Trabalho de Graduação

Projeto e desenvolvimento de interface por comando de voz para cadeira de rodas motorizada

Por,
Alexandre Aragão Souza Coelho

Brasília, Novembro de 2021

Engenharia Mecatrônica

Universidade de Brasília

UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia

Curso de Graduação em Engenharia de Controle e Automação

Trabalho De Graduação

Projeto e Desenvolvimento de Interface por Comando de Voz para Cadeira de Rodas Motorizada

Por

Alexandre Aragão Souza Coelho

Relatório submetido como requisito parcial para obtenção

do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Walter de Britto Vidal Filho, UnB/ MEC (Orientador) _____

Carlos Humberto Llanos Quintero, UnB/ MEC _____

Jones Yudi Mori Alves da Silva, UnB/ MEC _____

Brasília, Novembro de 2021

Ficha Catalográfica

Alexandre Aragão Souza Coelho

Projeto e Desenvolvimento de Interface por Comando de Voz para Cadeira de Rodas Motorizada,

[Distrito Federal] 2021.

xi,68p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2021). Trabalho de Graduação - Universidade de Brasília. Faculdade de Tecnologia.

1.Cadeira de Rodas 2.Automação

3.Comando de voz 4.Celular

I. Mecatrônica/FT/UnB

Referência Bibliográfica

Coelho, A. A. S., (2021). Projeto e Desenvolvimento de Interface por Comando de Voz para Cadeira de Rodas Motorizada. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 07/2021, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 80p.

Cessão de Direitos

AUTOR: Alexandre Aragão Souza Coelho

Projeto e Desenvolvimento de Interface por Comando de Voz para Cadeira de Rodas Motorizada: Desenvolvimento e implementação de um sistema comandado por voz para controle de uma cadeira de rodas motorizada.

GRAU: Engenheiro ANO: 2021

Alexandre Aragão Souza Coelho

SHIN 5 Conjunto 3 Casa 18 – Lago Norte

71505-730 – Brasília DF - Brasil

Resumo

Neste trabalho foi feito o projeto, a implementação e a validação de uma interface por comando de voz utilizando um dispositivo celular para uma cadeira de rodas motorizada. Para a validação do funcionamento da interface, uma vez que não esteve disponível uma cadeira de rodas motorizada, foi feito um robô móvel com sistema de controle de posição em malha fechada e sistema de detecção de objetos próximos. Para o auxílio do projeto e teste dos controladores foi feito um código de MatLab que gera um programa para um Arduino, no qual os sistemas de detecção, acionamento e controle foram implementados. Todos esses objetivos foram alcançados. Foi observada um alta latência e acurácia inconsistente, cuja maior parte vinha do reconhecimento de voz. O design da interface apresentou um fácil uso mas difícil aprendizado.
Palavras-Chave: 1.Cadeira de Rodas, 2.Automação, 3.Comando de voz, 4.Celular.

Abstract

This paper did develop, implement and validate a voice command interface for a motorized wheelchair. In order to verify the interface's functionality, since a motorized wheelchair was not available, a mobile robot with an obstacle detection system and a closed-loop control system for movement was also implemented and tested. A Matlab program was also implemented to assist in the development and programming efforts of an Arduino board in which the detection and control systems are built upon. Those goals were met. A high latency and inconsistent accuracy was observed, mostly from the voice recognition function. The designed interface was reported easy to use and difficult to learn.

Keywords: 1.Wheelchair, 2.Automation, 3.Voice Commands, 4.Cellphone.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

IBGE	Instituto Brasileiro de Geografia e Estatística
Unb	Universidade de Brasília
IFBA	Instituto Federal da Bahia
UTFPR	Universidade Tecnológica Federal do Paraná
IDE	Ambiente de Desenvolvimento Integrado
IFES	Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo
UEL	Universidade Estadual de Londrina
UFSA	Universidade Federal de Santa Catarina
MIT	Massachusets Institute of Technology
GIF	Formato de Intercâmbio Gráfico
iOS	Sistema Operacional do Iphone
RFID	Identificação por Radiofrequência
VANT	Veículo Aéreo Não-Tripulado
PCB	Placa de Circuito Impresso

Lista de Símbolos

v/V	(Volts)Diferença de Potencial Elétrico
A	(Amperes)Densidade de movimentação de cargas elétricas
Ah	(Ampere-hora)Carga Elétrica

1 Introdução

1.1 Contextualização

No Brasil, cerca de 1.3% da população apresentou em 2013 alguma deficiência física permanente [12] e em 2010, pouco mais de 4,4 milhões de brasileiros relatam grandes dificuldades em realizar atividades motoras [13]. Isso indica que há um espaço para soluções que auxiliem na mobilidade individual com pouca ou nenhuma necessidade de interação física.

Estima-se que 93% dos domicílios apresentem um telefone celular e 78% das pessoas acima de 10 anos de idade possuem o dispositivo para uso pessoal [14] e com o avanço da tecnologia em dispositivos móveis, estes são capazes de realizar uma gama de tarefas, podendo realizar a aquisição dos comandos e processamento de dados do usuário sem a necessidade de um hardware dedicado a isso.

Avanços na tecnologia de microcontroladores, como o Arduino e Raspberry Pi, permitem uma integração acessível entre um dispositivo de comando e atuadores/sensores necessários para realizar uma dada tarefa, abrindo a porta para soluções em automação de diversas atividades.

Em virtude das necessidades desses brasileiros, surge a procura por interfaces mais acessíveis para facilitar o comando dos que necessitem de uma cadeira de rodas motorizada. Já foram feitos estudos anteriores de adaptação de uma cadeira e esta foi objeto de outros 3 trabalhos na Unb, como desenvolvimento de uma interface por sopro e sucção [4], projeto do sistema de acionamento e sensoriamento [15] e de um sistema semi-autônomo com encoders e sensores de proximidade para controle de movimento [1].

Essa cadeira se encontra no laboratório do Grupo de Automação e Controle (GRACO) e possui um sistema de acionamento, medição de deslocamento e sensores para detecção de objetos mas não se encontra funcional.

Devido ao regime remoto de atividade na Universidade, não foi possível utilizar uma cadeira de rodas para validação do software, sendo assim, foi montado um robô móvel para emular o comportamento do sistema pronto na cadeira e assim poder estimar o desempenho da interface por comando de voz. A figura 1 apresenta a modelagem da cadeira e do robô que será utilizado para emulá-la.

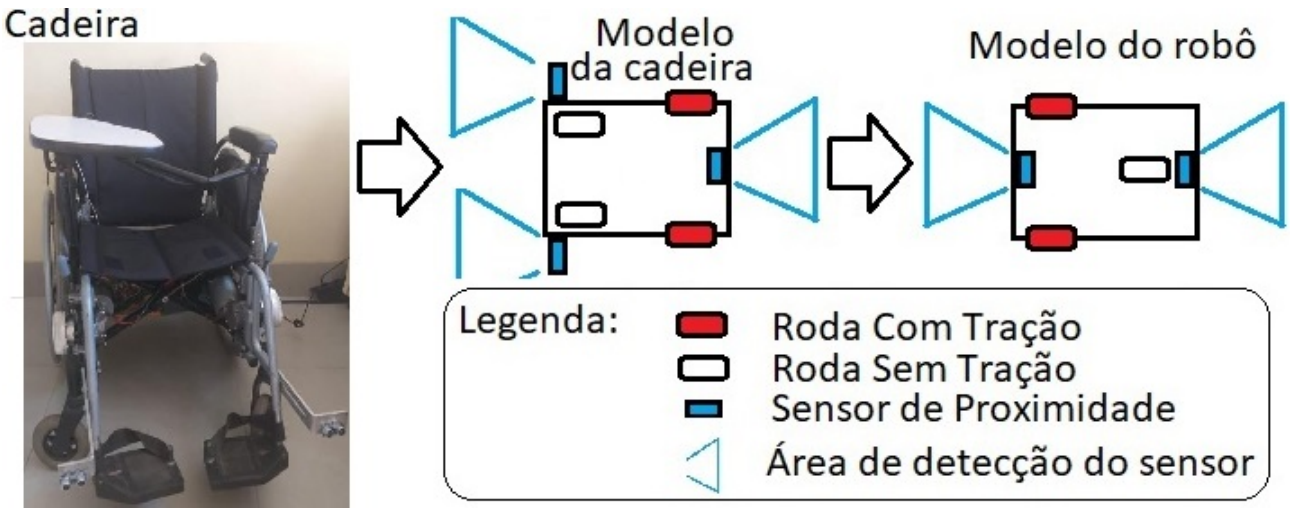


Figura 1: Similaridades entre a cadeira e o modelo utilizado, imagem da cadeira disponível em [1]

O robô montado apresenta 2 rodas tracionadas de forma independente, com um encoder para cada, de forma que o trabalho necessite o mínimo possível de adaptações no software para ser utilizado na cadeira. A relação entre a movimentação de cada roda e a do sistema como um todo entre os 2 modelos são análogas. O par de sensores dianteiros realiza a mesma função do sensor dianteiro do robô. O sensor traseiro em ambas as soluções é análogo.

Com exceção de valores numéricos, os sistemas de medição de deslocamento por encoders e os sistemas de acionamento por PWM são análogos.

1.2 Motivação

A fim de reduzir as dificuldades de locomoção de pessoas com deficiências físicas de uma forma cômoda ao usuário, deseja-se o mínimo de interações físicas e equipamentos especializados. Também é interessante que o usuário já esteja familiarizado com o equipamento e com a forma de interação.

Visto que o telefone celular pode ser utilizado para substituir o equipamento necessário para aquisição e processamento de dados, resta comparar esta solução com outras já realizadas. Caso ela seja melhor, ou no mínimo equivalente, em termos de desempenho e custo, o projeto será considerado um sucesso.

1.3 Objetivos

- Desenvolver e implementar uma interface por comando de voz para uma cadeira de rodas motorizada para que usuários com movimentação reduzida ou nula possam utilizá-la. A interface deve responder a 7 comandos, dos quais 5 são direcionais e 2 são para especificação do contexto de operação.
- Projetar e implementar um modelo físico para emular a cadeira de rodas para a verificação do software desenvolvido
- Implementar e validar um sistema de controle em malha fechada para garantir o mesmo deslocamento nas rodas, com isso obter um movimento retilíneo sem a necessidade de correções frequentes do usuário
- Realizar e testar um sistema de detecção de objetos próximos para evitar acidentes

1.4 Metodologia

1.4.1 Busca Por Estudos Similares

Foi feita uma busca por trabalhos anteriores nas áreas de controle, motores, baterias, programação em celulares, comunicação com dispositivos embarcados e cadeiras de rodas no banco de dados da Universidade da Brasília. Em seguida, buscas no Google Acadêmico nas áreas de programação em celulares e comunicação com dispositivos embarcados.

1.4.2 Planejamento do Projeto

O projeto é dividido em várias etapas:

1. Implementação da Interface com Usuário
2. Construção do Robô
3. Cálculo das Equações de Controle
4. Comunicação Interface-Robô
5. Implementação do controle no robô
6. Uso da interface para o Comandar o robô com sistema de controle

As etapas foram realizadas na ordem descrita na figura 2, onde em um mesmo nível foram feitas em paralelo, enquanto etapas em níveis distintos aguardaram as precedentes (com setas). A etapa de construção incluiu testes de verificação e funcionalidades auxiliares, como a detecção de obstáculos.

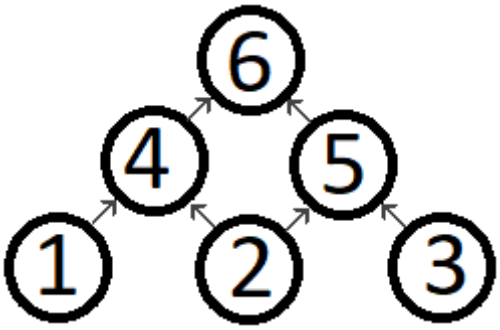


Figura 2: Diagrama da metodologia do projeto

2 Revisão Bibliográfica

Neste capítulo foi feita a pesquisa por trabalhos anteriores de temas que possam auxiliar no acionamento de motores em uma cadeira a partir de comandos de voz.

2.1 Cadeira de Rodas

Uma cadeira de rodas é, por regra geral, uma cadeira com 4 rodas, 2 grandes nas laterais e 2 menores na dianteira (figura 3). Elas costumam contar com aros nas rodas maiores para que o usuário a movimente e hastes no encosto para ser empurrada ou puxada por outras pessoas. Ela exige um elevado esforço físico e cuidado para subir ou descer rampas e terrenos irregulares.

Figura 3: Cadeira de rodas da Smart [2]

Uma outra iteração é a cadeira de rodas elétrica (figura 4), que utiliza um motor elétrico para gerar a força que move a cadeira, o que reduz drasticamente o esforço necessário do usuário porém o limita em termos de controle, visto que agora ele deve enviar comandos a uma interface em vez de diretamente movimentar à cadeira, além da necessidade de uma fonte de energia para o funcionamento do motor.

Figura 4: Cadeira de rodas elétrica da Smart [3]

Como dito anteriormente, a cadeira a ser utilizada também já foi objeto de outros 3 trabalhos de conclusão de curso (TCC) e seria adaptada com uma nova interface.

O primeiro dos 3 trabalhos que utilizaram a cadeira foi Pinto em 2016 [4], que fez uma interface por sopro e sucção e outra por joystick de queixo. O trabalho também mede a velocidade máxima do sistema presente na cadeira sem carga, de 1,56 m/s, e a velocidade média nessas mesmas condições, próxima de 0,6 m/s.

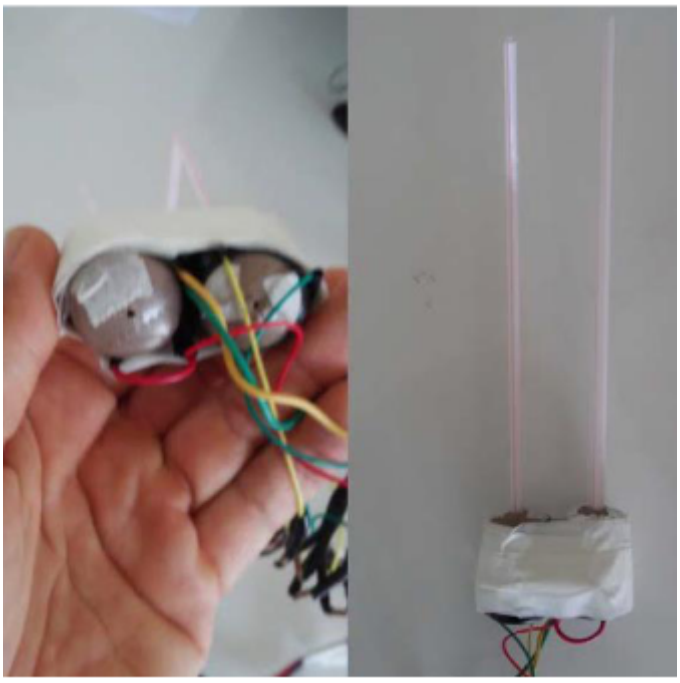


Figura 5: Interface por sopro e sucção desenvolvida por [4]

A primeira (figura 5) mede um diferencial de pressão em cada um dos 2 tubos, girando a cadeira quando um tubo é acionado e acelerando-a quando o outro é acionado. Como o nome sugere, a diferença de pressão medida pela interface deve ser causada por uma ação do usuário de empurrar ou puxar o ar do tubo.

O trabalho também descreve que o acionamento é proporcional à pressão medida, o que permite ao cadeirante controlar a velocidade em que o sistema executa os comandos. Uma desvantagem dessa solução é que sob condições normais, apenas um dos tubos pode ser acessado por vez.

O segundo dos trabalhos é de Ivo ,2017, [15] que desenvolveu o sensoriamento, o controle e o acionamento para a motorização de uma cadeira de rodas acionada por joystick. Nele está descrito o projeto, implementação e testes dos circuitos feitos.

O terceiro e mais recente dos trabalhos é de Oliveira ,2019, [1] que buscou desenvolver uma interface para portadores de tetraplegia e um sistema de navegação semi-automático. Nele, o autor apresenta uma comparação de várias interfaces com o usuário e será mostrada a tabela 1, uma versão reduzida do que foi apresentado.

Tabela 1: Tabela de comparação de interfaces com usuário, parte de outra tabela feita em [1]

Critério	Joysticks	Sopro e Sucção	Acelerômetro de cabeça	Encosto de cabeça	Movimento dos Olhos	Comando de Voz
Manobrabilidade	Alta	Média	Alta	Alta	Baixa	Baixa
Custo	Baixo	Baixo	Alto	Médio	Alto	Médio/Alto

Outros trabalhos já fizeram sistemas em áreas similares, como o visto no IFBA, 2016, [10] que motorizou uma em 2016, focando no projeto e construção do sistema de atuação. Eles descrevem com detalhe todo o processo de dimensionamento, construção e instalação se todo o sistema de acionamento, de motores até correias.

Além do processo de instalação e dimensionamento, o trabalho de 2016 [10] também levantou dados experimentais sobre a relação entre velocidade e peso levado pela cadeira motorizada utilizando 2 motores e 3 pisos diferentes. Uma síntese dos dados está apresentado nas tabelas 2 e 3 que serviram de referência para a escolha dos motores. De acordo com a norma ABNT NBR 9050, seção 4.2.2 [16], o peso máximo esperado de uma cadeira de rodas sem carga é de 60Kg, que deve se somado à carga para melhor representar a relação adequada de peso e torque.

Tabela 2: Testes de [10] com um motor de 12 volts

Piso	Carga (Kg)	Velocidade (Km/h)
Concreto Laminado	0	2,39
Concreto Laminado	45	2,10
Concreto Laminado	60	2,08
Concreto Laminado	75	0
Porcelanato	0	2,26
Porcelanato	45	1,96
Porcelanato	60	1,81
Porcelanato	75	0
PARALELEPÍPEDO	0	2,10
PARALELEPÍPEDO	45	1,02
PARALELEPÍPEDO	60	0

Tabela 3: Testes de [10] com um motor de 24 volts

Piso	Carga (Kg)	Velocidade (Km/h)
Concreto Laminado	0	2,36
Concreto Laminado	45	2,17
Concreto Laminado	60	2,02
Concreto Laminado	75	1,94
Porcelanato	0	2,18
Porcelanato	45	2,07
Porcelanato	60	1,99
Porcelanato	75	1,90
PARALELEPÍPEDO	0	2,09
PARALELEPÍPEDO	45	1,98
PARALELEPÍPEDO	60	1,85
PARALELEPÍPEDO	75	1,73

Visto que os melhores resultados foram obtidos pelos motores com maior torque, para a montagem do robô que irá representar a cadeira, foram utilizados os motores de maior torque disponíveis para tensões de 6v. Para alimentação destes, foram utilizadas baterias recarregáveis de 9v pois elas, em comparação com 4 pilhas de 6v, ocupam menos espaço e são mais leves. Para garantir um fornecimento adequado de corrente, foram utilizadas 3 em paralelo.

2.2 Interfaces com o Usuário

Todo sistema semi-autônomo com uma função recebe de uma pessoa um determinado comando. O trabalho da interface é fornecer à pessoa uma forma de enviar esse comando. Algumas das interfaces mais simples, como teclados, esperam que o usuário pressione um botão, e outras, como joysticks, utilizam a inclinação de uma haste.

Sobre o assunto, foram observadas apenas interfaces físicas, com as quais o usuário interage sem intermédio de um aparelho específico, como utilizar um mouse para clicar em ícones em uma tela, mas touchscreens por exemplo ainda são válidas. Os exemplos também não foram restritos a implementações conectadas a sistemas semi-autônomos.

A fim de comparar as diversas interfaces, foi feita uma análise qualitativa de 3 características: a facilidade de uso, que representa o quão fácil é utilizar devidamente a interface após o aprendizado, a intuitividade, que descreve o quão fácil é aprender a utilizar a interface e o desempenho, que leva em consideração desde o tempo de resposta como a taxa de falha típica da mesma.

Um teclado (figura 6), similar a um controle remoto, apresenta uma grande quantidade de botões, sendo pouco intuitivo mas de uso bastante simples. Por não necessitar de quase nenhum processamento para interpretação de comandos, esse tipo de interface costuma responder sem atraso perceptível e, se em boas condições, taxa de falha desprezível.

Figura 6: Parte de um teclado (Fonte: Autor)

Com os Smartphones, surgiu a tecnologia TouchScreen, que envia um comando ao aparelho dependendo de onde a tela do mesmo for tocada pelo usuário. Uma solução simples, intuitiva e eficaz para situações com pouca necessidade de comandos distintos. Atividades como digitar um texto ou interferências como o usuário usar luvas afetam drasticamente seu desempenho. Um exemplo pode ser visto na figura 7.

Figura 7: Exemplo de interface touchscreen (Fonte: Autor)

Outra possível interface é o joystick (figura 8). Ele consiste de uma haste que rotaciona em torno de um ponto enquanto envia 2 sinais analógicos proporcionais à rotação e um terceiro, caso seja pressionado. Isso permite a ele também ser simples, intuitivo e eficaz mas extremamente limitado no número de comandos distintos que podem ser entregues a cada instante.

Para estender a lista de ações basta programar um ou mais dos canais de dados para mudança de contexto, o que reduz levemente a simplicidade e intuitividade para cada mudança de contexto presente mas isso não é relevante para a comparação sendo feita.

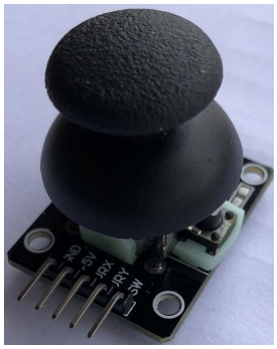


Figura 8: Um Joystick (Foto por: Autor)

É interessante também observar as interfaces que envolvem o processamento de áudio, como reconhecimento de voz [6], processamento de imagens, como rastreamento de olhos [17], rastreamento da cabeça [18] ou reconhecimento de movimentos em geral, dependendo da implementação, podem ser fáceis de utilizar e de aprender, mas costumam ser mais lentos, apresentar uma taxa considerável de falhas e de forma similar ao joystick, são limitados no número de funções

distintas que podem exercer. Além disso, elas exigem uma ação mais longa do usuário, o que reduz ainda mais a sua velocidade de operação.

Um exemplo de interface que utiliza comando de voz está representado na figura 9. Ela apresenta os possíveis comandos na tela e, com uma mudança de cores, indica ao usuário qual comando está sendo executado.

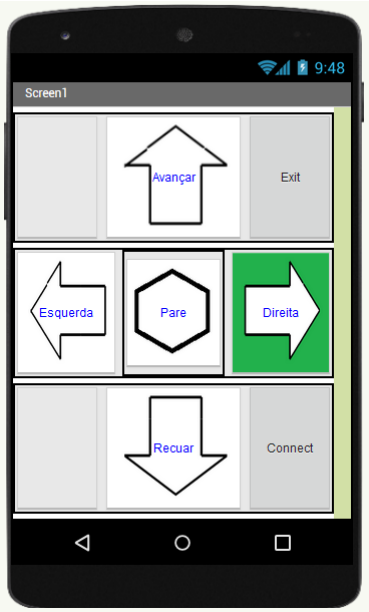


Figura 9: Exemplo de interface por comando de voz (Fonte: Autor)

O objetivo principal de uma interface é ser capaz de operar o sistema a ele acoplado, mesmo que isso custe uma ou mais de suas características desejáveis. A interface de um piloto com um avião por exemplo, é virtualmente impossível de se simplificar, pois ele deve ser capaz de receber uma enorme quantidade de comandos distintos e pode não ser capaz de guiar o avião ao ser reduzida.

Também é importante pontuar que as características observadas variam com a implementação de uma determinada interface, dessa forma, 2 implementações diferentes com o mesmo meio de interação podem ser distintas uma da outra. Comparando os exemplos das figuras 7 e 10, a segunda é mais intuitiva mesmo que ambas usem o mesmo meio físico.

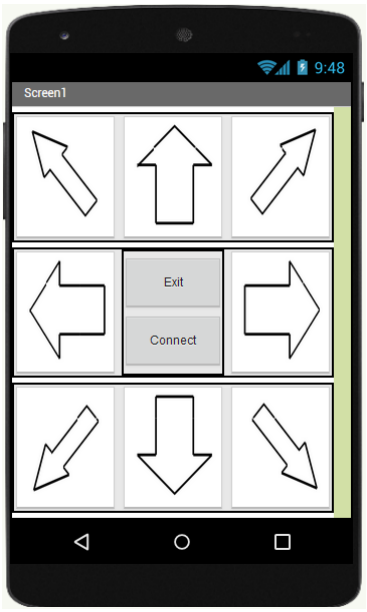


Figura 10: Outro exemplo de interface por touchscreen

Outra consideração a ser feita é sobre as limitações impostas pelo contexto na qual a solução é aplicada, seja devido ao usuário, aplicação ou mesmo o ambiente. Interfaces que envolvem processamento de imagem não operam bem diante de variações de luminosidade e/ou ambientes movimentados, interfaces por reconhecimento de voz não operam corretamente mediante ruído e interfaces que exigem movimento não são adequadas para usuários com perdas motoras.

Quanto ao design da interface, o trabalho [19] descreve um tamanho mínimo para o tamanho da fonte e dos botões para uma implementação em um celular, além de práticas a serem adotadas, como posicionar comandos próximos às variáveis afetadas por eles.

Também há diretrizes quanto ao processo de desenvolvimento em si. O trabalho [20] diz que o design deve ser feito junto de outras pessoas, se possível futuros usuários, da interface sendo feita, sendo revisada conforme comentários e críticas, um processo de contínuas melhoras e iterações.

Deseja-se permitir ao usuário enviar o máximo de comandos possível, o que traz a necessidade de informá-lo sobre a execução dos mesmos e, portanto, eleva a quantidade mínima de informação necessária para operação. Em outras palavras, reduz a intuitividade para obter um ganho de funcionalidade e/ou simplicidade.

Por outro lado, um usuário sempre irá preferir o que consegue entender, podendo escolher uma interface mais intuitiva mesmo que inferior em outras qualidades. Há ainda o problema do usuário, onde um design específico pode ser muito complexo para um e muito simples para

outro. Um exemplo disso é visto em jogos, na qual a mesma interface é simples para um usuário que esteja acostumado com videogames e incompreensível para outros públicos.

Sendo assim, é importante determinar a área de conhecimento esperado do público-alvo da interface. Esses conceitos sempre terão um grau de subjetividade, mas há formas quantificar o desempenho de um design específico. O livro [11] descreve formas de avaliá-lo em 5 pontos, apresentados na tabela 4.

Tabela 4: Parâmetros de análise para interfaces com usuário conforme [11]

Eficácia	Descreve se a interface executou a tarefa corretamente ou não
Eficiência	Número de ações e/ou tempo necessário para concluir uma atividade
Satisfação	Medida numérica de satisfação
Facilidade de aprender	Número de erros observados e/ou tempo extra gasto por um novo usuário para determinada tarefa
Tolerância a erro	Acurácia do resultado em comparação com tempo gasto em falhas

Onde a eficiência é análoga à simplicidade e facilidade de aprender é análoga à intuitividade. A interface desenvolvida neste trabalho utiliza 7 comandos:

- Avançar/Recuar para movimentações retilíneas para frente e para trás, respectivamente.
- Esquerda/Direita para as respectivas rotações
- Pare para interromper quaisquer movimentações
- Dorme para indicar ao sistema de reconhecimento que ele deve ignorar todos os comandos exceto ’Aurora’
- Aurora para indicar ao sistema de reconhecimento que ele deve voltar ao modo de operação normal

A escolha dos comandos foi feita com o intuito de tornar o reconhecimento mais simples, com palavras únicas foneticamente distintas entre si em vez de frases. Os 5 comandos principais de movimento são o necessário para manter uma experiência agradável ao usuário que ainda o permita se mover em quaisquer direção. Os 2 comandos auxiliares permitem que o usuário dialogue sem ter de desligar o sistema.

Os botões são grandes para facilitar os toques e a navegação, e no caso da tela de operação do reconhecimento de voz, informa o último comando recebido. Como pontos negativos, ela não informa os possíveis comandos em sua tela.

Exemplos de interface por comando de voz desenvolvidas para cadeiras de rodas são apresentados nas figuras 11 e 12.

Figura 11: Interface por comando de voz de [5]

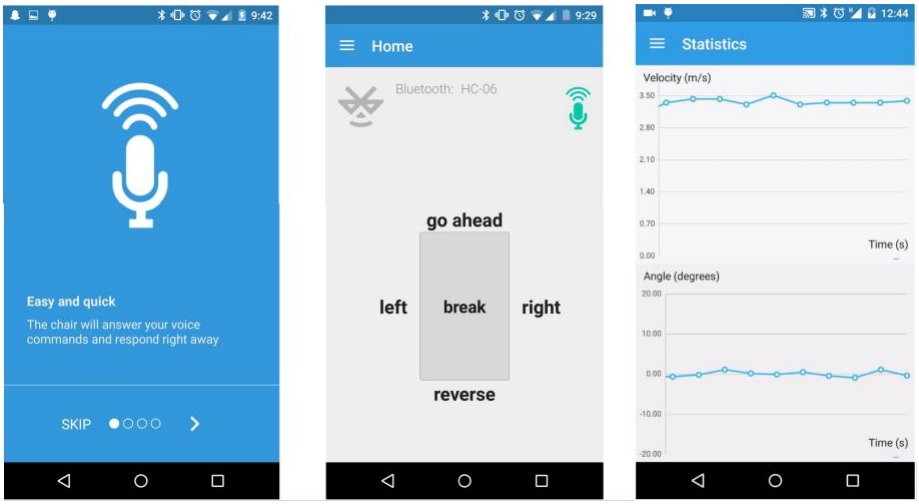


Figura 12: Interface por comando de voz de [6]

2.3 Reconhecimento de voz

O reconhecimento de voz se refere à capacidade de um dispositivo de identificar as palavras em uma fala. Ele exige um hardware capaz de captar sons e um software capaz de filtrar e comparar grandes quantidades de dados em um intervalo moderado de tempo, sendo pouco recomendado em aplicações sensíveis ao tempo.

