Universidade de Brasília

Faculdade de Tecnologia

Desenvolvimento do Sistema Embarcado e Controle dos Rotores Inclináveis de um Robô Aéreo Didático

Sarah Caetano de Almeida

Bianca Glycia Boueri

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2025

Universidade de Brasília
Faculdade de Tecnologia

Desenvolvimento do Sistema Embarcado e Controle dos Rotores Inclináveis de um Robô Aéreo Didático

Sarah Caetano de Almeida
Bianca Glycia Boueri

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Geovany Araujo Borges

Brasília
2025

FICHA CATALOGRÁFICA

Almeida, Sarah Caetano de.

Desenvolvimento do Sistema Embarcado e Controle dos Rotores Inclináveis de um Robô Aéreo Didático / Sarah Caetano de Almeida; Bianca Glycia Boueri; orientador Geovany Araujo Borges. -- Brasília, 2025.

132 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2025.

1. Veículo Aéreo Não Tripulado. 2. Robô Aéreo com dois Rotores Inclináveis. 3. Software Embarcado. 4. Internet das Coisas. 5. Controle Adaptativo por Modelo de Referência. I. Boueri, Bianca Glycia. II. Borges, Geovany Araujo, orient. III. Título

Universidade de Brasília
Faculdade de Tecnologia

Desenvolvimento do Sistema Embarcado e Controle dos Rotores Inclináveis de um Robô Aéreo Didático

Sarah Caetano de Almeida
Bianca Glycia Boueri

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 20 de fevereiro de 2025:

Prof. Dr. Geovany Araujo Borges,
UnB/FT/ENE
Orientador

Prof. Dr. Adolfo Bauchspiess,
UnB/FT/ENE
Examinador interno

Prof. Dr. Guilherme Caribe de Carvalho,
UnB/FT/ENM
Examinador interno

Brasília
2025

Dedico esse trabalho primeiramente à Deus, por sempre ter me iluminado e abençoado. Aos meus pais e meu irmão, que nunca deixaram de me apoiar e, às vezes, acreditam mais em mim do que eu mesma. E ao meu futuro, que essa seja a primeira de muitas grandes conquistas.

Sarah Caetano de Almeida

Dedico este trabalho ao meu crescimento acadêmico e profissional, assim como à minha determinação em aceitar desafios.

Bianca Glycia Boueri

Agradecimentos

Agradeço primeiramente a Deus pela sabedoria, força e orientação ao longo dessa caminhada.

Agradeço aos amigos e familiares que estiveram ao meu lado durante esse período tão importante da minha vida. E agradeço especialmente meus pais, Teresa e Caetano, e meu irmão, Lucas, por terem me apoiado nos momentos de dificuldade e comemorado comigo nos momentos de felicidade. Obrigada por sempre terem acreditado em mim.

Por fim, agradeço aos anos de crescimento acadêmico, social e pessoal que a Universidade de Brasília me proporcionou, e aos amigos e professores que tive a honra de conhecer, de algum modo todos influenciaram para a conclusão desse trabalho e desse curso.

Em especial, agradeço ao orientador desse projeto, professor Geovany, por todo o tempo que teve para poder me guiar no desenvolvimento e explicar inúmeros "porquês". E, por fim, agradeço à minha parceira Bianca, por termos conseguido superar muitas situações difíceis ao longo da faculdade juntas, e se tornar uma amizade para a vida toda.

Sarah Caetano de Almeida

Agradeço ao meu esposo, Matheus, por todo o apoio durante o trabalho; à minha família; aos amigos que conheci durante o curso; aos professores que tive ao longo do curso; e aos membros dos projetos dos quais participei, que contribuíram para o conhecimento que adquiri em projetos de engenharia. Agradeço também à orientação do professor Geovany; ao Leonardo, por ter permanecido como membro da equipe Tilt-Rotor; e à minha parceira Sarah, pela nossa amizade, por sua grande empolgação, dedicação e força de vontade, e por ter sempre acreditado na nossa capacidade de concluir este trabalho.

Bianca Glycia Boueri

“Se vi mais longe, foi por estar sobre ombros de gigantes.”

(Isaac Newton)

"O importante é não parar de questionar. A curiosidade tem sua própria razão de existir."

(Albert Einstein)

Resumo

O desenvolvimento de Veículos Aéreos Não Tripulados (*VANTs*) viabiliza a execução de tarefas que requerem o uso de aeronaves, porém com menor custo e complexidade em comparação às aeronaves tripuladas. O robô aéreo com dois rotores inclináveis, ou *Tilt-Rotor*, é um *VANT* bi-rotor capaz de inclinar seus rotores para permitir decolagem e pouso verticais, e voo horizontal; assim é uma aeronave versátil e eficiente que oferece aplicações em diversas áreas da engenharia. Este trabalho apresenta uma proposta de aperfeiçoamento do robô aéreo *Tilt-Rotor* desenvolvido no Laboratório de Automação e Robótica (*LARA*), tendo como foco: o desenvolvimento de um novo sistema embarcado, com alteração do microcontrolador para um *ESP32*, e que também realiza comunicação com um sistema de aquisição de dados; além da comparação e validação, em simulação, entre diferentes técnicas para controle de velocidade dos rotores. O software embarcado desenvolvido tem os seguintes propósitos: medição da distância entre o robô aéreo e o solo; medições de aceleração linear, velocidade angular e campo magnético, para o sistema de localização; envio do sinal de controle para os motores; medição da velocidade dos motores; e comunicação com o sistema de aquisição de dados, com gerenciamento de filas circulares, por meio de troca de mensagens *JSON* entre um servidor e um cliente *TCP/IP*, pelo modo de geração de uma rede *Wi-Fi* própria. Os testes do software embarcado, integrado com o protótipo do hardware montado em placa perfurada, avaliam a precisão e exatidão das medições dos sensores, assim como se os requisitos temporais das tarefas estão sendo cumpridos, por se tratar de um sistema de tempo real. O sistema de aquisição de dados desenvolvido consiste em uma solução para a aquisição, o armazenamento e a visualização dos dados adquiridos pelos sensores do microcontrolador *ESP32*, bem como para atualização dos parâmetros das técnicas de controle nele implementadas. Por fim, o desenvolvimento, a comparação e a validação por simulação, nas ferramentas computacionais *Simulink* e *Matlab*, de duas técnicas de controle de velocidade de rotação dos motores do robô aéreo, buscam a precisão e a acurácia necessárias para que o controlador de atitude e altitude, em uma estrutura em cascata com o controle dos motores, tenha a garantia de que a velocidade por ele requisitada seja a velocidade real dos motores, independentemente das características dinâmicas do sistema. A primeira técnica consiste no controle clássico Proporcional, Integral e Derivativo (*PID*) e a segunda se fundamenta nas técnicas de Controle Adaptativo por Modelo de Referência (*MRAC*). Os resultados mostram que técnicas de controle adaptativo são mais eficientes em sistemas de dinâmica variável, como os motores do *VANT* deste trabalho.

Palavras-chave: Veículo Aéreo Não Tripulado. Robô Aéreo com dois Rotores Inclináveis. Software Embarcado. Internet das Coisas. Controle Adaptativo por Modelo de Referência.

Abstract

The development of Unmanned Aerial Vehicles (UAVs) enables the execution of tasks that require the use of aircraft with lower cost and complexity compared to manned aircraft. The aerial robot with two tiltable rotors, known as a Tilt-Rotor, is a bi-rotor UAV capable of tilting its rotors to allow vertical takeoff and landing, as well as horizontal flight. This makes it a versatile and efficient aircraft with applications in various engineering fields. This work presents an improvement proposal for the Tilt-Rotor aerial robot developed at UnB. A study of previous works was conducted, and the mechanical structure of the platform was assembled as previously defined. The focus of this work is the development of a new embedded system due to the replacement of the microcontroller with an ESP32, along with the development of a data acquisition system and the IoT functionality for communication with this system. Additionally, different techniques for rotor speed control were compared and validated through simulation. The developed embedded software has the following purposes: measuring the distance between the aerial robot and the ground; measuring linear acceleration, angular velocity, and the magnetic field for the localization system; sending control signals to the motors; measuring motor speed; and communicating with the data acquisition system. This is achieved through circular queue management and message exchange using JSON between a TCP/IP server and client, via the generation of a dedicated Wi-Fi network. The embedded software tests, integrated with the hardware prototype assembled on a perforated board, evaluate the precision and accuracy of sensor measurements, as well as whether the system meets the real-time task scheduling requirements. The developed data acquisition system provides a solution for acquiring, storing, and visualizing sensor data collected by the ESP32 microcontroller, as well as updating the control parameters implemented in it. Finally, the development, comparison, and validation of two motor speed control techniques were performed using simulation tools such as Simulink and Matlab, aiming for the precision and accuracy necessary to ensure that the requested motor speed matches the actual motor speed, regardless of the system's dynamic characteristics. The first technique consists of the classical Proportional-Integral-Derivative (PID) control, while the second is based on Model Reference Adaptive Control (MRAC). The results demonstrate that adaptive control techniques are more efficient in systems with variable dynamics, such as the motors in the UAV developed in this study.

Keywords: Unmanned Aerial Vehicle. Tilt-Rotor. Embedded Software. Internet of Things. Model Reference Adaptive Control.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

e	Força contraeletromotriz
K_d	Constante do termo derivativo do controlador PID
K_i	Constante do termo integral do controlador PID
K_p	Constante do termo proporcional do controlador PID
K_t	Constante de torque do motor CC
K_v	Constante da força contraeletromotriz do motor CC
L_a	Indutância de armadura do motor CC
R_a	Resistência de armadura do motor CC
V_a	Tensão de armadura do motor CC
J	Função de perda - *loss function*
MIT	Instituto de Tecnologia de Massachusetts
MRAC	Controle Adaptativo por Modelo de Referência
PID	Controle Proporcional, Integral e Derivativo
SI	Sistema Internacional de Unidades
T	Período de amostragem
t	Tempo
u	Ação de controle
V	Tensão

Símbolos gregos

γ	Taxa de adaptação do MRAC
ρ	Constante de proporcionalidade do redutor no motor CC
θ_1	Parâmetro 1 de adaptação do MRAC
θ_2	Parâmetro 2 de adaptação do MRAC

Sumário

1 Introdução

1.1 Motivação

O desenvolvimento de VANTs (Veículos Aéreos Não Tripulados) pode ser visto como uma tecnologia emergente que oferece benefícios significativos para diversos setores. Esses veículos possibilitam a realização de tarefas que normalmente exigiriam uma aeronave, mas com custos de produção e manutenção muito menores, além de uma complexidade reduzida em comparação com aeronaves tripuladas. Assim, a utilização de VANTs também tem se expandido em áreas como agricultura de precisão, uso recreativo, além de aplicações militares, para inspeção de infraestruturas e também missões de resgate. Ainda, esses dispositivos podem ser divididos em três grupos em relação ao modelo de asas: asas rotativas ou multirotor, asas fixas e híbridos.

- Os de asas rotativas ou multirotores utilizam a decolagem e aterrizagem na vertical, e podem variar a velocidade relativa de cada rotor para alterar o torque, possibilitando diversos tipos de movimentos. Esse tipo de VANT domina certa de 70% do mercado, levando em consideração também voos com alta estabilidade, grandes flexibilidades de manobras e consegue estabilizar pairando no ar, além de suportar maior capacidade de carga útil de peso em relação ao drone de asa fixa devido ao seu *design*.

  Porém, esses drones são limitados pela suas baterias, vulneráveis à condições climáticas, como ventos fortes, e em condições adversas de voo, é recomendável optar pelos modelos de asa fixa.

- Os VANTs de asas fixas têm a vantagem de gerar sustentação que compensam o peso da carga transportada, possuem mais estabilidade em situações meteorológicas adversas que os VANTs de asas rotativas e podem atingir maiores distâncias em um único ciclo de bateria. São amplamente utilizados em situações que necessitam de longos períodos e velocidades de voo.

  Em contrapartida, eles necessitam de maior área para pouso e decolagem, dificultando o uso em regiões urbanas ou com grande vegetação, além de serem mais difíceis para transporte, também apresentam maior custo de aquisição quando comparados com os VANTs de asa rotativa.

- Por fim, os VANTs híbridos contam com vantagens em relação aos dois tipos supracitados, eles têm a capacidade de decolagem e pouso na vertical, por terem os rotores como os drones de asas rotativas, e após a decolagem, conseguem desenvolver o voo com velocidades de cruzeiro de aviões, inclinando esses rotores. Eles reúnem então a facilidade operacional dos drones de asas rotativas com a capacidade de longos

períodos e velocidades de voo dos de asa fixa.

A figura 1.1 mostra um modelo de cada tipo de VANT:

Figura 1.1 – Diferentes tipos de VANTs de acordo com as asas

Fonte: (Bernardino; Brotherhood, 2018)

Um exemplo notável de VANT híbrido é o bi-rotor Tilt-Rotor, que tem como exemplo mostrado na figura 1.2 o modelo V-280, capaz de inclinar seus rotores para permitir decolagem e pouso verticais, de forma semelhante a um helicóptero, e transição para voo horizontal, de maneira semelhante a um avião. Essa combinação de características confere ao Tilt-Rotor uma versatilidade única, tornando-o adequado para uma variedade de aplicações, com capacidade de operar em diferentes regimes de voo lhe garantindo uma eficiência significativa, atendendo às necessidades de alta performance em diferentes cenários operacionais.

Figura 1.2 – Tilt-Rotor

Fonte: (Rempfer, 2018)

As aplicações desse tipo de aeronave podem variar muito de acordo com o cenário em que se está inserido. No âmbito militar, ela é extremamente cobiçada para resgate,

alcançar lugares inóspitos de difícil acesso e reconhecimento espacial. Além disso, há o grande interesse também em âmbitos comerciais, como por exemplo, transporte de pessoas para ilhas e locais mais afastados que atualmente só é possível o acesso por meios menos favorecidos, como a ilha de Ogasawara, distante aproximadamente 1.000 km de Tóquio. Sendo assim, as evoluções e possíveis aplicações dessa aeronave em relação aos outros VANTs têm se mostrado muito interessantes em diferentes cenários, fazendo com que seja necessário um maior desenvolvimento nos estudos dessa tecnologia.

1.2 Contextualização

O presente trabalho é a continuação dos trabalhos (Cataldi, 2017) e (Botelho, 2020), realizados na UnB. No primeiro, foi desenvolvido um protótipo de robô aéreo birrotor com projeto mecânico, eletrônico e de software, além de um controlador e uma interface com joystick. No segundo, foram realizadas otimizações na estrutura mecânica da plataforma, foi feito um modelo dinâmico da plataforma com identificação do sistema, e foi desenvolvido um controlador PID para estabilização de atitude e altitude da plataforma.

Na Figura 1.3, observa-se um diagrama da estrutura mecânica do Tilt-Rotor, conforme foi definida em (Botelho, 2020). Percebe-se que a estrutura apresenta dois rotores inclináveis com duas hélices, uma base de alumínio, pés de fibra de carbono, e um compartimento no centro onde está localizado o sistema eletrônico. Apesar de a estrutura não apresentar asas, a base foi feita como um tubo de seção retangular, assim seu formato facilita o acoplamento de asas posteriormente.

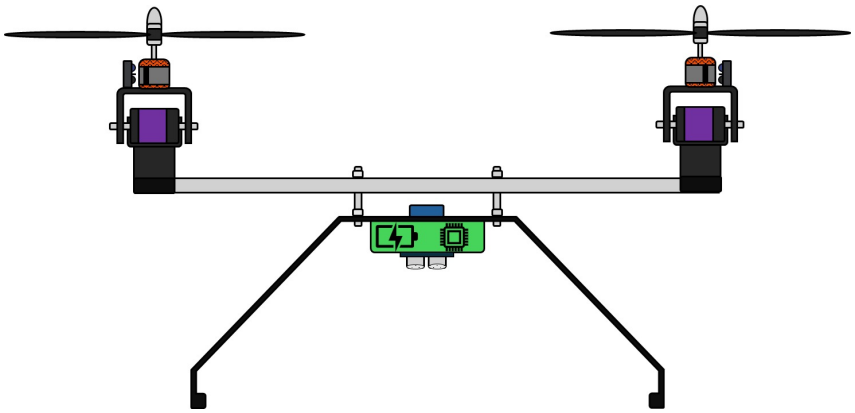


Figura 1.3 – Diagrama da estrutura mecânica do Tilt-Rotor.

Como pode ser observado na imagem, a estrutura engloba o posicionamento dos seguintes sensores e atuadores, com suas respectivas funcionalidades:

- Um sensor ultrassônico posicionado na parte inferior da estrutura, utilizado para a medição da distância entre o robô aéreo e o solo.

- Dois fotosensores posicionados próximos a cada rotor, utilizados para as medições de velocidade dos rotores.
- Uma IMU (acelerômetro, giroscópio e magnetômetro) localizada próxima do centro da estrutura, cujos dados são necessários para o sistema de localização do robô aéreo.
- Dois servo motores responsáveis pela inclinação dos rotores.
- Dois rotores com hélices, cada um controlado por um ESC (controlador eletrônico de velocidade).
- Uma bateria localizada na parte inferior da estrutura, do modelo "$Lipo\ ZIPPY\ Compact\ 2200mAh\ 3S\ 25C$", de 11,1$V$ em três células, definido em (Cataldi, 2017).

1.3 Delimitação do Problema e Objetivos

Foram observados os aprimoramentos que seriam necessários aos trabalhos anteriores. A estrutura mecânica da plataforma foi montada conforme definida nos trabalhos anteriores, e apresentou-se robusta e resistente. A Figura 1.4 é uma foto da estrutura mecânica da plataforma. Na foto, observa-se que: os quatro motores já foram fixados na estrutura; os quatro sensores estão posicionados de forma provisória, próximos ao local onde serão fixados posteriormente; e o compartimento central para o sistema eletrônico ainda não foi colocado na estrutura.

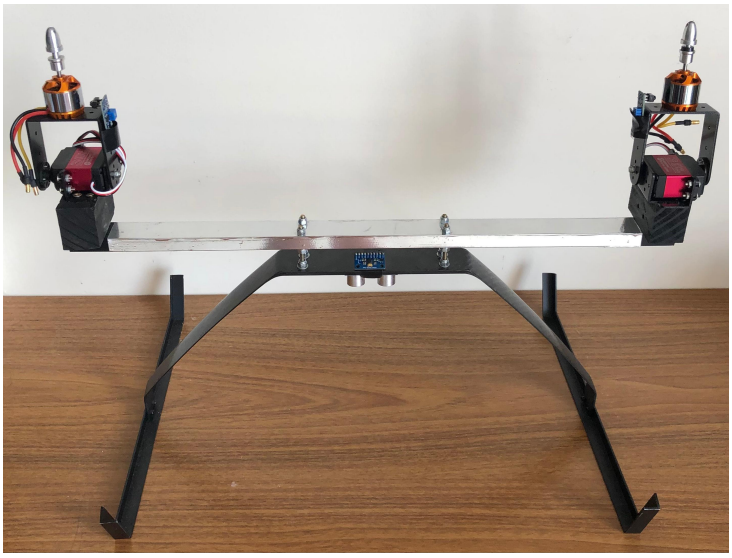


Figura 1.4 – Foto da estrutura mecânica da plataforma.

O sistema eletrônico, que utilizava um Raspberry Pi 3 Modelo B e um Arduino Pro Mini, poderia ser modificado de forma a se tornar mais adequado para a aplicação em um sistema embarcado. Percebeu-se que o Arduino Pro Mini apresentava capacidade de processamento insuficiente para a execução das ações de controle, o que comprometia sua

utilização em conjunto com as técnicas de controle. Além de que a substituição de ambos por um microcontrolador único e central consistiria em uma solução mais simples, desde que o microcontrolador fosse adequado e otimizado para sistemas embarcados. A topologia do sistema pode ser vista na Figura 1.5.

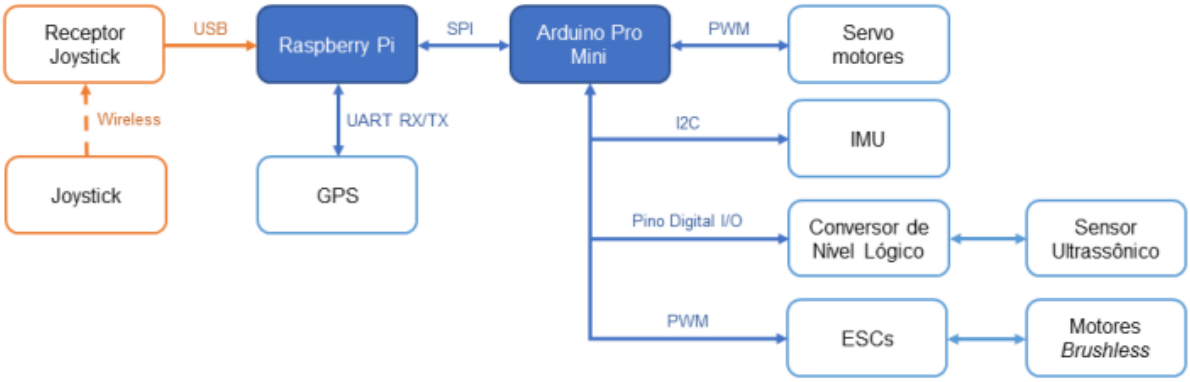


Figura 1.5 – Topologia do sistema.

Fonte: (Cataldi, 2017)

Nos trabalhos anteriores, não havia uma comunicação por protocolo entre o microcontrolador e um sistema de aquisição de dados. Um sistema com este objetivo poderia ter diversas funcionalidades durante o voo do Tilt-Rotor, como: imprimir dados relacionados ao voo em tempo real, por exemplo a velocidade dos motores; salvar os dados em arquivos, para análises posteriores; e mandar comandos para o robô aéreo, como um controle remoto. Ainda, o sistema poderia fornecer funcionalidades essenciais para a etapa de desenvolvimento do projeto, como, durante a implementação de técnicas de controle, a possibilidade de alteração de parâmetros do controlador e de visualização das respostas em forma de gráfico, em tempo real. Por isso, percebeu-se a necessidade do desenvolvimento de um sistema de aquisição de dados, com trocas de mensagens entre o microcontrolador e este sistema.

Nos controladores de atitude e altitude da plataforma, os rotores tinham apenas o controle de tensão em malha aberta oferecido pelo ESC, sem garantia de que a velocidade de cada rotor correspondesse, de fato, ao valor estabelecido pelos mesmos para a devida estabilização, ou seja, esse dispositivo não oferecia um controle de rotação. Dessa forma, tornou-se necessário o desenvolvimento de um controlador de rotação em malha fechada eficiente para os rotores, a ser utilizado em uma estrutura em cascata com o compensador de atitude e altitude, por meio do envio de um sinal de controle para o ESC. Essa abordagem assegura maior precisão e acurácia na regulação da velocidade dos rotores, contribuindo para garantir o desempenho esperado do sistema, conseguindo contornar situações como queda de bateria, flutuações de temperatura, pressão, bem como diversas outras situações eletrônicas e físicas adversas.

Neste trabalho, será apresentada uma proposta de aperfeiçoamento de um robô aéreo com dois rotores inclináveis, com foco no desenvolvimento de um novo sistema embarcado, devido à alteração do microcontrolador, juntamente com o desenvolvimento de um sistema de aquisição de dados, e da funcionalidade IoT de comunicação com este sistema, além da comparação e validação, em simulação, entre diferentes técnicas para controle de velocidade dos rotores.

1.4 Resultados Alcançados

Foi desenvolvido o software embarcado para os sensores e motores do robô aéreo, assim como para a funcionalidade de comunicação por protocolo TCP/IP. O software embarcado foi testado em conjunto com um protótipo do sistema eletrônico do robô aéreo, adaptado para o novo microcontrolador, e que foi montado em placa perfurada. Foram feitas análises de resultados experimentais do software embarcado.

Além disso, foi desenvolvido um sistema para adquirir, armazenar e analisar dados em tempo real a partir da medição dos sensores do robô aéreo, como também ser possível enviar valores de ajuste para seus controles.

Por fim, é feito o desenvolvimento e validação, por meio de simulações computacionais, de duas técnicas de controle de rotação dos motores para comparação frente às variações dinâmicas da planta apontadas por (Botelho, 2020) para a futura implementação na plataforma Tilt-Rotor com os devidos ajustes.

1.5 Organização do Trabalho

No capítulo 1, é apresentada a introdução ao presente trabalho, contando com a motivação, contextualização, delimitação de problemas e objetivos, finalizando com os resultados alcançados. No capítulo 2, é apresentada a base teórica utilizada para desenvolvimento do trabalho e compreensão do texto. No capítulo 3 são feitos os detalhamentos quanto ao desenvolvimento do trabalho, dividido em três áreas principais: o Sistema Embarcado, o Sistema de Aquisição de Dados e a Modelagem e Controle dos Motores. Assim, o capítulo 4 apresenta os resultados e discussões sobre os mesmos, validando o desenvolvimento por meio de simulações e análises de dados experimentais. Por fim, no capítulo 5 é apresentado um resumo do trabalho, reforçando os resultados principais, e finalizando com propostas para trabalhos futuros.

2 Fundamentação Teórica

2.1 Sistemas de Tempo Real

De acordo com (Oliveira, 2018), os sistemas computacionais com requisitos de tempo real são aqueles submetidos a requisitos de natureza temporal não triviais. Esses requisitos podem ser, por exemplo, prazos máximos para execução de tarefas, períodos que determinada tarefa deve ser repetida, intervalo de tempo máximo entre duas ações, entre outros. Sendo assim, o que caracteriza um sistema como sendo ou não de tempo real é a sua especificação, fazendo com que a solução (projeto) apenas precise atender aos requisitos temporais especificados. Ainda, esses sistemas precisam reagir aos estímulos do ambiente físico que os cercam, respeitando os requisitos, sendo que seus resultados devem estar corretos tanto lógica quanto temporalmente. A especificação do sistema reflete em seus requisitos temporais a dinâmica do mundo físico onde o sistema computacional está inserido.

Sob esse prisma, é importante salientar quais são os principais requisitos temporais que esses sistemas devem cumprir. Um dos tipos mais comuns é o período de cada tarefa, que deve ser desempenhada em determinados ciclos de tempo, devendo iniciar e terminar dentro de cada um deles. Outro requisito utilizado frequentemente é o prazo máximo no qual a tarefa deve terminar, podendo ser um prazo relativo à chegada dessa tarefa, ou absoluto, sendo determinado o prazo máximo para sua conclusão. Por fim, há também o requisito de atualização dos dados, sendo o maior período de tempo que o sistema deve tolerar para receber as informações oferecidas a ele, e, em geral, o sistema procura sempre tomar decisões baseadas nos dados mais recentes.

Nesse cenário, um dos principais conceitos utilizados é o de tarefa, significando a execução de um segmento de código que possui algum atributo ou restrição temporal própria, como um período ou tempo de execução máximo, e a definição de quais devem ser as tarefas do sistema cabe ao projetista, respeitando os requisitos temporais da aplicação. Quando várias tarefas disputam recursos do computador, como por exemplo o processador, sendo que estão aptas a serem executadas, com prazos a serem cumpridos, faz-se necessário que ocorra uma decisão de qual deve ser executada a cada momento. Ainda, como geralmente existem muito mais tarefas do que núcleos processadores, então mesmo com vários núcleos o problema permanece, surgindo o conceito de escalonamento.

Esse conceito está atrelado à solução para o problema de processamento, sendo que com o escalonamento, há a tomada de decisão sobre a ordem de execução das tarefas. Com isso, o escalonador é o módulo do sistema operacional responsável por gerenciar o processador e gerar a chamada escala de execução. Existem dois tipos de escalonamento quanto ao momento no qual as informações são adquiridas para basear a ordem de execução: o

escalonamento estático, no qual as decisões são tomadas em informações já disponíveis antes da execução do sistema; e o escalonamento dinâmico, no qual a tomada de decisão utiliza informações que ficam disponíveis durante a execução do sistema. Ainda, há a classificação quanto à forma pela qual o escalonador entra em ação, sendo elas: o escalonamento dirigido por eventos, no qual um evento externo gera uma interrupção, acionando o sistema operacional e acionando o escalonador para decidir qual tarefa deve ser executada; ou dirigido por tempo, no qual o único evento existente é a passagem temporal, sendo que o processador recebe apenas um tipo de interrupção do temporizador a cada período de tempo definido. Esse último é geralmente empregado em sistemas altamente críticos, e também são escalonadores conhecidos como “executivos cíclicos".

Diante disso, tarefas que devem ser executadas em períodos de tempo definidos, sendo possível calcular previamente os instantes nos quais essa tarefa se apresentará ao escalonador para execução, são classificadas como periódicas. Porém, se não há tal rigidez temporal para a execução da mesma, logo não sendo possível prever quando ela precisará ser executada, mas também existindo um intervalo mínimo para que ela seja solicitada novamente, então ela é classificada como esporádica. Por fim, se não é possível nem prever, nem se ter um limite quanto à frequência de chegadas, então ela é classificada como aperiódica.

Ainda, outro aspecto relevante em sistemas de tempo real são as chamadas seções críticas, que são todos os segmentos de código que acessam variáveis compartilhadas. O problema com essas seções na programação concorrente consiste em garantir que todas as tarefas possam executar seus códigos com variáveis compartilhadas sem interferir nas seções críticas de outras tarefas. Diante desse problema, foram desenvolvidas várias técnicas para resolvê-lo, uma delas utiliza o mecanismo de exclusão mútua, também conhecido como “mutex", que é um tipo abstrato de dados cujos atributos são um valor lógico e uma fila de tarefas bloqueadas. O primeiro registra o estado do mutex, podendo estar livre ou ocupado; já a segunda contém todas as tarefas bloqueadas esperando pela liberação do mutex. Essencialmente duas primitivas são mais importantes para um mutex: LOCK e UNLOCK, sendo que quando uma tarefa executa a primitiva LOCK em um mutex livre, este passa para o estado de ocupado e a tarefa é executada; se o mutex estiver ocupado, a tarefa é bloqueada, saindo da fila do processador e entra na fila do mutex. Caso a primitiva seja UNLOCK e o mutex estiver livre, nada ocorre; se o mutex estiver ocupado e não houver mais tarefas na sua fila, ele passa para o estado livre, mas se existirem novas tarefas nessa fila, o mutex permanece ocupado, mas libera a próxima tarefa para a fila do processador. Na primitiva UNLOCK, nenhuma tarefa é bloqueada.

Ademais, o chamado mecanismo de interrupções é um recurso associado à ocorrência de eventos que requer alguma resposta do processador, esses eventos são chamados de interrupções. Quando uma interrupção acontece, o processador termina a instrução de máquina em andamento e se volta para a execução de uma rotina específica denominada tratador de interrupção, realizando ações necessárias dada a ocorrência daquele evento.

Quando esta rotina termina, o processador retorna para as instruções de máquina seguintes àquela de quando a interrupção ocorreu. Assim, para a execução do tratador de interrupção, é necessário preservar os conteúdos dos registradores associados à rotina interrompida, para que quando esta for retomada esses conteúdos permaneçam os mesmos de antes da interrupção ocorrer.

Os computadores tem incluídos neles uma variada gama de periféricos, que são comandados por meio do controlador de periférico, sendo este um componente eletrônico que faz o intercâmbio de informações entre o processador e o componente periférico. Essas interrupções vindas de componentes físicos usam então os controladores de periféricos que geram interrupções para alertar o processador a ocorrência de eventos. Ainda, de acordo com o tipo de interrupção, definidas por números, tipicamente de 0 a 255, é possível ao processador identificar a origem da interrupção.

Por outro lado, as chamadas interrupções de sistema são geradas a partir da execução de uma instrução de máquina específica, tendo como parâmetro o número do tipo de interrupção a ser gerada. Seu efeito é semelhante à uma interrupção vinda de componentes físicos do mesmo tipo, porém o momento de ocorrência é controlado pelo programa em execução, que a diferencia da interrupção de periférico, as quais, em geral, não se consegue prever internamente ao programa quando elas irão ocorrer.

A partir desse cenário, o objetivo do algoritmo de escalonamento do processador é tanto aumentar a capacidade de processamento de dados, quanto reduzir custos associados ao sistema operacional. Esses algoritmos devem gerenciar qual das diversas tarefas aptas deve ser executada, sempre existindo pelo menos uma à espera do processador. Após essas considerações, uma das soluções para o problema de escalonamento é a atribuição de prioridades para as tarefas, e aquela que detiver a prioridade mais alta da fila ordenada, deve ser a primeira para a execução. Sendo assim, essas prioridades podem ser tanto definidas pelo próprio programador quanto pelo núcleo de processamento (*kernel*) do sistema operacional, sendo que este monitora o comportamento passado de cada tarefa e ajusta as prioridades de acordo com o período de utilização do processador, ordenando de forma que as mais rápidas sejam executadas primeiro de forma que as mais lentas não monopolizem o recurso e gere atrasos para as demais.

2.2 Transmissão de Dados - Protocolo TCP/IP

O protocolo de transmissão de dados do tipo TCP (Transmition Control Protocol, ou Protocolo de Controle de Transmissão, em tradução livre) foi projetado com a finalidade de oferecer um fluxo de bytes confiável em uma rede interligada não confiável, se configurando como um conjunto de regras de comunicação entre clientes e servidores. A rede interligada na camada de enlace possui problemas quando comparada à rede única porque podem existir diversas topologias, larguras de banda, atrasos, e vários outros parâmetros. O serviço TCP é

obtido quando uma comunicação por meio de soquetes entre o transmissor e o receptor de dados é criada (Tanenbaum; Wetherall, 2011).

Os chamados **soquetes** foram lançados inicialmente em 1983, pelos pesquisadores de Berkeley, Califórnia, Estados Unidos, que reescreveram o TCP/IP com uma nova interface de programação. Essencialmente, soquetes são "pontos finais"(também conhecidos como *endpoints*) da rede, servindo para definir os extremos da comunicação para conectar dois dispositivos. Eles são identificados de forma única a partir de uma combinação de **endereço IP** e a **porta**. Quando dois pontos finais se conectam, há a formação da chamada comunicação ponta a ponta.

Assim, o protocolo foi desenvolvido pelo Departamento de Defesa dos Estados Unidos durante a Guerra Fria (1947-1991), surgindo a partir da rede de pesquisa ARPANET, predecessora da Internet Mundial (também conhecida pelo termo em inglês, *World Wide Web*). Com o tempo, surgiu a necessidade de integração de diferentes dispositivos com seus respectivos protocolos, tendo como principal objetivo a capacidade de conectar várias redes de maneira uniforme, surgindo assim o **modelo de referência TCP/IP**.

Além disso, essa rede deveria sobreviver à perda de dispositivos físicos de sub-redes sem que as conexões entre as máquinas de origem e destino fossem interrompidas, sendo uma preocupação do Departamento de Defesa dos Estados Unidos com a até então chamada União Soviética, partindo do princípio que esta poderia interferir nas conexões a partir da destruição dos dispositivos. Por fim, mais um objetivo devido às necessidades de aplicações com diferentes requisitos foi firmado: a rede deveria ser flexível para que pudessem ocorrer diferentes trocas de dados. Finalmente, o protocolo foi desenvolvido sendo composto por 4 camadas, divergindo das 7 camadas convencionais da rede, que são as camadas de Enlace, Internet (que tem o sentido de ser uma "rede interligada", vindo do termo em inglês *interconnected network*), Transporte e Aplicação. A figura 2.1 ressalta as diferenças:

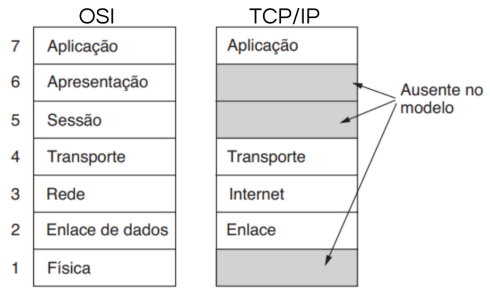


Figura 2.1 – O modelo de referência TCP/IP

Fonte: (Tanenbaum; Wetherall, 2011)

Partindo da **camada de enlace**, esta usa serviços da camada física para enviar e receber bits pelos canais de comunicação. Ela tem a função de fornecer uma interface de

serviço bem definidas à **camada de rede**, lidar com erros de transmissão e regular o fluxo de dados de tal forma que receptores lentos não recebam pacotes em excesso de transmissores rápidos. Assim, para atingir tais objetivos, essa camada recebe da camada de rede os pacotes a serem enviados e os encapsula em quadros para a transmissão. O gerenciamento desses quadros é a principal responsabilidade da camada de enlace, além de ser responsável por permitir a comunicação direta entre dispositivos, usando tecnologias de rede como Ethernet, IEEE 802.11 (também conhecido como Wifi), Bluetooth, entre outras. Nesse trabalho, foi escolhido o padrão IEEE 802.11, que é utilizado pela classe de serviço não orientado a conexões com confirmação, pois a conexão foi feita sem fios, sendo então um canal não confiável.

Em um segundo nível, há a **camada internet** que faz a integração da arquitetura, podendo corresponder à **camada de rede** no modelo de arquitetura OSI (Open Systems Interconnection, ou em tradução livre, Interconexão de Sistemas Abertos), que define as 7 camadas convencionais da rede. Essa camada tem a função de permitir que um dispositivo coloque pacotes em qualquer rede e garantir que eles trafegarão de maneira independente até o destino (que pode ser em uma rede diferente). Essa camada define um pacote oficial e os protocolos IP (*Internet Protocol*, ou em tradução livre, Protocolo de Internet) e ICMP (*Internet Control Message Protocol*, ou em tradução livre, Protocolo de Controle de Mensagens de Internet) para entregar esses pacotes IP onde são necessários.

Na **camada de transporte** subsequente ela tem por finalidade garantir que as entidades de origem e destino mantenham uma conversação, como na camada de transporte OSI. Existem dois protocolos para ela: o TCP, que é orientado a conexões confiáveis, permitindo a entrega sem erros de um fluxo de bytes com origem em determinada máquina até qualquer computador da internet; e o UDP (*User Datagram Protocol*, ou em tradução livre, Protocolo de Datagrama de Usuário), sendo um protocolo sem conexões, não confiável, muito utilizado para consultas isoladas, em aplicações que a entrega imediata é mais importante que a entrega precisa.

Nas conexões do TCP/IP ocorrem com a "negociação" de três vias, garantindo que os dados sejam entregues ordenadamente e sem perdas. Assim, para estabelecer uma conexão, o servidor aguarda passivamente por uma conexão de entrada, executando, na ordem, as primitivas de *escuta* e *aceitação*. No lado do cliente, este executa a primitiva de *conexão*, especificando o endereço IP e a porta à qual se deseja conectar. A última primitiva é responsável por enviar um segmento TCP com o bit de dado SYN ativado e um bit do sinal de ACK desativado, aguardando uma resposta. A figura 2.2 ilustra o processo de envio e recebimento de segmentos TCP.

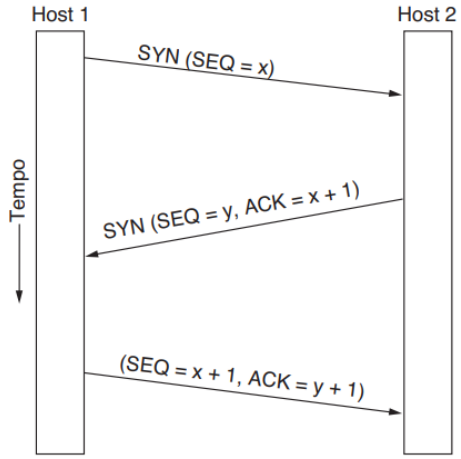


Figura 2.2 – Estabelecimento de conexão TCP em casos normais

Fonte: (Tanenbaum; Wetherall, 2011)

Quando o segmento chega no servidor, a entidade TCP verifica a existência de um processo que tenha executado a primitiva de *escuta* na porta informada pelo campo *Porta de destino*. Caso não exista o processo de *escuta*, o destino responde com a ativação do bit do sinal RST, rejeitando a conexão.

Caso contrário, se algum processo estiver na *escuta* da porta, ele receberá o segmento TCP de entrada, podendo aceitar ou recusar a conexão. Se o processo aceitar, um segmento de confirmação retorna para o cliente. Por fim, é possível perceber que o segmento SYN consome 1 byte de espaço de sequência, para que seja confirmado sem ambiguidade. Assim, a utilização de protocolos como o TCP/IP, que é organizado em camadas, é indicado para sistemas embarcados por garantir confiabilidade, ou seja, a entrega de todos os dados.

E por fim, na última camada, a chamada **camada de aplicação**, existem todos os protocolos de mais alto nível que inicialmente serviam para transferência de arquivos (FTP), protocolos de terminal virtual (TELNET) e de correio eletrônico (SMTP). Com o passar do tempo, foram incluídos os protocolos de mapeamento do nome de dispositivos para seus respectivos endereços (DNS, *Domain Name Service*), o HTTP, utilizado para busca de páginas na internet mundial, e o RTP que tem a função de entregar mídias em tempo real, como voz ou vídeo.

Por fim, faz-se necessário a citação do formato de texto de serialização de dados estruturados JSON (JavaScript Object Notation), sendo muito difundido devido à sua compactação e capacidade de fácil leitura, leveza e eficiência se comparado aos formatos XML ou YAML em termos de processamento e armazenamento, além da independência quanto aos formatos de linguagens de programação, podendo ser aplicado em sistemas bastante heterogêneos, vantagem bastante atrativa para aplicações em sistemas embarcados (Bray, 2014).

2.3 Sistemas Microcontrolados - ESP32

O microcontrolador "*ESP*32", do modelo ESP-WROOM-32 (Espressif, 2023), pode ser visto na Figura 2.3. Consiste em um microcontrolador desenvolvido e otimizado para aplicações em sistemas embarcados.

