MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

GABRIEL MARTINS LIMA

IMPLEMENTAÇÃO DIGITAL DE CONTROLADORES LINEARES PARA SVTRP

RIO DE JANEIRO
2023

GABRIEL MARTINS LIMA

IMPLEMENTAÇÃO DIGITAL DE CONTROLADORES LINEARES PARA SVTRP

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(es): Alberto Mota Simões, Dr.ISAE

Rio de Janeiro

2023

©2023
INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

Lima, Gabriel Martins.
Implementação Digital de Controladores Lineares para SVTRP / Gabriel Martins Lima. – Rio de Janeiro, 2023.
59 f.

Orientador(es): Alberto Mota Simões.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia Eletrônica, 2023.

1. robô. 2. ssl. 3. sistemas de controle. 4. sistemas embarcados. 5. svtrp. 6. microcontroladores. i. Mota Simões, Alberto (orient.) ii. Título

GABRIEL MARTINS LIMA

Implementação Digital de Controladores Lineares para SVTRP

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(es): Alberto Mota Simões.

Aprovada em 11 de outubro de 2023, pela seguinte banca examinadora:

Prof. Alberto Mota **Simões** - Dr. ISAE do IME - Presidente

Prof. Hebert **Azevedo** Sá - Ph .D. do IME

Prof. Antonio Eduardo **Carrilho** da Cunha - Dr. Eng. do IME

Rio de Janeiro

2023

Este trabalho é dedicado a todos os que me acompanharam, me incentivaram e acreditaram no meu potencial.

AGRADECIMENTOS

Gostaria de primeiramente de agradecer a Deus, por ter me trazido até aqui e me dado a oportunidade de realizar o sonho desta conquista. Sem Ele, nunca teria chegado onde cheguei.

Agradeço também à minha família, sobretudo meus pais e avós, por terem me guiado ao longo destes vinte e seis anos, e por terem apoiado cada uma das minhas escolhas que me trouxeram a este momento. O amor de todos foi fundamental para me trazer a força e as feramentas necessárias para superar todos os empecilhos e dificuldades desta trajetória.

Também só tenho a agradecer aos amigos que conquistei ao longo deste caminho, por terem me dado apoio nos meus momentos de maior dificuldade e por terem acreditado no potencial que eu tinha de chegar até aqui. Vocês também me trouxeram muitos momentos de felicidade e de alegria, além de terem me dado propósitos e motivos para querer sempre seguir em frente e realizar sonhos cada vez maiores.

Aos meus amigos, colegas e professores da RoboIME, por terem me trazido lindas experiências e um grande propósito para os meus cinco anos de IME, e por terem sempre me incentivado a me tornar um engenheiro cada vez melhor. Agradeço também por me despertarem a paixão pela robótica, e por me dar a imensa vontade de desenvolver este projeto.

Aos meus professores, pelo conhecimento que me trouxeram e pelo caminho que traçaram para meu sucesso como um excelente engenheiro eletrônico. Agradeço também por estarem sempre dispostos a tirar minhas dúvidas e me ajudarem nos meus momentos acadêmicos de maior dificuldade.

Aos meus orientadores, prof. Simões, prof. Carrilho, prof. Azevedo e prof. Rotava. Suas ajudas foram de extrema importância para que este projeto fosse desenvolvido e atingisse os resultados que atingiu. Seus apoios serão objetos da minha lembrança pelo resto da vida.

Gostaria também de dar agradecimentos especiais à Clarisse, à Iasmin e à Lorena. Vocês foram peças chave nos meus momentos de maior dificuldade ao longo destes anos. É graças ao apoio de vocês que eu tive forças para superar os maiores obstáculos que foram colocados no meu caminho. Sintam esta conquista como sendo também de vocês.

E, por fim, a todos os que sempre torceram pelo meu melhor e acreditaram que um dia eu chegaria a ter uma conquista de tamanha importância. Muito obrigado, de verdade!

Sometimes war is killing,
Sometimes it's saving lives.
(Sabaton, Ballad of Bull)

RESUMO

Em plataformas robóticas móveis com motores DC controlados por controle PID, é comum que mudanças repentinas de velocidade desejada provoquem deslizamento, o que causa perda de trajetória. Desta forma, este trabalho apresenta uma solução para detecção das condições de deslizamento, além de medidas corretivas que minimizam a perda de trajetória. A solução consiste em comparação da aceleração da roda com a aceleração do chassi e, sendo detectada inconsistência, o torque do motor é reduzido a fim de retomar a aderência. O projeto foi validado medindo-se o desvio de trajetória de um robô com rodas de diferentes coeficientes de atrito.

Palavras-chave: robô. ssl. sistemas de controle. sistemas embarcados. svtrp. microcontroladores.

ABSTRACT

In mobile robotic platforms powered by DC motors controlled by PID, it's common that sudden changes to desired speed cause slipping, which leads to trajectory loss. This project presents a solution to detect slip conditions and presents corrective measures that minimize trajectory loss. The solution consists in comparing the wheel acceleration with the chassis acceleration and, if inconsistencies are detected, the motor torque is reduced in order to regain the grip. The project was validated by measuring the trajectory deviation of a robot with wheels with different friction coefficients.

Keywords: robot. ssl. control systems. embedded systems. unmanned vehicles. microcontrollers.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SVTRP	Sistema de Viaturas Terrestres Remotamente Pilotadas
CTEx	Centro Tecnológico do Exército
SSL	Small Size League
IMU	Inertial Measurement Unit
dof	Degrees of freedom
GPS	Global Positioning System
PID	Proportional integral derivative
SLAM	Simultaneous location and mapping
LiPo	Lithium polymer
CPR	Contagens por rotação
DC	Direct current
CAD	Computer aided design
ADC	Analog to digital converter
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
LTI	Linear time invariant
USB	Universal Serial Bus
PWM	Pulse width modulation
ISR	Interrupt service routine
RAM	Random access memory
HAL	Hardware abstraction layer
SPI	Serial Peripheral Interface

LISTA DE SÍMBOLOS

V	Volt
m	Metro
s	segundo
Hz	Hertz
bps	bits por segundo
Ah	Ampére-hora
τ	Tau
K_t	Constante mecânica
i	Corrente elétrica
ϵ	Epsilon
K_e	Constante elétrica
θ_m	Posição angular
V_a	Tensão elétrica
R	Resistência elétrica
L	Indutância
b	Constante de atrito viscoso
K_p	Ganho da planta
T_{p1}	Constante de tempo da planta
$dutyCycle$	Ciclo de trabalho do PWM
kp	Constante proporcional do PID
ki	Constante integral do PID
kd	Constante derivativa do PID
$W(t)$	Perturbação mecânica
$V(t)$	Perturbação do sensor

$dError$	Derivada do erro
$deltaT$	Intervalo de tempo
$v0$	Velocidade inicial
vf	Velocidade final
$desiredSpeed$	Velocidade de referência
$threshold$	Limiar

SUMÁRIO

1 INTRODUÇÃO

O projeto de um sistema de viaturas terrestres remotamente pilotadas (SVTRP) é um dos projetos estratégicos desenvolvidos pelo Centro Tecnológico do Exército (CTEx) para o exército brasileiro, que tem por objetivo construir um pequeno veículo para vigilância e mapeamento de ambientes hostis e desconhecidos. Nesse sentido, plataformas robóticas móveis terrestres, como o SVTRP do CTEx, requerem que a velocidade e aceleração das rodas sejam controladas com precisão, a fim de permitir um movimento preciso e facilitar a pilotagem pelo operador. Além disso, um controle preciso das velocidades e acelerações permite o funcionamento autônomo do robô, o que é essencial em ambientes onde a comunicação sem fio não é possível, ou que seu uso traria consequências negativas estratégicas.

Dessa forma, uma maneira de garantir a precisão do movimento do robô é pelo uso de um controle de malha fechada para a velocidade das rodas, o que aproxima a velocidade de cada roda de uma velocidade de referência fornecida pelo operador ou pelo algoritmo de operação autônoma, o que garante que a trajetória do robô seja próxima àquela desejada. Por outro lado, um problema muito comum enfrentado por robôs controlados por malha fechada é a perda de direção nos casos de aceleração e frenagem quando uma das rodas perde a tração, causando uma derrapagem. Neste caso, ocorrerá uma perda de direção que não é tratada pelos sistemas de controle em malha fechada convencionais, sendo necessário um sistema de controle específico para evitar e corrigir as condições de derrapagem.

Este projeto do CTEx é bastante semelhante aos robôs desenvolvidos pela equipe RoboIME para a competição RoboCup Small Size League (SSL), já que ambos utilizam motores DC com escovas para movimentação, utilizam microcontroladores da mesma família, circuitos de potência e alimentação similares e são de tamanhos e massas semelhantes. Sendo assim, propõe-se desenvolver uma metodologia de desenvolvimento e projeto de sistema de controle linear que possa ser aplicada em ambas as plataformas robóticas.

Este documento apresenta no capítulo 2 as características, especificações e limitações dos robôs a terem seus sistemas de controle desenvolvidos. Em seguida, o capítulo 3 mostra a teoria envolvida nos sistemas de controle, bem como as peculiaridades de sua implementação no *hardware* dos robôs. O capítulo 4 descreve os problemas enfrentados pelos robôs com sistemas de controle convencionais, bem como descreve melhor os objetivos deste trabalho. Posteriormente, o capítulo 5 mostra as soluções propostas para os problemas de deslizamento, além de uma metodologia para desenvolvimento e calibração do sistema de controle convencional melhor do que a utilizada atualmente. Por fim, no capítulo 6 são discutidos os resultados obtidos com a aplicação do sistema de detecção de deslizamentos bem como com as medidas corretivas.

1.1 Objetivos do trabalho

1.1.1 Objetivo geral

O presente trabalho de fim de curso tem por objetivo desenvolver uma metodologia para projeto de sistemas de controle para ambos os robôs capazes de resolver os problemas de perda de direção relacionados à perda de tração. Todo o material desenvolvido neste trabalho encontra-se no repositório em (3)

1.1.2 Objetivos específicos

- Análise de ambos os modelos de robô citados
- Construção de um robô para desenvolvimento
- Projeto e calibração de sistema de controle convencional
- Desenvolvimento de algoritmo para detecção de deslizamento
- Desenvolvimento de medida corretiva para casos de deslizamento
- Implementação no *hardware*

2 CONTEXTO

2.1 Projeto SVTRP

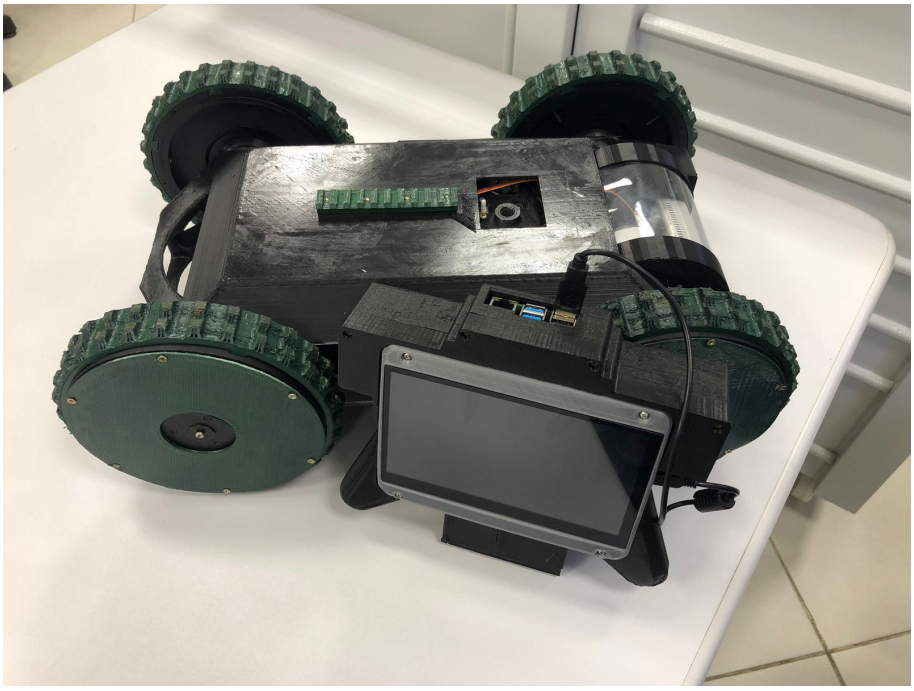

Figura 1 – Primeiro protótipo de SVTRP do CTEx

Atualmente, já foi desenvolvido pelo CTEx o primeiro protótipo da plataforma de SVTRP, representado na figura 1, e o segundo protótipo ainda está em desenvolvimento. Dessa forma, foi decidido tomar o primeiro protótipo como base para o desenvolvimento do presente projeto de fim de curso.

A plataforma SVTRP do CTEx possui quatro rodas paralelas fixas, tornando-se, assim, uma plataforma de movimento diferencial. Todos os motores são alimentados por pontes H e possuem *encoder* para medição de posição angular relativa, o que possibilita o controle de malha fechada da velocidade. Possui também um sensor inercial (IMU) de seis graus de liberdade (6-dof) contendo acelerômetro e giroscópio, mas que não contém bússola. Portanto, não é capaz de determinar sua orientação absoluta. Possui também um receptor GPS para determinação da posição global com precisão de alguns metros.

O microcontrolador responsável pelo controle de velocidade e também pela comunicação com os sensores é o STM32F103 "BluePill". É um modelo barato e facilmente encontrado no mercado nacional, porém de capacidade computacional suficiente para controle PID em malha fechada de motores com escovas e para comunicação com os sensores.

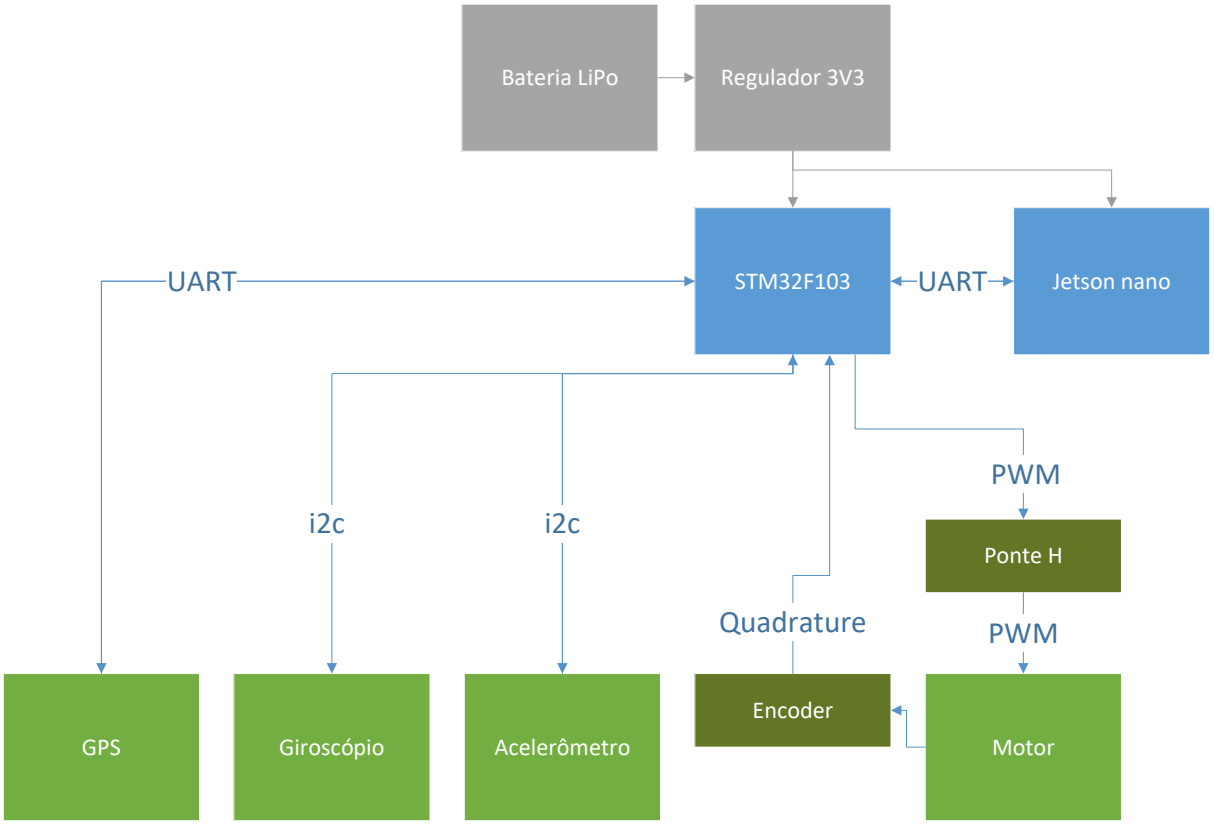

Figura 2 – Diagrama de blocos do SVTRP do CTEx

Além disso, o SVTRP do CTEx embarca um computador Jetson nano, responsável pela captação de imagens com as câmeras e pelo processamento do algoritmo de *Simultaneous Location and Mapping* (SLAM).

Todo o conjunto é alimentado por uma bateria de polímero de lítio (LiPo), com tensão regulada por um regulador chaveado de 3.3V.

As especificações da plataforma do CTEx podem ser vistas na tabela 1 e seu diagrama de blocos na figura 2.

Tabela 1 – Especificações do primeiro protótipo SVTRP

Componente	Especificação
Computador	Nvidia Jetson Nano
Microcontrolador	STMicroelectronics STM32F103C8
Ponte H	STMicroelectronics VNH2SP30
Sensor IMU	InvenSense MPU6050
Sensor GPS	u-blox NEO-6M-0-001
Motores	Pololu 37Dx68L

2.2 Robôs para a competição SSL da RoboCup

Devido às semelhanças mecânica e eletrônica entre os robôs de futebol de robôs categoria *Small Size League* (SSL) do IME, representado na figura 3 e a plataforma de SVTRP do CTEx, e aos problemas semelhantes enfrentados por ambos, foi levantada a possibilidade de desenvolver uma solução que atenda ambas as plataformas robóticas. Assim, tanto os interesses da equipe de robótica do IME (RoboIME), quanto do CTEx, poderão ser atendidos pelo mesmo projeto.