A tecnologia permite que o usuário se comunique com o dispositivo como se estivesse conversando, uma forma intuitiva, cômoda e transparente. Porém, ela costuma apresentar uma taxa de falha, atraso e vulnerabilidade a ruídos bem maior que outras interfaces, além de exigir um hardware específico e relativamente potente.

Uma das fontes de falhas decorre das várias diferentes formas de pronunciar palavras, mesmo em um idioma fixo, seja por causa de um sotaque ou devido até mesmo divergências de qual seria a pronúncia correta de uma palavra (como a pronúncia correta[21] de GIF[22]).

Uma das formas de amenizar este problema é tentar identificar o contexto das possíveis palavras sendo ditas e escolher a mais provável, o que exige um esforço bem maior de programação e leva mais tempo durante a execução, outra é utilizar de um software que se adapta aos padrões de fala do usuário, o que além do esforço a mais do programador, exige um tempo e só funciona para aquele mesmo usuário. Outra vem da obtenção do áudio da fala, que necessita de um pré-processamento antes de ser comparado a um banco de dados e levantados os graus de proximidade com o devido comando. O trabalho em [23] utilizou a transformada rápida de fourier para obter uma análise espectral do sinal de voz a fim de compará-lo. A técnica também permite remover parte do ruído de um determinado sinal para conseguir uma maior acurácia em ambientes barulhentos.

Foi feita uma busca por outros trabalhos que utilizaram alguma forma de reconhecimento de voz. Dentre os resultados encontrados estão Alvez e Goulart, 2014 [24], que fez 2 aplicativos no Android Studio para celular que operava por reconhecimento de voz, uma calculadora e um aplicativo que executa funções de outros aplicativos. Este ainda menciona que o reconhecimento no Android é feito online mas pode ser feito offline com acurácia reduzida.

Um outro trabalho foi Azevedo,Denadai,Lima, 2016 [5] apresentado no Congresso Brasileiro de Automática que fez uma interface por comandos de voz para uma cadeira do rodas motorizada.

Esse segundo trabalho utilizou a plataforma appinventor para realização e levantou diversos dados sobre o desempenho, incluindo a taxa de acerto e medição de atraso da implementação do reconhecimento de voz. O trabalho também mediu o desempenho do processamento no Arduino e na comunicação bluetooth.

2.4 Celular e Smartphone

Os celulares foram originalmente inventados como telefones portáteis para permitir que 2 pessoas em locais diferentes conversem, evoluindo ao longo de 25 anos para dispositivos capazes de realizar uma enorme gama de funções [25]. De câmera a roteador, os celulares da atualidade são pouco utilizados para a comunicação por voz, sendo mais próximos de um computador portátil do que um telefone.

Os Smartphones, que são celulares com várias funções, são também programáveis, podendo receber programas chamados aplicativos. Similar a computadores eles também têm um sistema operacional, sendo os 2 sistemas mais comuns da atualidade os sistemas Android e o iOS.

O sistema operacional Android possui IDEs livres, permitindo que o usuário faça os seus próprios aplicativos sem a necessidade de validá-lo, desde que tenha as ferramentas e o conhecimento em programação.

Na busca por trabalhos que desenvolveram programas para celulares, foi encontrado este trabalho de 2017 também publicado na Unb [26], continuação de um trabalho do ano anterior [7] que buscou implementar um veículo aéreo não-tripulado ou VANT controlado por um aplicativo de celular. Com uma proposta inicial de utilizar o acelerômetro interno do aparelho como forma de receber comandos do usuário, enviando-o a um por bluetooth a um Arduino, este enviaria o comando ao VANT por radiofrequência. O processo está resumido na figura 13.

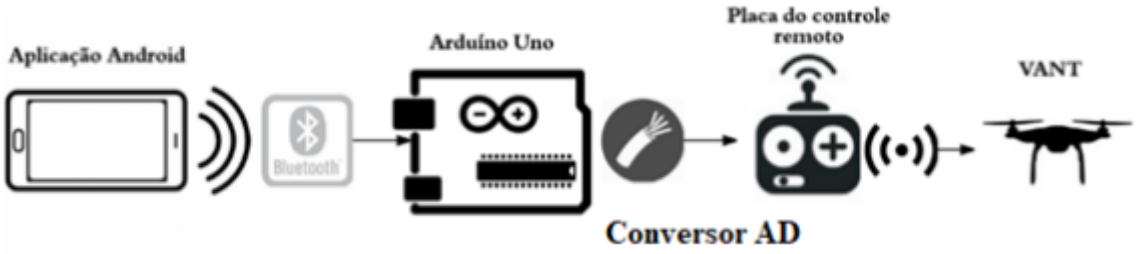


Figura 13: Proposta do trabalho de 2016[7]

Similar ao TCC de 2014 mencionado na seção anterior [24], os trabalhos do VANT utilizaram o Android Studio para programar diretamente no telefone.

A programação pode ser feita de diversa maneiras, como ReactNative, Expo, Android Studio ou pelo Appinventor. A mais simples de se programar e que será utilizada é o Appinventor [27]. Nele, a programação é feita por meio de blocos prontos para a maioria das atividades, da captação da fala até o envio dos comandos via bluetooth. A realização do programa bem como

algumas de suas funções são processadas de forma remota, o que caracteriza uma computação em nuvem.

Dessa forma, é possível desenvolver todo o programa do celular rapidamente e sem dominar as linguagens de programação normalmente necessárias para realização do aplicativo. Essa solução não é viável para aplicações de baixo nível, como processamento de imagem, mas é excelente para aplicações que utilizem sensores e/ou atuadores já existentes no aparelho. Além disso, em funções mais complexas de análise de dados o programa irá apresentar resultados similares independente do hardware do aparelho, visto que o processamento é feito em um servidor da Google caso haja disponibilidade de internet. Por outro lado, isso adiciona um atraso ao processamento que pode ser relevante em aplicações sensíveis ao tempo. É dito em [28] que a implementação utiliza aprendizado de máquina para realizar suas traduções.

Embora a função seja configurável, conforme descrito em [28], o uso do appinventor permite dois modos de uso, um que exige toque e preenche a tela do celular para apresentar resultados e outro que não exige toque mas não indica claramente nem o resultado nem quando está ativo. Devido ao requisito de toque, a primeira configuração foi descartada.

Visto que os aparelhos Android costumam vir com comunicação bluetooth integrada e trabalhos anteriores (como [7], [5] e [24]) mostraram que essa solução é viável para enviar comandos vindos de um celular portanto foi o caminho adotado para este projeto.

2.5 Comunicação sem fio

A comunicação sem fio se dá pelo envio de dados de um dispositivo para outro sem o uso de um cabo físico. Existem centenas de tecnologias, como RFID[29], Wi-fi[30] e Bluetooth[31] e protocolos para gerir essa comunicação.[32]

Cada tecnologia tem uma área em que é melhor aproveitada:

- RFID: Poucos dados a Curtas distâncias (ex: inventário)
- Bluetooth: Muitos dados a Curtas distâncias (ex: monitoramento local)
- Wifi: Muitos dados a Longas distâncias (ex: videoconferência)

Além da distância e quantidade de informações, cada tecnologia apresenta um tempo necessário para transmissão diferente, onde as que enviam menos dados (RFID) e/ou têm menos enlaces envolvidos na comunicação (Bluetooth) tendem a responder muito mais rápido.

2.6 Sensores de proximidade

Há uma gama de sensores que podem ser utilizados para detectar proximidade de outros objetos sem o toque, como o capacitivo [33], o indutivo [34], o óptico [35] e o ultrassom [36]. Sensores capacitivos e ópticos necessitam de um anteparo e só detectam objetos entre o sensor e o anteparo, sensores indutivos só detectam objetos metálicos e o ultrassom tem uma velocidade de operação limitada.

Como neste trabalho deve-se detectar quaisquer objetos sem o uso de nenhum anteparo, apenas o ultrassom foi abordado em maiores detalhes. Ele envia ondas sonoras de alta frequência e aguarda o retorno dessa onda, e sabendo que a velocidade de propagação do som no ar é aproximadamente constante, o sensor utiliza o tempo entre o envio e o retorno para calcular a distância a um objeto.

O limite de velocidade de operação vêm do fato de que o dispositivo não diferencia uma onda atual de outra anterior, sendo assim ele deve aguardar um limite máximo de tempo para a primeira onda retornar, caso contrário irá registrar objetos mais próximos do que o esperado.

O sensor também não pode enviar e ouvir a onda ao mesmo tempo, logo, caso uma onda retorne no momento em que ele envia a próxima onda ela não será detectada. Caso o objeto esteja dentro de um alcance mínimo (zona morta) a onda retornará antes de que o dispositivo esteja pronto para registrá-la e portanto também não será detectado.

O documento [37] indica que os valores máximo e mínimo de alcance são, respectivamente, 2cm e 4m, mas esses valores não foram os observados na implementação, que estavam mais próximos de 5cm a 1m.

Mais um fator limitante é que o sensor não ser capaz de distinguir tipos diferentes de objetos, sendo assim, se for obstruído, tanto por sujeira ou acidentalmente pelo usuário, o funcionamento dele será prejudicado.

2.7 Controle

Estabilidade

O conceito de estabilidade BIBO (Bounded Input Bounded Output) define que um sistema é dito estável se para uma entrada de amplitude limitada a saída também apresenta amplitude limitada. Neste conceito, pode-se determinar a estabilidade de um determinado sistema observando apenas sua função de transferência [38], não sendo necessário verificá-lo para várias entradas

limitadas distintas.

Controle em malha fechada

Controlar uma grandeza física é ditar seu comportamento ao longo do tempo, normalmente seguindo a um sinal chamado de referência. Isso pode ser feito de 2 formas distintas: malha aberta, na qual a grandeza não é observada, e malha fechada na qual ela é (exemplo na figura 14). A segunda é mais robusta, mantendo um comportamento adequado diante de variações no sistema para a qual foi projetado, independente de tais variações serem temporárias ou permanentes, mesmo que exija um ou mais sensores extras.

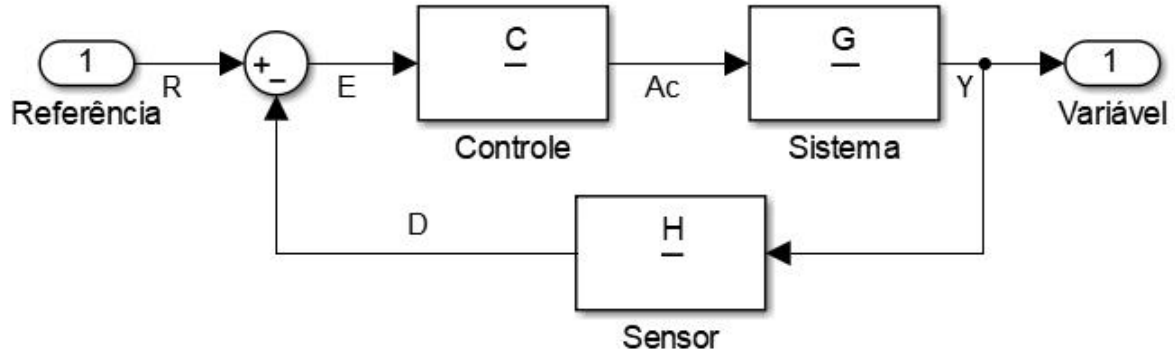


Figura 14: Representação em diagrama de blocos de uma realimentação

Existem diversa abordagens para o projeto de um sistema de controle, cada uma com suas vantagens e desvantagens.

Regime permanente e regime transitório

Na análise de um sistema, distingue-se também a resposta transitória, que seria um comportamento temporário diante de variações da referência no domínio da frequência, e um regime permanente (ilustrado na figura 15), que seria o comportamento depois de um dado tempo sem que hajam variações no sinal alvo. Utiliza-se a variação no domínio da frequência já que senóides apresentam variações no domínio do tempo mas não na frequência.

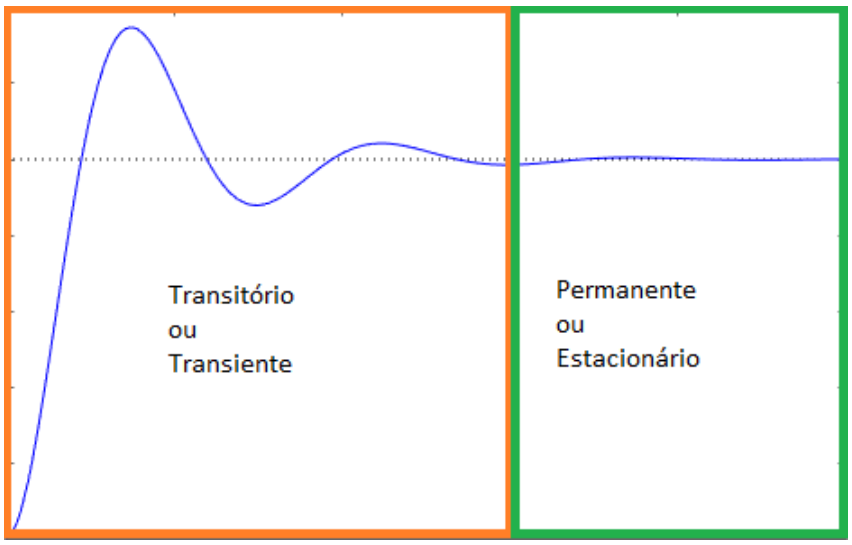


Figura 15: Exemplo de Resposta ao degrau de um sistema qualquer

Especificações de projeto

Para avaliar o desempenho de um sistema de controle em malha fechada, define-se um conjunto de parâmetros. A maioria delas utiliza um degrau de referência para medição e a aproximação se segunda ordem sem zeros para projeto.

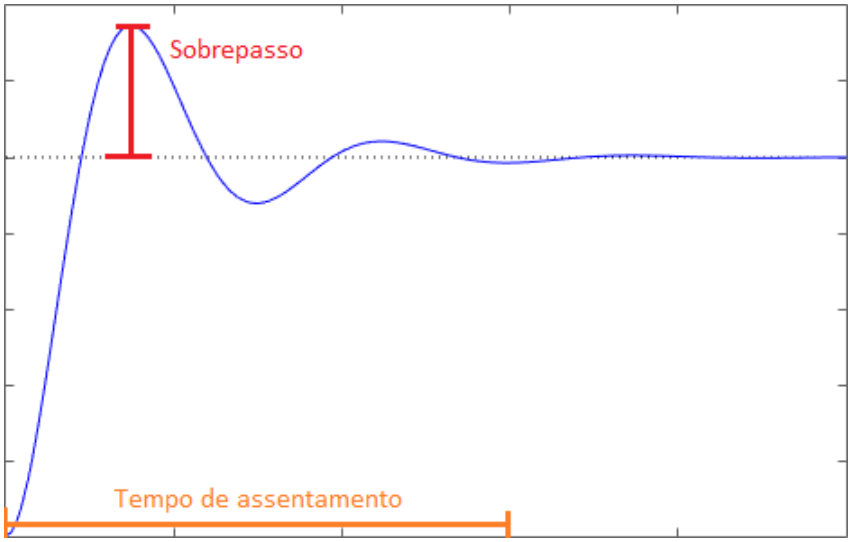


Figura 16: Especificações de Resposta ao degrau de um sistema qualquer

- Sobrepasso percentual

  O quanto a variável controlada ultrapassa o seu valor final em porcentagem deste mesmo valor, conforme a figura 16.

- Tempo de Assentamento

Tempo necessário para a variável controlada atingir uma dada porcentagem de seu valor final (exemplo na figura 16), mesmo que apresente um erro em relação ao desejado.

- Erro em regime permanente

  Valor percentual de erro entre o sinal alvo e o sinal controlado após atingir o regime permanente. Como este depende da entrada, este valor acompanha o tipo do sistema, descrito a seguir. Para estimá-lo, o sistema não pode ser instável e será utilizado o teorema do valor final.

- Tipo de um sistema

  Valor que indica qual o grau de entrada que pode ser acompanhada sem erro em regime permanente, onde um sistema de tipo n acompanha um sinal de referência com grau n-1 no tempo, supondo H = 1. Cada integrador no sistema ou em série com ele aumenta seu tipo em 1.

Abordagens Paramétricas de Projeto

São técnicas que necessitam dos parâmetros do sistema a ser controlado, exigindo uma etapa de identificação e modelagem antes de serem realizadas.

- Proporcional

  Controle que multiplica o erro entre a referência e a variável controlada por um dado valor e aplica isso ao sistema. Não modifica o tipo do sistema nem adiciona pólos/zeros.

- Avanço de fase

  Controle que insere um pólo e um zero no sistema, com o zero à direita do pólo. Utilizado para modificar o regime transitório em malha fechada. Não modifica o tipo do sistema.

- Atraso de fase

  Insere um pólo e um zero no sistema mas com o pólo à direita do zero. Utilizado para reduzir o erro em regime permanente sem modificar muito o transitório. Não modifica o tipo do sistema.

- Notch ou Avanço-Atraso

  Combina os 2 métodos anteriores a fim de melhorar tanto o regime transitório quanto o permanente. No total, adiciona 2 pólos e 2 zeros, onde o projeto do avanço é feito antes e os pólos/zeros de um não devem ficar muito próximos do outro.

- Proporcional-Integral

  Insere um integrador e um zero. Usado para zerar o erro em regime permanente.

- Proporcional-Integral-Derivativo

  Insere um integrador e dois zeros. Usado para zerar o erro em regime permanente e fornecer avanços de fase maiores. Implementado como uma combinação de Proporcional-Integral seguido de um Avanço, pois não é realizável em sua forma ideal.

Controlador de resposta Deadbeat

Um controlador com projeto simples (figura 17) mas com desempenho vulnerável a saturação e variações da função de transferência. Feito sob medida para um determinado desempenho e sistema, ele difere dos anteriores por ser realizável apenas no domínio discreto. O projeto consiste de cancelar a função de transferência do sistema, ele se torna complexo apenas quando há pólos ou zeros instáveis ou desejados no sistema sendo controlado. Caso contrário, o denominador só deve apresentar os integradores necessários para acompanhar a referência esperada.

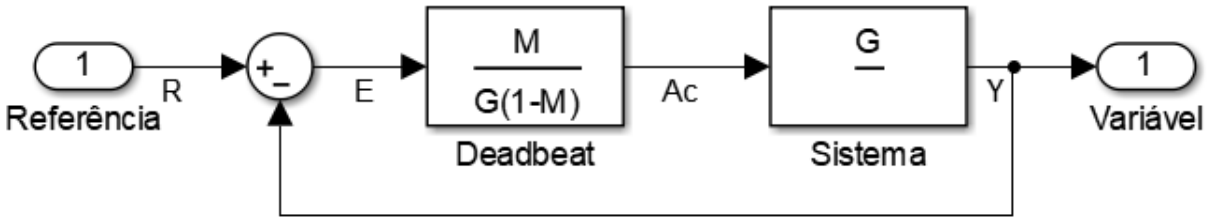


Figura 17: Exemplo de um controlador com resposta DeadBeat

2.8 Comunicação celular-acionamento

O próximo passo é encontrar uma forma de fazer a comunicação do celular com a plataforma que vai acionar os motores. Primeiramente, deve-se escolher entre uma comunicação com fio ou sem fio. Devido à variação de possíveis encaixes com o celular, limitações de alcance e até mesmo possível mal contato, é mais eficaz optar por fazer a comunicação sem fio.

Na escolha da cominucação sem fio, pode-se optar tanto pela tecnologia wi-fi como pela tecnologia bluetooth. Visto que a tecnologia wifi convencional requer um roteador e uma série de protocolos enquanto o bluetooth requer apenas um emparelhamento entre os dispositivos, será utilizado o bluetooth.

Outros trabalhos também optaram pelo Bluetooth, como os trabalhos do VANT mencionados

anteriormente [26] e [7] e o de 2014 que transmitia comandos de voz de um Android para um Arduino [24]. Este segundo trabalho ainda apresenta a existência de 3 classes de bluetooth:

- Classe 1 Com até 100 metros de alcance.
- Classe 2 Com até 10 metros de alcance.
- Classe 3 Com até 1 metro de alcance.

Visto que é esperando que uma pessoa esteja sentada na cadeira pra operá-la, não há necessidade de utilizar um módulo bluetooth classe 1, e para operar confortavelmente dentro do alcance do dispositivo, será escolhida a classe 2, se possível.

O protocolo também permite um sistema mestre-escravo, em que um dispositivo mestre gerencia qual outro dispositivo escravo estará ativo. Nessa arquitetura, uma rede possui 1 único mestre mas pode conectar vários escravos.

2.9 Acionamento

A plataforma escolhida para receber os comandos do celular e traduzi-los em um comando aos motores será o Arduino, como nos trabalhos anteriores que também trabalharam na cadeira [1] e [4], devido ao fácil acesso, simples uso e os vários módulos existentes para diversas funcionalidades.

A interface de desenvolvimento é open-source e bem simples, similar à linguagem de programação C. Os programas resultantes consistem de uma etapa de 'Setup' que é executada uma única vez e uma etapa de 'Loop' que se repete enquanto o programa não for sobrescrito ou apagado.

Quanto ao hardware escolhido, foi utilizado o Arduino Uno R3, que de acordo com o site filipeflop, é uma versão abundante e compatível com grande parte dos módulos para Arduino, os quais serão necessários para que o R3 realize quaisquer ações. Para a realização do projeto, deverá-se obter um módulo para a comunicação bluetooth e outro para o acionamento dos motores.

O primeiro módulo relevante é o módulo SPP-C JDY-31, um módulo bluetooth classe 2 que irá operar como escravo enquanto o programa mestre estará no celular. Este módulo será responsável por obter os comandos enviados do celular e entregues à placa Arduino.

O segundo módulo relevante é a ponte H, que irá permitir um acionamento bidirecional dos motores, sendo capaz de chavear até 35 Volts. A escolha das baterias foi de

O sensoriamento para detecção de objetos próximos deve ter um alcance longo o suficiente para frear a cadeira em velocidade máxima antes que ela percorra o alcance e toque o objeto detectado. De acordo com o trabalho apresentado no CBA2016 [5], o tempo de resposta esperado do Appinventor somado ao do Arduino é inferior a 2 segundos e de acordo com o trabalho [4], a velocidade esperada da cadeira é inferior a 2 metros por segundo.

Sendo assim, assumindo uma frenagem instantânea, o usuário pode interromper o movimento ao observar um obstáculo a pelo menos 4 metros de distância. Caso o objeto esteja mais próximo, o sensoriamento deve intervir então um alcance de 4 metros para os sensores de proximidade é razoável.

Por último, os sensores escolhidos para a medição de velocidade/posição foram ópticos, para a montagem de encoders para as rodas, similares em funcionalidade aos instalados na cadeira do GRACO.

2.10 Medição de carga da bateria

Como a cadeira irá se movimentar por médias a longas distâncias, ela necessita de uma bateria e uma forma de indicar ao usuário a autonomia restante. Trabalhos como Felipe A. A. B. em 2014 [39] e Márcio B. M. em 2006 [40] indicam que a relação entre tensão e corrente de uma bateria varia com a carga restante, e assumindo uma massa e eficiência para o sistema de acionamento, pode-se estimar a distância que a cadeira pode percorrer sem recarregar.

Para estimar a carga sem medição de corrente, pode-se medir a tensão fornecida. Em Orrico, M.V.M., 2013 [8], é indicado que a tensão cai rapidamente quando a carga está abaixo de 10% conforme a figura 18.

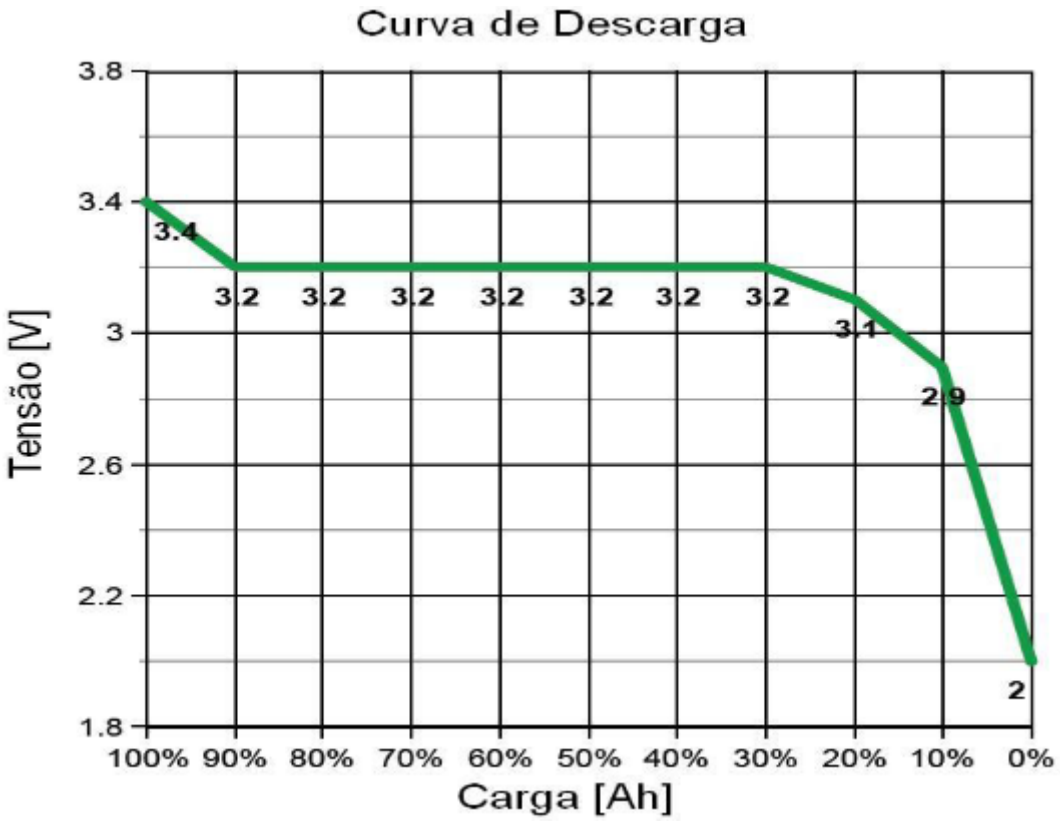


Figura 18: Exemplo de curva de descarga para uma bateria de 3.2V, disponível em [8]

3 Desenvolvimento

Devido ao regime remoto, não foi possível utilizar a cadeira de rodas do laboratório para validação do software, sendo assim, foi montado um robô móvel para simular o comportamento do sistema pronto na cadeira e assim poder estimar o desempenho da interface por comando de voz.

A lista de materiais utilizados para a montagem do robô modelo é apresentada a seguir.

3.1 Montagem do robô

3.1.1 Lista de Materiais

- 2 placas de acrílico 260mmx150mmx3mm
- 1 placa de acrílico 190mmx150mmx3mm
- 1 Arduino Uno R3
- Módulo bluetooth JDY-31-SPP
- Conectores Wago 221 (20A 300V)
- Jumpers macho-macho e macho-fêmea para protoboard
- 4 baterias 9V 250 mAh
- 2 motores JGA25370 6V 130RPM 3.6kgf
- 2 Rodas de 65mm de diâmetro
- Ponte H L298N
- Fita isolante
- 2 sensores de proximidade por ultrassom HC-SR04
- 2 sensores ópticos para os encoder
- 1 roda livre

3.1.2 Montagem

A estrutura consiste da combinação de 2 kits (figura 19), o que resultou em 3 andares separados por placas de acrílico. A montagem geral pode ser vista na figura 20.

Figura 19: Kits da Estrutura do robô, imagem e kits da Huinfinito[9]

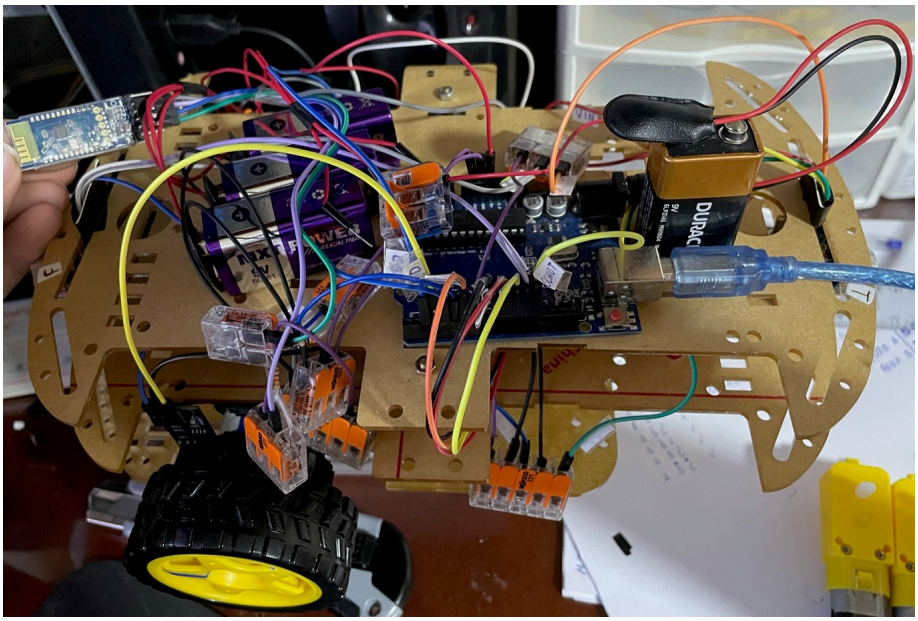


Figura 20: Montagem geral vista de cima

As baterias utilizadas na montagem da figura 20 não são as mesmas da lista de materiais e foram trocadas após descarregarem. As baterias recarregáveis utilizadas só foram obtidas após vários meses devido à disponibilidade de um carregador.