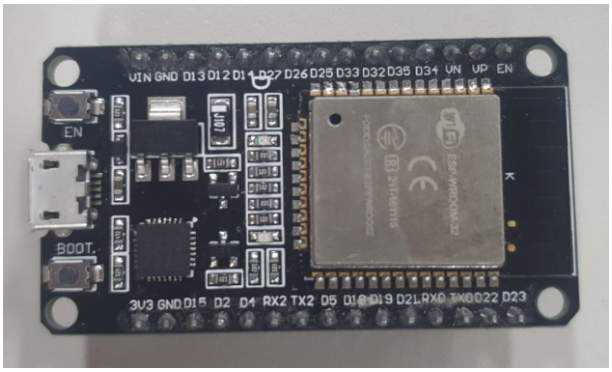


Figura 2.3 – Microcontrolador ESP-WROOM-32.

O "*ESP*32"é um chip único com suporte a Wi-Fi e Bluetooth de 2,4 GHz, projetado com a tecnologia TSMC (Taiwan Semiconductor Manufacturing Company) de 40nm, de baixo consumo de energia. Ele foi desenvolvido para alcançar o melhor desempenho em consumo de energia e frequência de rádio, oferecendo robustez, versatilidade e confiabilidade em uma ampla variedade de aplicações. O baixo consumo de energia, aliado ao uso de tecnologias avançadas de gerenciamento de energia, para que o "*ESP*32"possa alternar entre diferentes modos de consumo de energia, otimizando assim seu desempenho conforme necessário, o torna uma escolha ideal para dispositivos IoT em diversas áreas. Por exemplo, o modo de operação em que o "*ESP*32"consome menos energia é no modo chamado de "velocidade normal", no qual os seus dois núcleos rodam na frequência de 80MHz, assim o Wi-Fi e Bluetooth estão desativados ou em modo de economia de energia, mas os núcleos continuam funcionando; neste modo, o "*ESP*32"dual-core consome entre 20mA e 31mA (Espressif Systems, 2023).

O "*ESP*32"tem uma implementação exclusiva do FreeRTOS, um sistema operacional de tempo real compacto e eficiente, mas com capacidade de multiprocessamento simétrico dual-core (SMP), chamada de IDF FreeRTOS. O IDF FreeRTOS possibilita que o microcontrolador gerencie processamento em tempo real de forma eficiente em dois núcleos que operam de forma independente, com seu próprio conjunto de registradores, interrupções e tratamento de interrupções. Pode-se vincular uma tarefa a um núcleo específico, ou não vincular, assim a tarefa pode alternar entre os núcleos conforme necessário, com acesso atômico garantido no caso de núcleos diferentes solicitarem acesso a um mesmo endereço de memória (Systems, 2023).

O "*ESP*32"apresenta diversos periféricos e funcionalidades essenciais para o desenvolvimento de sistemas embarcados, os quais podem ser vistos na Figura 2.4, que mostra o diagrama de blocos das funcionalidades do "*ESP*32"(Espressif Systems, 2023). Destacam-se: a funcionalidade de servidor TCP/IP, por meio da geração de uma rede Wi-Fi própria, pelo modo Wi-Fi Access Point (AP); a funcionalidade de comunicação por protocolo I2C, eficiente para interconexão de sensores e dispositivos periféricos; a interface de comunicação serial UART, usada para monitoramento e depuração via terminal; o módulo de hardware LEDPWM, dedicado à geração de sinais PWM; e o módulo de hardware integrado MCPWM, que pode ser utilizado para leitura de sinais PWM, combinando interrupções de hardware e software.

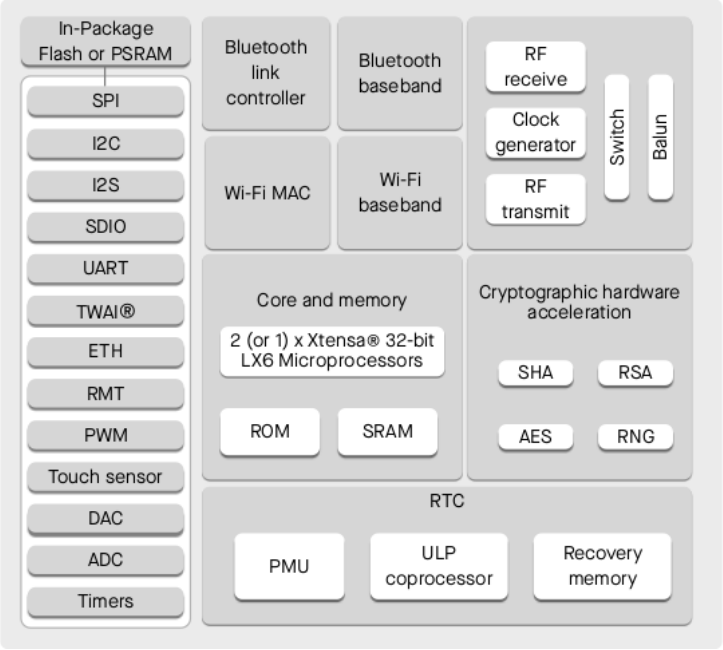


Figura 2.4 – Diagrama de blocos das funcionalidades do "*ESP*32".

Fonte: (Espressif Systems, 2023)

Para o desenvolvimento de software embarcado, pode-se utilizar a "*EspressifIDE*", um ambiente de desenvolvimento de aplicações para o "*ESP*32", mostrado na Figura 2.5, que usa o ESP-IDF (Espressif IoT Development Framework), o ambiente de desenvolvimento oficial da Espressif para o "*ESP*32". O ambiente possibilita a programação em linguagem C, otimizada para aplicações em sistemas embarcados (Espressif, 2024).

Figura 2.5 – Espressif IDE.

2.4 Sensores

2.4.1 Sensor Ultrassônico

Sensores ultrassônicos, como o "$HC-SR04$"(Morgan, 2014) que pode ser visto na Figura 2.6, são sensores que utilizam as ondas sonoras para determinação da distância entre o sensor e o objeto mais próximo a ele.

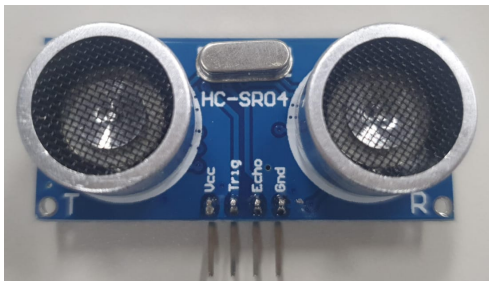


Figura 2.6 – Sensor "$HC-SR04$".

O sensor emite uma onda sonora em uma frequência específica, acima da audição humana, e aguarda essa onda refletir em um objeto e retornar, medindo o tempo entre o envio e retorno da onda sonora, como na Figura 2.7. Assim, é possível calcular, utilizando o valor da velocidade do som, a distância percorrida por meio da equação 2.1 (Morgan, 2014).

$$distancia = velocidade \cdot tempo \tag{2.1}$$

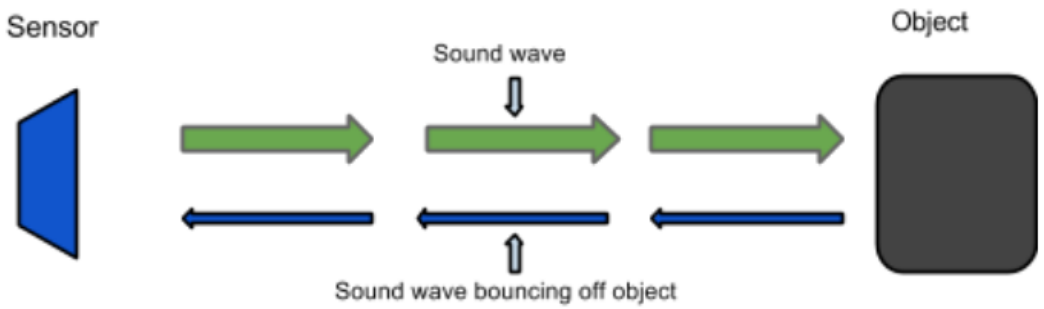


Figura 2.7 – Caminho percorrido pela onda sonora

Fonte: (Morgan, 2014)

O sensor ultrassônico apresenta uma faixa de detecção em forma de cone, pois seu feixe mantém um ângulo aproximadamente constante, o que faz com que a área de detecção aumente com a distância. Além disso, características do objeto afetam a sua capacidade de ser detectado pelo sensor. Um objeto que apresenta uma superfície plana e regular com orientação ortogonal ao sensor contribui com a reflexão da onda sonora, dessa forma o objeto é facilmente detectado. Em relação ao material do objeto, materiais acusticamente reflexivos favorecem a sua detecção, ao contrário de materiais que absorvem pressão sonora.

As especificações do sensor ultrassônico "$HC-SR04$"(Morgan, 2014) são:

- Fonte de alimentação: $5VDC$
- Corrente de operação: $15mA$
- Ângulo efetivo: $< 15°$
- Ângulo de medição: $30°$
- Faixa de distâncias medidas: 2 a $400cm$
- Resolução: $0{,}3cm$

Assim, este sensor tem quatro pinos:

- VCC: Pino de alimentação.
- $TRIG$: Pino de "gatilho".
- $ECHO$: Pino de "eco".
- GND: Pino de referência.

A utilização do sensor com um microcontrolador segue o diagrama de temporização da Figura 2.8. Inicialmente, o pino de controle "$TRIG$"deve ser configurado em nível alto por $10\mu s$, nesse momento o sensor envia uma onda sonora de oito ciclos a $40kHz$, e em seguida o pino "$ECHO$"é colocado em nível alto. O pino de dados "$ECHO$"permanece em nível alto até a onda sonora ser detectada novamente pelo sensor, então ele é colocado em

nível lógico baixo. O pino "*ECHO*"é utilizado para a medição da distância, pois o tempo no qual ele permanece em nível alto é o tempo desde que a onda sonora é enviada até quando retorna, logo este valor é utilizado na equação 2.1.

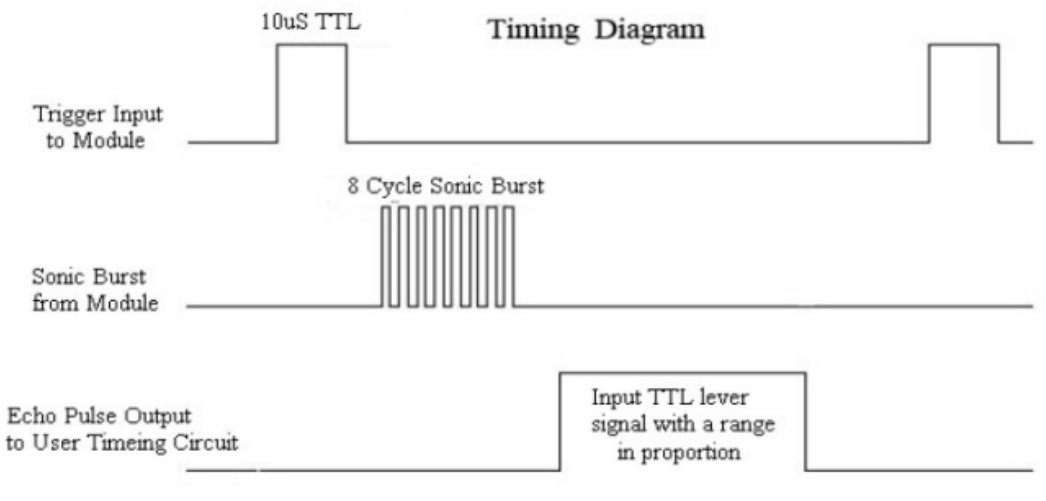


Figura 2.8 – Diagrama de temporização do sensor ultrassônico

Fonte: (Morgan, 2014)

2.4.2 Sensores Óticos

Os sensores óticos, ou fotosensores, são sensores que apresentam um emissor e um detector de luz. O emissor é uma fonte luminosa que produz feixes de luz no espectro visível ou invisível. Os sensores óticos podem ser utilizados como sensores discretos e de controle, ou seja, para detectarem o estado de um processo; assim, o emissor e o detector devem ser posicionados para que, por exemplo, o objeto a ser detectado bloqueie ou reflita o feixe luminoso na região de interesse. Também podem ser utilizados para permitirem a transmissão de um sinal elétrico entre dois circuitos isolados fisicamente, por meio dos feixes de luz (Automation, 2022).

2.4.2.1 Sensor Ótico Discreto para Aplicação em Medição de Velocidade Angular de um Rotor

O módulo "*MHSensorSeries*", com sensor óptico de reflexão "*TCRT*5000", e que pode ser visto na Figura 2.9, possui acoplado um *LED* infravermelho como emissor e um fototransistor como receptor, o qual é projetado para bloquear outras faixas de luz que não sejam a do emissor (Eletrogate, 2018).

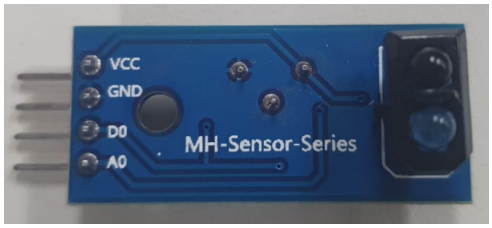


Figura 2.9 – Módulo "$MHSensorSeries/TCRT5000$".

O módulo pode ser utilizado como um sensor discreto, pelo modo de operação: luz difusa. Este modo apresenta o emissor e detector em um único corpo porém não utiliza refletores, pois a luz infravermelha é refletida, ou não, diretamente pelo objeto, como pode ser visto na Figura 2.10. O feixe de luz é focado em uma determinada extensão, cuja distância de sensibilidade é ajustável. Como a reflexão no objeto é difusa, a quantidade de luz que retorna é reduzida, por isso é necessária a utilização de lentes no receptor. Além disso, o sistema requer condições controladas, pois o material e a cor da superfície do objeto alteram a reflexão da luz, mesmo quando a distância entre o sensor e o objeto é a mesma (Automation, 2022).

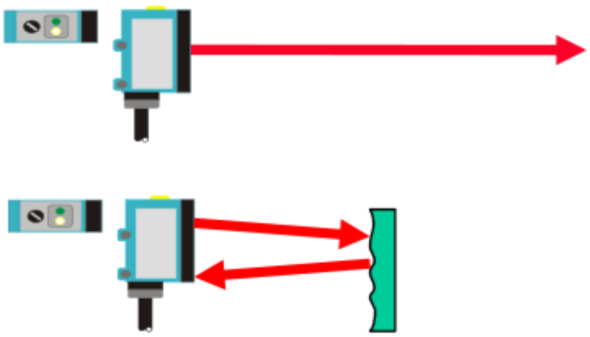


Figura 2.10 – Modo de luz difusa: a luz infravermelha é refletida, ou não, diretamente pelo objeto

Fonte: (Automation, 2022)

Esta aplicação do módulo utiliza três dos seus quatro pinos: (Eletrogate, 2018)

- VCC: Pino de alimentação, $5VDC$.
- $D0$: Pino digital.
- GND: Pino de referência.

A utilização do módulo com um microcontrolador usa o pino digital "$D0$"para a identificação da mudança de estado, pois ele é conectado em uma entrada do microcontrolador: o pino está em nível baixo, próximo de $0V$ quando o objeto reflete a luz; e está em nível alto, próximo de $4{,}3V$, quando a luz não está sendo refletida pelo objeto. A distância limite entre

o sensor e o objeto para que ocorra a mudança de estado é regulável pelo potenciômetro de ajuste do módulo.

O módulo pode, então, ser utilizado para medição da velocidade angular de um motor, de forma eficiente, uma aplicação essencial para VANT's. O rotor do motor deve apresentar uma superfície de uma única cor, lisa e reflexiva. Deve ser feita uma marcação no rotor em formato de uma linha vertical com um material não reflexivo, como um marcador permanente preto. Assim, quando o motor está em rotação, com o sensor posicionado próximo ao rotor e direcionado para ele, o pino digital "$D0$": está em nível alto no instante em que a marcação não reflexiva está sendo detectada pelo sensor; e está em nível baixo nos demais instantes. Dessa forma, é gerado um sinal que pode ser interpretado pelo microcontrolador como um sinal PWM (Modulação por Largura de Pulso), com amplitude $4{,}5V$, desconsiderando os tempos de subida e descida, e cujo tempo entre duas bordas de subida, ou seja, o período do sinal PWM, é o tempo para uma volta completa do motor. Logo, a velocidade angular pode ser calculada pela equação 2.2. Uma ilustração do sinal PWM pode ser vista na Figura 2.11.

$$velocidade\_angular(rad/s) = \frac{2 \cdot \pi}{tempo\_para\_uma\_volta\_completa(s)} \tag{2.2}$$

Figura 2.11 – Ilustração do sinal PWM.

2.4.2.2 Optoacoplamento

O optoacoplador, como o "$PC817$"(Microelectronics, 2003), é um componente eletrônico composto internamente por um LED infravermelho emissor de luz e um fototransistor, de modo que, quando um sinal luminoso é emitido, ele é captado pelo fototransistor; assim é possível enviar um sinal de controle sem contato físico entre os dois lados do optoacoplador. O isolamento galvânico entre partes de um sistema funciona como uma camada de segurança para, por exemplo, a operação de um microcontrolador em um sistema de potência.

O modo de operação deste sensor ótico, também como um sensor discreto, é: barreira de luz direta. Neste modo, o emissor e receptor estão em dois corpos separados, e são posicionados de forma a estarem alinhados, como pode ser visto na Figura 2.12; no "$PC817$", isso ocorre internamente no componente, logo, nesse caso, não há problemas relacionados

ao alinhamento. O estado lógico do sensor é determinado por: se o feixe de luz está ou não sendo detectado pelo fototransistor (Automation, 2022).

Figura 2.12 – Modo de barreira de luz direta

Fonte: (Automation, 2022)

O componente "$PC817$"(Microelectronics, 2003) apresenta um encapsulamento com quatro pinos: ânodo, cátodo, emissor e coletor; como pode ser visto na Figura 2.13.

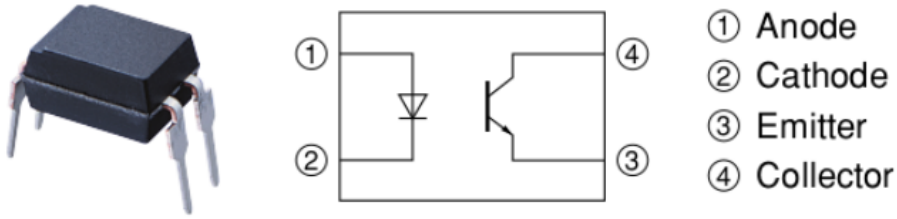


Figura 2.13 – Diagrama de conexão interna do componente "$PC817$"

Fonte: (Microelectronics, 2003)

2.4.3 Unidade de Medição Inercial (IMU)

Um dispositivo como a Unidade de Medição Inercial (IMU) "$MPU-9250$", que pode ser vista na Figura 2.14, é essencial para o sistema de localização de um VANT. É um módulo multichip que contém dois circuitos integrados em um único encapsulamento; um dos circuitos contém um acelerômetro de três eixos e um giroscópio de três eixos, enquanto o outro apresenta um magnetômetro de 3 eixos "$AK8963$". Por compartilharem um único encapsulamento, é possível um design compacto e robusto que oferece resistência a impactos de até $10000g$, ou seja, pode suportar uma aceleração instantânea equivalente a 10000 vezes a aceleração da gravidade sem sofrer danos permanentes, o que o torna ideal para aplicações em ambientes sujeitos a impactos extremos. O dispositivo simplifica a integração e a qualificação dos sensores, garantindo um desempenho otimizado (InvenSense, 2016).

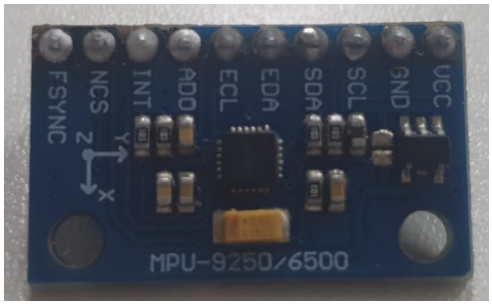


Figura 2.14 – Unidade de Medição Inercial (IMU) "$MPU-9250$".

O "$MPU-9250$"apresenta firmware de calibração em tempo real, ou seja, um software que ajusta os dados em tempo real para garantir medições mais precisas. Também tem como diferenciais: filtros digitais programáveis, sensor de temperatura integrado, interrupções configuráveis e um relógio de alta precisão (InvenSense, 2016).

O dispositivo opera com os protocolos de comunicação I^2C ($400kHz$) e SPI ($1MHz$) para comunicação com os registradores, sendo possível aumentar a velocidade para $20MHz$. O barramento de comunicação I^2C dedicado para sensores permite a fusão de dados dos 9 eixos de forma direta. Uma porta auxiliar I^2C permite, também, que ele seja conectado a sensores digitais não inerciais, como sensores de pressão (InvenSense, 2016).

O "$MPU-9250$"possui três conversores analógico-digitais (ADCs) de $16bits$ para os dados do giroscópio, do acelerômetro e do magnetômetro. Além disso, oferece configurações programáveis para as faixas de medições de cada sensor. Cada sensor tem as seguintes características: (InvenSense, 2016)

- Giroscópio:
  - Sensores digitais de taxa angular (eixos X, Y e Z) com ADCs de $16bits$.
  - Faixa programável: ±250, ±500, ±1000 e ±2000°/s.
  - Filtro passa-baixa digital programável.
  - Consumo de energia: 3,2mA em operação normal e 8μA no modo de suspensão.
  - Calibração de fábrica e autoteste.
- Acelerômetro:
  - Sensores digitais de aceleração (eixos X, Y e Z) com ADCs de $16bits$.
  - Faixa programável: ±2g, ±4g, ±8g e ±16g.
  - Interrupções programáveis.
  - Consumo de energia: 450μA em operação normal, 8,4μA a 0,98Hz no modo de baixo consumo, e 19,8μA a 31,25Hz também no modo de baixo consumo. Possui um modo que utiliza interrupção para redução do consumo.

  - Autoteste.

- Magnetômetro:
  - Sensor magnético de efeito hall (eixos X, Y e Z) com concentrador magnético. O sensor apresenta alta precisão e eficiência energética.
  - Resolução: $14 bits$ ($0{,}6 \mu T/LSB$).
  - Faixa de medição: $\pm 4800 \mu T$.
  - Consumo de energia: $280 \mu A$ em taxa de repetição de $8 Hz$.
  - Autoteste com fonte magnética interna para validação do sensor.

A utilização do "$MPU-9250$"por meio do protocolo de comunicação I^2C é possível com a conexão dos seguintes pinos: (InvenSense, 2016)

- VCC: Pino de alimentação, $3{,}3 VDC$ ($2{,}4 V$ a $3{,}6 V$)
- GND: Pino de referência.
- SCL: Pino de sincronização.
- SDA: Pino para transmissão de dados.

A implementação em software embarcado em um microcontrolador do "$MPU-9250$"exige que sejam seguidas as definições do mapa de registradores do dispositivo, disponível na documentação (InvenSense, 2015). Além disso, (Earsuit, 2018) consiste em um material complementar.

2.4.4 Exatidão e Precisão de Medições

De acordo com (Automation, 2022), a exatidão é a qualidade que determina se a medida está próxima do valor verdadeiro ou referência. É calculada pela equação 2.3.

$$exatidao[\%] = (1 - \frac{erro\_absoluto}{valor\_real}) \cdot 100 \tag{2.3}$$

A precisão é a qualidade que avalia a dispersão das medições, ou seja, o quão próximas elas estão entre si, independentemente do valor verdadeiro, assim representando a reprodutibilidade ou consistência das medições (Automation, 2022). Uma forma de medir a precisão é pelo desvio padrão amostral: quanto menor o desvio padrão amostral, maior a precisão. A relação entre exatidão e precisão pode ser vista na Figura 2.15.

Figura 2.15 – Comparação entre exatidão e precisão de medições.

Fonte: (Automation, 2022)

2.5 Motores Elétricos

2.5.1 Motor BLDC

O motor "$Turnigy\ D2830/11\ 1000KV$", que pode ser visto na Figura 2.16, é fabricado para aplicações em veículos aéreos. Ele é definido como um motor "$BLDC$", ou seja, um motor de corrente contínua sem escovas, porém não é um motor CC, como será explicado posteriormente nesta seção (Modelismo, 2024).

Figura 2.16 – Motor "$BLDC\ Turnigy\ D2830/11\ 1000KV$"e controlador eletrônico de velocidade "$30A\ BLDC\ ESC$".

O valor $1000KV$ do motor sem escovas indica que ele tem uma constante de velocidade de 1000 rotações por minuto (rpm) para cada $1V$ aplicado sem carga, ou seja, sem resistência mecânica. Isso significa que a velocidade máxima do motor depende da tensão aplicada. No caso do motor "$Turnigy\ D2830/11\ 1000KV$"com a bateria de $11{,}1V$, sua velocidade máxima

é definida pela equação 2.4 (Modelismo, 2024).

$$1000 \cdot 11{,}1 = 11100 rpm \tag{2.4}$$

De acordo com (Yu, 2011), há três classificações para motores "*BLDC*": monofásico, bifásico e trifásico; os três são motores síncronos sem escovas, a diferença está no número de fases, ou seja, no número de conjuntos de bobinas (enrolamentos) no estator. Em relação ao motor "*BLDC Turnigy D*2830/11 1000*KV*", observa-se que ele tem alimentação trifásica no estator, e não tem outra fonte de alimentação para o rotor; assim ele consiste em: um motor elétrico síncrono trifásico sem escovas e com ímãs permanentes no rotor.

O motor síncrono trifásico é uma máquina de corrente alternada (CA). Quando os enrolamentos de armadura do estator são alimentados por tensão alternada trifásica, gerando circulação da corrente na armadura do estator, cria-se um campo magnético girante no entreferro da máquina. O rotor com ímãs permanentes, e girando próximo à velocidade síncrona, tem seus polos norte e sul entrando em sincronismo com o campo magnético girante: um polo norte do rotor está em sincronismo com um polo sul do estator, e vice-versa (Chapman, 2013).

Um motor síncrono sempre está funcionando na velocidade síncrona, a velocidade do campo girante do estator. Se o motor sair do sincronismo, por conta de uma carga maior do que o torque máximo desenvolvido, o motor para. Dessa forma, sua velocidade é constante, independentemente da carga, para determinada frequência da alimentação trifásica. Considerando a velocidade mecânica de rotação em rpm, "n", a frequência elétrica do estator em Hz, "f", e o número de polos do motor, "P", a taxa fixa de rotação é dada pela equação 2.5 (Chapman, 2013).

$$n = \frac{120 \cdot f}{P} \tag{2.5}$$

A utilização de motores síncronos em aplicações que exigem controle de velocidade depende de um dispositivo como o "30*A BLDC ESC*", que também pode ser visto na Figura 2.16, um controlador eletrônico de velocidade. O dispositivo controla eletronicamente o campo magnético do estator, pelo controle da frequência. Ao fornecer sinais trifásicos controlados, ele regula a velocidade do motor de acordo com a frequência da alimentação (Digital, 2015).

No motor "*BLDC*", o rotor envolve os três enrolamentos do estator, e contém ímãs permanentes que formam pares de polos magnéticos. Os enrolamentos do estator, juntamente com os ímãs permanentes no rotor, geram uma densidade de fluxo magnético praticamente uniforme no entreferro, assim as bobinas do estator são acionadas por uma tensão CC comutada (ou chaveada) eletronicamente de uma bobina do estator para a próxima, gerando uma forma de onda de tensão AC com formato trapezoidal; por isso ele é chamado de

"*BLDC*". A comutação eletrônica utiliza um sensor, como um sensor de efeito hall ou um encoder rotativo acoplado, para identificar a posição do rotor, e determinar quando a corrente deve ser alternada (Yu, 2011). Na Figura 2.17 pode-se observar uma simplificação da seção transversal de um motor "*BLDC*"trifásico, com ímãs permanentes no rotor.

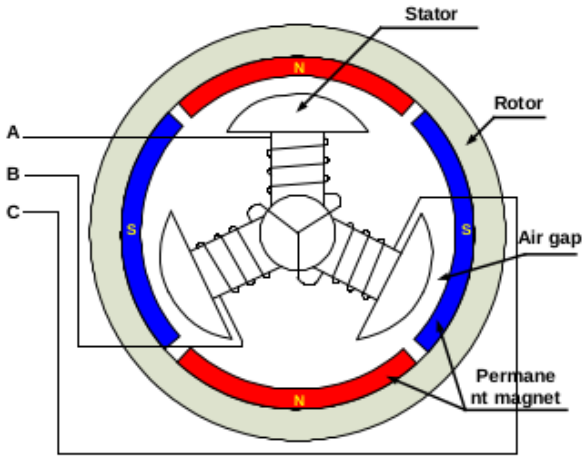


Figura 2.17 – Seção transversal de um motor "*BLDC*"trifásico, com ímãs permanentes no rotor

Fonte: (Yu, 2011)

A diferença entre o motor "*BLDC*"trifásico com ímãs permanentes no rotor e os motores síncronos trifásicos com ímãs permanentes "*PMSM*", é que o primeiro utiliza enrolamentos trapezoidais no estator, enquanto o segundo utiliza enrolamentos senoidais no estator. O motor é projetado para gerar um determinado formato de força eletromotriz ($f.e.m.$) induzida nos enrolamentos do estator, devido ao padrão de enrolamento do estator e à forma dos ímãs do rotor. No motor "*BLDC*", o formato de onda dessa tensão induzida é trapezoidal, diferentemente do motor "*PMSM*", no qual a forma da $f.e.m.$ é senoidal. O motor com comutação trapezoidal e corrente trifásica chaveada é mais simples e mais barato, porém menos eficiente, do que os motores com forma da $f.e.m.$ senoidal, os quais apresentam um método de controle diferente (Yu, 2011).

A Figura 2.18 mostra a rotação de um motor "*BLDC*"trifásico com imãs permanentes; a operação do motor é baseada na atração ou repulsão entre polos magnéticos. Quando o estator recebe alimentação trifásica, inicialmente a corrente flui através de um dos três enrolamentos do estator, gerando um polo magnético que atrai o ímã permanente de polo oposto mais próximo. O rotor se move quando a corrente é deslocada para o enrolamento adjacente. Energizando-se cada enrolamento sequencialmente, ou seja, ativando-se cada uma das três fases sequencialmente, faz com que o rotor gire seguindo o campo girante. O torque desenvolvido depende da amplitude da corrente, do número de espiras nos enrolamentos do estator, da força e tamanho dos ímãs permanentes, do tamanho do entreferro entre o rotor e os enrolamentos e do comprimento do braço rotativo (Yu, 2011).

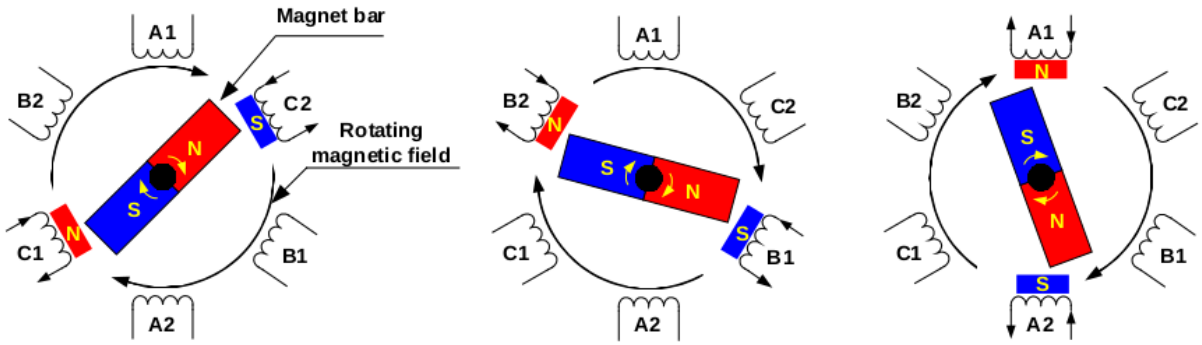


Figura 2.18 – Rotação de um motor "*BLDC*"trifásico com imãs permanentes

Fonte: (Yu, 2011)

A aplicação do motor "*BLDC Turnigy D*2830/11 1000*KV*"com o controlador eletrônico de velocidade "30*A BLDC ESC*", juntamente com um microcontrolador, utiliza as seguintes conexões do controlador eletrônico (Digital, 2015):

- Conector para ligação direta com a fonte de alimentação: contendo um fio vermelho, entre 7,4 e 14,8V; e um fio preto, para referência.
- Conexões do motor *BLDC*: três fios azuis, são a alimentação trifásica do estator.
- Conexões com o microcontrolador: um fio branco, para o sinal de controle PWM que determina a velocidade do motor; um fio vermelho, para uma saída de alimentação de 5V e 2A; e um fio preto, para referência.

Neste trabalho, será desenvolvido um controlador para a planta que consiste no conjunto composto pelo motor "*BLDC Turnigy D*2830/11 1000*KV*"e pelo dispositivo controlador eletrônico de velocidade "30*A BLDC ESC*"; e foi mostrado que a determinação da velocidade do motor é feita pelo fio de conexão entre o microcontrolador e o dispositivo controlador eletrônico, que consiste em um sinal de controle "*PWM*". Dessa forma, para a modelagem da planta, foi considerada a seguinte abstração: a saída do controlador é um sinal de tensão, entre 0V e 12V, proporcional à velocidade angular desejada para o motor. O motivo da abstração é fazer referência à bateria de 12V. A conversão da saída de controle para o sinal "*PWM*"efetivamente enviado será mostrada na seção 3.1.4, com foco no desenvolvimento.

2.5.2 Servo Motor

O servo motor "*DS*3230", que pode ser visto na Figura 2.19, é ideal para aplicações que exigem controle preciso da posição de saída, a qual para este modelo é entre 0° e 270°. Ele não necessita de um "*driver*"externo, assim, o eixo pode ser movido, no sentido anti-horário, para o ângulo desejado por meio de um sinal "*PWM*"vindo de um pino de saída digital do microcontrolador. Este modelo é um servo motor de alta qualidade, e é adequado para

aplicações de alto torque e velocidade, por ser capaz de gerar torque de até $30kgf \cdot cm$. Ele tem engrenagens metálicas, de cobre e alumínio, e uma construção classificada como "$IP66$", para proteção contra água e poeira (Probots, 2024).

Figura 2.19 – Servo motor "$DS3230$".

As suas especificações são (Probots, 2024):

- Tensão de operação: 4,8 6,8VDC
- Torque: 39,5$kgf \cdot cm$ para 5V, 34,5$kgf \cdot cm$ para 6,8V
- Velocidade: 0,2$seg/60°$ para 5V, 0,17$seg/60°$ para 6,8V
- Banda morta do sinal: 3μs
- Peso: 58$g \pm 2g$
- Frequência de trabalho: 1520$\mu s/333Hz$

A sua pinagem é composta por três fios:

- Fio branco: sinal de controle "PWM", com amplitude entre 3,3 e 5V, e frequência entre 50 e 330Hz
- Fio vermelho: para alimentação
- Fio preto: para referência

Para a utilização do servo motor, é necessário que sua alimentação seja realizada por um componente regulador, como o modelo "$Xl4015$ 5A DC DC"que tem ajuste de corrente e tensão, por dois potenciômetros "$trimpots$"(Robótica, 2023). Este regulador tem as seguintes especificações:

- Tensão de entrada: 5V a 32VDC
- Tensão de saída: 1,25 a 30VDC
- Corrente de saída máxima: 5A

- Eficiência máxima de conversão: 95%
- Frequência de comutação: $300kHz$
- Escala de corrente: ±0.5%
- Escala de tensão: ±2.5%
- Proteção contra curto circuito: sim
- Proteção reversa da polaridade da entrada: não

Assim, o regulador pode ser conectado a uma bateria de $12V$, para fornecer $6V$ para o servo motor, com o limite máximo da corrente. A tensão de alimentação do servo motor pode ser ajustada, se necessário.

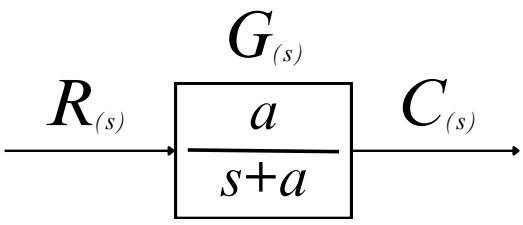
O microcontrolador, então, gera o sinal de comando $"PWM"$para o servo motor. O que determina a angulação da saída é o tempo do pulso em nível alto, e não a frequência do sinal. A largura de pulso de $0{,}5ms$ é a angulação mínima, 0°; enquanto a largura de pulso de $2{,}5ms$ é a angulação máxima, 270°. Para a maior frequência, $330Hz$, a angulação entre 0° e 45°, e com a resolução do sinal de $13bits$, é possível o controle da angulação do servo motor com passos de 0,05°.

2.6 Estratégias de Controle

2.6.1 Função de Transferência de Primeira Ordem

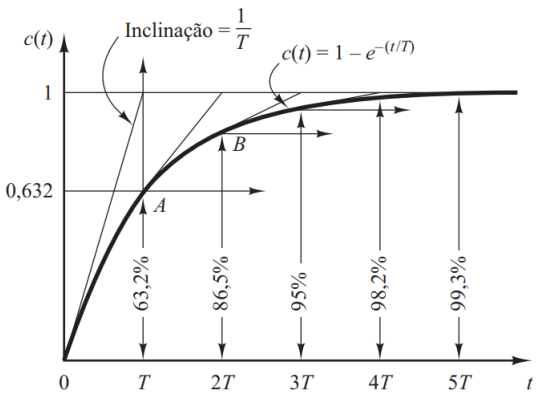
Sistemas de primeira ordem são caracterizados por apresentarem apenas um polo em $-a$ como mostrado na figura 2.20, e a característica da constante de tempo da resposta ao degrau desses sistemas é destacada na figura 2.21, como será desenvolvida a discussão a seguir.

Figura 2.20 – Diagrama de blocos de sistemas de 1 ordem



Fonte: Adaptado de (Nise, 2012)

Figura 2.21 – Resposta temporal de sistemas de 1 ordem



Fonte: (Ogata, 2010)

Para a resposta ao degrau $R_{(s)} = \frac{1}{s}$, é obtida a equação equação (2.6):

$$C_{(s)} = \frac{a}{s(s+a)} \tag{2.6}$$

Aplicando a transformada inversa de Laplace em $C_{(s)}$, tem-se a equação no tempo como:

$$c_{(t)} = 1 - e^{-at} \tag{2.7}$$

Assim, é possível fazer várias análises quanto ao seu desempenho, como por exemplo, do tempo de subida, tempo de acomodação, bem como a constante de tempo do mesmo é relacionada à esse polo como mostra a equação (2.8):

$$\tau = \frac{1}{a} \tag{2.8}$$

A constante de tempo pode ser entendida como o tempo para que a curva e^{-at} decaia 37% do seu valor inicial, alternativamente, pode-se entendê-la como o tempo para que a mesma atinja 63% do seu valor final (Nise, 2012). Logo, quanto mais longe o polo $-a$ estiver do eixo imaginário $j\omega$ no plano s, então a constante de tempo τ será proporcionalmente menor, e então uma característica interessante do sistema será sua grande velocidade de resposta, um exemplo de sistemas assim são sistemas elétricos. Em contrapartida, sistemas térmicos ou mecânicos são caracterizados por terem altas constantes de tempo e tempos de respostas muito maiores.

Outro parâmetro de desempenho da resposta temporal importante para análises é o tempo de subida (t_r) da curva, sendo o tempo necessário para que a resposta passe de 10 a 90% ou de 0% a 100% do valor final. Para sistemas de primeira ordem, é mais comum a utilização do primeiro critério, já para sistemas subamortecidos, o segundo é mais relevante.

Por fim, o tempo de acomodação (t_s) tem uma importante influência para análise do sistema, sendo o tempo necessário para que a curva de resposta alcance valores em uma faixa (2% ou 5%) em torno do valor final e permanecendo nela indefinidamente. Esse parâmetro está relacionado a maior constante de tempo do sistema de controle, podendo ser possível determinar a porcentagem mais adequada a ser utilizada no critério de erro, a partir dos objetivos do projeto (Ogata, 2010).

2.6.2 Discretização

Com o objetivo de implementar um controle contínuo em dispositivos como os microcontroladores, faz-se necessário a conversão de sinais contínuos em sinais discretos, pois esses dispositivos processam informações em formato digital. Nesse contexto, a dinâmica de um sistema de controle discreto é descrita por equações de diferenças, cuja solução determina a resposta do sistema a uma dada entrada. Assim, a Transformada Z surge como

um método de solução para as equações de diferenças, transformando-as em expressões algébricas que facilitam a análise e implementação do controle.

Considerando a transformada Z de uma função temporal $x_{(t)}$, então apenas os valores amostrados $x_{(0)}$, $x_{(T)}$, $x_{(2T)}$,... são considerandos, tomando T como sendo o período de amostragem do sinal. Logo, tem-se a definição dessa transformada de acordo com (Ogata, 1995):

Definição 2.1. A transformada Z de um sinal $x_{(t)}$, onde $t \geq 0$, ou $x_{(kT)}$, onde $k \in \mathbb{N}$ é definida como sendo:

$$X_{(z)} = \mathcal{Z}\left[x_{(t)}\right] = \mathcal{Z}\left[x_{(kT)}\right] = \sum_{k=0}^{\infty}\left[x_{(kT)}\right] z^{-k}$$

Para uma sequência de números $x_{(k)}$, a transformada Z é definida como sendo:

$$X_{(z)} = \mathcal{Z}\left[x_{(k)}\right] = \sum_{k=0}^{\infty}\left[x_{(k)}\right] z^{-k}$$

Se o sinal contínuo $x_{(t)}$ é amostrado com impulsos, matematicamente o sinal pode ser representado pela equação (2.9):

$$x^*_{(t)} = \sum_{k=0}^{\infty} x_{(t)}\delta_{(t-kT)} \tag{2.9}$$

A transformada de Laplace do sinal amostrado por impulso $x^*_{(t)}$ é tomada como sendo igual à transformada Z do sinal $x_{(t)}$ se $e^{Ts} = z$ (Ogata, 1995).

