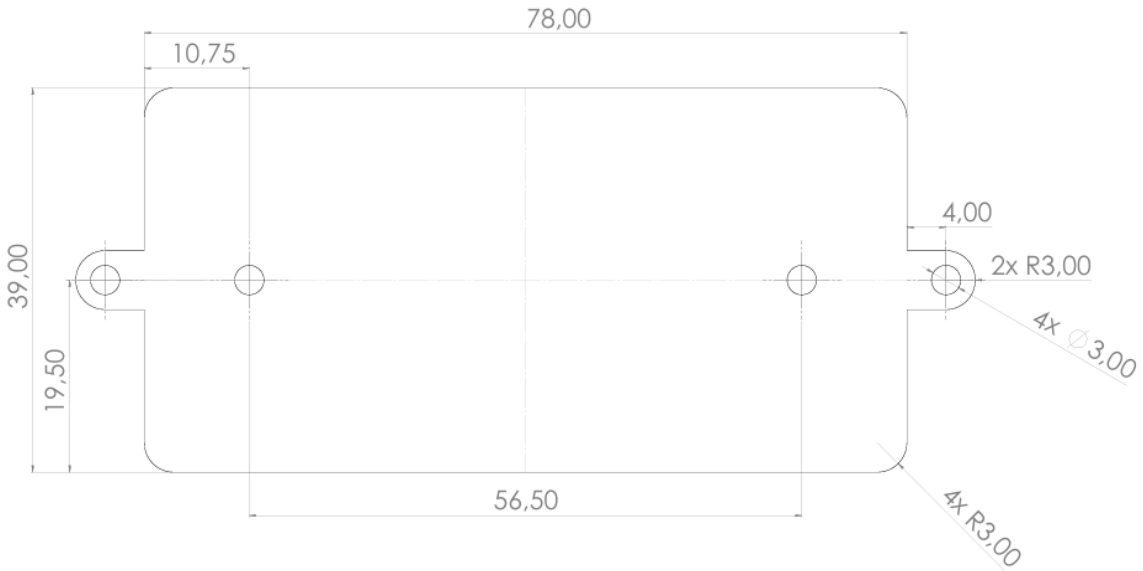


Fonte: elaborado pelo autor (2025)

Para o suporte das baterias e da câmera, as furações já estavam definidas por esses componentes, de forma que apenas foram acrescentados dois furos a essas

peças. No caso do suporte das baterias, os dois furos centrais fixam o suporte à placa e aos espaçadores.

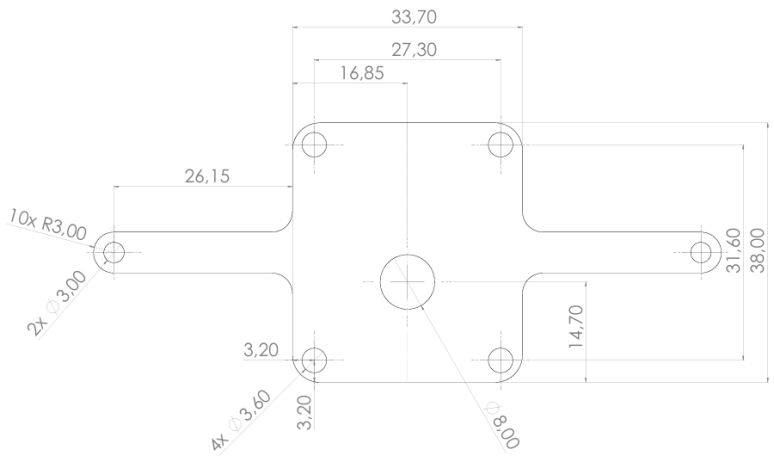
Figura 3 - Desenho do suporte das baterias



Fonte: elaborado pelo autor (2025)

Para o suporte pan/tilt, os quatro furos de maior diâmetro ficam responsáveis por fixá-lo e os dois furos mais afastados fixam os espaçadores. O furo central serve para acessar o parafuso que fixa o servo motor ao suporte.

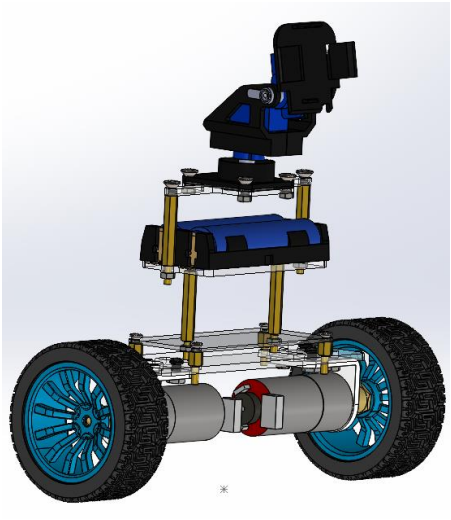
Figura 4 - Desenho do suporte para a câmera



Fonte: elaborado pelo autor (2025)

Por fim, a montagem final, com os motores, as rodas e os suportes, espaçadores, porcas, parafusos, baterias e o suporte pan/tilt já com os servos. Com este modelo tridimensional já é possível extrair os parâmetros após assinalar os materiais.

Figura 5 - Montagem completa do robô



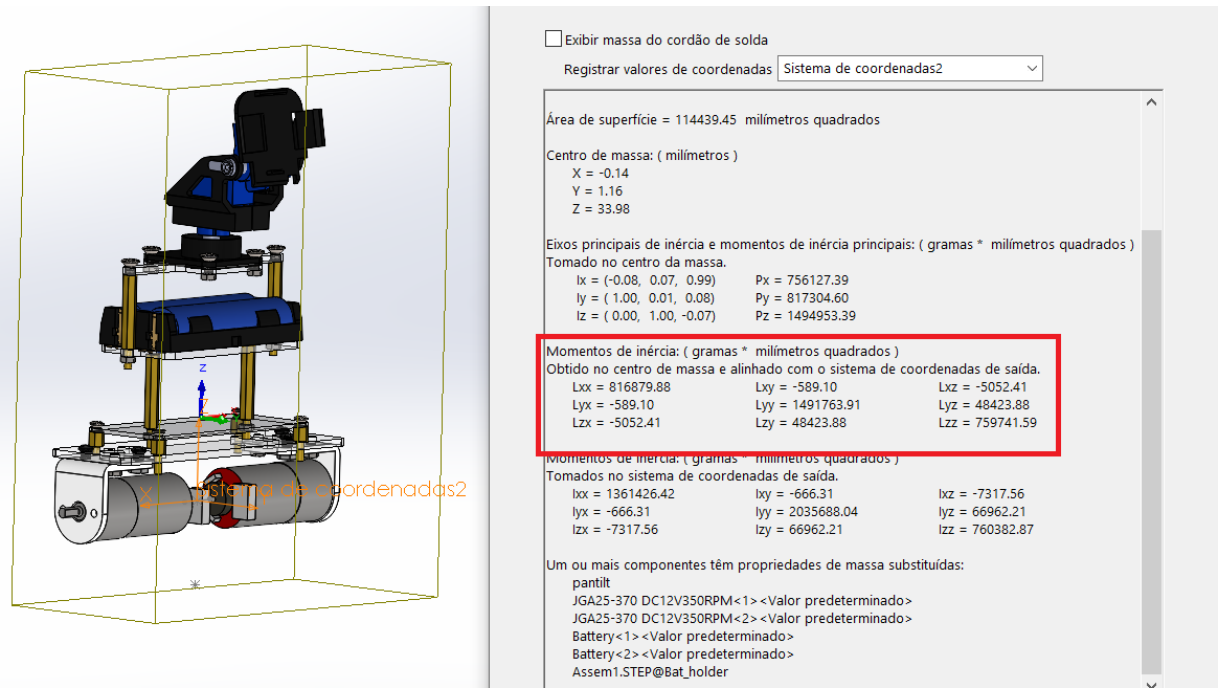
Fonte: elaborado pelo autor (2025)

4.2. Parâmetros estimados

Finalizada a montagem, segue-se para a aquisição dos parâmetros. Para isso, precisamos definir os materiais dos corpos homogêneos, como os suportes de acrílico e os espaçadores de latão, além de fornecer as massas de outros componentes não homogêneos, como os motores, as rodas, as baterias etc.

Para a modelagem matemática o tensor de inércia do corpo sem as rodas é necessário, ou seja, a matriz de segundo grau dos momentos de inércia do corpo, que está assinalada na caixa em vermelho na imagem abaixo. É definido um sistema de coordenada (“Sistema de Coordenadas2”) que será utilizado para a modelagem e para o cálculo dos momentos de inércia. Na imagem a seguir, ele está no eixo das rodas, na linha de simetria do robô, com o eixo y apontando para a frente e o eixo x para a direita, com o eixo z apontando para cima, fechando o sistema rotativo no sistema horário.

Figura 6 - Tensor de inércia estimado do robô



Fonte: elaborado pelo autor (2025)

Além disso, o momento de inércia das rodas sem o restante do corpo também será necessário e é obtido da mesma forma que o anterior.

4.3. Desenvolvimento da Eletrônica

O hardware necessário foi dividido em duas grandes partes: lógica e potência. O circuito de lógica é necessário para aquisição das informações vindas dos sensores, o processamento dessas informações com o controle, a instrução para o driver de potência acionar os atuadores e a comunicação com outros dispositivos, estando presente em duas placas de circuito: a inferior que ficará próximo às rodas e a superior que ficará próxima à câmera. O circuito de potência é necessário para energizar os atuadores conforme orientação do circuito de lógica, energizar os circuitos da placa e gerenciar o carregamento das baterias.