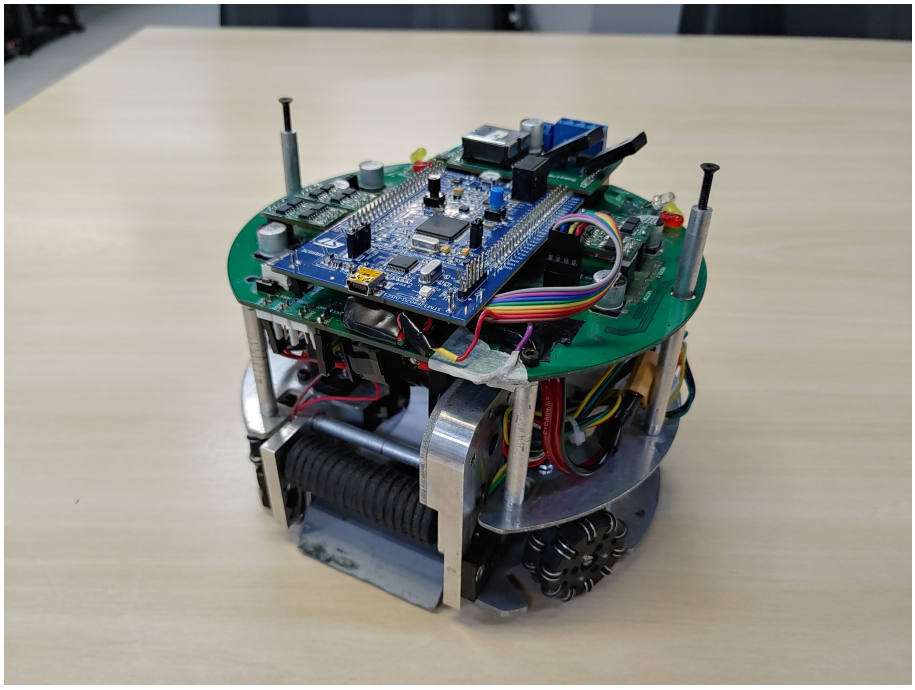

Figura 3 – Robô de SSL do IME, sem carenagem

A competição SSL é realizada anualmente no evento internacional RoboCup. Consiste em partidas de futebol entre equipes de robótica de diversas universidades. Nesta competição, as partidas são disputadas por um time de até seis robôs autônomos, que são controlados pelo computador da equipe (*Team computer*), o qual não pode ser operado durante a execução do jogo. Sobre o campo existe uma câmera conectada a um computador (*Vision computer*) responsável por processar as imagens e reconhecer os padrões de cores das carenagens dos robôs de ambos os times, assim como da bola. Estas informações são enviadas para os computadores de ambas as equipes, responsáveis por tomar as decisões e calcular as estratégias de jogo. Uma ilustração pode ser vista na figura 4

O robô de SSL do IME foi desenvolvido visando atender ao regulamento (4) da competição RoboCup 2023. Sendo assim, deve caber dentro de um cilindro com $180mm$ de diâmetro e $150mm$ de altura. Deve ser capaz de atingir velocidades de até 4m/s e de chutar a bola a uma velocidade de até $6,5m/s$.

Possui quatro rodas omnidirecionais, como a da figura 5, dispostas nos ângulos

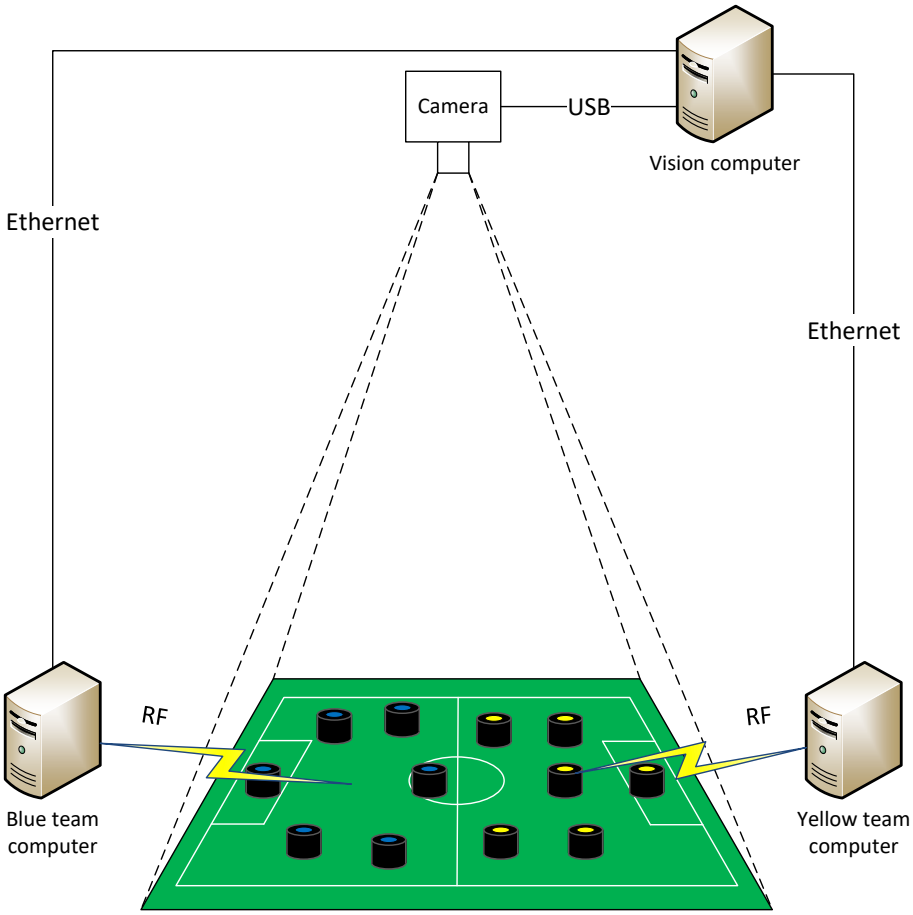

Figura 4 – Topologia de uma partida de SSL

representados na figura 6. Sendo assim, é capaz de se movimentar em três graus de liberdade, chamados aqui de tangente, normal e angular, conforme a figura 7. As rodas dianteiras estão anguladas em 30° em relação à direção do movimento tangente, enquanto as traseiras estão em um ângulo de 45°. Isso torna o movimento tangente mais preciso do que o movimento normal. As rodas são acopladas aos motores por uma redução de $4:1$, e possuem rodilhas dispostas de forma alternada, conforme a figura 8.

Todos os motores são do tipo DC *brushed* (com escovas), e possuem *encoder* de 52 contagens por rotação (CPR) para medição de posição angular relativa. Assim, o robô emprega controle PID em malha fechada para a velocidade de cada uma das rodas, individualmente. São alimentados por uma ponte H da figura 9 desenvolvida também pela equipe RoboIME (5).

Conforme o diagrama da figura 10, o microcontrolador utilizado é um STM32F407VG, da mesma família daquele empregado no SVTRP do CTEx. Dessa forma, o *firmware* desenvolvido para um dos robôs pode ser facilmente adaptado para o outro, sendo necessárias apenas pequenas alterações de compatibilidade. Para comunicação com o computador, o robô utiliza um rádio $2.4GHz$ modelo Semtech SX1280, embarcado em um módulo Ebyte E28-2G4M27S. Este rádio é capaz de se comunicar a distâncias de centenas de metros em

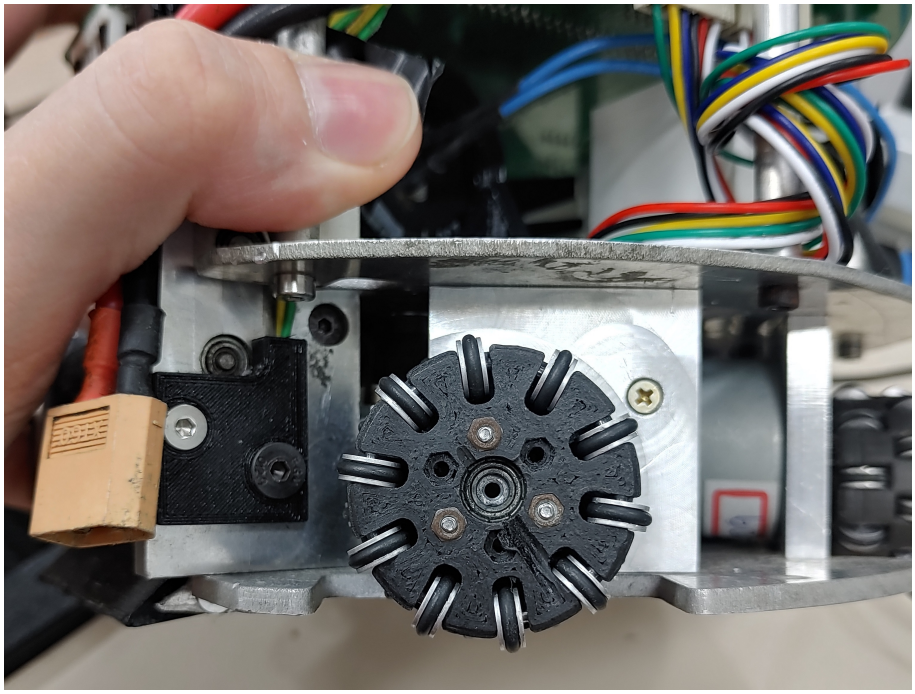

Figura 5 – Roda omnidirecional do robô de SSL do IME

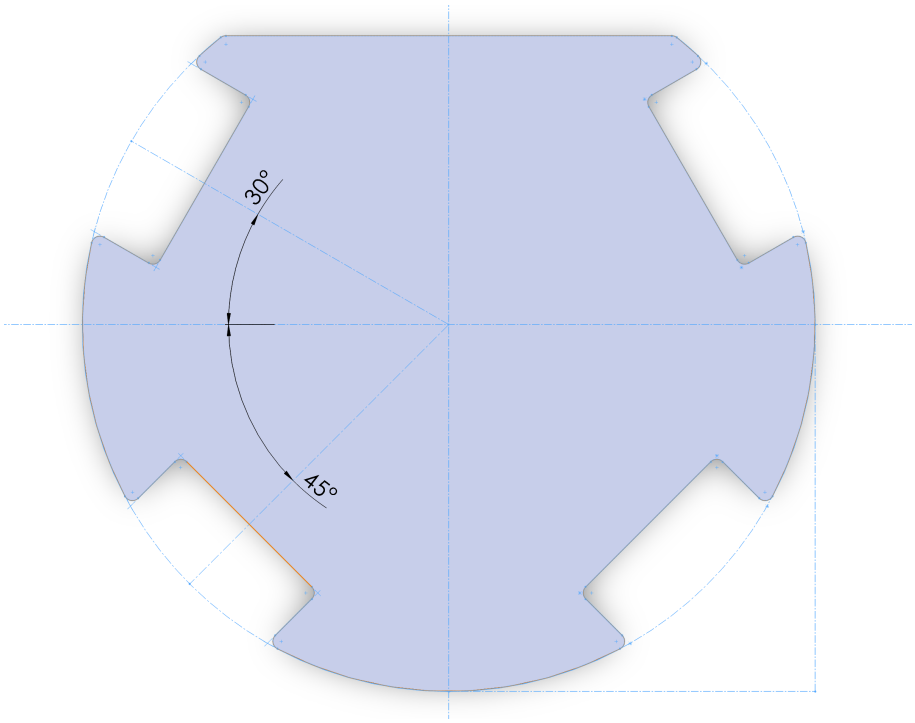

Figura 6 – Disposição das rodas no robô de SSL

campo aberto, a taxas de até 1.3$Mbps$.

Possui um sensor IMU com nove graus de liberdade (9-dof), contendo acelerômetro, giroscópio e magnetômetro. Portanto, é capaz de determinar sua orientação absoluta, porém incapaz de determinar posição absoluta sem ajuda de sistema de câmeras externo

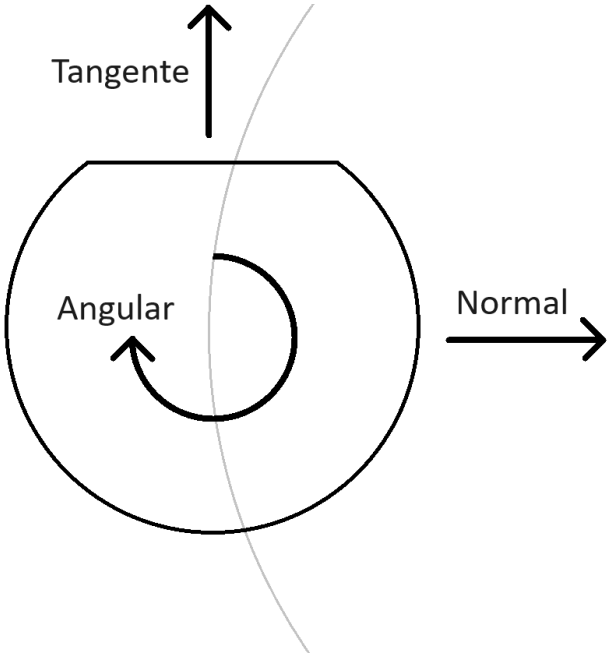

Figura 7 – Cinemática do robô de SSL

Figura 8 – Rodilhas do robô de SSL do IME

por não possuir GPS.

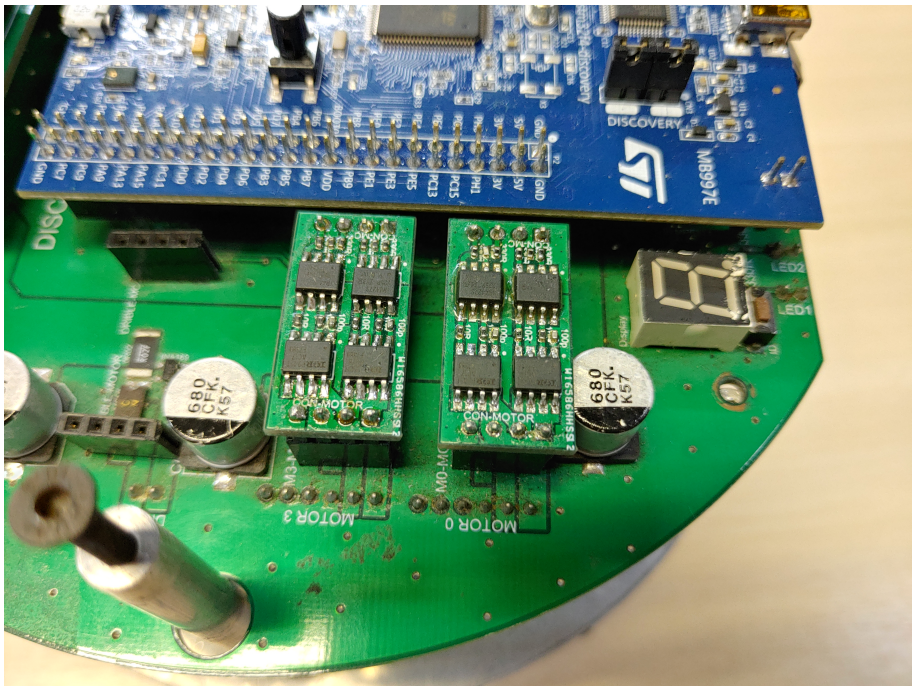

Figura 9 – Pontes H RoboIME 2016

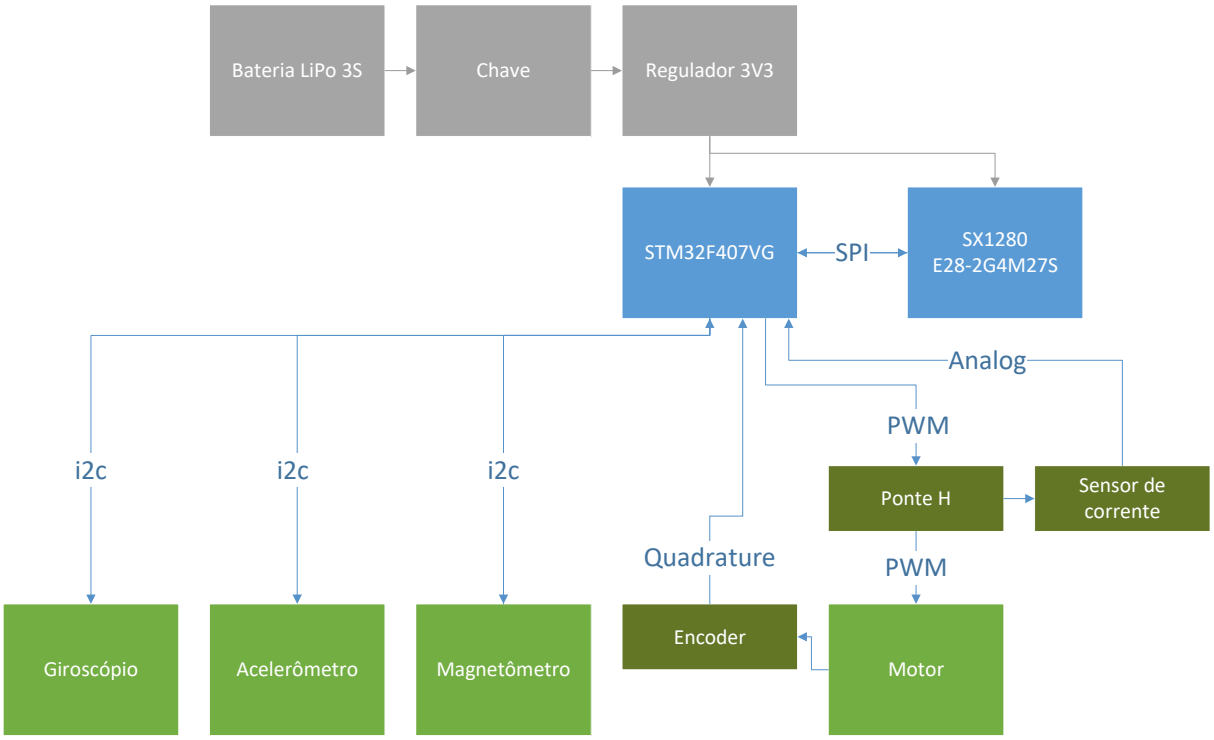

Figura 10 – Diagrama de blocos do Robô de SSL do IME

Tabela 2 – Especificações do robô de SSL do IME

Componente	Especificação
Bateria	LiPo 3S 2200mAh
Microcontrolador	STMicroelectronics STM32F407VG
Ponte H	RoboIME 2016
Módulo de chute	RoboIME 2022
Sensor IMU	InvenSense MPU9250
Sensor de corrente	Texas Instruments INA169NA
Rádio	Ebyte E28-2G4M27S
Motores	Generic JGB37-520

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será abordada a teoria utilizada para o desenvolvimento do projeto, além das considerações e simplificações adotadas na aplicação desta teoria aos casos particulares presentes no escopo do projeto. Serão também mencionadas as dificuldades e limitações de implementação prática desta teoria.