Assim, para se ter sinais amostrados, é preciso a utilização de *amostradores*, que convertem sinais contínuos em um trem de impulsos que ocorrem nos instantes de amostragem $t = 0, T, 2T...$, em que T é o período de amostragem. Depois são usados os *seguradores*, que fazem o papel de reconstrução de um sinal amostrado para um sinal contínuo a partir das amostras, com o objetivo de reconstruir o sinal de entrada do amostrador. Assim, os seguradores podem ser de n ordens, que utilizam os $n + 1$ impulsos para gerar o sinal resultante. A acurácia da reconstrução do sinal entre duas amostras aumenta à medida que a quantidade de amostras utilizadas aumenta. Porém, isso diminui a estabilidade de sistemas em malha fechada, devido ao atraso que esses cálculos acarretam, podendo levá-los à instabilidade em alguns casos. Assim, o segurador mais simples é o Segurador de Ordem Zero (também chamado de ZOH - *Zero Order Hold*) e é representado pela equação (2.10):

$$h_{(kT+t)} = x_{(kT)} \quad | \quad 0 \leq t < T \tag{2.10}$$

Ou seja, esse circuito preserva a amplitude da última amostra até a próxima, fazendo com que sua saída seja uma função escada. Em (Ogata, 1995) é demostrado que o ZOH é representado pela função de transferência mostrada na equação (2.11):

$$G_{h0}(s) = \frac{1 - e^{-Ts}}{s} \tag{2.11}$$

Com isso, é demonstrado que um sinal tem sua transformada Z como segue na equação (2.12):

$$X_{(z)} = \mathcal{Z}\left[X_{(s)}\right] = (1 - z^{-1})\mathcal{Z}\left[\frac{G_{(s)}}{s}\right] \tag{2.12}$$

Por fim, para se ter amostragens de sinais sem perder informações do sinal original, é preciso respeitar também o teorema 2.1:

Teorema 2.1 (Teorema da Amostragem (Nyquist)). Se ω_s definida como $\frac{2\pi}{T}$, em que T é o período de amostragem do sinal é maior que $2\omega_1$ ou

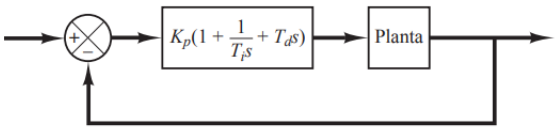
$$\omega_s > 2\omega_1 \tag{2.13}$$

Em que ω_1 representa a frequência mais alta presente no sinal contínuo $x_{(t)}$, então esse sinal pode ser reconstruído completamente pelo sinal amostrado $x^*_{(t)}$.

2.6.3 Controle PID

O controlador PID, que implementa as ações de controle Proporcional, Integral e Derivativa, é um dos mais usados e difundidos na indústria e mais da metade dos controladores industriais em uso atualmente emprega esquemas de controle PID ou PID modificado. A figura 2.22 mostra o diagrama de blocos desse compensador:

Figura 2.22 – Controle PID



Fonte: (Ogata, 2010)

Nas seções 2.6.3.1, 2.6.3.2 e 2.6.3.3 há uma breve explicação sobre cada ação e sua respectiva influência no sistema. Ainda, esse controlador calcula um erro, $e_{(t)}$, como sendo a diferença entre a variável de saída da planta $y_{(t)}$ e a variável de entrada desejada $r_{(t)}$. A equação (2.14) mostra o cálculo do erro:

$$e_{(t)} = r_{(t)} - y_{(t)} \tag{2.14}$$

A utilidade dos controles PID está na sua aplicabilidade geral à maioria dos sistemas de controle, quando se tem ambientes com dinâmica invariável, que garantem respostas idênticas sempre que são submetidos à mesma entrada, como impulso, degrau ou rampa. Na prática, existem os esquemas básicos de controle PID e PID modificados mostram que

têm um desempenho de controle satisfatório, embora em muitas situações eles possam não proporcionar controle ótimo. Assim, como a maioria desses controladores é ajustada em campo, diferentes regras de sintonia vêm sendo propostas na literatura, possibilitando ajustes finos *in loco*. Não obstante, métodos de sintonia automática vêm sendo desenvolvidos, e alguns controladores PID têm a capacidade de fazer sintonia automática on-line (Ogata, 2010). Essas sintonias podem ser feita por meio de diferentes maneiras, como o método tradicional do Lugar Geométrico das Raízes, que é descrito na seção 2.6.3.4, por exemplo.

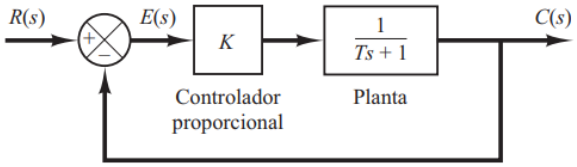
2.6.3.1 Ação Proporcional do controlador PID

A ação proporcional é uma das mais simples utilizadas, pois o controle é proporcional ao sinal de erro entre a entrada de referência e a saída do sistema, como mostra a equação (2.15):

$$u_{(t)} = K_p e_{(t)} \xrightarrow{\mathcal{L}} U_{(s)} = K_p E_{(s)} \tag{2.15}$$

Portanto, essa ação é eficaz quando utilizada para a rápida redução da diferença entre a referência e a saída do sistema enquanto o sistema ainda está sendo ajustado. Porém, para uma planta que não contenha um integrador ($\frac{1}{s}$), essa ação de controle sempre leva a um erro no regime estacionário, conhecido como erro residual , e quanto maior o valor do ganho proporcional K_p, menor é o erro. Como demonstrado por (Ogata, 2010), o erro estacionário representado pelo Teorema do Valor Final para um sistema de primeira ordem representado pela figura 2.23 é mostrado na equação (2.16):

Figura 2.23 – Controle Proporcional



Fonte: (Ogata, 2010)

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \frac{1}{K+1} \tag{2.16}$$

Um alto ganho proporcional resulta em uma grande variação na saída para uma dada variação no erro, o que pode acelerar a resposta do sistema. No entanto, isso também pode levar à instabilidade e a oscilações indesejadas. Por outro lado, ganhos proporcionais pequenos resultam em uma resposta mais suave e lenta, fazendo o sistema reagir languidamente a variações no erro.

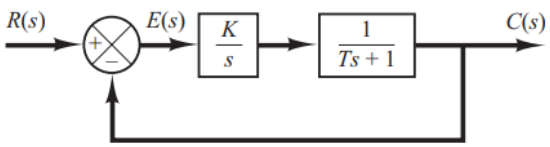
2.6.3.2 Ação Integral do controlador PID

Essa ação de controle é representada por somar a diferença instantânea ao longo do tempo e a integral retornar o erro residual acumulado que deveria ter sido corrigido, além de acelerar o movimento do processo. Essa ação é representada na equação (2.17):

$$u_{(t)} = K_i \int_0^t e_{(\tau)}\, d\tau \xrightarrow{\mathcal{L}} U(s) = \frac{K_i}{s} E_{(s)} \tag{2.17}$$

A principal contribuição dessa ação de controle é a eliminação do erro em regime estacionário, como mostra a equação (2.18), representando um avanço em relação ao controle proporcional puro (Ogata, 2010). Um esquemático em diagrama de blocos é mostrado na figura 2.24:

Figura 2.24 – Controle Integral



Fonte: (Ogata, 2010)

$$e_{ss} = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE_{(s)} = 0 \tag{2.18}$$

Por causa do integrador, a função de transferência de malha fechada do sistema contém um termo quadrático em s no numerador, fazendo o limite, obtém-se um erro nulo em regime estacionário. É comum a utilização de controladores PI, em que a ação de controle é proporcional ao erro como também à integral dele. Isso pode ser ilustrado no cenário do aumento do ganho de K_p, fazendo o sistema tender a oscilação à medida que o erro aumenta, logo se configura como um sistema de erro não-nulo. Assim, a ação integrativa se torna complementar à ação proporcional em anular o erro em regime permanente, não permitindo a geração da instabilidade oscilatória devido ao alto ganho proporcional.

2.6.3.3 Ação Derivativa do controlador PID

O principal objetivo dessa ação de controle é prever o erro atuante, antecipando uma ação corretiva, diminuindo o tempo de resposta, o que tende a aumentar a estabilidade do sistema. Quando uma ação derivativa é adicionada ao controle proporcional, obtém-se um controlador de alta sensibilidade. Essa ação é representada na equação (2.19):

$$u_{(t)} = K_d \frac{d}{dt} e(t) \xrightarrow{\mathcal{L}} U(s) = K_d s E_{(s)} \tag{2.19}$$

A ação de controle derivativa responde a uma taxa de variação do erro atuante e pode produzir uma correção significativa antes que o valor do erro se torne muito elevado. Além disso, embora ele não afete diretamente o erro estacionário, ele aumenta o amortecimento do sistema, permitindo maiores valores de K_p, resultando em uma maior precisão no regime permanente. Como ele opera sobre uma taxa de variação do erro atuante e e não sobre o próprio erro, na maioria dos casos essa ação não é aplicada sozinha, sendo utilizado conjuntamente com controladores P ou PI (Ogata, 2010).

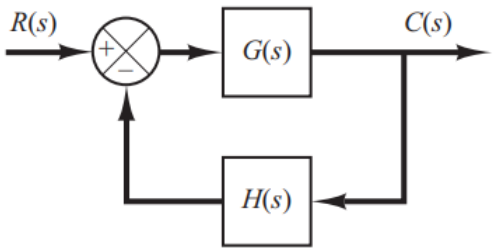
2.6.3.4 Métodos de sintonia de controladores PID: Lugar Geométrico das Raízes

Como contextualização histórica, o método do Lugar Geométrico das Raízes (LGR) foi desenvolvido pelo especialista em teoria de controle Walter Richard Evans em 1948 (Evans, 2004). Esse método permite que as raízes da equação característica sejam representadas graficamente para todos os valores de um parâmetro do sistema, comumente esse parâmetro é o ganho da função de transferência de malha aberta (Ogata, 2010).

Uma característica básica da resposta transitória de um sistema em malha fechada está intimamente relacionada à localização dos polos de malha fechada, que são as raízes da equação característica. Por sua vez, essa equação representa o denominador da função de transferência de malha fechada do sistema. Se o ganho do sistema for variável, então a localização dos polos de malha fechada também o será, logo é imprescindível que se tenha conhecimento de como os mesmos se comportam à medida que o ganho varia. Em alguns casos, apenas o ajuste de ganho já se configura adequado para o sistema atingir os requisitos de projeto, porém quando isso não ocorre, faz-se necessário adicionar um compensador ao sistema. A determinação das raízes da equação característica pode não ser suficiente, pois, como apresentado anteriormente, se o ganho da função de transferência de malha aberta varia, a equação característica se altera e os cálculos devem ser refeitos.

Utilizando o método do Lugar Geométrico das Raízes, é possível analisar o comportamento do sistema em relação à variação de ganho, adição de polos ou zeros na função de transferência de malha aberta sobre os polos da função de transferência da malha fechada. Para esboçá-lo é necessário atender as condições de módulo e fase de um sistema que pode ser descrito como mostra a figura 2.25:

Figura 2.25 – Sistema de controle



Fonte: (Ogata, 2010)

Então sua função de transferência de malha fechada é como mostra a equação (2.20):

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{2.20}$$

E sua equação característica é obtida igualando a zero o denominador da equação (2.20) representada por:

$$\begin{aligned} 1 + G(s)H(s) &= 0 \\ G(s)H(s) &= -1 \end{aligned} \tag{2.21}$$

Assim, a partir da equação (2.21), tem-se a condição de módulo e fase representadas pelas equações 2.22a e 2.22b:

$$|G(s)H(s)| = 1 \tag{2.22a}$$

$$\angle G(s)H(s) = \pm 180^{\circ}(2k+1) \mid k \in \mathbb{Z} \geq 0 \tag{2.22b}$$

Como a condição de módulo expressa pela Equação 2.22a está relacionada ao ganho K de malha aberta do sistema em diferentes frequências s, então pode-se analisar como a amplitude da saída do sistema varia em função da frequência de entrada, e consequentemente como essa variação de ganho afeta sua estabilidade. Se este parâmetro for muito alto, as raízes podem se mover para o semi-plano direito do plano complexo e levar o sistema à instabilidade, em contrapartida, se o ganho for muito baixo, então o sistema poderá ter um comportamento subamortecido, muito parecido com o sistema aberto, sem o real controle dinâmico.

Como complemento, a condição de fase expressa pela Equação 2.22b descreve o deslocamento angular da resposta do sistema em relação à entrada, dependendo da frequência. Essa condição delimita a margem de fase, a qual expressa em que ponto o sistema atinge um comportamento crítico, podendo se tornar instável a partir daquela frequência. Nesse

contexto, para análise de comportamento do sistema, o software Matlab oferece uma ferramenta de visualização e ajuste do LGR pelo comando *sisotool(system)*, em que *"system"* representa a função de transferência em malha aberta do processo estudado.

2.6.4 Controle Adaptativo

O controle adaptativo pode ser definido sucintamente de acordo com (Åström; Wittenmark, 2008):

Definição 2.2. Um controle adaptativo é um controlador com parâmetros ajustáveis e mecanismos para ajuste desses parâmetros.

Como contextualização história, o controle adaptativo surgiu no âmbito da aviação em 1950, mais precisamente para aviões de alta performance, como mostrado na figura 2.26. Esse tipo de aeronave opera em uma ampla gama de velocidades e altitudes, surgindo a necessidade de um controlador mais sofisticado que pudesse atender esses requisitos. Após algum tempo de desenvolvimento e estudos, surgiu a técnica de escalonamento de ganhos, que serviu bem para suprir a necessidade aeronáutica (Åström; Wittenmark, 2008).

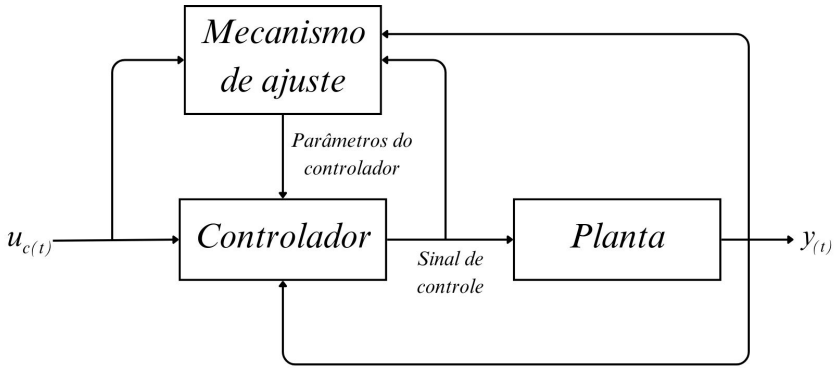
Figura 2.26 – Aeronave experimental X-15 para sistemas de controle avançados

Fonte: (NASA, 1961)

Foi perceptível ao longo das décadas o desenvolvimento do controle adaptativo, com o surgimento das teorias de espaço de estados e estabilidade, além do controle estocástico, programação dinâmica, dentre várias outras tecnologias que foram surgindo para que fosse possível tornar o controle adaptativo comercial e atualmente, técnicas de sintonia automática de controles são uma realidade. Esse tipo de controle pode ser compreendido como possuindo dois laços: um mais interno, dedicado à realimentação, que opera de forma mais rápida, e

outro mais externo, responsável pelo ajuste de parâmetros, operando de maneira mais lenta. A estrutura geral de um controlador desse tipo pode ser vista na figura 2.27:

Figura 2.27 – Diagrama de funcionamento de um controle adaptativo



Fonte: Adaptado de (Åström; Wittenmark, 2008)

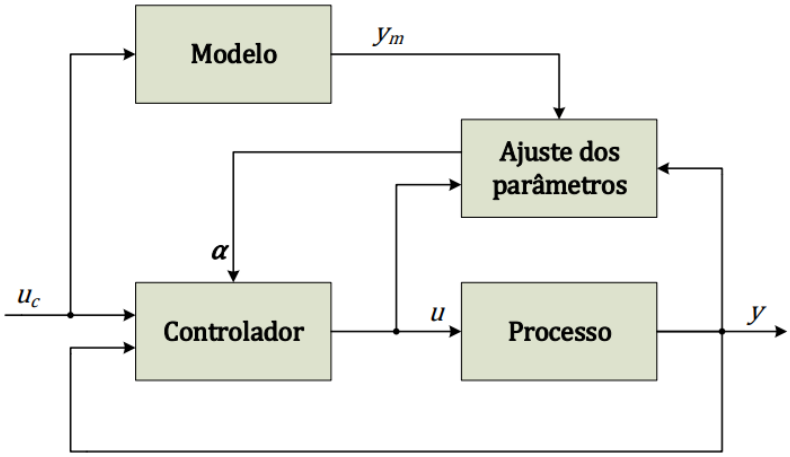
Não obstante, é necessário ressaltar que os controles adaptativos, sendo inerentemente não-lineares, são mais complexos que controladores de ganho fixo. Eles devem ser usados em casos especiais em que a dinâmica do processo é variável e imprevisível, caso contrário controladores com parâmetros constantes, como o PID, conseguiriam suprir essa necessidade. Existem quatro tipos de controles adaptativos, sendo eles: o Escalonamento de ganhos, os Reguladores Autoajustáveis (STR), o Controle Dual e o Controle Adaptativo por Modelo de Referência (MRAC). Nesse trabalho foi escolhido o último mecanismo, pois é um controle que não se restringe a uma faixa delimitada de pontos de operação, como ocorre com o método de escalonamento de ganhos. Além disso, a atualização dos seus parâmetros é feita de forma direta, com base no erro de rastreamento entre a saída do modelo de referência e a saída da planta real, ocorrendo rápida adaptação, diferentemente dos modos STR e Controle Dual, que precisam ainda fazer a identificação do modelo em tempo real para a estimação os parâmetros do processo antes de atualizá-los (Biagioni; Carlos, 2013) (Åström; Wittenmark, 2008).

2.6.4.1 Controle Adaptativo por Modelo de Referência

Esse método parte do pressuposto de que o processo deve chegar à uma referência, ou seja, os parâmetros do controlador devem ser ajustados a partir dos mecanismos de ajuste para que o erro e entre o sinal de saída da planta e o sinal de saída do modelo de referência seja nulo. Um esquemático do MRAC é mostrado na figura 2.28 e, como citado anteriormente, o desempenho do processo é definido pelo modelo de referência (Modelo), que fornece a saída desejada y_m ao sinal de referência u_c. Ainda, como discutido por (Åström; Wittenmark, 2008), a adaptação ocorre em dois laços: um externo, responsável por realimentar a saída y do processo no controlador e um interno, responsável por atualizar os parâmetros α. Com base no sinal de erro calculado como a diferença entre o sinal y e y_m, sendo as saídas da

planta e do modelo, respectivamente.

Figura 2.28 – Diagrama de funcionamento de um MRAC



Fonte: (Castro, 2018)

Para a implementação do ajuste dos parâmetros existem vários mecanismos para tal como destacados na literatura o método baseado na teoria de estabilidade de Lyapunov e método do gradiente de minimização do erro quadrático e^2 também conhecido como regra do MIT. Para esse trabalho, devido ao aspecto de simplicidade concomitante à robustez, e portanto, alta aceitação no meio acadêmico, foi escolhido o último método para o controle dos motores BLDC para o Tilt-rotor.

A regra MIT foi desenvolvida na instituição de mesmo nome, no Laboratório de Instrumentação. Para desenvolvê-la, considera-se um sistema em malha fechada em que o controlador tenha apenas um parâmetro de ajuste θ. A resposta desejada em malha fechada é especificada por um modelo que sua saída é y_m e o erro entre a saída da planta e a desejada é $e = y - y_m$. Uma possibilidade para ajustar esse parâmetro é considerar que a seguinte função de perda representada pela equação (2.23) é minimizada:

$$J = \frac{1}{2}e^2 \tag{2.23}$$

Para isso, deve-se alterar os parâmetros tal que eles sigam na direção do gradiente negativo de J, ou seja:

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta} \tag{2.24}$$

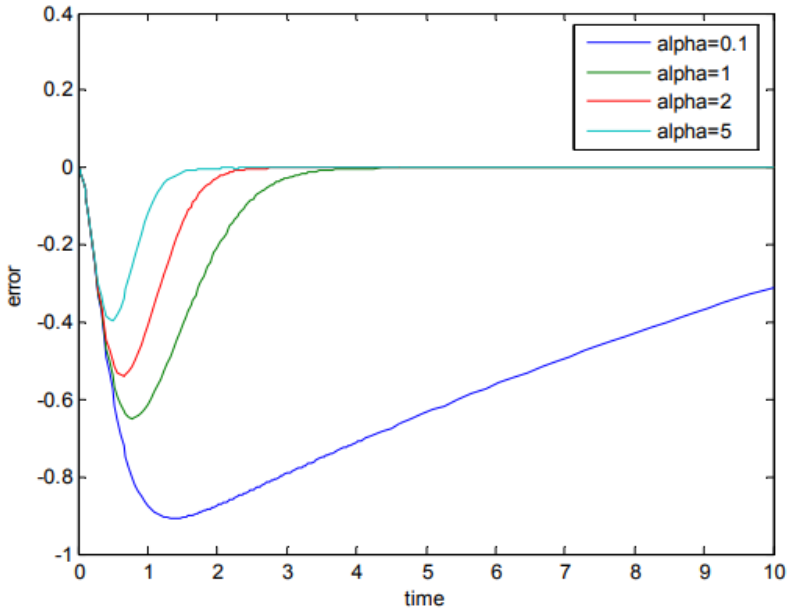
A equação (2.24) é a equação que representa a Regra MIT, com a derivada parcial $\frac{\partial e}{\partial \theta}$ sendo conhecida como *derivada de sensibilidade* do sistema, que indica como o erro é influenciado pelo parâmetro de ajuste. Existem várias formas de funções de perdas que podem ser consideradas, como por exemplo a função $J_{(\theta)} = |e|$, que o método gradiente

resulta em:

$$\frac{d\theta}{dt} = -\gamma \frac{\partial e}{\partial \theta} sgn(e) \tag{2.25}$$

Por fim, é mostrado que a taxa de convergência depende diretamente do parâmetro γ, que representa a taxa de adaptação do sistema. Experimentos simulados mostram que os parâmetros de adaptação convergem lentamente quanto menor for o valor de γ, e que a taxa de convergência cresce concomitantemente (Åström; Wittenmark, 2008). Não obstante, são notáveis os estudos com métodos de simulação para se obter um direcionamento quanto ao valor desse parâmetro, como feito em (Swarnkar; Jain; Nema, 2011), onde foi mostrado que ganhos muito pequenos não levavam a resposta da planta a aproximar-se do valor esperado da resposta da referência. Logo, quanto maiores os valores desse parâmetro nos testes feitos dentro da faixa analisada do parâmetro variante em $0.1 < \gamma < 5$, mais perto de nulo o valor do erro ficava. Os resultados desse experimento podem ser vistos na figura 2.29 que mostra o erro tendendo a ser nulo mais rapidamente à medida que a taxa de adaptação é elevada:

Figura 2.29 – Curva de erro ao longo do tempo de acordo com a variação da taxa de adaptação do MRAC



Fonte: (Swarnkar; Jain; Nema, 2011)

2.6.5 Revisão da Literatura: Técnicas de Controle MRAC e PID

Esta seção trata de uma revisão de literatura relacionada ao desenvolvimento de controles adaptativos por modelo de referência e como podem ser comparados com o controle convencional PID quando aplicados a diferentes contextos. Essa revisão surge como uma referência teórica para o trabalho desenvolvido, apontando como as técnicas e os proble-

mas de implementação de controle foram tratados por diferentes pesquisadores a partir de diferentes estratégias.

Em (Ban; Crnosija, 2003), há a aplicação de uma versão simplificada do MRAC em motores DC, considerando-os como um sistema de ordem elevada com parâmetros variáveis. Esse controle foi implementado no microcontrolador dSPACE DS1102 em cascata ao controle de velocidade e ficou evidente a partir dos resultados experimentais que essa aplicação reduziu significativamente a influência da variação dos parâmetros da planta e dos distúrbios no comportamento do sistema.

Sob outro contexto, um caso de comparação entre os métodos de ajuste de parâmetros para controladores PID convencionais foi desenvolvido em (Hassan; Al-Shamaa; Abdalla, 2017) para controlar a posição com a tensão, resultando em uma planta de terceira ordem. Com as comparações desenvolvidas para o sistema considerado, o método de Ziegler-Nichols modificado apresentou o melhor comportamento, devido à maior resposta dinâmica aos distúrbios de carga com um sobressinal aceitável.

Nos trabalhos (Sachit; Vinod, 2022) e (Mosaad, 2023), há a comparação entre a aplicação do MRAC com controladores PI. No primeiro, há a implementação de um PI tradicional para o estudo, em contrapartida, no segundo, é usado um ajuste de parâmetros baseado em uma recente abordagem chamada EHO (*Elephant Herding Optimization*, ou em tradução livre, Otimização do Pastoreio de Elefantes). É mostrado que mesmo com a nova abordagem de sintonia de parâmetros, esse controle não pode ser implementado em tempo real, pois é um processo lânguido, além disso, como o controle é ajustado para determinado ponto de operação, não é garantido que em uma variação do sistema o controle ainda tenha desempenhos otimizados como para aquele ponto que foi projetado.

Ainda, em (Swarnkar; Jain; Nema, 2011) e (Jain; Nigam, 2013) há a implementação de MRACs para sistemas de segunda ordem e um estudo em relação à alta sensibilidade do mecanismo ordinário de adaptação do controlador frente às variações do ganho de adaptação e às variações de amplitude do sinal de entrada, características que são limitadas a um determinado intervalo de valores para que o sistema não se torne instável. O primeiro aborda como os diferentes ganhos de adaptação afetam o sistema, e o segundo sugere o desenvolvimento de uma adaptação da Regra MIT convencional para um melhor desempenho do sistema. Por fim, em (Kanso, 1991), há o desenvolvimento de uma nova técnica de seleção do ganho em questão para conservação de estabilidade.

3 Desenvolvimento

Neste capítulo será abordado o desenvolvimento do trabalho, incluindo o desenvolvimento do sistema embarcado, do sistema de aquisição de dados, e dos controladores para os motores.

3.1 Sistema Embarcado

Nesta seção, será explicado como foi feito o desenvolvimento do sistema embarcado, envolvendo hardware e software embarcado, além das definições dos requisitos funcionais e temporais do sistema.

3.1.1 Requisitos Funcionais e Definições dos Sensores, dos Motores e do Microcontrolador

Conforme a seção 1.2, com base nas funcionalidades esperadas para a continuação dos trabalhos anteriores, foram definidos os seguintes requisitos funcionais para o desenvolvimento do hardware e do software embarcado:

- Um sensor capaz de medir a distância entre o robô aéreo e o solo, entre 2 e 400cm;
- Dois sensores capazes de ser utilizados para medição de velocidade angular, um para cada motor;
- Uma unidade de medição inercial contendo acelerômetro, giroscópio e magnetômetro, confiável e robusta;
- Dois motores, nos quais as hélices são acopladas, apropriados para aplicações em veículos aéreos;
- Dois servo motores, para realizarem a inclinação das hélices, apropriados para aplicações em veículos aéreos, e com controle de angulação entre 0° e 45°.

Os seguintes sensores foram escolhidos por cumprirem os requisitos citados anteriormente:

- Um sensor ultrassônico "$HC-SR04$", explicado em 2.4.1;
- Dois módulos "$MHSensorSeries/TCRT5000$", um para cada rotor, explicados em 2.4.2.1;
- Uma unidade de medição inercial "$MPU-9250$", explicada em 2.4.3.

Os seguintes motores foram escolhidos por cumprirem os requisitos citados anteriormente:

- Dois motores "$BLDC\ Turnigy\ D2830/11\ 1000KV$", com um controlador eletrônico de velocidade "$30A\ BLDC\ ESC$"para cada motor, explicados em 2.5.1;

- Dois servo motores "$DS3230$", explicados em 2.5.2.

O microcontrolador "$ESP-WROOM-32$"foi escolhido como o microcontrolador único para o sistema por suas vantagens para aplicações em sistemas embarcados, as quais foram explicadas em 2.3.

Em relação à bateria, foi utilizada a mesma que havia sido definida nos trabalhos anteriores, o modelo "$Lipo\ ZIPPY\ Compact\ 2200mAh\ 3S\ 25C$"de $11{,}1V$ em três células, que foi citado na seção 1.2.

3.1.2 Hardware

O sistema eletrônico do robô aéreo adaptado para o microcontrolador "$ESP32$"consiste em duas placas de circuito impresso: o Sistema de Controle Eletrônico e o Sistema de Alimentação. O protótipo do hardware foi montado em duas placas perfuradas, que podem ser vistas nas Figuras 3.1 e 3.2.

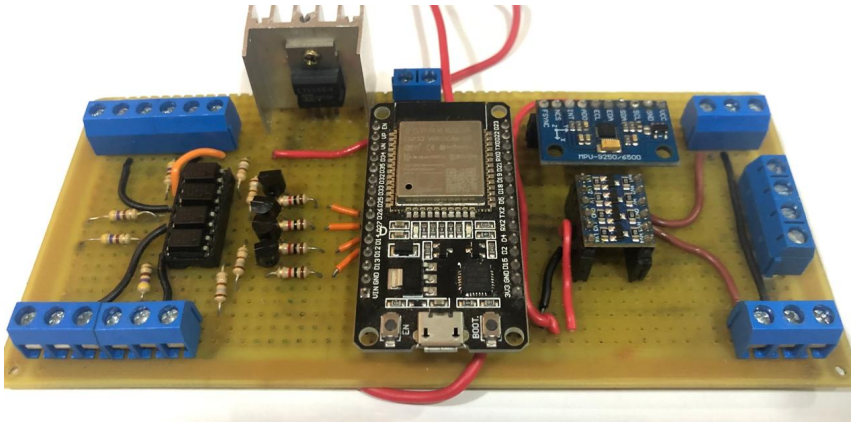


Figura 3.1 – Protótipo do Sistema de Controle Eletrônico, montado em placa perfurada.

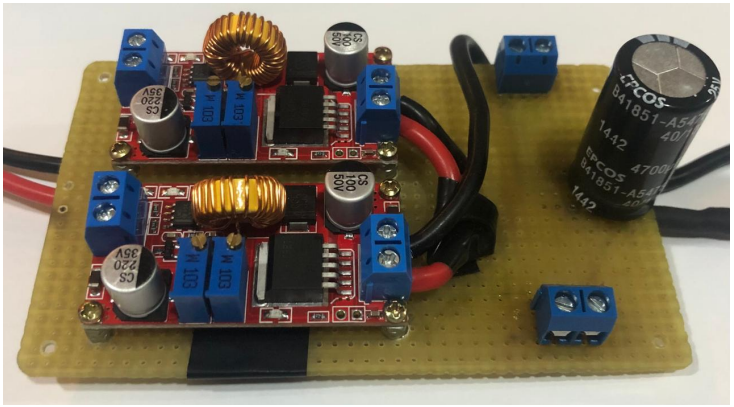


Figura 3.2 – Protótipo do Sistema de Alimentação, montado em placa perfurada.

3.1.2.1 Sistema de Controle Eletrônico

O Sistema de Controle Eletrônico (SCE) tem como principal função conectar as entradas e saídas do microcontrolador "$ESP32$"aos respectivos sensores e atuadores. Um diagrama que representa a placa do Sistema de Controle Eletrônico pode ser visto na Figura 3.3.

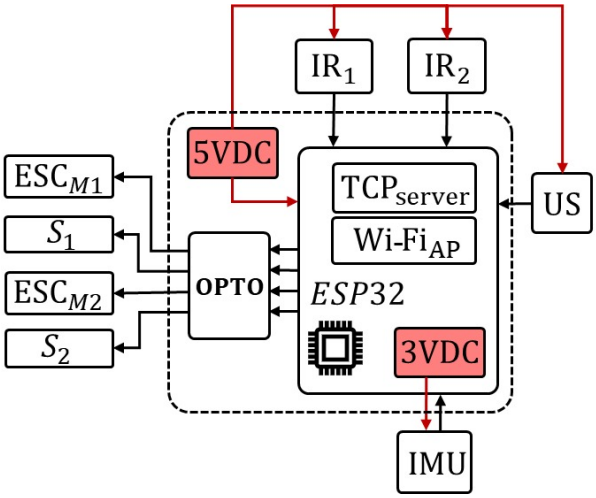


Figura 3.3 – Diagrama referente ao Sistema de Controle Eletrônico (SCE).

Os sensores: ultrassônico (US), fotosensores (IR_1 e IR_2) e a unidade de medição inercial (IMU), conectam-se às entradas do microcontrolador. Como o nível lógico do "$ESP-32$"é $3.3V$ e o do sensor ultrassônico e dos fotosensores é $5V$, é necessário um conversor de nível lógico (Huinfinito, 2025) entre esses sensores e os pinos do microcontrolador.

Os dois servo motores (S_1 e S_2), além do "ESC"de cada motor "$BLDC$"(ESC_{M1} e ESC_{M2}), são conectados às saídas do microcontrolador. O opto acoplamento, o bloco "$OPTO$"no diagrama, consiste em, para cada motor, um circuito com o optoacoplador $PC817$ sendo acionado pelo transistor $2N2222$ (Semiconductor, 2021). Este bloco tem, então, a função transmitir o sinal de controle do microcontrolador para cada motor por opto acoplamento. O objetivo deste circuito é isolar fisicamente o microcontrolador e os motores, funcionando como uma camada de segurança para a operação do microcontrolador em um sistema de potência. Dessa forma, o lado do optoacoplador que contém o LED é alimentado pela própria alimentação da placa, enquanto o lado que contém o transistor é alimentado diretamente pela saída de $5V$, no caso do motor "$BLDC$", e pela alimentação do servo motor, no caso do servo motor.

O esquemático da placa do Sistema de Controle Eletrônico foi feito no software "$Altium\ Designer$", a fim de ser utilizado em uma futura fabricação da placa de circuito impresso, após validação do protótipo.

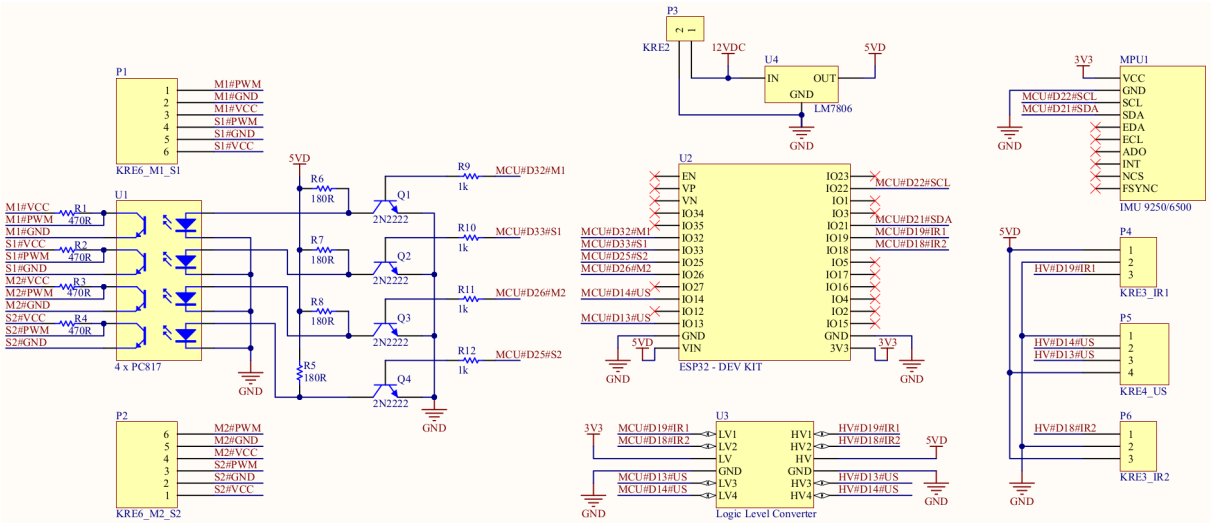


Figura 3.4 – Esquemático da placa do Sistema de Controle Eletrônico, feito no software "*Altium Designer*".

3.1.2.2 Sistema de Alimentação

O Sistema de Alimentação é a parte do hardware que fornece tensão de alimentação a todos os componentes do Tilt-Rotor, além disso consiste em um sistema de potência, pois as correntes requisitadas da bateria são significativamente mais altas do que as correntes de controle. Um diagrama que representa a placa do Sistema de Alimentação pode ser visto na Figura 3.5.

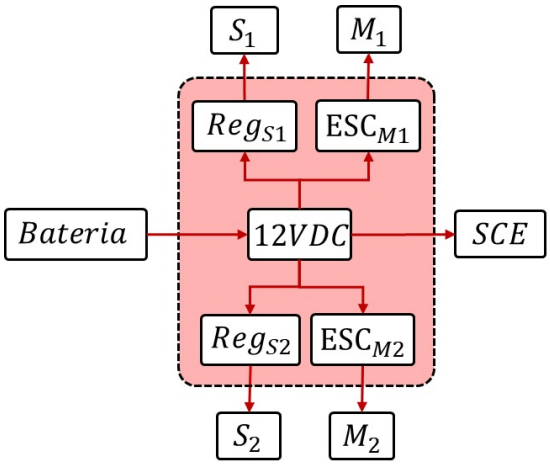


Figura 3.5 – Diagrama referente ao Sistema de Alimentação.

A bateria fornece tensão $12VDC$ com capacidade de corrente alta. Nas etapas de desenvolvimento do projeto, a bateria foi substituída por uma fonte de bancada de mesmo valor de tensão. Um capacitor de $4700\mu F/25V$ foi colocado em paralelo com a fonte de alimentação para funcionar como suporte contra picos de corrente, proteção contra quedas

de tensão, filtragem de ruídos e estabilização da tensão.

Em série com a fonte de alimentação, foi conectada uma chave gangorra para funcionar como o botão liga-desliga do Tilt-Rotor. Em paralelo com a chave, foi colocado um diodo de roda livre no sentido contrário, pois ele cria um caminho para a corrente reversa, a qual vem dos motores e aparece quando a chave abre; assim o diodo, do modelo "$UF5408$ $Schottky$"(Semiconductor, 2022), cria um caminho para a corrente retornar para a fonte.

A alimentação dos motores sem escovas conecta-se diretamente no "ESC"de cada motor. Já a alimentação de cada servo motor passa por um regulador de tensão (Reg_{S1} e Reg_{S2}) (Robótica, 2023), que reduz a tensão fornecida pela bateria, mas não limita a corrente.

A alimentação da placa "SCE"consiste na parte de menor demanda de corrente, em comparação com os motores. Nela há o regulador 7805 (Instruments, 2016), que reduz a tensão para $5VDC$ e alimenta os sensores e o microcontrolador; com exceção da "IMU", cuja alimentação é fornecida pelo regulador de $3.3VDC$ do ESP-32.

O esquemático da placa do Sistema de Alimentação foi feito no software "$Altium\ Designer$", a fim de ser utilizado em uma futura fabricação da placa de circuito impresso, após validação do protótipo.

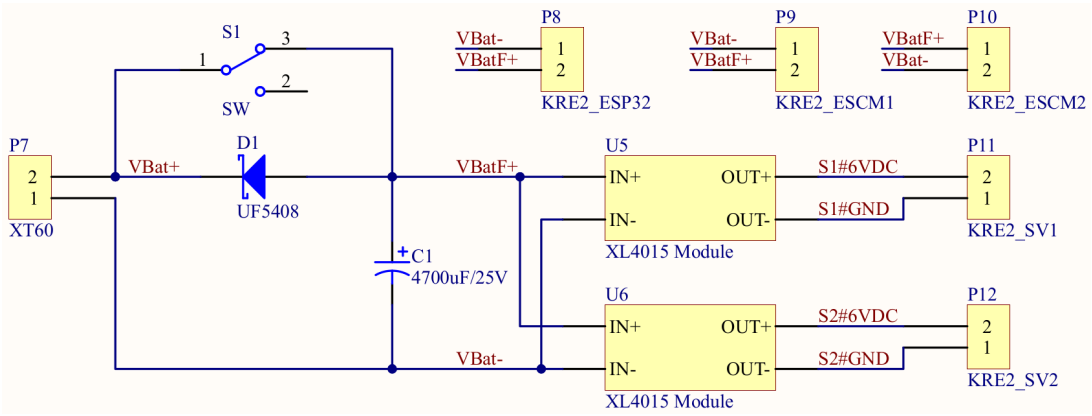


Figura 3.6 – Esquemático da placa do Sistema de Alimentação, feito no software "$Altium\ Designer$".

3.1.3 Requisitos Temporais

A partir dos requisitos funcionais e dos sensores que foram definidos na seção 3.1.1, foi definida a periodicidade de execução de cada uma das tarefas relacionadas às medições dos sensores. Os requisitos temporais podem ser vistos na Tabela 3.1. Nesta seção, será explicado como foram definidas as periodicidades.

Tarefa	Período(ms)	Tipo de execução
Medição de distância pelo sensor ultrassônico	50	periódica
Medição de velocidade angular com o sensor ótico	variável	esporádica
Medições do acelerômetro, giroscópio e magnetômetro	10	periódica

Tabela 3.1 – Requisitos temporais das tarefas relacionadas aos sensores.

A documentação do sensor ultrassônico 2.4.1 indica que o período mínimo entre suas medições deve ser entre 40 e 60ms, para que a onda sonora emitida não interfira nas medições seguintes. Escolhendo o período de 50ms, para a velocidade do som de 343m/s, a distância máxima medida pelo sensor é conforme a equação 3.1. A divisão por 2 é necessária para considerar que a onda sonora é enviada e retorna. Assim, 50ms é o tempo necessário para o sensor receber o eco de um objeto há 8,575m; como a faixa de medição do sensor é até 4m, então há uma margem que evita medições erradas.

$$distancia[m] = \frac{343[m/s] \cdot 0{,}050[s]}{2} = 8{,}575[m] \tag{3.1}$$

A utilização do sensor óptico 2.4.2.1 para medição da velocidade angular do motor consiste em uma tarefa esporádica, pois seu período é variável e sua ativação depende da ocorrência de um evento externo, no caso a interrupção gerada pela passagem da marcação no motor, além de que o período tem valores limites máximo e mínimo. O período mínimo da tarefa, ou seja, o menor tempo para uma volta completa do motor, consiste na velocidade máxima do motor. Foi adicionado um saturador que limita a velocidade do motor para um valor próximo de 9500RPM, aproximadamente 994,84rad/s. Pela equação 2.2, calcula-se que o período mínimo é 6,316ms. O período máximo foi definido como 1000ms, pois, pela mesma equação, é o período equivalente a 60RPM; dessa forma, para velocidades abaixo de 60RPM, a tarefa define que o motor está parado. Em suma, o período da tarefa varia entre 6,316ms e 1000ms.