Optou-se pelo projeto de uma placa de circuito com componentes SMD visando reduzir o tamanho do hardware e o consumo de energia.

Para isso, com a variante de solução definida, criou–se o esquemático macro das ligações entre os circuitos.

Em seguida, dentro de cada um dos circuitos, desenvolveu-se o esquemático de ligações. Visando facilitar a aquisição dos componentes, optou-se por utilizar

módulos prontos, como devkits de ESP32, e adaptá-los à placa, recriando seus esquemáticos para posterior aproveitamento dos componentes.

Iniciando pelo circuito de lógica, desenvolveu-se o esquemático de ligações do microcontrolador ESP32-WROOM-32D de arquitetura da Xtensa 32 bit e dois núcleos com 4MB de memória FLASH, 520KB de memória SRAM, WIFI, Bluetooth, 25 GPIOs, conversor USB para Serial CH340C e interface USB-C. Ele será responsável pela aquisição das informações vindas dos sensores, processamento de informações, controle do driver de potência e comunicação com outros dispositivos.

Para o circuito de aquisição de ângulos e velocidades angulares desenvolveu-se o esquemático de ligações da IMU MPU6050 da Invensense.

Para a parte de potência, desenvolveu-se o esquemático de ligações do driver de potência para motor DC da Toshiba, o TB6612FNG, que receberá os sinais da ESP de PWM e sentido de rotação e chaveará a carga (motores) com a ponte H interna de seu circuito.

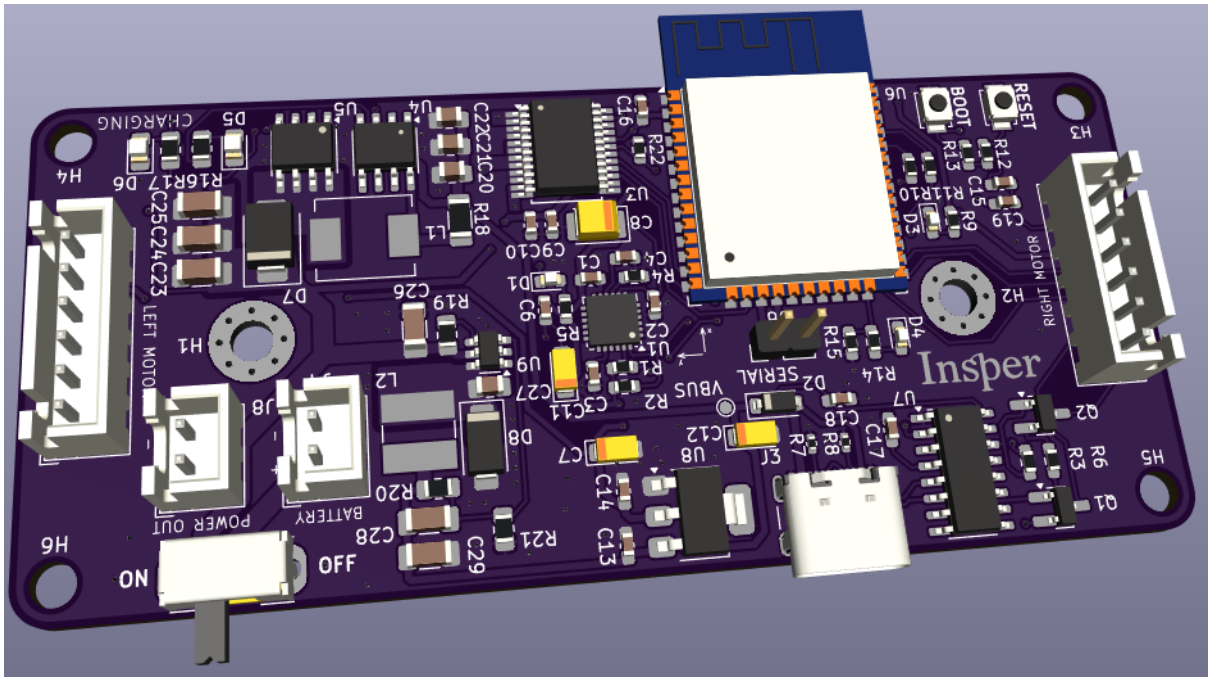
Para o carregamento das baterias, desenvolveu-se o esquemático de ligações para o driver de carregamento CN3302 que permite o carregamento de duas baterias de lítio 18650 em configuração 2S, ou seja, duas baterias em série.

Para o regulador chaveado step down, desenvolveu-se o esquemático para o regulador DC-DC ME3116, de 8,4V vindos da bateria para 5V com 1 A.

Por fim, os conectores da bateria, dos motores e para comunicação serial foram posicionados e a construção do layout da placa foi iniciado.

Para evitar as forças de Coriolis, restringiu-se o posicionamento do circuito integrado da IMU para o centro da placa. O circuito da ESP ficou à direita, o driver dos motores ficou acima da IMU, o circuito de carregamento ficou à esquerda. Cada um dos conectores para os motores ficou em uma lateral da placa, o conector para comunicação serial ficou próximo do módulo ESP.

Figura 7 - Modelo 3D da placa principal



Fonte: elaborado pelo autor (2025)

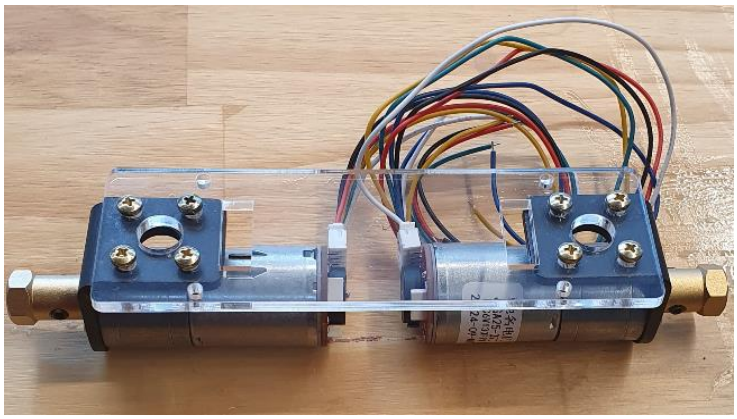
5. Construção

Com a virtualização mecânica finalizada, as partes modeladas puderam ser fabricadas e montadas para a validação das cotas.

5.1. Construção Mecânica

Após completar a modelagem tridimensional, as peças de acrílico, que foram fabricadas, já podiam ter seus desenhos exportados para o formato DXF e enviados para o software RDWorksV8, que gera um arquivo para corte na cortadora a laser. Com isso, a base com os motores pôde ser montada.

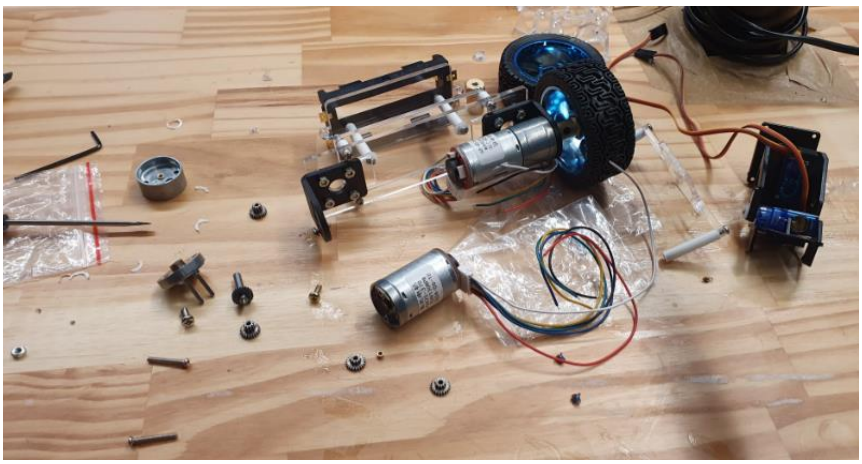
Figura 8 - Base montada com os suportes, os motores e os cubos de roda



Fonte: acervo pessoal do autor (2025)

Então fixou-se os suportes dos motores na base com parafusos e porcas M3 e os motores aos seus suportes também com parafusos M3. O cubo de roda foi fixado com dois parafusos allen sem cabeça que vieram com o kit dos motores e um parafuso M4 travou a roda ao cubo. Ao fixar os dois motores, no entanto, um deles apresentou eixo travado e uma desmontagem posterior revelou pedaços do ímã que obstruem a livre movimentação do rotor, sendo necessária a substituição desse.

Figura 9 - Motor apresentou problemas ao ser testado



Fonte: acervo pessoal do autor (2025)

Para o restante da montagem, é necessário a fixação dos espaçadores de latão. Então, espaçadores provisórios de PLA foram impressos com as mesmas dimensões dos de latão e a montagem pôde ser finalizada utilizando parafusos para fixação entre acrílico e espaçador e parafuso com porca para o caso do suporte pan/tilt. Para esse, uma adaptação precisou ser feita: a furação era de 2.5 mm, então com uma broca de 3 mm os furos foram alargados para padronizar a fixação das peças.

Para a montagem do suporte pan/tilt, foi necessário adaptar as peças dos servomotores para que coubessem nos espaços reservados para elas no suporte, tanto no suporte em cruz, quanto na outra junta do suporte. A fixação desses foi feita com os parafusos que vieram junto com o kit do servomotor.