O primeiro andar da montagem (figura 21) contém apenas os motores, fixados com uma combinação de peças sobressalentes das estruturas, fita isolante. Conectados aos motores estão os sensores ópticos fixados por fita isolante e elásticos de costura. As rodas, que são de uma

montagem anterior, têm um eixo de tamanho diferente dos motores utilizados e foram encaixadas com o auxílio de fita isolante para preencher a diferença entre os eixos e assim reduzir oscilações.

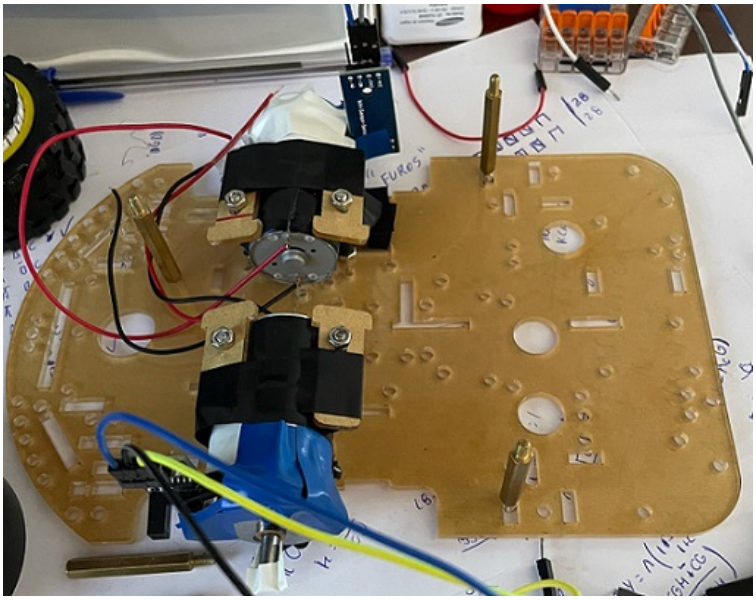


Figura 21: Primeiro andar do robô montado (Foto de: Autor)

Essa solução se provou eficaz apenas para curtos períodos de tempo pois o desgaste causado pelo uso desgasta a fita e a roda fica bamba, saindo do eixo após longos períodos de uso. Devido à falta de acesso a meios melhores, como impressão 3d, não foi possível resolver esse problema de maneira satisfatória.

O segundo andar (figura 22) contém a ponte H e os sensores de proximidade. A roda livre (figura 23) para apoio traseiro também se encontra fixada no segundo nível do robô.

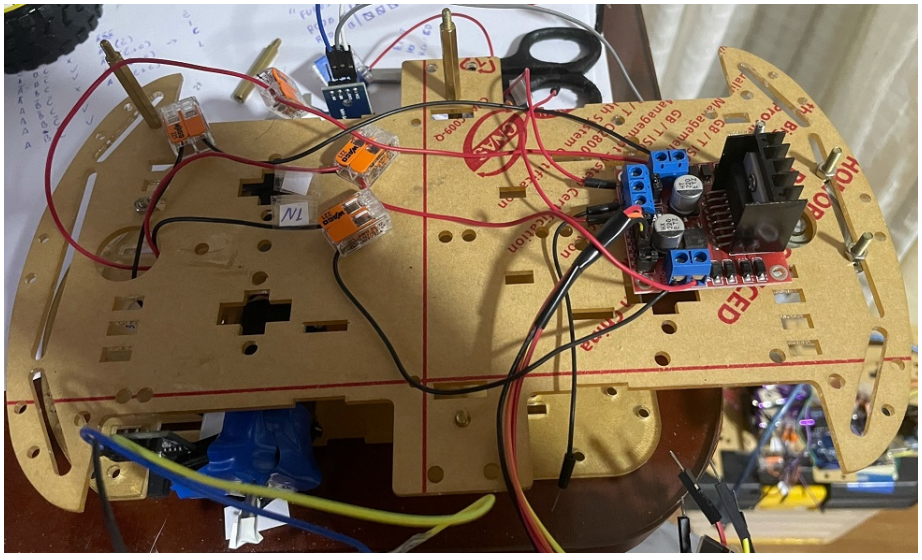


Figura 22: Segundo andar do robô montado (Foto de: Autor)

Figura 23: Imagem da roda livre (Foto de: Autor)

O terceiro andar (figura 20) leva as baterias, o arduino, e o módulo bluetooth. Das 4 baterias, delas 3 operam em paralelo para fornecer aos motores a corrente necessária e a quarta é exclusiva para alimentação do Arduino durante a operação. Todas fornecem 9 volts de tensão nominal, são recarregáveis e apresentam uma carga de 250mAh.

Na montagem, apenas os acrílicos, a ponte H e a roda livre são fixas por parafusos, o resto dos materiais é mantido no lugar por atrito ou está solto no robô. As conexões dos fios são feitas com conectores Wago (figura 24), que embora sejam fabricados para uso em instalações elétricas, permitiram que o robô fosse montado sem PCBs nem protoboards, além de serem removíveis.

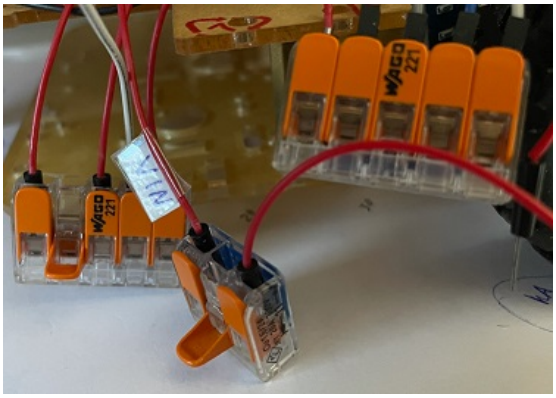


Figura 24: Imagem dos conectores Wago utilizados (Foto de: Autor)

Os sensores de proximidade foram posicionados um na frente e um atrás do robô, encaixados em furos presentes nas peças de acrílico e fixados com auxílio de fita isolante. A figura 25 apresenta uma visão geral enquanto as figuras 26 e 27 mostram os sensores dianteiro e traseiro, respectivamente.

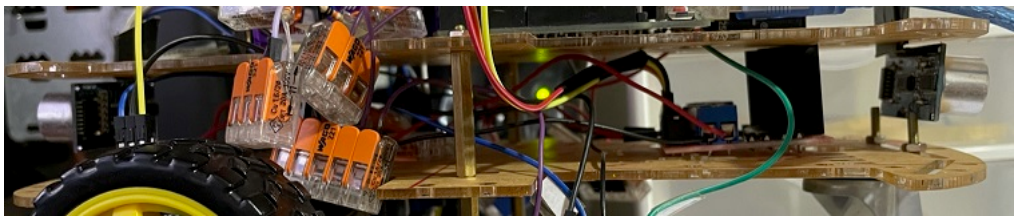


Figura 25: Fixação dos sensores no robô montado (Foto de: Autor)

Figura 26: Sensor Dianteiro (Foto de: Autor)

Figura 27: Sensor Traseiro (Foto de: Autor)

Os discos dos encoders foram feitos à mão com fita isolante preta, papéis e partes de garrafas pet, cobertos de fita adesiva transparente e posicionados nas rodas. Os sensores óticos presos aos motores podem medir a distância percorrida pelo robô contando o número de vezes que o feixe é interrompido pelas fitas presas à roda. A razão da fita transparente é dar uma vida útil maior ao encoder, que em versões anteriores, perdia trechos da fita isolante conforme a roda girava.

Figura 28: Encoder improvisado

3.1.3 Outros problemas da montagem

Um dos problemas encontrados durante o trabalho foi das não-linearidades dos motores a baixas velocidades. O primeiro par de motores utilizado, com um torque de 1,92 kgf.cm, não foi capaz de executar apropriadamente os comandos da ação de controle na velocidade desejada. A troca para um segundo par com 3,6 kgf.cm reduziu o problema a níveis aceitáveis.

Em trabalhos futuros, recomenda-se o uso de uma maior redução de velocidade, o que não foi feito no presente trabalho devido à limitações do modelo.

Outro problema visto foi sobre a escolha da fonte de energia. A primeira versão do modelo contava com 4 pilhas em série para gerar os 6v de tensão dos motores. Com a necessidade de maiores torques, foram utilizados 2 conjuntos em paralelo, o que ocupou muito espaço e era de difícil manutenção, além de apresentar uma curva de tensão em relação à carga bastante desfavorável ao seu uso.

O conjunto de pilhas foi substituído por baterias de 9v, dos quais são enviados no máximo 6v aos motores via modulação por largura de pulso (PWM). Mais tarde o conjunto foi modificado para utilizar baterias recarregáveis de 9v para reduzir gastos.

As baterias do modelo também deve permanecer fisicamente desconectadas do restante do circuito enquanto não estiver sendo utilizado, pois elas descarregam mesmo que o restante do modelo esteja desligado.

3.2 Geração do programa do Arduino

Nesses códigos foi utilizada a versão 1.8.16 da IDE do Arduino e a versão do MatLab utilizda foi a 2015a, em um Windows 10 PRO versão 21H1.

Para facilitar a implementação do sistema, o programa feito no matlab gera todo o código utilizado pelo Arduino, criando vários arquivos com a extensão .h e um com a extensão .ino mesmo eles sendo escritos como arquivos de texto. Dessa forma, todo o código é mantido e atualizado e com o mínimo de chance de falhas na passagem do projeto para o Arduino. Por organização, todo arquivo também inicia com o nome do script que o escreveu.

Pode-se ver um resumo do processo que escreve e depois move os arquivos a seguir e um trecho do arquivo escrito pelo resumo na figura 29.

O destino, nome e índice do arquivo são salvos em uma variável 'FileCell', em seguida o programa abre o arquivo no modo escrita.

```
FileCell = {'data','Motores_Bluetooth_ML.h',NFiles+1};
NFiles = NFiles + 1;
fid = fopen(FileCell{NFiles,2},'w');
```

Utiliza-se uma variável auxiliar 'Tmp1' para guardar todo o texto do arquivo.

```
Tmp1 = [Tmp1 '#include "Mainprog.h" \n'];
```

Em seguida o arquivo escrito na memória.

```
fprintf(fid,Tmp1);
fclose(fid);
```

E ao final do código principal, todos os arquivos na variável 'FileCell' são movidos às respectivas pastas.

```
A = size(FileCell);
for n = 1:A(1)
    switch MyStringComparator(FileCell{n,1},...
            {'subrotinas';'modules';'data'})
        case 1
            TgtFolder = [Folders{1},'/',Folders{2},...
                '/',Folders{3},'/',Folders{4}];
        case 2
            TgtFolder = [Folders{1},'/',Folders{2},...
                '/',Folders{3}];
        case 3
            TgtFolder = [Folders{1},'/',Folders{2}];
        case 0
            TgtFolder = Folders{1};
    end
    movefile(FileCell{n,2},TgtFolder);
end
```

Os arquivos são escritos como se fossem texto (.txt) mas recebem uma extensão (.ino ou .h) dependendo da função que eles têm dentro do programa. Como o único arquivo aberto no computador é o principal e ele contém apenas um comando de incluir outros arquivos, não é necessário reiniciar o programa do Arduino para que os arquivos sejam atualizados.

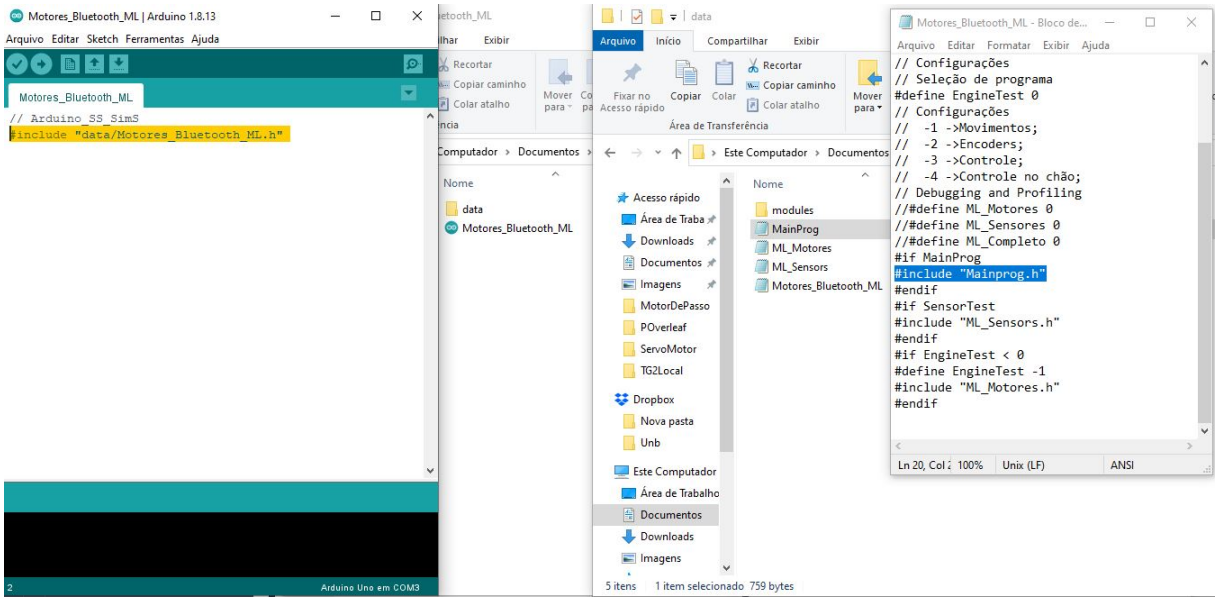


Figura 29: Exemplo dos arquivos do Arduino

3.3 Implementação dos Encoders

O código para uso dos encoders necessitou de maior atenção. Implementado com interrupções nos pinos 2 e 3 do Arduino e conectadas aos pinos digitais dos sensores, eles disparam uma interrupção sempre que detectam uma borda de subida. Os sensores ópticos utilizados apresentavam um ruído durante a troca de nível fazendo com que disparasse a interrupção associada não apenas na subida, mas quanto na descida e potencialmente, múltiplas vezes para cada troca. Foi feito um curto vídeo mostrando o problema no apêndice (aqui).

Sendo assim, mostrou-se necessário implementar um tempo mínimo entre incrementos para reduzir o efeito do problema. Esse tempo utilizado foi de 2ms determinado experimentalmente. É apresentado abaixo o código para implementação de uma dessas interrupções. Ele deve ser escrito em um arquivo de forma similar ao resumo apresentado na seção 3.2.

```
1       Tmp1 = ...
2       [...
3       'void inter_DDCounter_FWD(){ \n'...
4       '   if((millis() - Timer_AuxD) >'...
5       ',num2str(EncoderDelay),'){\n'...
6       '     Timer_AuxD = millis();\n'...
7       '     Counter_DDP = Counter_DDP + 1;\n'...
8       '   }'...
9       ];
```

Os encoders utilizados não diferenciam a movimentação para frente da movimentação para trás, sendo necessário o uso de 4 interrupções, uma para cada sentido de rotação de cada motor. Quando é enviada uma tensão ao motor, o programa ativa a respectiva interrupção para o respectivo motor e quaisquer rotação será medida como sendo na direção desejada, independente do sentido de rotação real.

3.4 A interface de comando

O programa do celular, que servirá de interface com o usuário, foi desenvolvida no site appinventor, versão '2.62 nb187c'. Nele, a programação é feita por blocos e utiliza sensores e serviços prontos no aparelho. Dentre eles, destaca-se o serviço de reconhecimento de voz e o serviço de comunicação por bluetooth.

A tela inicial do aplicativo (figura 30) permite ao usuário escolher um contato para emergências, que deverá receber uma mensagem de texto caso a bateria do celular ou da cadeira esteja baixa. Embora a navegação e funcionamento dos botões, manipulação de texto, medição de bateria do celular (feita pela extensão [41]) e persistência de memória tenham sido implementadas, a funcionalidade do envio de mensagens não pode ser realizada.

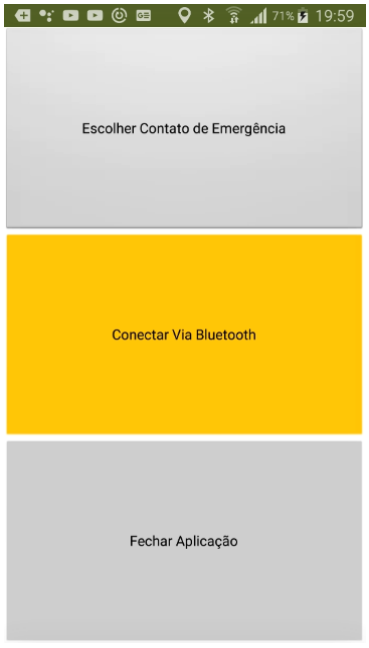


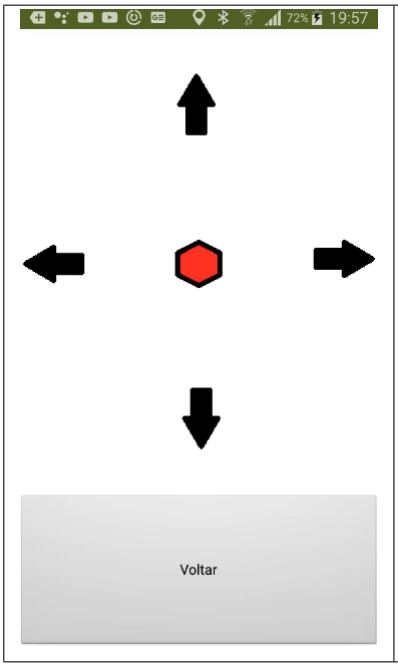
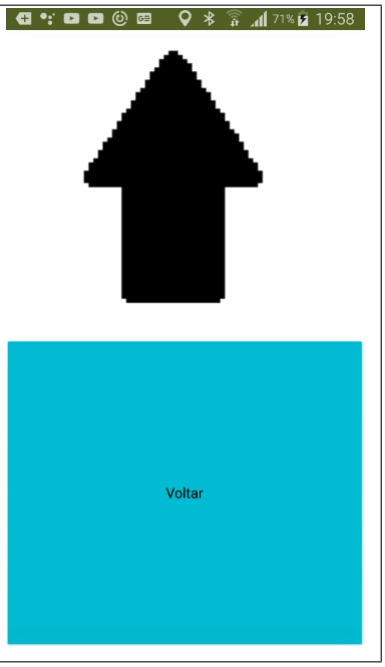
Figura 30: Tela inicial do aplicativo

Ainda na tela inicial, o usuário pode se conectar com um dispositivo bluetooth, que irá conectá-lo ao módulo do Arduino. Pela forma que a conexão está implementada, o dispositivo se conecta ao primeiro bluetooth encontrado, algo que deve ser resolvido em implementações

futuras mas é adequado para testes em ambientes controlados. Ao se conectar com um dispositivo, a tela irá mudar para a de 'Seleção de Comando', apresentada na tabela 5. Ao selecionar um modo, a tela se altera para a respectiva imagem à direita ou à esquerda da tabela 5. Esse processo está descrito na figura 31.

A tela 'Comando por Voz' altera sua imagem para apresentar o comando enviado ao Arduino, conforme apresentado na figura 32 e que no exemplo, o faz ir para frente enquanto a tela de 'Comando Manual', cujo comportamento está apresentado na figura 33, envia um comando correspondente ao ícone tocado na tela e ao remover a mão do ícone, ele comanda o Arduino a parar. O botão de parada no centro da tela não tem funcionalidade relevante e deve ser removido em implementações futuras. Uma demonstração da interface pode ser vista no anexo (aqui)

Tabela 5: Telas de escolha e de modos de uso

Comando Manual	Seleção de Comando	Comando por Voz
		

A interface apresentada na figura 9 com algumas modificações, é uma boa candidata a substituir a tela da interface por comando de voz implementada neste trabalho, por apresentar o comando sendo executado bem como os comandos possíveis, mas não houve tempo hábil para implementá-la.

As funções descritas estão representadas nos diagramas a seguir, onde o comportamento de navegação está apresentado na figura 31, o comportamento no modo de operação por voz é visto na figura 32 e o comportamento no modo de operação manual é visto na figura 33.

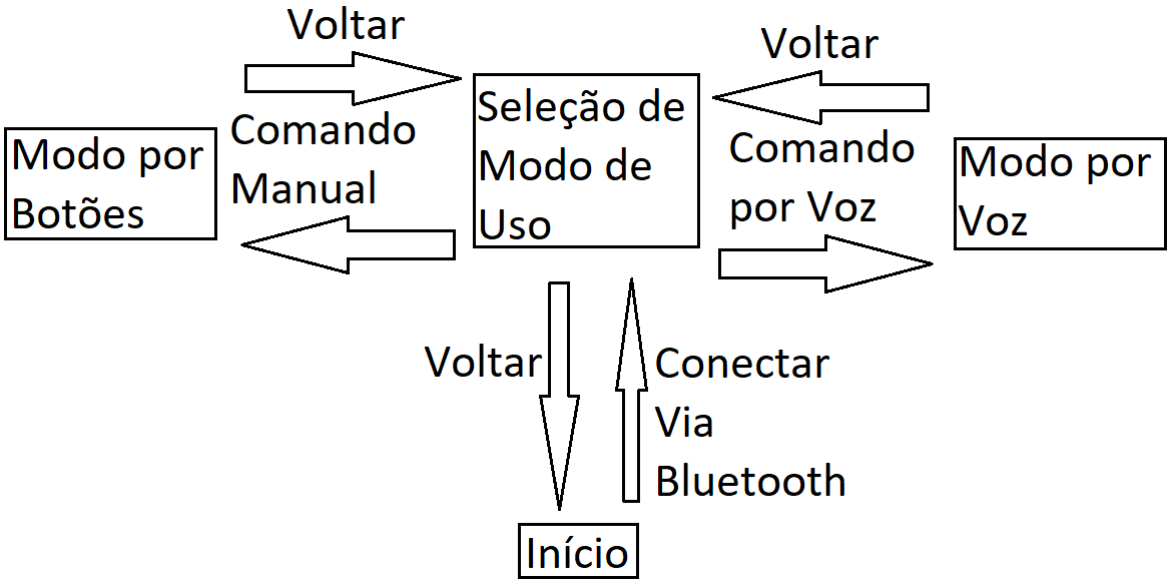


Figura 31: Diagrama da Navegação

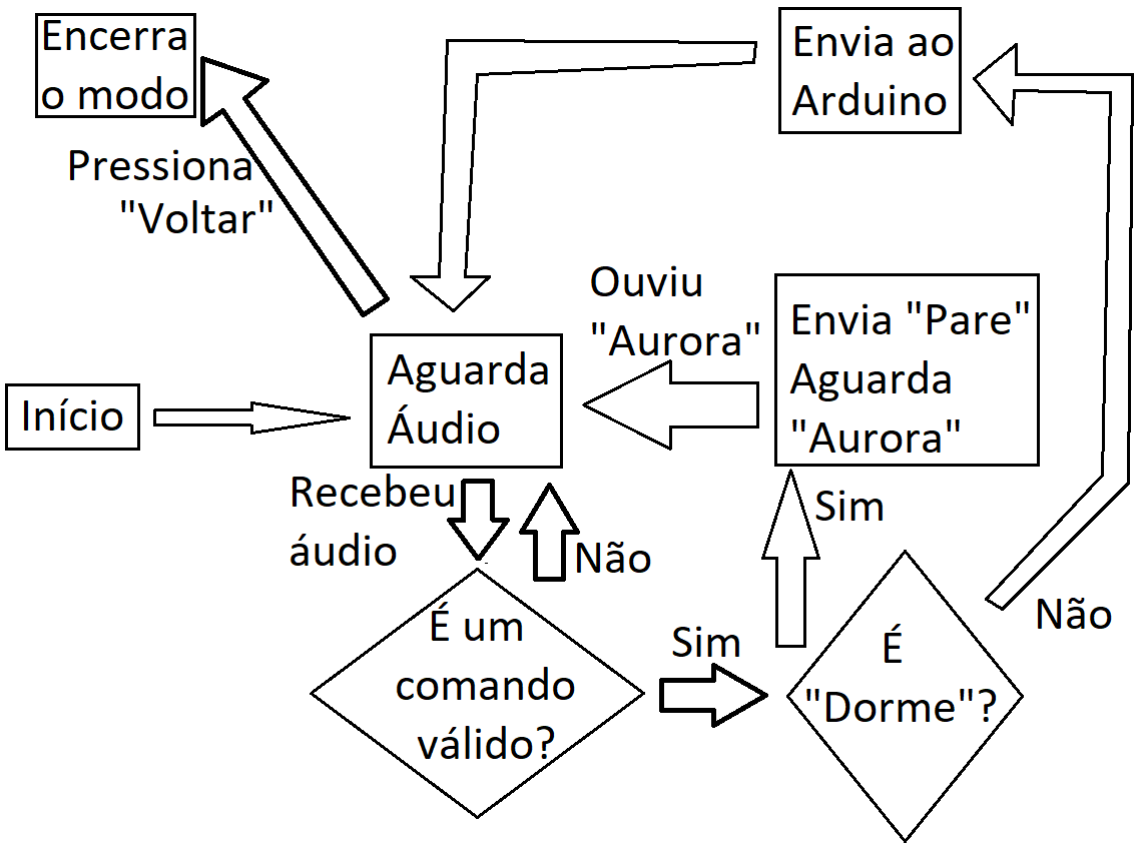


Figura 32: Diagrama da operação por voz

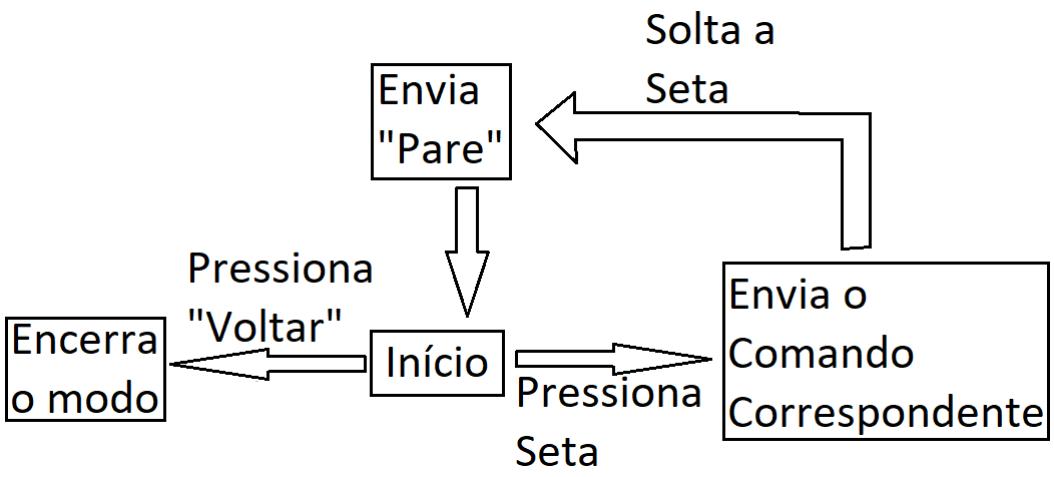


Figura 33: Diagrama da operação manual

3.5 Identificação

Esta seção descreve a metodologia utilizada para encontrar a função de transferência dos motores com carga, ou seja, durante a movimentação no chão.

3.5.1 Obtenção de Dados

O Arduino foi carregado com um arquivo de teste que o movia para frente e em seguida para trás enquanto enviava ao MatLab os dados obtidos nos encoders via bluetooth. Nesta configuração de teste, a tensão aplicada nos motores era igual a 6 volts.

O Arduino enviava dados de Tempo e Posição de cada roda, permitindo montar a resposta dos motores a um degrau de tensão em ambas as direções, ou seja, 4 funções de transferência.

3.5.2 Estimando as Funções de Transferência

A partir dos dados e um chute inicial, foi utilizada uma função de 'fit' do matlab que determina os coeficientes de uma determinada função que melhor se adequam aos dados lidos. Tal função é montada a partir da resposta ao degrau de um sistema de primeira ordem sem zeros e sem atrasos.

O trecho de código que realiza essa estimativa de dados é apresentado a seguir.

```
1 funct      = ['(b)*(1 - exp(-a*x))'];
2 funct_Vel  = fit(X,Data,funct,'Start',[10 100 0])
```

```
A  = funct_Vel.a;
B  = funct_Vel.b;
G2 = zpk([],[0 -A],B*A);\%Tensão para Posição
G1 = zpk([],[-A],B*A);\%Tensão para velocidade
```

As funções de transferência obtidas são apresentadas na tabela 15, disponível no anexo (aqui).

Para cada figura nas seções 3.4, 3.5 e 3.6 são apresentadas 4 siglas, sendo um para cada sentido de movimento de cada roda, de acordo com a tabela 6.