Em relação à unidade de medição inercial 2.4.3, sua documentação informa que o magnetômetro tem a possibilidade de ser configurado para duas frequências de amostragem, foi escolhida a mais rápida, de 100Hz, ou seja, periodicidade de 10ms. O acelerômetro e o giroscópio podem ser configurados para frequências de mostragem mais de dez vezes maiores, o que torna o magnetômetro o gargalo. Posteriormente, é possível a separação do magnetômetro em outra tarefa, para que a frequência de mostragem do acelerômetro e do giroscópio possa ser aumentada.

3.1.4 Software Embarcado

O software embarcado para o microcontrolador "$ESP32$"foi desenvolvido e testado em etapas. Todas as camadas (do firmware às aplicações de nível mais alto) apresentam processamento em tempo real. O software embarcado foi dividido em módulos, que serão

explicados nesta seção. Um diagrama que representa a arquitetura do software embarcado e a execução concorrente das tarefas pode ser visto na Figura 3.7.

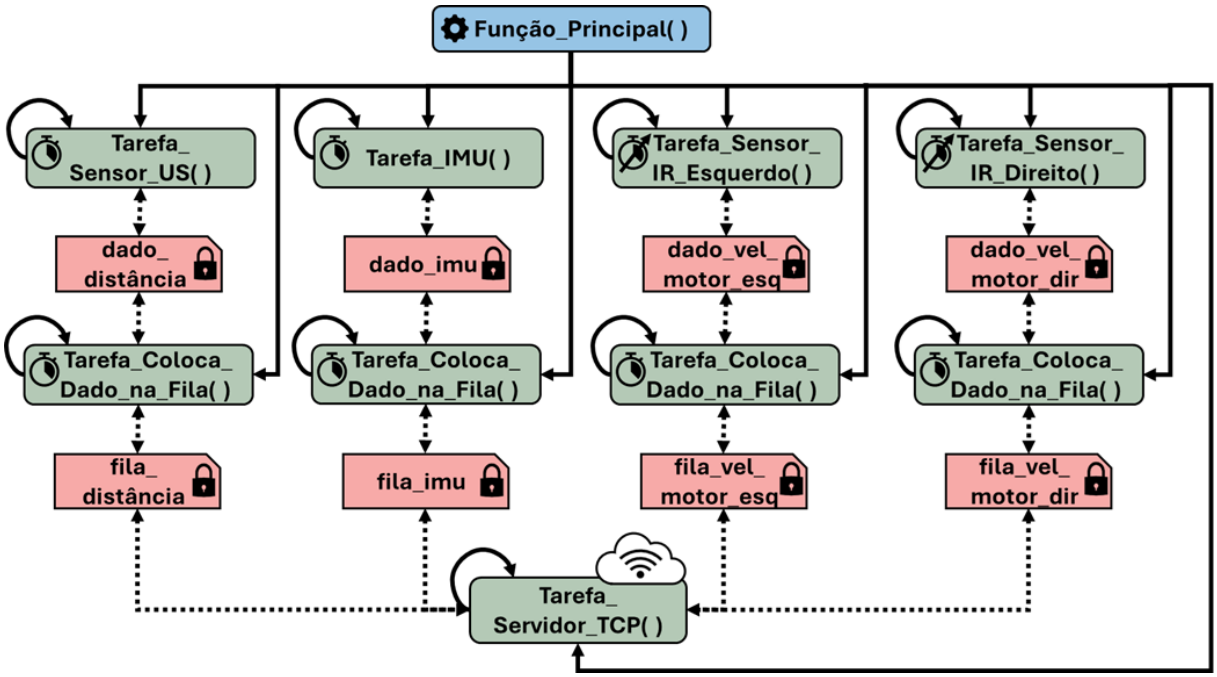


Figura 3.7 – Diagrama da arquitetura do software embarcado e da execução concorrente das tarefas.

3.1.4.1 Módulo principal

Este módulo se refere à implementação do código-fonte "*main.c*", que contém a função "*app_main*". O módulo declara as instâncias das estruturas de todos os sensores, inicializando o *mutex* para cada uma, e definindo o valor inicial zero para suas variáveis. Também cria as tarefas dos demais módulos, atribuindo prioridades para elas e não definindo afinidades com nenhum núcleo específico do microcontrolador. As tarefas das medições dos sensores têm a maior prioridade, 10; as tarefas que colocam dados na filas têm prioridade 9; e a tarefa da comunicação TCP/IP tem a menor prioridade, 8. Além disso, o módulo chama as funções de inicialização: da geração de PWM, dos motores, e da geração da rede $Wi-Fi$. Ao final, o servidor TCP/IP é criado e a função retorna.

3.1.4.2 Módulo do sensor ultrassônico

Nesta seção, detalha-se a implementação do código-fonte "*ultrasonic_sensor.c*", responsável por gerar as leituras do sensor ultrassônico e armazenás-la em uma estrutura declarada no cabeçalho "*ultrasonic_sensor.h*". Utiliza-se o módulo de hardware integrado $MCPWM$. Cada função tem o seguinte propósito:

- *ultrasonic_mcpwm_init*: Esta é a função de inicialização. É executada uma única vez, pois ela realiza a alocação dos recursos de hardware do $MCPWM$ que serão

utilizados pelo módulo. Ela define: um temporizador de captura; um canal de captura, nas bordas de subida e de descida, configurando o pino *echo* como uma entrada; a função de *callback*, a qual é chamada sempre que ocorre um evento de captura, que consiste em uma interrupção de hardware; e o pino *trigger*, configurando-o como uma saída inicialmente com o nível lógico baixo. Em seguida, habilita o canal e o temporizador, e inicia o temporizador no canal, para que comece a captura de eventos.

- *hc_sr*04*_echo_callback*: É a função responsável pelo processamento do tempo entre as bordas de subida e descida. Se o evento de captura é uma borda de subida, inicia-se uma amostra, armazenando o valor do temporizador na variável "começo da amostra". Se o evento é uma borda de descida, armazena-se o valor do temporizador na variável "fim da amostra". A diferença entre essas duas variáveis é o tempo de propagação da onda sonora. No momento em que é gerado esse tempo, a função envia uma notificação, a qual consiste em uma interrupção de software, para a tarefa principal.
- gen_trig_output: É a função que gera um pulso único, de $10\mu s$, no pino *trigger*, para que se inicie uma nova amostra.
- $Ultrasonic\_task$: É a tarefa principal do módulo. Ela chama a função de inicialização e, em seguida, entra no laço, que é executado de forma periódica a cada $50ms$. A tarefa periódica inicialmente chama a função que gera o pulso do *trigger*, e aguarda a notificação de que o tempo de propagação foi gerado. Ao chegar a notificação, o tempo de propagação é convertido em largura de pulso, na unidade μs, e, como consiste no tempo de ida e de volta da onda sonora, seu valor deve ser dividido por 2 para ser utilizado no cálculo da distância. Então, se a largura de pulso (p) estiver dentro da faixa de alcance do sensor, a distância (d), em metros, é calculada pela seguinte relação aproximadamente linear, considerando a velocidade do som no ar (v) igual a $343m/s$.

$$d = \frac{p \cdot 10^{-6}}{2} \cdot v \tag{3.2}$$

Por fim, a distância gerada, que é a leitura do sensor, é armazenada em uma estrutura (*struct*) global por meio de um *mutex* com a finalidade de proteger o acesso a esse recurso compartilhado. O tempo no qual essa medição foi gerada também é armazenado na estrutura global, ele é retornado de maneira precisa pela função "*esp_timer_get_time*()"fornecida pela ESP-IDF (Espressif, 2024), e consiste no tempo em microsegundos desde que o sistema foi inicializado.

3.1.4.3 Módulo da IMU

Nesta seção, será explicada a implementação do código-fonte "*imu_sensors.c*", responsável por gerar as leituras dos sensores: acelerômetro, giroscópio e magnetômetro; e

armazená-las em uma estrutura declarada no cabeçalho "*imu_sensors.h*". Utiliza-se um controlador "*I2C*"dedicado no hardware do "*ESP*32". Cada função tem o seguinte objetivo:

- *i2c_master_init*: É a função de inicialização do I2C, e é executada uma única vez. Ela configura os parâmetros do controlador I2C para operar no modo mestre, define os pinos SDA e SCL, define a frequência do barramento I2C como $400kHz$, instala o driver do controlador I2C, e inicializa o controlador, habilitando a comunicação I2C.
- *mpu*9250*_register_read*: Função sempre chamada pela tarefa principal. Ela lê uma sequência de bytes de registradores dos sensores acelerômetro e giroscópio, no endereço da MPU9250.
- *magnetometer_register_read*: Função sempre chamada pela tarefa principal. Ela lê uma sequência de bytes de registradores do sensor magnetômetro AK8963. É semelhante à anterior, apenas altera o endereço do sensor.
- *mpu*9250*_register_write_byte*: Função sempre chamada pela tarefa principal. Ela escreve um byte em um registrador dos sensores acelerômetro e giroscópio, no endereço da MPU9250.
- *magnetometer_register_write_byte*: Função sempre chamada pela tarefa principal. Ela escreve um byte em um registrador do sensor magnetômetro AK8963. É semelhante à anterior, apenas altera o endereço do sensor.
- *Imu_task*: É a tarefa principal do módulo. Inicialmente, ela chama a função de inicialização do I2C, e verifica se a MPU9250 está sendo reconhecida pelo microcontrolador, em seguida realiza as configurações dos sensores: define as faixas de operação para o acelerômetro e o giroscópio; e para o magnetômetro ativa-se o modo bypass, que faz com que ele possa ser acessado diretamente pelo endereço do sensor AK8963, lê-se os seus valores de ajuste de sensibilidade, e é feita a sua configuração para modo de medição contínua 2, com a frequência $100Hz$. Então, a função entra no laço que é executado de forma periódica, a cada $10ms$. Assim, a a leitura dos sensores é feita periodicamente, por meio das funções de leitura de bytes dos registradores. Para o acelerômetro e o giroscópio, a leitura analógica de cada eixo é lida dos registradores de forma dividida em dois bytes, o primeiro é o byte mais significativo e o segundo é o byte menos significativo, e é armazenada como dois inteiros sem sinal de 8 bits; depois os dois valores são convertidos para um único valor inteiro com sinal de 16 bits. Para o magnetômetro, primeiramente verifica-se o bit $DRDY$, o qual informa se o dado está pronto, e também o bit $HOFL$, que indica se houve ou não um $overflown$; se o dado estiver pronto, e se não houve $overflown$, é feita a leitura dos dados analógicos de cada eixo do magnetômetro da mesma forma que para os outros sensores, a diferença é que o primeiro byte é o menos significativo e o segundo byte é o mais significativo. É feita também a interpretação dos dados lidos, encontrando o fator de escala de sensibilidade pela região de operação, para cada um dos três sensores.

Para o acelerômetro, o fator de sensibilidade é utilizado para encontrar a leitura na unidade g, e então a leitura é convertida para a unidade m/s^2, utilizando o valor da aceleração da gravidade, $9{,}79 m/s^2$, na equação a seguir:

$$a_{m/s^2} = a_g \cdot 9.79 \tag{3.3}$$

Para o giroscópio, o fator de sensibilidade é utilizado para encontrar a leitura na unidade $^\circ/s$, e então a leitura é convertida para a unidade rad/s, pela seguinte equação:

$$g_{rad/s} = \frac{g_{^\circ/s} \cdot \pi}{180^\circ} \tag{3.4}$$

Para o magnetômetro, utiliza-se o fator de sensibilidade e os valores de ajuste de sensibilidade para encontrar a leitura na unidade μT, e não é feita nenhuma conversão de unidade. Por fim, com os dados interpretados e convertidos para as unidades desejadas, as nove leituras: eixos x, y e z do acelerômetro, eixos x, y e z do giroscópio, e eixos x, y e z do magnetômetro, são armazenadas em uma estrutura (*struct*) global, por meio de um *mutex* com a finalidade de proteger o acesso a esse recurso compartilhado. O tempo no qual as leituras foram geradas também é armazenado na estrutura global, ele é retornado de maneira precisa pela função "*esp_timer_get_time*()"fornecida pela ESP-IDF (Espressif, 2024), e consiste no tempo em microsegundos desde que o sistema foi inicializado.

3.1.4.4 Módulo dos motores

Nesta seção, será detalhada a implementação do código-fonte "*motors.c*", responsável por implementar as funções que enviam os sinais de controle *PWM* para os motores: dois servo motores e dois motores sem escovas. As funções desse módulo posteriormente serão chamadas pelo módulo de controle dos motores, logo este módulo não tem uma tarefa periódica. Utiliza-se o módulo de hardware integrado "*LEDC*". Cada função tem o seguinte propósito:

- *pwm_init*: Função de inicialização. É executada uma única vez pois realiza a alocação dos recursos de hardware do *LEDC* que serão utilizados pelo módulo para geração de PWM. Ela define um temporizador e um canal associados para cada motor, configurando para os motores sem escovas: resolução de $13 bits$, adequada para os testes de desenvolvimento mas que pode ser ajustada se necessário durante os testes de controle dos motores, e frequência de $50 Hz$, valor fixo requerido pelo ESC dos motores; e configurando para os servo motores: resolução de $13 bits$, e frequência de $330 Hz$; além de definir a porta digital de saída para cada motor, e largura de pulso inicial para zero.

- *motor_init*: Esta função, executada apenas uma vez, realiza uma inicialização simples, porém necessária para os ESC's dos motores sem escovas. Ela envia um comando com o valor equivalente à menor largura de pulso, para cada motor. Assim, os motores sem escovas estão configurados para receberem próximos comandos.
- *linear_conversion*: Função chamada pelas funções que enviam sinais PWM para os motores. Ela recebe como parâmetro uma tensão, vinda do módulo de controle dos motores, e retorna, por meio de uma conversão linear, um valor de largura de pulso, necessário para a geração de PWM. A função também adiciona um saturador, responsável por limitar, por segurança, a velocidade máxima que o motor é capaz de alcançar, e cujo valor sempre é ajustado durante os testes de desenvolvimento.
- *set_left_motor_pwm*: A função que envia o sinal de controle PWM para o motor sem escovas esquerdo, antes chamando a função de conversão linear.
- *set_right_motor_pwm*: A função que envia o sinal de controle PWM para o motor sem escovas direito, antes chamando a função de conversão linear.
- *set_left_servo_motor_pwm*: A função que envia o sinal de controle PWM para o servo motor esquerdo.
- *set_right_servo_motor_pwm*: A função que envia o sinal de controle PWM para o servo motor direito.

3.1.4.5 Módulo dos fotosensores

Nesta seção, detalha-se a implementação do código-fonte "*foto_sensors.c*", responsável por gerar as leituras dos fotosensores e por meio delas calcular a da velocidade dos motores, armazenando os dados de velocidade em uma estrutura declarada no cabeçalho "*foto_sensors.h*". Utiliza-se o módulo de hardware integrado *MCPWM*. Cada função tem o seguinte propósito:

- *left_fotosensor_mcpwm_init*: Esta é a função de inicialização do fotosensor esquerdo. É executada uma única vez, pois ela realiza a alocação dos recursos de hardware do *MCPWM* que serão utilizados pelo módulo. Ela define: um temporizador de captura; um canal de captura, na borda de subida apenas, configurando o pino de entrada; e a função de *callback*, a qual é chamada sempre que ocorre um evento de captura, que consiste em uma interrupção de hardware. Em seguida, habilita o canal e o temporizador, e inicia o temporizador no canal, para que comece a captura de eventos.
- *right_fotosensor_mcpwm_init*: Esta é a função de inicialização do fotosensor direito. É equivalente à função anterior.
- *left_fotosensor_callback*: É a função responsável pelo processamento do tempo entre as bordas de subida, para o fotosensor esquerdo. Ela sempre armazena o "valor

atual"e o "último valor"do temporizador. A diferença entre essas duas variáveis é o tempo para uma volta completa do motor. No momento em que é gerado esse tempo, a função envia uma notificação, a qual consiste em uma interrupção de software, para a tarefa principal.

- $right\_fotosensor\_callback$: É a função responsável pelo processamento do tempo entre as bordas de subida, para o fotosensor direito. É equivalente à função anterior.
- $Left\_Fotosensor\_task$: É a tarefa principal do módulo, para o fotosensor esquerdo. Ela chama a função de inicialização e, em seguida, entra no laço que é executado de forma esporádica, sempre que ocorre a interrupção externa de notificação. A chegada da notificação indica que o tempo para uma volta completa foi gerado, assim esse tempo é convertido em período, na unidade μs. Então, se o período (p) estiver dentro da faixa de valores aceitáveis, é feito o cálculo da velocidade angular do motor (v), em rad/s, por meio da equação:

$$v = \frac{2 \cdot \pi}{p \cdot 10^{-6}} \tag{3.5}$$

  A lógica da função também engloba o caso no qual não ocorrem bordas de subida, ou seja, o motor está parado; nesse caso, a velocidade angular é atualizada para o valor zero, se houve um intervalo de tempo de $1s$ sem a ocorrência de nenhuma borda de subida. Por fim, a velocidade angular gerada, e também o tempo no qual essa velocidade foi gerada, são armazenados em uma estrutura ($struct$) global por meio de um $mutex$ com a finalidade de proteger o acesso a esse recurso compartilhado; este tempo é retornado de maneira precisa pela função "$esp\_timer\_get\_time()$"fornecida pela ESP-IDF (Espressif, 2024), consiste no tempo em microsegundos desde que o sistema foi inicializado.
- $Right\_Fotosensor\_task$: É a tarefa principal do módulo, para o fotosensor direito. É equivalente à tarefa anterior.

3.1.4.6 Módulos de gerenciamento das filas circulares e Módulo de comunicação TCP/IP

Nesta seção, explica-se a implementação dos códigos-fontes responsáveis por inserir elementos nas filas circulares, além do módulo que as esvazia enviando os dados como uma mensagem JSON de resposta à solicitação de um cliente TCP/IP conectado na rede Wi-Fi gerada pelo "$ESP32$".

Os quatro módulos representados no diagrama da Figura 3.7 pelas "tarefas que colocam um dado na fila", fazem parte da estrutura completa do software embarcado do robô aéreo, pois neles serão colocados, em etapas futuras do projeto, respectivamente, os blocos do controlador de atitude e altitude discretizado, do sistema de localização (Filtro de Kalman Estendido), e do controlador de velocidade angular discretizado para cada motor. Por enquanto,

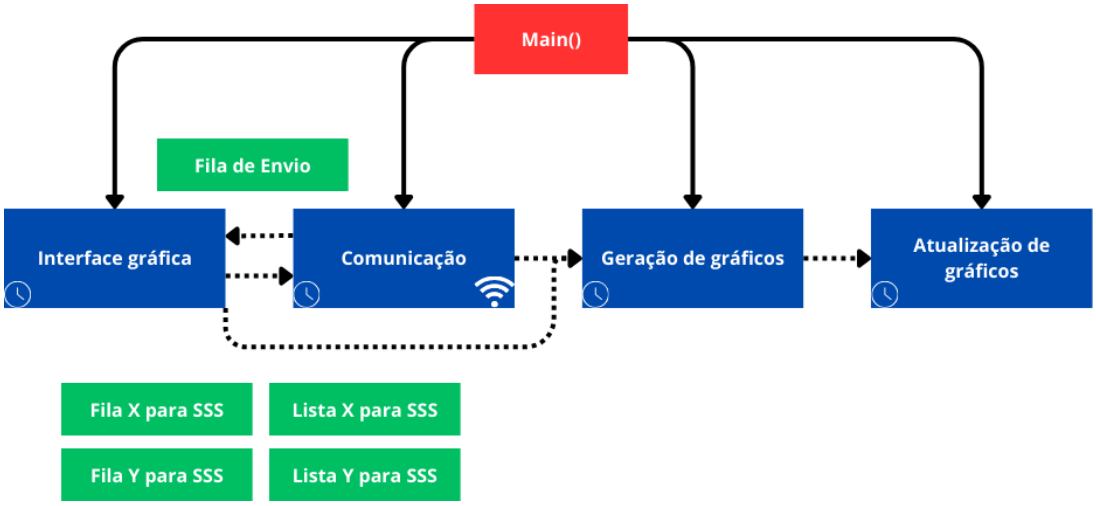
cada módulo tem uma tarefa principal, cujo período futuramente será o do controlador, que acessa, por meio de *mutex*, a estrutura onde está armazenada a leitura do respectivo sensor, além do tempo no qual a medição foi gerada, e coloca esses dados em uma fila circular. A biblioteca utilizada para as filas circulares foi disponibilizada em "*gqueue.h*"pelo professor orientador (Borges, 2018); a qual consiste em um gerenciador de índices que transforma um vetor de tamanho finito em uma fila circular. As filas circulares foram definidas como vetores de memória temporária de tamanho igual a 10; é necessário um maior estudo sobre a memória do "*ESP*32" para que seja possível um aumento do tamanho destes vetores.

Em paralelo, o microcontrolador disponibiliza seu próprio ponto de acesso Wi-Fi, por meio da biblioteca "*softap.h*"fornecida pela ESP-IDF (Espressif, 2024), e gera um servidor TCP/IP, pela criação da tarefa que pode ser vista no diagrama da Figura 3.7, da biblioteca "*tcp_server.h*"também fornecida pela ESP-IDF (Espressif, 2024). Sempre que uma conexão com um cliente TCP/IP é iniciada, é chamada a função "*communication*", a qual foi desenvolvida no código-fonte "*communication.c*". A função tem o objetivo de lidar com a comunicação entre o servidor e o cliente, ou seja, receber e enviar mensagens por TCP/IP, da seguinte forma: inicialmente é feita uma validação para verificar se foram recebidos dados sem erros; em seguida, a *string* da mensagem é transformada em um objeto JSON manipulável; então são extraídos os campos da mensagem; se o comando recebido na mensagem for um comando específico que solicita os dados de um determinado sensor, é criado um objeto padrão para a resposta JSON, e a fila circular referente ao sensor é esvaziada para que seus dados sejam colocados no objeto de resposta; por fim, o objeto JSON é convertido para uma resposta em formato *string*, e a resposta é enviada para o cliente TCP/IP.

3.2 Sistema de Aquisição de Dados

O sistema de aquisição de dados foi feito na linguagem de programação *Python* e tem a finalidade de adquirir informações lidas pelos sensores conectados ao ESP-32, armazená-los em arquivos, exibi-los em gráficos e enviar informações para o microcontrolador. Um diagrama do código pode ser visto na figura 3.8 e será explicado.

Figura 3.8 – Diagrama de funcionamento do sistema de dados



Esse sistema consegue fazer a conexão com o microcontrolador a partir do protocolo TCP/IP, com o padrão de comunicação IEEE 802.11, e utiliza diferentes unidades de execução denominadas tarefas, para que seja possível ao sistema trabalhar com execuções de códigos concorrentes. O arquivo **main.py** é o arquivo principal do sistema, nele há a criação de filas, utilizando a biblioteca **Queue**, para recebimento de dados vindos da conexão TCP/IP, e a criação de listas, para a plotagem de dados nos gráficos. Essas estruturas de dados são construídas na classe *Criar_Filas*, tendo as filas tamanho finito de elementos, transformando-as nas chamadas filas circulares. Com isso, são criadas as estruturas para os sensores: ultrassom, infravermelhos esquerdo e direito e IMU (magnetômetro, acelerômetro e giroscópio). Além disso, é criada uma fila para envio de mensagens pela conexão TCP/IP. Depois, é inicializada a tarefa de comunicação, utilizando a estrutura da biblioteca **Threading**, que tem como alvo a função de nome **cliente** do arquivo **cliente.py**, passando como argumento todas as filas e listas criadas para a devida manipulação de dados.

Ainda, é feita a configuração da tela inicial da interface gráfica utilizando a biblioteca **Tkinter**, criação e posicionamento de rótulos e botões para abertura das janelas de gráficos para cada sensor, além das configurações das ações desses últimos. As listas são os argumentos das funções que cuidam da abertura das janelas dos gráficos para ser possível a geração dos mesmos com os valores nelas armazenados, chamadas **janelaSSS**, em que "SSS" representa os diferentes sensores que podem ser requisitados. Assim, é inicializada a função **menu.mainloop()** para que a interface gráfica fique aberta e seja possível interagir com a mesma.

O arquivo **cliente.py** é o arquivo responsável pela lógica da comunicação TCP/IP do sistema. Nele é criado o soquete com o servidor, o microcontrolador ESP-32, a partir da função **create_client_socket**, e onde são criadas as mensagens em formato JSON que requisitam dados dos sensores, e conta com a seguinte estrutura de requisição dos dados:

Código 3.1 – Estrutura da mensagem JSON

```
1     {
2     "command": "get_SSS",
3     "id": ""
4     }
```

Em que "SSS"no "valor" da chave "comando" representa os diferentes sensores que podem ser requisitados. A chave "id" sempre é um valor aleatório gerado a partir da biblioteca **random**, e adicionada posteriormente à mensagem, antes do envio para o servidor, sendo uma maneira de depuração de perdas de pacotes na comunicação. Após a criação da mensagem, é iniciado o laço infinito de envio e recebimento de mensagens enquanto o soquete estiver ativo. Primeiramente as mensagens são inseridas na fila de envio, sendo criado um arquivo de registro das mesmas, com a função **armazenar_json_em_log**. Se não houver nenhuma mensagem na fila de envio, são enviadas mensagens de teste para conferir se o servidor ainda está ativo, essas não são incluídas na fila, são apenas enviadas diretamente e se o servidor não recebê-las, houve um erro de conexão e o soquete é fechado.

Se houver mensagens na fila de envio e chegar a resposta do servidor, é conferido primeiro o tamanho em bytes da mensagem para depois efetivamente adquirir os bytes recebidos da mensagem. Depois, esses dados são decodificados para formato JSON e "carregados" para a estrutura de dados de dicionário Python. Em um segundo momento, há a validação dos IDs das mensagens de envio e de recebimento. Se forem iguais, dependendo da mensagem enviada, é feito o processamento. Esse último é análogo para todos os sensores e para que seja possível receber novos dados as filas de interesse devem estar vazias, assim, cada item contido no valor da chave "tempo", é extraído e colocado na fila do eixo X, o mesmo procedimento é feito para a chave "distancia", sendo colocado na fila do eixo Y.

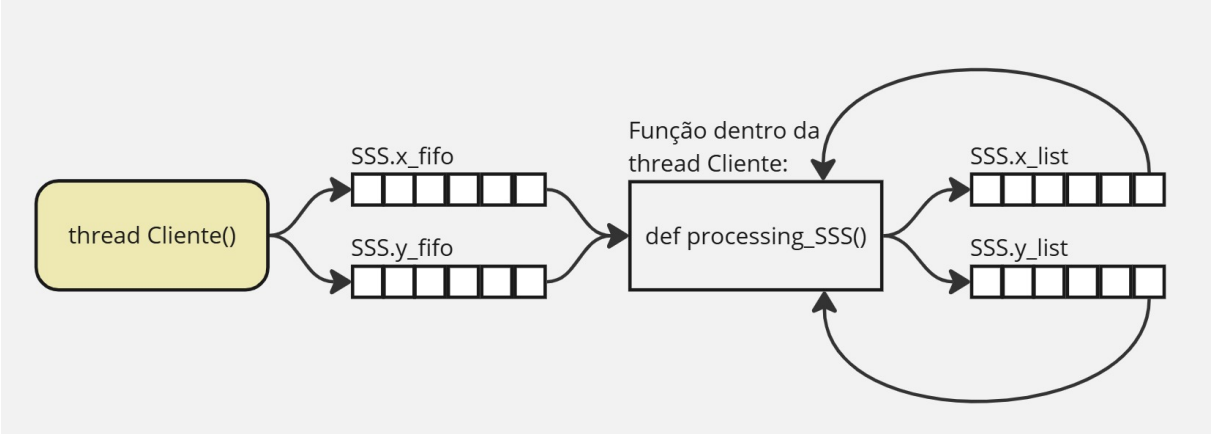
Uma vez preenchidas as filas, a função de processamento é chamada, tendo o nome de **processing_SSS** do arquivo **SSS.py**, em que "SSS" representa o título dos diferentes sensores que podem ser requisitados. Uma função de processamento foi feita para cada sensor, e está localizada em arquivos separados com seus respectivos nomes. Nessa função, se as filas de interesse estiverem cheias, as listas destinadas à geração dos gráficos correspondentes passam por um processo de limpeza para garantir que somente os novos dados serão plotados. Depois, até que as filas preenchidas pela conexão TCP/IP fiquem vazias por completo, cada item contido na fila de dados para o eixo X do gráfico daquele sensor é inserido na lista do eixo X e o processo análogo ocorre para a lista de onde será extraído os dados para a construção gráfica do eixo Y. Por fim, após o esvaziamento das filas, há a gravação dos dados a partir das listas em arquivos CSV para posterior análise de dados utilizando uma função chamada utilizando a biblioteca **csv**. A função tem o nome de **csv_SSS**, em que "SSS" representa os diferentes sensores que podem ser requisitados criando arquivos que têm o nome do respectivo sensor que se está recebendo os dados, sendo colocados na pasta

"Dados".

Essas funcionalidades descritas até então funcionam no momento em que o sistema é aberto: criação das filas e listas, abertura do soquete de comunicação, envio e recebimento de mensagens JSON entre servidor e cliente, além da própria visualização da interface gráfica do Tkinter, sendo a tarefa principal, e a tarefa secundária a de comunicação com o servidor que exerce as funções previamente descritas. Porém, quando há a ação de seleção de qualquer botão da página inicial, é aberta uma janela para determinado sensor. Então, é chamada a função **janelaSSS** do arquivo **janelas.py**, sendo passadas como argumento as listas para a geração dos gráficos. Nessa função, a interface gráfica é desenhada, e especificamente nas janelas de controles (MRAC e PID) há também o campo de interação com o usuário possibilitando que seja inserido um valor para ajuste de parâmetros em tempo real no código do microcontrolador, e enviado para o servidor selecionando o botão "Enviar", por fim também é criado o evento de interrupção da tarefa do gráfico quando a janela é fechada a partir da função **on_closing_SSS**.

Em última instância, é inicializada a tarefa de criação do gráfico da janela de nome **cria_grafico_SSS**, passando como argumento a janela na qual o gráfico deve ser gerado, as listas que têm os dados e o evento de interrupção para a mesma. Nela, há a criação do ambiente do gráfico, o quadro, os eixos e suas dimensões e ainda, a criação de uma tarefa específica para a atualização desse quadro, chamada **atualiza_grafico_SSS**, na qual há sempre a atualização a cada 2 segundos, tempo necessário para a tarefa de comunicação interagir com o servidor e atualizar os dados das filas. Quando não há mais mensagens a serem enviadas, a fila de envio fica vazia e a conexão com o servidor é fechada. Se essa falha de comunicação ocorre, por diferentes motivos, é feita a tentativa de reconexão mais 4 vezes, se não obtiver resposta, o cliente fecha o soquete. Se houver a reconexão, a cada 2 segundos é enviada uma nova mensagem e o processo se repete. O funcionamento do sistema na visão da tarefa do **cliente** que insere e retira elementos das filas e listas é mostrado na figura 3.9:

Figura 3.9 – Ilustração do funcionamento do sistema de aquisição de dados



Fonte: Autora

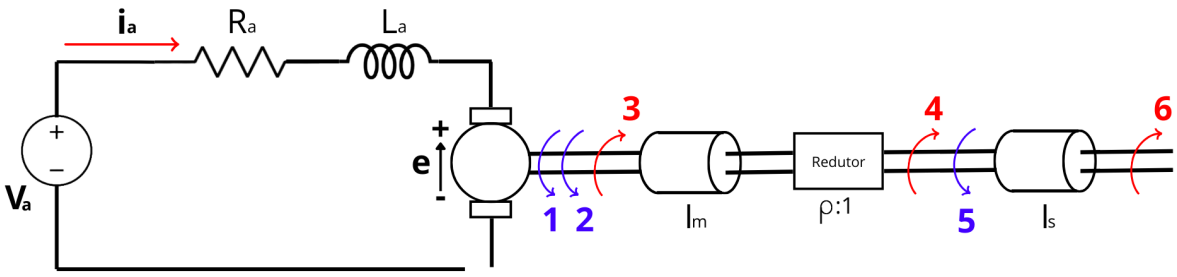
3.3 Modelagem e Controle de Motores CC

Como abordagem inicial, é importante ressaltar que o dispositivo "30A $BLDC$ ESC" é utilizado como um intermediário entre o microcontrolador ESP-32 e o motor BLDC, fazendo com que o microcontrolador precise apenas enviar um sinal de controle PWM ao ESC, o qual, por sua vez, gerencia a sequência de acionamento das bobinas do motor, ajustando a velocidade conforme necessário. Segundo (Hein, 2021), no tópico "Sistemas Simplificados de motores BLDC", quando um sistema representado pelo motor BLDC é acionado por um dispositivo de gerenciamento de comutação de fases como o ESC, o sistema apresenta comportamento dinâmico semelhante ao do motor CC, validando a consideração do modelo deste motor para o controle do sistema real. Assim, foi considerada a modelagem do motor CC para a aplicação das técnicas de controle nesse trabalho.

3.3.1 Modelagem do Motor CC

Para a modelagem matemática do motor CC, são considerados os parâmetros elétricos (resistência e indutância de armadura), parâmetros mecânicos (momento de inércia e coeficiente de atrito viscoso) e parâmetros eletromecânicos (constantes de força eletromotriz e torque) (Canal; Valdiero; Reimbold, 2017). Um esquemático do motor é mostrado na figura 3.10 e os números representam os torques e velocidades angulares envolvidos na modelagem, sendo listados em sequência.

Figura 3.10 – Modelo Motor CC



1. $K_t i_a$ representa o torque gerado pelo rotor em $N.m$, proporcional a corrente i_a em A;
2. ω_m representa a velocidade angular do rotor em rad/s;
3. $B_m\omega_m$ representa o torque contrário devido ao atrito viscoso do motor em $N.m$;
4. $B_s\omega_s$ representa o torque contrário devido ao atrito viscoso de saída em $N.m$;
5. ω_s representa a velocidade angular de saída em rad/s;
6. τ_l representa o torque consumido pela carga em $N.m$;

7. $e = K_v\omega_m$ é a tensão gerada devido ao Efeito Hall na armadura, também chamada de força contraeletromotriz (fcem). Como há a passagem de corrente no circuito de armadura que gira sob efeito de um campo magnético, essa tensão é proporcional à velocidade de rotação. K_v é uma constante de proporcionalidade, chamada de constante de fcem em V.

Estruturando a equação do circuito elétrico a partir das Leis de Kirchhoff, por meio dos somatórios de tensões, tem-se como resultado a equação (3.6):

$$V_a - K_v\omega_m = R_a i_a + La\frac{di_a}{dt} \tag{3.6}$$

Ainda, devido à Lei de Ampère que estabelece que a corrente elétrica que passa por um condutor gera um campo magnético ao redor dele ($\oint \vec{B} \cdot d\vec{s} = \mu_0 i_{in}$), como ocorre no estator devido à corrente do enrolamento de campo, e à força de Lorentz ($\vec{F_B} = I(\vec{L} \times \vec{B})$), (Walker; Resnick; Halliday, 2014) que dado o campo magnético constante gerado no estator e corrente que flui no enrolamento de armadura, então uma força de torque é gerada nos enrolamentos da armadura fazendo o rotor girar. Como o torque é dado pela equação: $\tau = \vec{r} \times \vec{F}$, quando substituída a força da lei de Lorentz, tem-se que o torque é proporcional à corrente de armadura: $\tau = K_t i_a$. Então, analisando o sistema em relação ao movimento, tem-se o momento de inércia (I), os torques e o coeficiente de amortecimento (B), tem-se a equação mecânica associada ao motor CC:

$$I_m\dot{\omega}_m = K_t i_a - B_m\omega_m - B_s\frac{\omega_s}{\rho} - I_s\frac{\dot{\omega}_s}{\rho} - \frac{\tau_l}{\rho} \tag{3.7}$$

O redutor fornece as seguintes relações:

- $\omega_m = \rho\omega_s$
- $\dot{\omega}_m = \rho\dot{\omega}_s$

Fazendo o ajuste de variáveis na equação (3.7):

$$I_m\rho\dot{\omega}_s = K_t i_a - B_m\rho\omega_s - B_s\frac{\omega_s}{\rho} - I_s\frac{\dot{\omega}_s}{\rho} - \frac{\tau_l}{\rho} \tag{3.8}$$

Por fim, obtém-se a equação da parte mecânica representada pela equação (3.9):

$$I\dot{\omega}_s = K_t\rho i_a - B\omega_s - \tau_l \tag{3.9}$$

Tal que $I = I_s + \rho^2 I_m$ representa a inércia total vista da saída e $B = B_s + \rho^2 B_m$ o coeficiente de atrito viscoso visto da saída. Partindo das equações em 3.6 e da 3.9, suas transformadas de Laplace podem ser representadas conforme

$$I_{a_{(s)}} = \frac{1/R_a}{\tau_a s + 1}V_{a_{(s)}} - \frac{K_v\rho/R_a}{\tau_a s + 1}\Omega_{s_{(s)}} \tag{3.10}$$

e

$$\Omega_{s_{(s)}} = \frac{K_t\rho/B}{\tau_m s + 1} I_{a_{(s)}} - \frac{1/B}{\tau_m s + 1}\tau_l \tag{3.11}$$

respectivamente, sendo $\tau_a = \frac{L_a}{R_a}$ a constante de tempo elétrica e $\tau_m = \frac{I}{B}$ a constante de tempo mecânica. Acionando o motor usando uma fonte de tensão, deve-se substituir a equação (3.10) na equação (3.11), além de ser considerado para a simulação um motor sem carga mecânica, ou seja, $\tau_l = 0$, resultando em:

$$\Omega_{s_{(s)}} = \frac{\frac{K_t\rho}{R_a B}}{(\tau_m s + 1)(\tau_a s + 1) + \frac{K_t K_v \rho^2}{R_a B}} V_{a_{(s)}} \tag{3.12}$$

Porém, na prática, é muito comum ter $\tau_m \gg \tau_a$, resultado da dinâmica elétrica ser muito mais rápida que a dinâmica mecânica, devido a indutância de armadura desse motor ser muito pequena, como mostrado na tabela 4.1, fazendo com que o polo relacionado a parte mecânica ($\frac{1}{\tau_m}$) esteja mais próximo do eixo $j\omega$ no plano s e ocorra a dominância do mesmo, implicando na consideração de que $\tau_a = 0$ no modelo. Assim, a seguinte função de transferência é usada como planta para as simulações desse trabalho:

$$\frac{\Omega_{s_{(s)}}}{V_{a_{(s)}}} = \frac{\frac{K_t\rho}{R_a B}}{(\tau_m s + 1) + \frac{K_t K_v \rho^2}{R_a B}} \tag{3.13}$$

ou ainda:

$$\frac{\Omega_{s_{(s)}}}{V_{a_{(s)}}} = \frac{K_m}{\tau s + 1} \tag{3.14}$$

Em que $K_m = \frac{K_t\rho}{R_a B + K_t K_v \rho^2}$, medida em $\frac{A}{kg}\frac{s^2}{m^2}$, é chamada de *constante do motor* e $\tau = \frac{R_a I}{R_a B + K_t K_v \rho^2}$, medida em segundos, é a *constante de tempo* da função de transferência do motor.

3.3.2 Controle PID

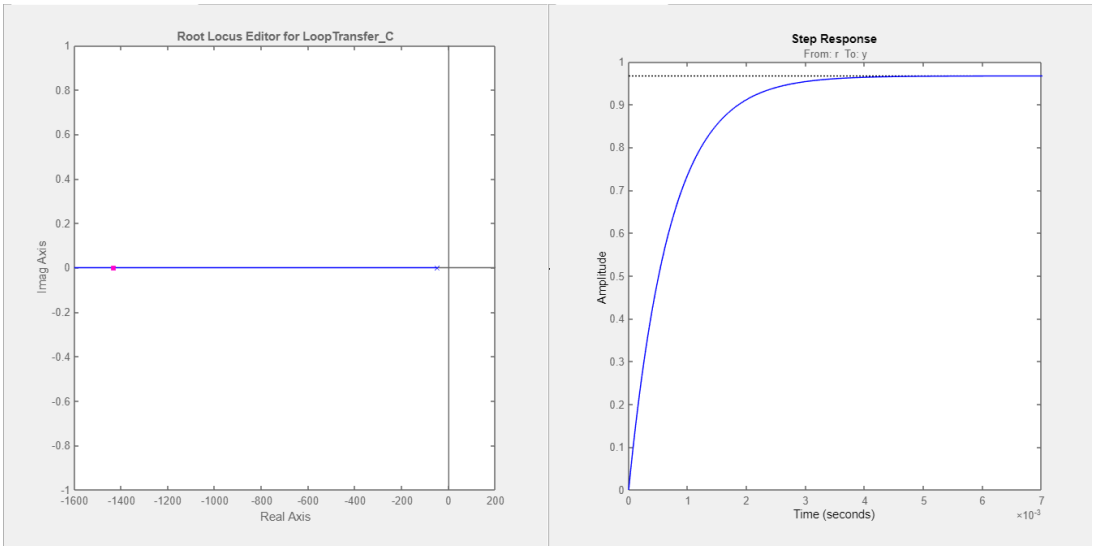
Para o desenvolvimento do controlador PID citado na seção 2.6.3, foi utilizado o comando *sisotool* do Matlab e passado como argumento a função de transferência do motor CC modelado. A figura 3.11 mostra a tela do simulador com o Lugar Geométrico das Raízes à esquerda, destacando o respectivo valor do polo da função de transferência em malha aberta, representado pela equação (3.15a) e o ganho da equação (3.15b), e a resposta ao degrau, à direita.

$$\frac{1}{\tau} = 44{,}8 \tag{3.15a}$$

$$\frac{Km}{\tau} = 1.3030 \tag{3.15b}$$

Com o software é possível alterar a localização dos polos de malha fechada manualmente, além de adicionar polos e zeros para que a resposta ao degrau acompanhe o sinal de referência.