Figura 10 - Suporte Pan/tilt com parafusos, kit dos servos, ESP32-CAM e chave para aperto dos parafusos

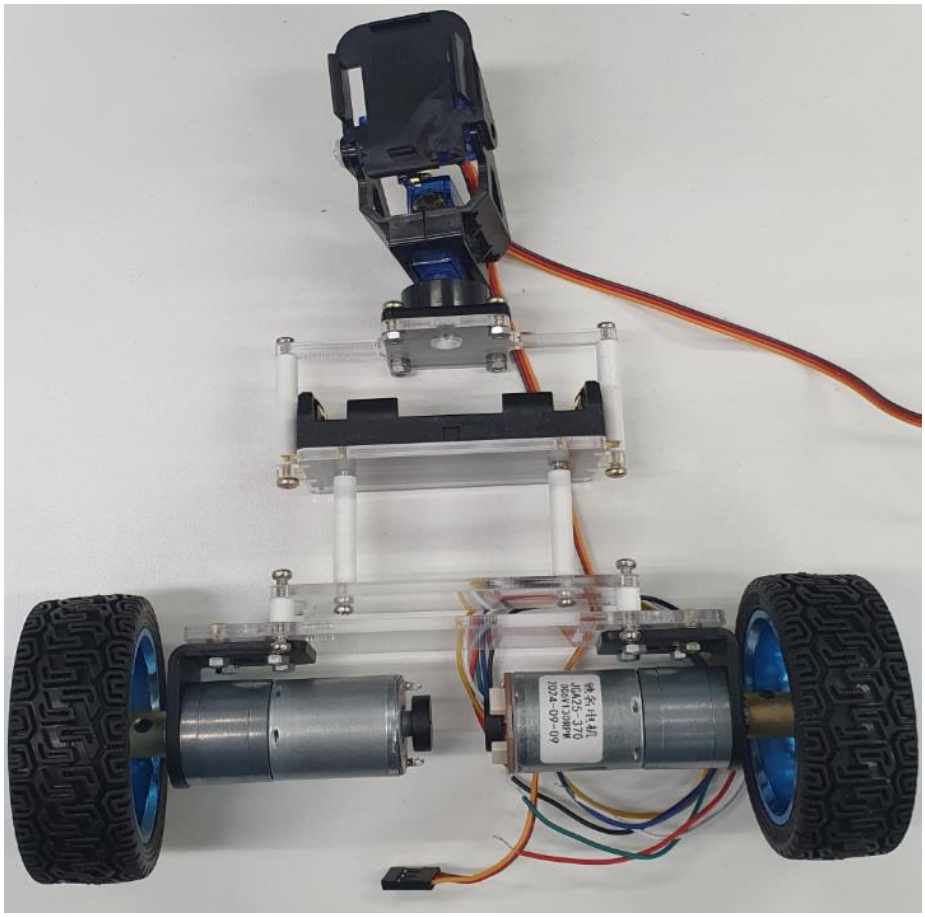
Fonte: acervo pessoal do autor (2025)

Figura 11 - Peças do servo adaptadas para encaixe no suporte

Fonte: acervo pessoal do autor (2025)

Com a montagem finalizada, seguiu-se para o desenvolvimento da eletrônica do robô.

Figura 12 - Montagem completa

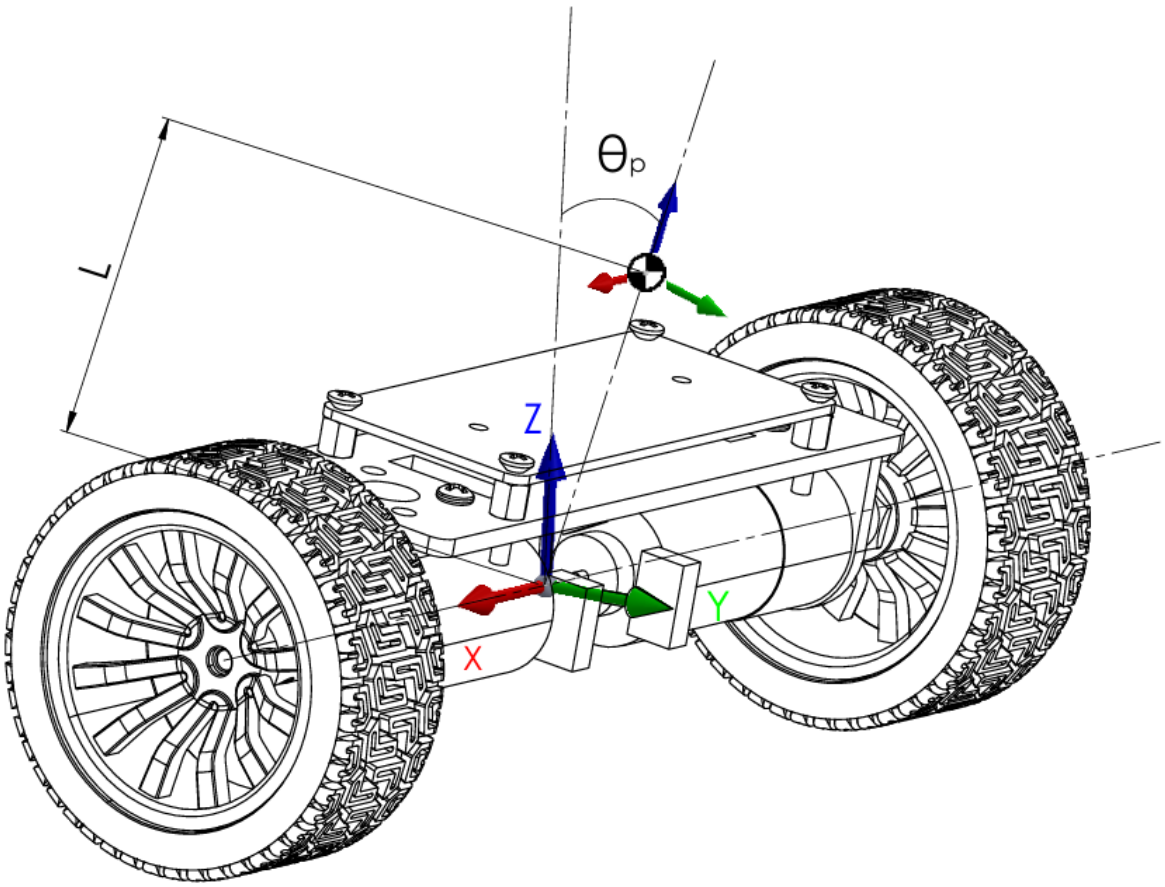


Fonte: acervo pessoal do autor (2025)

6. Modelagem do Sistema Físico

Para a modelagem do robô, dois sistemas de coordenadas foram definidos: o primeiro está no eixo de rotação das rodas, com o eixo x apontando para a roda direita e o eixo z está na linha de simetria longitudinal do robô perpendicular ao solo e apontando para cima. O segundo eixo é móvel e está localizado no centro de massa do robô, com seus eixos nas mesmas direções e sentidos do sistema anterior.

Figura 13 -Sistema de coordenadas



Fonte: elaborado pelo autor (2025)

Com os sistemas definidos, é possível aplicar o método de Lagrange. Para isso, foi necessário resolver a equação a seguir:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau_{ext}, \tag{1}$$

em que 'L' representa a Lagrangiana que é a diferença entre a energia cinética e a energia potencial do sistema, τ_{ext} são os torques externos e 'q' representa os graus

de liberdade do sistema, logo serão necessárias três equações: uma para cada ângulo de rolagem das rodas e outra para o ângulo de arfagem do robô.

Para o cálculo da energia potencial, apenas a massa do corpo do robô (m_p) foi contabilizada, pois o centro de massa das rodas está na mesma altura da origem do sistema de coordenadas, logo a energia potencial do sistema é representada por

$$U = m_p g L cos(\theta_p) \text{ (2)},$$

sendo 'g' a gravidade, 'L' a altura do centro de massa e 'θ_p' o ângulo de inclinação do robô.

Para o cálculo da energia cinética, tanto o corpo quanto as rodas possuem contribuição, de forma que foram calculadas separadamente e depois somadas.

O robô, quando em movimento, faz com que as rodas possuam tanto energia cinética de translação quanto energia cinética de rotação, desta forma, sua energia cinética é determinada por

$$K_i = \frac{1}{2} m_r \omega_i^2 r^2 + \frac{1}{2} J_r \omega_i^2 \text{ (3)},$$

sendo 'm_r' a massa das rodas, 'ω' a velocidade angular da roda, 'r' o raio das rodas e 'J_r' o momento de inércia da roda. Haverá, portanto, um K_R e um K_L, sendo a energia cinética de cada uma das rodas.

Para a energia cinética do corpo do robô também há translação do centro de massa e rotação em torno do eixo das rodas

$$K_c = \frac{1}{2} \vec{V}_{cm}^{\top} m \vec{V}_{cm} + \frac{1}{2} \vec{\Omega}_{cm}^{\top} I \vec{\Omega}_{cm} \text{ (4)},$$

sendo 'I', o tensor de inércia obtido via modelagem computacional, 'V_{cm}' a velocidade do centro de massa em relação ao sistema de coordenada fixo e '$\vec{\Omega}_{cm}$' a velocidade de rotação do corpo tomado no centro de massa.

A velocidade de translação do centro de massa é a composição da velocidade do sistema de coordenadas da base com a velocidade angular do corpo devido à sua rotação em torno do eixo dos motores, em outras palavras

$$\vec{V}_{cm} = \vec{V}_0 + \vec{\Omega}_0 \times (CM - 0) \text{ (5)},$$

sendo CM a posição do centro de massa e os vetores com índice '0' correspondem as velocidades do sistema de coordenada fixo ao eixo dos motores, e 'cm'correspondem às velocidades calculadas no sistema de coordenada fixado no centro de massa.