3.1 Modelagem de um motor DC

O motor DC utilizado nos robôs é do tipo com escovas (*brushed*) de ímã permanente. Uma representação gráfica desse tipo de motor pode ser vista na figura 11

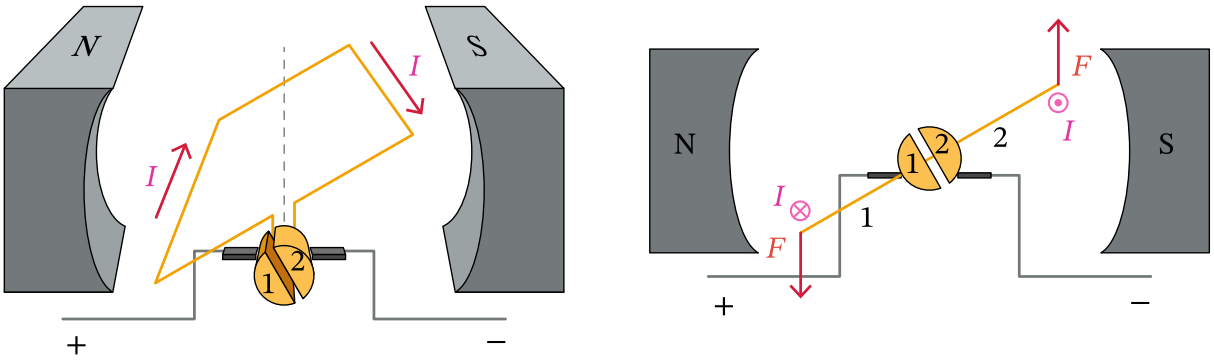

Figura 11 – Representação gráfica de um motor DC. Fonte: (1)

Na figura 11 podemos ver que o motor gira devido à corrente que passa em seu enrolamento presente no rotor. A corrente cria um campo magnético não paralelo ao criado pelos ímãs permanentes do estator. Esse ângulo entre os campos magnéticos faz com que seja gerado um torque entre o rotor e o estator, proporcional à corrente que circula pelo enrolamento do rotor e ao cosseno do ângulo entre os campos magnéticos. Em motores com muitos enrolamentos no rotor chaveados pelas escovas, o que é o caso da maioria dos motores comerciais, inclusive os utilizados neste projeto, podemos considerar que o ângulo entre os campos magnéticos é próximo de 90 graus e, portanto o seu cosseno será aproximadamente igual a 1. Assim, o torque será proporcional apenas à corrente circulante no enrolamento (2) e não dependerá do ângulo do rotor. Podemos então levantar a seguinte equação:

$$\tau = K_t \cdot \ i \tag{3.1}$$

Onde τ é o torque gerado pelo motor, K_t é uma constante de proporcionalidade a ser determinada experimentalmente, chamada aqui de constante de torque, e i é a corrente circulante no motor.

Além disso, o rotor girando em torno de um campo magnético criado pelos ímãs permanentes do estator faz com que seja induzida uma força eletromotriz sobre o enrolamento. Esta força eletromotriz será proporcional à taxa de variação do fluxo magnético no interior do enrolamento. Como o enrolamento tem área atravessada pelo campo magnético constante, a força eletromotriz induzida será proporcional à velocidade angular do rotor (2), conforme a seguinte equação:

$$e = K_e \cdot \dot{\theta}_m \tag{3.2}$$

em que e é a força eletromotriz induzida, K_e é uma constante de proporcionalidade a ser determinada experimentalmente, chamada aqui de constante elétrica, e $\dot{\theta}$ é a velocidade angular do motor.

Sendo assim, o motor DC pode ser modelado segundo o seguinte circuito elétrico e diagrama de corpo livre (2):

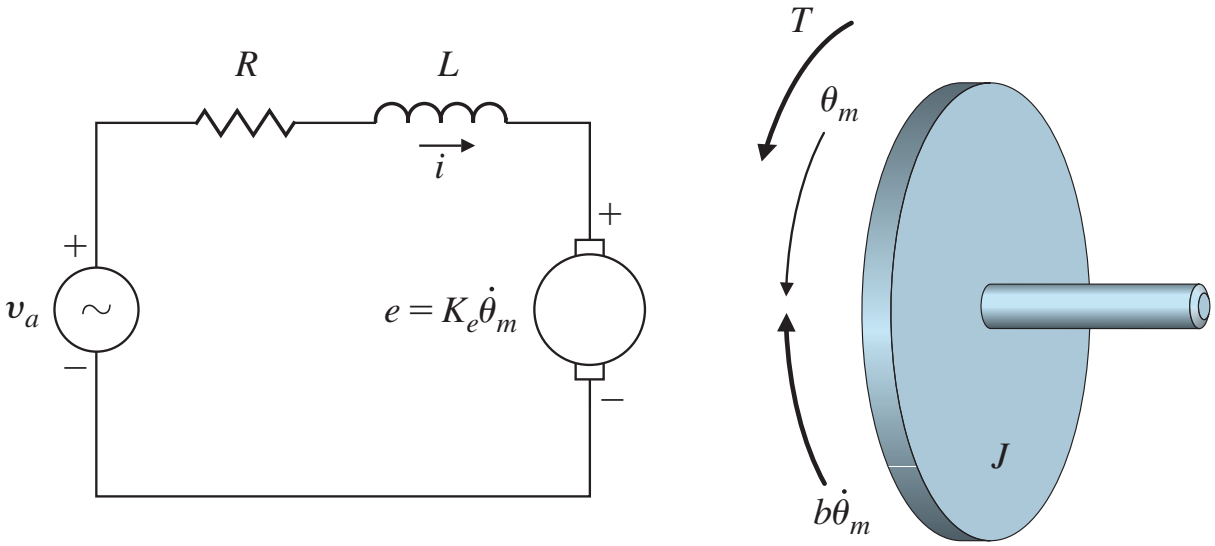

Figura 12 – Circuito equivalente e diagrama de corpo livre de um motor DC. Fonte: (2)

Agora, podemos modelar o motor DC pela seguinte função de transferência:

$$\frac{\theta_m(s)}{V_a(s)} = \frac{K_t}{s[(Js+b)(Ls+R)+K_tK_e]} \tag{3.3}$$

em que θ_m é o ângulo do motor, V_a é a tensão aplicada ao enrolamento, K_t é a constante de torque do motor DC, K_e é a constante elétrica, J é o momento de inércia, b é a constante de atrito viscoso, L é a indutância do enrolamento e R é a resistência do enrolamento.

Caso queiramos a função de transferência da velocidade, que é a derivada da posição, podemos calculá-la multiplicando a função de transferência da posição por s, assim:

$$\frac{\dot{\theta}_m(s)}{V_a(s)} = \frac{K_t}{(Js+b)(Ls+R)+K_tK_e} \tag{3.4}$$

Na maioria dos motores DC comerciais, inclusive nos utilizados neste trabalho, o efeito da indutância é muito pequeno quando comparado com o efeito do momento de inércia, para as frequências típicas de operação. Assim, a função de transferência 3.4 pode ser simplificada (2) para:

$$\frac{\dot{\theta}_m(s)}{V_a(s)} = \frac{K_p}{1+T_{p1}s} \tag{3.5}$$

Onde:

$$K_p = \frac{K_t}{bR+K_tK_e} \tag{3.6}$$

$$T_{p1} = \frac{RJ}{bR+K_tK_e} \tag{3.7}$$

As constantes K_p e T_{p1} podem tanto ser calculadas matematicamente pelas equações acima quanto determinadas experimentalmente com o auxílio de uma bancada de testes e ferramentas computacionais.

Diante disso, percebe-se que o motor DC pode ser modelado como um sistema linear invariante no tempo (LTI) de primeira ordem e, portanto, é atendido por toda a teoria de controle desenvolvida para esse tipo de sistema.

3.2 Controle PID

O controle proporcional integral derivativo (PID) é um dos mais utilizados na robótica devido à sua versatilidade e capacidade de controlar com precisão sistemas com as mais variadas características. É um tipo de controle que se encaixa muito bem com sistemas LTI de primeira ordem, como o motor DC. A malha de controle de um sistema controlado por PID pode ser vista na figura 13.

Em um sistema contínuo a tempo contínuo ideal, a função de transferência do controlador PID será a da equação 3.8

$$\frac{U(s)}{E(s)} = kp + \frac{ki}{s} + kd \cdot s \tag{3.8}$$

Onde kp, ki e kd são constantes chamadas respectivamente de proporcional, integral e derivativa, $U(s)$ é a saída do controlador PID e $E(s)$ é o sinal de erro que entra no

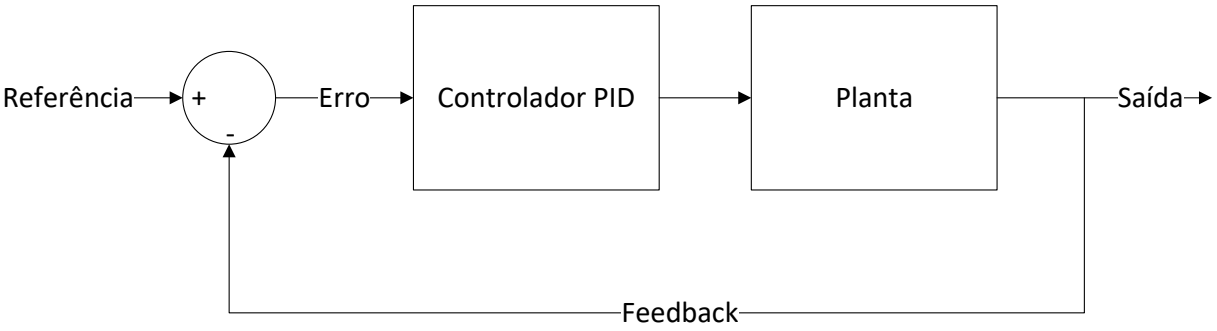

Figura 13 – Diagrama de blocos de um sistema controlado por PID

controlador PID. Influenciam diretamente no comportamento do controlador PID e, portanto, do sistema de controle do robô de forma geral. Encontrar os melhores valores para estas constantes torna-se então um trabalho de otimização que pode ser auxiliado por métodos como os de Ziegler-Nichols (2).

Contudo, esta função de transferência representa um controlador não-causal e, portanto, não pode ser aplicada diretamente a sistemas reais. Sendo assim, é necessário realizar adaptações ao implementar nos robôs.

Nos robô deste trabalho, a planta pode ser entendida como sendo o conjunto que engloba a ponte H e o motor, enquanto as forças externas aplicadas ao motor serão vistas como perturbações $W(t)$ no sistema. Além disso, existem ainda perturbações $V(t)$ causadas por imprecisões dos sensores utilizados. O controlador PID e os cálculos dos sinais de referência e de erro serão implementados no *firmware* do microcontrolador, o que faz com que o robô precise ser um sistema discreto a tempo discreto. Dessa forma, o diagrama de blocos do sistema de controle dos robôs deste trabalho é o da figura 14.

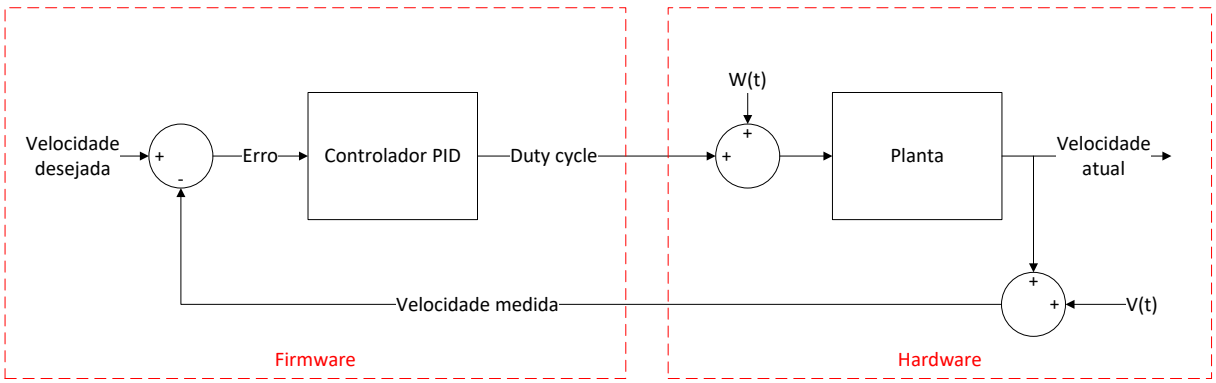

Figura 14 – Diagrama de blocos do controle dos robôs

4 OS PROBLEMAS

De posse da fundamentação conceitual, parte-se então para a descrição dos problemas a serem resolvidos pelo presente trabalho.

4.1 Futebol de robôs SSL

Em disputas de futebol de robôs SSL, a agilidade dos robôs é muito importante, pois o time que tiver os robôs mais ágeis terá vantagem em situações de disputa de bola e de dribles. Eletronicamente, essa agilidade está ligada à dinâmica do sistema de controle, pois robôs com dinâmica mais rápida tendem a alcançar seus objetivos de posição e velocidade mais rapidamente.

Além disso, a precisão do movimento do robô é um fator de grande importância nas finalizações e nos passes, pois é o que determina se o chute será a gol e se o passe vai ser recebido pelo outro robô. No sistema de controle, a precisão está relacionada com o erro de estado estacionário e com o *overshoot*.

Sendo assim, a boa escolha dos componentes e motores e a boa calibração do controlador PID são duas etapas de grande importância no desenvolvimento do robô e na preparação para a competição.

4.2 Deslizamento e perda de direção

Um motor com especificações adequadas para um robô competitivo pareado com um controlador PID bem calibrado é capaz de realizar torques elevados, que em muitos casos pode superar a máxima força de atrito que a roda pode suportar. Nesses casos, pode ocorrer o deslizamento das rodas em relação ao chão.

Quando este deslizamento ocorre de forma desigual entre as quatro rodas do robô de SSL, ocorrerá também uma mudança de direção, o que é indesejado e faz com que o robô desvie de sua trajetória calculada. Este efeito pode ser observado em casos de grandes mudanças de velocidade desejada, como em arrancadas e frenagens bruscas.

4.3 Capotamento

Em princípio, uma possível solução para o problema do deslizamento seria utilizar rodas de maior aderência, pois, por terem maior coeficiente de atrito, fariam com que o

atrito máximo superasse o torque máximo dos motores, evitando, assim, o problema da perda de direção.

Contudo, a aplicação de grandes forças de atrito às rodas do robô pode levar ao capotamento, mesmo em robôs com centro de massa próximo ao chão, o que em um jogo de SSL provocaria uma falta com cartão amarelo.

Caso sejam utilizadas rodas com aderência intermediária, Ocorre o deslizamento em algumas ocasiões e o capotamento em outras, o que mostra a insuficiência da otimização da aderência das rodas na solução dos dois problemas.

4.4 Atrasos no processamento da câmera

Outra possibilidade de solução para os desvios de trajetória causados pelos deslizamentos seria utilizar uma segunda malha de controle, colocando pontos da trajetória desejada como entradas de referência e a posição atual do robô como saída. Neste caso, o sensoriamento da posição atual do robô deve ser feito pela câmera colocada acima do campo. Porém, leituras realizadas por esta câmera possuem um atraso causado pela soma de pequenos atrasos oriundos de diversas fontes, como codificação de vídeo dentro da câmera, transferências por cabo USB, processamento dos pacotes USB pelo *driver* da câmera, processamento da imagem recebida para identificação da posição do robô, filtragens de ruídos no posicionamento e atrasos na transmissão da informação para o computador da equipe por rede Ethernet. A soma destes atrasos inviabiliza o uso de controle em malha fechada utilizando a câmera, pois a taxa de controle neste caso teria que ser muito baixa, o que levaria a uma dinâmica muito lenta e, portanto, pouco competitiva.

Diante dos problemas apresentados, nota-se que apenas correções mecânicas não serão suficientes para obter-se um resultado satisfatório, visto que não há ponto de equilíbrio entre o deslizamento e o capotamento para todas as situações de jogo. Além disso, nota-se que aplicar controle de malha fechada apenas com a câmera levará o robô a uma dinâmica muito lenta. Dessa forma, enxerga-se a necessidade de aprimorar o sistema de controle do robô, sem o uso de sensores externos.