Figura 3.11 – LGR e Resposta ao degrau não compensada do motor CC



Por fim, tomando como ponto de partida as equações 2.15, 2.17 e 2.19 nas suas formas laplacianas, é possível descrever o formato do controlador como segue a equação (3.16):

$$\begin{aligned} G_{c(s)} &= K_p + \frac{K_i}{s} + K_d s \\ G_{c(s)} &= \frac{K_d s^2 + K_p s + K_i}{s} \end{aligned} \tag{3.16}$$

ou ainda no formato ZPK, tem-se a equação (3.17), que explicitam os polos e zeros a serem inseridos pelo controlador:

$$\begin{aligned} G_{c(s)} &= K\frac{(s + z_{c1})(s + z_{c2})}{s} \\ G_{c(s)} &= \frac{K(z_{c1}z_{c2})s^2 + K(z_{c1} + z_{c2})s + K}{s} \end{aligned} \tag{3.17}$$

Em que z_{c1} e z_{c2} são os zeros do compensador. Comparando as equações 3.16 e 3.17, tem-se que os coeficientes podem ser calculados a partir das equações em 3.18:

$$\begin{aligned} K_p &= K(z_{c1} + z_{c2}) \\ K_i &= K \\ K_d &= K(z_{c1}z_{c2}) \end{aligned} \tag{3.18}$$

Assim, para que fosse possível obter o PID, foram adicionados dois zeros representantes dos termos derivativo (K_d) e proporcional (K_p) e um polo representando o termo integral

(K_i), este último na origem do LGR, aumentando o tipo do sistema por um resultando em um melhor desempenho em regime permanente ao diminuir o erro entre o sinal de referência e a resposta do sistema. Ainda, essa referência aponta que quanto mais próximo o zero está dos polos dominantes, maior é seu efeito no resíduo, ou a amplitude, de uma componente da resposta na fase transitória. Por fim, é importante ressaltar que determinados polos são considerados dominantes quando outros polos do sistema estão a uma distância de referência de ser 5 vezes maior à esquerda no eixo real, sendo muito mais negativos, como mostrado em (Nise, 2012).

Neste contexto, como mostrado em (Castrucci; Bittar; Sales, 2011), o comportamento de zeros adicionais é análogo ao de polos em relação aos polos dominantes. Assim, o zero representante ao termo derivativo foi posicionado em um primeiro momento à uma distância 5 vezes mais longe ao polo da planta no eixo real, de forma que a ação derivativa que ele implementa fosse atenuada, resultando em uma reação mais controlada aos degraus da entrada, evitando respostas com subidas rápidas às variações do sistema, que podem levar à instabilidade como discutido na seção 2.6.3.3. Ademais, a escolha da localização dos polos em malha fechada foi feita visando o controle de amortecimento do sistema. Assim, quanto mais próximos do eixo real, a influência de frequências complexas são atenuadas, resultando em menores oscilações e efeito de sobressinal para o sistema.

A arquitetura do controlador é mostrada na figura 3.12 e, com a sintonia dos ganhos e zeros adicionados para que a resposta ao degrau tivesse um erro de regime permanente nulo e que as características do sistema em regime transitório como tempo de resposta, tempo de subida e sobressinal não fossem alteradas, o bloco "C" foi configurado como mostra a figura 3.13 e o resultado desse procedimento pode ser visto na resposta ao degrau do LGR compensado mostrado na figura 3.14.

Figura 3.12 – Arquitetura do controle PID

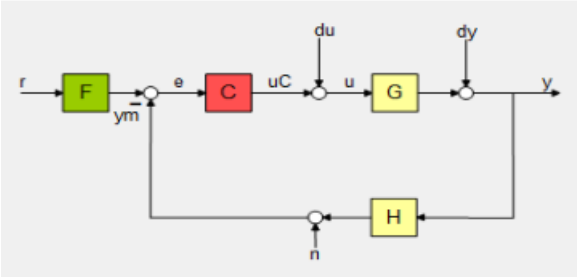


Figura 3.13 – Controle PID da planta do motor CC

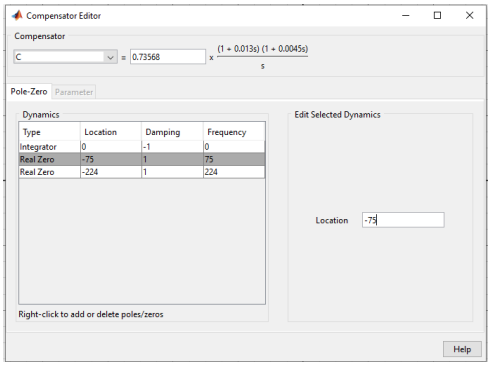
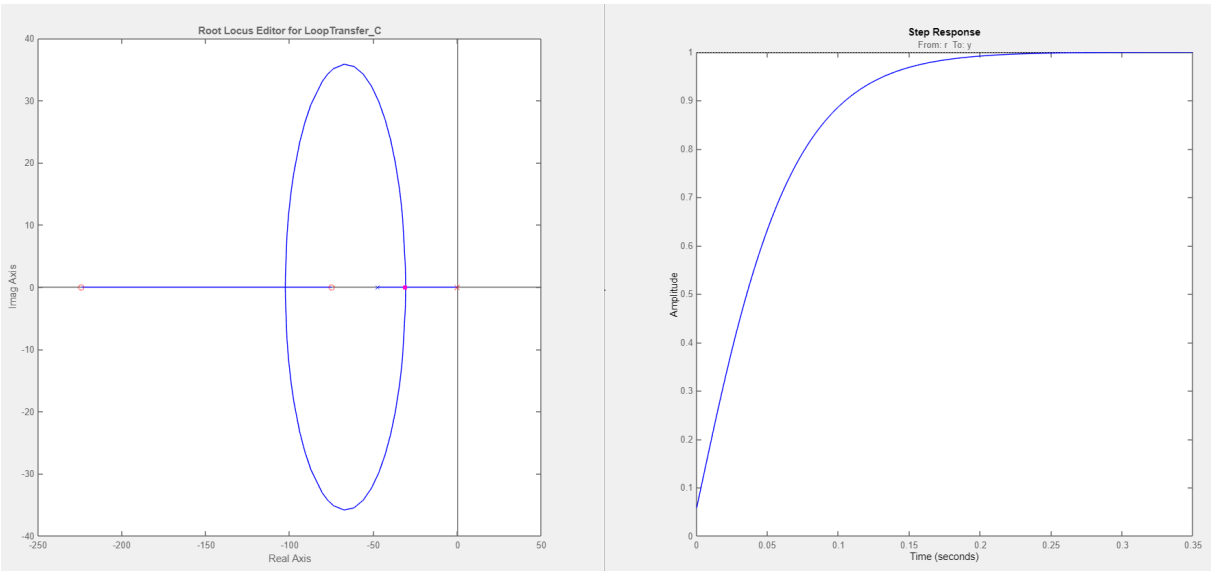


Figura 3.14 – Resposta ao degrau e LGR do motor CC



3.3.3 Controle Adaptativo por Modelo de Referência

Com o objetivo de modelar o controle MRAC faz-se necessário primeiramente estipular o modelo de referência desejado. Este modelo representa o comportamento desejado da planta em malha fechada, assim, como referência para a planta desse projeto, foi utilizada uma função de transferência de primeiro grau, para que os graus das funções do modelo e do processo fossem os mesmos, não sendo necessário aumentar a complexidade do sistema. E a constante de tempo foi escolhida com valor igual ao da função de transferência da planta, já que não é necessário que o sistema controlado seja mais rápido ou mais lento que o da planta original. Assim, a função de transferência de referência para a planta é da forma como mostra a equação (3.19):

$$\frac{dy_{m(t)}}{dt} = -a_m y_{(t)} + b_m u_{c(t)} \xrightarrow{\mathcal{L}} Y_{m(s)} = \frac{b_m}{s + a_m} U_{c(s)} \tag{3.19}$$

Com os coeficientes genéricos sendo definidos como: $a_m = b_m = \frac{1}{\tau}$, tem-se a equação desejada para esse trabalho, e pode ser descrita no domínio das frequências tomando a transformada de Laplace como mostra a equação (3.20):

$$G_{m(s)} = \frac{Y_{m(s)}}{U_{c(s)}} = \frac{1}{\tau s + 1} \tag{3.20}$$

Ainda, a planta tem sua função descrita na equação (3.14), que pode ser descrita genericamente no domínio do tempo como mostra a equação (3.21):

$$\frac{dy_{(t)}}{dt} = -a y_{(t)} + b u_{(t)} \xrightarrow{\mathcal{L}} Y_{(s)} = \frac{b}{s + a} U_{(s)} \tag{3.21}$$

Com os os parâmetros $a = \frac{1}{\tau}$ e $b = \frac{K_m}{\tau}$.

Por fim, é necessário estipular a forma do controlador, aqui tem-se um modelo de dois parâmetros que é mostrado na equação (3.22):

$$u_{(t)} = \theta_1 u_{c(t)} - \theta_2 y_{(t)} \xrightarrow{\mathcal{L}} U_{(s)} = \theta_1 U_{c(s)} - \theta_2 Y_{(s)} \tag{3.22}$$

Utilizando as formas em Laplace das equações (3.21) e (3.22) e fazendo-se as devidas manipulações, tem-se que a saída da planta está diretamente relacionada aos parâmetros de adaptação e à entrada de controle como mostra a equação (3.23):

$$Y_{(s)} = \frac{b\theta_1}{s + a + b\theta_2} U_{c(s)} \tag{3.23}$$

Se for possível conhecer a planta e se deseja-se obter uma que siga perfeitamente o modelo estipulado, ou seja, a referência, então pode-se substituir a equação (3.22) na equação (3.21) e por fim igualar o resultado com a equação (3.19). Após esses procedimentos e eliminando-se as variáveis $y_{(t)}$, $u_{(t)}$ e $u_{c(t)}$ tem-se os valores de θ_1 e θ_2 para que seja possível que o erro entre a saída da planta e do referência seja nulo:

$$\begin{aligned} \theta_1 &= \frac{b_m}{b} \\ \theta_2 &= \frac{a_m - a}{b} \end{aligned} \tag{3.24}$$

Porém, para o controle adaptativo, não é necessário conhecer os parâmetros a e b da planta, mas sim que o controle os calcule de acordo com a dinâmica dela, ou seja, sua saída. Nesse sentido, para aplicar a regra MIT, o erro é da forma:

$$e_{(t)} = y_{(t)} - y_{m(t)} \xrightarrow{\mathcal{L}} E_{(s)} = Y_{(s)} - Y_{m(s)} \tag{3.25}$$

Substituindo-se as equações (3.19) e (3.23) acima para tomar as derivadas parciais dos parâmetros em relação ao erro, tem-se que:

$$E_{(s)} = \frac{b\theta_1}{s + a + b\theta_2} U_{c(s)} - \frac{b_m}{s + a_m} U_{c(s)} \tag{3.26}$$

E as derivadas parciais resultantes são da forma:

$$\begin{aligned} \frac{\partial E_{(s)}}{\partial \theta_1} &= \frac{b}{s + a + b\theta_2} U_{c(s)} \\ \frac{\partial E_{(s)}}{\partial \theta_2} &= -\frac{b^2\theta_1}{(s + a + b\theta_2)^2} U_{c(s)} = -\frac{b}{s + a + b\theta_2} Y_{(s)} \end{aligned} \tag{3.27}$$

Porém, para ser possível utilizar essas fórmulas é necessário fazer considerações, já que os parâmetros a e b não precisam ser conhecidos do ponto de vista do controle. Assim, quando o sistema for um bom seguidor do modelo estipulado, uma aproximação razoável é a de que

$s + a + b\theta_2 \approx s + a_m$ como mostrado em (Åström; Wittenmark, 2008). Então, as equações ficam da forma:

$$\begin{aligned} \frac{\partial E_{(s)}}{\partial \theta_1} &= \frac{b}{s + a_m} U_{c(s)} \\ \frac{\partial E_{(s)}}{\partial \theta_2} &= -\frac{b}{s + a_m} Y_{(s)} \end{aligned} \tag{3.28}$$

Com as derivadas parciais dos parâmetros em relação ao erro calculadas, então aplica-se a última parte da regra MIT: para que o ajuste de parâmetros seja alcançada, é considerando a função de perda como a da 2.23. Ainda, há a troca de variáveis $\gamma = -\gamma' \frac{b}{a_m}$, finalmente a derivada dos parâmetros no tempo deve ser da forma:

$$\begin{aligned} \frac{d\theta_1}{dt} &= -\gamma \frac{a_m}{s + a_m} U_{c(s)} e \\ \frac{d\theta_2}{dt} &= \gamma \frac{a_m}{s + a_m} Y_{(s)} \end{aligned} \tag{3.29}$$

Assim, como já definido anteriormente, o parâmetro $a_m = \frac{1}{\tau}$ e as equações para o projeto desse controlador são:

$$\begin{aligned} \frac{d\theta_1}{dt} &= -\gamma \left(\frac{1}{\tau s + 1} U_{c(s)} \right) e \\ \frac{d\theta_2}{dt} &= \gamma \left(\frac{1}{\tau s + 1} Y_{(s)} \right) e \end{aligned} \tag{3.30}$$

4 Resultados e Discussões

Neste capítulo serão mostrados os resultados obtidos no trabalho, divididos em: testes do sistema embarcado, testes do sistema de aquisição de dados, e simulações e análises das técnicas de controle.

4.1 Testes do Sistema Embarcado

Nesta seção, serão mostrados testes do sistema embarcado. Foi decidido que o foco dos testes seria a validação experimental do funcionamento do software embarcado, utilizando o protótipo do hardware em placa perfurada. Instrumentos de medição foram utilizados para o cálculo da precisão e exatidão das medições dos sensores. Também foram feitos testes que mostram a periodicidade de execução das tarefas, para análise de se estão cumprindo os requisitos temporais do sistema. Todos os testes que serão apresentados consistem em testes de integração, e foram realizados para cada bloco do software embarcado.

4.1.1 Geração de ponto de acesso Wi-Fi e criação do servidor TCP/IP

A Figura 4.1 mostra que a geração do ponto de acesso Wi-Fi e a criação do servidor TCP/IP foram feitas com sucesso e o *socket* aguarda até que um cliente solicite uma conexão. Em seguida, é possível ver que a solicitação foi realizada por um cliente, e a conexão foi aceita. Esta sequência se repete em todos os demais testes que envolvem a comunicação por TCP/IP.

```
I (16882) wifi softAP: wifi_init_softap finished. SSID:WIFI_TILTROTOR password:12345678 channel:1
I (16882) esp_netif_lwip: DHCP server started on interface WIFI_AP_DEF with IP: 192.168.4.1
I (16902) main_task: Returned from app_main()
I (16902) example: Socket created
I (16902) example: Socket bound, port 3333
I (16912) example: Socket listening
I (98432) wifi:new:<1,1>, old:<1,1>, ap:<1,1>, sta:<255,255>, prof:1
I (98432) wifi:station: 14:ac:60:3e:51:55 join, AID=1, bgn, 40U
I (98452) wifi softAP: station 14:ac:60:3e:51:55 join, AID=1
I (98482) esp_netif_lwip: DHCP server assigned IP to a client, IP is: 192.168.4.2
I (98492) wifi:<ba-add>idx:2 (ifx:1, 14:ac:60:3e:51:55), tid:0, ssn:1, winSize:64
I (121352) example: Socket accepted ip address: 192.168.4.2
```

Figura 4.1 – Inicialização da rede Wi-Fi e do servidor TCP/IP, e conexão com o cliente.

4.1.2 Medição da distância entre o robô aéreo e o solo

4.1.2.1 Simulação simplificada de decolagem

Como um teste inicial da medição de distância pelo sensor ultrassônico, foi utilizado um anteparo, uma caixa de papelão de altura e largura iguais a $57cm$, para simular o solo se afastando do robô aéreo, enquanto este estava fixo, a fim de simular de forma simplificada

uma situação de decolagem. O anteparo foi inicialmente colocado em uma distância próxima, aproximadamente a 8cm do sensor, por ser uma distância próxima ao valor real que seria lido no momento em que o robô aéreo estivesse no solo e o sensor estivesse realmente fixado no local já definido da estrutura mecânica. O anteparo foi, então, sendo afastado, em uma velocidade aproximadamente constante, até a distância próxima de 1,6m, simulando um voo com esta altitude.

As medições foram realizadas e armazenadas na estrutura global. Para a realização dos testes, foi criada uma nova tarefa, uma "tarefa de visualização", que tem como objetivo ler a estrutura global e imprimir os dados no monitor serial do "$ESP32$", utilizando a função "ESP_LOGI". Esta tarefa acessa a estrutura por meio do "$mutex$", e para este teste foi executada de forma periódica a cada "500ms". Os valores das medições de distância em metros, assim como o valor do tempo em milissegundos no qual a tarefa de visualização foi executada a cada vez, contado a partir da inicialização do microcontrolador, podem ser vistos na Figura 4.2, que mostra o monitor serial do "$ESP32$"no momento do teste. Os dados de distância têm a resolução na faixa dos milímetros, com incerteza de ±3mm.

```
I (167858) UltrasonicSensor: Measured distance: 0.081m
I (168358) UltrasonicSensor: Measured distance: 0.082m
I (168858) UltrasonicSensor: Measured distance: 0.101m
I (169358) UltrasonicSensor: Measured distance: 0.140m
I (169858) UltrasonicSensor: Measured distance: 0.198m
I (170358) UltrasonicSensor: Measured distance: 0.238m
I (170858) UltrasonicSensor: Measured distance: 0.288m
I (171358) UltrasonicSensor: Measured distance: 0.346m
I (171858) UltrasonicSensor: Measured distance: 0.429m
I (172358) UltrasonicSensor: Measured distance: 0.504m
I (172858) UltrasonicSensor: Measured distance: 0.592m
I (173358) UltrasonicSensor: Measured distance: 0.672m
I (173858) UltrasonicSensor: Measured distance: 0.747m
I (174358) UltrasonicSensor: Measured distance: 0.822m
I (174858) UltrasonicSensor: Measured distance: 0.867m
I (175358) UltrasonicSensor: Measured distance: 0.917m
I (175858) UltrasonicSensor: Measured distance: 1.003m
I (176358) UltrasonicSensor: Measured distance: 1.091m
I (176858) UltrasonicSensor: Measured distance: 1.162m
I (177358) UltrasonicSensor: Measured distance: 1.267m
I (177858) UltrasonicSensor: Measured distance: 1.363m
I (178358) UltrasonicSensor: Measured distance: 1.445m
I (178858) UltrasonicSensor: Measured distance: 1.492m
I (179358) UltrasonicSensor: Measured distance: 1.567m
I (179858) UltrasonicSensor: Measured distance: 1.633m
```

Figura 4.2 – Monitor serial do "$ESP32$"mostrando as medições da distância entre o robô aéreo e o anteparo em uma simulação simplificada de decolagem.

Em seguida, estes dados mostrados no monitor serial durante o teste foram utilizados para a geração de um gráfico, pelo software "$Matlab$", da distância medida pelo sensor pelo tempo no qual a tarefa de visualização foi executada. O tempo foi deslocado, subtraindo-se o tempo necessário para a inicialização do microcontrolador, para que, no gráfico, o teste se

iniciasse no tempo zero. O gráfico pode ser visto na Figura 4.3.

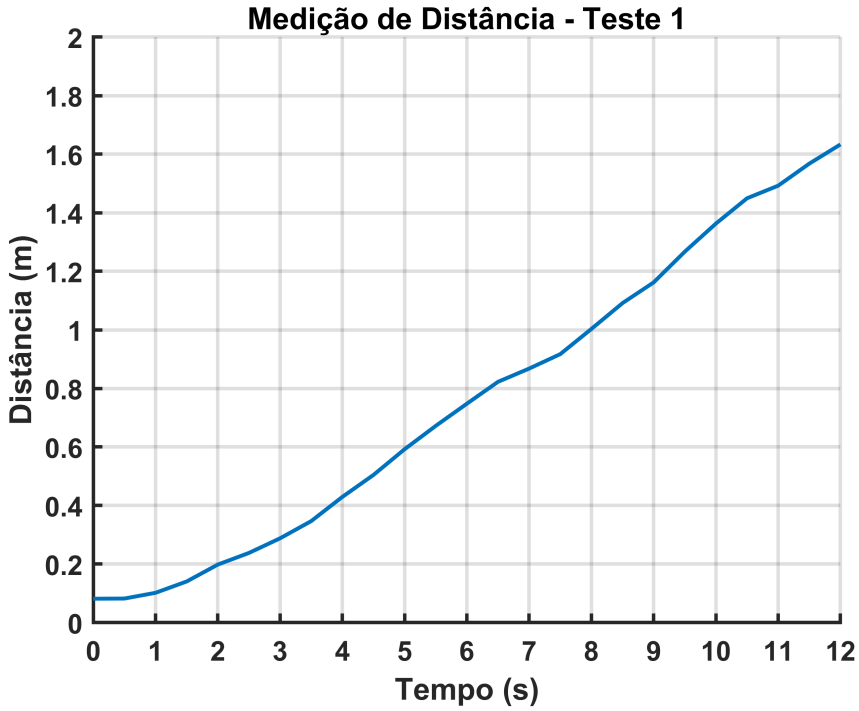


Figura 4.3 – Gráfico da distância medida pelo tempo de execução da tarefa de visualização, gerado no software "$Matlab$"utilizando os dados do monitor serial.

A não linearidade visível no gráfico ocorre por conta da movimentação da caixa ter sido feita de forma manual, por isso a velocidade não é exatamente constante. A curva vista no gráfico mostra que a medição está condizente com o resultado esperado para o teste; porém, os próximos testes são necessários para que a medição e a periodicidade da tarefa de medição sejam realmente avaliadas.

4.1.2.2 Validação de precisão e exatidão da medição

Estes testes têm como objetivo validar a medição, assim como calcular a precisão e exatidão da medição; assim os testes não avaliam requisitos temporais.

Foi utilizado o mesmo anteparo do teste anterior para simulação do solo se afastando do robô aéreo, enquanto o mesmo estava em uma posição fixa. Foram feitas marcações no chão com uma fita branca, com o auxílio do instrumento de medição trena, para que fossem consideradas como os valores de referência para as medições de distância. Foram feitas nove marcações, tendo início na distância $0m$, até a distância de $4m$, por este ser o alcance do sensor conforme sua documentação; a distância entre uma marcação e a marcação seguinte é $0{,}5m$. A mesa onde estava localizado o robô aéreo estava na posição $0m$. A Figura 4.4 mostra um diagrama que ilustra como os testes foram realizados; enquanto a Figura 4.5 é uma foto do local de realização dos testes, que mostra as marcações feitas para os testes.

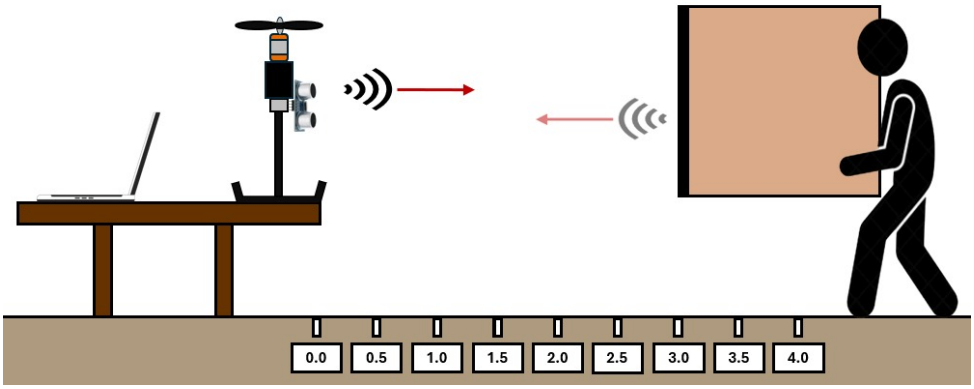


Figura 4.4 – Diagrama que representa como os testes foram realizados.

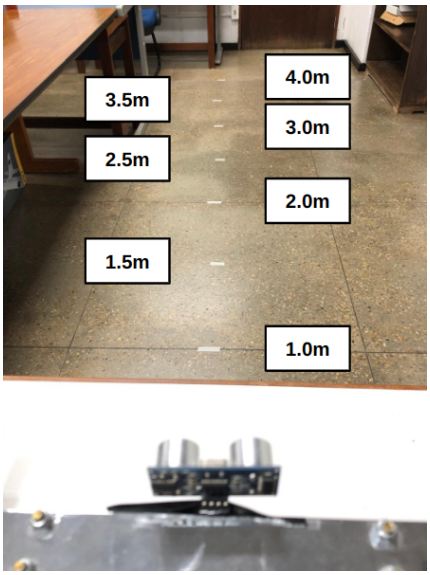


Figura 4.5 – Foto mostrando as marcações feitas para os testes de precisão e exatidão das medições de distância.

Cada teste foi realizado de forma estática, ou seja, o anteparo foi primeiramente posicionado na marcação, e depois iniciaram-se as medições. O anteparo permaneceu aproximadamente fixo durante todo o tempo em que as medições foram feitas, e com a sua superfície detectada pelo sensor posicionada aproximadamente em orientação vertical. Foi feito um número grande de medições para cada marcação, sempre maior do que 100 medições. O processo foi repetido igualmente para cada marcação.

Para visualização das medições, foi utilizada a mesma "tarefa de visualização"do teste anterior, mas alterou-se seu período para $50ms$, o mesmo período da tarefa que realiza as medições, para que um maior número de medições fosse visto e considerado. Os dados dos testes mostrados no monitor serial foram coletados e, por meio do software "$Matlab$", foram calculadas a precisão e a exatidão das medições do sensor, conforme explicação em 2.4.4.

Primeiramente, foram excluídas as 10 primeiras medições, pois foi percebido que as primeiras medições, quando o microcontrolador está inicializando, pareciam inconsistentes. Em seguida, foram escolhidas 10 medições aleatórias, entre as mais de 100 restantes, para representarem a amostra. Então, foi calculada a média entre os valores medidos, e o desvio padrão amostral. Todo o processo foi repetido para cada marcação. Os resultados podem ser vistos na tabela da Figura 4.6.

Referência (m)	0,020	0,500	1,000	1,500	2,000	2,500	3,000	3,500	4,000
Amostras aleatórias das medições de distância (m)	0,022	0,490	1,002	1,488	1,976	2,440	2,984	4,000	3,959
	0,022	0,494	1,004	1,487	1,973	2,442	2,966	4,000	4,000
	0,026	0,492	1,003	1,488	1,989	2,449	2,969	3,458	3,958
	0,022	0,490	1,008	1,488	1,986	2,435	2,989	4,000	4,000
	0,026	0,487	1,006	1,490	1,980	2,447	2,992	3,464	3,953
	0,022	0,493	1,001	1,481	1,972	2,443	2,969	3,470	3,970
	0,026	0,495	1,003	1,486	1,974	2,452	4,000	3,449	3,962
	0,026	0,495	1,005	1,488	1,973	2,449	2,971	3,477	3,954
	0,026	0,493	1,003	1,494	1,977	2,441	2,991	4,000	3,964
	0,026	0,496	1,002	1,485	1,982	2,441	2,955	4,000	4,000
Média (m)	0,024	0,493	1,004	1,487	1,978	2,444	3,079	3,732	3,972
Desvio padrão amostral (m)	0,002	0,003	0,002	0,003	0,006	0,005	0,324	0,283	0,020

Figura 4.6 – Tabela com as amostras aleatórias das medições de distância, e desvio padrão amostral.

Pela tabela, é possível ver que o desvio padrão amostral, em metros, se aproxima de zero para todas as marcações de referência; apesar de, geralmente, aumentar à medida que a distância aumenta. Como o desvio padrão amostral é pequeno, a precisão da medição é alta 2.4.4. A quantidade de dados para cada amostra, 10, foi escolhida pois, pelo fato de os dados serem escolhidos de forma aleatória, o cálculo do desvio padrão amostral para apenas 10 dados mostrou-se próximo do valor para números maiores; isto foi percebido após a observação do desvio padrão amostral para diversas amostras distintas.

Para a determinação da exatidão das medições, foram calculados, para cada medição de cada amostra: o erro absoluto, que consiste na diferença entre o valor medido e o valor de referência, em valores absolutos; e, em seguida, a exatidão percentual, por meio da equação mostrada em 2.4.4. Por fim, foi calculada a média das exatidões, para cada marcação de referência. Todos os resultados podem ser vistos na tabela na Figura 4.7.

Exatidão (%) para cada amostra aleatória	92,00	97,96	99,81	99,18	98,78	97,58	99,46	85,71	98,98
	92,00	98,78	99,64	99,15	98,63	97,68	98,85	85,71	100,00
	68,50	98,32	99,70	99,21	99,45	97,97	98,96	98,80	98,96
	92,00	97,94	99,17	99,19	99,31	97,39	99,63	85,71	100,00
	68,50	97,48	99,40	99,34	99,02	97,89	99,73	98,97	98,82
	92,00	98,68	99,88	98,76	98,58	97,73	98,98	99,13	99,26
	68,00	98,92	99,71	99,03	98,72	98,08	66,67	98,54	99,05
	68,50	99,02	99,53	99,18	98,66	97,96	99,03	99,33	98,85
	68,00	98,68	99,70	99,58	98,85	97,63	99,69	85,71	99,11
	68,50	99,14	99,77	98,98	99,12	97,63	98,50	85,71	100,00
Exatidão média (%)	77,80	98,49	99,63	99,16	98,91	97,75	95,95	92,33	99,30

Figura 4.7 – Tabela que mostra a exatidão das medições, e exatidão média.

Pela tabela, é possível perceber que a exatidão média é alta, maior que 90%, para a maioria das marcações, exceto para a primeira marcação, o que mostra que o sensor é menos exato para uma distância próxima do valor mínimo do seu alcance.

4.1.2.3 Simulação de decolagem com análise dos requisitos temporais

Este teste tem o objetivo de validar se o requisito de execução temporal da tarefa está sendo cumprido. Para o teste, foi utilizado o mesmo anteparo para simular o solo que se afasta do robô aéreo, enquanto ele está em uma posição fixa. Dessa forma, foi feita uma nova simulação de uma situação de decolagem, começando na distância mínima de $14cm$, entre o anteparo e o sensor, até a distância máxima de $4m$, por ser o alcance máximo do sensor, com o anteparo se afastando a uma velocidade aproximadamente constante.

A mesma "tarefa de visualização"foi utilizada para imprimir os dados medidos no monitor serial. O período da tarefa se manteve o mesmo da tarefa de medição, $50ms$. Porém, para a análise correta da periodicidade da tarefa de medição, o tempo considerado neste teste foi o tempo em microssegundos coletado no momento em que a medição foi realizada e armazenado na mesma estrutura da medição. Os dados coletados no teste podem ser vistos no gráfico da medição pelo tempo no qual a medição foi realizada, mostrado na Figura 4.8. O tempo foi deslocado, subtraindo-se o tempo necessário para a inicialização do microcontrolador, e o anteparo manteve-se parado até o tempo igual a $4s$.

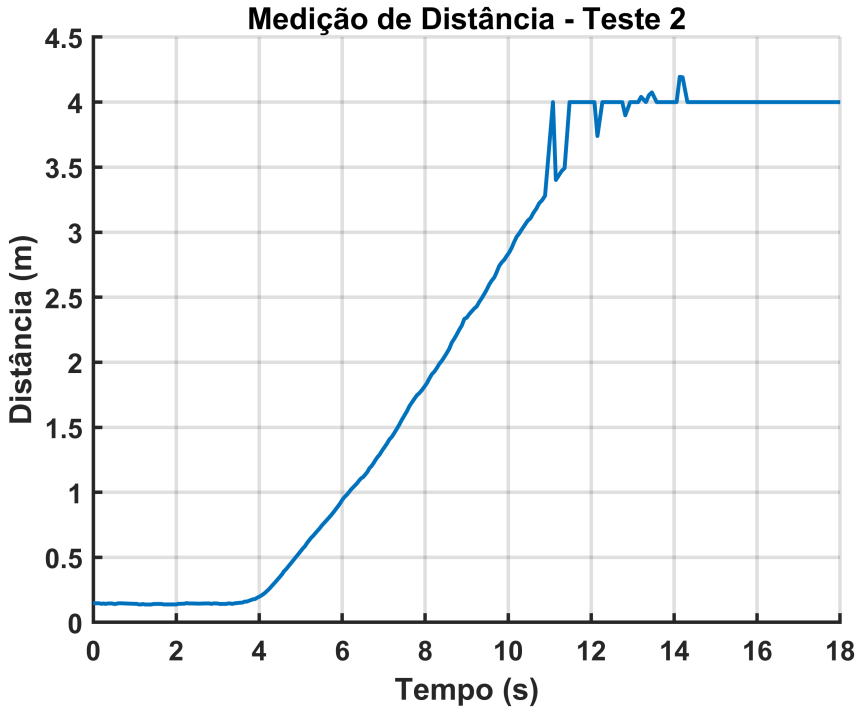


Figura 4.8 – Gráfico da medição pelo tempo no qual a medição foi realizada.

No gráfico, observa-se que a partir do tempo igual a 4s, a curva da medição é aproximadamente linear, conforme o esperado. É possível notar que a medição satura em 4m, comportamento definido no software por ser o alcance máximo do sensor conforme sua documentação. As medições acima de 4m não eram esperadas. A partir do tempo igual a 11s, observa-se presença de não linearidades, passando para o valor de saturação, o que demonstra as limitações do ângulo de medição do sensor, perceptível pelo tamanho limitado da superfície do anteparo utilizado. Múltiplas reflexões no solo e na superfície do anteparo também são possíveis causas das não linearidades.

Então, os dados do mesmo teste foram utilizados para geração do gráfico da Figura 4.9, que mostra o período da tarefa de medição, para cada medição no tempo do teste.

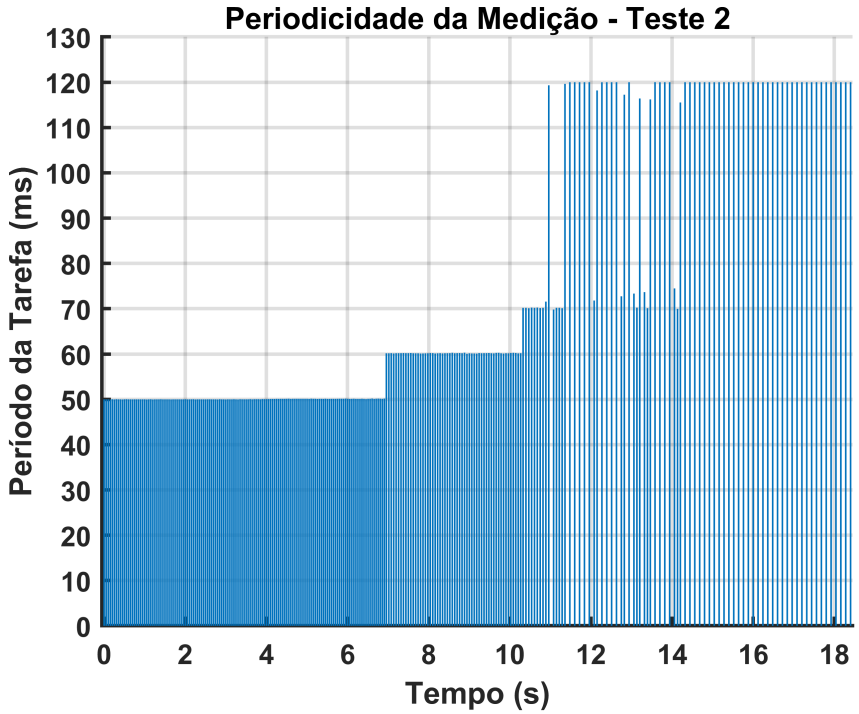


Figura 4.9 – Gráfico da periodicidade da tarefa de medição.

Pelo gráfico, observa-se que na região até o tempo igual a $7s$ a periodicidade cumpre o requisito temporal estabelecido, $50ms$. Após $7s$ e até o final do teste, a tarefa não cumpre o requisito temporal, tendo um período variável entre $50ms$ e $120ms$, e que aumenta à medida que a distância aumenta. Como considera-se a distância máxima para a determinação do tempo de ciclo da tarefa, então a tarefa não cumpre o requisito temporal. Pela análise do software embarcado, conclui-se que a causa deste problema é que o período definido para a tarefa, $50ms$, soma-se ao tempo em que a tarefa espera a ida e o retorno da onda sonora, por isso o valor aumenta com a distância. Isto deve ser corrigido no software embarcado, para que o requisito temporal seja cumprido para todos os valores na região de alcance do sensor.

4.1.2.4 Integração da medição com as filas circulares e a comunicação TCP/IP

Por fim, este teste mostra a integração da tarefa de medição com a tarefa que coloca seus dados na fila circular, e com a tarefa que esvazia a fila circular, enviando os dados em uma mensagem JSON por TCP/IP. Para o teste, foi utilizada a ferramenta "*Netcat*", que permite a inicialização de um cliente TCP, pelo comando no terminal: "*nc* 192.168.4.1 3333"; com o endereço IP do servidor, e a porta na qual o servidor está escutando conexões.

O resultado do teste de comunicação TCP/IP pode ser visto na Figura 4.10. Quando ocorre a conexão, inicia-se a comunicação, então o cliente envia o comando "*get_sonar*", e o servidor responde com uma mensagem JSON contendo as medições das distâncias que estavam armazenadas na fila circular, assim como o tempo no qual cada medição foi realizada.

```
I (193830) communication: Received 41 bytes: { "command": "get_sonar", "id": 234624 }

I (193840) communication: Response sent: {
        "response":     "get_sonar",
        "id":   234624,
        "distance":     [0.113948875, 0.11393825, 0.10920587500000002, 0.00918, 0.109210125, 0.10860875],
        "time": [192612500, 192712473, 192912445, 193011853, 193112441, 193212437]
}
```

Figura 4.10 – Mensagens entre o servidor e o cliente TCP/IP, para o comando que solicita os dados de medições de distância.

4.1.3 Medição da velocidade dos motores

É possível que o ESC seja configurado para que o motor funcione em velocidades menores do que as apresentadas nestes testes, e pode-se retirar o saturador para que ele alcance velocidades maiores, porém não consiste no foco dos testes de validação da medição de velocidade angular.

4.1.3.1 Simulação simplificada de decolagem

Como um teste inicial da medição de velocidade utilizando o sensor ótico, foi realizada uma simulação simplificada de uma decolagem, na qual inicialmente o motor estava parado, e depois foi enviado um comando para que ele atingisse a velocidade angular de 993rad/s, ou 9482RPM, sem carga.

Durante todo o teste, a velocidade angular do motor foi medida e armazenada na estrutura global. Para a visualização dos dados, foi criada uma nova tarefa, uma "tarefa de visualização", para ler a estrutura global e imprimir os dados no monitor serial do "$ESP32$. A tarefa acessa a estrutura por meio de "$mutex$", e neste teste foi executada de forma periódica a cada 50ms. Os valores das medições de velocidade angular em rad/s, assim como o tempo em milissegundos no qual a tarefa de visualização foi executada a cada vez, a partir da inicialização do microcontrolador, podem ser vistos na Figura 4.11, que mostra o monitor serial no momento do teste.

```
I (16392) FotoSensor: Angular speed: 0.0000 rad/s
I (16442) FotoSensor: Angular speed: 0.0000 rad/s
I (16492) FotoSensor: Angular speed: 0.3892 rad/s
I (16542) FotoSensor: Angular speed: 155.6406 rad/s
I (16592) FotoSensor: Angular speed: 279.7419 rad/s
I (16642) FotoSensor: Angular speed: 280.2804 rad/s
I (16692) FotoSensor: Angular speed: 476.2740 rad/s
I (16742) FotoSensor: Angular speed: 721.8008 rad/s
I (16792) FotoSensor: Angular speed: 925.9860 rad/s
I (16842) FotoSensor: Angular speed: 993.2045 rad/s
I (16892) FotoSensor: Angular speed: 993.2045 rad/s
I (16942) FotoSensor: Angular speed: 993.2045 rad/s
I (16992) FotoSensor: Angular speed: 993.2045 rad/s
I (17042) FotoSensor: Angular speed: 993.2045 rad/s
I (17092) FotoSensor: Angular speed: 993.2045 rad/s
```

Figura 4.11 – Monitor serial do "*ESP*32"mostrando as medições de velocidade angular do motor em uma simulação simplificada de decolagem.

Em seguida, os dados mostrados no monitor serial durante o teste foram utilizados para geração de um gráfico, pelo software "*Matlab*", da velocidade angular do motor medida pelo tempo no qual a tarefa de visualização foi executada. O tempo foi deslocado, subtraindo-se o tempo de inicialização do microcontrolador. O gráfico pode ser visto na Figura 4.12

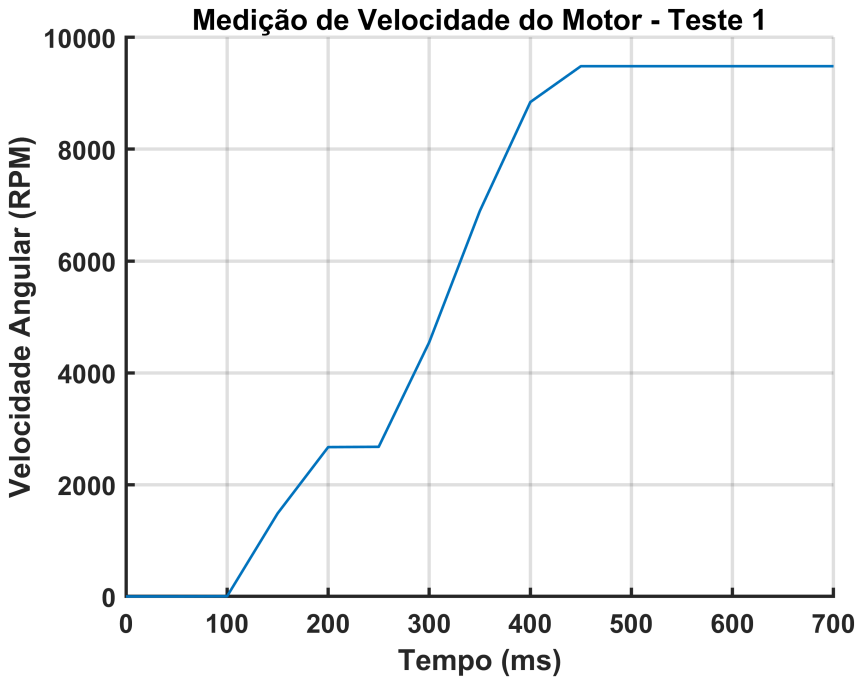


Figura 4.12 – Gráfico da velocidade angular do motor medida pelo tempo no qual a tarefa de visualização foi executada, gerado no software "*Matlab*"utilizando os dados do monitor serial.