Tabela 6: Legenda das figuras das seções 3.4, 3.5 e 3.6

EF	Motor Esquerdo em rotação horária
ET	Motor Esquerdo em rotação anti-horária
DF	Motor Direito em rotação anti-horária
DT	Motor Direito em rotação horária

3.5.3 Comparação das Estimativas com os Dados

A comparação (figura 34) mostra, em um mesmo gráfico, a resposta ao degrau das funções de transferência estimadas e as devidas distâncias e velocidades medidas, a fim de determinar sua acurácia. Para um segundo parâmetro de comparação, é utilizada a função estimada para simular a resposta de posição para este mesmo degrau. A curva laranja representa os dados lidos, a curva azul a aproximação contínua obtida e a curva em vermelha a aproximação discreta obtida.

Devido à taxa de amostragem de 0.073, experimentalmente determinada como a menor possível para o programa no Arduino, as funções contínua e discreta estão praticamente sobrepostas.

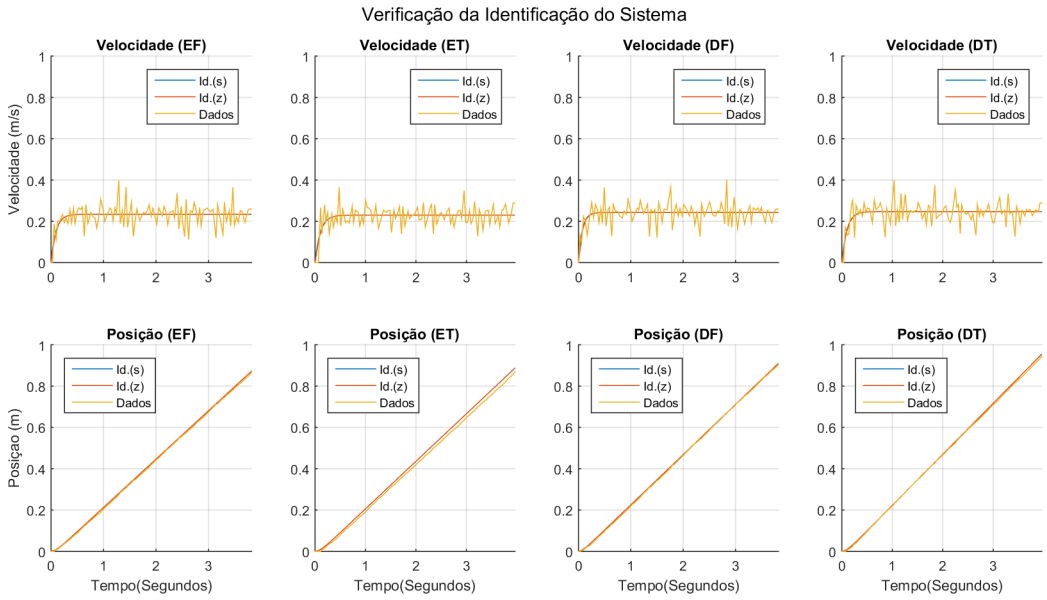


Figura 34: Identificação do sistema

A curva de velocidade não pôde ser adequadamente comparada com sua estimativa devido a oscilações bruscas de valores nos dados, o que pode ser reduzido ao agrupar conjuntos de velocidades. A curva de posição estimada se adéqua bem aos dados. Utilizando um filtro de média móvel ponderada, pode-se reduzir o ruído do sinal de velocidade e obter uma comparação (figura 35) melhor.

Foi utilizada uma razão de 0.75 no filtro apresentado pois valores menores não apresentaram mudanças significativas no sinal de velocidade.

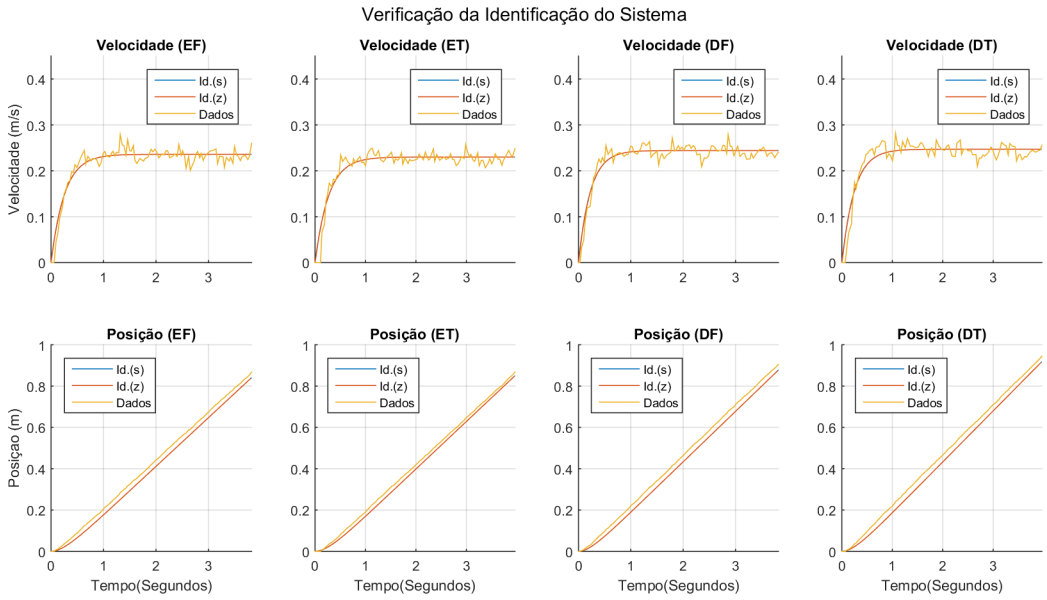


Figura 35: Identificação com filtro de média móvel de razão 0.75

Esse filtro não pôde ser implementado corretamente no Arduino e embora tenha reduzido as oscilações de velocidade observadas também introduziu um erro na estimativa de posição, conforme visto na figura 35.

3.6 Projeto dos Controladores

A partir das funções de transferência obtidas, projetou-se um controlador para cada com 10 técnicas. Cada uma será apresentada em seções seguintes acompanhada de um teste feito com o robô suspenso. Para simular cada caso foi utilizado um diagrama de blocos no Simulink apresentado abaixo (figura 36).

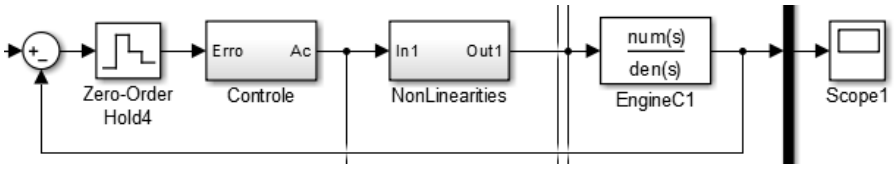


Figura 36: Sistema de teste no Simulink

Para cada caso, são feitas duas simulações, uma ideal e uma estimando as não-linearidades de atrito estático e saturação dos motores. Essa segunda utiliza um bloco 'NonLinearities' (figura 37) apresentado abaixo para simular as não-linearidades.

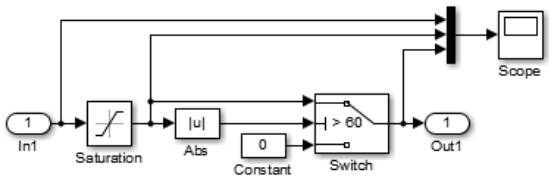


Figura 37: Bloco para simular as atrito e zona morta

Além disso, para que o controlador fosse realizável pelo microcontrolador ele foi traduzido em somas, multiplicações e atrasos por meio de uma realização na forma canônica controlável [42]. Para verificar que essa tradução não interfere no funcionamento do controle, ela foi feita em um bloco de controle (figura 38) e comparada à função de transferência original. Apenas a simulação ideal utiliza um bloco com função de transferência no formato convencional.

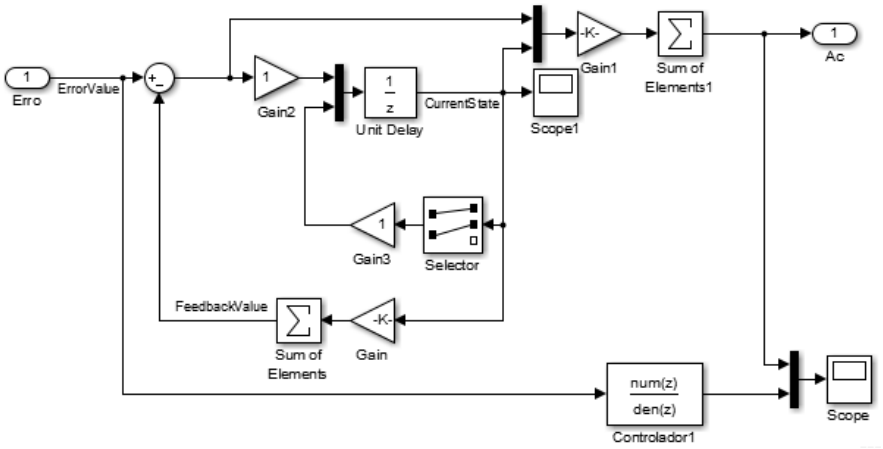


Figura 38: Comparador da tradução

Os valores apresentados aqui serão os utilizados pelo Arduino considerando funções de transferência de $\frac{Furos}{Volts}\frac{9}{255}$ enquanto os gráficos obtidos mostram metros, volts e segundos. As conversões para SI não são feitas no microcontrolador para acelerar o processamento.

Os nomes de cada controlador são relacionados à roda e sentido de movimentação na qual ele age, de acordo com a tabela 7.

Tabela 7: Legenda dos controladores na seção 3.6

CEF	Controle do Motor Esquerdo em rotação horária
CET	Controle do Motor Esquerdo em rotação anti-horária
CDF	Controle do Motor Direito em rotação anti-horária
CDT	Controle do Motor Direito em rotação horária

Para resumir os resultados das simulações de cada controlador feito, é apresentada a tabela 9 e uma legenda para a interpretação dessa tabela se encontra na tabela 8. Cada simulação, fórmula e código será apresentada na seção 5.

Tabela 8: Legenda das Abreviações da tabela 9

Controlador	Abreviação
Deadbeat (Tipo 1 Normal, Velocidade, Amostragem de 0.073s)	DB1N_Vel_T73
Proporcional Integral (Com Cancelamento, Velocidade, Amostragem de 0.073s)	PIC_Vel_T73
Proporcional Integral (Sem Cancelamento, Velocidade, Amostragem de 0.073s)	PIN_Vel_T73
Deadbeat (Tipo 1 Normal, Posição, Amostragem de 0.073s)	DB1N_Pos_T73
Deadbeat (Tipo 2 Normal, Posição, Amostragem de 0.073s)	DB2N_Pos_T73
Deadbeat (Tipo 1 Amortecido, Posição, Amostragem de 0.073s)	DB1A_Pos_T73
Deadbeat (Tipo 2 Amortecido, Posição, Amostragem de 0.073s)	DB2A_Pos_T73
Proporcional (Subamortecido, Posição, Amostragem de 0.073s)	P_Pos_T73
Lag (2.5x , Posição, Amostragem de 0.073s)	Lag_Pos_T73
Avanço (Com Cancelamento, Posição, Amostragem de 0.073s)	Lead_Pos_T73

Tabela 9: Tempos de assentamento e sobrepasso percentual observados no sinal de controle da simulação com estimador de não-linearidades

Nomes		Tempo de Assentamento			
Indice	Abreviação	EF	ET	DF	DT
1	DB1N_Vel_T73	0.29191	0.29195	0.21898	0.29183
2	PIC_Vel_T73	0.29191	0.29195	0.21898	0.29183
3	PIN_Vel_T73	Inf	0.58389	Inf	Inf
4	DB1N_Pos_T73	2.2621	2.4815	1.8242	2.262
5	DB2N_Pos_T73	Inf	Inf	Inf	Inf
6	DB1A_Pos_T73	1.4592	1.605	1.0211	1.387
7	DB2A_Pos_T73	Inf	1.0219	Inf	Inf
8	P_Pos_T73	1.5322	1.6053	1.2408	1.5323
9	Lag_Pos_T73	1.4596	1.5326	1.313	1.4596
10	Lead_Pos_T73	0.51069	0.51041	0.65622	0.51068
Nomes		Sobrepasso Percentual			
Indice	Abreviação	EF	ET	DF	DT
1	DB1N_Vel_T73	115.3403	110.0397	105.3109	125.8471
2	PIC_Vel_T73	115.3403	110.0397	105.3109	125.8471
3	PIN_Vel_T73	Inf	110.0397	Inf	Inf
4	DB1N_Pos_T73	115.3349	110.0273	122.2928	125.8413
5	DB2N_Pos_T73	Inf	9.439	Inf	Inf
6	DB1A_Pos_T73	115.3396	110.0381	121.8677	125.8463
7	DB2A_Pos_T73	Inf	110.0392	Inf	Inf
8	P_Pos_T73	47.0962	44.1166	36.6176	47.007
9	Lag_Pos_T73	40.6701	37.8254	32.9308	40.5837
10	Lead_Pos_T73	20.049	26.525	2.6555	20.2586

A escolha dos controladores a serem testados foi feita baseada nos dados de tempo de assentamento e sobrepasso do sinal de controle, apresentados na tabela 9, para os controladores 1 a 8. Os casos 9 e 10 foram feitos após a escolha do proporcional e não se mostraram melhores em seus respectivos testes, apresentados no anexo (aqui). .

3.7 Testes de Movimento

Para esta etapa de testes, foram enviados comandos de movimento ao robô e observado como ele respondia, observando se ele era capaz de manter as velocidades/posições nos valores desejados. Deseja-se que ele apresente o mesmo deslocamento a todo momento para ambas as rodas, dessa forma ele executa giros sem deslocamento nos comandos de rotação e movimentações retilíneas quando comandado a ir para frente ou para trás.

Isso pode ser feito tanto com um controle de velocidade quanto de posição, onde a versão por posição é gerada conforme ordens são recebidas. O robô também não deve fazer ajustes enquanto parado, logo não serão utilizadas soluções que acompanhem a posição que contenham um canal integral.

Para o caso das figuras que representem testes, a linha azul é a referência e a linha laranja é a variável medida enquanto para as simulações, a azul é o comportamento esperado do sistema ideal e a vermelha do sistema com não-linearidades estimadas.

Deadbeat Tipo 1

O controle desse tipo busca acompanhar a referências do tipo degrau no menor tempo possível, sendo sujeito a problemas de saturação e variações na função de transferência. Ele também não garante que o sistema não oscila entre os instantes de amostragem.

O primeiro controlador projetado, visto que ele satisfaz os requisitos de erro e é o mais simples de se projetar. Nesta implementação, ele se comporta de maneira similar a um Proporcional-Integral com cancelamento do pólo e ganho para posicionar o pólo restante em 0.

A simulação do projeto pode ser vista na figura 39 e acompanha a referência sem erro após aproximadamente 0.5 segundo, um bom resultado. Em seguida esse mesmo controlador foi implementado no Arduino e foram testados os 4 comandos principais. O resultado do teste pode ser visto na figura 40.

$$C = \frac{1}{G}\frac{1}{(z-1)} \tag{1}$$

$$CEF = \frac{6.3182z - 3.6867}{z-1} \tag{2}$$

$$CET = \frac{6.827z - 4.1291}{z-1} \tag{3}$$

$$CDF = \frac{5.2338z - 2.6846}{z - 1} \tag{4}$$

$$CDT = \frac{6.0348z - 3.5257}{z - 1} \tag{5}$$

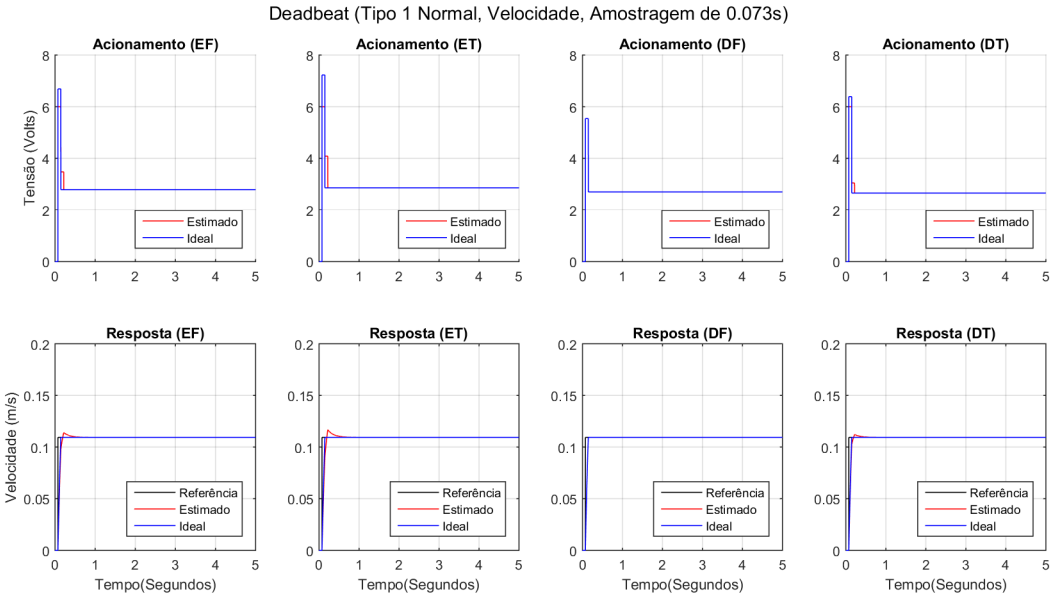


Figura 39: Simulação do DeadBeat Tipo 1

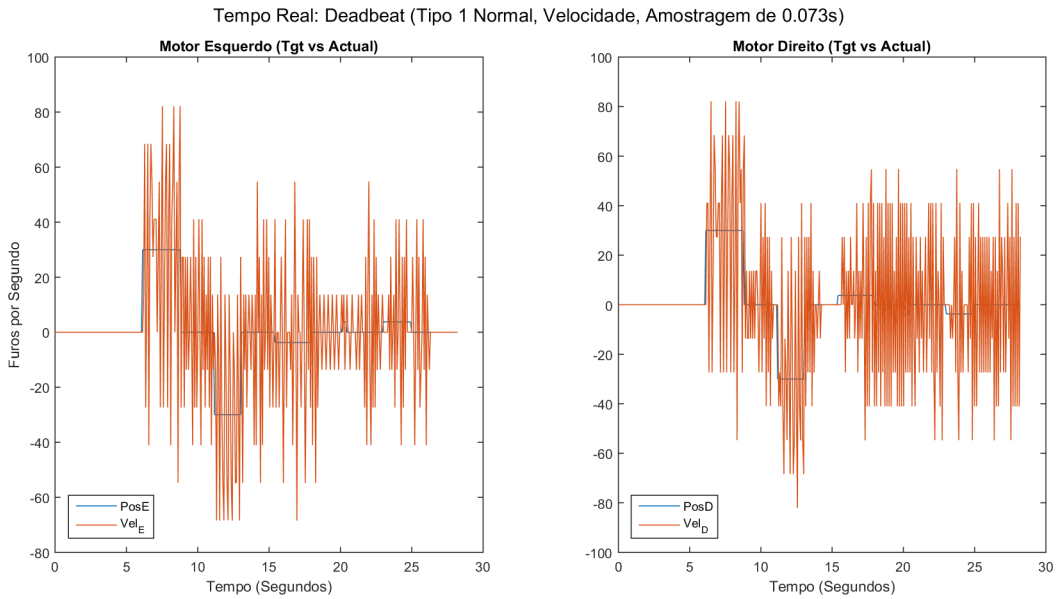


Figura 40: Teste Suspenso do DeadBeat Tipo 1

Observou-se na figura 40 que a medição de velocidade apresenta grandes quantidades de

ruído. Será então feita uma variação deste mesmo controle mas que segue a posição em vez da velocidade. A simulação é apresentada na figura 41 e o teste na figura 42.

$$C = \frac{1}{G}\frac{1}{(z-1)} \tag{6}$$

$$CEF = \frac{158.902z - 92.71986}{z + 0.83595} \tag{7}$$

$$CET = \frac{172.6354z - 104.4135}{z + 0.84595} \tag{8}$$

$$CDF = \frac{129.1284z - 66.23432}{z + 0.80107} \tag{9}$$

$$CDT = \frac{151.8027z - 88.68756}{z + 0.8363} \tag{10}$$

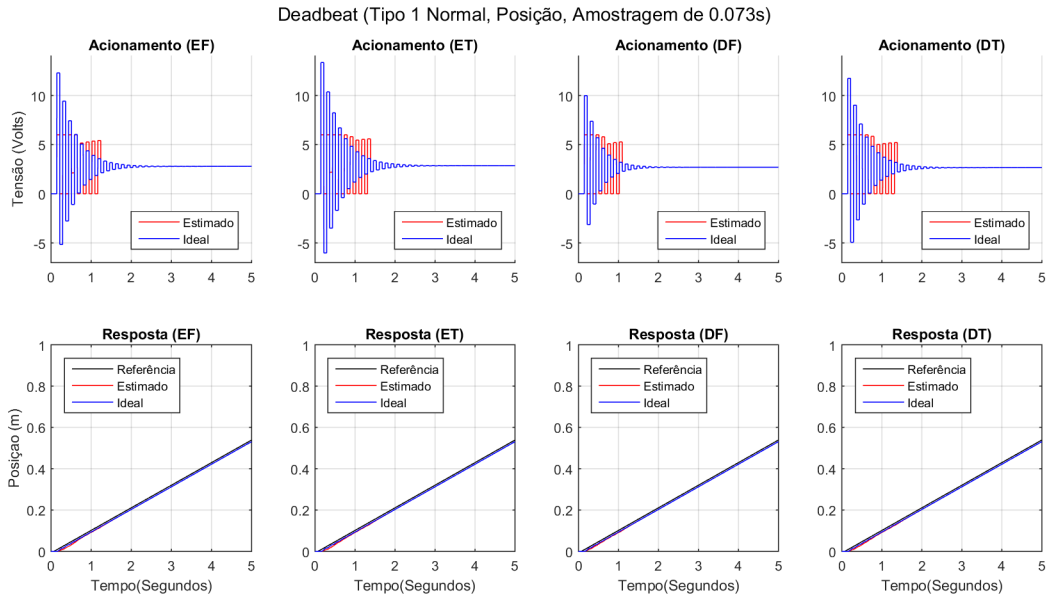


Figura 41: Simulação do DeadBeat Tipo 1

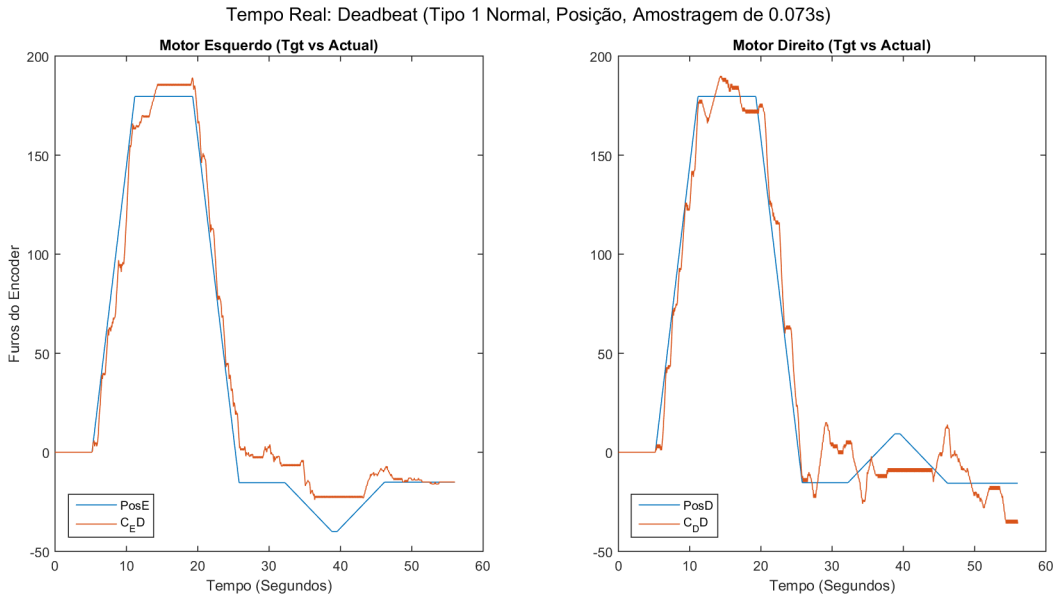


Figura 42: Teste Suspenso do DeadBeat Tipo 1

Mesmo que a simulação (figura 41) apresente uma resposta que segue ao comando de posição, no teste (figura 42) ele não se comportou da mesma maneira. Foi introduzido então um pólo e um zero, arranjados de forma similar a um compensador em atraso, de forma a reduzir a excursão do sinal de controle.

O controlador seguinte será chamado de DeadBeat Amortecido, visto que é de se esperar um maior amortecimento na aproximação de segunda ordem devido ao compensador em atraso. Como nos casos anteriores, é apresentada a simulação em 43 e o teste em 44.

$$C = \frac{1}{G}\frac{0.6z + 0.4}{(z-1)(z+0.4)} \tag{11}$$

$$CEF = \frac{95.3412z^2 + 7.92889z - 37.0879}{z^2 + 1.236z + 0.33438} \tag{12}$$

$$CET = \frac{103.5812z^2 + 6.406029z - 41.76542}{z^2 + 1.246z + 0.33838} \tag{13}$$

$$CDF = \frac{77.477z^2 + 11.9108z - 26.4937}{z^2 + 1.2011z + 0.32043} \tag{14}$$

$$CDT = \frac{91.0816z^2 + 7.50852z - 35.475}{z^2 + 1.2363z + 0.33452} \tag{15}$$

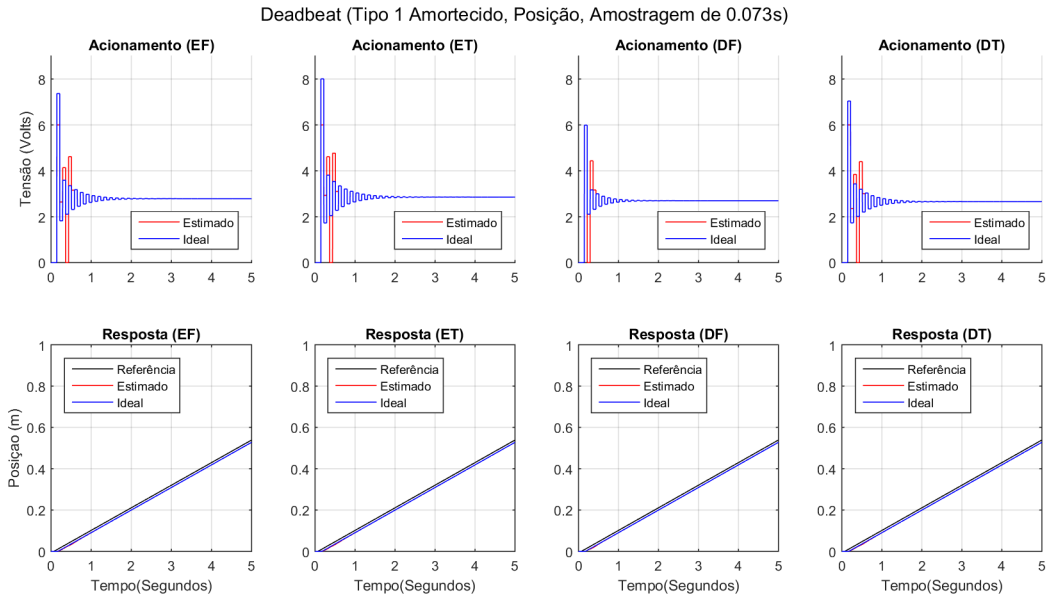


Figura 43: Simulação do DeadBeat Amortecido Tipo 1

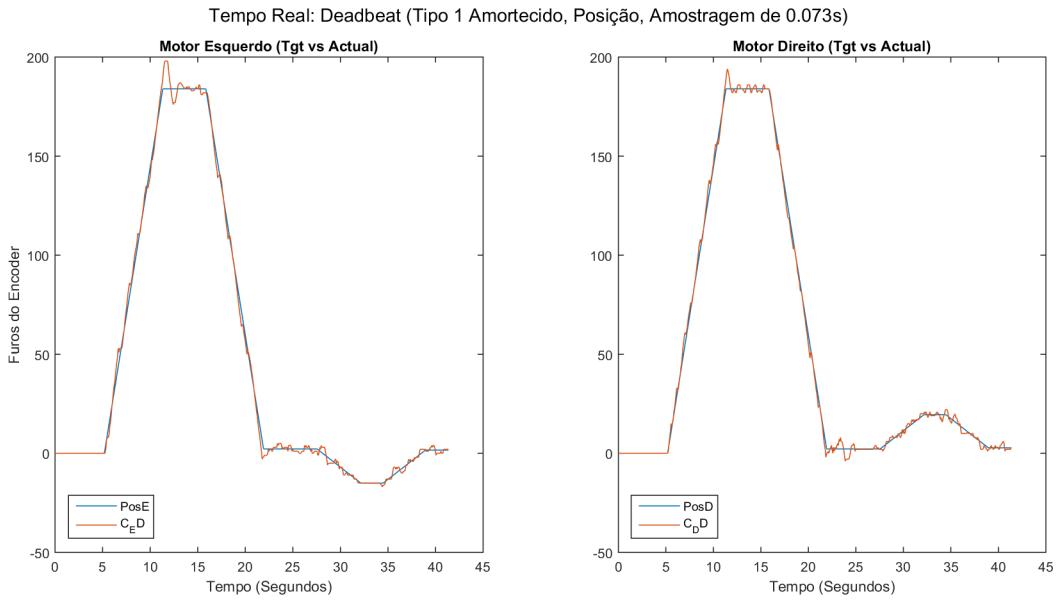


Figura 44: Teste Suspenso do DeadBeat Amortecido Tipo 1

Observou-se que nesta solução, a excursão do sinal de acionamento (figura 43) é drasticamente reduzida, mesmo que se mantenha alta. O comportamento no teste (figura 44) também foi muito melhor, seguindo o sinal de referência com um pequeno mas devido às movimentações enquanto parado, essa solução também será descartada.