5 PROJETO DO CONTROLE ANTI-DERRAPAGEM

5.1 Robô de desenvolvimento

Devido à maior dificuldade de acesso no dia-a-dia ao robô SVTRP do CTEx e à maior complexidade do robô de SSL devido às rodas omnidirecionais, foi decidido construir um novo robô visando a modularidade e simplicidade da dinâmica. Dessa forma, foi construído o robô da figura 15, cujo diagrama de blocos está representado na figura 16

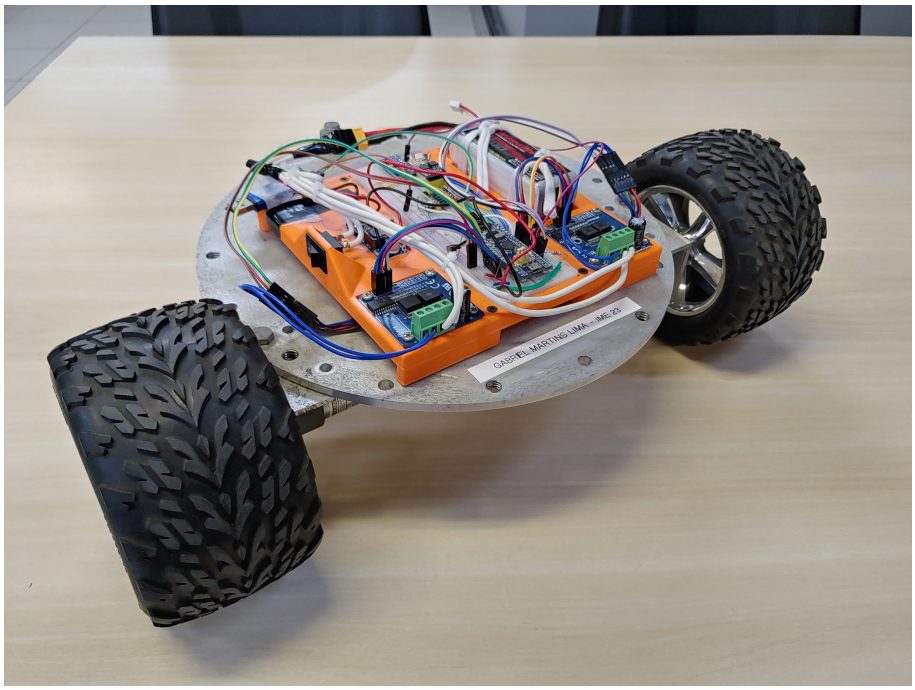

Figura 15 – Robô de desenvolvimento visto de frente

Este robô é movimentado por motores semelhantes àqueles do robô de SSL, possuindo encoder de 52 CPR, porém com redução de 1:30, devido às rodas maiores. Sua cinemática é diferencial, semelhante à do SVTRP do CTEx, porém possui apenas duas rodas tracionadas e uma roda "boba"para apoio, que pode ser vista na figura 17. As rodas tracionadas foram escolhidas de forma que deslizem caso seja aplicado pelo motor um torque elevado, e o motor foi escolhido para ser capaz de aplicar torque que leve as rodas ao deslizamento.

Para possibilitar a modularidade e o experimento com diferentes sensores, motores e microcontroladores, foi decidido colocar sobre o robô uma protoboard, sustentada por uma base projetada em software CAD e manufaturada em impressora 3D, conforme figuras 18 e 19.

Possui um sensor IMU 9-dof igual ao do robô de SSL, instalado no centro do robô,

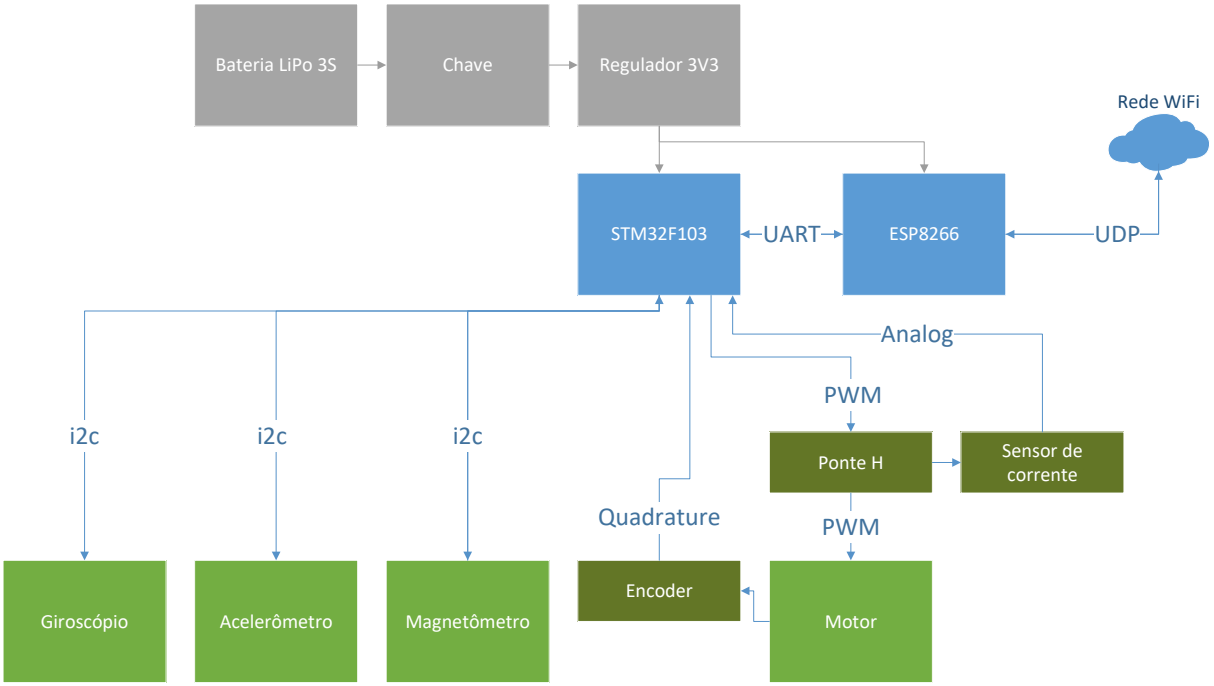

Figura 16 – Diagrama de blocos do Robô de desenvolvimento

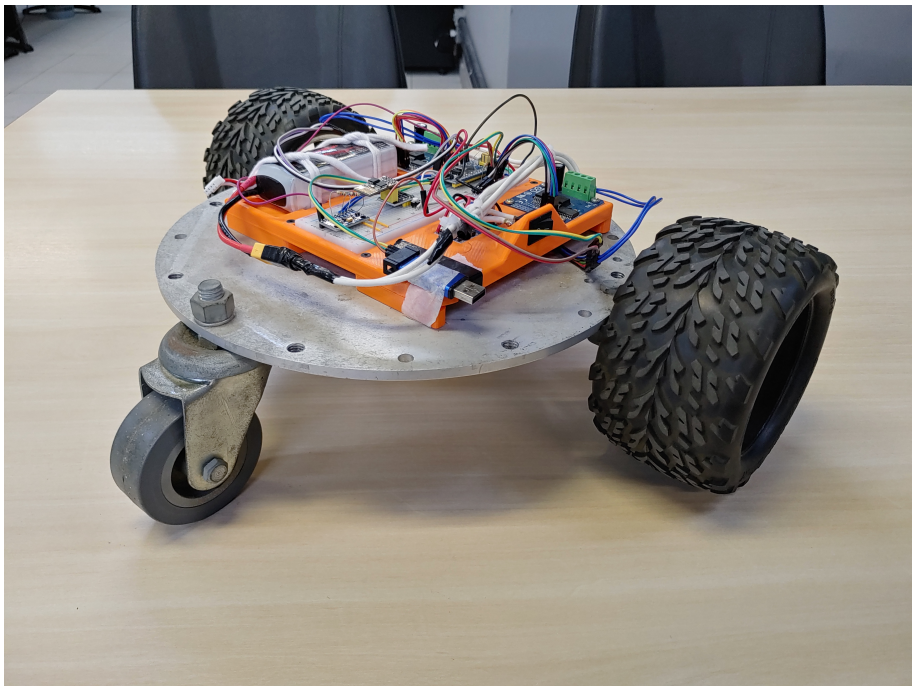

Figura 17 – Robô de desenvolvimento visto de trás

pontes H com sensores de corrente integrados, de forma que a corrente que entra em cada motor leve a uma tensão proporcional, que pode ser lida por um conversor analógico-digital (ADC) do microcontrolador.

O microcontrolador principal escolhido foi o STM32F103C8 "BluePill", o mesmo

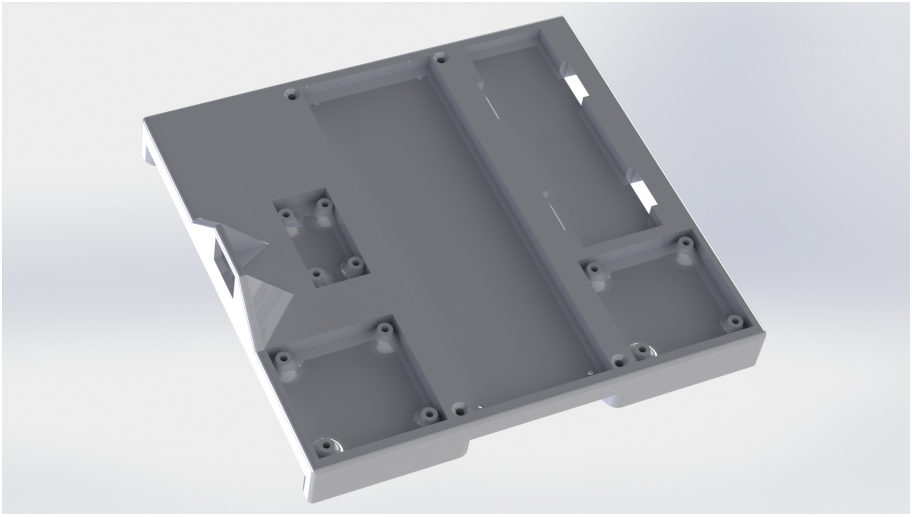

Figura 18 – Base modelada no software CAD SolidWorks

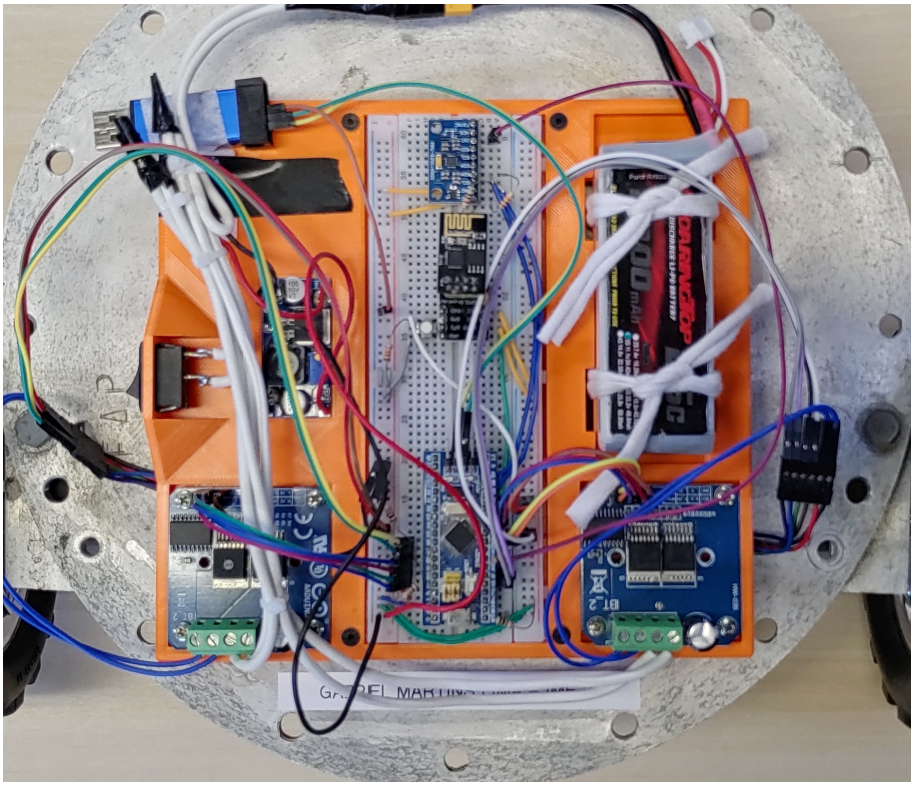

Figura 19 – Base impressa e montada no robô

do SVTRP do CTEx, pois este possui características e especificações suficientes para o desenvolvimento do projeto e, por ser o mesmo do SVTRP do CTEx, faz com que fique mais simples de instalar o *firmware* desenvolvido no SVTRP do CTEx posteriormente.

Para o comando, telemetria e coleta de dados, foi decidido utilizar um módulo ESP8266, capaz de se conectar a uma rede WiFi. Para o ESP8266 foi desenvolvido um *firmware* que traduz a comunicação UART do microcontrolador principal em pacotes UDP *multicast* e vice-versa.

Para auxílio no desenvolvimento, foi desenvolvida uma interface gráfica em LabVIEW que se comunica com o robô por UDP *multicast*. Esta interface envia para o robô os comandos de velocidade desejada, da mesma forma como é feito no robô de SSL, e pode receber de volta os dados de telemetria desejados, como velocidade e aceleração das rodas medidas pelos encoders, acelerações medidas pelo IMU, corrente consumida pelos motores e tensão atual da bateria. A interface está representada na figura 20

Nesta interface, podem ser vistos gráficos das velocidades e acelerações em função do tempo, preenchidos em tempo real. Pode ser utilizada também para enviar comandos de velocidade desejada para o robô a partir de entradas do operador ou de rotinas de testes automáticos.

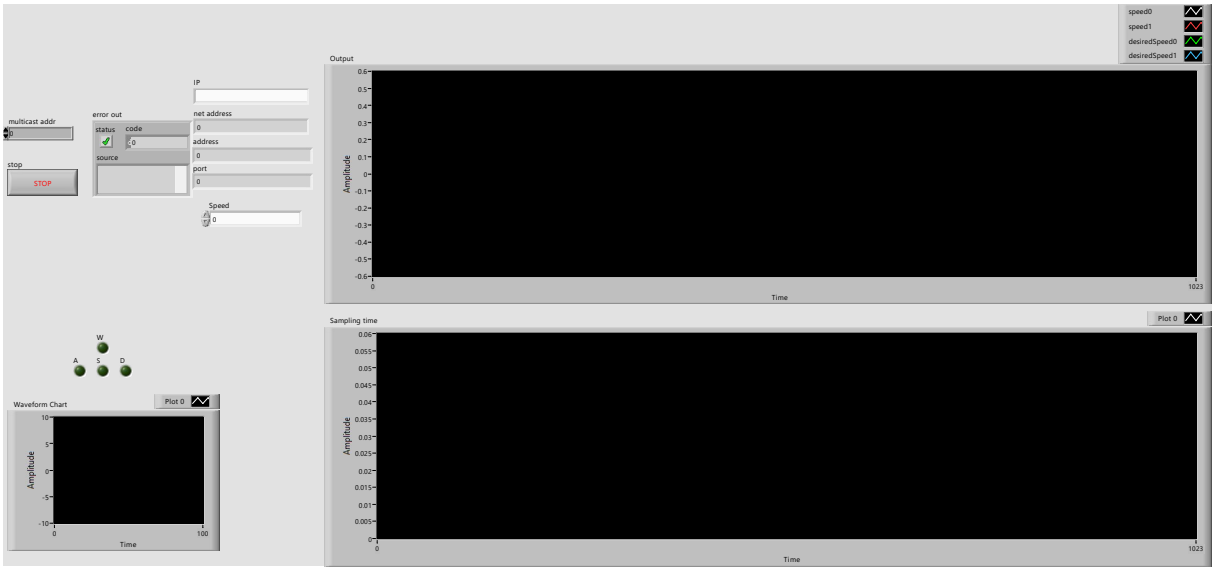

Figura 20 – Interface gráfica no LabVIEW

5.2 Comunicação entre STM32 e GUI

Devido à necessidade de realizar a comunicação entre o robô e o computador sem que cabos prejudiquem as medições e o funcionamento, foi adotado o protocolo UDP, por ser possível o uso de comunicação WiFi. Para tal, foi implantado no robô um microcontrolador ESP8266, que possui um periférico UART, o qual pode ser utilizado para comunicação com o microcontrolador principal STM32, e um periférico WiFi, que pode ser utilizado para comunicação com o computador. Foi desenvolvido um firmware em $C++$ para o ESP8266 que converte os pacotes enviados por UART em pacotes UDP e vice-versa. Os pacotes UDP são transmitidos e recebidos por multicast, o que permitiria que fossem utilizados múltiplas interfaces gráficas para telemetria do robô.

5.3 Desenvolvimento do *Firmware* do STM32

Foi desenvolvido um *firmware* para o microcontrolador STM32 do robô de desenvolvimento.

5.3.1 Recursos e características do microcontrolador

O microcontrolador do robô de desenvolvimento é o STM32F103C8T6, embarcado em uma placa *BluePill*, amplamente encontrada no mercado nacional. A placa *BluePill* pode ser vista na figura 21.

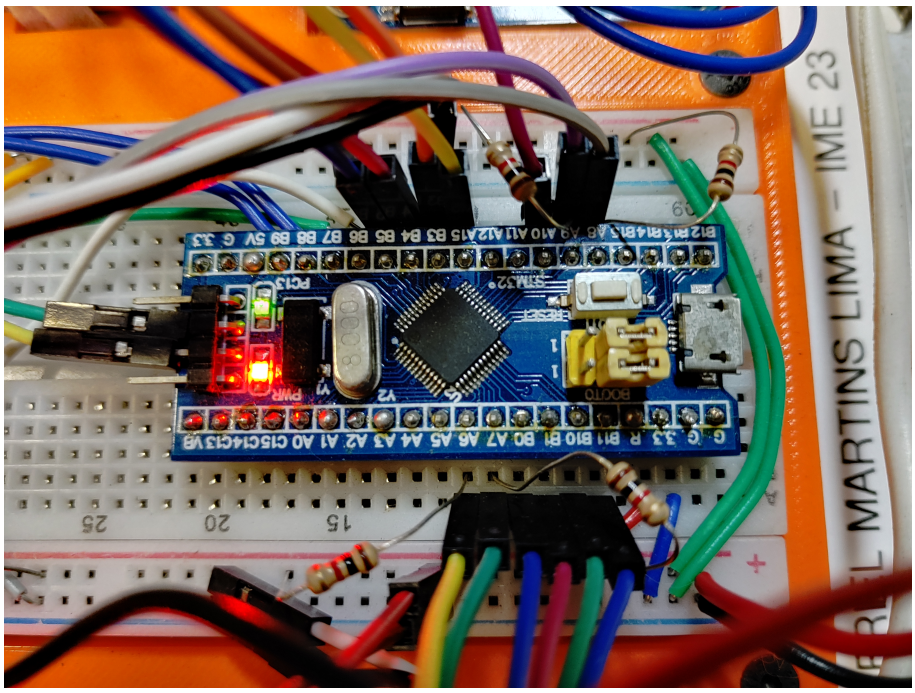

Figura 21 – Placa *BluePill* com STM32F103C8T6

O STM32F103C8T6 possui as especificações técnicas da tabela 3

Dos quatro timers do microcontrolador, um deles é utilizado para gerar os sinais PWM para alimentação dos motores, e dois são utilizados para leitura dos encoders, deixando um para interrupções à taxa de controle, o que permite que a programação seja feita orientada a interrupções.