A curva vista no gráfico mostra que a medição da velocidade angular está condizente com o resultado esperado para o teste. Porém, são necessários os próximos testes para que a medição e a periodicidade da tarefa de medição sejam realmente avaliadas.

4.1.3.2 Validação de precisão e exatidão da medição

Estes testes têm o objetivo de validar a medição, assim como calcular a precisão e exatidão da medição; por isso os testes não avaliam os requisitos temporais.

Para cada teste, foi escolhido um valor de referência de velocidade angular, para ser enviado como comando para o motor e validado utilizando o instrumento de medição tacômetro. Foram escolhidos seis valores de referência, com intervalos de entre 200 e 600RPM entre eles, e colocou-se um saturador para que o motor não atingisse sua velocidade máxima. Em todos os testes, o motor está sem carga. A medição utilizando o tacômetro foi feita conforme a foto na Figura 4.13.

Figura 4.13 – Foto mostrando como foram feitas as medições com o tacômetro.

Cada teste foi realizado de forma estática, ou seja, os testes foram realizados separadamente, e esperou-se a velocidade do motor ficar estável, por um curto tempo após o comando, para que o valor da medição fosse considerado. Foi feito um número grande de medições para cada valor de referência, acima de 100 medições. O processo foi repetido igualmente para cada valor de referência.

Para visualização das medições, foi utilizada a mesma "tarefa de visualização"do teste anterior, mas alterou-se seu período para 10ms, para que um número maior de medições fosse visto e considerado. Os dados dos testes mostrados no monitor serial foram coletados e, por meio do software "$Matlab$", foram calculadas a precisão e a exatidão das medições do sensor, conforme explicação em 2.4.4.

Foram escolhidas 10 medições aleatórias para representarem a amostra. Então, foi calculada a média entre os valores medidos, e o desvio padrão amostral. Todo o processo foi repetido para cada valor de referência. Os resultados podem ser vistos na tabela da Figura 4.14

Referência (RPM)	7515,00	8075,00	8390,00	8764,00	9084,00	9334,00
Amostras aleatórias das medições de velocidade angular do motor (RPM)	7507,06	8073,03	8386,05	8764,43	9075,47	9332,51
	7525,15	8081,12	8386,81	8758,42	9070,59	9328,76
	7538,53	8075,75	8397,75	8775,17	9071,75	9324,12
	7520,65	8079,67	8394,60	8745,65	9074,00	9327,13
	7518,46	8083,16	8394,90	8766,84	9074,36	9329,77
	7519,57	8066,16	8397,94	8759,43	9069,92	9327,25
	7518,67	8078,37	8387,30	8759,88	9072,27	9328,79
	7520,02	8080,11	8389,35	8778,80	9075,97	9323,36
	7518,99	8078,81	8389,34	8765,47	9075,71	9331,33
	7520,74	8079,31	8387,78	8763,23	9072,30	9318,31
Média (RPM)	7520,78	8077,55	8391,18	8763,73	9073,23	9327,13
Desvio padrão amostral (RPM)	7,73	4,88	4,63	9,19	2,17	4,22

Figura 4.14 – Tabela com as amostras aleatórias das medições de velocidade angular, e desvio padrão amostral.

Pela tabela, é possível ver que o desvio padrão amostral, em RPM, é pequeno, pois os valores de referência têm valor elevado, e não há uma relação perceptível de aumento à medida que a velocidade angular aumenta. Como o desvio padrão amostral é pequeno, a precisão da medição é alta 2.4.4. A quantidade de dados para cada amostra, 10, foi escolhida pois, pelo fato de os dados serem escolhidos de forma aleatória, o cálculo do desvio padrão amostral para apenas 10 dados mostrou-se próximo do valor para números maiores; isto foi percebido após a observação do desvio padrão amostral para diversas amostras distintas.

Para determinação da exatidão das medições, foram calculados, para cada medição de cada amostra: o erro absoluto, que consiste na diferença entre o valor medido e o valor de referência, em valores absolutos; e, em seguida, a exatidão percentual, por meio da equação mostrada em 2.4.4. Por fim, foi calculada a média das exatidões, para cada valor de referência. Todos os resultados podem ser vistos na tabela na Figura 4.15.

Exatidão (%) para cada amostra aleatória	99,89	99,98	99,95	100,00	99,91	99,98
	99,86	99,92	99,96	99,94	99,85	99,94
	99,69	99,99	99,91	99,87	99,87	99,89
	99,92	99,94	99,95	99,79	99,89	99,93
	99,95	99,90	99,94	99,97	99,89	99,95
	99,94	99,89	99,91	99,95	99,85	99,93
	99,95	99,96	99,97	99,95	99,87	99,94
	99,93	99,94	99,99	99,83	99,91	99,89
	99,95	99,95	99,99	99,98	99,91	99,97
	99,92	99,95	99,97	99,99	99,87	99,83
Exatidão média (%)	99,90	99,94	99,95	99,93	99,88	99,93

Figura 4.15 – Tabela que mostra a exatidão das medições, e exatidão média.

Pela tabela, é possível perceber que a exatidão média é bem alta, acima de 99%, para todos os valores de referência, o que mostra que a exatidão da medição é alta.

4.1.3.3 Simulação de decolagem com análise dos requisitos temporais

Este teste tem o objetivo de validação do requisito de execução temporal da tarefa de medição está sendo cumprido. Para o teste, foi feita uma nova simulação de uma situação simplificada de decolagem seguida de pouso. O motor encontrava-se inicialmente parado, em seguida foi enviado o comando para o motor atingir a velocidade angular de, aproximadamente, $9500RPM$, então esperou-se um tempo curto para que leitura estabilizasse, e, por último, enviou-se o comando para o motor parar.

A mesma "tarefa de visualização"foi utilizada para imprimir os dados medidos no monitor serial, mantendo sua execução como periódica a cada $10ms$. Porém, para a análise correta da periodicidade da tarefa de medição, o tempo considerado neste teste foi o tempo em microssegundos coletado no momento em que a medição foi realizada e armazenado na mesma estrutura da medição. Os dados coletados no teste podem ser vistos no gráfico da velocidade angular medida pelo tempo no qual a medição foi realizada, apresentado na Figura 4.16. O tempo foi deslocado, subtraindo-se o tempo necessário para a inicialização do microcontrolador.

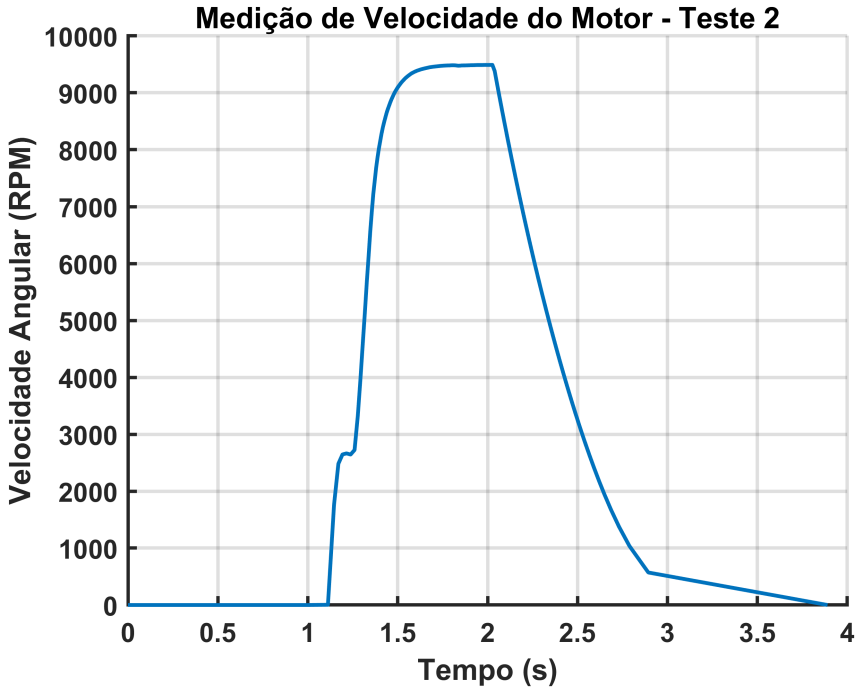


Figura 4.16 – Gráfico da medição pelo tempo no qual a medição foi realizada.

No gráfico, observa-se um resultado conforme o esperado: o motor inicialmente está parado, estabiliza no valor esperado, e retorna à velocidade zero. Então, os mesmos dados do teste foram utilizados para geração do gráfico da Figura 4.17, que mostra o período da tarefa de medição, para cada medição no tempo do teste. Os períodos foram obtidos subtraindo os tempos de execução consecutivos, dois a dois.

Em relação à elaboração do gráfico da periodicidade da medição, ocorreram alguns problemas relacionados à tarefa de visualização, e não à tarefa de medição; pois a tarefa de medição é esporádica, enquanto a tarefa de visualização é periódica, executando a cada $10ms$. Nas velocidades mais baixas do motor, a tarefa de medição fica por mais tempo no momento em espera pela notificação, ou seja, a espera pela interrupção de software que informa que o dado está pronto para o cálculo da velocidade angular. Nesse momento, a tarefa de visualização é executada diversas vezes antes de uma nova execução da tarefa de medição, que atualiza a estrutura global, assim percebeu-se que eram "geradas várias leituras repetidas", com o mesmo valor de velocidade e o mesmo tempo. Essas leituras foram então omitidas do gráfico de periodicidade, para não corromperem a medição correta dos períodos, porque elas geravam casos de período zero, que não aconteceram na realidade. Já nas velocidades mais altas do motor, ocorreu o problema contrário, a tarefa de medição se tornava mais rápida do que a tarefa de visualização, então, algumas vezes, a estrutura era atualizada sem seu valor ter sido lido pela tarefa de visualização. Isso também corrompeu a medição correta dos períodos, pois causou o aparecimento de períodos de valor duplicado no gráfico, que são incorretos pois não aconteceram na realidade. Então esses casos também

foram omitidos do gráfico da periodicidade.

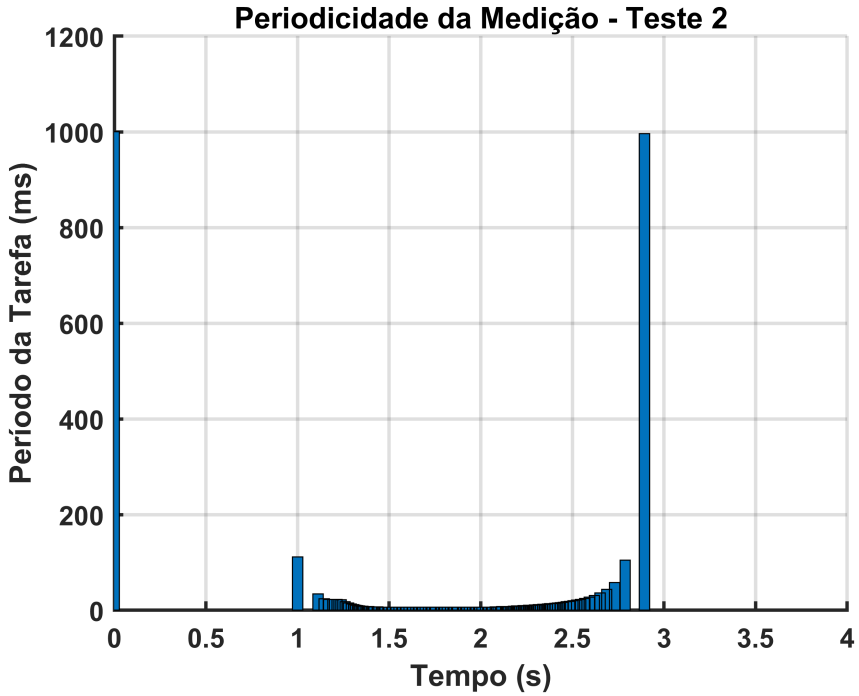


Figura 4.17 – Gráfico da periodicidade da tarefa de medição.

A Figura 4.18 mostra o mesmo gráfico da periodicidade da medição, porém ampliado, pois no gráfico anterior não era possível ver sua região central.

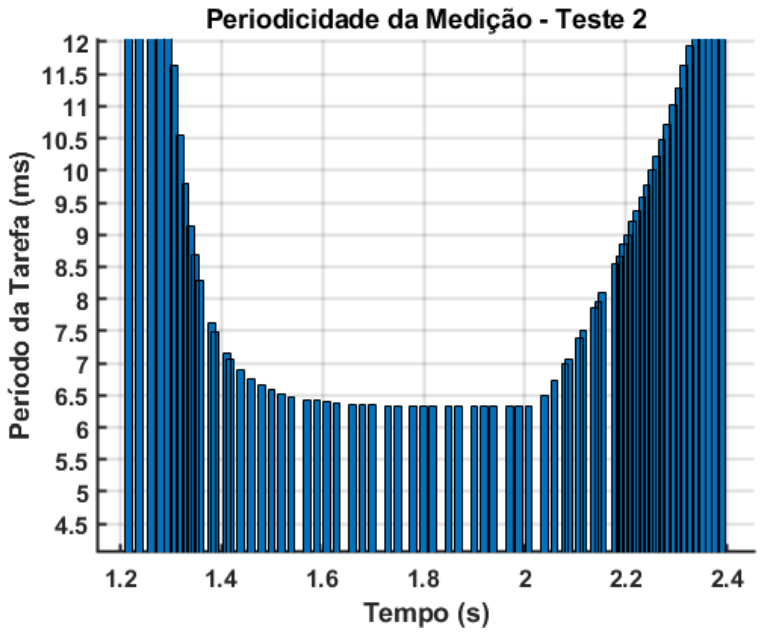


Figura 4.18 – Mesmo gráfico da periodicidade da tarefa de medição, porém ampliado.

Observando-se as três figuras, é possível fazer validações. Conforme explicação em 2.4.2.1, o período de uma volta completa do motor consiste no período de execução da tarefa de medição, logo há uma relação entre a periodicidade da tarefa e a velocidade angular medida, que pode ser vista na equação 4.1, considerando a velocidade angular em RPM e o período em segundos.

$$velocidade\_angular[RPM] = \frac{60}{periodo[s]} \tag{4.1}$$

Pela equação, para a velocidade máxima do teste, $9500RPM$, esperava-se um período de $6{,}316ms$. No gráfico da periodicidade ampliado, é possível observar que este é o valor do menor período, o que está correto. Além disso, observa-se que o período de 6,3 ms se mantém estabilizado entre os tempos 1,7 s e 2 s; observando o gráfico das medições, entre esses mesmos valores de tempo, consiste no tempo no qual a velocidade do motor está estabilizada em 9500 RPM, o que também está correto e conforme o esperado.

Esta validação pode ser feita para outros pontos, por exemplo: no gráfico da periodicidade, o período de $7{,}5ms$ ocorre no tempo de, aproximadamente, $2{,}1s$; de acordo com a equação, o valor de rotação esperado para um período de $7{,}5ms$ é $8000RPM$; no gráfico das medições, visualmente percebe-se que o valor da rotação em $2{,}1s$ é aproximadamente $8000RPM$, o que está conforme o esperado.

Por fim, observa-se no gráfico da periodicidade que não está ampliado, o valor de período máximo de $1000ms$. Este valor foi assim definido no software para limitar o tempo de espera da tarefa de medição; assim, passados $1000ms$, se nenhuma notificação foi recebida, a tarefa considera que o motor está parado. Como este valor é visto no gráfico como o período máximo, o resultado do teste foi conforme o esperado.

Na seção 3.1.3, a execução da tarefa de medição foi definida como esporádica, de período variável entre $6{,}3ms$ e $1000ms$. Dessa forma, o requisito temporal da tarefa de medição de velocidade angular do motor foi cumprido.

4.1.3.4 Integração da medição com as filas circulares e a comunicação TCP/IP

Por fim, este teste mostra a integração da tarefa de medição com a tarefa que coloca seus dados na fila circular, e com a tarefa que esvazia a fila circular, enviando os dados em uma mensagem JSON por TCP/IP. Para o teste, foi utilizada a ferramenta "$Netcat$", que permite a inicialização de um cliente TCP, pelo comando no terminal: "nc 192.168.4.1 3333"; com o endereço IP do servidor, e a porta na qual o servidor está escutando conexões.

O resultado do teste de comunicação TCP/IP pode ser visto na Figura 4.19. Quando ocorre a conexão, inicia-se a comunicação, então o cliente envia o comando "get_left_ir", e o servidor responde com uma mensagem JSON contendo as medições das velocidades angulares que estavam armazenadas na fila circular, assim como o tempo no qual cada medição foi realizada.

```
I (57312) communication: Received 43 bytes: { "command": "get_left_ir", "id": 234624 }

I (57322) communication: Response sent: {
        "response":     "get_left_ir",
        "id":   234624,
        "angular_speed":        [983.426443918656, 987.638815462479, 989.3222023586186, 991.038691984162, 994.65096936106067],
        "fotosensor_time":      [52047656, 53046857, 54044419, 55045412, 56043527]
}
```

Figura 4.19 – Mensagens entre o servidor e o cliente TCP/IP, para o comando que solicita os dados de medições de velocidade angular do motor esquerdo.

4.1.4 Medições da IMU

A unidade de medição inercial, conforme explicação em 2.4.3, consiste em um dispositivo comercial e robusto, e cujo software embarcado foi desenvolvido conforme sua documentação. Além disso, não havia o acesso a equipamentos que realizassem medições de referência para validação dos sensores da IMU. Por esses motivos, a validação das suas medições não foi feita da mesma forma que a dos outros sensores.

Para a realização das medições da IMU, primeiramente é feita a inicialização do protocolo de comunicação serial *"I2C"*, como pode ser vista na Figura 4.20.

```
I (313) main_task: Started on CPU0
I (323) main_task: Calling app_main()
I (323) i2c-imu: I2C initialized successfully
I (323) i2c-imu: WHO_AM_I = 71
I (323) i2c-imu: Accel: X=0.00000 m/s², Y=0.00000 m/s², Z=0.00000 m/s²
I (333) i2c-imu: Gyro: X=0.00000 rad/s, Y=0.00000 rad/s, Z=0.00000 rad/s
I (343) i2c-imu: Mag: X=0.00000 µT, Y=0.00000 µT, Z=0.00000 µT
```

Figura 4.20 – Monitor serial do microcontrolador mostrando a inicialização do protocolo de comunicação serial *"I2C"*.

O resultado de um teste de medições do acelerômetro, giroscópio e magnetômetro pode ser visto na Figura 4.21. O teste simula uma situação simplificada de voo, na qual o robô aéreo estava se movimentando.

```
I (152824) i2c-imu: Accel: X=0.52822 m/s², Y=0.45891 m/s², Z=8.79809 m/s²
I (152824) i2c-imu: Gyro: X=0.09721 rad/s, Y=-0.13409 rad/s, Z=-0.52331 rad/s
I (152824) i2c-imu: Mag: X=-11.54530 µT, Y=18.14261 µT, Z=-29.83785 µT

I (153334) i2c-imu: Accel: X=-0.03346 m/s², Y=0.48520 m/s², Z=8.98691 m/s²
I (153334) i2c-imu: Gyro: X=-0.06578 rad/s, Y=0.34714 rad/s, Z=0.58230 rad/s
I (153334) i2c-imu: Mag: X=-11.54530 µT, Y=17.54286 µT, Z=-29.53797 µT

I (153844) i2c-imu: Accel: X=-0.99669 m/s², Y=1.59661 m/s², Z=9.50557 m/s²
I (153844) i2c-imu: Gyro: X=0.39255 rad/s, Y=0.29388 rad/s, Z=0.18589 rad/s
I (153844) i2c-imu: Mag: X=-15.29377 µT, Y=13.19463 µT, Z=-32.38681 µT

I (154354) i2c-imu: Accel: X=-0.71943 m/s², Y=2.58374 m/s², Z=9.04906 m/s²
I (154354) i2c-imu: Gyro: X=-0.06005 rad/s, Y=-0.52957 rad/s, Z=-0.54661 rad/s
I (154354) i2c-imu: Mag: X=-10.34579 µT, Y=18.14261 µT, Z=-33.43639 µT

I (154864) i2c-imu: Accel: X=1.62051 m/s², Y=2.37341 m/s², Z=8.46108 m/s²
I (154864) i2c-imu: Gyro: X=0.27790 rad/s, Y=-0.78257 rad/s, Z=-0.81719 rad/s
I (154864) i2c-imu: Mag: X=-4.94799 µT, Y=20.54164 µT, Z=-28.93822 µT

I (155374) i2c-imu: Accel: X=2.15351 m/s², Y=1.58466 m/s², Z=6.42708 m/s²
I (155374) i2c-imu: Gyro: X=-0.93623 rad/s, Y=0.26019 rad/s, Z=0.19414 rad/s
I (155374) i2c-imu: Mag: X=-4.94799 µT, Y=20.24176 µT, Z=-29.23810 µT

I (155884) i2c-imu: Accel: X=-2.40209 m/s², Y=-1.05166 m/s², Z=8.43718 m/s²
I (155884) i2c-imu: Gyro: X=-0.36685 rad/s, Y=-0.16432 rad/s, Z=0.62891 rad/s
I (155884) i2c-imu: Mag: X=-10.34579 µT, Y=17.54286 µT, Z=-33.13651 µT

I (156394) i2c-imu: Accel: X=-3.15020 m/s², Y=-0.08604 m/s², Z=8.23402 m/s²
I (156394) i2c-imu: Gyro: X=0.04953 rad/s, Y=-0.29534 rad/s, Z=0.50906 rad/s
I (156394) i2c-imu: Mag: X=-12.89475 µT, Y=15.29377 µT, Z=-33.28645 µT
```

Figura 4.21 – Medições da IMU para o robô aéreo se movimentando.

O resultado de um segundo teste de medições do acelerômetro, giroscópio e magnetômetro pode ser visto na Figura 4.22. Neste teste, o robô aéreo encontrava-se em uma posição fixa e imóvel. O tempo mostrado no monitor serial é o tempo em milissegundos da execução da "tarefa de visualização", criada da mesma forma que as tarefas de visualização dos demais sensores, e com execução periódica a cada 20ms. O período de execução da tarefa de visualização não valida a periodicidade da tarefa de medição, cujo requisito temporal é 10ms.

```
I (3523) i2c-imu: Accel: X=0.30833 m/s², Y=0.30355 m/s², Z=9.17573 m/s²
I (3523) i2c-imu: Gyro: X=0.02463 rad/s, Y=-0.02503 rad/s, Z=0.00759 rad/s
I (3523) i2c-imu: Mag: X=0.00000 µT, Y=-18.59243 µT, Z=-28.78828 µT

I (3543) i2c-imu: Accel: X=0.30116 m/s², Y=0.26770 m/s², Z=9.18052 m/s²
I (3543) i2c-imu: Gyro: X=0.02716 rad/s, Y=-0.02463 rad/s, Z=0.00626 rad/s
I (3543) i2c-imu: Mag: X=-0.29988 µT, Y=-18.59243 µT, Z=-30.58755 µT

I (3563) i2c-imu: Accel: X=0.35135 m/s², Y=0.27965 m/s², Z=9.14944 m/s²
I (3563) i2c-imu: Gyro: X=0.02823 rad/s, Y=-0.02543 rad/s, Z=0.00772 rad/s
I (3573) i2c-imu: Mag: X=1.94921 µT, Y=-19.64200 µT, Z=-28.93822 µT

I (3583) i2c-imu: Accel: X=0.31789 m/s², Y=0.27248 m/s², Z=9.16856 m/s²
I (3583) i2c-imu: Gyro: X=0.02863 rad/s, Y=-0.02011 rad/s, Z=0.01132 rad/s
I (3593) i2c-imu: Mag: X=-0.44982 µT, Y=-20.24176 µT, Z=-30.43761 µT
```

Figura 4.22 – Medições da IMU para o robô aéreo em uma posição fixa e imóvel.

O eixo Z do acelerômetro consiste no eixo que monitora a aceleração vertical do robô aéreo. Para o primeiro teste, com o robô aéreo em movimento, percebe-se que o valor varia entre 6,4 e 9,5m/s^2, sua variação era esperada. Para o segundo teste, com o robô aéreo imóvel, esperava-se um valor com pouca variação, e em torno do valor da aceleração da gravidade, e isto pode ser observado na Figura 4.22.

4.1.5 Considerações Adicionais

Os testes documentados que integram as filas circulares com a comunicação TCP/IP mostraram que ao se utilizar uma fila circular de tamanho 100, o limite de alocação de memória do "$ESP32$"é atingido. Os testes, então, utilizaram filas de tamanho máximo igual a 10. Para que seja possível o aumento do tamanho das filas, é necessário um maior estudo relacionado à memória do microcontrolador, para que esta seja utilizada com maior otimização.

Os servo motores, assim como a placa de alimentação, foram pouco testados até esta etapa do projeto. Os testes do software embarcado, por terem tido como maior foco a validação das medições utilizando o monitor serial, não incluíram os servo motores por não haver sensores relacionados a eles. Além disso, não é recomendado alimentar o "$ESP32$"por duas fontes ao mesmo tempo, então adicionou-se um botão gangorra na placa que é utilizado para, durante o desenvolvimento, selecionar se a alimentação do microcontrolador, e também da placa que contém o microcontrolador, é realizada pela porta USB, por onde é possível a utilização do monitor serial, ou pelo regulador 7805, que permite a conexão da placa de alimentação. Por isso, os testes documentados alimentaram a placa que contém o microcontrolador pela porta USB, e não por meio da placa de alimentação; com a alimentação do motor sendo feita pela bateria conectada diretamente no ESC, e também por uma fonte de bancada de 12V para substituição da bateria. Foi observado um aumento de volume na

bateria, e ela demonstra estar descarregando mais rapidamente, o que não foi um problema até esta etapa do projeto, porém deve ser feita a compra de uma nova bateria para que o robô aéreo possa realizar voos com segurança.

4.2 Testes do Sistema de Aquisição de Dados

Para os testes do sistema de dados, foi feito um servidor de simulação para o ESP-32, que criava a porta 5000 de conexão no servidor local, ficando no aguardo de conexão de um cliente. Assim, a partir da tarefa de comunicação na aplicação em *Python*, foi criado um cliente para conectar ao servidor através de um soquete. Assim, com a comunicação estabelecida, a troca de mensagens no formato JSON foi feita, com a requisição do lado do cliente sendo feita ao servidor e este, por sua vez, enviava dados para serem mostrados em gráficos pela aplicação do cliente. Assim, o sistema de dados tem as seguintes funcionalidades principais:

- Conexão TCP/IP: por meio de um soquete, comunica-se com o servidor para recebimento de informações;
- Opera com diferentes tarefas: processamento paralelo de conexão, representação gráfica em tempo real de recebimento de dados e comandos pela interação com o cliente;
- Gravação de dados em arquivos para cada sensor com objetivo de posterior análise e geração de gráficos de acordo com períodos desejados, funcionando como uma "memória de voos";
- Envio de dados pela comunicação para o servidor, ajustando valores de parâmetros em tempo real no microcontrolador, sem a necessidade de alterações diretas no código;
- Permite a visualização de várias janelas de gráficos ao mesmo tempo;
- Gravação em arquivos de registro das mensagens e horário que foram enviadas.

Figura 4.23 – Menu do Sistema de Dados

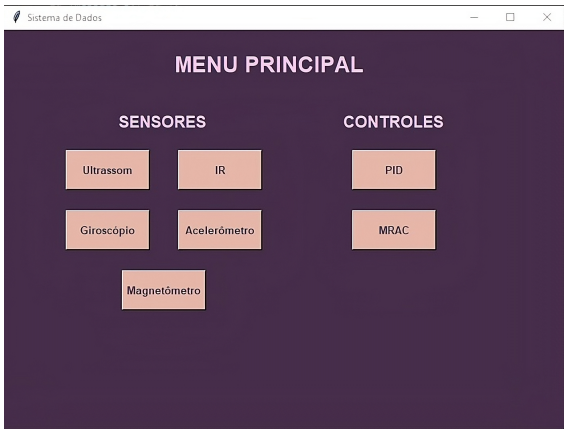


Figura 4.24 – Janela de visualização de gráficos

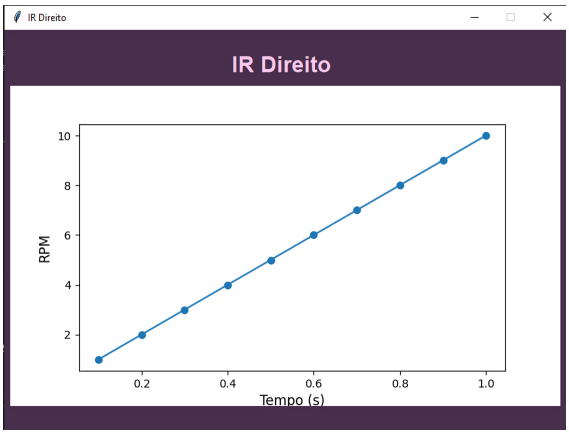
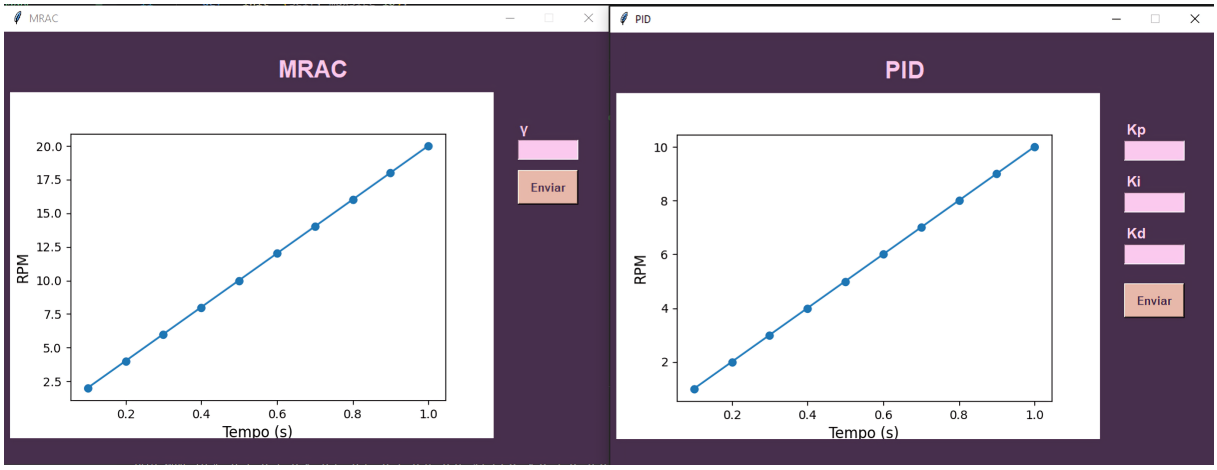


Figura 4.25 – Janela dos controles MRAC e PID da aplicação de Coleta de Dados



Na figura 4.23 é mostrada a tela inicial do sistema para interação com o usuário, que conta com os botões para a escolha dos gráficos desejados para visualização. Concomitantemente, ao iniciar o sistema, é aberta a comunicação com o servidor, recebendo os dados dos sensores e gravando-os em arquivos para posterior análise dos mesmos. A interação do usuário com a interface a partir da escolha dos botões inicia duas novas tarefas: abertura das janelas requisitadas e representação gráfica dos dados que estão nas filas recebidos da conexão TCP/IP, mostrado na figura 4.24. Quando uma janela é fechada, essas tarefas são finalizadas ao mesmo tempo, e ainda que a conexão seja encerrada por falhas de comunicação, por exemplo, os dados mostrados são os últimos recebidos, operando de maneira similar a osciloscópios. Para fins de simulação, foi utilizado um tempo de 2 segundos entre cada requisição do cliente para o servidor. Por fim, a figura 4.25 mostra a tela dos controles funcionando concorrentemente, sendo que o gráfico mostra a leitura dos sensores de velocidade em rpm que os motores estão girando. E além do gráfico representante do que está sendo recebido da planta, há também o campo de escrita de valores para enviar ao microcontrolador e ajustar os parâmetros dos controles.

4.3 Simulações e Análises das Técnicas de Controle

Para essa seção, é importante destacar que todas as simulações foram feitas nos *softwares* Matlab R2021a e Simulink, da *MathWorks* utilizando o método de integração *ode23t* para solução das equações diferenciais, adequado para resolver equações diferenciais ordinárias em problemas com moderada rigidez, não precisando de passos de tempo extremamente pequenos para manter a solução estável, além de ser usado o passo variável com 1×10^{-3} de tolerância relativa.

Ainda, para análise temporal do sinais dessa sessão foram utilizados os critérios discutidos na seção 2.6.1. Para o tempo de acomodação (t_s), o tempo considerado para que a curva de resposta alcançasse valores na faixa de 2% em torno do valor final foi considerado, devido

ao fato de que é necessária uma alta precisão para sistemas de controle de velocidade. Para o tempo de subida (t_r), quando o sinal tem características de subamortecimento, considerou-se medir o tempo que era necessário para este de ir de 0% a 100% do valor final; quando o sistema apresentava características de superamortecimento, então foi considerado o critério de tempo de alcance dos valores entre 10% a 90% do valor final. A constante de tempo para sistemas de primeira ordem foi medida a fim de comparação entre a simulação e os valores inseridos na planta, sendo calculado o instante de tempo necessário para o sinal atingir 63% do seu valor final. Já para sistemas subamortecidos, o sobressinal foi calculado a partir do valor máximo de pico da curva de resposta ($c_{(t_p)}$) e o valor em regime permanente ($c_{(\infty)}$), utilizando a porcentagem para expressá-lo indicando diretamente a estabilidade relativa do sistema sendo representado pela equação (4.2). Todos esses parâmetros foram medidos na escala de milissegundos (ms) por adequação melhor da precisão e representação dos dados para as devidas análises e comparações de desempenho.

$$M_p = \frac{c_{(t_p)} - c_{(\infty)}}{c_{(\infty)}} \times 100\% \tag{4.2}$$

4.3.1 Motor de Corrente Contínua

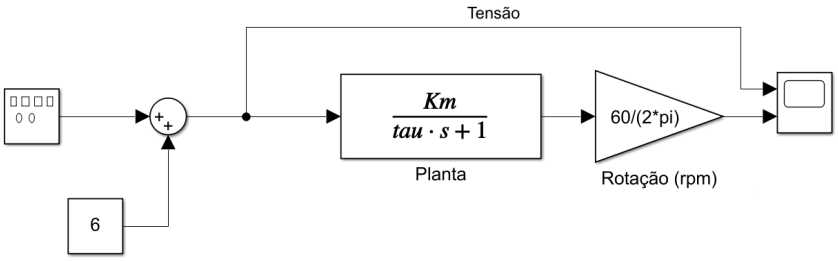
A simulação do modelo da planta utilizada nesse trabalho para validação da implementação das técnicas de controle é representada pela equação (3.14) e utiliza os parâmetros do motor CC Maxon F2130.906 que conta com uma tensão nominal de 12 V, como especificado em (Maxon Motor AG, 2013). A partir desse documento, os dados utilizados para as simulações são discriminados na tabela tabela 4.1 sendo colocados em código no Matlab e construído o diagrama de blocos para a planta. Nele, a entrada da simulação da planta do motor foi feita utilizando o bloco "gerador de sinais", configurado para fornecer uma onda quadrada simulando as variações de tensão de 0 V a 12 V, para que não fossem testadas tensões negativas nem que pudessem queimar o motor, com uma frequência de 0,5 Hz, sendo que esse valor foi inserido na simulação devido ao fato de que este é um controle que deve ser implementado em cascata com os controles de atitude e altitude desenvolvidos por (Botelho, 2020), logo, deve-se apresentar uma frequência menor que as desses controles, para primeiro ocorrer suas devidas atualizações e, por fim, atualizar o controle dos motores. Assim, o diagrama de blocos pode ser visto na figura 4.26.

Tabela 4.1 – Parâmetros do Motor de corrente contínua

Parâmetros	Valores
$J\ [Kg/m^2]$	3.75×10^{-7}
$B\ [Kg \cdot m^2/s]$	8.4×10^{-6}
$K_t\ [Nm/A]$	0.0172
$K_v\ [Vs/rad]$	0.0172
$L_a\ [H]$	1.71×10^{-3}
$R_a\ [\Omega]$	35.2
ρ	1

Fonte: (Maxon Motor AG, 2013)

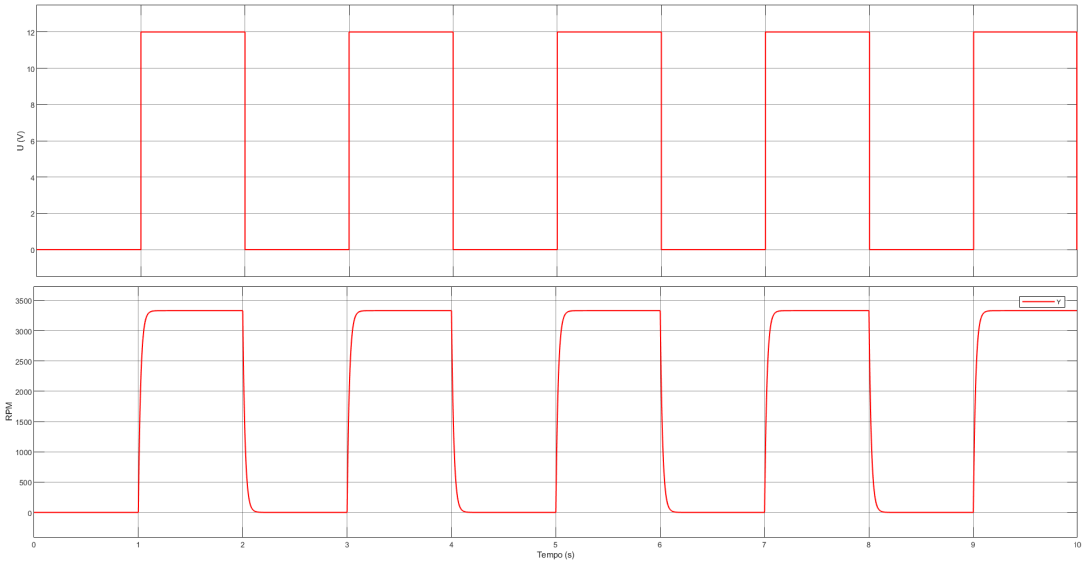
Figura 4.26 – Diagrama no Simulink do motor de corrente contínua



Após 10 segundos de simulação, é obtido o sinal como mostra a figura 4.27, com as rotações na saída variando de 0 rpm a 3.332rpm, sendo possível observar as seguintes características:

- constante de tempo: $\tau \approx 22{,}0\ ms$
- tempo de subida: $t_r \approx 41{,}0\ ms$
- tempo de acomodação: $t_s \approx 68{,}3\ ms$
- sobressinal: $M_p = 0\%$

Figura 4.27 – Simulação da planta do motor de corrente contínua



Porém, os sinais de comando para os motores serão uma referência de rotação, em rpm, resultado dos controles de atitude e altitude para a devida estabilização, desenvolvidos no trabalho de (Botelho, 2020). Com isso, o sistema deve estar preparado para receber valores de referência de rotação, sendo necessário aplicar técnicas de controle para que a entrada de referência seja atingida pela saída da planta. Assim, o sinal gerado pelo bloco "gerador de sinais" foi adaptado para a simulação dos controladores com o objetivo de representar uma onda quadrada que deve variar tanto positiva quanto negativamente a uma amplitude de 500 rpm com a mesma frequência de simulação da planta do motor de 0.5 Hz com um bloco somador de um valor constante de 2.000 rpm, para ajuste do valor médio do sinal, fazendo-o variar de 1.500 rpm a 2.500 rpm. simulando valores de rotação máximos e mínimos para levantar a plataforma.

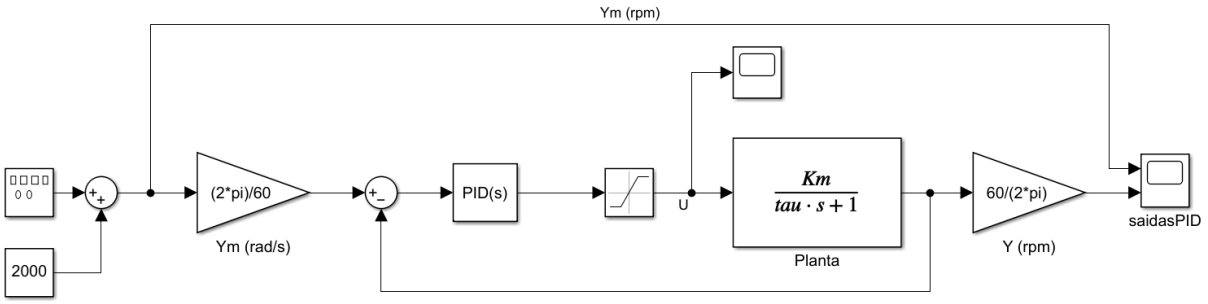
Essa faixa de valores foi definida para o motor CC modelado, que conta com uma rotação nominal de 2.780 rpm, como especificado em (Maxon Motor AG, 2013), porém para a implementação prática com a dinâmica do motor descrito pela figura 4.16, na qual a velocidade mínima para o motor conseguir levantar a plataforma é de 7.500 rpm e a velocidade em regime permanente é de 9.500 rpm será necessário fazer ajustes no ganho da planta aqui modelada, já que este parâmetro depende de características físicas, que variam para cada motor. É garantido que os controles irão atuar da mesma maneira para o sistema real com esse ajuste devido às suas características lineares, assim, as simulações foram feitas para fins de validação dos mesmos.