Proporcional

Visto que as outras soluções não foram satisfatórias, foi projetado um controlador mais simples. Ele apresentou uma resposta mais lenta na simulação em 45 mas operou bem no teste em 46 ao atender os critérios desejados de acompanhar a referência e não se movimentar após parado, portanto, foi usado no restante do trabalho.

$$C = K \tag{16}$$

$$CEF = 12.7534 \tag{17}$$

$$CET = 12.311 \tag{18}$$

$$CDF = 14.8422 \tag{19}$$

$$CDT = 12.1357 \tag{20}$$

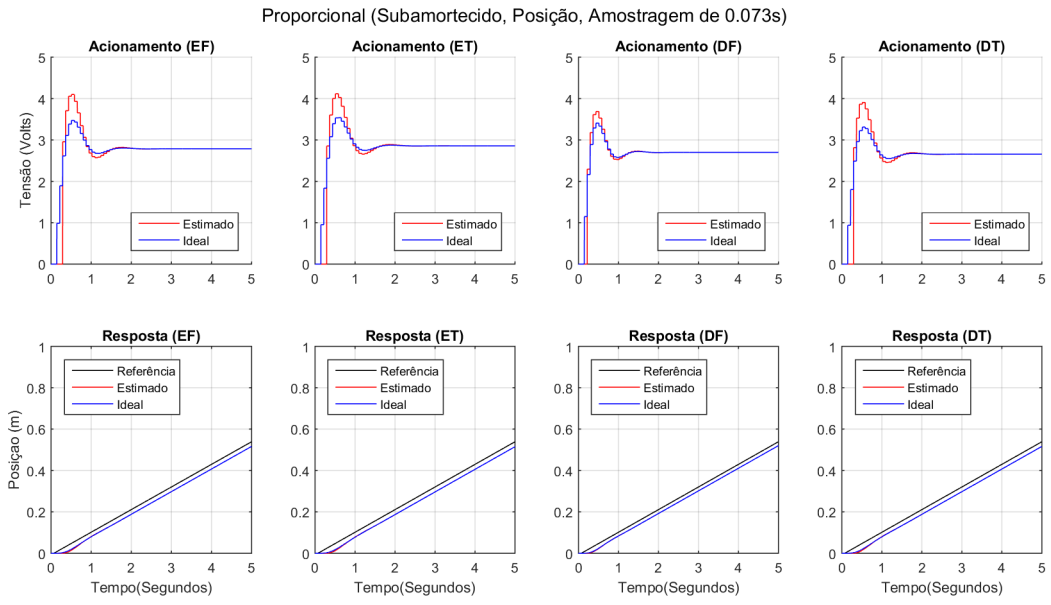


Figura 45: Simulação do Proporcional

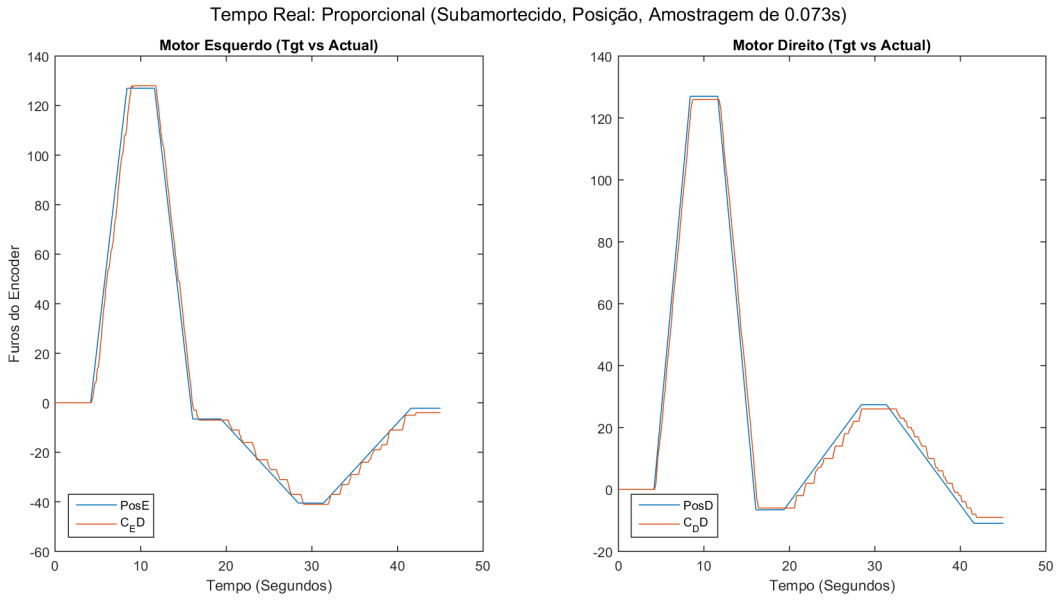


Figura 46: Teste Suspenso do Proporcional

3.7.1 Testes do controle em campo

Utilizando estes controladores, foi feito um teste de movimentação retilínea, onde o robô era posicionado sobre uma fita de 2 metros de extensão e ao final da movimentação deveria estar posicionado o mais próximo possível do centro dessa fita. Os deslocamentos laterais do centro do robô em relação ao centro da fita, em metros, estão apresentados na tabela 10. Erros para a esquerda foram considerados negativos e erros para a direita como positivos.

Entre as movimentações o móvel era reposicionado e realinhado mas não eram feitas alterações como reiniciar a medição de posição ou ajeitar as rodas. A fita a servir de referência pode ser vista na figura 47 e um exemplo da movimentação pode ser acompanhado no apêndice (aqui).

Figura 47: Fita de referência para o teste de movimento (Fronte: Autor)

Tabela 10: Deslocamento lateral durante a movimentação retilínea (metros em relação à fita)

Erro Para Frente	0.05	-0.1	0.08	0.23	0.11	0	0.1	-0.11	0.06	0
Erro Para Trás	0	0.06	0.06	-0.1	-0.06	-0.06	-0.03	-0.26	-0.14	-0.05

Da tabela 10, pode-se notar que o erro em graus oscila entre 8 graus para a esquerda e 7 graus para a direita, utilizando os maiores erros observados na movimentação.

3.8 Testes do comando de voz

Para os testes dessa seção foi utilizado um celular do modelo Galaxy S5, onde foram enviados 20 comandos de voz ao programa no celular e verificadas as taxas de sucesso e falha do mesmo. No caso de sucessos, foi verificado o intervalo de tempo entre o envio do comando e a resposta do programa. Os comandos válidos são apresentados na tabela 11.

Tabela 11: Tabela dos comandos válidos

Avançar	Direita	Esquerda	Recuar	Pare	Dorme	Aurora

Quaisquer palavras que contenham essas mesmas sequências de letras irão ser falsos-positivos visto que nessa implementação, a implementação não é capaz de diferenciá-los. A palavra 'Dormente' por exemplo, contém a string 'Dorme' então sempre será reconhecida como um comando válido.

Devido a esses problemas estão feitos os comandos de 'Aurora' e 'Dorme', para que o usuário possa dialogar enquanto na cadeira sem desligá-la, desde que esteja parada.

Estes testes foram divididos em etapas, nas quais eram observados subconjuntos de fatores. Problemas de início e falha da ativação do reconhecimento por exemplo, foram ignorados fora de sua respectiva secção.

3.8.1 Medição do Atraso

Foi medido o tempo entre o envio e o reconhecimento dos comandos. As figuras 49 e 50 mostram os resultados dos testes agrupados em intervalos. Valores na transição entre 2 faixas foram tratados como se fossem menores.

A medição foi feita gravando um vídeo e enviando-o a um editor de vídeos onde, para cada comando identificado, era feito um corte ao fim da fala e outro após a resposta do celular. O tamanho do clipe entre os cortes era medido pelo software e é o tempo de atraso buscado.

O editor utilizado foi o 'VideoPad' e o procedimento está resumido na figura 48

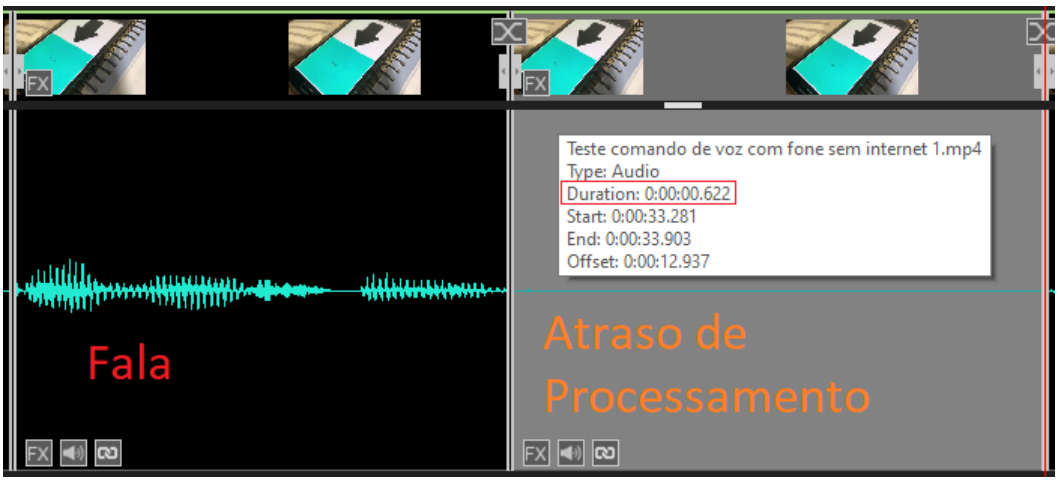


Figura 48: Processo de medição de atraso

Vale ressaltar que embora o atraso de processamento e comunicação seja grande, o tempo necessário para a verbalização da palavra também é. Isso torna a interface ainda menos responsiva.

Também foi observado que a disponibilidade de internet afeta significativamente o atraso do processo, então foi feito um conjunto de testes com internet (figura 49) e outro conjunto sem internet (figura 50). O processo de medição do atraso apresentado na figura 49 pode ser visto em um vídeo no anexo (aqui).

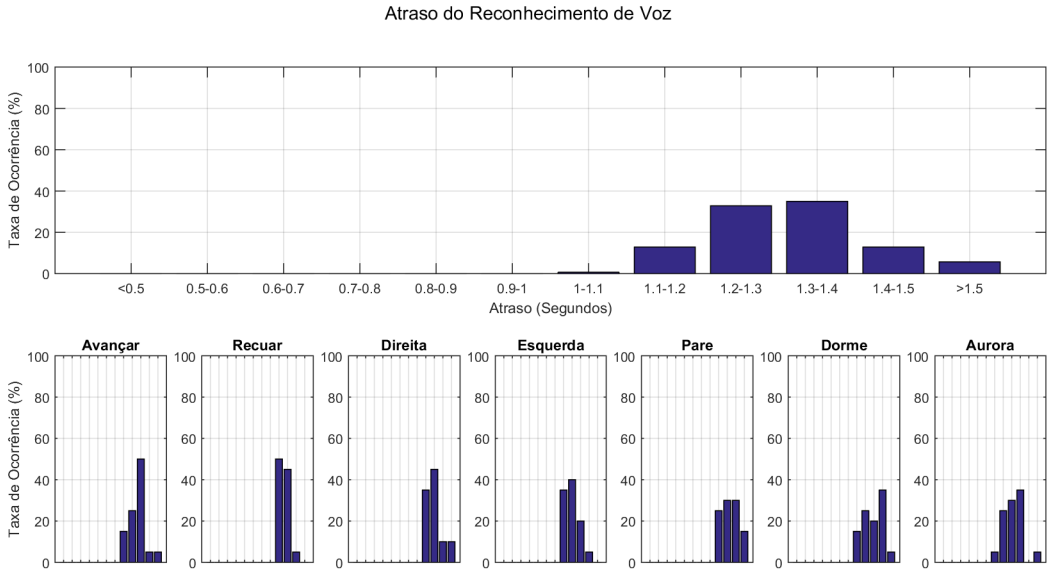


Figura 49: Medição do Atraso Com Internet

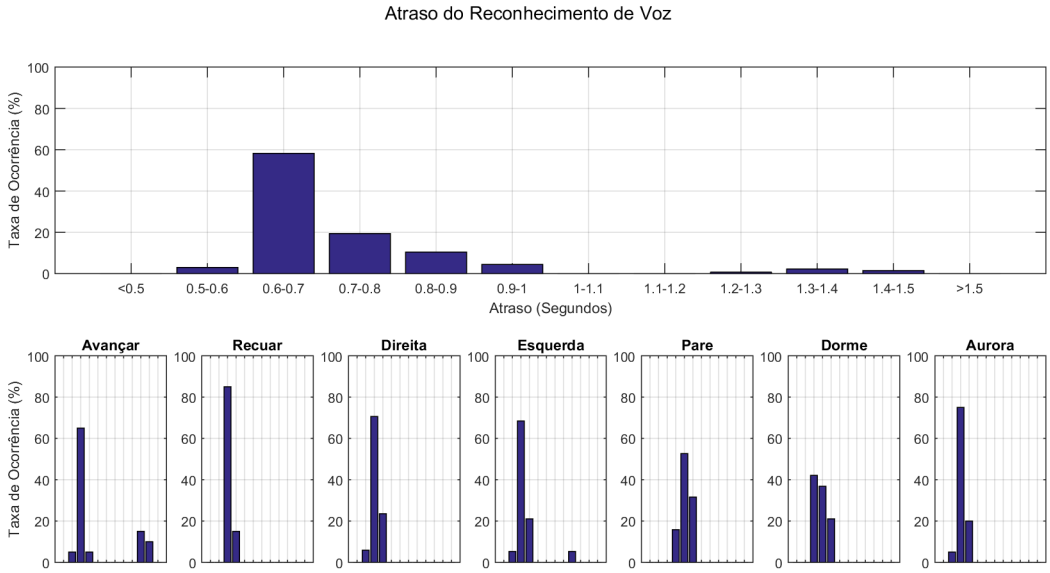


Figura 50: Medição do Atraso Internet

Observa-se que conforme informado em [24], no teste com internet (figura 49) os atrasos são significativamente maiores, com média próxima de 1.3 segundos. Nos resultados de reconhecimento sem internet (figura 50), embora ainda existam atrasos de até 1.5 segundos e a média esteja em 0.75 segundos, a maioria dos atrasos observados caiu para menos de 0.7 segundos, uma redução de 47% no tempo de resposta em relação da média anterior.

Embora esse reconhecimento seja mais rápido por não aguardar uma resposta do servidor da Google, ele apresenta uma acurácia reduzida, conforme visto na figura 50.Neste trabalho, optou-se pela solução com tempo de resposta menor.

3.8.2 Teste do microfone

A fim de determinar se o uso de um microfone melhora significativamente a acurácia do programa, foi feito mais um conjunto de testes em que eram enviados 20 comandos à interface. A figura 51 mostra o microfone utilizado e a figura 52 Compara as taxas de erro com e sem microfone, considerando apenas casos em que foi obtido o áudio.

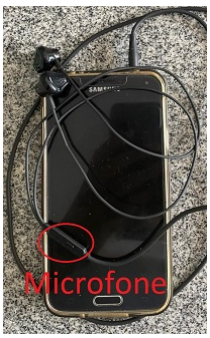


Figura 51: Celular e Microfone Utilizados

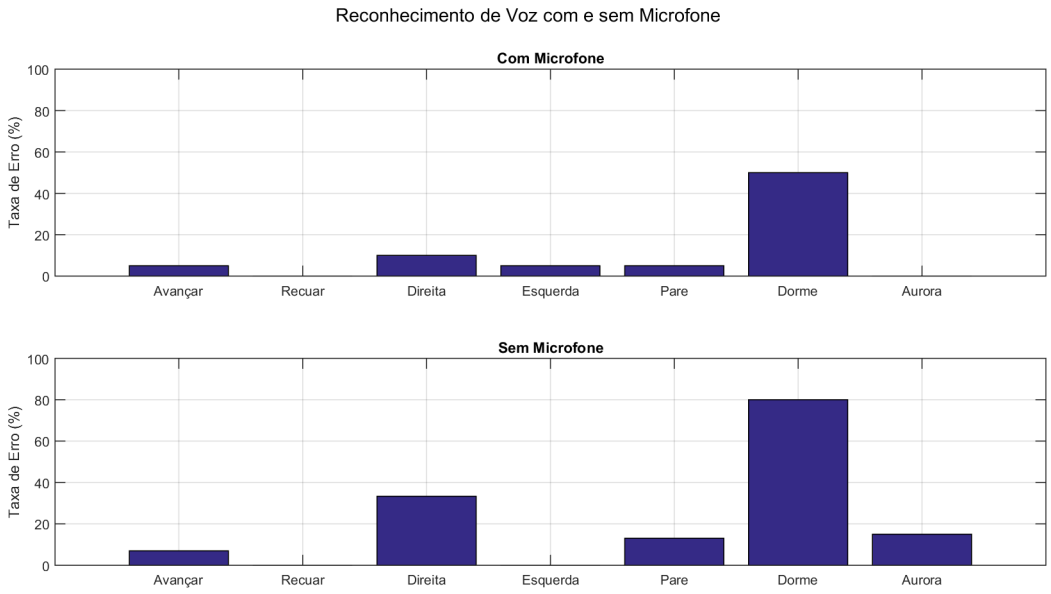


Figura 52: Teste do Microfone

Pode-se ver que de acordo com a figura 52, o uso de um microfone reduz levemente a taxa de erro em quase todos os casos, sendo assim, os testes operacionais utilizarão um.

3.8.3 Taxa de falso-positivos

Série de testes para averiguar a taxa falha devido a palavras similares, onde eram enviadas palavras bem próximas das que o software esperava enquanto era observado se o aparelho executou tal comando. Os resultados podem ser vistos na figura 53 Foram utilizadas as seguintes palavras na tabela 12:

Tabela 12: Indicação dos falso-positivos testados

Comando	Avançar	Direita	Esquerda	Pare	Recuar	Dorme	Aurora
Falso-Positivo	Avançado	Direito	Esquerdo	Para	Recuado	Dormir	Agora

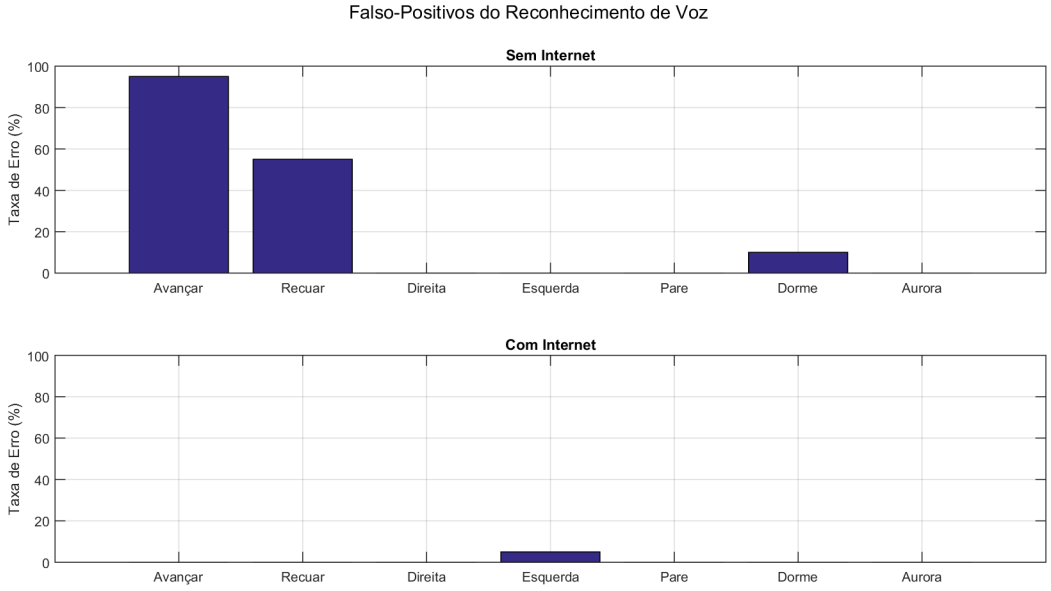


Figura 53: Taxa de falso-positivos

Nota-se na figura 53 a taxa de falso-positivos é desprezível com internet porém significativamente alta sem internet para os 2 primeiros comandos, chegando a quase 100% para a palavra ’Avançado’ e 60% para a palavra ’Recuado’.

3.8.4 Taxa de falso-negativos

Nesta etapa foram enviados 20 comandos válidos e verificadas as taxas de sucesso no reconhecimento e subsequente execução do dito comando. Para cada comando executado, foi medido o tempo entre o envio e a execução. Os resultados podem ser vistos na figura 54.

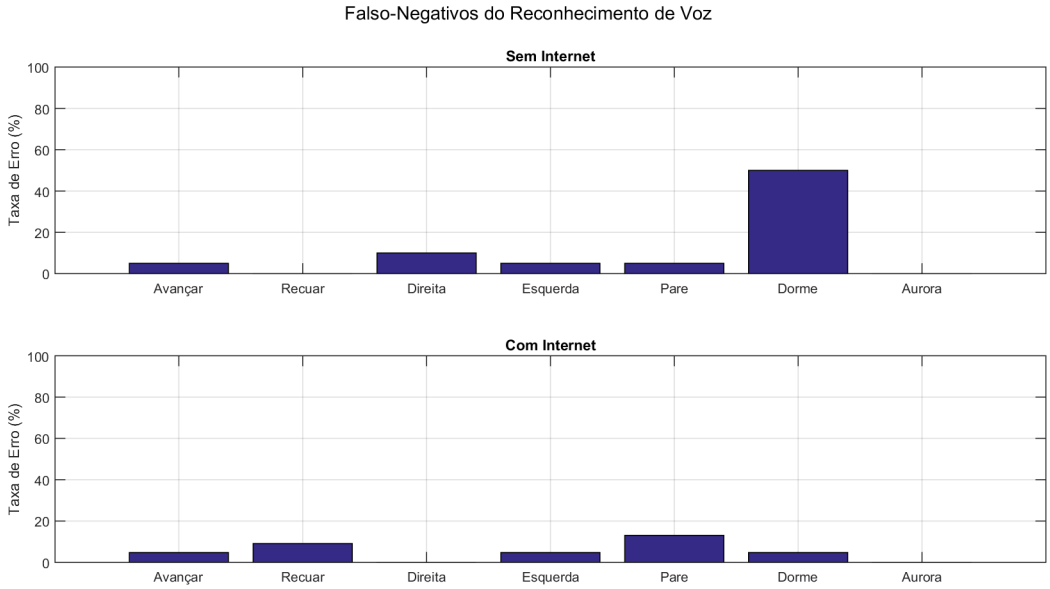


Figura 54: Teste Sem Internet

A taxa de falha observada na figura 54 apresenta um aumento na taxa de erro no reconhecimento sem internet para a maioria dos comandos, em especial, o 'Dorme'. Embora espera-se que ele não seja muito utilizado, pode ser uma experiência frustrante ao usuário.

3.8.5 Taxa de falha geral

Taxa na qual o reconhecimento falha por quaisquer razão, desde a obtenção do áudio até a identificação da fala. Algumas de suas causas, como o início e fim da obtenção de áudio, ocorrem de forma randômica e não foi possível resolvê-las em tempo hábil.

Para obter essa taxa foi proposto que usuários movessem o robô por um trajeto enquanto era avaliada apenas a taxa de sucesso do reconhecimento de voz. Visto que deseja-se o menor tempo de atraso possível, a versão utilizada nestes testes não utilizava internet e contava com um microfone. Resultados apresentados na figura abaixo.

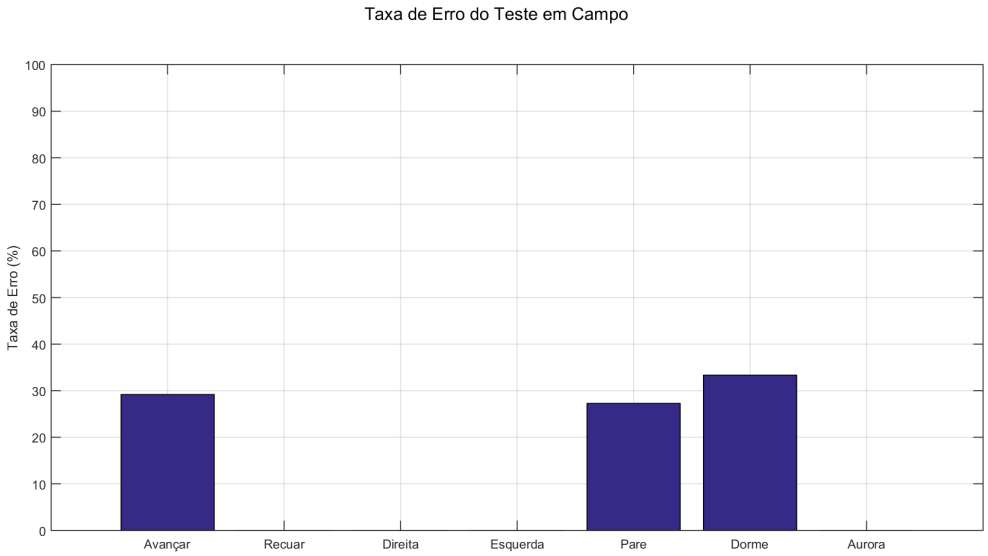


Figura 55: Teste de Uso

De acordo com a figura 55, a taxa de falha oscila entre 30% e 40%, valores mais altos para a maioria dos comandos enquanto mais baixo para o comando ’Dorme’. Isso pode ser devido ao pequeno conjunto de dados quando comparado ao teste feito nas figuras 52 e 54.

3.8.6 Custo de falha

O atraso médio sem internet foi de 0,75 segundo, assumindo que o usuário leva 1 segundo para proferir um comando e que o processamento será igual ao tempo médio, a latência típica seria de 1,75 segundos. Em caso de falha, seria necessário um segundo comando, sendo assim a cadeira iria percorrer a distância correspondente ao dobro do que ela percorreria em apenas 1 comando.

Sabendo que a velocidade média da cadeira motorizada do GRACO é de 0,6 m/s e adotando uma margem de segurança de 0,4 m/s, será utilizado o valor de 1m/s para representar a velocidade típica durante a operação. Com 3,5 segundos entre o comando e a parada, a cadeira percorreria 3,5 metros. Sendo assim, o sistema de detecção de objetos deveria detectar objetos a esta distância ou o sistema e controle deveria movimentar a cadeira a uma velocidade alvo reduzida. Recomenda-se utilizar a segunda abordagem por segurança.

Vale pontuar que diante de uma situação de estresse ou perigo, como a cadeira andar em direção a uma escada, será observada uma taxa ainda mais elevada de erros mas dessa vez devido a erros usuário, por exemplo, enviando ’Para’ em vez de ’Pare’ à cadeira. Uma solução

proposta seria de utilizar partes da string (como ’Par’ em vez de ’Pare’) que aumenta as taxas tanto de acerto quanto de falso-positivos.

3.9 Detecção de Obstáculos

3.9.1 Implementação

Os sensores de proximidade, posicionados conforme a figura 25 devem ser capazes de medir a distância de objetos próximos e interromper a movimentação caso tal objeto esteja dentro de um limiar. O seguinte código programa a medição de distância:

```
Tmp1 = [...
'if (millis() - SensorTimer > 50){\n'...
'  SensorTimer = millis();\n'...
'  digitalWrite(TrigT,LOW);\n'...
'  digitalWrite(TrigT,HIGH);\n'...
'  delayMicroseconds(15);\n'...
'  digitalWrite(TrigT,LOW);\n'...
'  if (SensorSwitch){\n'...
'    SensorSwitch = 0;\n'...
'    DistF = pulseInLong(EchoF,HIGH,40000)*Enc_SegToCm;\n'...
'  }else{\n'...
'    SensorSwitch = 1;\n'...
'    DistT = pulseInLong(EchoT,HIGH,40000)*Enc_SegToCm;\n'...
'  }\n'...
'}\n'...
];
```

E o trecho que faz a detecção está apresentado abaixo:

```
Tmp1 = [...
'\n#include "Sensors_Medicao.h"\n'...
'// Gera a detecção\n'...
'if(DistT == 0){\n'...
'  DistT = 4000;\n'...
'}\n'...
'if(DistF == 0){\n'...
'  DistF = 4000;\n'...
'}\n'...
'Sensor = (Dist > DistT) | ((Dist > DistF) << 1);\n'...
```

```
11  'FT_TPFDE = ((Sensor << 5) | FT_TPFDE);\n'...
12  ];
```

Ele leva cerca de 100 milisegundos para fazer ambas as medições mas não acrescenta atrasos significativos ao programa. Em seguida deve-se comparar a distância medida a um limite seguro determinado pela velocidade de movimento esperada do móvel. O código deve ser enviado ao Arudino conforme descrito na seção 3.2.

A referência de velocidade está ajustada para pouco mais de 0.1 m por segundo, sendo assim, espera-se que ele seja capaz de parar ao detectar um objeto a até 10 cm de distância. Com uma margem de erro de 2x, será utilizado a distância de 20 cm como alvo para interromper a movimentação.