Possui um controlador de interrupções que permite preempções, além de definição de prioridades de preempção e de despacho das rotinas de serviço de interrupção (ISRs).

O nível lógico de 3.3V torna o microcontrolador compatível com a maioria dos componentes e sensores encontrados no mercado, e com todos os presentes no robô de desenvolvimento, sem a necessidade de conversores de nível, o que permite taxas de comunicação maiores.

Tabela 3 – Especificações do STM32F103F8T6

Componente	Especificação
Arquitetura	ARM Cortex M3
Frequência de clock	72MHz
Memória flash	64kB
Memória RAM	20kB
Tensão de operação	3.3V
Timers	4
Periféricos SPI	2
Periféricos I^2C	2
Periféricos UART	3
GPIOs	37
ADCs	2 (10 canais)

O núcleo ARM Cortex M3 possui um conjunto de instruções de complexidade intermediária para um microcontrolador. Possui instruções para divisão em hardware, porém não embarca unidade de ponto flutuante. Possui um registrador contador de clocks, o que permite medição de intervalos de tempo com precisão de $1/F_{clk}$, onde $f_{clk} = 72MHz$. Possui também unidade de debug, que pode auxiliar durante o processo de desenvolvimento do firmware.

5.3.2 Linguagens de programação

Foram usadas as linguagens de programação C e $C++$, devido à sua capacidade de acomodar o programa compilado em pequenas quantidades de memória RAM e *flash*, o que é uma característica dos microcontroladores. Além disso, as linguagens escolhidas permitem que o programa seja executado em tempos determinísticos, aumentando a precisão do sistema de controle.

A biblioteca *HAL* da STMicroelectronics foi utilizada como camada de abstração para o *hardware* e os periféricos do microcontrolador, o que facilita o desenvolvimento tirando a necessidade de conhecer profundamente a estrutura do microcontrolador e o mapeamento de memória e de registradores. Além disso, o uso da biblioteca *HAL* facilita a futura portabilidade do *firmware* para outros microcontroladores da mesma família.

5.3.3 Ferramenta STM32Cube

Em programação de microcontroladores, é necessário chamar as rotinas de inicialização dos periféricos. Porém, mesmo com o USO da biblioteca *HAL*, seria necessário conhecer as capacidades e limitações do microcontrolador, para que não fosse feita uma inicialização não suportada, e para que as inicializações dos periféricos fossem feitas da forma correta.

Dessa forma, a ferramenta STM32Cube facilita este processo, pois permite a configuração e inicialização dos periféricos por uma interface gráfica, já adaptada às características e limitações de cada microcontrolador. Após a configuração, a ferramenta gera o código em *C* da configuração e inicialização da biblioteca *HAL*.

A interface de configuração do STM32Cube está na figura 22.

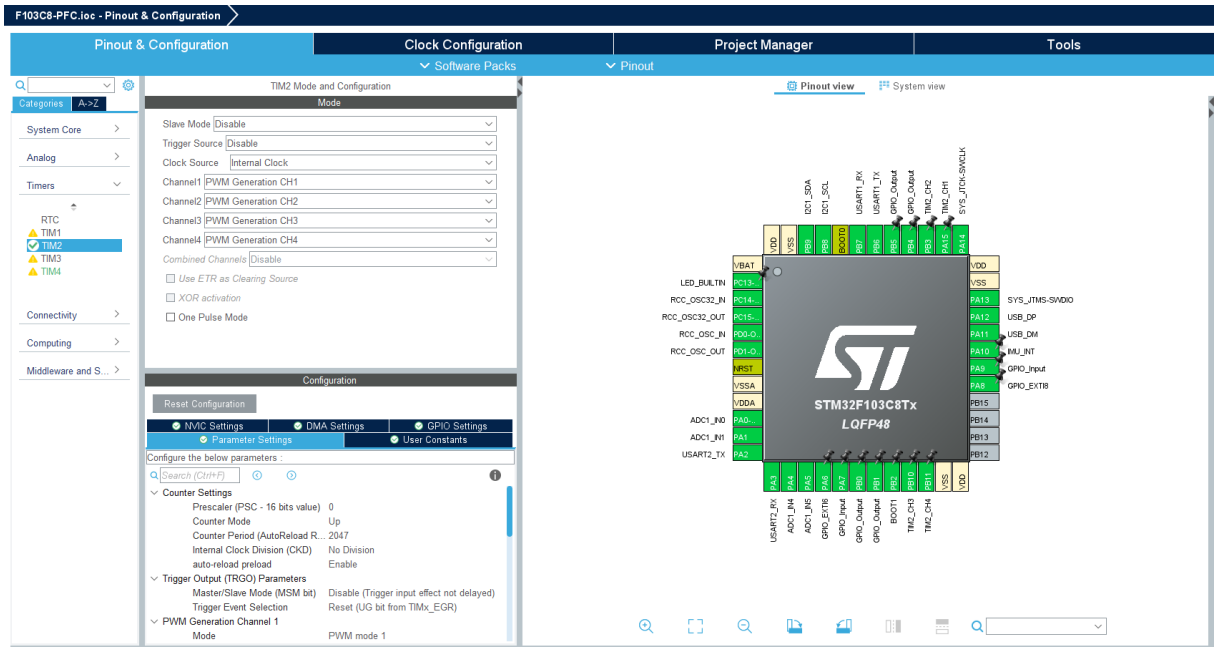

Figura 22 – Interface do STM32Cube

5.3.4 Programação orientada a interrupções

Foi utilizado um fluxo de programa chamado de programação orientada a interrupções, em que toda a lógica do programa acontece dentro de uma rotina de serviço de interrupção (ISR), que é chamada em intervalos de tempo fixados por um timer. A frequência com que essa ISR chama-se taxa ou frequência de controle, pois cada iteração do sistema de controle do robô, um sistema a tempo discreto, ocorre em uma chamada da ISR. A única restrição para este tipo de programação é que cada execução da ISR deve ser finalizada antes da próxima chamada, o que pode ser verificado por meio de ferramentas de *debug*.

5.3.5 Implementação da leitura do encoder

Encoders de quadratura magnéticos, como os presentes nos motores do robô de desenvolvimento são dispositivos capazes de capturar a posição de um motor relativa à sua posição inicial. para isto, possuem dois sensores de efeito *hall*, posicionados próximos

a um disco solidário ao eixo do motor, com magnetizações que variam entre norte e sul, conforme representação na figura 23.

Quando o motor gira, os *encoders* geram duas ondas quadradas de mesma frequência, mesma amplitude, porém fases diferentes. A frequência da onda gerada é proporcional à velocidade angular do motor em que o *encoder* está acoplado, enquanto a diferença de fase pode ser de $+90°$ ou $-90°$, a depender da direção do movimento do motor. A medição da onda quadrada gerada por um encoder do robô de desenvolvimento pode ser vista na captura de tela do osciloscópio na figura 24.

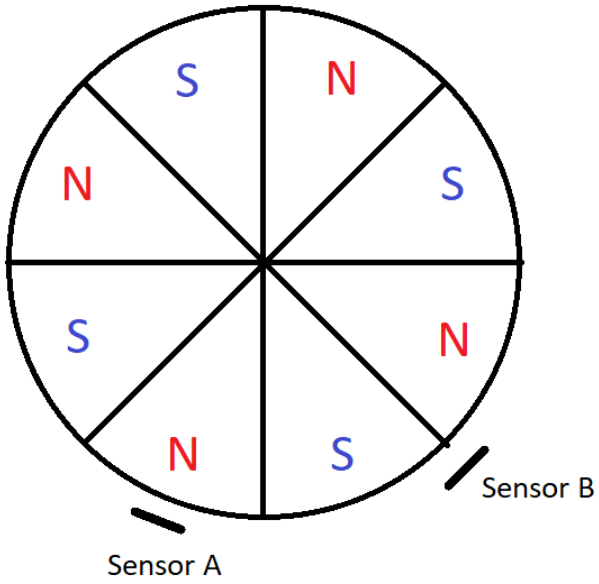

Figura 23 – Representação dos sensores no disco do encoder

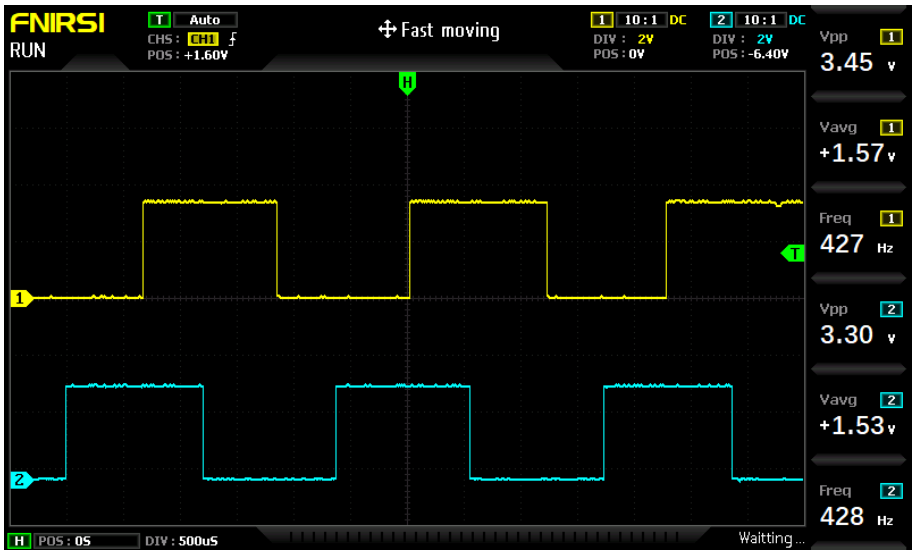

Figura 24 – Leitura do encoder no osciloscópio

O encoder pode ter sua posição lida por um *timer* do microcontrolador, onde cada variação na posição do motor, capturada pelas bordas de subida e descida das ondas da

figura 24, leva a um incremento ou decremento do valor do *timer*, a depender da direção do movimento. Esta metodologia possui duas importantes limitações.

A primeira limitação se deve o fato de o *encoder* ter um número finito de contagens por rotação. No caso do robô de desenvolvimento, são 52 contagens. Assim, o valor do *timer* só incrementa ou decrementa a cada uma variação de 1/52 rotações, sendo impossível medir variações angulares menores. Assim, as variações angulares entre duas iterações da ISR de controle serão quantizadas.

A segunda limitação é que a velocidade do motor é lida fazendo-se a divisão entre a variação da posição entre duas iterações da ISR e o período de controle. Assim, caso seja aumentada a taxa de controle, o erro de quantização relacionado à primeira limitação irá aumentar, levando a medidas de velocidade menos precisas.

5.3.6 Implementação do controlador PID

O controlador PID foi implementado em uma função do $C++$, que tem como entrada a velocidade linear desejada para a roda e como saída o ciclo de trabalho (duty cycle) do PWM.

Para implementação de controladores PID a sistemas reais, há dois problemas:

O primeiro, é que o termo derivativo do controlador PID ideal, representado na equação 3.8, requer que o sistema não seja causal e, portanto, não pode ser implementado fisicamente, conforme (2). Este problema pode ser resolvido em um sistema discreto calculando-se o termo derivativo segundo a equação 5.1, o que traz um termo derivativo calculado com meia amostra de atraso e, portanto, podendo ser calculado em um sistema causal.

$$dError = \frac{erro - erroAnterior}{deltaT} \tag{5.1}$$

O segundo problema é que, segundo (2), a teoria dos controladores PID ideais considera que o motor pode realizar torque infinito, visto que a saída do cálculo do PID da equação 3.8 pode ir de $-\infty$ a ∞, não considerando a limitação do duty cycle de ir de -1 a 1. Dessa forma, em situações onde o duty cycle calculado pelo PID estiver fora dos limites físicos, o cálculo do termo integral pode ser prejudicado por tais inconsistências com o sistema ideal, o que deixa o sistema instável. Este problema foi resolvido limitando-se por saturação o termo integral entre dois valores, determinados experimentalmente fazendo-se um balanço entre erro de estado estacionário e estabilidade do sistema.

Além disso, foi necessário determinar uma taxa de controle que permitisse uma alta largura de banda para o sistema sem ter problemas com os erros de quantização citados

na seção 5.3.5. Dessa forma, foi determinado experimentalmente que a taxa de $100Hz$ leva a um bom equilíbrio.

5.3.7 Implementação do sensor IMU

Conforme mencionado no capítulo 2, foi implementado no robô um sensor IMU 9-DOF. O sensor se comunica com o microcontrolador por protocolo SPI, que possui periférico dedicado no microcontrolador. Após a análise das possibilidades de desenvolvimento de biblioteca para a interface entre o sensor IMU e o microcontrolador STM32, foi decidido adaptar uma biblioteca feita para Arduino (6), tendo em vista sua simplicidade de adaptação e de uso, frente à grande dificuldade de desenvolvimento de uma biblioteca do zero para este sensor e à má qualidade das bibliotecas para STM32 encontradas na internet.

5.4 Determinação das constantes do PID

Tendo desenvolvido o *firmware*, é necessário agora determinar as constantes do controlador PID. Em versões anteriores do *firmware*, as constantes kp, ki e kd foram determinadas experimentalmente por tentativa e erro. Os resultados não eram satisfatórios e levavam o robô a um movimento impreciso, sobretudo no tocante a erro de estado estacionário e overshoot.

Foi então aplicada uma nova técnica de calibração do controle PID utilizando a ferramenta *pidTuner* do *Matlab*, em conjunto com as ferramentas de *software* descritas no capítulo 2.

5.4.1 Identificação da planta

Para a calibração de sistemas de controle por projeto baseado em modelo, é necessário identificar a planta, por uma função de transferência ou representação em espaço de estados. Para este projeto, foi decidido utilizar a função de transferência. Conforme descrito no capítulo 3, a função de transferência do motor DC do robô de desenvolvimento é da forma da equação 3.5. Assim, a identificação da planta se resume a encontrar os valores de K_p e T_{p1}.

Uma forma de encontrar K_p e T_{p1} é encontrar uma planta que, quando feita a simulação, tenha resposta ao degrau em malha aberta próxima à do sistema real. A ferramenta *pidTuner* do *Matlab* aliada ao *Systems Identification Toolbox* é capaz de encontrar esta planta que se ajusta quando alimentada com a resposta ao degrau da planta real.

Assim, a primeira etapa para identificação da planta é capturar a resposta ao degrau do motor DC do robô de desenvolvimento, o que foi feito com a ferramenta em *LabVIEW*, conforme a figura 25

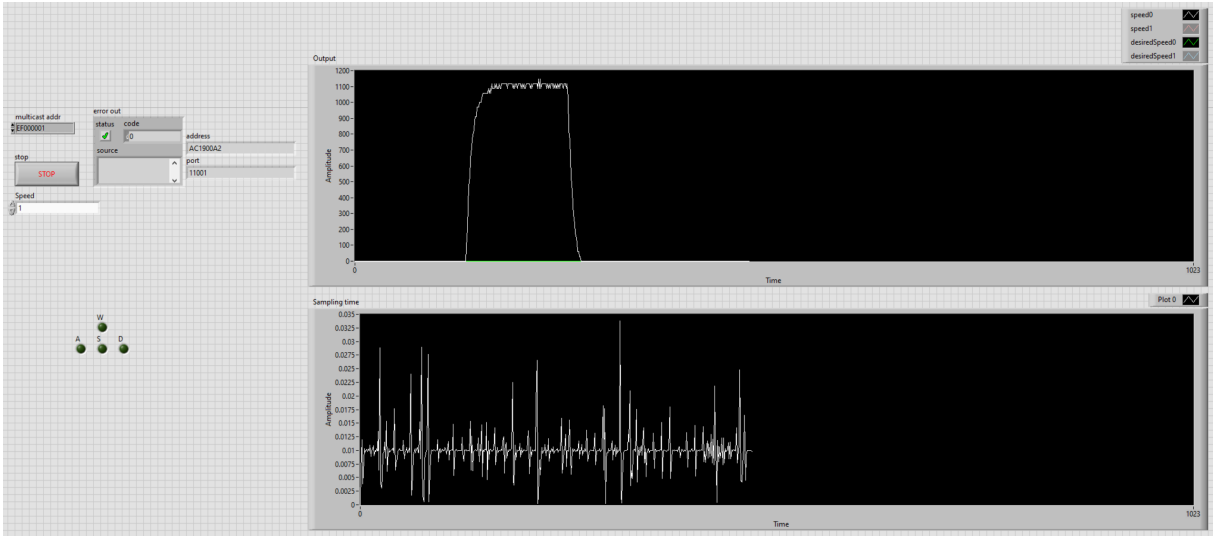

Figura 25 – Ferramenta em LabVIEW capturando a resposta ao degrau do motor

Em seguida, os dados do gráfico *Output* foram exportados para o *Matlab*. Foram selecionados apenas os dados relevantes para identificação da planta e apenas o intervalo de tempo que fazia com que a aplicação do degrau fosse no tempo $t = 0$, desconsiderando o intervalo antes do degrau ser aplicado e depois de ser aplicado o degrau de descida.

Foi então aberta a ferramenta *pidTuner*, e utilizado o comando *Import step response*, que abriu a janela da figura 27, onde foi possível importar o *riseArray* gerado no passo anterior. Nesta mesma janela foram colocados os dados da entrada degrau, como amplitude de 1 e tempo de amostragem de $0.01s$.

Em seguida, a ferramenta automatizada do *Matlab* foi utilizada para encontrar por métodos numéricos iterativos uma planta que, quando simulada, melhor se ajusta à resposta ao degrau da planta a ser identificada. Na figura 28, a resposta ao degrau importada pode ser vista no gráfico à direita com linha verde, enquanto a resposta ao degrau da planta simulada pode ser vista em azul. Note que a resposta ao degrau importada é de um sistema discreto a tempo discreto. Assim, os valores de tempo e velocidade são discretos.