A velocidade do sistema de coordenadas da base pode ser escrita em função das velocidades angulares das rodas e do raio (R), da forma de que

$$\vec{V}_0 = \frac{1}{2}(\vec{\omega}_R + \vec{\omega}_L)R \text{ (6)},$$

Já a velocidade de rotação obtida em relação ao primeiro sistema de coordenadas é dada pela composição das velocidades angulares em torno do sistema de coordenada

$$\vec{\Omega}_0 = -\dot{\theta}_p\hat{\imath} + 0\hat{\jmath} + (\omega_R - \omega_L)\frac{R}{D}\hat{k} \text{ (7)},$$

sendo 'ω_p' a velocidade angular de inclinação do robô, 'ω_R' e 'ω_L' as velocidades angulares de cada uma das rodas e D a distância entre as rodas.

A distância CM do sistema de coordenadas é dada por:

$$CM = 0\hat{\imath} + Lsin(\theta_p)\hat{\jmath} + Lcos(\theta_p)\hat{k} \text{ (8)}$$

Por fim, a velocidade angular do corpo em torno do centro de massa é dada por:

$$\vec{\Omega}_{cm} = \begin{pmatrix} 1 & 0 & 0 & 0 & cos\,cos\,\theta_p & \theta_p & 0 & \theta_p & cos\,cos\,\theta_p \end{pmatrix}\vec{\Omega}_0 \text{ (9)}.$$

Com as energias cinéticas e potencial definidas, pode-se definir a Lagrangiana

$$L = K_c + K_R + K_L \text{ (10)}$$

que em seguida, aplicando equação de Euler-Lagrange, apresentado em (1), para cada variável $q = \begin{bmatrix}\theta_p\ \theta_R\ \theta_L\end{bmatrix}^{\top}$, e considerando os torques externos como torques das rodas, sendo que para o ângulo de arfagem ele é a soma dos torques das rodas esquerda e direita, têm-se as três equações:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = -(\tau_R + \tau_L) \text{ (11)},$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau_R \text{ (12)},$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau_L \text{ (13)}$$

O torque de um motor DC é diretamente proporcional a corrente que passa por ele: $\tau = K_t.i$, sendo K_t a constante de torque do motor.

Um motor DC pode ser modelado da seguinte forma

$$PWM \cdot V_{max} - R_m i - L_m\frac{di}{dt} - K_e\theta_p = 0 \text{ (14)}$$

sendo PWM um valor entre 0 e 1, V_{max} a tensão máxima fornecida ao motor, R_m a resistência do enrolamento da bobina do motor, 'i' a corrente consumida pelo motor, L_m a indutância da bobina e K_e a constante de velocidade do motor. O termo indutivo

pode ser desprezado uma vez que a frequência do PWM (*Pulse Width Modulation*) utilizada é consideravelmente alta, em torno de 10kHz. Ao isolar a corrente de (2) e considerando o ângulo relativo entre o estator e o rotor dos motores anexados às rodas ($\theta = \theta_i - \theta_p$) obtém-se

$$\tau_R = \frac{K_t}{R_m}\left[PWM \cdot V_{max} - K_e\left(\theta_R - \theta_p\right)\right] \text{ (15)},$$

$$\tau_L = \frac{K_t}{R_m}\left[PWM \cdot V_{max} - K_e\left(\theta_L - \theta_p\right)\right] \text{ (16)}$$

Agora com a Lagrangiana e os torques definidos, é possível resolver as equações e encontrar o modelo não linear do sistema. Antes, no entanto, é necessário realizar uma mudança de variáveis de interesse, pois o controle deverá ser feito sobre o ângulo de inclinação (arfagem) do robô (ângulo de $pitch$), o ângulo de guinada (ângulo de *yaw*) e a sua velocidade linear de translação. Para isso, duas equações cinemáticas são desenvolvidas. A primeira estabelece uma relação entre o deslocamento linear do robô y como uma média do deslocamento das rodas

$$y = \frac{R}{2}(\theta_R + \theta_L) \text{ (17)}.$$

A segunda estabelece uma relação entre o ângulo de guinada do robô e a diferença entre o deslocamento das rodas

$$\alpha = \frac{R}{D}(\theta_R - \theta_L) \text{ (18)}.$$

O modelo não linear pode ser reescrito na sua forma geral para sistemas mecânicos, da forma

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = P \cdot u \text{ (19)},$$

onde $M(q)$ é a matriz de inércia, $V(q,\dot{q})$ é a matriz de Coriolis e forças centrífuga, $G(q)$ é a matriz de forças generalizadas conservativas e P a matriz das entradas aplicadas. Finalmente, o modelo não linear pode ser obtido isolando $\ddot{q}$, resultando em

$$\ddot{q} = M(q)^{-1}\left(Pu - V(q,\dot{q}) - G(q)\right) \text{ (20)}$$

Com as matrizes definidas, iniciou-se o processo de linearização do modelo em torno das condições de equilíbrio em que $\theta_p = 0, \dot{\theta}_p = 0, \alpha = 0, \dot{\alpha} = 0$ e $\dot{y} = 0$. Resultando o modelo em espaço de estados

$$\dot{x}(t) = Ax(t) + Bu(t) \text{ (21)}$$

onde o vetor de variáveis de estado é $x = [\theta_p \quad \dot{\theta}_p \quad \alpha \quad \dot{\alpha} \quad \dot{y}]$ e as matrizes são

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 113.5059 & -13.2905 & 0 & 0 & 0 & 408.9379 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & -45.7580 & 0 & 0 \\ -8.6516 & 3.0155 & 0 & 0 & 0 & -92.7855 \end{bmatrix}$$

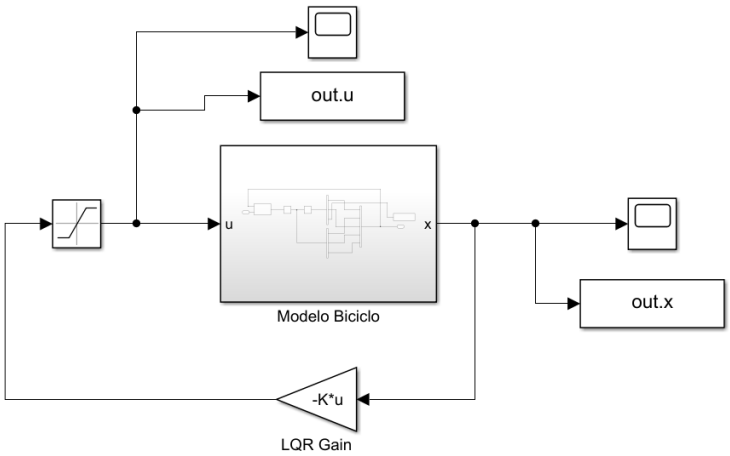
$$B = \begin{bmatrix} 0 & 0 \\ -97.4243 & -97.4243 \\ 0 & 0 \\ 114.7502 & -114.7502 \\ 22.1050 & 22.1050 \end{bmatrix}$$

Com isso, um Regulador Linear Quadrático foi implementado para regular os estados de interesse. Esse regulador foi necessário para especificar corretamente os parâmetros do motor que inicialmente haviam sido escolhidos de forma arbitrária.

7. Simulação da Estratégia de Controle

Em posse das matrizes e dos ganhos do LQR, foi possível criar um simulador em Simulink para avaliar o comportamento do sistema e auxiliar no desenvolvimento do controlador.

Figura 14 -Simulador do biciclo

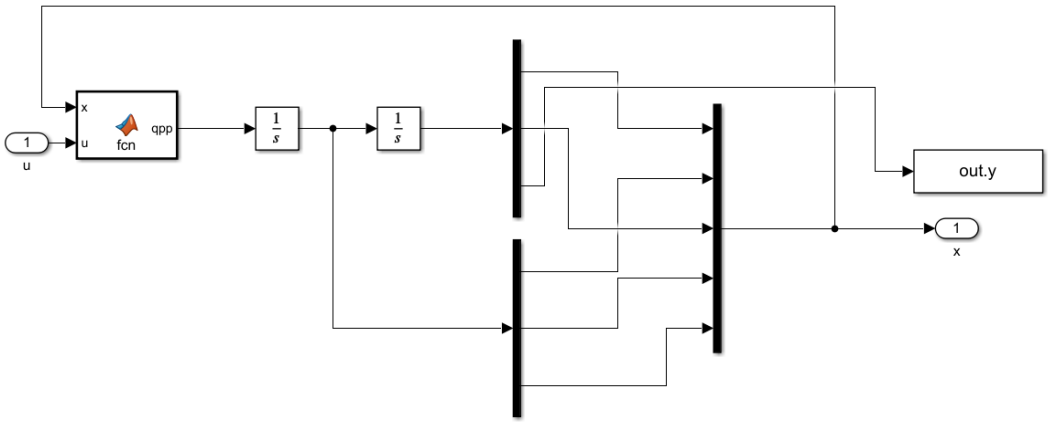


Fonte: elaborado pelo autor (2025)

Nessa configuração há apenas um regulador de estados implementado, visto que a primeira iteração serviu para validar os parâmetros escolhidos para o motor.

O bloco “Modelo Biciclo” contém uma MATLAB *function* que calcula os estados utilizando as matrizes M, G, P e V, mencionadas no capítulo anterior, e retorna o vetor de estados desejados.