4.3.2 Controle PID

Na seção 3.3.2, foi desenvolvido o controlador na forma ZPK (ganhos-polos-zeros) tendo como resultado os valores mostrados na figura 3.13 com os valores dos ganhos K_p,

K_i e K_d calculados de acordo com as equações em 3.18. Assim, esses valores são mostrados na coluna "Valores Calculados" da tabela 4.2 e o bloco "PID" foi ajustado de acordo com eles, em série com a planta. O diagrama de blocos mostrado na figura 4.28 foi simulado, mostrando conversões de rpm para rad/s com os blocos de ganho, como também há um bloco de saturação assegurando que o valor de tensão do sinal de controle para o motor não ultrapasse sua tensão nominal de 12 V, podendo queimar o motor.

Figura 4.28 – Diagrama de blocos do PID



Porém, após as primeiras simulações, percebeu-se que o sinal de controle estava oferecendo pulsos muito altos de resposta ao degrau, devido ao alto valor do termo derivativo em relação aos valores do termo integrativo e proporcional, resultando em uma influência notável no sinal de controle nas transições do degrau. Então, foram necessários ajustes finos dos valores dos coeficientes para se obter um sinal de controle menos abrupto em relação às transições, o que na prática representaria um desgaste mecânico excessivo a cada pulso. Assim, os valores atualizados dos coeficientes após o ajuste são mostrados na coluna "Valores Ajustados" da tabela 4.2.

Tabela 4.2 – Coeficientes do controle PID

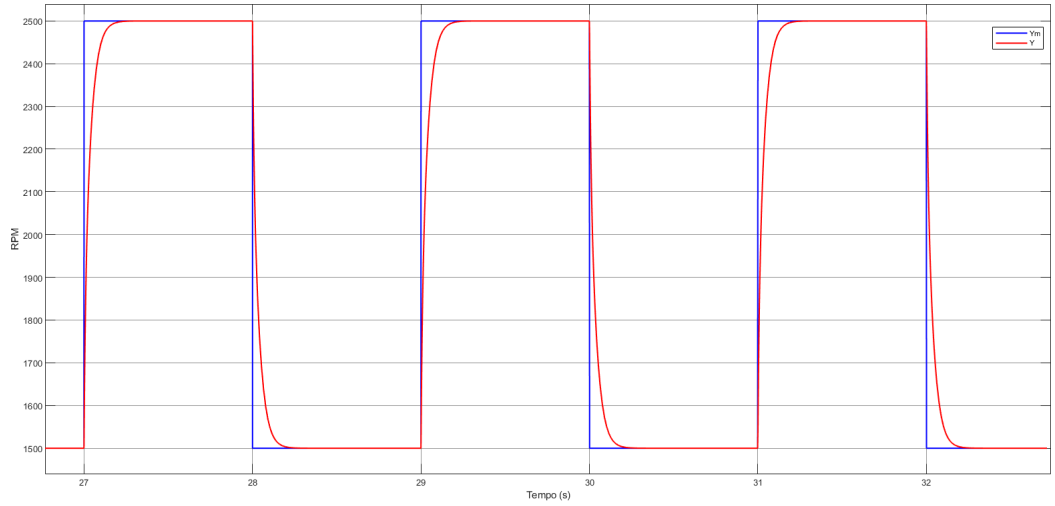
Coeficientes	Valores Calculados	Valores Ajustados
K_p	0.013	0.03
K_i	0.736	1.114
K_d	0.0001	0.0001

A simulação desse controle sem a influência de fatores externos no ponto de operação para o qual o controle foi projetado é como mostra a figura 4.29. O tempo total de simulação foi de 300 segundos, ou 5 minutos e as características de desempenho desse sistema são listadas.

- constante de tempo: $\tau \approx 29\ ms$
- tempo de subida: $t_r \approx 75{,}5\ ms$
- tempo de acomodação: $t_s \approx 102{,}7\ ms$

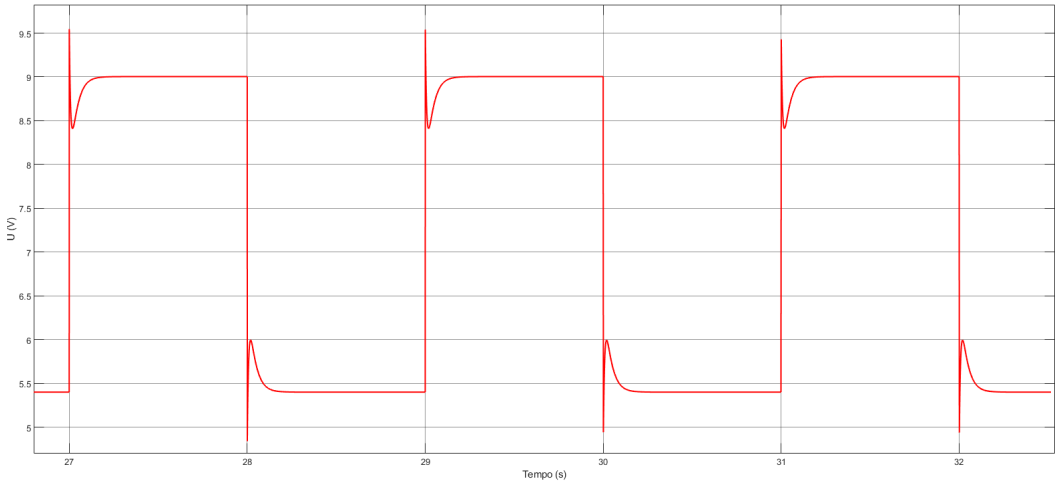
- sobressinal: $M_p = 0\%$

Figura 4.29 – Resposta simulada do motor com controle PID



Para a análise quanto ao sinal de controle, tem-se o resultado mostrado na figura 4.30. Os maiores picos do sinal ocorrem com aproximadamente 9,6 V de amplitude e os mínimos ocorrem com 4,8 V, variando a tensão de entrada em regime permanente de 5,4 V a 9V, tendo esse sinal uma média de 7,3 V. Assim, é possível perceber que o sinal não está saturando na tensão máxima de 12 V, conseguindo manter o controle da planta seguindo as especificações do sistema.

Figura 4.30 – Sinal de controle PID

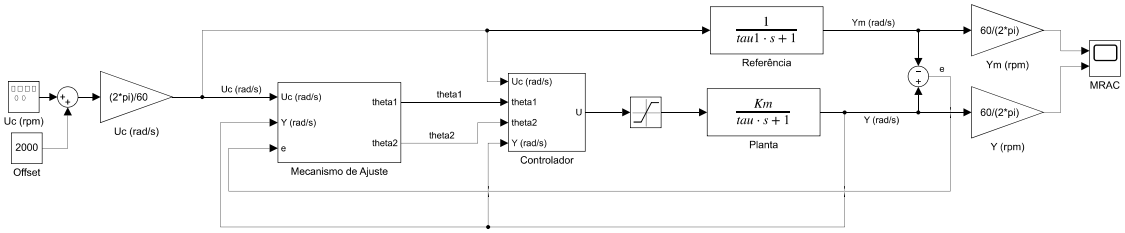


4.3.3 Controle MRAC

A partir dos cálculos desenvolvidos na seção 3.3.3, o diagrama de blocos do controle MRAC é como mostra a figura 4.31. O bloco de ganho $U_c(rad/s)$ serve para fazer a conversão de unidades da entrada de rpm para rad/s, já que a função de transferência do motor está nas

unidades do SI e a entrada de referência será recebida em rpm, como discutido previamente. Ademais, o bloco de saturação foi implementado por segurança, para que o sinal de controle U não ultrapasse o nível de tensão de 12 V, podendo queimar o motor. Por fim, depois do cálculo do erro entre os sinais de saída da planta e da referência na unidade de rad/s, seus valores são convertidos novamente para seus respectivos valores em rpm e suas curvas são mostradas pelo bloco *Scope*.

Figura 4.31 – Diagrama de blocos do MRAC



O bloco "Controlador" é a representação em simulação da equação (3.22), sendo mostrada na figura 4.32. E o bloco "Mecanismo de Ajuste" representa a equação (3.30), sendo mostrado na figura 4.33.

Figura 4.32 – Bloco "Controlador"

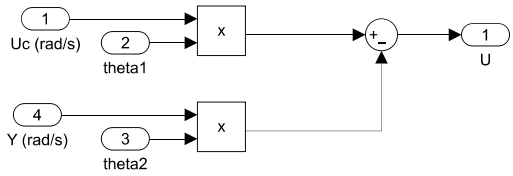
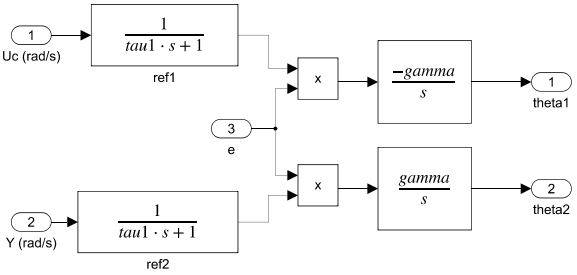


Figura 4.33 – Bloco "Mecanismo de ajuste"



Por fim, para a determinação do valor da taxa de adaptação foi utilizado o método de testes de ajuste demonstrado por (Swarnkar; Jain; Nema, 2011). Nesse contexto, é discutido sobre a faixa de possíveis valores na qual esse parâmetro deve estar contido visando a rápida estabilidade do sistema com baixas oscilações. Se o valor de γ estiver além do máximo, o sistema pode ser levado à instabilidade, caso esteja inferior ao mínimo, ele pode não conseguir se adaptar. Para o sistema desse trabalho, seu valor foi definido para ser $\gamma = 1{\times}10^{-5}$, atendendo aos requisitos necessários citados.

A simulação desse controle sem a influência de fatores externos é como mostra a figura 4.34, com os sinais da entrada de controle U_c em azul, da saída de referência Y_m em roxo e da saída da planta Y em vermelho, com o tempo total de simulação de 300 segundos, ou 5 minutos. As características de desempenho em alguns instantes de tempo nesse intervalo são listadas na tabela 4.3. Ainda, foi levado em consideração que para sobressinais menores que 1% a resposta do sistema era considerada como sendo de primeira ordem, sendo calculado

seu tempo de subida com os instantes em que a resposta leva para atingir de 10% a 90% do seu valor final. Por fim, pela simulação, é possível perceber que o maior sobressinal se apresenta nos instantes iniciais de tempo, e o mesmo se torna aproximadamente nulo a partir do instante de simulação 100 segundos, como também é possível analisar que o erro entre os sinais desejado e real tende a diminuição ao longo do tempo, como mostra a coluna "Erro (rpm)".

Figura 4.34 – Resposta simulada do motor controlado por controle MRAC

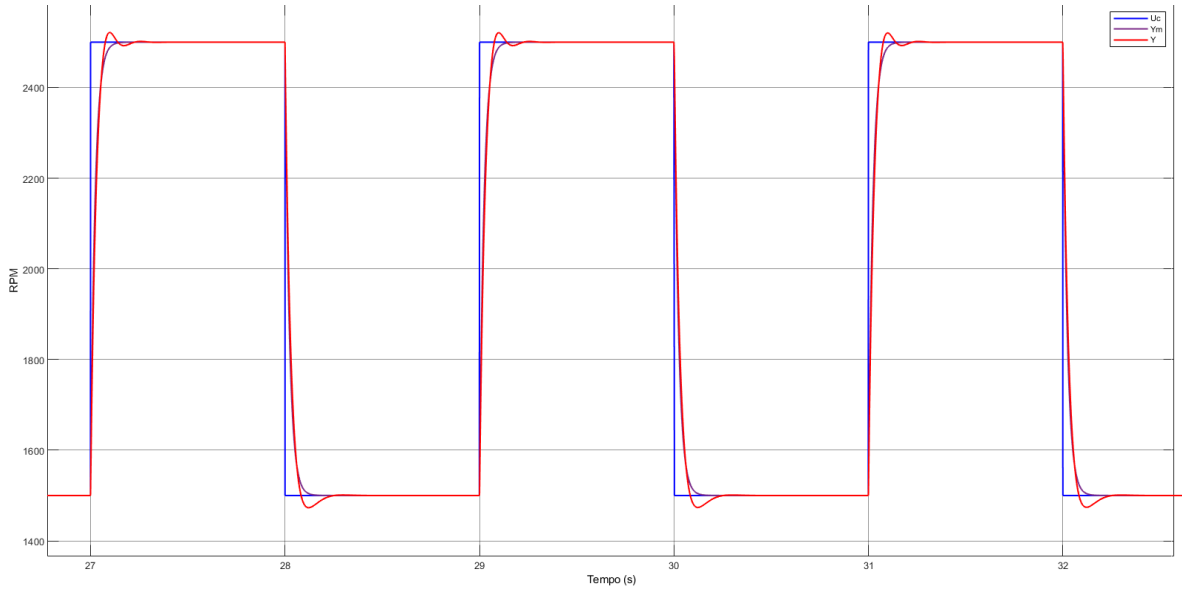
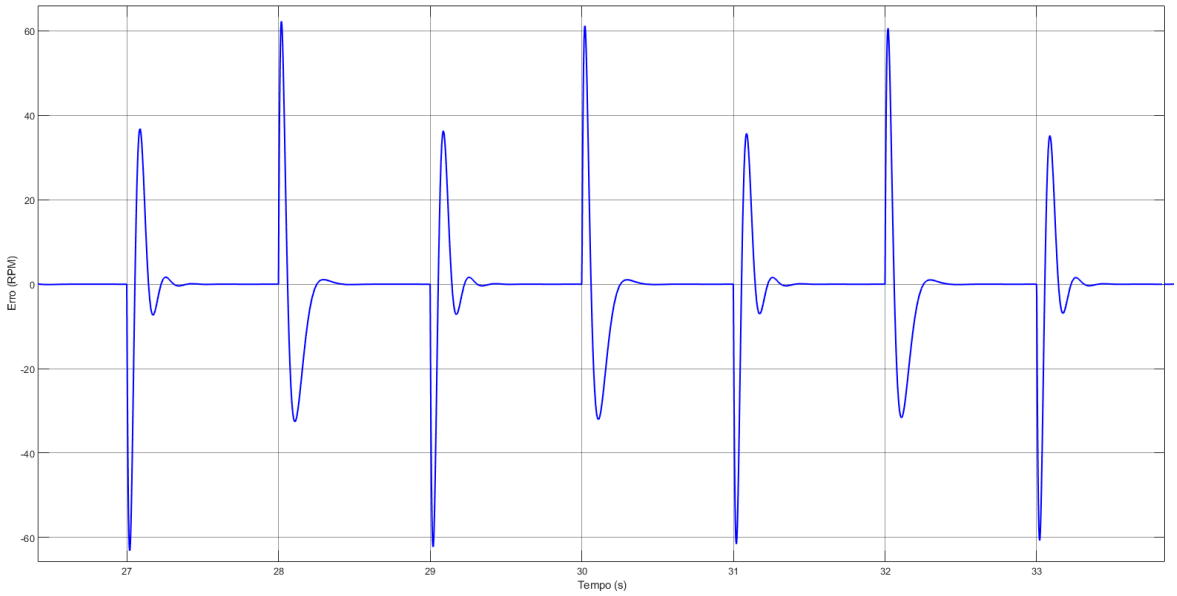


Tabela 4.3 – Características da resposta ao MRAC para diferentes instantes de tempo

Tempo (s)	Tempo de subida (ms)	Tempo de acomodação (ms)	Sobressinal (%)	Erro (rpm)
1	73,3	60,6	1,2	75,3
25	49,5	61,5	0,9	63,8
85	49,3	62,9	0,4	44,0
125	49,3	63,6	0,2	34,8
175	49,2	66,2	0	26,5
250	49,8	67,6	0	17,5

Na figura 4.35 é mostrado como o erro calculado pela equação (3.25) se comporta, sendo possível notar que esse sinal reduz o valor de picos a cada iteração, bem como seu tempo de acomodação. Ainda, é possível notar pela simulação que, o maior valor de diferença entre as saídas é de 74,2 rpm no instante de simulação de 2 s, levando aproximadamente 400 ms para se anulado. A tabela 4.3 mostra em sua última coluna valores do erro, em módulo, entre os sinais de referência e reais em rpm para alguns instantes de tempo.

Figura 4.35 – Erro entre as saídas do MRAC



Por fim, a figura 4.36 mostra as adaptações dos parâmetros ao longo do tempo sendo possível perceber que esses valores têm tendência à convergir para um determinado resultado. Ao final dos 300 segundos, com a análise temporal completa, tem-se uma estabilidade com pequenas variações, resultantes da adaptação ao processo. A tabela 4.4 destaca os valores para os quais esses parâmetros convergiram.

Figura 4.36 – Parâmetros adaptativos do MRAC

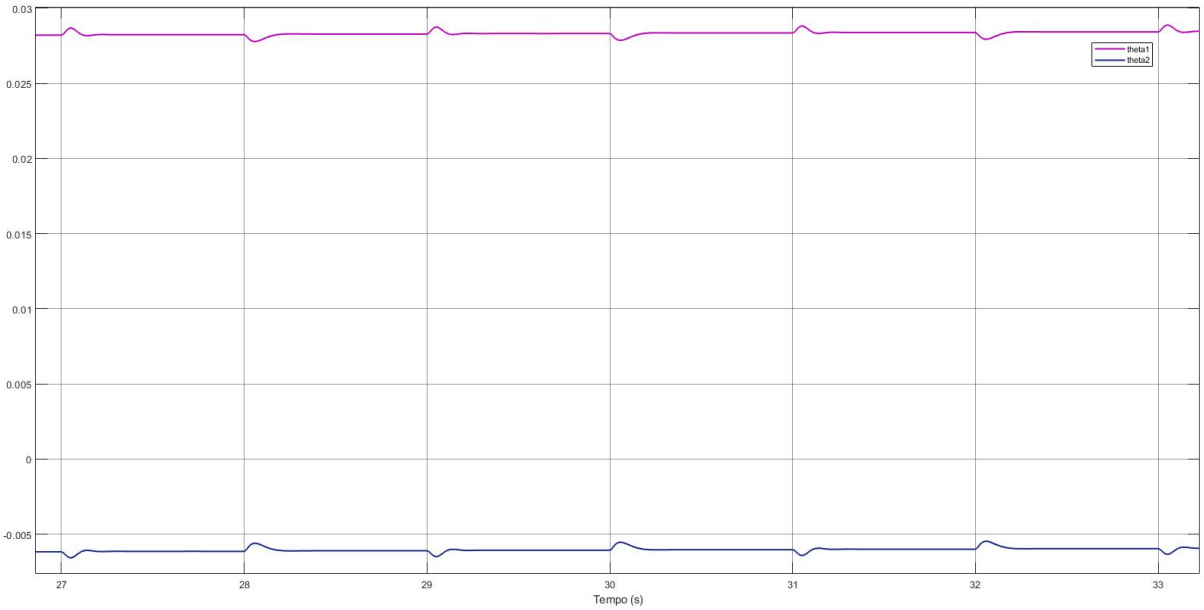


Tabela 4.4 – Parâmetros do controlador MRAC

Parâmetros	Valores
θ_1	3.38×10^{-2}
θ_2	-5.68×10^{-4}

4.3.4 Comparação entre o controle PID e o MRAC

Pode-se fazer uma análise comportamental entre os dois controles para observar como ocorre a adaptação com mudanças no processo a ser controlado. É possível atingir esse objetivo ao simular a variação na planta alterando sua resposta temporal, ou seja, alterando-se o parâmetro τ, a constante de tempo do sistema. Dois cenários são possíveis para essa avaliação: aumentar o valor dessa constate, representando um sistema mais lento, ou se obter um sistema mais rápido, com a redução desse valor (Ogata, 2010). Assim, foram simuladas as respostas dos dois controles para diferentes valores de τ durante o intervalo de tempo escolhido de aproximadamente 207 s a 210 s para fins de comparação dos comportamentos.

Assim, valor de $\tau = 0.02\ s$ característico do motor foi alterado por uma escala de 5 vezes, representando a mudança da constante de tempo da planta na equação (3.14). A tabela 4.5 mostra os novos valores inseridos para comparação dessas mudanças:

Tabela 4.5 – Alterações da constante de tempo τ

τ	Valor (s)	Característica da resposta
τ_1	0,004	Rápida
τ_2	0,1	Devagar

O comportamento do controle PID com a influência das novas constantes de tempo, sem alteração dos parâmetros K_p, K_i e K_d, é mostrado na figura 4.37, com as características de desempenho sendo listadas na tabela 4.6.

Figura 4.37 – Simulação do controle PID com mudanças da planta

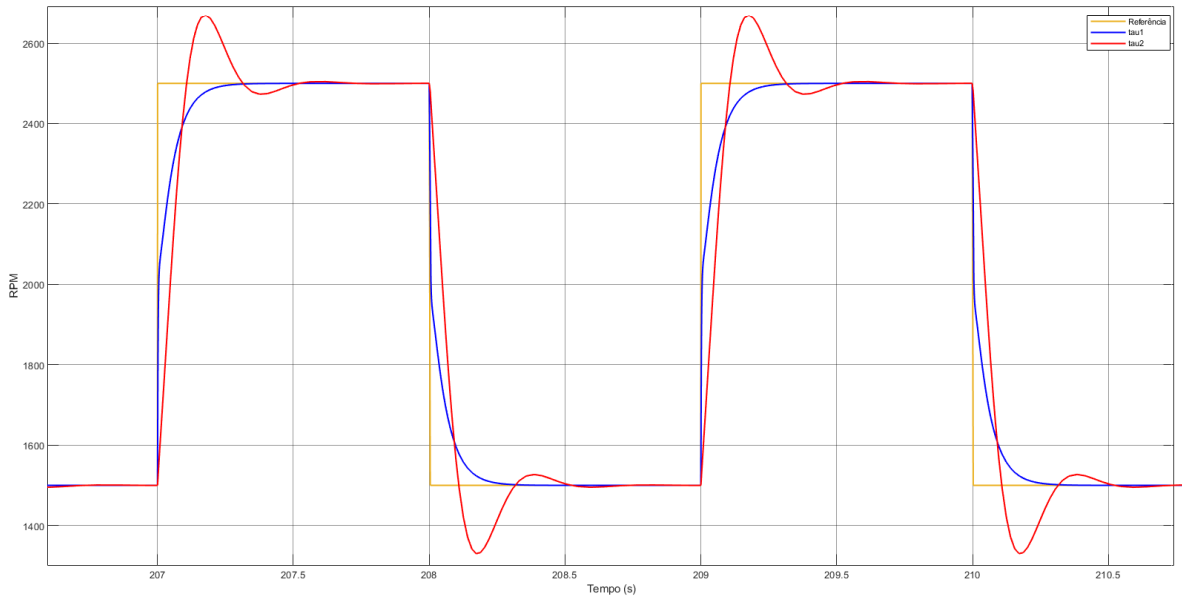
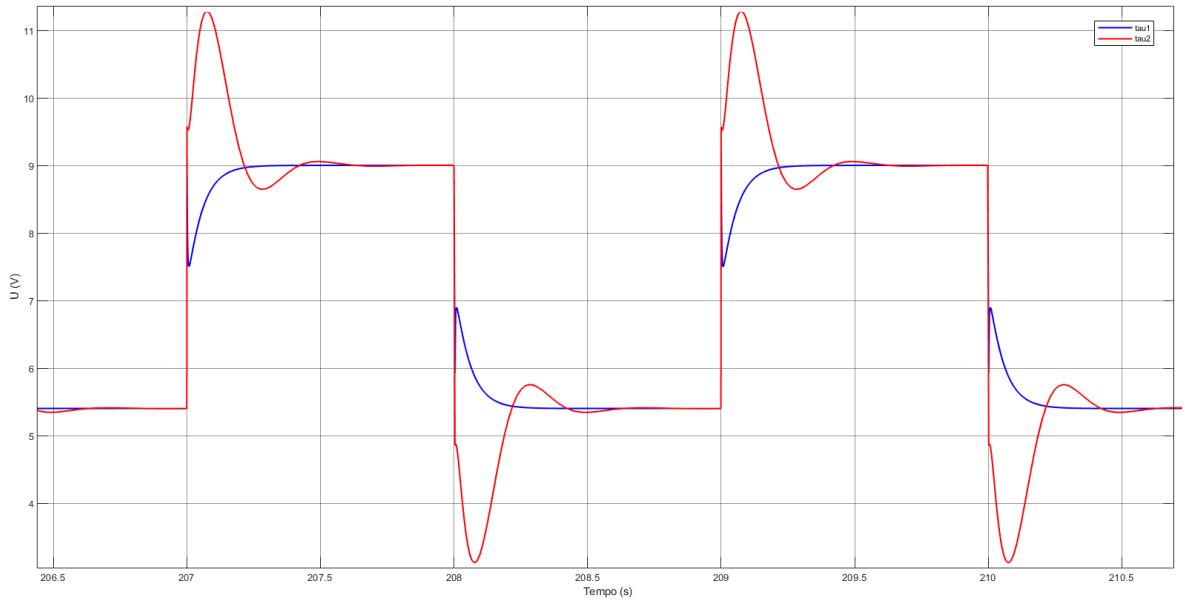


Tabela 4.6 – Análise da Resposta Temporal do PID com mudanças na planta

τ	**Tempo de subida (ms)**	**Tempo de acomodação (ms)**	**Sobressinal (%)**
τ_1	92,7	133,6	0
τ_2	107,0	269,9	6,7

Para análise do sinal de controle, tem-se o resultado mostrado na figura 4.38 sendo que o comportamento do mesmo em resposta ao comportamento mais rápido da planta, representada por τ_1, é mais abrupto, dando um salto no momento do degrau, apresentando picos de tensão em 9,3 V, mas seguindo em regime permanente 9 V de máximo e 5,4 V de mínimo. Já a resposta ao comportamento mais devagar da planta, representada por τ_2, apresenta picos e oscilações maiores no sinal de 11,29 V de amplitude e os mínimos ocorrem com 3,11 V. Importante observar também que os dois sinais de controle têm pulsos abruptos verticais quando ocorre a transição do degrau.

Figura 4.38 – Sinal de controle PID com mudanças da planta



Para a técnica de controle MRAC, o comportamento é mostrado na figura 4.39, mantendo a mesma taxa de aprendizagem γ, com as características de desempenho em determinados instantes de tempo apresentadas na tabela 4.7. É possível perceber o controle consegue se adaptar melhor ao longo do tempo à planta com a constante de tempo maior, contando com uma dinâmica mais lenta, por exemplo analisando a redução de sobressinal da resposta, fazendo com que seja menos amortecida a cada iteração. No caso da constante de tempo menor, simbolizando uma dinâmica mais rápida da planta, a resposta se comporta como um sistema de primeira ordem, sem sobressinal, oferecendo uma adaptação também mais rápida, porém mais abrupta do ponto de vista de características temporais, como pode-se perceber pelas características de tempo de subida e de acomodação.

Figura 4.39 – Simulação do controle MRAC com mudanças da planta

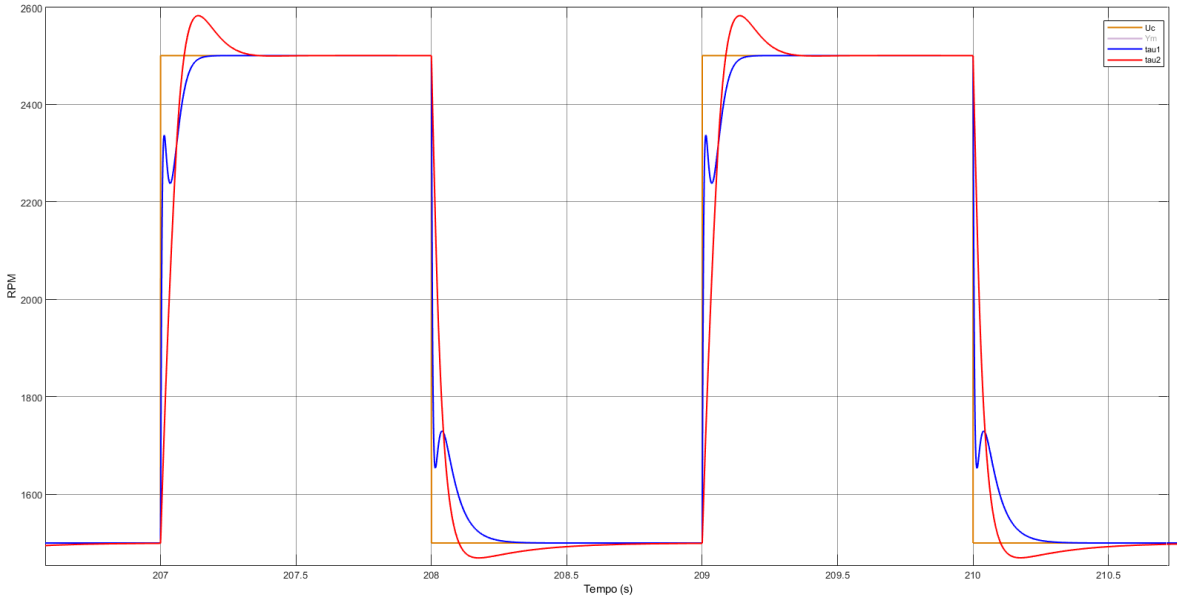


Tabela 4.7 – Análise da Resposta Temporal do MRAC com mudanças na planta para diferentes instantes de tempo

Tempo (s)	**Tempo de subida (ms)**		**Tempo de acomodação (ms)**		**Sobressinal (%)**		**Erro (rpm)**	
	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2	τ_1	τ_2
1	75,5	101,9	93,7	535,8	0	13,7	444,0	457,1
25	76,0	96,2	94,2	331,8	0	7,6	447,5	336,9
85	77,1	92,6	95,8	219,6	0	4,3	456,2	268,4
125	76,1	89,1	96,7	211,4	0	3,8	457,4	269,1
175	77,8	87,7	96,4	204,3	0	3,5	464,0	268,8
250	77,8	85,7	98,2	194,9	0	3,1	473,9	142,2

Os valores dos parâmetros adaptativos foram alterados de acordo com cada dinâmica. Pelas figuras 4.40 e 4.41, pode-se notar que o parâmetro da planta com dinâmica mais rápida resultou em uma convergência mais rápida, e os 300 segundos de simulação foram suficientes para estabilizá-los. Por outro lado, os parâmetros referentes à planta com dinâmica mais lenta não haviam terminado de convergir para seus valores finais, devido ao fato de a adaptação ainda não ter sido completa nesse intervalo de tempo analisado. Os valores dos parâmetros para as duas constantes de tempo ao final da simulação são destacados na tabela 4.8.

Figura 4.40 – Alteração dos parâmetros do MRAC ao longo do tempo para ajuste à variação da planta com τ_1

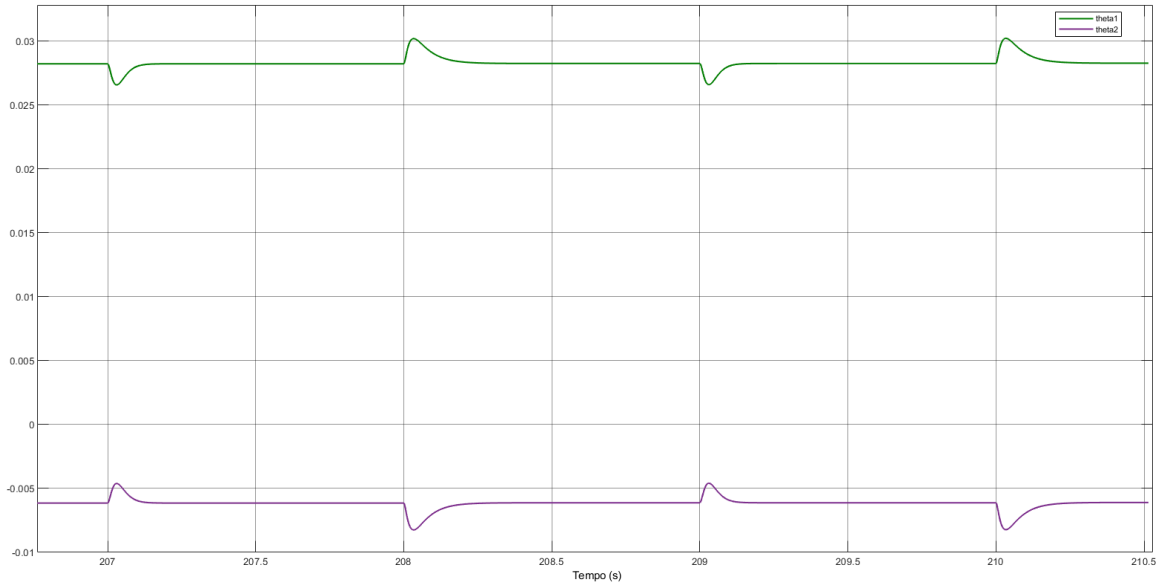


Figura 4.41 – Alteração dos parâmetros do MRAC ao longo do tempo para ajuste à variação da planta com τ_2

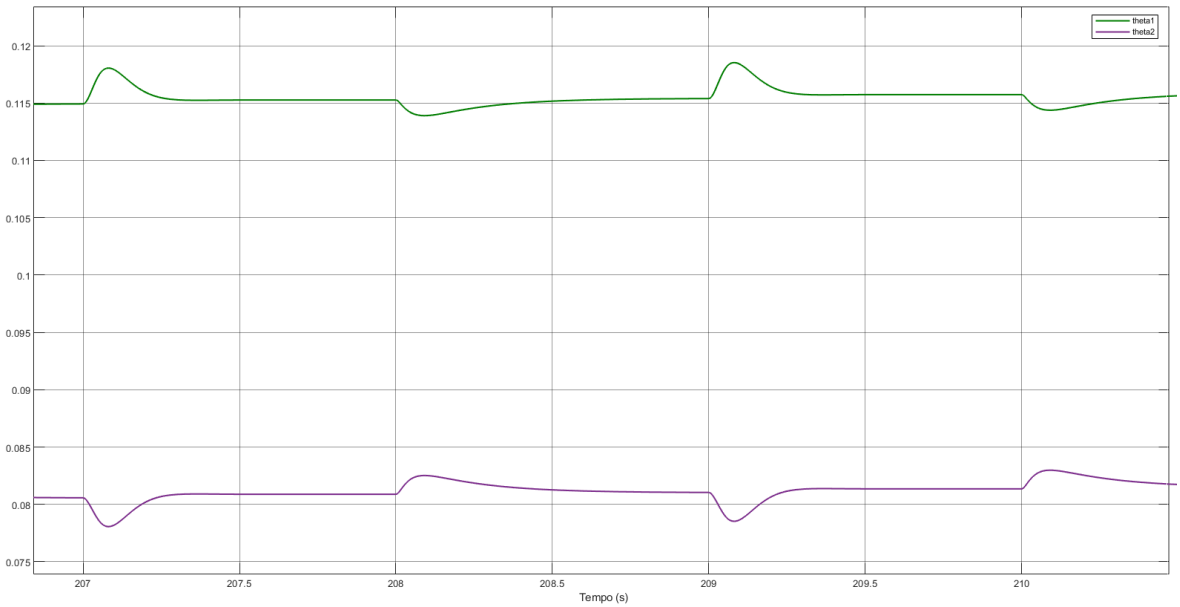
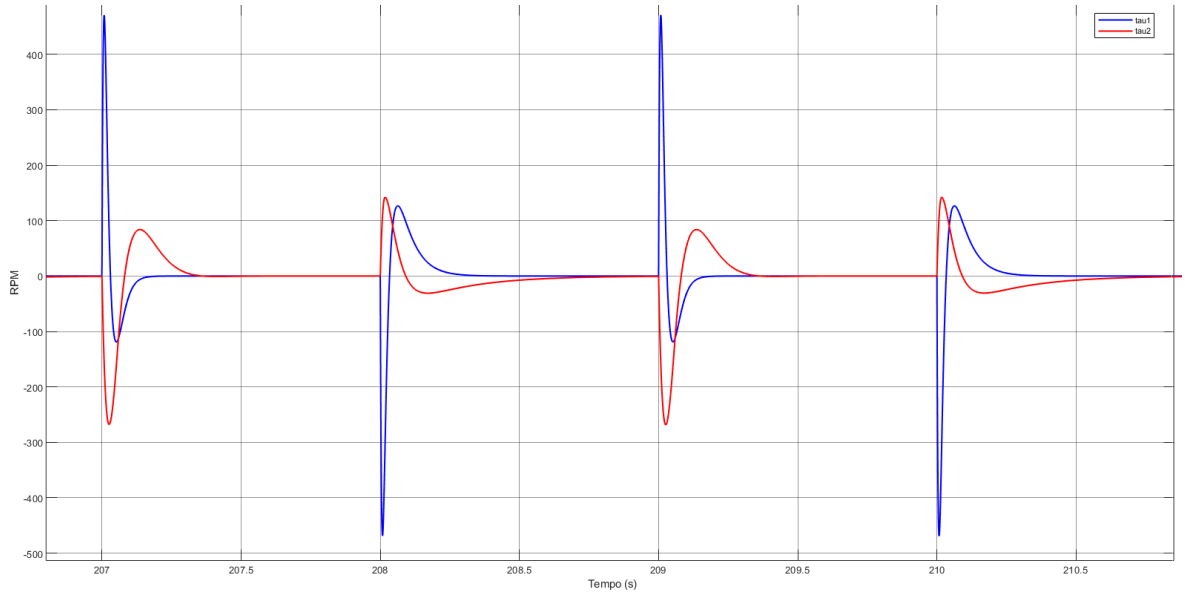


Tabela 4.8 – Parâmetros do controlador MRAC com alteração temporal

Parâmetros	τ_1	τ_2
θ_1	$3{,}17 \times 10^{-2}$	$1{,}76 \times 10^{-1}$
θ_2	$-2{,}68 \times 10^{-3}$	$1{,}42 \times 10^{-1}$

Por fim, o sinal de erro entre as saídas desejada e da planta é como mostra a figura 4.42, e na última coluna da tabela 4.7 é possível perceber como o erro se comporta à medida que o tempo passa para as duas dinâmicas de tempo. Pela simulação, é possível perceber que esse sinal diminui ao longo do tempo, mostrando que a diferença entre os sinais desejado e reais vão diminuindo.

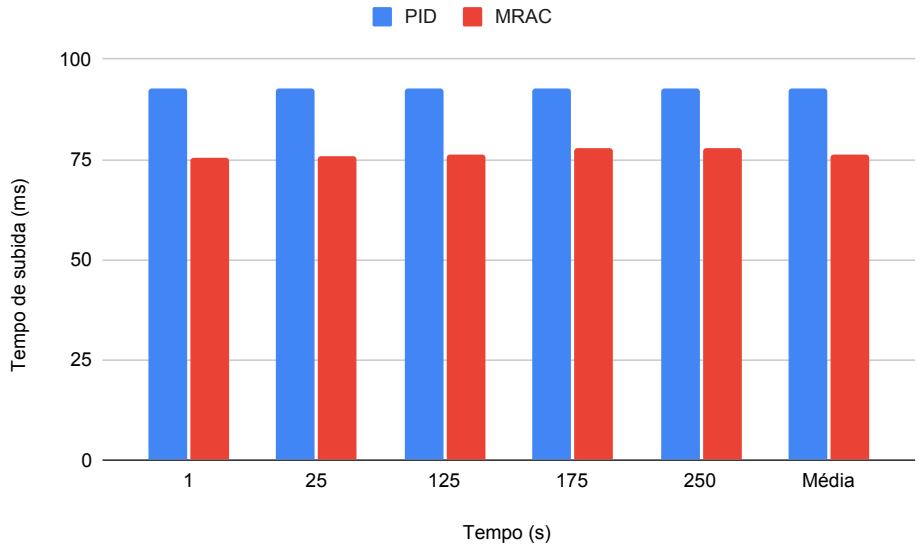
Figura 4.42 – Erro entre as saídas do MRAC ao longo do tempo



Em suma, é possível fazer a análise direta de diferença de comportamento das respostas dos controles PID e do MRAC devido às alterações da dinâmica da planta, por meio dos parâmetros temporais medidos e mostrados nas Tabelas 4.6 e 4.7. Para a tabela do MRAC foram feitas várias medidas ao longo do tempo devido à ser um controle feito "ao vivo", ou seja, a atualização dos parâmetros do controlador é dinâmica, e teve com o objetivo a análise de adaptação do mesmo. Por outro lado, como o PID é um controle convencional, que não tem a adaptação de parâmetros dinâmica, não foi necessário fazer diferentes medições, já que ele sempre se comporta de maneira rígida para cada entrada, bastando as medições de um período.