Assim, foi encontrada a planta cuja resposta ao degrau melhor se ajusta à da planta a ser identificada, o que pode ser validado pela figura 29. Sua respectiva função de transferência, bem como os valores de Kp e $Tp1$, podem ser vistas na ferramenta e nas equações 5.2 e 5.3, onde $v(s)$ é a velocidade linear da roda em mm/s e $dutyCycle$ é o ciclo de trabalho do PWM aplicado ao motor, normalizado entre -1 e 1.

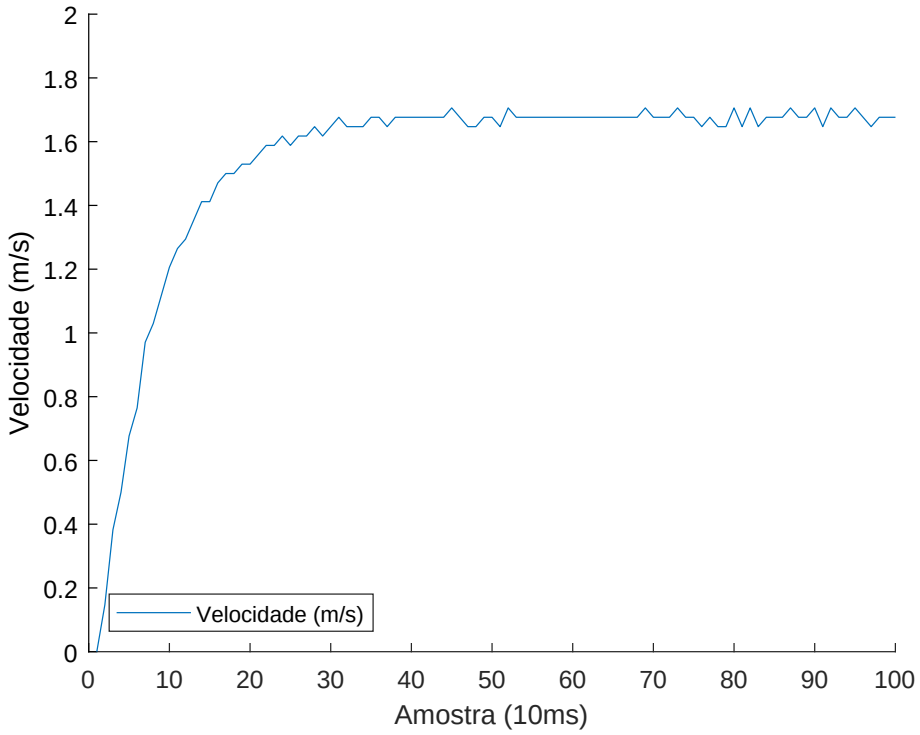

Figura 26 – Gráfico com a resposta ao degrau importada no Matlab

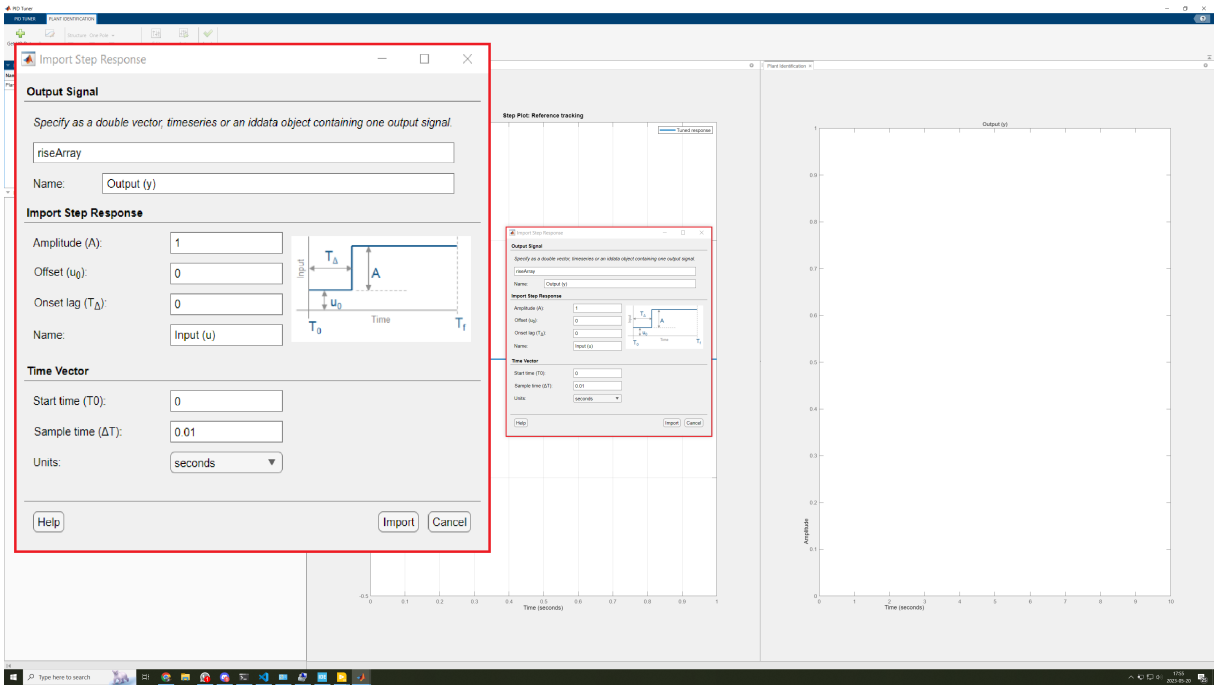

Figura 27 – Importação da resposta ao degrau no pidTuner

$$\frac{v(s)}{dutyCycle(s)} = \frac{K_p}{1 + T_{p1}s} \tag{5.2}$$

$$K_p = 1108,7 \quad T_p1 = 0,073597 \tag{5.3}$$

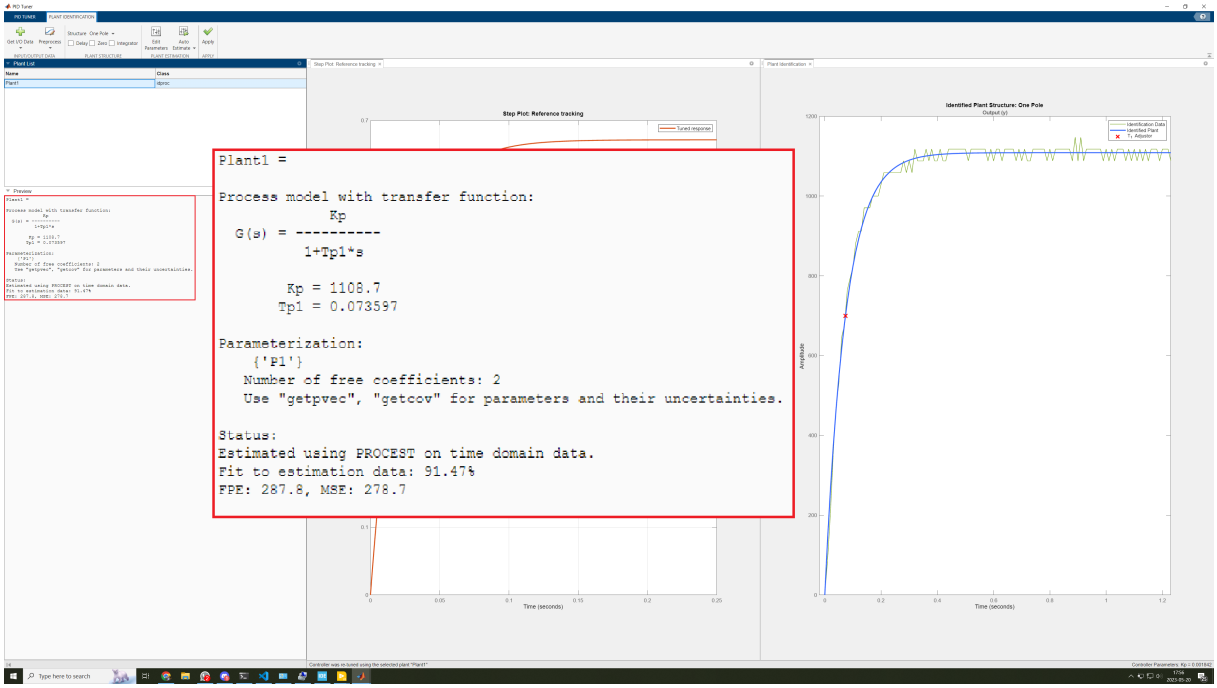

Figura 28 – Planta sendo identificada

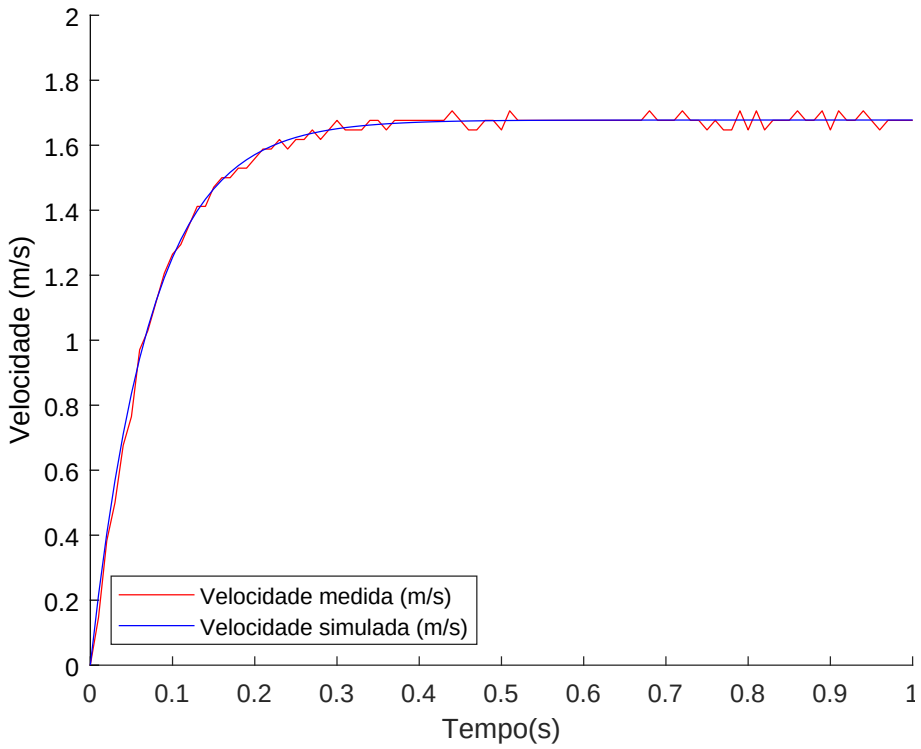

Figura 29 – Comparação em malha aberta da planta simulada com a resposta medida.

5.4.2 Otimização do controle PID

Tendo obtido a função de transferência da planta, encontrar as constantes *kp*, *ki* e *kd* do controle PID torna-se um problema de otimização, com determinadas restrições. O

controle deve ser otimizado para ter o menor *overshoot*, menores tempos de subida e de acomodação, e menor erro de estado estacionário. Porém, deve respeitar a limitação de não ultrapassar os máximos de *dutyCycle* dos motores (-1 a 1). Além disso, como o robô é um sistema discreto, e a calibração do PID é feita neste caso por aproximação de um sistema contínuo, a largura de banda do controlador tem que ser tal que esta aproximação seja satisfatória.

Por determinações experimentais, observou-se que a largura de banda do controlador deve ser maior que a frequência de corte da mecânica do motor, observando-se o sistema de função de transferência 5.2 como sendo um filtro passa-baixas de primeira ordem. Além disso, a largura de banda deve ser muito menor do que a taxa de controle de $100Hz$, para que a aproximação do sistema discreto para um sistema contínuo ainda seja válida. Desta forma, observou-se que a largura de banda de $5Hz$ para o controlador PID satisfaz ambos os requisitos.

Na figura 30, vê-se a ferramenta mostrando a resposta ao degrau em malha fechada para o controlador PID desejado, além das constantes *kp*, *ki* e *kd* para o mesmo controlador. Na mesma figura 30 vê-se também a resposta do robô de desenvolvimento com o controlador aplicado. 30

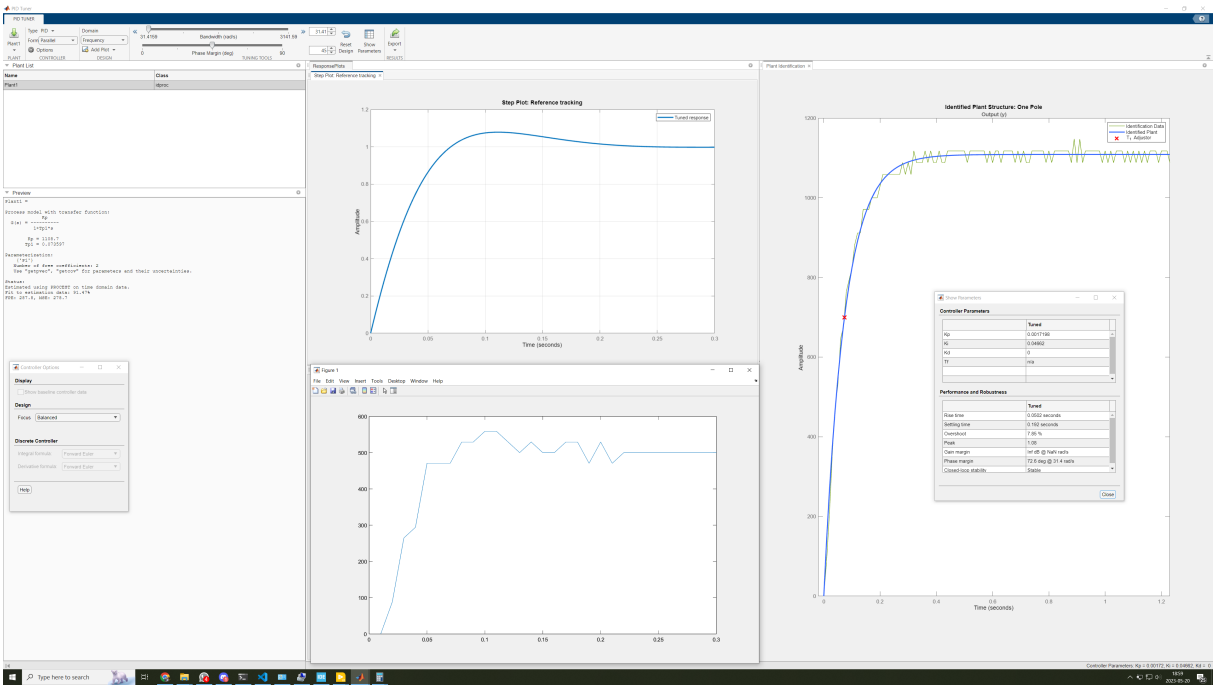

Figura 30 – Robô com PID otimizado

5.5 Identificação do deslizamento

Tendo o robô com controlador PID calibrado, o primeiro passo para resolver os problemas de deslizamento é identificar as condições que levam a deslizamento. Após extensas observações, foi decidido utilizar a comparação da aceleração das rodas, medida pela segunda derivada da posição dos encoders, com a aceleração do chassi do robô, medida pelo sensor acelerômetro do IMU. Sendo assim, considera-se que há deslizamento quando a diferença entre as acelerações é maior do que um certo valor limite. Caso o deslizamento seja identificado, serão adotadas as medidas corretivas a definir.

A figura 31 apresenta o fluxograma da detecção do deslizamento, implementada no programa em LabVIEW, junto com a interface gráfica.

Em primeiro momento, é coletada a entrada do operador do robô, sendo esta a velocidade de reverência desejada para o robô, chamada de *desiredSpeed* no programa. Em seguida, *desiredSpeed* e *slip* são enviados para o robô pela comunicação UDP, e um pacote de *feedback* é recebido, contendo a velocidade do robô medida pelo *encoder*, chamada de *currentSpeed* e a aceleração do chassi, medida pelo sensor IMU. *currentSpeed* é então derivada, obtendo-se a aceleração *encoderAccel*, que é a aceleração da roda medida pelo *encoder*. Além disso, a aceleração do chassi é filtrada também por filtro passa-baixa, obtendo-se *imuAccel*.

As acelerações são então comparadas e, caso sua diferença supere o valor de *threshold* definido no programa, é considerado que o robô está em condição de deslizamento, e a variável *slip* é definida como verdadeira e enviada para o robô junto com o próximo comando, para que o robô, sabendo de sua condição de deslizamento, tome as medidas corretivas cabíveis.

5.6 Correção do deslizamento

A solução corretiva adotada foi desabilitar o motor nos casos de deslizamento. Desabilitando o motor, seu torque será zerado e, portanto, o único torque aplicado sobre a roda será aquele realizado pela força de atrito. Assim, a roda voltará a ter aderência com o chão e o deslizamento será corrigido.