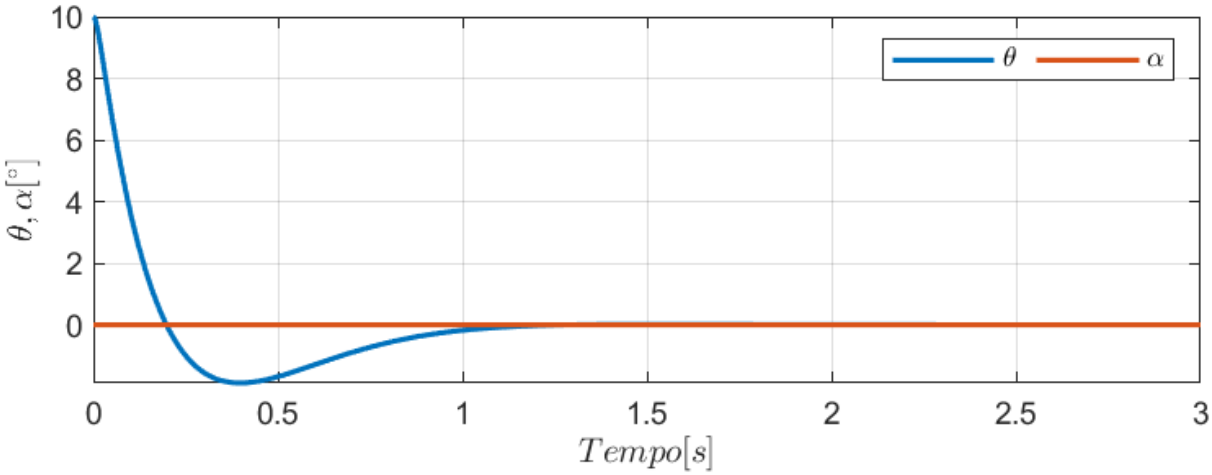
Figura 15 -Simulador do biciclo



Fonte: elaborado pelo autor (2025)

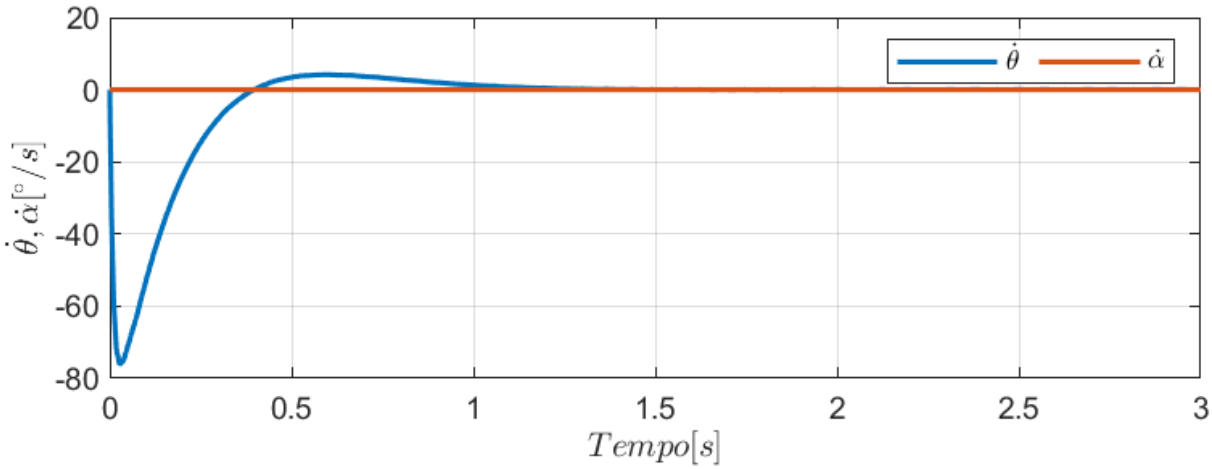
Foram simuladas a condição de inclinação do robô de 10°. A seguir, o comportamento dos estados de interesse sob essa condição.

Figura 16 -Arfagem iniciando em 10° e guinada constante



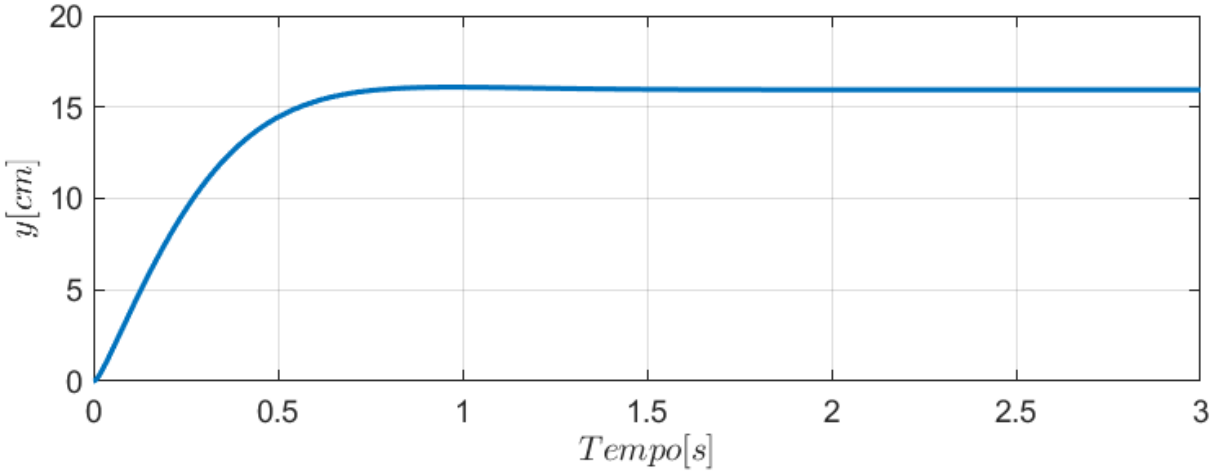
Fonte: elaborado pelo autor (2025)

Figura 17 -Velocidades de arfagem e guinada iniciando para ângulo de 30° de arfagem



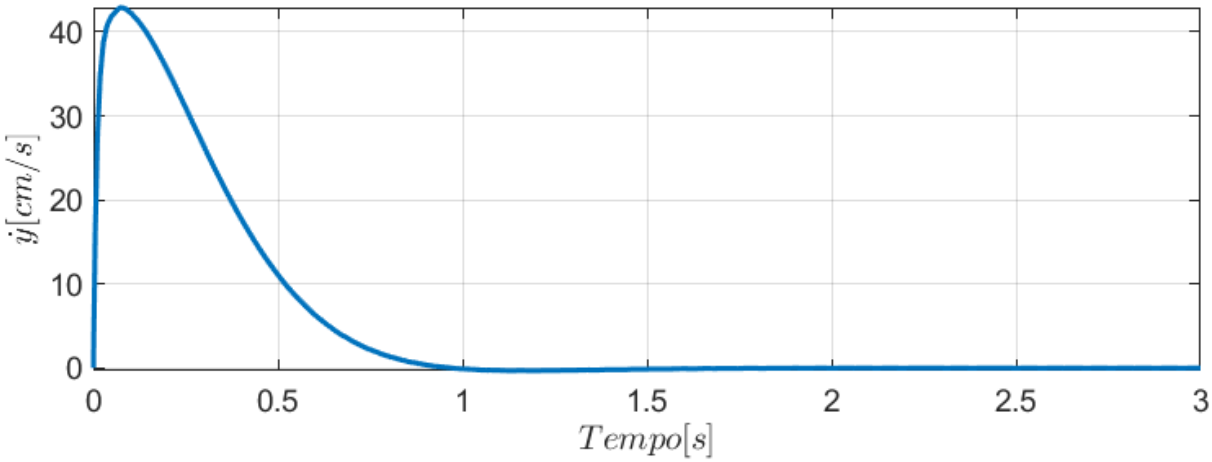
Fonte: elaborado pelo autor (2025)

Figura 18-Deslocamento linear do biciclo devido arfagem de 10°



Fonte: elaborado pelo autor (2025)

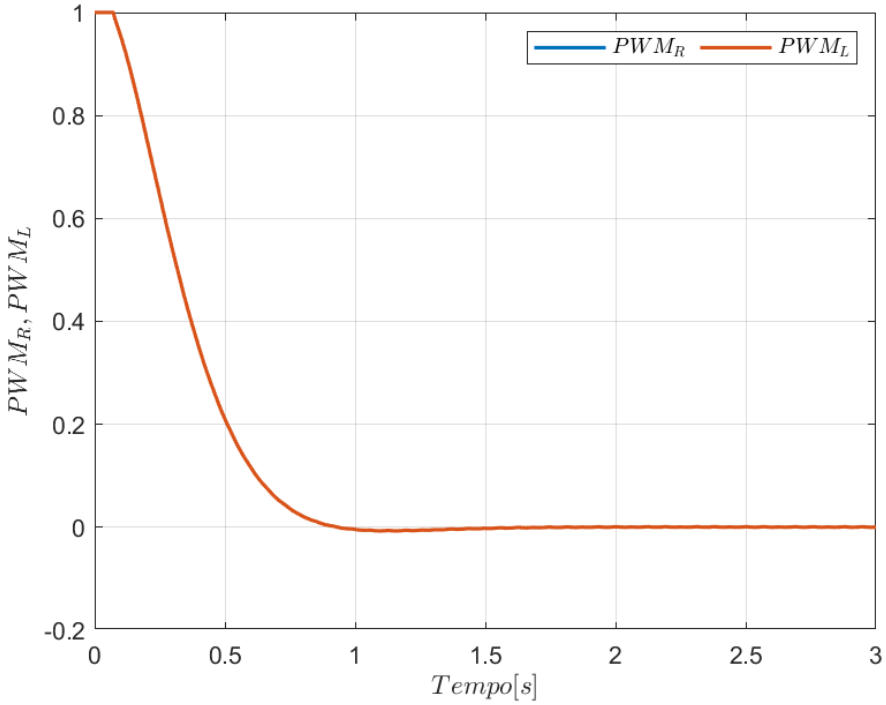
Figura 19 -Velocidade linear do biciclo devido arfagem de 10°



Fonte: elaborado pelo autor (2025)

Como pode ser visto no gráfico a seguir, nessa situação houve ligeira saturação dos atuadores.