3.9.2 Testes da Detecção

O teste isolado, no qual o robô se encontrava suspenso foi bem sucedido. A movimentação alvo era interrompida quando o respectivo sensor fosse obstruído. O alcance de detecção obtido também era significativamente inferior ao informado pelo datasheet, detectando objetos a até 1 metro de distância em vez dos 4 metros indicados.

O teste em uso dos sensores funcionou para impedir o início da movimentação e pode ser acompanhado em no anexo. Ocasionalmente, ele também interrompia a movimentação para frente, provavelmente devido a uma inclinação indesejada do sensor dianteiro de proximidade, que às vezes detectava o chão.

Observou-se também que objetos detectados podiam ser perdidos caso estivessem no limite de alcance, o que pode ser resolvido implementando um efeito de histerese na detecção, em que uma vez detectado um objeto a 20 cm, o sensor só se move novamente ao não detectar nada dentro de 30 cm, por exemplo. Esse processo não foi feito neste trabalho.

4 Resultados do sistema completo

Para testar o sistema completo, foi feito um trajeto, apresentado na figura 56, em que foram medidos os tempos necessários para que o robô executasse uma volta completa no mesmo.

Figura 56: Local de testes do sistema completo

Para referência, foi obtido o tempo necessário utilizando a interface no modo manual, que foi de aproximadamente 2 minutos. Segue na tabela 13, em minutos:segundos, o tempo necessário para percorrer o trajeto.

Tabela 13: Tempo gasto em uma volta

Tentativa	1	2	3	4	5	6	7
Tempo	2:14	2:15	2:27	2:02	2:20	2:25	2:24

Pode-se verificar pela tabela 13 que a interface por comandos de voz implementada apresenta um desempenho geral um pouco mais lento, de até 25%, em relação à interface por botões. Ela também apresenta uma variação grande de desempenho devido à ocorrência de falhas. Com um tempo médio de 2:18, a perda de desempenho não é grande o suficiente para inviabilizar a solução.

Para propósitos de apresentação de dados, está representado na tabela 14 o subconjunto dos parâmetros indicados na introdução com seus respectivos valores. Ela se refere aos testes sem

internet com fone.

Tabela 14: Parâmetros de [11] preenchidos de acordo com os resultados

Eficácia (Taxa de Acerto)	89%
Eficiência (Tempo para percorrer o trajeto)	2mim 27s
Tolerância a erro (Diferença do maior pro menor tempo)	25s
Facilidade de aprender (Percentagem de comandos válidos entregues)	65%
Satisfação (Nota média do usuário ao sistema)	- Não Medido -

Para propósitos de apresentação, links para vídeos de um teste com internet e um teste sem internet no percurso final estão disponíveis no anexo (aqui).

5 Conclusões

A implementação da interface com o usuário apresentou atrasos maiores do que o esperado, mostrando ter menor acurácia e/ou maior tempo de resposta. Embora isso não inviabilize a solução, as desvantagens observadas neste trabalho frente a outras soluções, principalmente quanto ao tempo de resposta, resulta em uma experiência pouco agradável para o usuário.

O reconhecimento de voz utilizado também não é contínuo e isso contribuiu para uma piora significativa da taxa de acerto. Sendo assim, em trabalhos futuros, sugere-se buscar uma alternativa contínua e com processamento de reconhecimento de voz local, visto que o uso da internet adiciona um atraso relevante nesta aplicação.

O robô montado não apresentou problemas de mal contato ou fios que se desconectam, mesmo sem o uso de uma placa impressa. O restante da estrutura apresentou vários problemas, como desalinhamento dos sensores e rodas além de ocasionalmente perder uma das rodas laterais durante a movimentação. Tais problemas interferem no desempenho e não foi possível corrigi-los no tempo disponível.

A comunicação da interface desenvolvida com o robô mostrou uma certa instabilidade, onde a conexão podia demorar para se estabelecer e ocasionalmente encerrava, mesmo dentro do alcance do bluetooth. O atraso de comunicação por outro lado, é imperceptível a ”olho nu”.

Os vários controladores implementados funcionaram bem em simulação, mas nem tanto no robô. Mesmo assim, foi obtido experimentalmente um erro inferior a 10 graus de erro, satisfatório para movimentações simples e que pode ser reduzido com um sistema físico mais robusto.

O sistema completo foi capaz de executar os comandos dados a maior parte do tempo mas mostrou-se necessário um esforço extra do usuário de compensar o atraso de resposta. Com o controlador proporcional, o robô móvel era capaz de seguir em linha reta enquanto a roda não desmontasse nem a comunicação desconectasse.

Vale mencionar que nenhum desses dois problemas ocorreu durante os testes de movimento com outras pessoas e as maiores reclamações vinham do atraso e da interface não indicar quais os comandos válidos para o robô.

Também foi observado que devido às não-linearidades dos motores, após rotações, a direção do chassi apresentava um erro em relação à alvo do software, tornando muitas vezes necessárias correções após curvas. Em raras ocasiões, a medição de posição apresentava saltos, mesmo sem nenhum acionamento. Não foi possível determinar a causa nem a frequência em que essa falha

ocorre.

De forma geral, a implementação da interface por comandos de voz, a montagem e programação do robô móvel para verificação da interface foram um sucesso. O programa de apoio do MatLab também foi feito e agilizou os testes desde a obtenção até a e apresentação de resultados.

Referências

[1] OLIVEIRA, Francisco Matheus Pereira de. Estudo e desenvolvimento de uma interface de comando de cadeira de rodas motorizada para pessoas tetraplégicas. 2019. 80 f. Trabalho de conclusão de curso (Bacharelado em Engenharia Mecatrônica)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/24741> Acesso em : 05/11/21.

[2] Smart, Infinity Pop (PF). Disponível em: <http://smartcr.com.br/produto/infinity-pop-pf> Acesso em : 05/11/21.

[3] Smart, Unawheel. Disponível em: <http://smartcr.com.br/produto/unawheel> Acesso em : 05/11/21.

[4] PINTO, Frederico Fernandes. Interfaces de controle de cadeiras de rodas motorizadas para pessoas com tetraplegia. 2016. x, 53 f., il. Trabalho de conclusão de curso (Bacharelado em Engenharia Mecatrônica)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/17086> Acesso em : 05/11/21.

[5] Azevedo, Abílio Marcos Coelho de.,Denadai, Walcimar Z., Lima , Luis E. M. INTERFACE HOMEM-MÁQUINA COM USO DE COMANDOS DE VOZ EMBARCADA EM PLATAFORMA ANDROID PARA CONTROLE DE DIRIGIBILIDADE DE UMA CADEIRA DE RODAS MOTORIZADA. Congresso Brasileiro de Automática, XXI, 2016,Av. Vitória, 1729 - Jucutuquara, Vitória - ES. Disponível em: <https://abilioazevedo.files.wordpress.com/2017/05/cba2016-0978.pdf> Acesso em : 05/11/21.

[6] Borges, A.K., Buriola, J.F.L., Eloi,J.,Teles, R.F., SmartChair: Cadeira de rodas controlada por voz. 2015. Disponível em: <http://paginapessoal.utfpr.edu.br/gustavobborba/if66j-s71-projetos/files/IF66J-15a_RT_SmartChair.pdf> Acesso em : 05/11/21.

[7] SILVA, Bruno Castro da; LIBARDI, Priscila Ortolan. Sistema de controle de um VANT. 2016. xi, 67 f., il. Trabalho de conclusão de curso (Bacharelado em Engenharia Elétrica)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/15956> Acesso em : 05/11/21.

[8] ORRICO, Marcos Vinícius de Melo. Procedimento para seleção de motor e bateria para veículo elétrico. 2013. xi, 102 f., il. Trabalho de conclusão de curso (Bachare-

lado em Engenharia Mecatrônica)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/8232> Acesso em : 05/11/21.

[9] Huinfinito, Arduino & Acessórios: Chassis (Plataformas) . Disponível em: <https://www.huinfinito.com.br/61-chassis-plataformas> Acesso em : 05/11/21.

[10] LIMA, F.G.,SANTOS,R.N.,COSTA,S.O.,SILVA,.V.H.B.S.S,. PROJETO DE CADEIRA DE RODAS MOTORIZADA EQUIPADA COM COBERTURA AUTOMÁTICA. 2016. 97 f. Trabalho de conclusão do curso (Técnico em Eletromecânica)—Instituto Federal de Educação, Ciência e Tecnologia da Bahia, Santo Amaro. Disponível em: <http://www.ifba.edu.br/professores/elvio/tcc/TCC-Cadeira-de-Rodas-Motorizada.pdf> Acesso em : 05/11/21.

[11] Stone, et.al. User Interface Design and Evaluation, p445. Disponível em: <https://books.google.com.br/books?id=VvSoyqPBPbMC&lpg=PR21&ots=d8L-M0oMM6&dq=interface design&lr&hl=pt-BR&pg=PA445#v=onepage&q&f=false>. Acesso em 15/11/2021.

[12] IBGE, Pesquisa Nacional de Saúde, 2013. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv94522.pdf> Acesso em : 05/11/21.

[13] IBGE, Censo Demográfico, 2010. Disponível em: <biblioteca.ibge.gov.br/visualizacao/periodicos/94/cd_2010_religiao_deficiencia.pdf> Acesso em : 05/11/21.

[14] IBGE, Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal, 2017. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/livros/liv101631_informativo.pdf> Acesso em : 05/11/21.

[15] IVO, Regina Marcela. Sistema de controle de cadeira de rodas motorizada para usuários portadores de tetraplegia. 2017. 90 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/20103> Acesso em : 05/11/21.

[16] ABNT NBR 9050: Acessibilidade a edificações, mobiliário, espaços e equipamentos urbanos. Disponível em : http://www.sedhast.ms.gov.br/wp-content/uploads/2015/01/ABNT-NBR9050-2004-Acessibilidade.pdf.

[17] Poole, A., Ball, L.J.,Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects. 2010. Disponível em: <http://www.alexpoole.info/blog/wp-content/uploads/2010/02/PooleBall-EyeTracking.pdf> Acesso em : 05/11/21.

[18] Birchfield, S.,Elliptical Head Tracking Using Intensity Gradients and Color Histograms. In: IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, California. 1998. Disponível em: <citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.2696&rep=rep1&type=pdf> Acesso em : 05/11/21.

[19] SOUZA, Estefânia; GUTEMBERG, Luciano Mendes de. Breck App. 2018. 105 f., il. Trabalho de Conclusão de Curso (Bacharelado em Comunicação Social)—Universidade de Brasília, Brasília, 2018. Disponível em: <https://repositorio.ufpe.br/bitstream/123456789/31884/1/LIMA, Marina Fernandes de Souza.pdf> Acesso em : 18/11/21.

[20] SOUZA, Estefânia; GUTEMBERG, Luciano Mendes de. Breck App. 2018. 105 f., il. Trabalho de Conclusão de Curso (Bacharelado em Comunicação Social)—Universidade de Brasília, Brasília, 2018. Disponível em: <https://bdm.unb.br/handle/10483/22512> Acesso em : 05/11/21.

[21] Tom Scott, It’s pronounced GIF. Disponível em: <https://youtu.be/N1AL2EMvVy0> Acesso em : 05/11/21.

[22] TechTerms, GIF Definition. Disponível em: <https://techterms.com/definition/gif> Acesso em : 05/11/21.

[23] MORAES, André Luis Krueger de. Projeto de um equipamento de reconhecimento de comandos de voz. 2007. 40 f. Monografia (Bacharelado em Engenharia Elétrica)-Universidade de Brasília, Brasília, 2007. Disponível em: <https://bdm.unb.br/handle/10483/19258> Acesso em : 15/11/21.

[24] ALVES, Rafael José; GOULART, Vítor Schwingel. Desenvolvimento de aplicativos para dispositivos móveis com reconhecimento de voz em português. 2014. vi, 46 f., il. Monografia (Bacharelado em Engenharia de Redes de Comunicação)—Universidade de Brasília, Brasília. Disponível em: <https://bdm.unb.br/handle/10483/13525> Acesso em : 05/11/21.

[25] Flávio Renato, A história dos telefones celulares, 2012. Disponível em: <https://www.techtudo.com.br/artigos/noticia/2012/06/historia-dos-telefones-celulares.html> Acesso em : 05/11/21.

[26] PANTOJA, Cássio Rodrigues; GARCIA, Paulo Vitor de Araújo. Sistema de controle de VANT com emprego de plataforma inercial. 2017. xii, 124 f., il. Trabalho de conclusão de curso (Bacharelado em Engenharia Mecatrônica)—Universidade de Brasília, Brasília, 2017. Disponível em: <https://bdm.unb.br/handle/10483/19258> Acesso em : 05/11/21.

[27] Appinventor. Plataforma para programação de aplicativos. Disponível em: <https://appinventor.mit.edu/> Acesso em : 05/11/21.

[28] Google, Princípios básicos da Speech-to-Text. Disponível em: <https://cloud.google.com/speech-to-text/docs/basics?hl=pt-br>.

[29] Grupo de Teleinformática e Automação, O que é RFID. Disponível em: <https://www.gta.ufrj.br/grad/07_1/rfid/RFID_arquivos/o que e.htm> Acesso em : 05/11/21.

[30] Cisco, O que é Wi-Fi?. Disponível em: <https://www.cisco.com/c/pt_br/products/wireless/what-is-wifi.html> Acesso em : 05/11/21.

[31] Marlon Câmara, Bluetooth: O que é e como funciona. Disponível em: <https://www.techtudo.com.br/artigos/noticia/2012/01/bluetooth-o-que-e-e-como-funciona.html> Acesso em : 05/11/21.

[32] Pedro César Tebaldi Gomes, Conheça os principais protocolos de rede e seus usos. Disponível em: <https://www.opservices.com.br/protocolos-de-rede/> Acesso em : 05/11/21.

[33] Cristiano Bertulucci Silveira, Sensor Capacitivo : O que é e como funciona?. Disponível em: <https://www.citisystems.com.br/sensor-capacitivo/> Acesso em : 05/11/21.

[34] Cristiano Bertulucci Silveira, Sensor Indutivo: O que é e como funciona?. Disponível em: <https://www.citisystems.com.br/sensor-indutivo> Acesso em : 05/11/21.

[35] Cristiano Bertulucci Silveira, Sensor Óptico: Como Funciona?. Disponível em: <https://www.citisystems.com.br/sensor-optico/> Acesso em : 05/11/21.

[36] Cristiano Bertulucci Silveira, Sensor Ultrassônico: 10 Aplicações Para a Indústria. Disponível em: <https://www.citisystems.com.br/sensor-ultrassonico/> Acesso em : 05/11/21.

[37] Elijah J. Morgan, HCSR04 Ultrasonic Sensor. Disponível em : https://pdf1.alldatasheetpt.com/datasheet-pdf/view/1132203/ETC2/HC-SR04.html.

[38] Ogata, Kafsutuko. Modern Control Engineering. 4th ed. Prentice Hall (New Jersey): Aeeizh, 2002.

[39] MACHADO, MÁRCIO BENDER. UM MONITOR DO ESTADO DE CARGA DA BATERIA DE DISPOSITIVOS ELETRÔNICOS IMPLANTÁVEIS. 2006. 68p. Dissertação (Mestrado em Engenharia Elétrica). Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/88685/237658.pdf> Acesso em : 05/11/21.

[40] Borges, Felipe Andrade Allemand. Implementação de sistema de acionamento para plataformas robóticas móveis com ênfase em estimador de estado de carga de bateria. 2014. 120p. Dissertação (Mestrado em Engenharia Elétrica). Universidade Estadual de Londrina, Londrina. Disponível em: <http://www.uel.br/pos/meel/disserta/2014_FELIPE_A_ALLEMAND.pdf> Acesso em : 05/11/21.

[41] Pura Vida Apps. App Inventor Extensions : Battery Manager Extension. Disponível em: <https://puravidaapps.com/battery.php>.

[42] Paulo Sérgio Pereira da Silva, Formas Canônicas e Teoria da Realização. Disponível em: <http://www.lac.usp.br/ paulo/cap4_8.pdf>.

[43] Youtube. Plataforma de compartilhamento de vídeos. Disponível em: <https://www.youtube.com/> Acesso em : 19/11/21.

Anexo I

Variação da Inteface

Após os testes de desempenho, foi feita uma variação da interface originalmente desenvolvida. Ela pode ser visualizada nas figuras 57 e 58.

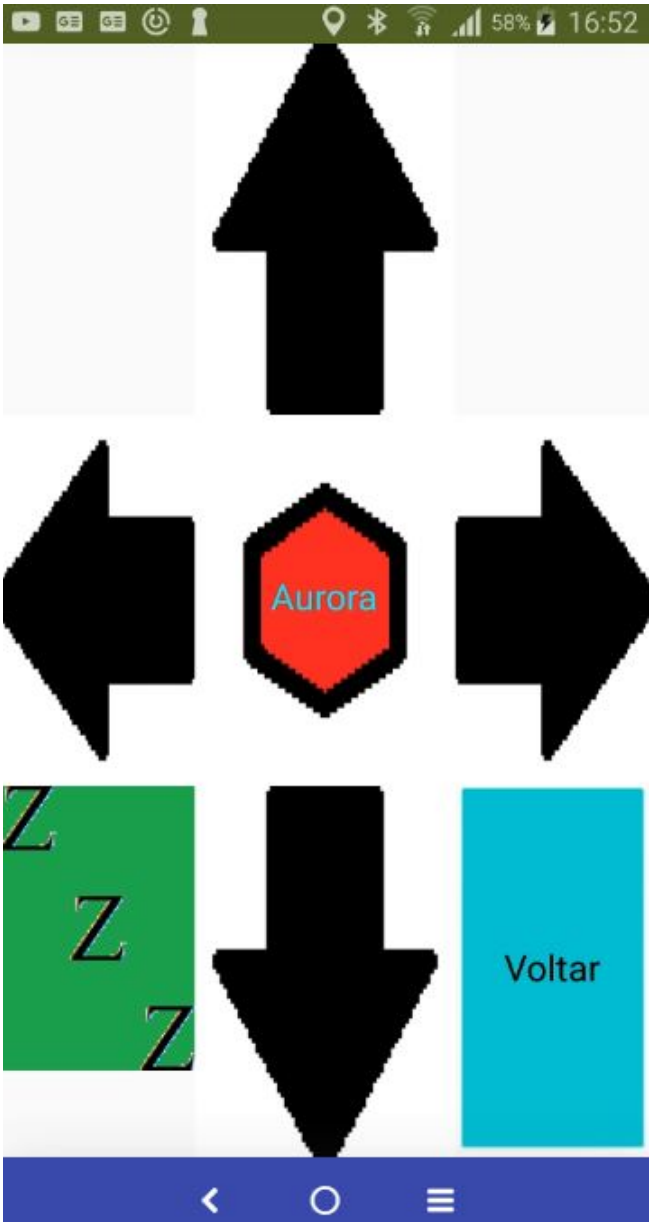


Figura 57: Segunda Interface no modo ’Dorme’

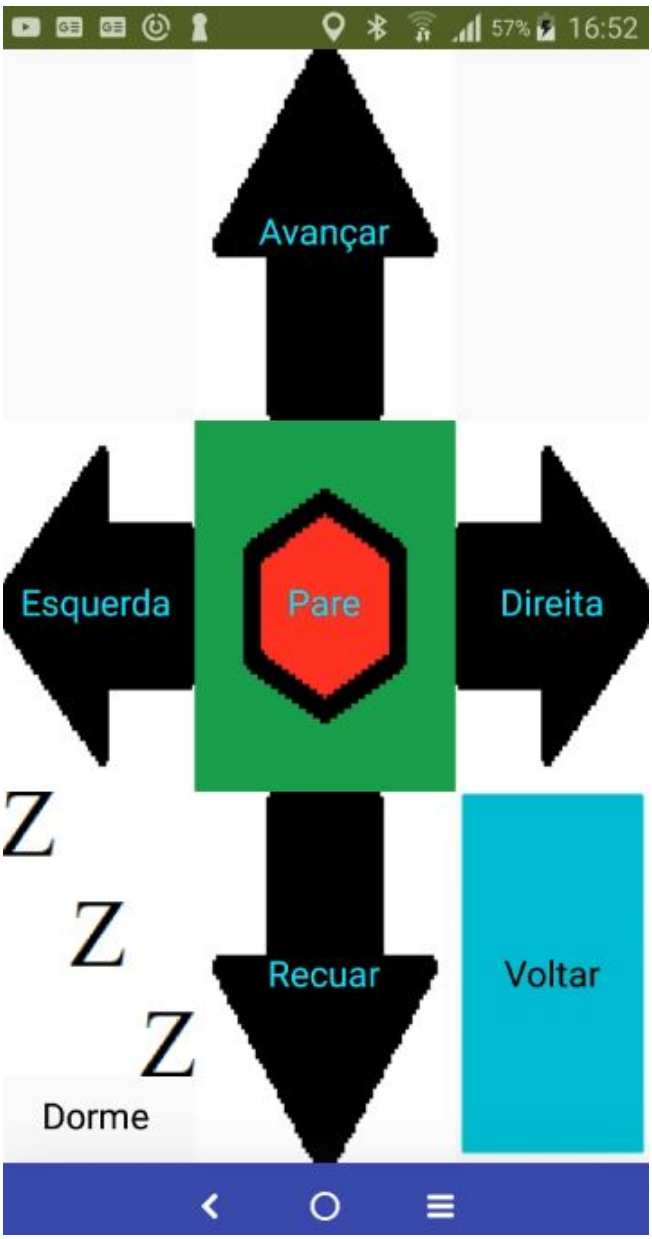


Figura 58: Segunda Interface no modo ’Dorme’

Ela conta com uma fonte maior para facilitar a leitura e, inspirada em 9, apresenta os comandos válidos na tela para facilitar a aprendizagem. Em contrapartida, o aumento no número de elementos pode assustar a um novo usuário, anulando o principal propósito da mudança.

Funções de transferência dos motores

Tabela 15: Funções de transferência estimadas

Velocidade(EF)	Velocidade(ET)	Velocidade(DF)	Velocidade(DT)
$\frac{0.15827}{z-0.5835}$	$\frac{0.14648}{z-0.6048}$	$\frac{0.19107}{z-0.5129}$	$\frac{0.16571}{z-0.5842}$
Posição(EF)	Posição(ET)	Posição(DF)	Posição(DT)
$\frac{0.0062932(z+0.836)}{(z-1)(z-0.5835)}$	$\frac{0.0057926(z+0.846)}{(z-1)(z-0.6048)}$	$\frac{0.0077442(z+0.8011)}{(z-1)(z-0.5129)}$	$\frac{0.0065875(z+0.8363)}{(z-1)(z-0.5842)}$

Clique em 'Retorno' para retornar à menção desta tabela no documento.

Gráficos dos outros Controladores Testados

Aqui estão apresentados os gráficos de testes na tabela 9 que não serviram para o desenvolver do raciocínio do projeto. Clique em 'Retorno' para retornar ao documento.

Proporcional-Integral Por Velocidade Sem Cancelamentos

Projetado com diretamente via polinômio característico em malha fechada para obter um atraso de 2 tempos de amostragem até establizar, esperava-se que essa versão acompanhasse à referência com menores valores na ação de controle visto que há uma folga maior de tempo, mas foi observado o contrário.

$$C = \frac{1}{N(z-1)} + \frac{1+P}{N} \tag{21}$$

$$CEF = \frac{10.0048z - 3.68667}{z-1} \tag{22}$$

$$CET = \frac{10.9562z - 4.12914}{z-1} \tag{23}$$

$$CDF = \frac{7.9183z - 2.6846}{z-1} \tag{24}$$

$$CDT = \frac{9.5604z - 3.5257}{z-1} \tag{25}$$

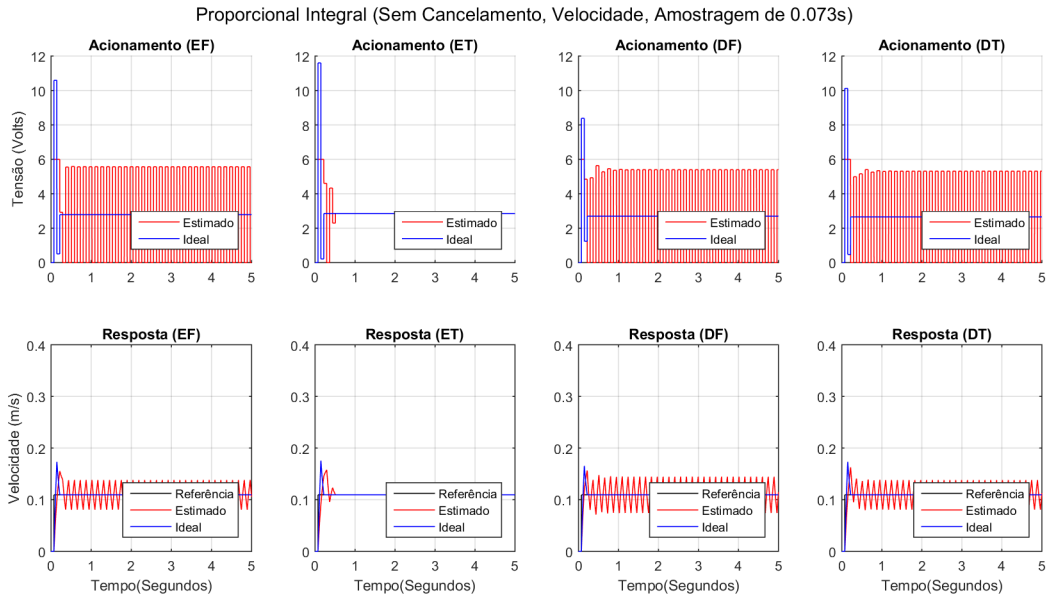


Figura 59: Simulação do PI sem cancelamentos

Visto que o resultado (59) não acompanhou à referência quando as não-linearidades era estimadas, esse controlador não foi testado.

Proporcional-Integral Por Velocidade Com Cancelamentos

Projetado também diretamente no polinômio característico, esse projeto realiza o cancelamento de um dos pólos do sistema com um zero. Esperava-se um controlador com o mesmo resultado porém mais simples que o Deadbeat tipo 1 normal, mas foi obtido exatamente o mesmo controlador.

$$C = \frac{1-P}{N(z-1)} + \frac{1}{N} \tag{26}$$

$$CEF = \frac{6.3182z - 3.6867}{z-1} \tag{27}$$

$$CET = \frac{6.827z - 4.1291}{z-1} \tag{28}$$

$$CDF = \frac{5.2338z - 2.6846}{z-1} \tag{29}$$

$$CDT = \frac{6.0348z - 3.5257}{z-1} \tag{30}$$

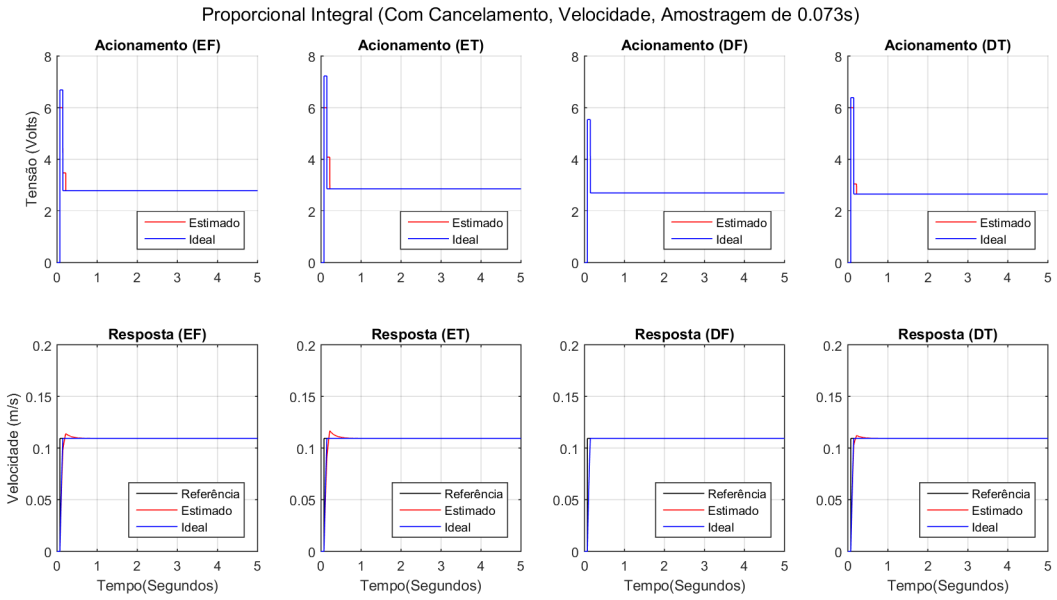


Figura 60: Simulação do PI com cancelamentos

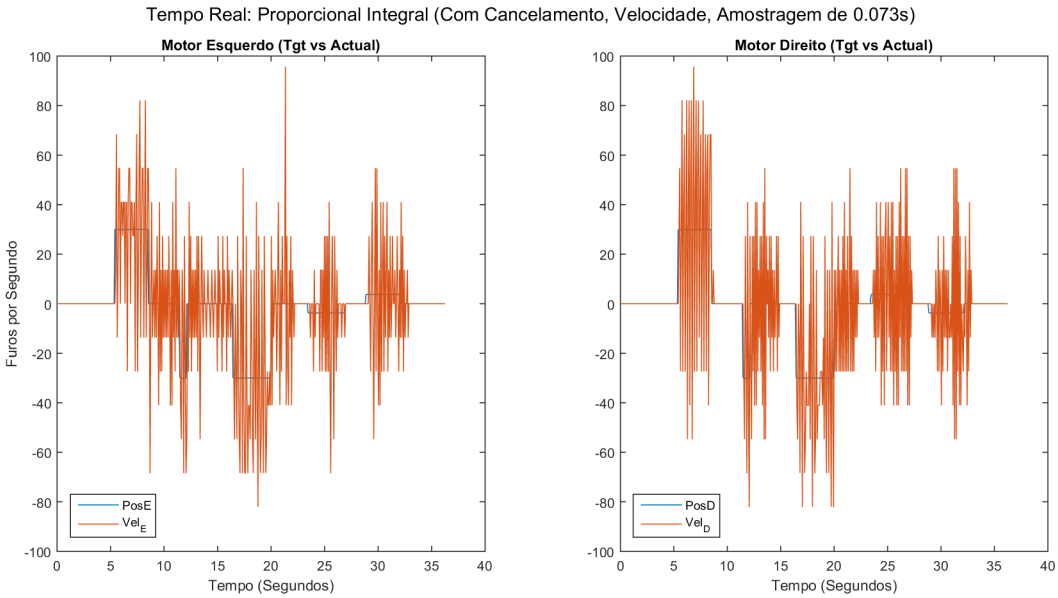


Figura 61: Teste Suspenso do PI com cancelamentos

Por ser idêntico, ao deadbeat tipo 1 normal(60), optou-se por não apresentar esse controlador.