A ponte H utilizada no robô de desenvolvimento, com seu esquema elétrico simplificado na figura 32, permite que cada terminal do motor seja conectado aos potenciais de terra, $VBAT$, ou deixar desconectado. Em condições normais, caso se queira rotação para a frente, o PWM do motor é controlado chaveando-se entre os MOSFETs MAH e MAL, enquanto MBH é mantido conduzindo e MBL é mantido aberto. Para rotação reversa, é adotado o caso análogo. Em condições de deslizamento, todos os MOSFETs são mantidos em aberto, cessando a passagem de corrente pelo motor e, consequentemente, zerando seu

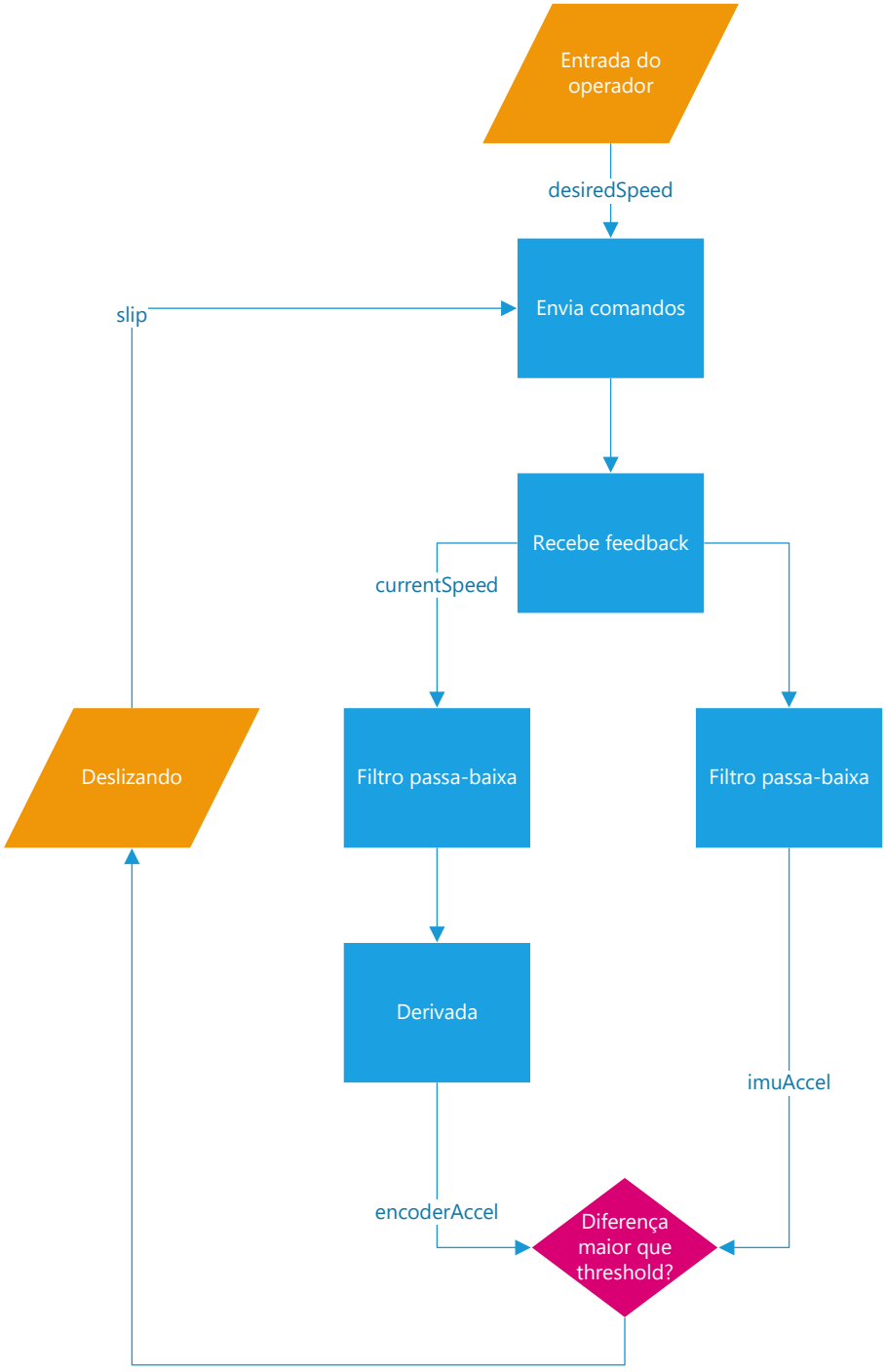

Figura 31 – Fluxograma da detecção do deslizamento

torque.

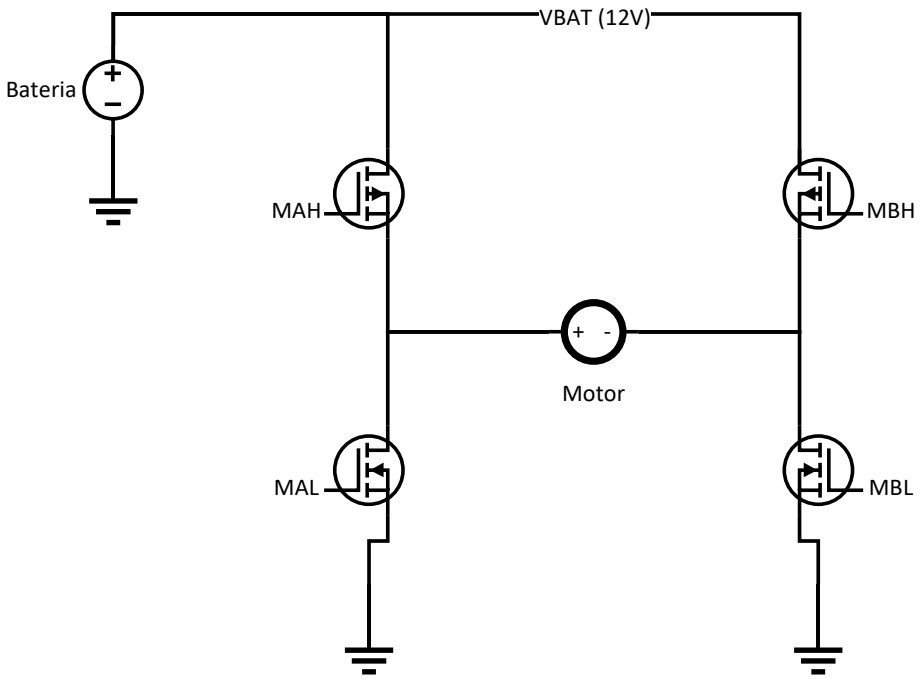

Figura 32 – Esquema elétrico de uma ponte H semelhante à utilizada no robô

6 RESULTADOS

6.1 Identificação do deslizamento

Nos gráficos 33, 34, 35, 36 e 37 temos as medidas das acelerações da roda e do chassi, bem como suas comparações. Nos experimentos, o robô começa em repouso, com velocidade $v0 = 0.0m/s$, em seguida é aplicado um degrau de velocidade de referência, com amplitudes variando entre $vf = 0.2m/s$ e $vf = 1.0m/s$. Em um terceiro momento, a velocidade de referência é novamente zerada, levando a uma frenagem do robô.

No gráfico *encoderAccel*, é medida a aceleração da roda pela segunda derivada da medida de posição do *encoder*. No gráfico *imuAccel* temos a aceleração do chassi, medida pelo sensor IMU. No gráfico *desiredSpeed* temos a velocidade de referência para o robô, indicando quando o degrau de velocidade foi aplicado ao controlador.

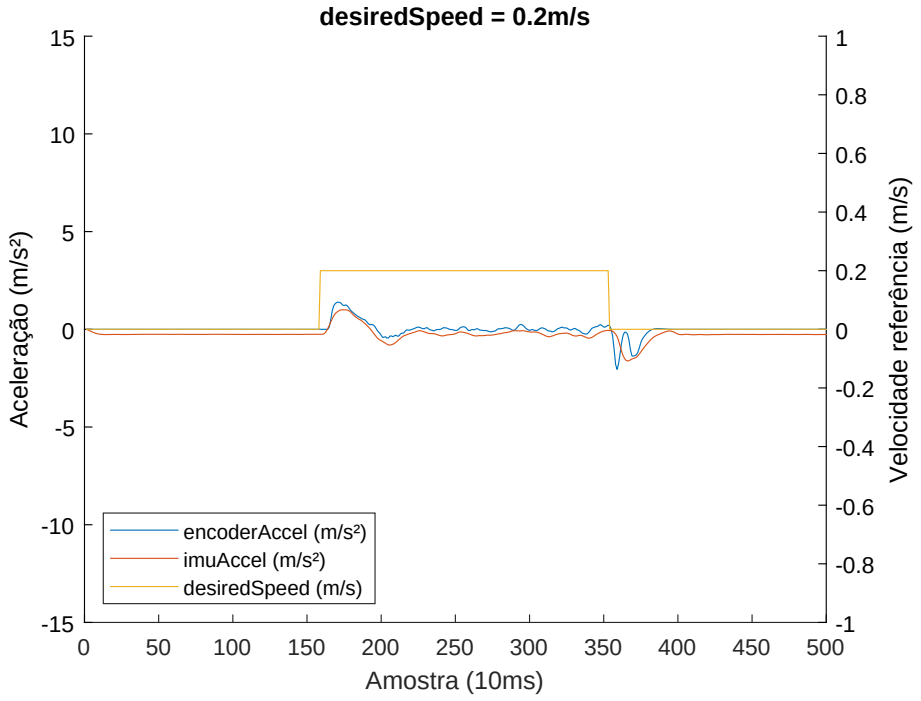

Figura 33 – v0 = 0.0, vf = 0.2

Nos experimentos dos gráficos 33 e 34, foram aplicados degraus de pequena amplitude, o que não levou a deslizamento. Isto pode ser evidenciado pela semelhança entre os gráficos de aceleração pelo *encoder* e pelo IMU que, a menos de ruídos nas medições, mediram os mesmos valores de aceleração, mostrando que o movimento da roda está solidário ao movimento do chassi, o que evidencia o não deslizamento.

Nos experimentos dos gráficos 35, 36 e 37 houve deslizamentos. Tais deslizamentos podem ser vistos pelas diferenças entre os gráficos de aceleração das rodas, medidas pelo

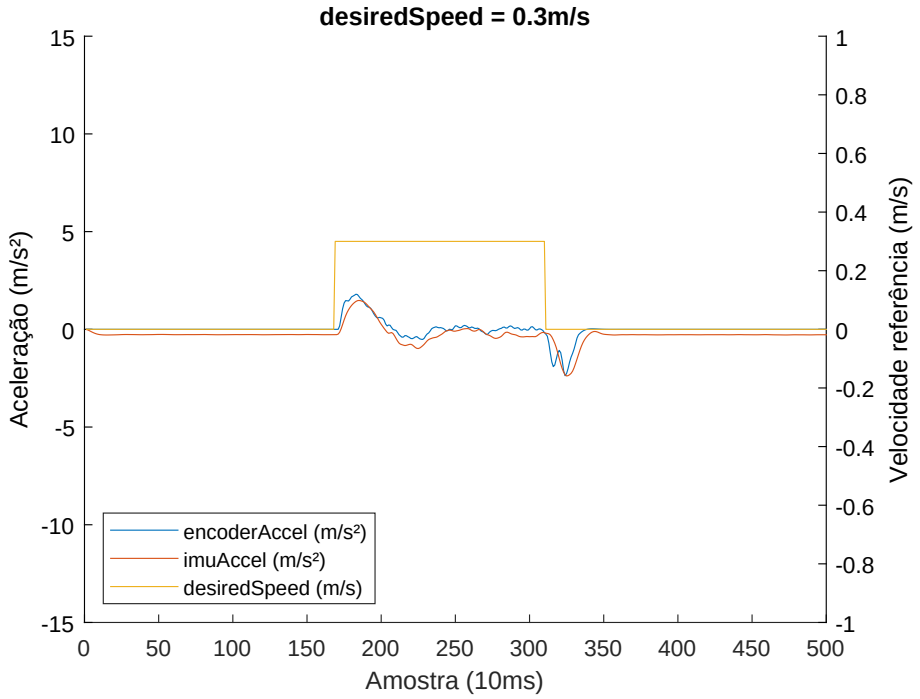

Figura 34 – v0 = 0.0, vf = 0.3

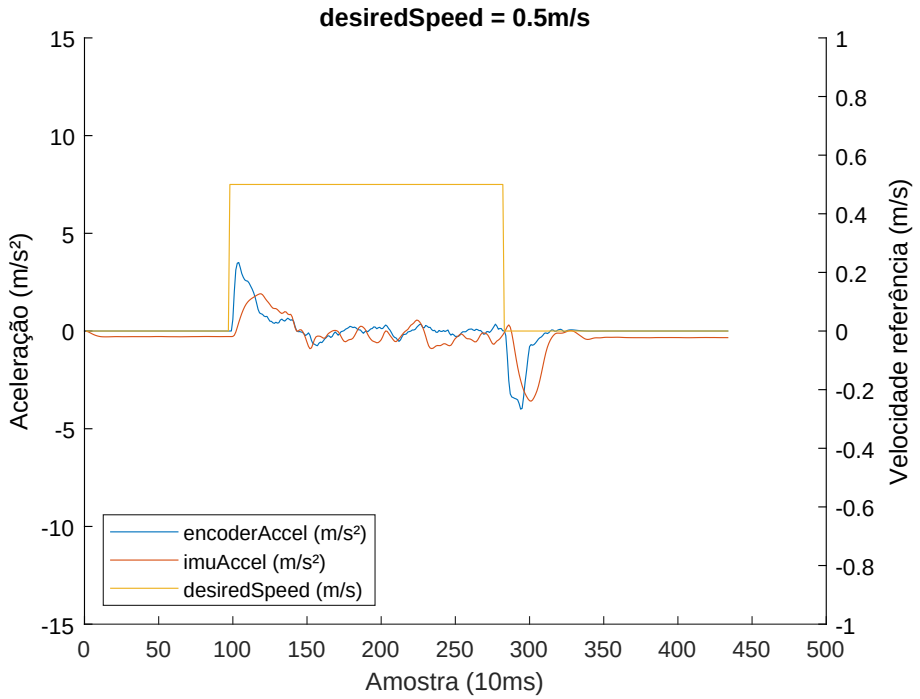

Figura 35 – v0 = 0.0, vf = 0.5

encoder, e do chassi, medidas pelo IMU. Pode-se observar que para os três experimentos a aceleração do chassi não passou de um valor limite, em torno de $3m/s^2$ e $4m/s^2$, determinado pela máxima força de atrito entre as rodas e o chão. Enquanto isso, os gráficos de aceleração das rodas chegaram a valores maiores, limitados apenas pela força máxima do motor, que é maior do que o máximo do atrito.

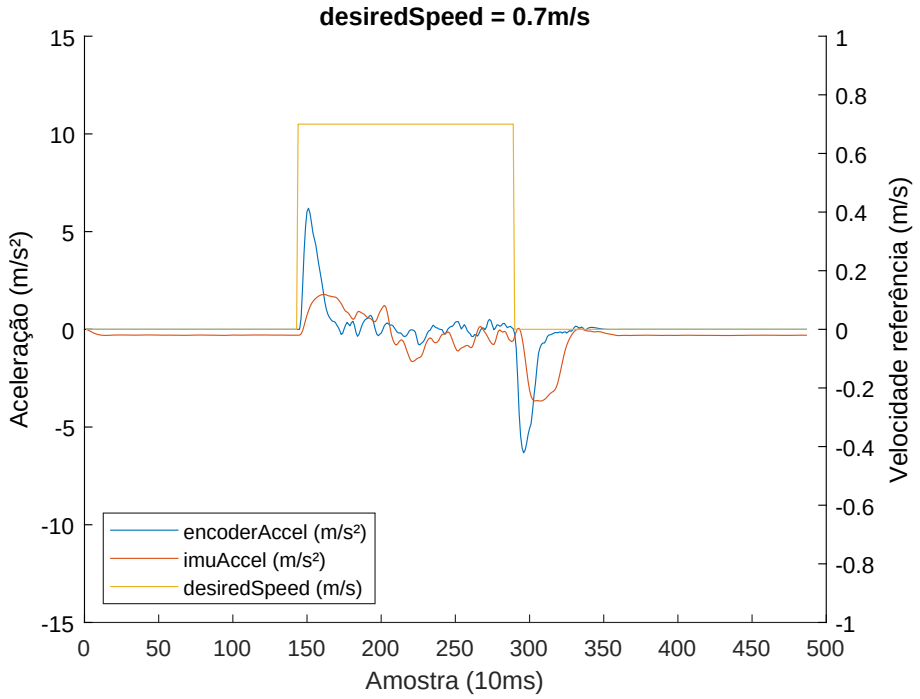

Figura 36 – v0 = 0.0, vf = 0.7

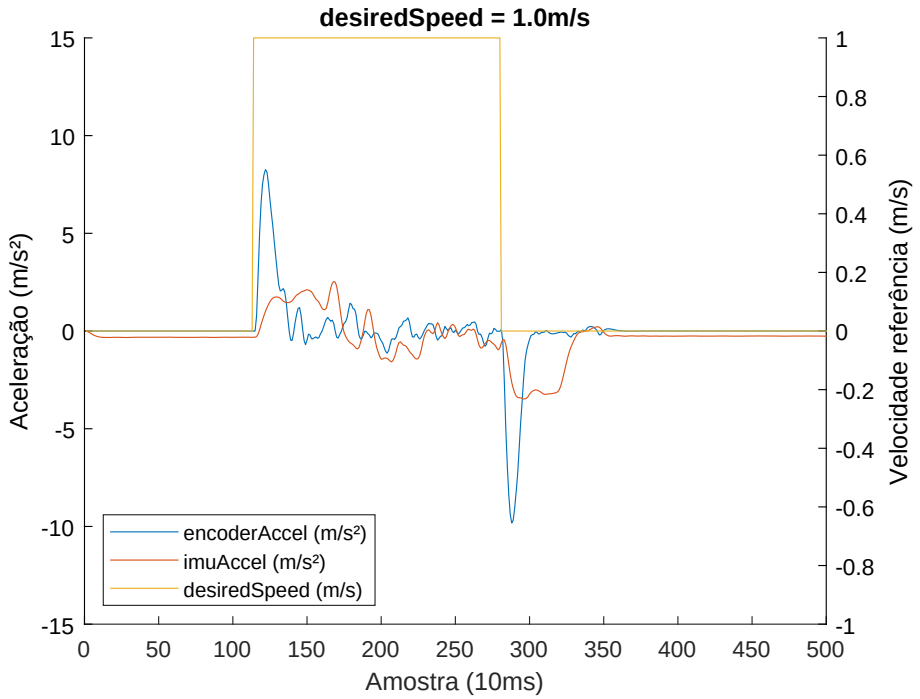

Figura 37 – v0 = 0.0, vf = 1.0

Desta forma, a fim de compensar pelas imprecisões dos sensores e aplicar correção apenas aos deslizamentos mais severos, pode-se decidir por tomar as medidas corretivas caso a diferença entre as acelerações da roda e do chassi supere $5m/s^2$.

6.2 Correção do deslizamento

Nos testes realizados com o robô modificado da figura 38, a correção aplicada apresentou resultados satisfatórios, evidenciados quando os coeficientes de atrito das duas rodas do robô de desenvolvimento são diferentes. No caso da figura 37, sem aplicação de correção o desvio da trajetória chegou a 30°, enquanto com a correção o desvio ficou em até 5°.