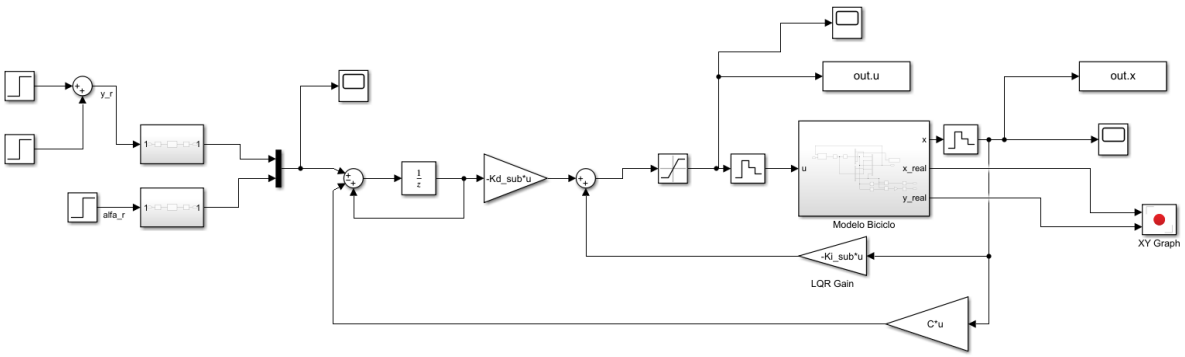
Figura 20 -Esforço de controle dos motores



Fonte: elaborado pelo autor (2025)

Posteriormente, foi adicionado ao simulador as referências para que o robô se deslocasse no espaço. Para isso, expandiu-se o sistema adicionando dois novos estados: o de erro de velocidade angular e o de erro de velocidade linear do centro de massa.

Figura 21 -Simulador com referência



Fonte: elaborado pelo autor (2025)

Com isso, as matrizes aumentadas A, B e C foram as seguintes:

```
A_aug =

    1.0054    0.0076         0         0    0.0759         0         0
    0.9220    0.6582         0         0   10.6842         0         0
         0         0    1.0000    0.0051         0         0         0
         0         0         0    0.2197         0         0         0
   -0.0052    0.0194         0         0    0.4022         0         0
         0         0   -1.0000         0         0    1.0000         0
         0         0         0         0   -1.0000         0    1.0000
```

```
B_aug =

   -0.0140   -0.0140
   -1.9696   -1.9696
   -0.0098    0.0098
   -1.5740    1.5740
    0.1102    0.1102
         0         0
         0         0
```

```
C_aug =

     0     0     1     0     0     0     0
     0     0     0     0     1     0     0
```

Após os testes, percebeu-se que um grande esforço do controlador causava um pico de corrente e uma queda de tensão suficiente para reiniciar o microcontrolador, de forma que foi necessário ponderar as matrizes Q e R para evitar um controle agressivo.

```
Q =

   1.0e+04 *

    0.0015         0         0         0         0         0         0
         0    0.0000         0         0         0         0         0
         0         0    0.0004         0         0         0         0
         0         0         0    0.0000         0         0         0
         0         0         0         0    5.0000         0         0
         0         0         0         0         0    0.0000         0
         0         0         0         0         0         0    0.0000
```

$$R = 1e6 * diag([1\ 1])$$

Essa configuração gerou um vetor de ganhos capaz de estabilizar o sistema, rejeitar distúrbios e seguir referências.

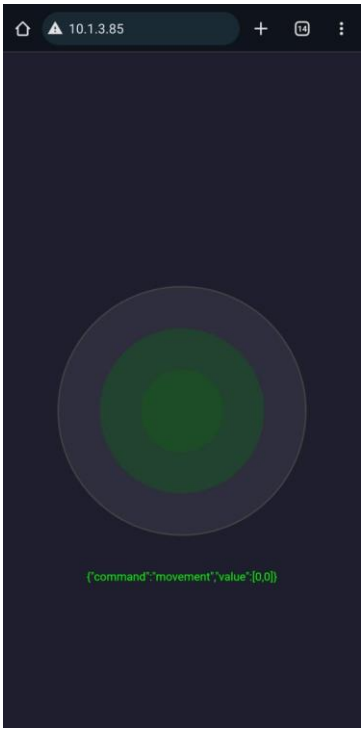
```
Kd =

   -3.5443   -0.3058   -0.0593   -0.0004   -5.2384    0.0001    0.0002
   -3.5443   -0.3058    0.0593    0.0004   -5.2384   -0.0001    0.0002
```

8. Controle Embarcado Discreto

Com o vetor de ganhos definido, partiu-se para o desenvolvimento do controle embarcado usando a ESP-IDF através da extensão do Visual Studio Code. Para isso, criou-se um código em C++, com três componentes: o primeiro para configuração da interface I2C para a aquisição de telemetrias vindas da IMU e usando a biblioteca da Espressif para uso do pulse counter para os encoders dos motores. O segundo componente para controle dos motores e leitura da velocidade angular e o terceiro para comunicação, além de uma partição para o código em html e javascript da página que envia setpoints de velocidade para o controle.

Figura 22 -Página web da ESP32 principal com joystick para movimentar o robô



Fonte:acervo pessoal do autor (2025)

O sistema utilizou o framework FreeRTOS, que é um agendador de tarefas, e contou com tasks de aquisição de dados dos sensores (imu e encoders), do servidor e uma para controle dos motores, todas rodando nos dois núcleos do microcontrolador.

A calibração da IMU foi feita colhendo 2000 amostras brutas de aceleração e velocidade angular vindas dos acelerômetros e giroscópios, de cada um dos seis eixos, calculando a média e salvando os valores na memória flash do dispositivo, de forma que a calibração é necessária apenas uma vez ou sempre que a memória flash

seja apagada. Esses valores brutos serão removidos das leituras antes de passar pelo filtro de Kalman. Após o filtro, verificou-se na comunicação serial o comportamento dos ângulos, verificando se os ruídos haviam sido estabilizados e se as medições estavam coerentes, ou seja, ângulos notáveis coerentes e sinais dos ângulos e velocidades corretos.

Para os encoders, calculou-se, a tensão necessária para uma revolução por segundo no eixo de saída do motor. Em cada canal são 11 pulsos por rotação e a redução é de 46:1, como especificado pelo fabricante. Então aplicou-se o valor percentual de duty cycle do PWM necessário, considerando a tensão da bateria e verificou-se via serial a velocidade de 6.28 rad/s na frequência de amostragem definida pelo controle, como esperado.

Com isso, criou-se o vetor de ganhos que serão usados no produto matricial $u = -Kx$. Da IMU, os estados de pitch e pitch rate foram obtidos após passarem por uma fusão de sensores do filtro de Kalman. Dos encoders, o estado de yaw, yaw rate e a velocidade linear na direção do eixo y, usando as equações discretizadas da seção 6. Os estados de erro são obtidos a partir das velocidades angular e linear desejadas e obtidas. Por fim, calcula-se o esforço de controle ‘u’ a partir do produto mencionado acima e os aplica aos atuadores a uma taxa de 100 Hz, como discretizado no modelo.

9. Considerações finais

Com a simulação, a modelagem, a montagem mecânica, a confecção da placa de circuito, a calibração dos sensores e a criação do controle embarcado discreto com a aquisição de *setpoint* via *webpage* gerada pelo dispositivo conectado à mesma rede wifi do dispositivo que o controla, foi possível iniciar os testes de estabilidade e controle.

Inicialmente, o controle estava recebendo sinais de ângulo e velocidade angular de pitch diferentes aos convencionados na modelagem, por conta da orientação do circuito integrado na placa, então conferiu-se a convenção de sinais, inclusive das velocidades das rodas, e alterou-se os sinais para manter a mesma convenção da modelagem.

Após isso, diversos testes foram realizados e à medida que aumentava-se a influência da inclinação no vetor de ganhos do controlador, mais oscilante e agressivo o robô ficava, enquanto quanto menor, mais o biciclo se deslocava e caía por acelerar até atingir um limite de velocidade que a bateria permitia. No entanto, ao aumentar a influência da velocidade angular, o robô suavizou as oscilações.

Ao adicionar os integradores, o robô conseguiu girar em torno do próprio eixo e se deslocar no sentido do eixo y.

A estabilidade, no entanto, estava com pouca margem para rejeição de distúrbios, de forma que após alguns instantes em torno da posição de equilíbrio, acabava por se desestabilizar ao encostar em alguma parede ou por acelerar demais buscando equilíbrio.

Além disso, percebeu-se que ao se instalar a câmera com o suporte o conjunto deslocava o centro de massa para frente, de modo que apenas realizando a calibração do robô de forma inclinada que foi possível certa estabilidade. No entanto, esse deslocamento foi suficiente para que a estabilidade não fosse obtida com os ganhos antes verificados. Além disso, recalculou-se a matriz de inércia, a altura do centro de massa e o vetor de ganhos.