Deadbeat Tipo 2 Normal por posição

Antes de determinar a especificação de parada, foi testado um deadbeat tipo 2.

$$C = \frac{1}{G}\frac{2z-1}{(z-1)^2} \tag{31}$$

$$CEF = \frac{317.804z^2 - 344.3417z + 92.71986}{z^2 - 0.16405z - 0.83595} \tag{32}$$

$$CET = \frac{345.2708z^2 - 381.4625z + 104.4135}{z^2 - 0.15405z - 0.84595} \tag{33}$$

$$CDF = \frac{258.2568z^2 - 261.597z + 66.23432}{z^2 - 0.19893z - 0.80107} \tag{34}$$

$$CDT = \frac{303.6053z^2 - 329.1778z + 88.68756}{z^2 - 0.1637z - 0.8363} \tag{35}$$

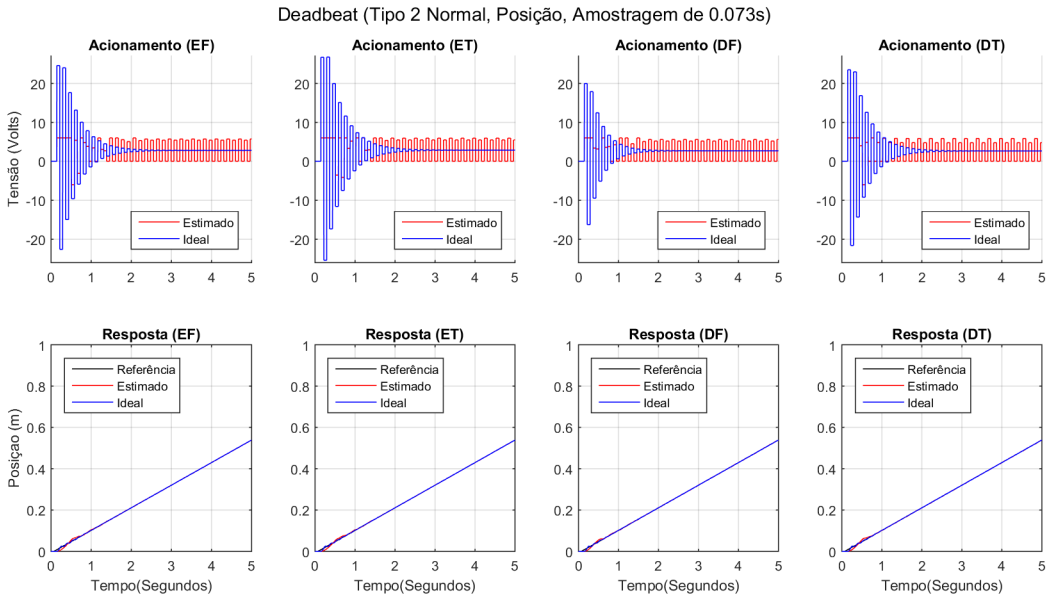


Figura 62: Simulação do DeadBeat Tipo 2

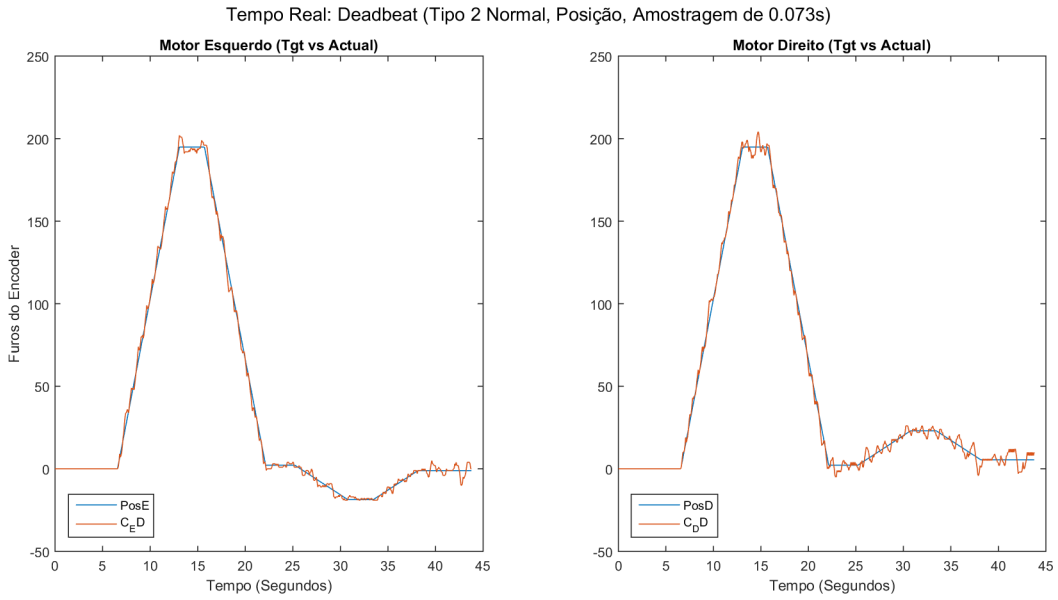


Figura 63: Teste Suspenso do DeadBeat Tipo 2

Visto que ele não é capaz de permanecer parado (63), ele não foi utilizado.

Deadbeat Tipo 2 Amortecido por posição

Similar ao apresentado no Deadbeat tipo 1, foi feita uma versão mais lenta do tipo 2, com pólos extras. Embora ele tenha uma excursão de sinal menor, ele também apresenta (65) os mesmos problemas do anterior.

$$C = \frac{1}{G}\frac{1.1z^3 + 0.5975z^2 - 0.495z - 0.2025}{(z-1)^2(z+0.45)^2} \tag{36}$$

$$CEF = \frac{174.7922z^4 - 7.047889z^3 - 134.0566z^2 + 13.71867z + 18.77577}{z^4 + 0.73595z^3 - 0.7811z^2 - 0.78558z - 0.16928} \tag{37}$$

$$CET = \frac{189.8989z^4 - 11.70525z^3 - 147.8416z^2 + 16.72604z + 21.14374}{z^4 + 0.74595z^3 - 0.7821z^2 - 0.79255z - 0.1713} \tag{38}$$

$$CDF = \frac{142.0412z^4 + 4.296462z^3 - 103.4936z^2 + 6.637489z + 13.41245}{z^4 + 0.70107z^3 - 0.77761z^2 - 0.76125z - 0.16222} \tag{39}$$

$$CDT = \frac{166.9829z^4 - 6.854233z^3 - 128.1331z^2 + 13.16031z + 17.95923}{z^4 + 0.7363z^3 - 0.78113z^2 - 0.78582z - 0.16935} \tag{40}$$

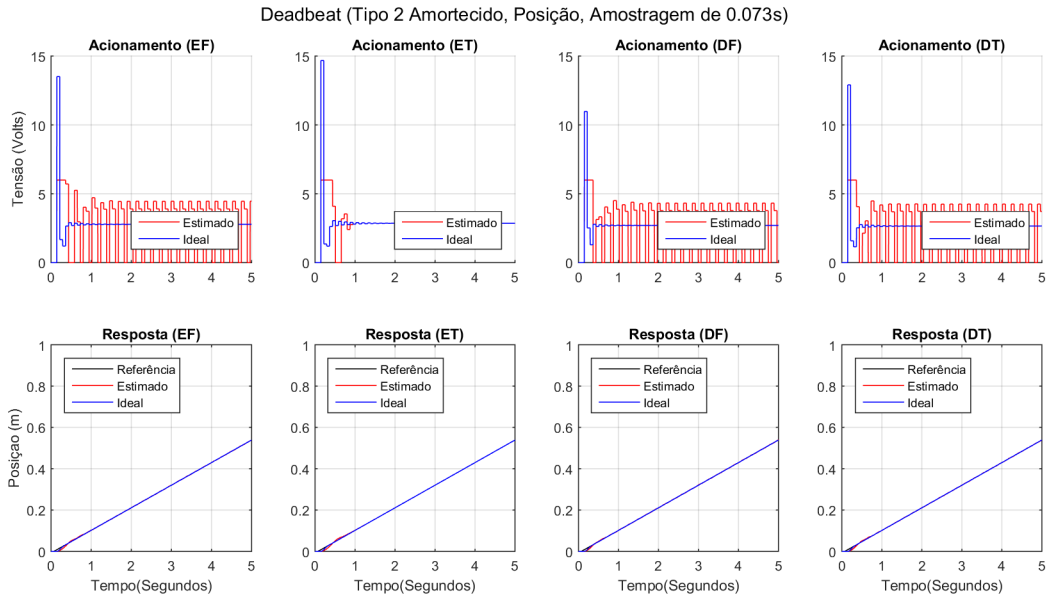


Figura 64: Simulação do DeadBeat Amortecido Tipo 2

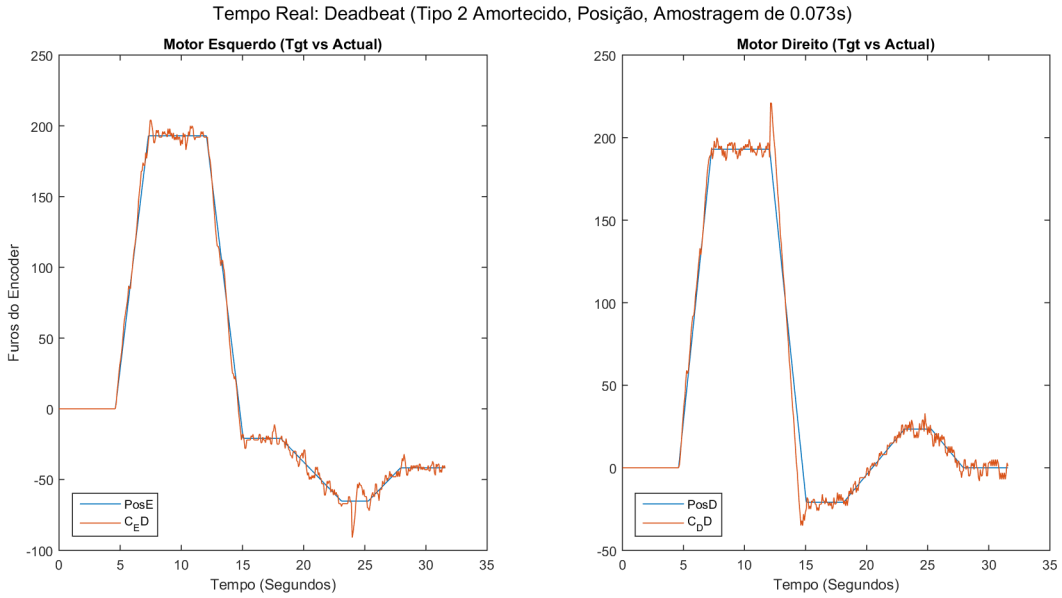


Figura 65: Teste Suspenso do DeadBeat Amortecido Tipo 2

Compensador em atraso

Feito após a escolha do controlador proporcional, o compensador em atraso buscou reduzir o erro em regime permanente.

$$C = \frac{K(z - A)}{(z - P)} \tag{41}$$

$$CEF = \frac{8.5023z - 7.652}{z - 0.93333} \tag{42}$$

$$CET = \frac{8.2074z - 7.3866}{z - 0.93333} \tag{43}$$

$$CDF = \frac{9.8948z - 8.9053}{z - 0.93333} \tag{44}$$

$$CDT = \frac{8.0905z - 7.2814}{z - 0.93333} \tag{45}$$

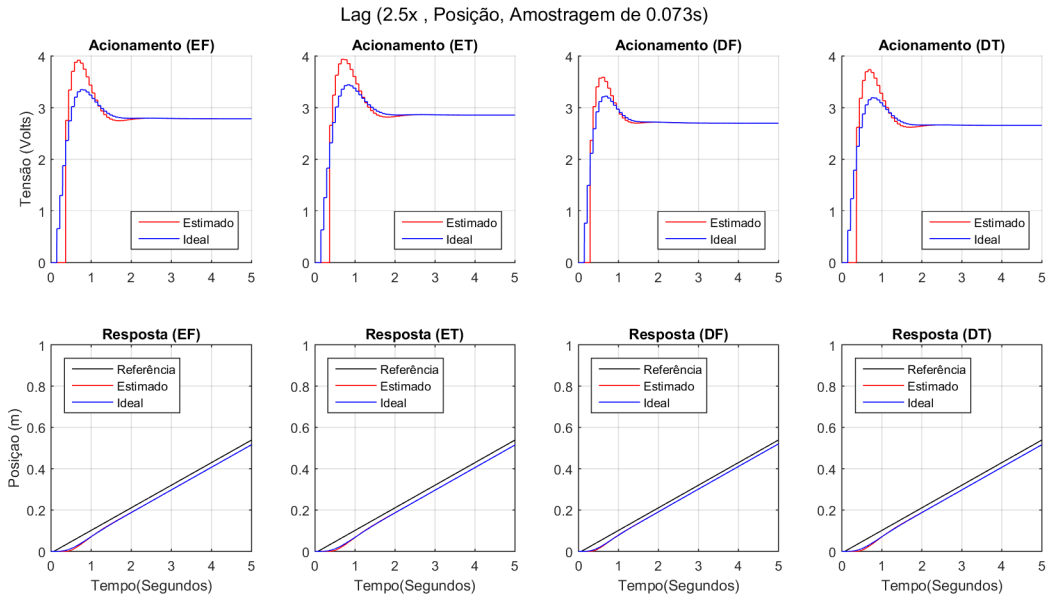


Figura 66: Simulação do Atraso

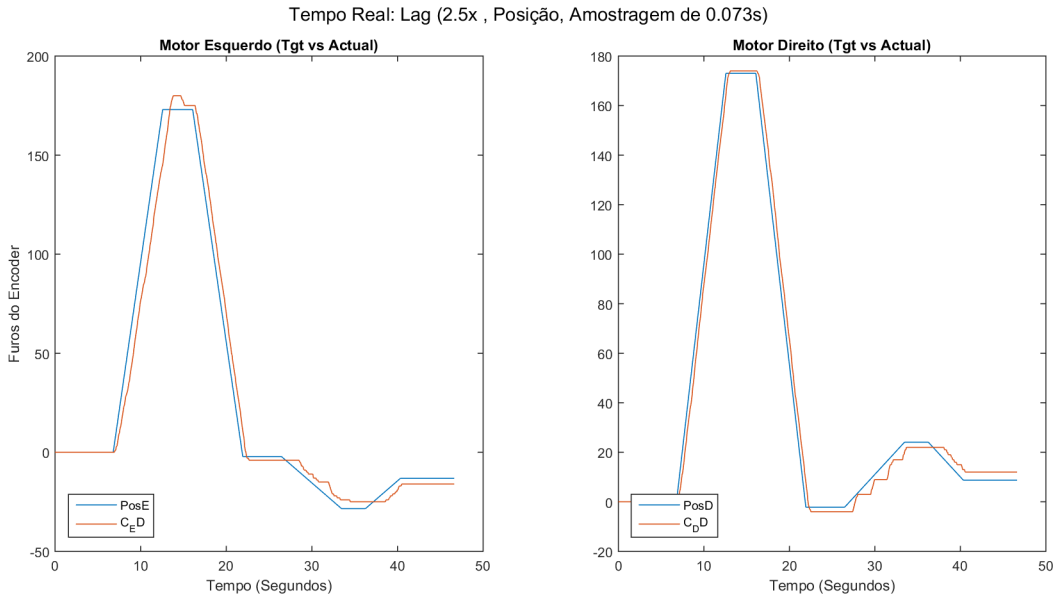


Figura 67: Teste Suspenso do Atraso

Mesmo apresentando um resultado razoável (67), ocasionalmente, ele partia o carrinho do repouso e portanto não foi utilizado.

Compensador em avanço

Também feito após a escolha do proporcional, ele foi projetado para tempo mínimo para a referência de posição, similar aos PIs apresentados anteriormente, mas sem um canal integral.

Por ter sido feito após os outros controles, o ganho foi reduzido pela metade para reduzir a excursão do sinal e controle.

$$C = \frac{0.5}{GK(1 - GN_1)} * \frac{z - GD_2}{z - \frac{GN_1}{1 - GN_1}} \tag{46}$$

$$CEF = \frac{43.2751z - 25.2512}{z + 0.45532} \tag{47}$$

$$CET = \frac{46.7606z - 28.2818}{z + 0.45827} \tag{48}$$

$$CDF = \frac{35.8477z - 18.3875}{z + 0.44477} \tag{49}$$

$$CDT = \frac{41.3339z - 24.1485}{z + 0.45543} \tag{50}$$

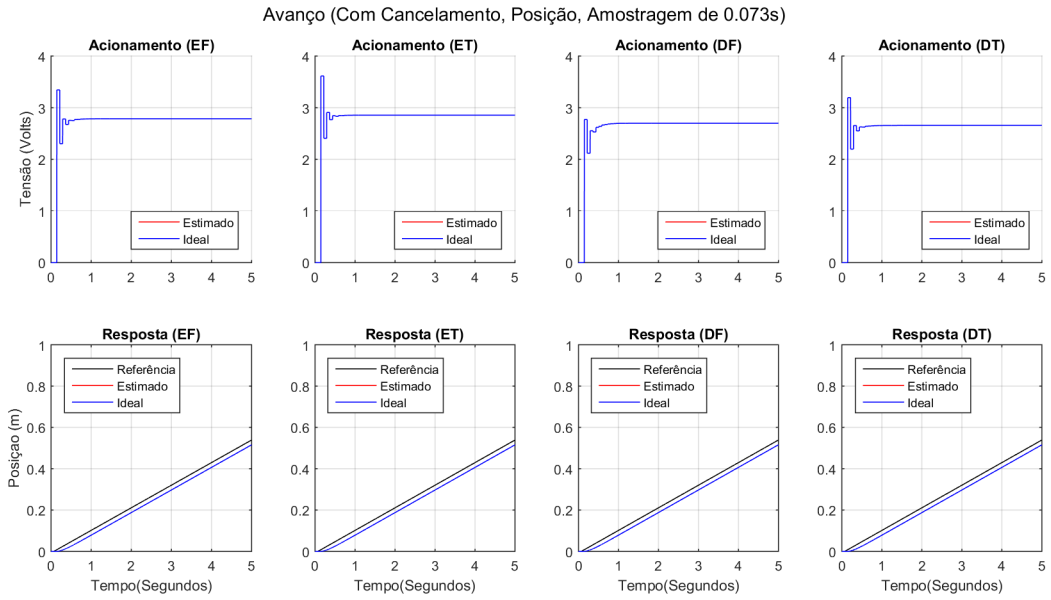


Figura 68: Simulação do Avanço

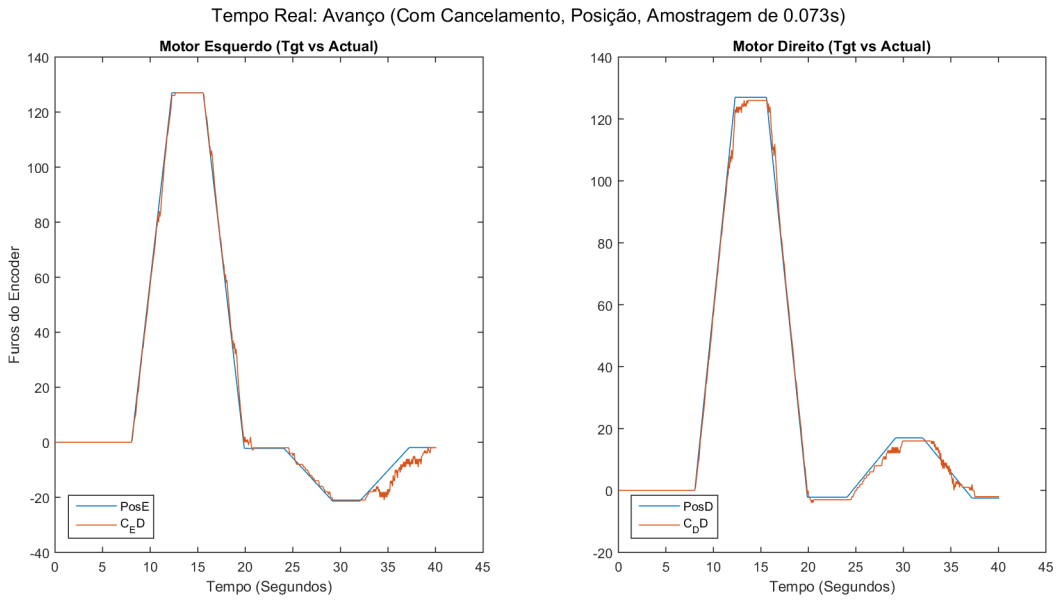


Figura 69: Teste Suspenso do Avanço

Embora tenha iniciado bem, ele não acompanhou apropriadamente o sinal de controle (69) e por isso não foi utilizado.

Formas de Simulação

O diagrama do Simulink utilizado para verificar a realização é apresentado na figura 70:

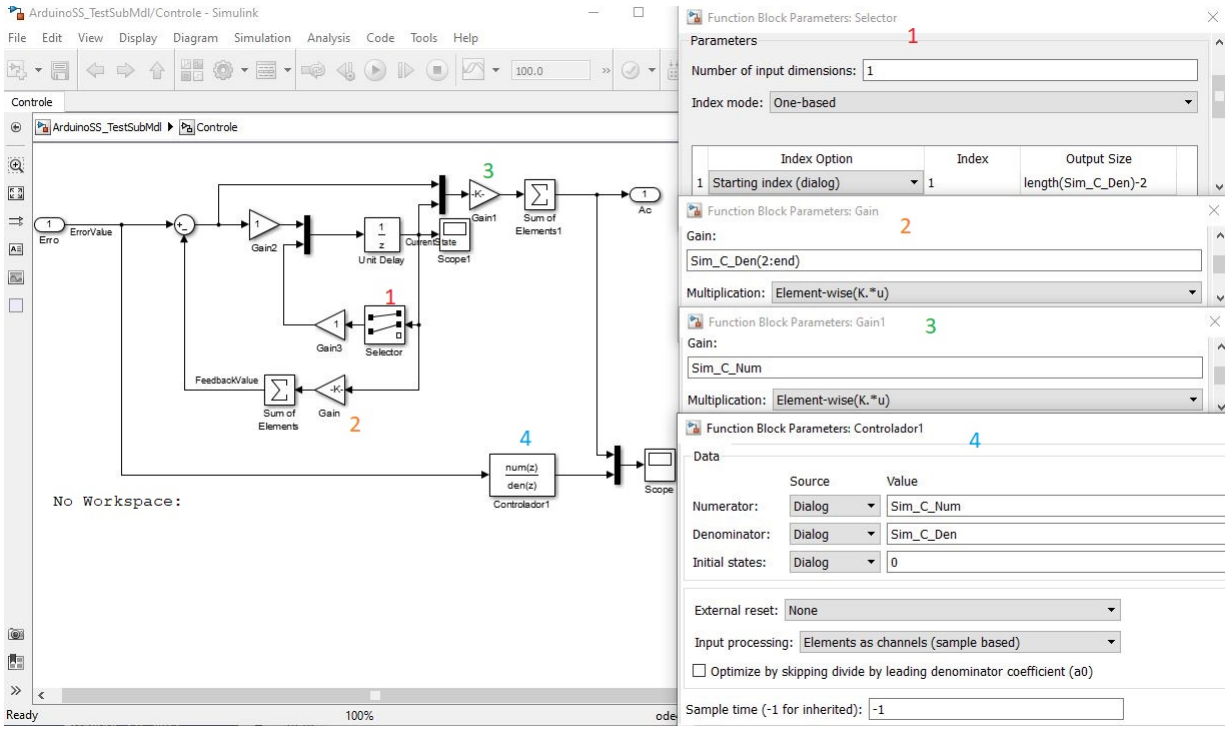


Figura 70: Realização da Simulação dos controladores no Simulink

Configurado e testado no workspace da forma a seguir:

```
1 [Sim_C_Num,Sim_C_Den] = tfdata(CDF,'v');
2 [Sim_C_Num,Sim_C_Den] = Custom_MatchVecs(4,0,Sim_C_Num,Sim_C_Den);
3 sim('ArduinoSS_TestSubMdl.slx',Duracao)
```

A função na linha 5 extende o tamanho dos vetores para um mínimo de 4 termos, inserindo zeros ao fim dos polinômios que não afetam a tradução mas evitam erros de execução no simulink para controladores menores.

Apêndice I

Códigos do MatLab

Instruções Gerais de Uso

O código desenvolvido no MatLab deve servir de apoio para desenvolvimentos futuros de sistemas de controle no Arduino e pode ser adaptado para outras funções com a escrita de textos. Nesta seção, será explicado o básico de como utilizar o código no processo de identificação do sistema do Arduino.

No script Arduino_SS_SimS, configure a variável RunMain para -2 (Próximo à linha 25). Em seguida execute o script. Assim que o mesmo encerrar, utilize o arquivo criado Motores_Bluetooth_ML.ino e envie o programa ao Arduino. Prepare o robô para movimentação, emparelhando os dispositivos bluetooth e execute o script ArduinoSS_DT1b. O móvel deve se mover para frente até obter 100 leituras e em seguira para trás, até outras 100 leituras. Será criado um arquivo com o nome em Tmp1 (linha 118) com vetores de tempo e posição observados.

Uma vez obtidos esses dados, as variáveis PosData_Identify_F e PosData_Identify_T estarão preenchidas e o processo de obter as funções de transferência pode ser realizado executando mais uma vez o script Arduino_SS_SimS.

Para a escolha do controlador, tem-se 2 variáveis: PosCon e CVer (linhas 42 e 43). A primeira escolhe se o sistema deve acompanhar a posição ou a velocidade do móvel, a segunda escolhe qual controlador será utilizado. Para propósitos de comparação e teste, a opção de $CVer = 0$ irá desativar o controle do sistema e ele sempre aplicará a máxima tensão permitida aos motores, de 6v nesta implementação (modificável em Arduino_SS_ConfigGeral na variável MaxVel onde $V = MaxVel * \frac{9}{255}$).

Os nomes nos gráficos dos controladores são configurados de acordo com o controlador escolhido e são preparados no script Arduino_SS_GeraControle nas linhas 61 a 139. Para adicionar um novo nome, basta adicionar um novo Case ao Switch ali presente. Para adicionar um novo controle, deve-se editar a função Temp_Identifier, na qual o Switch que inicia na linha 80 faz o controlador anteriormente determinado, e adicionar um novo Case correspondente ao adicionado ao nome e programar a lógica de projeto dentro desse mesmo Case. Os scripts esperam um controlador discreto e não contínuo.

O programa permite adicionar e deletar variáveis globais declaradas no Arduino no script Arduino_SS_GeraSetup, basta adicionar uma linha no formato:

```
VarsProg = [VarsProg ;...
{'<Tipo da Variável>','<Nome da Variável>',...
<Valor(es) Inicial(is)>,VarsProg{end,4}+1};
```

Embora não hajam restrições de onde adicionar/remover variáveis, é preferencial que linhas adicionadas/removidas estejam próximas do final do código, visto que alterações nos índices de variáveis monitoradas (presentes no vetor ArduinoSS_DT1RTVecSetup) afetarão as funções de apresentação de dados.

Falando nisso, as linhas 62 a 71 do Arduino_SS_SimS configuram no vetor mencionado anteriormente os índices das variáveis em VarsProg que serão mostradas nos Serial.print() do Arduino, podendo ser modificadas à vontade para apresentar quaisquer variável de interesse para observação ou debugging.

Outras configurações importantes de Arduino_SS_SimS são:
GenerateFiles <= Decide se vai ou não gerar arquivos ao executar.
Simulate <= Decide se vai ou não simular os controladores no simulink ao executar.
S_BaudRate <= Guarda o BaudRate do canal Serial do Arduino.
B_BaudRate <= Guarda o BaudRate do canal Bluetooth do Arduino.