Foram realizados quatro testes. Nos dois primeiros, foi aplicado um degrau de velocidade de referência de $desiredSpeed = 100m/s$, fazendo com que o controlador PID fique com sua saída saturada, ou seja, $dutyCycle = 100$. No terceiro e no quarto experimento, foram aplicados degraus de $desiredSpeed = 1m/s$, o que é uma velocidade atingível pelo robô de referência sem a saturação do controlador PID.

Além disso, nos experimentos ímpares, foi colocado $threshold = 100m/s^2$, o que desabilita as medidas corretivas, pois a diferença entre as acelerações $encoderAccel$ e $imuAccel$ nunca chegarão a $100m/s^2$. Enquanto isso, nos experimentos pares, foi colocado $threshold = 2m/s^2$, habilitando as medidas corretivas caso a diferença entre $encoderAccel$ e $imuAccel$ supere $2m/s^2$.

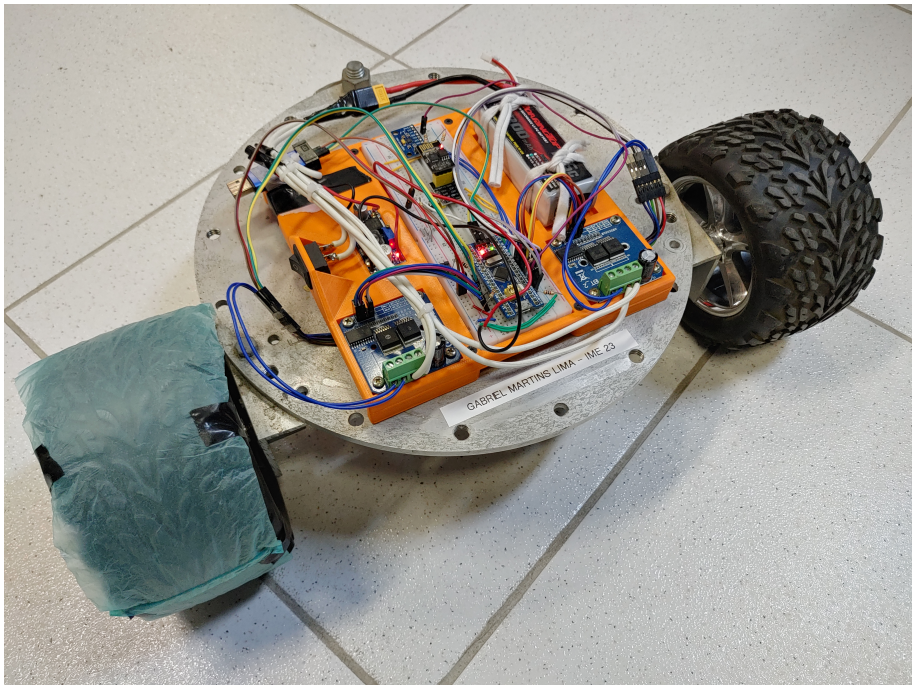

Figura 38 – Robô modificado para coeficientes de atrito diferentes

6.2.1 Primeiro experimento

O primeiro experimento, do gráfico da figura 39, mostra o caso de maior deslizamento. Foi aplicado ao motor o seu maior torque e não foi realizada nenhuma medida

corretiva. Nota-se que, logo após o degrau de subida de *desiredSpeed*, a *encoderAccel* chega a ter valores muito maiores do que a *imuAccel*. Em seguida, *encoderAccel* se aproxima novamente de zero, mostrando que a roda já chegou em sua velocidade terminal, enquanto *imuAccel* leva um tempo maior para diminuir. Isto evidencia que a roda chegou em sua velocidade terminal muito antes do chassi, o que é consequência da condição de deslizamento.

Além disso, no degrau de descida de *desiredSpeed*, nota-se que *encoderAccel* se aproximou de zero enquanto *imuAccel* ainda permanecia com magnitude significativa. Isto mostra uma condição de travamento das rodas na frenagem, o que também provocou desvio na trajetória devido aos coeficientes de atrito diferentes entre as duas rodas do robô da figura 38.

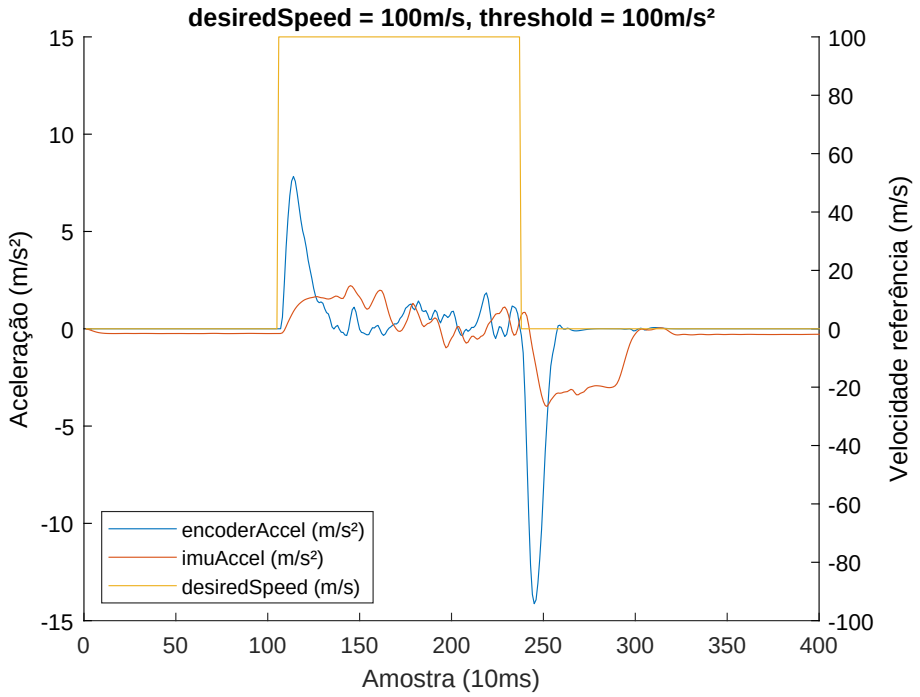

Figura 39 – Gráfico do primeiro experimento

6.2.2 Segundo experimento

O segundo experimento, do gráfico da figura 40, apresenta o caso com o motor de torque saturado, porém com aplicação de medidas corretivas. Neste caso, *encoderAccel* não atingiu valores tão altos quanto no experimento anterior, devido às medidas corretivas.

Além disso, neste experimento, *encoderAccel* e *imuAccel* se estabilizaram próximos de zero em instantes próximos, o que mostra um menor escorregamento na subida de *desiredSpeed* e um não travamento das rodas na descida.

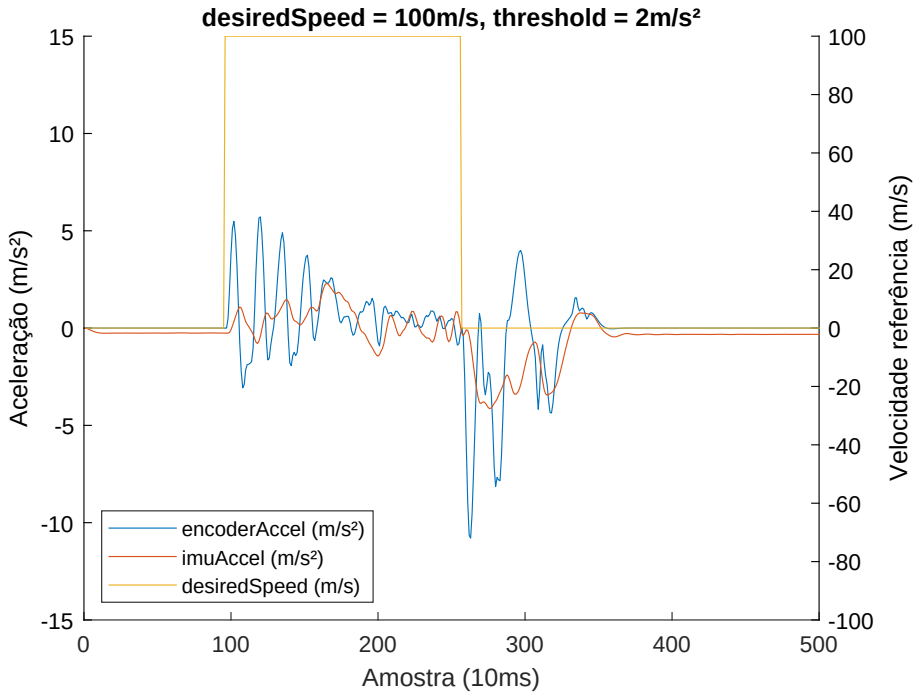

Figura 40 – Gráfico do segundo experimento

6.2.3 Terceiro experimento

O terceiro experimento, do gráfico da figura 41 foi realizado aplicando-se um degrau $desiredSpeed = 1$, o que não leva à saturação do torque dos motores. Esta é uma condição mais realista do que os experimentos anteriores, visto que o motor é dimensionado para que o torque nunca precise ser saturado, pois a saturação cria um erro de estado estacionário indesejado para a velocidade. Como não foram aplicadas medidas corretivas, a aceleração $encoderAccel$ atingiu valores elevados, e em seguida estabilizou-se próximo de zero antes de $imuAccel$, semelhante ao que ocorreu no primeiro experimento. Desta forma, fica evidente o deslizamento na largada e o travamento das rodas na frenagem, devido ao resultado parecido com o do primeiro experimento.

6.2.4 Quarto experimento

O quarto experimento, do gráfico da figura 42 representa um caso prático, por ter uma amplitude para o degrau $desiredSpeed = 1$ que, assim como no terceiro experimento, não leva à saturação do torque do motor. Diferente do terceiro experimento, neste são aplicadas medidas corretivas para o deslizamento, com $threshold = 2$. Pelo gráfico, temos uma resposta semelhante à do segundo experimento, em que o deslizamento foi corrigido tanto na subida quanto na descida de $desiredSpeed$.

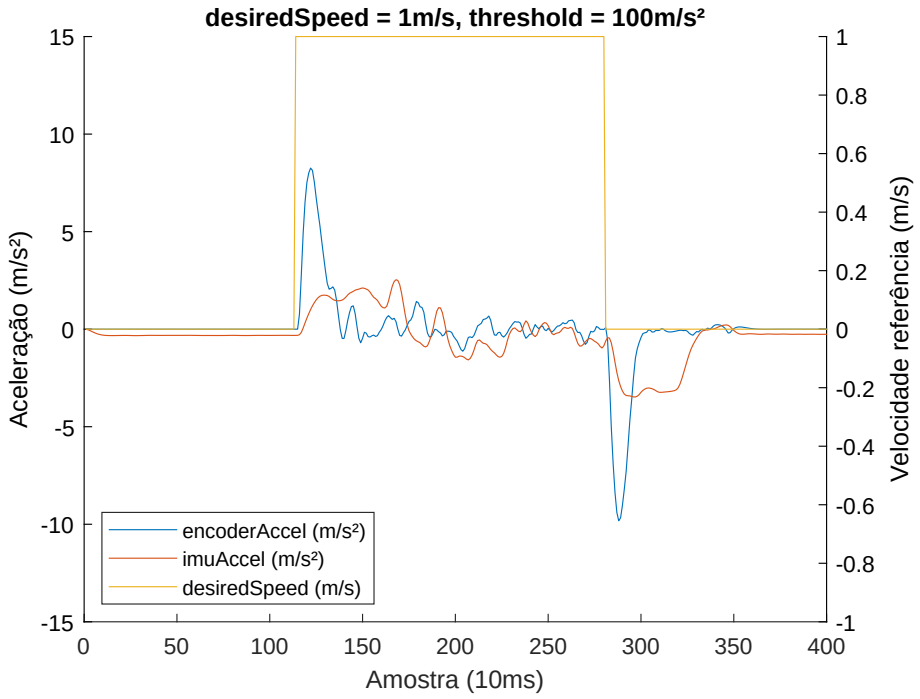

Figura 41 – Gráfico do terceiro experimento

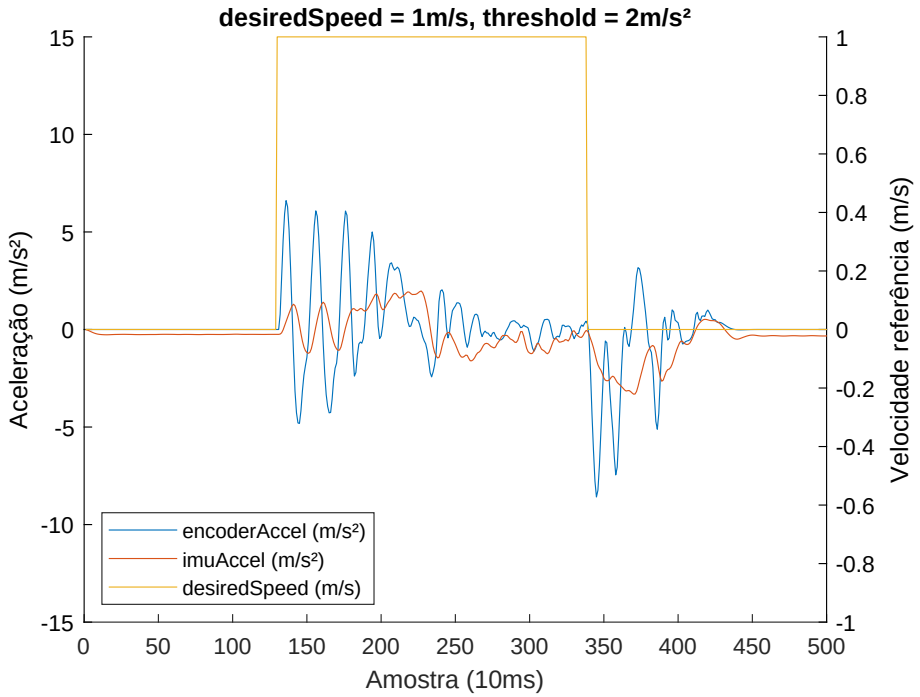

Figura 42 – Gráfico do quarto experimento

7 CONCLUSÃO

Foi desenvolvido um *firmware* capaz de ser portado para os três modelos de robô citados no capítulo 2, incluindo a otimização do controle PID para o robô de desenvolvimento.

Foram também desenvolvidas lógicas para detecção e correção de deslizamentos, testadas no robô de desenvolvimento com uma das rodas modificada para diminuição do coeficiente de atrito, o que evidencia a perda de direção provocada pelo deslizamento.

7.1 Perspectivas

- Integração ao robô do CTEx
- Solução bidimensional
- Integração ao robô da SSL (solução tridimensional)
- Melhorias nos filtros
- Implementação standalone no robô

7.1.1 Integração ao robô do CTEx

A integração ao robô do CTEx pode ser feita utilizando-se o mesmo firmware do robô de desenvolvimento, tendo em vista que ambos utilizam o mesmo microcontrolador, apenas adaptando para os diferentes pinos do microcontrolador que os componentes estão conectados. Além disso, deve-se repetir os passos tomados para a calibração das constantes do controle PID e da detecção de deslizamento.

7.1.2 Solução bidimensional

Para que a detecção e a correção dos deslizamentos também seja possível em casos de arrancadas e frenagens em curvas, é necessário adaptar os cálculos para um problema bidimensional, onde deve ser considerada a aceleração angular do movimento, não apenas a tangente.

7.1.3 Integração ao robô da SSL (solução tridimensional)

Para integrar o projeto ao robô da SSL é necessário fazer adaptações no *firmware*, tendo em vista que o robô da SSL utiliza um microcontrolador de modelo diferente, porém

do mesmo fabricante e da mesma família. Além disso, como o robô da SSL movimenta-se com rodas omnidirecionais, será necessário também adaptar os cálculos da dinâmica e da comparação, sendo necessário considerar também a aceleração normal do movimento, não só a angular e a tangente.

7.1.4 Melhorias nos filtros

Os filtros passa-baixa aplicados às leituras do IMU e do *encoder* provocaram atrasos diferentes nas medições, o que levou a alguns falsos positivos na detecção do deslizamento. Uma possível melhoria é uma melhor implementação dos filtros que faça com que eles tenham atrasos semelhantes e, preferencialmente, menores do que os atuais.

7.1.5 Implementação standalone no robô

Atualmente, A detecção e a correção do deslizamento estão implementados no programa de LabVIEW, executado no computador. A implementação das lógicas de detecção e correção, incluindo seus filtros, no robô, pode trazer reações mais rápidas do robô ao deslizamento, o que pode melhorar ainda mais os resultados das medidas corretivas.

REFERÊNCIAS

1 NAGWA. *Lesson Explainer: Direct Current Motors*. 2023. 27 set. de 2023. Disponível em: <https://www.nagwa.com/en/explainers/246108560531/>.

2 FRANKLIN, G. F.; POWELL, D.; EMAMI-NAEINI, A. *Feedback control of dynamic systems, global edition*. 8. ed. London, England: Pearson Education, 2019.

3 GABRIEL LIMA. *Repositório para o projeto de fim de curso de graduação IME 2023*. 2023. 11 out. de 2023. Disponível em: <https://git.gmlima.com/Gabriel/PFC>.

4 ROBOCUP SSL TECHNICAL COMMITTEE. *Rules of the RoboCup Small Size League - 2023*. 2023. 04 mai. de 2023. Disponível em: <https://robocup-ssl.github.io/ssl-rules/>.

5 COSENZA, C. S.; COUTO, G. C. K.; BARREIRA, L. de S.; FARIAS, L. D. P.; RODRIGUES, L. R. L.; SEGRE, J. L. L.; CASTRO, M. C.; OLIVEIRA, N. S. M. M. de; SILVEIRA, O. C. B.; SOUZA, R. P. de; BRAMIGK, V.; NIHARI, Y.; ROSA, P. F. F. Roboime: on the road to robocup 2017. 2017. 22 set. de 2023. Disponível em: <https://ssl.robocup.org/wp-content/uploads/2019/01/2017_TDP_RoboIME.pdf>.

6 HIDEAKI TAI. *Arduino library for MPU9250 Nine-Axis (Gyro + Accelerometer + Compass) MEMS MotionTracking™ Device*. 2022. 7 abr. de 2022. Disponível em: <https://github.com/hideakitai/MPU9250>.