Com as iterações, a estabilidade foi obtida, no entanto, com menor range que no teste sem o conjunto suporte-câmeras.

O sistema respondia mais lentamente e estabilizava com tensões mais baixas, como a tensão vinda de uma porta USB (5V), mas na bateria, quando baixava a tensão, por conta do uso ou de um pico de corrente, para algo em torno de 7,5 V, o

regulador de tensão apresentava ruído excessivo e fazia com que o microcontrolador reiniciasse ou entrasse em estados com comportamento incerto. Uma fonte foi usada para fornecer tensão e corrente mais estáveis, mas essa prática acabou danificando permanentemente o regulador da placa, interrompendo os testes. O desempenho foi registrado em vídeo.

O conjunto do suporte e da câmera foi desenvolvido separadamente usando a Arduino IDE. Uma página *web* é gerada onde a transmissão da imagem pelo microcontrolador é exibida. O *joystick* responsável pela movimentação dos servos envia setpoint de ângulos para cada um dos servos.

Figura 23 -Página web da ESP32-S3 com joystick para os servos

Fonte: acervo pessoal do autor (2025)

10. Trabalhos Futuros

Para trabalhos futuros, seria válido refazer a o circuito de auto boot, o de carregamento das baterias, substituir o diodo de proteção da porta USB e adicionar proteções para o regulador de 5V e um circuito que monitore a tensão da bateria. Seria interessante desenvolver uma placa de circuito para o microcontrolador da câmera e para conectar os servos. Para o controlador, seria interessante adicionar um integrador de posição para que a planta se incline naturalmente em resposta aos deslocamentos do centro de massa devido a câmera. Além disso, novos testes com mais vetores de ganhos ou a substituição em tempo real dos vetores de ganhos. Por fim, a integração entre câmera com os servos e o controle de estabilidade, de forma que uma única página possa visualizar as imagens transmitidas pelo microcontrolador e dois joysticks, um para a movimentação do robô e o outro para a dos servos, além de um algoritmo para algumas funções autônomas, processando as imagens no próprio dispositivo.

Referências

DIAS,S. Controle Adaptativo Robusto para um Modelo Desacoplado de um Robô Móvel. 2010. **Tese de Doutorado** – Programa de Pós-Graduação em Engenharia Elétrica e de Computação. Universidade Federal do Rio Grande do Norte, Natal, 2010.

GOES, S.; NEVES, G; ANGÉLICO, B. **Robô de duas rodas autoequilibrando seguidor de linha**. 2018. Escola Politécnica da Universidade de São Paulo e Departamento de Engenharia de Telecomunicações e Controle, São Paulo, 2018.

NEVES, G. Modeling, construction and control of a self-balancing unicycle. 2017. **Tese de Mestrado** - Escola Politécnica da Universidade de São Paulo, São Paulo, 2017.

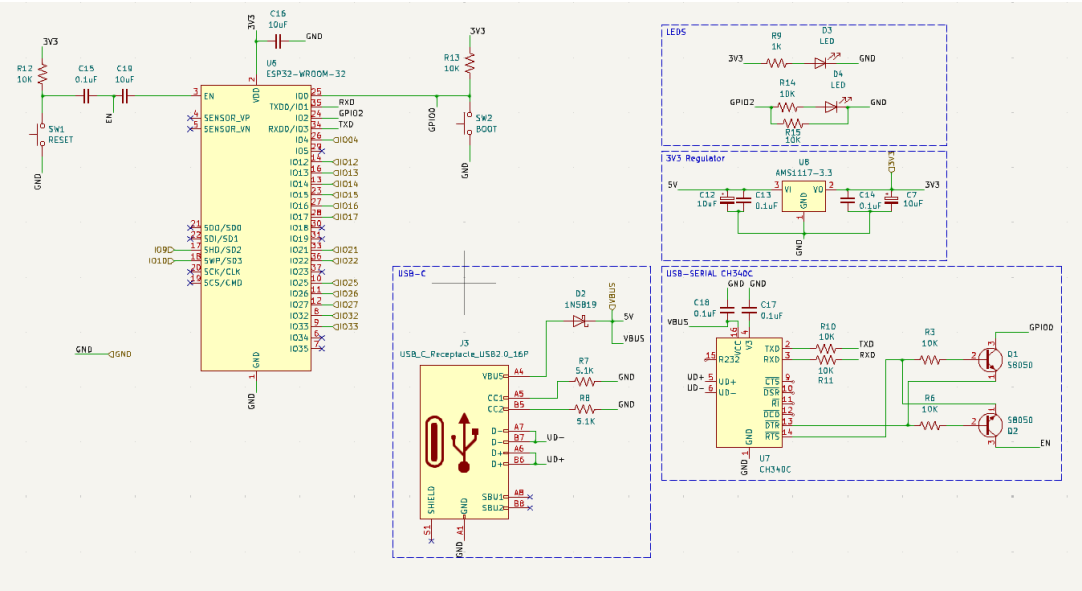
ROZENFELD, H.; FORCELLINI, F. A.; AMARAL, D. C.; et al. **Gestão de Desenvolvimento de Produto**: uma referência para a melhoria do processo. 1 ed. São Paulo: Saraiva, 2006.

VEEN, J. **Stabilization and Trajectory Tracking of a Segway. 2018**. Bachelor Integration Project IE&M. University of Groningen, Países Baixos, Groningen, 2018.

Apêndice A - Esquemáticos

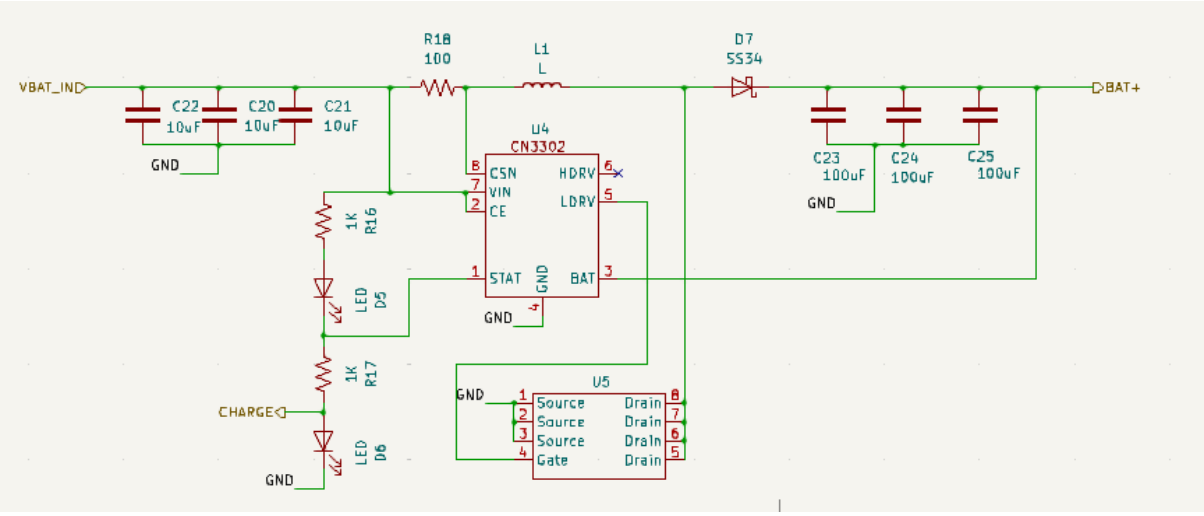
A seguir, esquemáticos da placa, código do Matlab e repositório do Github.

Figura 24 - Esquemático da ESP-32



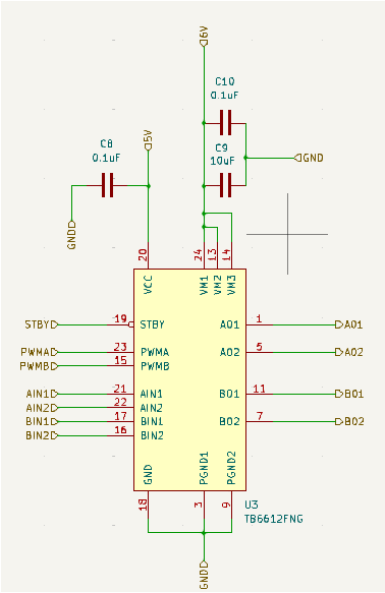
Fonte: elaborado pelo autor (2025)

Figura 25 - Esquemático do carregador 2S



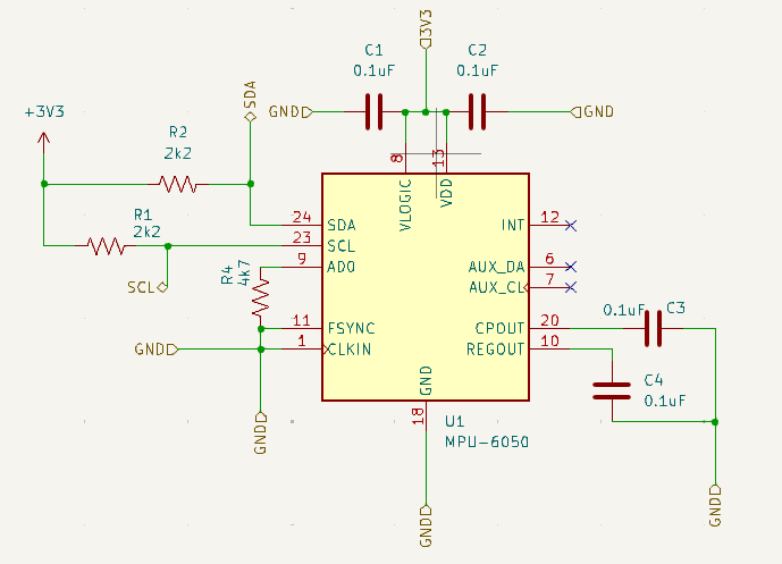
Fonte: elaborado pelo autor (2025)