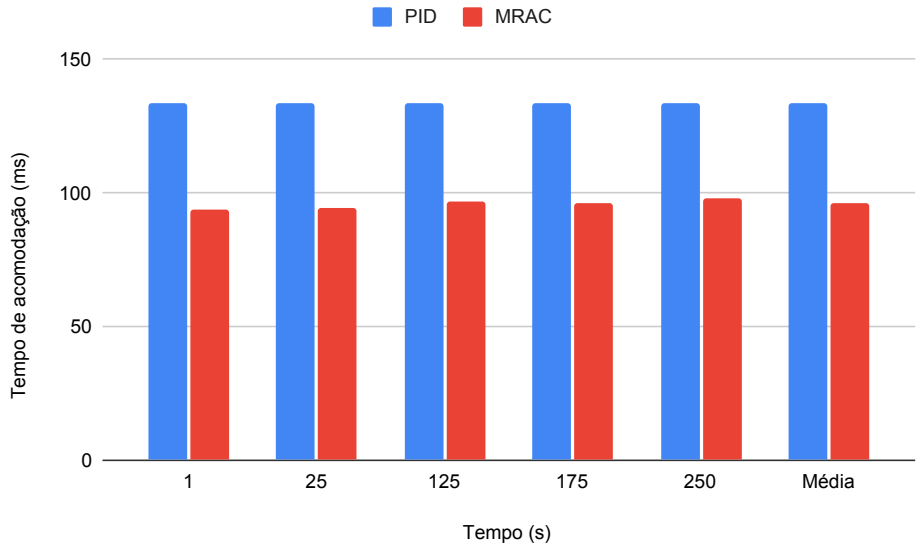
De posse desses dados, é notável que o comportamento do PID para τ_1 tem um maior tempo de subida se comparado à mesma dinâmica observada no sinal de resposta do MRAC para todos os instantes de tempo medidos, foi usada a média de valores de τ_1 do MRAC já que o sinal conseguiu seguir com valores muito próximos ao longo do tempo em relação ao sinal de entrada com dinâmica mais rápida. Assim, obteve-se o resultado de que o MRAC é aproximadamente 17,37% mais rápido que o PID. A figura 4.43 mostra os dados quanto ao tempo de subida graficamente, bem como a média calculada desses valores para o MRAC em comparação com o valor constante do PID no último par de colunas.

Figura 4.43 – Comparação do tempo de subida entre o PID e o MRAC para τ_1



Para análise do tempo de acomodação, percebe-se novamente que o MRAC é mais rápido que o PID com uma diferença percentual de 28,26% analisando a média de valores medidos do MRAC. Por fim, os dois sinais não apresentam sobressinal para τ_1, não sendo possível comparar a dinâmica por essa característica temporal. A figura 4.44 mostra os dados quanto ao tempo de acomodação graficamente.

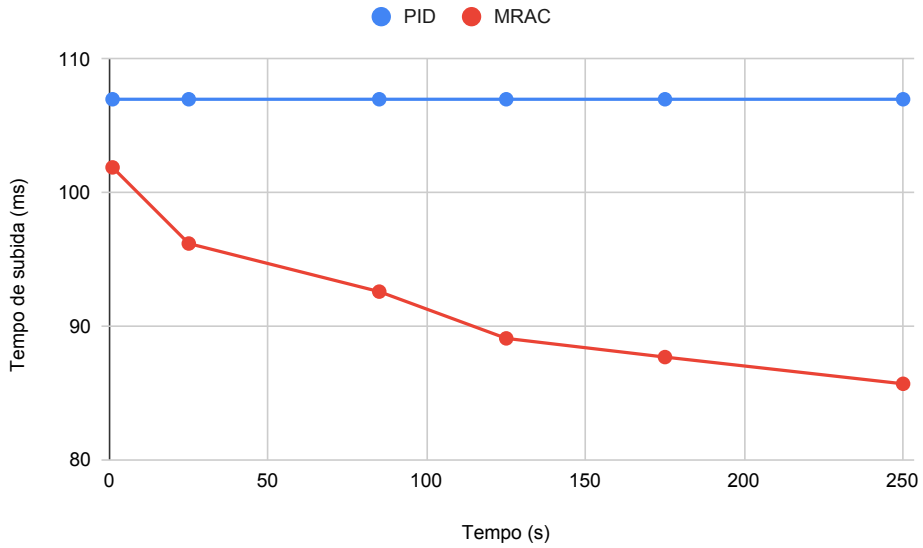
Figura 4.44 – Comparação do tempo de acomodação entre o PID e o MRAC para τ_1



Para τ_2, as comparações não puderam ser condensadas e feita a média de valores como para τ_1 devido ao fato de a faixa de valores do MRAC apresentar muitas diferenças ao longo do tempo, mostrando a adaptação. Assim, foi feita uma análise mais detalhada em relação à passagem do tempo para comparar as duas técnicas, ilustrada na figura 4.45. Para o

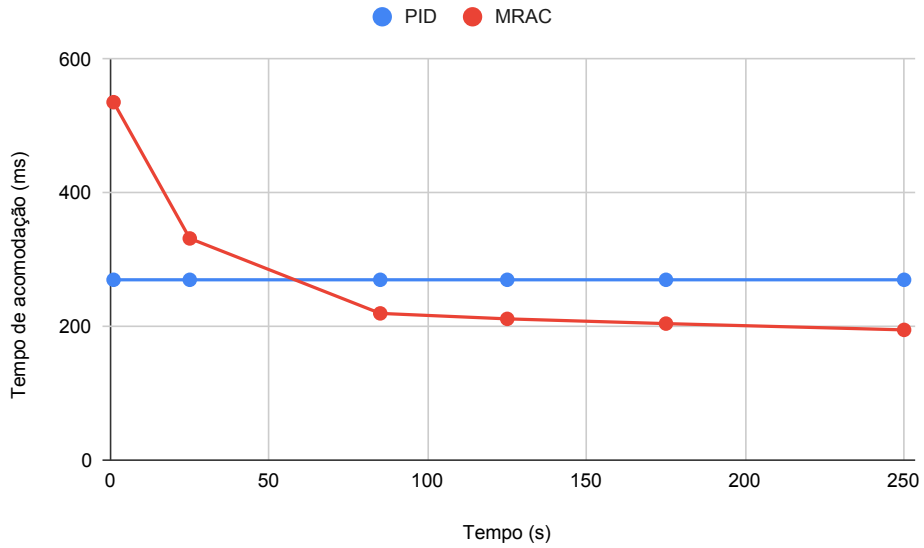
tempo de subida, o MRAC tem uma resposta 4,8% mais rápida que o PID desde sua primeira iteração, e, ao longo do tempo, essa porcentagem é incrementada devido à adaptação dos parâmetros, na última medição feita do MRAC a diferença percentual é de aproximadamente 20%.

Figura 4.45 – Comparação do tempo de subida entre o PID e o MRAC para τ_2



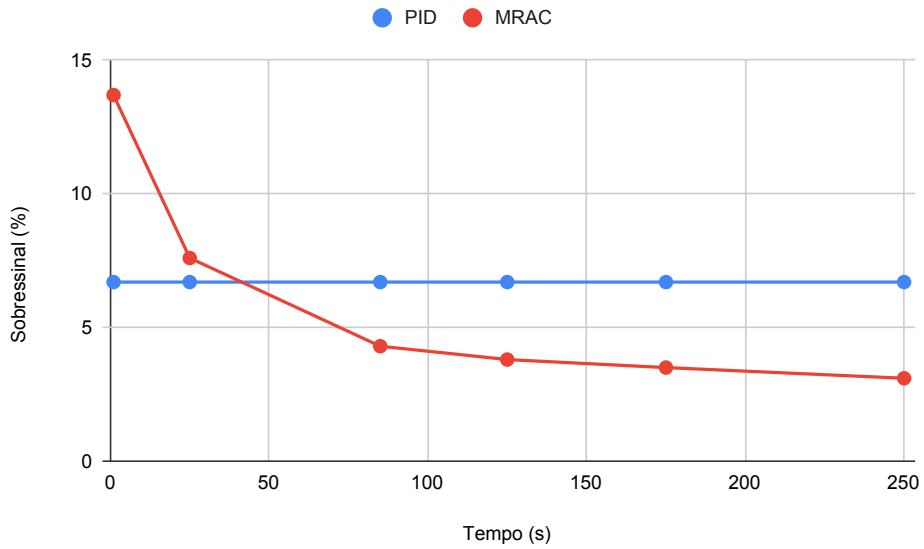
Em um segundo momento, com a finalidade de comparação do tempo de acomodação, até entre os instantes de tempo medidos de 1 *s* a 25 *s*, o PID apresentava melhor desempenho, podendo ser analisado o instante inicial da simulação, sendo aproximadamente 50% mais rápido que o MRAC. Porém, entre os instantes medidos de 25 *s* a 85 *s* o MRAC conseguiu ajustar seus parâmetros, mostrando um desempenho melhor que o PID. No último tempo medido do MRAC com valor de 194,9 *ms* de tempo de acomodação, ele apresentava a diferença de aproximadamente 28% em relação ao valor rígido de 269,9 *ms* do PID. A figura 4.46 ilustra esse comportamento:

Figura 4.46 – Comparação do tempo de acomodação entre o PID e o MRAC para tau_2



Por fim, para análise quanto ao sobressinal, no início da simulação a resposta do PID conta com esse parâmetro aproximadamente 51% menor em relação ao MRAC, contando com um bom amortecimento inicial. Porém, com o ajuste de parâmetros entre os instantes de 25 s a 85 s, o MRAC consegue superar o PID e, no último valor medido desse controle durante a simulação, a diferença para o PID se torna de aproximadamente 54%, mostrando um sobressinal muito mais aceitável. A figura 4.47 ilustra esse comportamento:

Figura 4.47 – Comparação do sobressinal entre o PID e o MRAC



Assim, é notável o desempenho do MRAC em relação ao PID, mostrando como sua adaptação de parâmetros é importante para mudanças da planta, representadas pela dinâmica tanto mais rápida quanto mais lenta. Mesmo que esse controle ainda custe um

tempo para a adaptação, controles rígidos como o PID não conseguem variar seus parâmetros, fazendo com que seu desempenho deixe a desejar, podendo levar o sistema à instabilidade apenas com um pouco de variação do seu ponto de operação. Ainda, deve-se considerar também a influência da taxa de adaptação no controle adaptativo estudado, que para o caso analisado nesse trabalho teve um bom desempenho, porém para variações da planta muito mais abruptas pode ser necessário ajustá-la para não levar o sistema à instabilidade nem tampouco ser muito lenta para se adaptar ao cenário desejado.

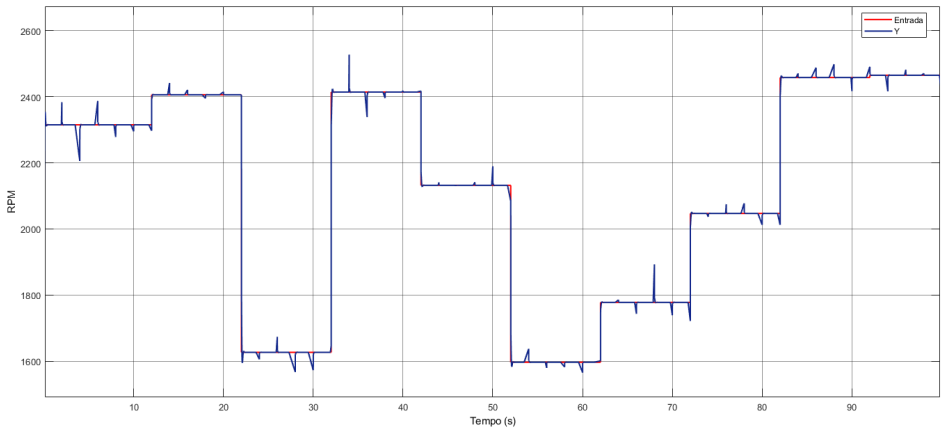
4.3.5 Comparação entre o controle PID e o MRAC - Uma abordagem alternativa

Para que fosse possível uma melhor avaliação de desempenho entre as técnicas de controle, foram alterados algumas características na simulação:

1. Entrada da simulação: bloco de “tempo” e em seguida um bloco “Função do Matlab”, em que a função em código inserida neste último tem como objetivo alterar a escala do degrau de entrada;
2. Ruído aleatório na saída da planta: bloco de “Ruído Branco limitado por Banda” com período de amostragem de 10 segundos, e potência de ruído de 25 $(rad^2/s^2)/Hz$.

A primeira alteração foi realizada com o objetivo de simular diferentes requisições de rotação dos motores a partir do controle de estabilidade em (Botelho, 2020). Já a segunda alteração foi realizada com o objetivo de representar ruídos nas medições de velocidade, testando o comportamento dos controles frente à erros de medições da planta com 1% de erro em relação à velocidade máxima de 2500 rpm. As simulações foram feitas e os resultados são mostrados nas Figuras 4.48 e 4.49:

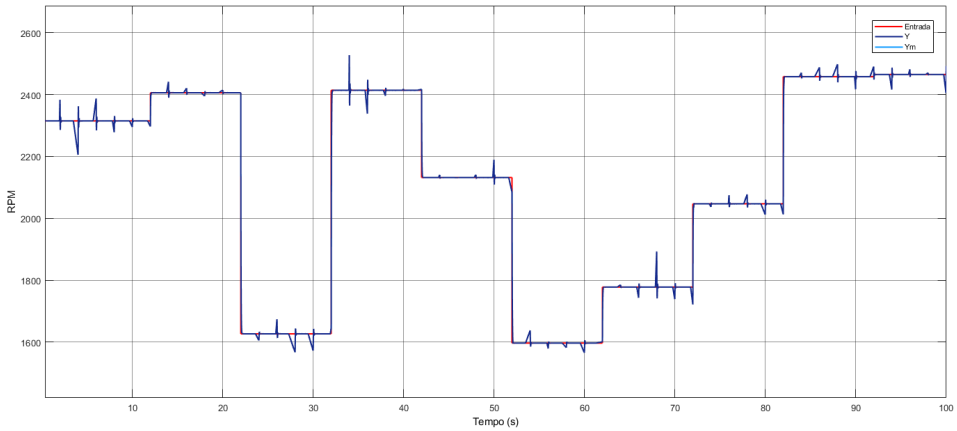
Figura 4.48 – Saída da planta ao controle PID



Uma primeira consideração pertinente para ser feita fundamenta-se na questão da influência do canal derivativo no controlador PID. É possível observar que esse canal agrega

em oscilações indesejadas frente aos ruídos, resultando em certas instabilidades para o controle. Assim, um controlador PI oferece um desempenho e resultado mais robustos, sem a influência derivativa.

Figura 4.49 – Saída da planta ao controle MRAC



Para uma comparação direta entre os dois controles PI e MRAC, pode-se observar que os picos quando ocorrem os ruídos têm sobressinal muito próximos de 2.526 rpm para os dois controles, porém é notável que o MRAC tem maiores oscilações quando ocorrem os ruídos, logo, obtendo maior tempo de acomodação e menor controle que o PI. As Figuras 4.50 e 4.51 mostram em destaque os sinais de comportamento dos dois controles sobre essas análises:

Figura 4.50 – PI

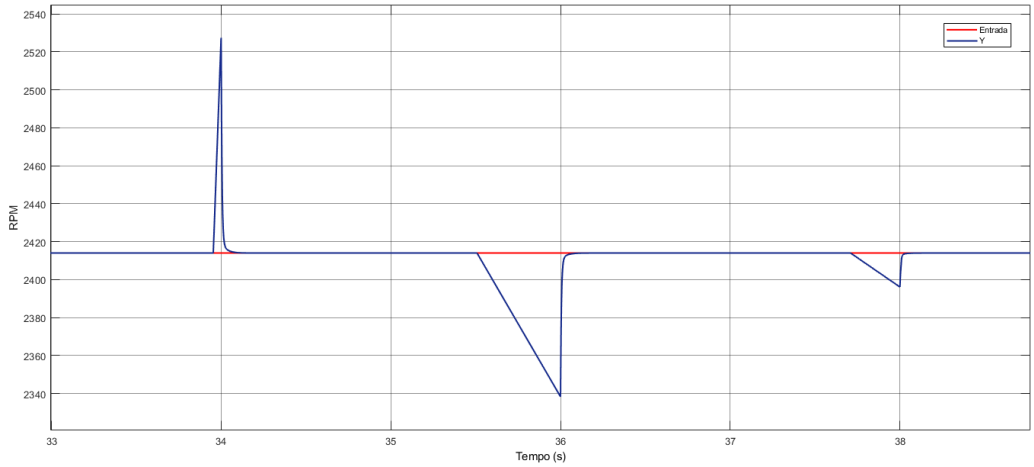
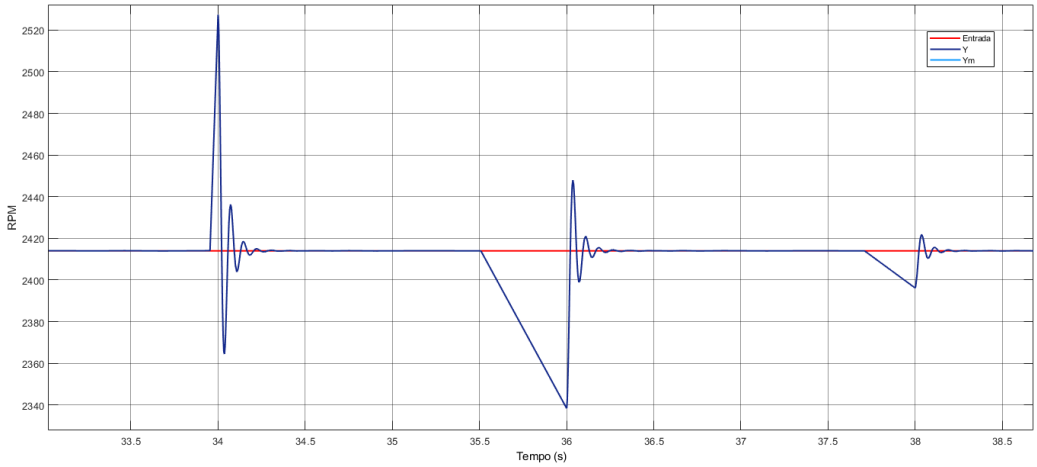


Figura 4.51 – MRAC



4.3.6 Preparação para Implementação Experimental

Para que fosse possível a implementação do controle adaptativo desenvolvido na seção 4.3.3 nos códigos do microcontrolador ESP-32 utilizado nesse trabalho e uma futura validação experimental, foi necessária a alteração da configuração contínua para uma configuração discreta. Assim, as funções de transferência e dinâmicas do controle adaptativo foram discretizadas por meio da discretização exata e feitas equações de diferenças aplicando-se o segurador de ordem zero, que é frequentemente referido como ZOH. Assim, para a função de transferência de referência, foram feitos os seguintes cálculos, considerando o

período de amostragem $T = 1\ ms$:

$$\begin{aligned} G_{m(s)} &= \frac{1}{\tau s + 1} \\ G_{m(z)} &= (1 - z^{-1})\mathcal{Z}\left[\frac{G_{m(s)}}{s}\right] \\ G_{m(z)} &= \frac{1 - e^{-\frac{T}{\tau}}}{z - e^{-\frac{T}{\tau}}} \end{aligned} \tag{4.3}$$

Como $G_{m(s)} = \frac{Y_{m(s)}}{U_{c(s)}}$, então foi desenvolvida a equação de diferenças para $G_{m(s)}$ e implementado no código do Matlab a equação (4.4):

$$y_{[k]} = (1 - e^{-\frac{T}{\tau}})u_{[k-1]} + e^{-\frac{T}{\tau}} y_{[k-1]} \tag{4.4}$$

Para cada bloco do sistema mostrado na figura 4.31 foi feita uma equação de diferenças como acima e implementado em um bloco *Matlab Function* as equações referentes a cada bloco que podem ser vistas no apêndice A. Assim, elas foram testadas em simulação para corroborar com as respostas do sistema contínuo, como pode ser visto o resultado na figura 4.52 e a tabela 4.9 mostra as características de desempenho em alguns instantes de tempo simulados. Ainda, foi levado em consideração que para sobressinais menores que 1% a resposta do sistema era considerada como sendo de primeira ordem, sendo calculado seu tempo de subida com os instantes em que a resposta leva para atingir de 10% a 90% do seu valor final.

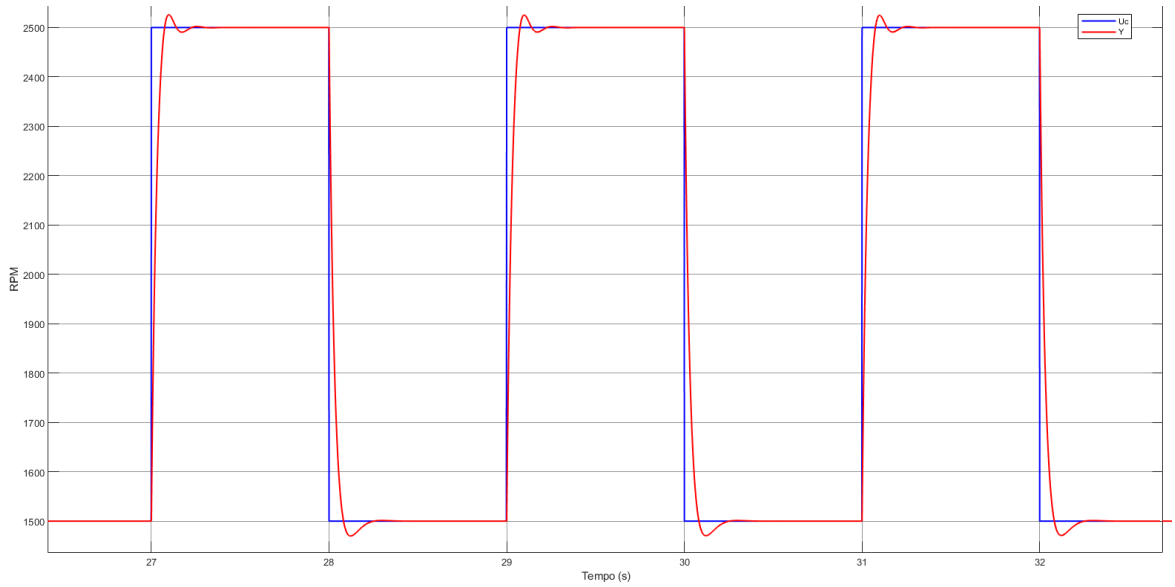


Figura 4.52 – Simulação do MRAC discretizado

Tabela 4.9 – Características da resposta ao MRAC discretizado para diferentes instantes de tempo

Tempo (s)	Tempo de subida (ms)	Tempo de acomodação (ms)	Sobressinal (%)
1	72,9	61,4	1,4
25	75,4	62,1	1,1
85	49,6	63,5	0,5
125	49,1	64,4	0,3
175	49,9	64,8	0.1
250	49,3	64,8	0

Por fim, o sinal de controle discreto simulado enviado para a planta é como mostra a figura 4.53:

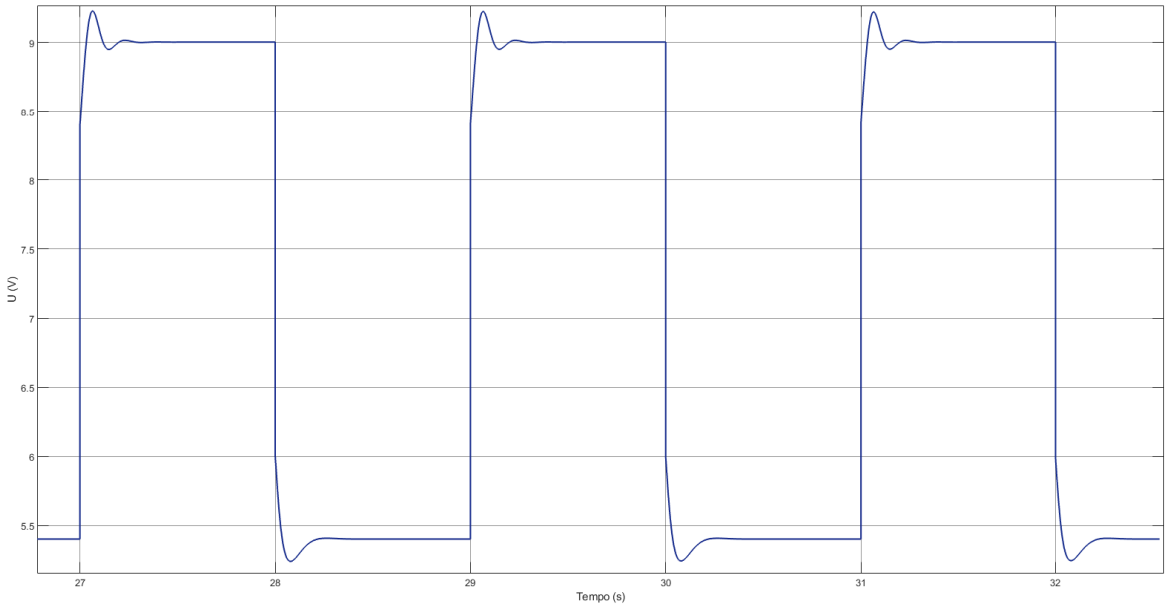


Figura 4.53 – Simulação do controle discretizado

5 Conclusões

5.1 Considerações finais

Com base nos trabalhos anteriores desenvolvidos na UnB, este trabalho realizou: o desenvolvimento de um novo sistema embarcado para "*ESP*32", com foco em análises dos resultados do software embarcado; a criação de um sistema de aquisição de dados, para monitoramento dos dados de voo e apoio na etapa de desenvolvimento; além de simulações e discussões a respeito da implementação de técnicas de controle de rotação dos motores, para um robô aéreo com dois rotores inclináveis.

Os testes do software embarcado, os quais foram realizados de forma modularizada, tiveram resultados bem-sucedidos, utilizando o hardware em placa perfurada, que consiste no protótipo do sistema eletrônico do Tilt-Rotor. Para as medições de distância e de velocidade angular dos motores, os resultados mostraram boa precisão e exatidão das medições; já em relação aos requisitos temporais, estes foram cumpridos para a medição de velocidade angular, e não foram cumpridos para o pior caso da medição de distância. Os resultados das medições do acelerômetro, giroscópio e magnetômetro foram mostrados no monitor serial, assim como os resultados do sistema de trocas de mensagens por TCP/IP.

Ainda, a implementação do sistema de dados como uma interface de monitoramento remoto se mostrou uma ferramenta bastante útil para a plataforma, de forma a facilitar a interação entre o operador e o próprio microcontrolador do robô aéreo, podendo-se obter, em tempo real, tanto a visualização gráfica das medições dos sensores quanto a atualização dos parâmetros dos controladores.

Por fim, esse trabalho mostrou diferentes resultados tanto para a técnica de controle PID quanto para a técnica de controle MRAC devido às condições impostas no sistema, como os diferentes sinais de entrada e testes com ruídos no sistema, além de variações dinâmicas na planta simulados no ambiente Simulink do Matlab. Assim, pode-se notar que o MRAC tem um custo computacional maior e, sob determinadas circunstâncias como as mostradas nesse trabalho, melhor desempenho em relação ao PID. Porém, é possível perceber que este último tem grande flexibilidade com as alterações de amplitudes de entrada, oferecendo respostas menos oscilantes e mais rápidas para o sistema em comparação ao MRAC. Além disso, oferecendo um foco maior então à técnica PID, pôde-se notar que o termo derivativo oferece contribuições negativas, devido à sua tentativa de previsão de erros frente aos distúrbios inseridos. Assim, um controlador PI se mostrou mais robusto após as análises dos tipos de técnicas de controle para os motores.

5.2 Sugestões para trabalhos futuros

Como sugestão para trabalhos futuros, pode-se seguir com a validação e comparação experimental dos controles MRAC e PI, a partir da implementação dos códigos discretizados no microcontrolador, juntamente com o controle em cascata de altitude e atitude, observando como será o comportamento da velocidade de rotação dos motores do robô aéreo em relação às mudanças dinâmicas da planta. Ainda, se torna interessante desenvolver também controles de trajetória para o Tilt-rotor, para ser possível atingir voos com objetivos pré-determinados, como transporte de cargas, reconhecimento de áreas, entre outros.

Além disso, propostas gerais para continuação do projeto são: a realização de mais validações e otimizações do software embarcado e também do protótipo do hardware, para que possa ser feita a fabricação das duas placas de circuito impresso, utilizando os esquemáticos já desenvolvidos; a compra de uma nova bateria, baseada na estimação da autonomia desejada para o robô aéreo; a implementação do sistema de localização (com calibração dos sensores); a adição de mais funcionalidades de comunicação com o Sistema de Aquisição de Dados; a integração de todo o sistema para a realização de voos com decolagem e pouso verticais; e, no futuro, alcançar a evolução do projeto para possibilitar a transição para voo horizontal, a qual exige modificações como receber comandos de um controle remoto, incluir um dispositivo GPS e adicionar asas na estrutura mecânica.

Referências

AUTOMATION, R. **'Sensores Discretos'**. [*S.l.*], 2022. Citado nas pp. 33, 34, 36, 38 e 39.

BAN, Z.; CRNOSIJA, P. Application of the mrac with simplified discrete parameter adaptation algorithm for control of the dc electromotor drive. p. 506–511, 2003. Citado na p. 56.

BERNARDINO, P.; BROTHERHOOD, L. **Drones nas geociências**. 2018. https://www.cratonconsultoria.com/post/drones-nas-geociencias. Online; acesso em 19 de jan. de 2024. Citado na p. 18.

BIAGIONI, I. M.; CARLOS, S. **IMPLEMENTAÇÃO DE CONTROLE ADAPTATIVO POR SIMULINK**. 2013. Citado na p. 53.

BORGES, P. D. G. A. **'GDATLogger: A Data Logger for Embedded Systems', Repositório no GitHub, LARA-UNB**. 2018. Disponível em: https://github.com/lara-unb/gdatalogger. Acesso em 30 Nov. 2024. Citado na p. 69.

BOTELHO, L. C. **'Modelagem e Estabilização de Plataforma Aérea Tilt-Rotor', Trabalho de Graduação**. 2020. Citado nas pp. 19, 22, 102, 104 e 119.

BRAY, T. The javascript object notation (json) data interchange format. **ECMA International**, v. 1st Editio, 2014. ISSN 2070-1721. Citado na p. 28.

CANAL, I. P.; VALDIERO, A. C.; REIMBOLD, M. P. Modelagem matemática de motor de corrente contıinua e analise dinâmica. *In*: **CNMAC 2016 - XXXVI Congresso Nacional de Matematica Aplicada e Computacional**. [*s.n.*], 2017. Disponível em: https://proceedings.sbmac.org.br/sbmac/article/view/1358. Citado na p. 73.

CASTRO, D. S. **Controle Adaptativo Aplicado a um Sistema de Rastreamento Solar**. 2018. Citado na p. 54.

CASTRUCCI, P. B. de L.; BITTAR, A.; SALES, R. M. **Controle Automático**. 1. ed. [*S.l.*]: GEN/LTC, 2011. ISBN 9788521617860. Citado na p. 77.

CATALDI, G. L. P. **'Adequação de Hardware e Implementação de um Sistema de Controle de um Robô Aéreo com Dois Rotores Articulados', Trabalho de Graduação**. 2017. Citado nas pp. 19, 20 e 21.

CHAPMAN, S. J. **Fundamentos de Máquinas Elétricas**. 5. ed. [*S.l.*]: AMGH EDITORA LTDA, 2013. Citado na p. 40.

DIGITAL, O. **30A BLDC ESC**. 2015. Disponível em: https://www.optimusdigital.ro/ro/index.php?controller=attachment&id_attachment=451. Acesso em 30 Nov. 2024. Citado nas pp. 40 e 42.

EARSUIT. **How to read data from MPU9250**. 2018. Disponível em: https://longnight975551865.wordpress.com/2018/02/11/how-to-read-data-from-mpu9250/. Acesso em 30 Nov. 2024. Citado na p. 38.

ELETROGATE. **MH-Sensor-Series, Fotosensor com Sensor Infravermelho TCRT5000**. 2018. Disponível em: https://www.eletrogate.com/modulo-seguidor-de-linha-tcrt5000. Acesso em 30 Nov. 2024. Citado nas pp. 33 e 34.

ESPRESSIF, S. **'ESP32 Technical Reference Manual', Version 5.0**. 2023. Disponível em: https://www.espressif.com. Acesso em 10 Jan. 2025. Citado na p. 29.

ESPRESSIF, S. **'ESP-IDF Programming Guide', Release v5.3.1**. [*S.l.*], 2024. Citado nas pp. 30, 64, 66, 68 e 69.

ESPRESSIF SYSTEMS. **ESP32 Series Datasheet**. [*S.l.*], 2023. 2.4 GHz Wi-Fi + Bluetooth® + Bluetooth LE SoC. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Citado nas pp. 29 e 30.

EVANS, G. Bringing root locus to the classroom. **IEEE Control Systems Magazine**, v. 24, n. 6, p. 74–81, 2004. Citado na p. 50.

HASSAN, A. A.; AL-SHAMAA, N. K.; ABDALLA, K. K. Comparative study for dc motor speed control using pid controller. **International Journal of Engineering and Technology**, ENGG Journals Publications, v. 9, p. 4181–4192, 12 2017. ISSN 23198613. Citado na p. 56.

HEIN, M. **Brushless-DC Made Simple – Sensored Motor Control (Rev. B)**. [*S.l.*], 2021. Acesso em: 03 fev. 2025. Disponível em: https://www.ti.com/lit/pdf/slva980b. Citado na p. 73.

HUINFINITO. **Módulo Conversor Nível Lógico 5V/3.3V - Bidirecional (4 Canais)**. [*S.l.*], 2025. Disponível em: https://www.huinfinito.com.br/modulos/1523-modulo-conversor-nivel-logico-5v33v-bidirecional-4-canais.html. Disponível em: https://www.huinfinito.com.br/modulos/1523-modulo-conversor-nivel-logico-5v33v-bidirecional-4-canais.html. Citado na p. 59.

INSTRUMENTS, T. **LMS40, LM340A and LM7805 Family Wide VIN 1.5-A Fixed Voltage Regulators**. [*S.l.*], 2016. SNOSBT0L –FEBRUARY 2000–REVISED SEPTEMBER 2016. Disponível em: https://www.ti.com/lit/ds/symlink/lm340.pdf. Citado na p. 61.

INVENSENSE. **'MPU-9250 Register Map and Descriptions', Revision 1.6, Document Number: RM-MPU-9250A-00**. [*S.l.*], 2015. Citado na p. 38.

INVENSENSE. **'MPU-9250 Product Specification', Revision 1.1, Document Number: PS-MPU-9250A-01**. [*S.l.*], 2016. Citado nas pp. 36, 37 e 38.

JAIN, P.; NIGAM, M. J. Design of a model reference adaptive controller using modified mit rule for a second order system. v. 3, p. 477–484, 2013. ISSN 2231-1297. Disponível em: http://www.ripublication.com/aeee.htm. Citado na p. 56.

KANSO, W. M. **MODEL REFERENCE ADAPTIVE CONTROL ALGORITHM ON A SECOND ORDER DYNAMIC MODEL**. 1991. Citado na p. 56.

Maxon Motor AG. **Datasheet do Motor Maxon 2130 – 30 mm, Escovas de Grafite, 3 W**. 2013. Acesso em: 06 fev. 2024. Disponível em: https://www.maxongroup.com/medias/sys_master/8807088717854.pdf. Citado nas pp. 102, 103 e 104.

MICROELECTRONICS, S. **PC817X Series**. [*S.l.*], 2003. Citado nas pp. 35 e 36.

MODELISMO, A. **Motor Brushless TURNIGY D2830-11 1000KV com Spinner e Suporte Aerodrive D2830**. 2024. Disponível em: https://www.actionmodelismo.com.br/motor-eletrico-esc/motorbrushlessturnigyd2830-11-1000kvcomspinneresuporteaerodrived2830. Acesso em 30 Nov. 2024. Citado nas pp. 39 e 40.

MORGAN, E. J. **HC-SR04 Ultrasonic Sensor**. 2014. Disponível em: https://www.alldatasheet.com/datasheet-pdf/download/1132204/ETC2/HCSR04.html. Acesso em 30 Nov. 2024. Citado nas pp. 31, 32 e 33.

MOSAAD, M. Dc motor control using model reference adaptive control. **Yanbu Journal of Engineering and Science**, Yanbu Industrial College, v. 20, 5 2023. ISSN 1658-5321. Citado na p. 56.

NASA. **Aeronave experimental X-15 para sistemas de controle avançados**. 1961. https://images.nasa.gov/details/ET61-0140. Acesso em: [27/08/2024]. Citado na p. 52.

NISE, N. S. **Engenharia de Controle de Sistemas**. 6. ed. Rio de Janeiro: LTC, 2012. Citado nas pp. 44, 45 e 77.

OGATA, K. **DISCRETE-TIME CONTROL SYSTEMS**. 2. ed. [*S.l.*]: Prentice-Hall, 1995. ISBN 0-13-328642-8. Citado na p. 46.

OGATA, K. **Engenharia de Controle Moderno**. 5. ed. [*S.l.*]: Pearson Prentice Hall, 2010. Citado nas pp. 44, 45, 47, 48, 49, 50, 51 e 110.

OLIVEIRA, R. S. D. **Fundamentos dos Sistemas de Tempo Real**. [*S.l.: s.n.*], 2018. ISBN 9781728694047. Citado na p. 23.

PROBOTS. **MG90S, DS3230 PRO Servo Motor Datasheet**. 2024. Disponível em: https://probots.co.in/ds3230-30kg-180-degrees-high-torque-servo-metal-gear-ip65.html. Acesso em 30 Nov. 2024. Citado na p. 43.

REMPFER, K. **The V-280 tilt-rotor aircraft could change the way air assault troops operate**. 2018. https://www.armytimes.com/news/your-army/2018/03/01/

the-v-280-tilt-rotor-aircraft-could-change-the-way-air-assault-troops-operate/. Online; acesso em 21 de dez. de 2023. Citado na p. 18.

ROBóTICA, C. da. **Regulador com Ajuste de Corrente e Tensão XL4015 5A DC DC**. [*S.l.*], 2023. Disponível em: https://www.casadarobotica.com. Disponível em: https://www.casadarobotica.com/fonte-e-conversores/fontes/reguladores-de-tensao/regulador-com-ajuste-de-corrente-e-tensao-xl4015-5a-dc-dc?srsltid=AfmBOopbN1DBqN1BNESYoGaW64SnIVkOLP6eYJWwW-tAqCCwf9hn8eMU. Citado nas pp. 43 e 61.

SACHIT, S.; VINOD, B. R. Mras based speed control of dc motor with conventional pi control — a comparative study. **International Journal of Control, Automation and Systems**, Institute of Control, Robotics and Systems, v. 20, 1 2022. ISSN 20054092. Citado na p. 56.

SEMICONDUCTOR, O. **P2N2222A - Amplifier Transistors NPN Silicon**. [*S.l.*], 2021. Disponível em: https://www.onsemi.com/pdf/datasheet/p2n2222a-d.pdf. Disponível em: https://www.onsemi.com/pdf/datasheet/p2n2222a-d.pdf. Citado na p. 59.

SEMICONDUCTOR, V. G. **UF5408 - Ultrafast Rectifier Diode**. [*S.l.*], 2022. Revision: 31-Mar-2022 Document Number: 88756. Disponível em: https://www.vishay.com/docs/88516/uf5408.pdf. Citado na p. 61.

SWARNKAR, P.; JAIN, S.; NEMA, R. K. **Effect of Adaptation Gain in Model Reference Adaptive Controlled Second Order System**. 2011. 70-75 p. Disponível em: www.etasr.com. Citado nas pp. 55, 56 e 107.

SYSTEMS, E. **FreeRTOS API Reference - ESP-IDF Programming Guide**. 2023. https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos_idf.html. Acessado em: 10 de junho de 2024. Citado na p. 29.

TANENBAUM, A. S.; WETHERALL, D. J. **Redes de Computadores**. 5. ed. [*S.l.*]: Pearson, 2011. Citado nas pp. 26 e 28.

WALKER, J.; RESNICK, R.; HALLIDAY, D. **Halliday & Resnick fundamentals of physics**. 10th edition. ed. [*S.l.*]: Wiley, 2014. ISBN 978-1-118-23376-4 978-1-118-23072-5 978-1-118-23073-2. Citado na p. 74.

YU, J. Z. **Brushless DC Motor Fundamentals - Application Note**. [*S.l.*], 2011. Citado nas pp. 40, 41 e 42.

ÅSTRöM, K. J.; WITTENMARK, B. **Adaptive Control**. 2nd. ed. Mineola, NY: Dover Publications, 2008. Citado nas pp. 52, 53, 55 e 80.

Apêndices

Apêndice A – Códigos Matlab

Código A.1 – Código do Controle MRAC discretizado

```
function U_atual = controle(Y_atual, Uc_atual, T, gamma, tau)
    persistent Uc_anterior_1 Y_anterior_1 Ym_anterior_1;
    persistent w1_anterior_1 w2_anterior_1 w3_anterior_1;
    persistent w6_anterior_1 w7_anterior_1 w8_anterior_1;

    if isempty(Uc_anterior_1) || isempty(Y_anterior_1) ||
       isempty(Ym_anterior_1) || isempty(w1_anterior_1) ||
       isempty(w2_anterior_1) || isempty(w3_anterior_1) ||
       isempty(w6_anterior_1) || isempty(w7_anterior_1) ||
       isempty(w8_anterior_1)
        Uc_anterior_1 = 0;
        Ym_anterior_1 = 0;
        Y_anterior_1 = 0;
        w1_anterior_1 = 0;
        w2_anterior_1 = 0;
        w3_anterior_1 = 0;
        w6_anterior_1 = 0;
        w7_anterior_1 = 0;
        w8_anterior_1 = 0;
    end

    Ym_atual = ((1-exp(-T/tau))*Uc_anterior_1) +
       (exp(-T/tau)*Ym_anterior_1);
    erro = Y_atual - Ym_atual;
    w1_atual = ((1-exp(-T/tau))*Uc_anterior_1) +
       (exp(-T/tau)*w1_anterior_1);
    w6_atual = ((1-exp(-T/tau))*Y_anterior_1) +
       (exp(-T/tau)*w6_anterior_1);
    w2_atual = w1_atual * erro;
    w7_atual = w6_atual * erro;
    w3_atual = (-gamma*T*w2_anterior_1) + w3_anterior_1;
    w8_atual = (gamma*T*w7_anterior_1) + w8_anterior_1;
    w4_atual = w3_atual * Uc_atual;
    w5_atual = w8_atual * Y_atual;
    U_atual = w4_atual - w5_atual;

    Uc_anterior_1 = Uc_atual;
    Ym_anterior_1 = Ym_atual;
    w1_anterior_1 = w1_atual;

    w2_anterior_1 = w2_atual;
    w3_anterior_1 = w3_atual;

    Y_anterior_1 = Y_atual;
    w6_anterior_1 = w6_atual;

```

```
40      w7_anterior_1 = w7_atual;
41      w8_anterior_1 = w8_atual;
42
43  end
```