Figura 26 - Esquemático do driver do motor



Fonte: elaborado pelo autor (2025)

Figura 27 - Esquemático da IMU



Fonte: elaborado pelo autor (2025)

Apêndice B - Código Matlab

A seguir, código do Matlab.

```
%% Modelo biciclo
clear all; close all; clc;

%% Definir parametros
syms thP(t) thR(t) thL(t)
syms D R L mr mp Ix Iy Iz Jr
syms Kt Kv Rm
syms g

% valores numericos
D = 0.18278;  R = 0.065/2;  L = 0.03625;
mr = 0.038; mp = 0.534;
Ix =   0.00096277;  Iy = 0.00210613;  Iz =  0.00129223; Jr = 0.00002786;
Kt = 3.3e-2*9.81/1.8; % T = Kt*i
Kv = 6/(2*pi*130/60); %
Rm = 2.6;

thP_p = diff(thP);
thP_pp = diff(thP_p);
thR_p = diff(thR);
thR_pp = diff(thR_p);
thL_p = diff(thL);
thL_pp = diff(thL_p);

Ip = diag([Ix; Iy; Iz]);

%% Definir as velocidade
% Rot = [1 0 0; 0 cos(thP) -sin(thP); 0 sin(thP) cos(thP)];
Rot = [1 0 0; 0 sin(thP) cos(thP) ; 0 -cos(thP) sin(thP)];
w0 = [-thP_p; 0; R/D*(-thR_p+thL_p)];
v0 = [0; R/2*(thR_p+thL_p); 0];

wcm = Rot*w0;
vcm = v0+cross(w0,[0;L*sin(thP);L*cos(thP)]);

%% Energias
% energia potencia
U = mp*g*L*cos(thP);
```

```
% energia das rodas
KR = 1/2*thR_p^2*(Jr+mr*R^2);
KL = 1/2*thL_p^2*(Jr+mr*R^2);

% Energia do corpo
KP = 1/2*wcm.'*Ip*wcm+ 1/2*vcm.'*mp*vcm;

% Lagrangiano:
Lagr = simplify(expand(KP+KR+KL-U));

%% Metodo de Lagrange

% pitch
LagrP = diff(diff(Lagr,thP_p))-diff(Lagr,thP);

% Roda direita
LagrR = diff(diff(Lagr,thR_p))-diff(Lagr,thR);

% Roda esquerda
LagrL = diff(diff(Lagr,thL_p))-diff(Lagr,thL);


% Substituição de variaveis
syms thP_ thP_d thP_dd
syms thR_ thR_d thR_dd
syms thL_ thL_d thL_dd

LagrP_ = subs(LagrP,thP_pp,thP_dd);
LagrP_ = subs(LagrP_,thP_p,thP_d);
LagrP_ = subs(LagrP_,thP,thP_);
LagrP_ = subs(LagrP_,thR_pp,thR_dd);
LagrP_ = subs(LagrP_,thR_p,thR_d);
LagrP_ = subs(LagrP_,thR,thR_);
LagrP_ = subs(LagrP_,thL_pp,thL_dd);
LagrP_ = subs(LagrP_,thL_p,thL_d);
LagrP_ = subs(LagrP_,thL,thL_);

LagrR_ = subs(LagrR,thP_pp,thP_dd);
LagrR_ = subs(LagrR_,thP_p,thP_d);
LagrR_ = subs(LagrR_,thP,thP_);
LagrR_ = subs(LagrR_,thR_pp,thR_dd);
LagrR_ = subs(LagrR_,thR_p,thR_d);
LagrR_ = subs(LagrR_,thR,thR_);
```

```
LagrR_ = subs(LagrR_,thL_pp,thL_dd);
LagrR_ = subs(LagrR_,thL_p,thL_d);
LagrR_ = subs(LagrR_,thL,thL_);

LagrL_ = subs(LagrL,thP_pp,thP_dd);
LagrL_ = subs(LagrL_,thP_p,thP_d);
LagrL_ = subs(LagrL_,thP,thP_);
LagrL_ = subs(LagrL_,thR_pp,thR_dd);
LagrL_ = subs(LagrL_,thR_p,thR_d);
LagrL_ = subs(LagrL_,thR,thR_);
LagrL_ = subs(LagrL_,thL_pp,thL_dd);
LagrL_ = subs(LagrL_,thL_p,thL_d);
LagrL_ = subs(LagrL_,thL,thL_);

% Torque (VR e VL variam -1 a 1)
syms VR VL
tauR = (Kt/Rm)*(5*VR-Kv*(thR_d-thP_d));
tauL = (Kt/Rm)*(5*VL-Kv*(thL_d-thP_d));

% Resolver eq.
s1 = simplify(expand(LagrP_+(tauR+tauL)));
s2 = simplify(expand(LagrR_-tauR));
s3 = simplify(expand(LagrL_-tauL));

% mudança de coordenada
syms y_ y_d y_dd
syms alpha_ alpha_d alpha_dd
thR_ = y_/R-D/(2*R)*alpha_;
thL_ = y_/R+D/(2*R)*alpha_;
thR_d = y_d/R-D/(2*R)*alpha_d;
thL_d = y_d/R+D/(2*R)*alpha_d;
thR_dd = y_dd/R-D/(2*R)*alpha_dd;
thL_dd = y_dd/R+D/(2*R)*alpha_dd;
s1 = simplify(expand(subs(s1)));
s2 = simplify(expand(subs(s2)));
s3 = simplify(expand(subs(s3)));

%variaveis de estado
q_dd = [thP_dd; alpha_dd; y_dd];
q_d = [thP_d;alpha_d; y_d];
x = [thP_; thP_d; alpha_; alpha_d; y_d];
u = [VR;VL];

M=simplify(expand(jacobian([s1; s2; s3],q_dd)));
```

```
G=simplify(expand(jacobian([s1; s2; s3],g)*g));
P=simplify(expand(-jacobian([s1; s2; s3],u)));
V=simplify(expand([s1; s2; s3]-M*q_dd-G+P*u));
g=9.81; G = subs(G);
M = M(0);
G = G(0);
V = V(0);
P = P(0);
matlabFunction(M,'File','function_M','Vars',{thP_});
matlabFunction(V,'File','function_V','Vars',{alpha_d, thP_, thP_d, y_d});
matlabFunction(G,'File','function_G','Vars',{thP_});
matlabFunction(P,'File','function_P');



%% Modelo Linear
xpp = simplify(expand(M^-1*(P*u-V-G)));
xpp = [thP_d; xpp(1); alpha_d; xpp(2); xpp(3)];
A = jacobian(xpp,x);
B = jacobian(xpp,u);

A = double(subs(A,[alpha_d, thP_, thP_d, y_d, VR, VL],[0,0,0,0,0,0]));
B = double(subs(B,[alpha_d, thP_, thP_d, y_d, VR, VL],[0,0,0,0,0,0]));

%% Controle Simples
Q = diag([1;3;1;1;1]);
R = eye(2);
K = lqr(A,B,Q,R);

%% Controle Discreto
Ts = 0.01;  % Tempo de amostragem
C = [0 0 1 0 0;0 0 0 0 1];

% Criando a matriz identidade I_2x2
I_2 = eye(2);

% Criando o sistema no espaço de estados
G = ss(A, B, C, zeros(2,2));

% Discretizando o sistema usando Zero-Order Hold (ZOH)
G_d = c2d(G, Ts, 'zoh');

% Construindo as matrizes aumentadas
A_aug = [G_d.A zeros(5,2); -G_d.C I_2];  % Matriz A aumentada (7x7)
```

```
B_aug = [G_d.B; zeros(2,2)];          % Matriz B aumentada (7x2)
C_aug = [G_d.C, zeros(2,2)];        % Matriz C aumentada (2x7)
D = 0;                        % Matriz D (normalmente 0)

teta = 1/(15*pi/180)^2;
alfa = 1/(30*pi/180)^2;

Q = diag([1*teta 1*0.1^2/teta^2 alfa 0.01^2/alfa^2 5/(0.01)^2 1e-2 1e-1]);
%Q = diag([100 1 0.1 0.01 10 1e-3 1e-2]);
R = 1e6*diag([1 1]);
Kd = dlqr(A_aug,B_aug, Q, R);
fprintf("{\n");
for i = 1:size(Kd, 1)
    fprintf("    { ");
    for j = 1:size(Kd, 2)
        fprintf("%.15f", Kd(i, j));
        if j < size(Kd, 2)
            fprintf(",  ");
        end
    end
    if i < size(Kd, 1)
        fprintf(" }, // u_vr\n");
    else
        fprintf(" }  // u_vl\n");
    end
end
fprintf("};\n");
Kd_sub = Kd(end-1:end, end-1:end);
Ki_sub = Kd(:, 1:5);

%% Simulação
trun = 3;
v_ic = [0;0;0];
x_ic = [20*pi/180;0*pi/180;0];

% simulacao = sim('Sim_Biciclo.slx');
```

A seguir, repositório Github
Link para repositório.