O programa do Arduino consiste de vários arquivos .h e diretivas de #include. Esses arquivos são gerenciador por uma variável 'FileCell'. Para adicionar um novo arquivo, basta adicionar em qualquer local do programa, uma nova entrada na variável no seguinte formato:

```
FileCell = {'<Pasta Alvo>','<Nome Do Arquivo>.h',NFiles+1};
```

E incrementar o contador de arquivos NFIles. Em uma implementação futura, deve-se substituir NFiles+1 por FileCellend,3+1 por facilidade de uso. O restante do processo já foi descrito na seção 3.2.

Tabela De Navegação

Para facilitar a leitura online do código, as funções serão apresentadas em seções distintas, as quais podem ser acessadas pela tabela 16.

Tabela 16: Tabela de navegação dos códigos

Função ou Script	Breve Descrição de funcionalidade
Arduino_SS_SimS	Script principal de configuração. Chama a maioria das outras funções apresentadas no relatório.
Temp_Identifier	Faz a identificação dos motores e o cálculo dos controladores.
Arduino_SS_DT1b	Se conecta ao Arduino por bluetooth para coleta de dados da identificação.
Arduino_SS_GeraControle	Gera os arquivos para realizar o controle no Arduino
Arduino_SS_GeraSetup	Escreve a maioria das operações que serão realizadas pelo Arduino antes do Loop.
Arduino_SS_GeraAcionamento	Escreve as funções que se acionam os motores.
Arduino_SS_GeraPosicao	Escreve o arquivo que gera a referência do sistema de controle robô.
Arduino_SS_MLPrintFiles	Faz o código que envia no canal serial as variáveis indicadas pelo script principal.
Arduino_SS_GenerateData	Escreve os arquivos de modo de funcionamento
Arduino_SS_GeraBluetooth	Faz as funções do Arduino que usam o Bluetooth.
Arduino_SS_GeraSensores	Gera a função que utiliza os sensores de proximidade.
Arduino_SS_ConfigGeral	Escreve arquivos e faz configurações que não encaixam nos outros scripts.
Arduino_SS_GeraTestes	Faz códigos de teste do Arduino.
Arduino_SS_MoveFiles	Move os arquivos do Arduino para as respectivas pastas.
Arduino_SS_DT1	Faz a comunicação do Arduino com o Matlab.

Funções Auxiliares

Conjunto de funções desenvolvido durante o trabalho para uso geral.

1. Custom_Files_SeqAssist

   Esta função foi feita para simplificar operações com vetores. Por conveniência, ela adiciona uma quebra de linha ao final da string que receber de argumento.

```
function [Str] = Custom_Files_SeqAssist(varargin)
%   [Str] = Custom_Files_SeqAssist(Str,NumVec)
%   [Str] = Custom_Files_SeqAssist(Str,NumVec,...)
%
%   By Alexandre Aragão
%
%   Takes a group of strings and numeric arrays and joins them into strings
%   It returns a line for each value in the largest numeric input array
%   Returns a single line in case there are no numeric arrays
%
%   If there are many numeric arrays, the smaller ones will have their last
%   value repeated. For automation purposes, if one of the input arrays is
%   empty then the function also returns an empty array.
%
%   The function also adds a ' \n' (space and a line break) at the end of
%   every line
%
%   Examples:
%   Custom_Files_SeqAssist('A','A')            Returns 'AA \n'
%   Custom_Files_SeqAssist('A',2,'A',1)        Returns 'A2A1 \n'
%   Custom_Files_SeqAssist('A',1,[2 1],'A')    Returns 'A12A \nA11A \n'
%   Custom_Files_SeqAssist('A',[2 1],[3 2 1]) Returns 'A23 \nA12 \nA11 \n'
%   Custom_Files_SeqAssist('A',[],[3 2 1])     Returns [](Empty Array)

Str = [];
M = 1;
Empty = 0;
Nums = zeros(1,nargin);
if ~isempty(varargin)
    %Vamos pegar o valor do tamanho do maior vetor numérico
    for N = 1:nargin
        if ~isempty(varargin{N})
            Nums(N) = isnumeric(varargin{N});
            if Nums(N)
                if M < length(varargin{N})
                    M = length(varargin{N});
                end
            end
        else
            Empty = 1;
        end
    end
    %Gera as strings por concatenação
```

```
44      for N = 1:M
45          for P = 1:length(Nums)
46              if Nums(P)
47                  V = varargin{P};
48                  Str = [Str num2str(V(min(N,end)))];
49              else
50                  Str = [Str varargin{P}];
51              end
52          end
53          Str = [Str ' \n'];
54      end
55  else
56      Empty = 1;
57  end
58
59  if Empty %Se alguma string estiver vazia, normalmente numéricas, cancela
60      Str = [];
61  end
62  end
```

2. MyParseFloat

Esta função foi feita para traduzir qualquer número representado como uma string para seu valor numérico. Não compreende notações complexas, como potências e caracteres diferentes de números, sinais, vírgulas e pontos.

```
1   function [Num] = MyParseFloat(S)
2   %Parses a string S into a float Num
3   %   Stops on the first NaN character
4   %   [Num] = MyParseFloat(S)
5   %   Alexandre Aragão
6
7   Num = 0;
8   Pos = 1;
9   L = length(S);
10  A = 1;
11  if L == 0
12      Num = 0;
13  else
14      if S(1) == '+'
15          A = 2;
16      else
```

```
17          if S(1) == '-'
18              A = 2;
19              Pos = 0;
20          end
21      end
22      for n = A:L
23          if (((S(n) <  48)|| (S(n) >  57)) && Num == 0)
24              if (S(n) == 44 || S(n) == 46)
25                  ;
26              else
27                  Num = n-1;
28              end
29          end
30      end
31      if Num > 0
32          L = Num;
33      end
34      Num = 0;
35      Comma = 0;
36      B = 0;
37      for n = A:L
38          if (S(n) == ',' || S(n) == '.')
39              if Comma == 0
40                  Comma = n;
41              else
42                  B = B + 1;
43              end
44          else
45              Num = Num*10 + (S(n)-48);
46          end
47      end
48      if Comma > 0
49          Num = Num/(10^(L - Comma - B));
50      end
51
52      Num = Num*(Pos) - Num*(~Pos);
53  end
54  end
```

3. MyStringComparator

   Função feita para comparar strings que foi estendida para buscar elementos em listas de strings.

```
function [varargout] = MyStringComparator(varargin)
%String Comparator
%    [C] = MyStringComparator(A,B)
%    Returns false if the strings have different sizes.
%    Returns in C wether or not A and B are the same string.
%    Returns -1 in case of error.
%    Case-Sensitive
%    By Alexandre Aragão
%
%    Examples:
%    [C] = MyStringComparator('A','a') returns 0
%    [C] = MyStringComparator('A ','A') returns 0
%    [C] = MyStringComparator(' A','A') returns 0
%    [C] = MyStringComparator('A','A') returns 1
%
%    If A is a cell array with strings and B is a string then C will be a
%    column array with the resulting comparation between each element of A
%    and the string B.
%    Example:
%    [C] = MyStringComparator({'A';'a'},'a') returns [0;1]
%
%    If A is a string and B is a cell array then C will return the index of
%    the string in B that is equal to the string A. Also returns 0 if none
%    is found.
%    Example:
%    [C] = MyStringComparator('A',{'a';'A';'b'}) returns 2
%    [C] = MyStringComparator('B',{'a';'A';'b'}) returns 0
%
%    If both A and B are string cells then C will return a column vector
%    with the indexes of B's strings that are the same as in A's lines.
%    Example:
%    [C] = MyStringComparator({'A';'b';'B'},{'a';'A';'b'}) returns [2;3;0]

if nargin == 2 % Ou 2 Strings, ou 1/2 arrays de strings
    if ischar(varargin{1})
        LA = 0;%Entrada 1 "Driver" é uma string
    else
        if iscell(varargin{1})%Entrada 1 "Driver" é um conjunto de strings
            LA =  size(varargin{1},1);
        else
            LA = -1;
        end
    end
```

```
44      if ischar(varargin{2})
45          LB = 0;%Entrada 2 "Target" é uma string
46      else
47          if iscell(varargin{2})%Entrada 2 "Target" é um conjunto de strings
48              LB =  size(varargin{2},1);
49          else
50              LB = -1;
51          end
52      end
53
54      if LA < 0 || LB < 0
55          varargout{1} = -1;% Erro de tipo de entradas invalidas
56      else
57          if LA == 0
58              varargout{1} = 0;
59              if LB == 0
60                  varargout{1} = MyStrCmp(varargin{1},varargin{2});
61              else
62                  for n = 1:LB
63                      if MyStrCmp(varargin{1},char(varargin{2}(n)))
64                          varargout{1} = n;
65                      end
66                  end
67              end
68          else
69              varargout{1} = zeros(LA,1);
70              if LB == 0
71                  for n = 1:LA
72                      varargout{1}(n) = MyStrCmp(char(varargin{1}(n)), ...
                            varargin{2});
73                  end
74              else
75                  for na = 1:LA
76                      for nb = 1:LB;
77                          if ...
                                MyStrCmp(char(varargin{1}(na)),char(varargin{2}(nb)))
78                              varargout{1}(na) = nb;
79                          end
80                      end
81                  end
82              end
83          end
84      end
```

```
85  else
86      varargout{1} = -1;
87  end
88
89  end
```

Arduino_SS_SimS

Retorne à tabela clicando nesta frase

```
1  %Scripts de gerenciamento do Arduino Surveillance System
2  %   Por Alexandre Aragão
3  %% Início
4
5  close all;
6
7  GenerateFiles = 1;%Controla de vai gerar os arquivos do Arduino
8  Simulate = 1;%Realiza as simulações dos controladores
9  S_BaudRate = 115200;%Baudrate do Serial
10 B_BaudRate =  57600;%Baudrate do Bluetooth
11 MaxVel = floor(6*(255/9));%Calcula a tensão máxima segura
12
13 %Temporização
14 T = 0.073;%Tempo de amostragem. 0.073 Esse foi o mais rápido realizável.
15 EncoderDelay = 2;%Tempo de espera para eliminar ruído no encoder
16 Duracao = max([20*T, 5]);%Escolhe uma duração para as simulações
17
18 %Configurações gerais
19 Sim_M = 30;%Velocidade alvo em furos por segundo (máxima de 60)
20 Sim_U = [1 0 0 0 0];%Coeficientes do vetor de suavização para a simulação
21 Sim_TurnAdj = 0.125;%Ajusta a velocidade durante curvas (para 12.5%)
22 DeadZone    =  60;%Valores de tensão que não vencem o atrito (p/ simulação)
23 Saturation  = MaxVel;%Valor de tensão máxima aplicável
24 SkipControl =   0;%Desliga a modificação do controlador (mas deve existir um)
25 RunMain     =   1;%Escolhe qual versão do código será gerado:
26 %  1 -> Comportamento Normal
27 %  0 -> Teste dos sensores
28 % -1 -> Teste dos motores
29 % -2 -> Script de identificação dos motores
30 % -3 -> Script de teste preliminar do controlador
31 UseSensors  =    1;%Liga os sensores (Para debug)
```

```
DumpVars     =     0;%Dá print em todas as variáveis do Arduino, TODAS
TestMode     =     1;%Indica que o robô está suspenso
Run_Chao     =     0;%Tira os prints todos do arduino
VelGrouping = 0.00;%Coeficiente da Media Movel Ponderada. 0 desliga
load('TesteNormal_M2_10_07');

if ¬exist('Auto','var')
    Auto = 0;
end
if Auto == 0
    PosCon   =  1;%Controla por posição (0 para ir por velocidade)
    CVer     = 09;%Escolhe o controlador (Lista abaixo)
end
%   0 - None (For Tests)
%   1 - Type 1 Deadbeat
%   2 - Type 2 Deadbeat
%   3 - Type 1 Dampened Deadbeat
%   4 - Type 2 Dampened Deadbeat
%   5 - Proportional-Integral (DB speed)
%   6 - Proportional-Integral (Normal)
%   7 - Proportional
%   8 - Lag
%   9 - Lead
if CVer == 0
    SkipControl = 1;
end

%Setup dos prints
%O Arduino irá enviar os valores das variáveis em 'VarsProg'
%Com esses índices. O script Arduino_SS_GeraSetup gera o VarsProg
A = [24];%Índice de: Tempo
if 1
    if PosCon
        A = [A 8 7 16 17];%Índice de: PosD/E, C_DD/ED
    else
        A = [A 8 7 31 30];%Índice de: PosD/E, Vel_D/E
    end
end
%A = [A 51 52];%DistT/F
%A = [A 32 33];%Controle_ACD/E
ArduinoSS_DT1RTVecSetup = A;%VarsProg
%Para uma lista, digite VarsProg no terminal após executar este script
%Para adicionar variáve
```

```
%is novas, adicione linhas modificando o script
%Digitando 'open Arduino_SS_GeraSetup' e modificando a variável


%Specs esperados:
%140 RPM a 6v, então em torno de 130 furos por segundo
%Diâmetro exteno com pneu: 65 mm;
%r = 0.0325
%2*pi*r = 0.2042m
%2*pi*r/56 = 0.003646m (distância por furo)
%1.9N de força

%Tradução de furos pra metros
NFuros = 56;%28 tiras, mas ele conta subida e descida
r = 0.0325;
aux1 = 2*pi*r/NFuros;
Sim_M = Sim_M*aux1;

%Gerenciador de arquivos
Folders = {'Motores_Bluetooth_ML','data','modules','subrotinas'};

%% Cria as pastas onde vão ficar os arquivos do Arduino

if GenerateFiles
    %Variável de organização de scripts
    ScriptName = '// Arduino_SS_SimS \n';
    NFiles = 0;
    %Gera as pastas
    [~,~] = mkdir(Folders{1});
    [~,~] = mkdir(Folders{1},Folders{2});
    [~,~] = mkdir([Folders{1},'/',Folders{2}],Folders{3});
    [~,~] = mkdir([Folders{1},'/',Folders{2},'/',Folders{3}],Folders{4});
    %Começa o vetor de arquivos
    FileCell = {'data','Motores_Bluetooth_ML.h',NFiles+1};
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = [ScriptName,...
    '// Seleção de programa@'...
    '#define MainProg ',num2str(int32(RunMain==1)),'@'...
    '// Configurações@'...
    '// Seleção de programa@'...
    '#define SensorTest ',num2str(int32(RunMain==0)),'@'...
    '// Configurações@'...
```

```
118       '// Seleção de programa@'...
119       '#define EngineTest ',num2str(RunMain*(RunMain < 0)),'@'...
120       '// Configurações@'...
121       '//  -1 ->Movimentos;@'...
122       '//  -2 ->Encoders;@'...
123       '//  -3 ->Controle;@'...
124       '//  -4 ->Controle no chão;@'...
125       '// Debugging and Profiling@'...
126       '//#define ML_Motores 0 @'...
127       '//#define ML_Sensores 0@'...
128       '//#define ML_Completo 0@'...
129       '#if MainProg@'...
130       '#include "Mainprog.h"@'...
131       '#endif@'...
132       '#if SensorTest@'...
133       '#include "ML_Sensors.h"@'...
134       '#endif@'...
135       '#if EngineTest < 0@'...
136       '#define EngineTest ',num2str(-RunMain),'@'...
137       '#include "ML_Motores.h"@'...
138       '#endif@'...
139       ];
140       Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
141       fprintf(fid,Tmp1);
142       fclose(fid);
143
144       FileCell = [FileCell ; {'main','Motores_Bluetooth_ML.ino',NFiles+1}];
145       NFiles = NFiles + 1;
146       fid = fopen(FileCell{NFiles,2},'w');
147       Tmp1 = ScriptName;
148       Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
              "data/Motores_Bluetooth_ML.h"')];
149       fprintf(fid,Tmp1);
150       fclose(fid);
151   end
152
153   %% Calcula e gera o controlador
154
155   run Arduino_SS_GeraControle
156
157   %% Gera as subrotinas de acionamento
158
159   run Arduino_SS_GeraAcionamento
```

```
160
161 %% Gera a subrotina de geração de posição
162
163 run Arduino_SS_GeraPosicao
164
165 %% Generate remaining files via auxiliary scripts
166
167 run Arduino_SS_GeraSetup %Faz arquivos com funções anteriores ao loop()
168
169 run Arduino_SS_MLPrintFiles %Faz arquivos de envio de dados
170
171 run Arduino_SS_GenerateData %Faz os 3 arquivos gerais na pasta /data
172
173 run Arduino_SS_GeraBluetooth %Faz arquivos que gerenciam o bluetooth
174
175 run Arduino_SS_GeraSensores %Faz o arquivo que controla os sensores
176
177 run Arduino_SS_ConfigGeral %Arquivos que não encaixam nos outros scripts
178
179 run Arduino_SS_GeraTestes %Gera as subrotinas de teste
180
181 %% Fecha os arquivos abertos
182 if GenerateFiles
183     fclose('all');
184 end
185
186 %% Move os arquivos para as pastas deles
187
188 if GenerateFiles
189     run Arduino_SS_MoveFiles
190 end
191
192 %% Salva um arquivo com as variáveis do workspace
193 %Ele sobreescreve e acaba mantendo uma pra cada dia
194
195 run Arduino_SS_Savefiles
196
197 %pause();
198 %close all;
199 display ScriptEnded
```

Temp_Identifier

Função que identifica a função de transferência dos motores e projeta os controladores do sistema.

Retorne à tabela clicando nesta frase

```
function [GDisc1,GCont1,GDisc2,GCont2,C] = ...
    Temp_Identifier(Data,X,Type,Freq,T,CVer,Mag)

%   [GDisc1,GCont1,GDisc2,GCont2,C] = Temp_Identifier(Data,X,Type,Freq,T,CVer)
%   Data would be the Y axis
%   X is the time/sample axis
%   Freq is a data grouping variable
%   Type asks for the expected order of the data (1->speed, 2-> Position)
%   CVer is the controller version to be in the output.

%Mag = MaxVel; % Tensão que o Arduino pôs no motor
%T = T*10^6 % Passa pra segundo

%% Contas Preliminares
%{
A função de transferência tem ordem 2, um pólo da parte elérica e 1 da
parte mecânica. O da parte elétrica é desprezível em relação ao mecânico

O da parte mecânica é (s + B/J) onde B representa forças contrárias ao
movimento vindas da velocidade (como atrito) e J o momento de inércia

Para o caso da roda, J = (m/2)*(R1^2+R2^2) então o teste no chão é mais
lento que o suspenso. O motor tem massa de 50g e o carrinho de 470g. Por
margem de erro, vamos usar 1 kg.

Daí, o momento de inércia é 20x maior que
o suspenso então supondo que não haja atrito extra, o sistema é 20 vezes
mais lento.
%}


%% Correções em caso de indefinição


if (exist('PosCon') ≠ 1)
```

```
35      PosCon = Type - 1;
36  end
37
38  %% Achar a aproximação
39
40  funct        = ['(b)*(1    -    exp(-a*(x)))'];
41  %funct        = ['      b*1 - b*exp(-a*x) '];
42  %B = abs(mean(Data(ceil(end/2):end)))
43  %A = (B - Data)/B;
44  %A = diff(A)./diff(X);
45  %A = max(A);
46  %funct_Vel    = fit(X,Data,funct,'Start',[A B])
47  funct_Vel    = fit(X,Data,funct,'Start',[10  100])
48  A  = funct_Vel.a;
49  B  = funct_Vel.b;
50  G2 = zpk([],[0 -A],B*A);
51  %G2.InputDelay = funct_Vel.c;
52  G1 = zpk([],[-A],B*A);
53  %G1.InputDelay = funct_Vel.c;
54
55
56  GCont1 = (1/Mag)*G1;
57  GCont2 = (1/Mag)*G2;
58  %% Discretizar
59
60  GDisc1 = c2d(GCont1,T);
61  GDisc1 = GDisc1;%*zpk([],[0],1,T);
62  GDisc1.IODelay = 0;
63  GDisc2 = c2d(GCont2,T);
64  GDisc2 = GDisc2;%*zpk([],[0],1,T);
65  GDisc2.IODelay = 0;
66
67  %GDisc1 = c2d(GCont1,T);
68  %GDisc2 = c2d(GCont2,T);
69
70  %% Controles
71
72  if Type == 1
73      GDisc = GDisc1;
74      GCont = GCont1;
75  else
76      GDisc = GDisc2;
77      GCont = GCont2;
```

```
78  end
79
80  DV = CVer;
81  switch DV
82      case {1 , 2 , 3 , 4} % Deadbeats
83          if DV == 1 || DV == 3
84              Den = 1;
85              if DV == 3
86                  [Z,P,¬] = zpkdata(GDisc,'v');
87                  if ¬isempty(Z)
88                      P = 1/prod(abs(Z - [P ; 1]));
89                      P = max([Z-P Z+P]);
90                      Den = [-0.4 Den];
91                  else
92                      Den = [-0.56 Den];
93                  end
94                  %Den = conv([1 0.75 0.35],Den);
95              end
96          else
97              Den = [1 1];
98              [Z,¬,¬] = zpkdata(GDisc,'v');
99              if DV == 4
100                 %P = abs(1/(Z-1))*abs(1/());
101                 %if abs(Z) > P
102                 %A = -(Z + P);
103                 %Den = conv([1 1 0.28095],Den);
104                 Den = [-0.45 -0.45 Den];
105                 %Den = [A A Den];
106             end
107         end
108         Den = poly(Den);
109         C = tf(-Den(2:end),Den,T);
110         C = zpk(C);
111         C = (1/GDisc)*C;
112     case 5  %Proporcional Integral DB
113         display PI
114         [Z,P,N] = zpkdata(GDisc,'v')
115         if Type == 2
116             C = zpk([1-T],[1],1.5/sqrt(T),T);
117             C = 1*C;
118         else
119             Kp = (     1)/(N);
120             Ki = (1 - P)/(N);
```

```
121             C = zpk([],[1],Ki,T) + zpk([],[],Kp,T);
122         end
123         %K = 1;
124     case 6  %Proporcional Integral Normal
125         [Z,P,N] = zpkdata(GDisc,'v')
126         if PosCon
127             C = zpk([1-T],[1],1.5/sqrt(T),T);
128             C = 1*C;
129         else
130             Kp = (1 + P)/(N);
131             Ki = (1     )/(N);
132             C = zpk([],[1],Ki,T) + zpk([],[],Kp,T);
133         end
134     case 7 %Proporcional
135         if PosCon
136             [CN,CD] = tfdata(GDisc,'v')
137             CN = CN(2:end);
138             CD = CD(2:end);
139             A = CN(1)^2;
140             B = (2*CD(1)*CN(1) - 4*CN(2));
141             C = (CD(1)^2 - 4*CD(2));
142             KB = (-B + sqrt(B^2 - 4*A*C))/(2*A);
143             KA = (-B - sqrt(B^2 - 4*A*C))/(2*A);
144             K = 3*min([KA,KB]);
145         else
146             [CN,CD] = tfdata(GDisc,'v');
147             K = (-CD(2))/CN(2);
148         end
149         C = zpk([],[],K,T);
150
151     case 8 % Atraso
152         if PosCon
153             [CN,CD] = tfdata(GDisc,'v');
154             CN = CN(2:end);
155             CD = CD(2:end);
156             A = CN(1)^2;
157             B = (2*CD(1)*CN(1) - 4*CN(2));
158             C = (CD(1)^2 - 4*CD(2));
159             KB = (-B + sqrt(B^2 - 4*A*C))/(2*A);
160             KA = (-B - sqrt(B^2 - 4*A*C))/(2*A);
161             K = 2*min([KA,KB]);
162         else
163             [CN,CD] = tfdata(GDisc,'v');
```

```
164             K = (-CD(2))/CN(2);
165         end
166         %Kp = 10/(K*CSD_Ess(GDisc));
167         Kp = 1.5;
168         Z = 0.9;
169         C = zpk([Z],[(Z+Kp-1)/Kp],K,T);
170     case 9  %Avanço
171         if PosCon
172             [Z,P,K] = zpkdata(GDisc,'v');
173             Z = sort(Z);
174             %[N,D]    =  ...
                    tfdata(minreal(series(zpk(P(2),Z/(1-Z),(K)*(1/(1-Z)),T),GDisc)) ...
                    ,'v');
175             %[N,D]    =  tfdata(minreal(series(zpk([],-0.437,1,T),tf(N,D,T))), ...
                    'v');
176             C = zpk(P(2),Z(1)/(1-Z(1)), (0.5/K)*(1/(1-Z(1)))/1.0,T);%1.6
177             %pause();
178         end
179     case 10  %Zieger-Nichols, Como descrito na página 40 do TCC a Regina
180         [A,B]  = step(GCont);
181         [R,Tr] = max(diff(A)./diff(B))%Variação máxima / Tempo de subida
182         Yr     = A(Tr)%Saída no instante de pico
183         Vrp    = mean(A(ceil(end/2):end))%Regime permanente
184
185         ZN_T = B(Tr) - (Yr/R);%Variáveis intermediárias
186         ZN_L = Vrp/R;
187
188         C_Kp = 1.2*(ZN_T/ZN_L)%Fórmula
189         C_Ki = 2*ZN_L
190         C_Kd = 0.5*ZN_L
191
192         %Essa aqui é em acordo com o bloco do matlab 'PID discreto'
193         %Página 56 tem o resultado dela, parecido com o meu. Leeeento.
194         PID_N = 0.01;
195         C = tf(1*C_Kp,1,T) + tf(C_Ki*T,[1 -1],T) + tf(C_Kd*PID_N*[1 -1],[1 ...
                -1+PID_N*T],T);
196     case 11
197         C = Digital_Deadbeat(GDisc,[],[],2);
198     case 12 %Control-Based
199         [N,D] = tfdata(GDisc,'v');
200         CN(1) = 1/D(1); CD(1) = 1;
201         B = max(abs(D(2:end)./N(2:end)));
202         A = poly([(1/B)*ones(1,size(D,2)-1)]);
```

```
203             for n = 2:size(D,2)
204                 CN(n) = A(n);
205                 CD(n) = -(CN(n)*D(n))/N(n);
206             end
207             C = tf(CN,CD,T);
208 
209 end
210 [Z,P,K] = zpkdata(C,'v');
211 if max([size(Z,1) size(P,1)]) > 0
212     C = minreal(C);
213 else
214     C = minreal(C);
215 end
216 
217 end
```

ArduinoSS_DT1b

Função de aquisição de dados via comunicação bluetooth, utilizada na identificação dos motores.

Retorne à tabela clicando nesta frase

```
1  %% Script para identificação dos motores e teste dos controles
2  
3  %% Configuração de variáveis
4  
5  ParseVars = {};%Guarda as variáveis que o matlab espera ver na saída
6  Tmp1 = zeros(10^5,1);%Prealocação para acelerar o código(removível)
7  %Variáveis do VarsProg que o parser vai atrás
8  A = ArduinoSS_DT1RTVecSetup;
9  
10 for n = 1:size(A,2)
11     ParseVars = [ParseVars ; {VarsProg{A(n),2} ,Tmp1,n,0,VarsProg{A(n),2}(1), ...
           A(n)}];
12 end
13 
14 %% Reinicia a comunicação
15 
16 delete(instrfind);%Desfaz conexões ativas
17 BL = Bluetooth('Aurora',1);%Conecta por bluetooth
```

```
18
19 %% Aguarda a comunicação iniciar
20 fopen(BL);
21 while MyStringComparator(BL.Status,'open') == 0
22     %;
23 end
24 pause(5.0);
25 %Esvazia o buffer de leitura
26 while BL.BytesAvailable > 0
27     fscanf(BL);
28 end
29 %% Envia comandos e mede a resposta.
30 %Espera-se que um códugo de teste esteja carregado
31 %Com RunMain = -2 ou RunMain = -3
32 Tmp2 = 100;
33 z = 0;
34 while z < 2
35     y = 0;
36     while y < Tmp2
37         fprintf(BL,num2str(z+1));
38         Tmp1 = strsplit(fscanf(BL));
39         y = y + 1;
40         for n = 1:size(ParseVars,1)
41             ParseVars{n,2}(y + Tmp2*z) = MyParseFloat(Tmp1{n});
42         end
43     end
44     fprintf(BL,num2str(3));
45     fscanf(BL);
46     pause(3.0);
47     z = z + 1;
48 end
49
50 %% Manda os dados pro PosData_Identify
51
52
53 % Pra Frente
54 PosData_Identify_F{1,2}(1:100) = ParseVars{1,2}(1:100)/(10^6);%Tempo
55 PosData_Identify_F{2,2}(1:100) = ParseVars{4,2}(1:100);%Direita
56 PosData_Identify_F{6,2}(1:100) = ParseVars{5,2}(1:100);%Esquerda
57 for n = 1:9 %Corta os tamanhos
58     PosData_Identify_F{n,2} = PosData_Identify_F{n,2}(1:100);
59 end
60
```

```
% Pra Trás
PosData_Identify_T{1,2}(1:100) = ParseVars{1,2}(101:200)/(10^6);%Tempo
PosData_Identify_T{2,2}(1:100) = ParseVars{4,2}(101:200);%Direita
PosData_Identify_T{6,2}(1:100) = ParseVars{5,2}(101:200);%Esquerda
for n = 1:9 %Corta os tamanhos
    PosData_Identify_T{n,2} = PosData_Identify_T{n,2}(1:100);
end

% Ajeita as posições iniciais e sinais
PosData_Identify_F{2,2} = PosData_Identify_F{2,2} - PosData_Identify_F{2,2}(1);
PosData_Identify_F{6,2} = PosData_Identify_F{6,2} - PosData_Identify_F{6,2}(1);
PosData_Identify_T{2,2} = PosData_Identify_T{2,2}(1) - PosData_Identify_T{2,2};
PosData_Identify_T{6,2} = PosData_Identify_T{6,2}(1) - PosData_Identify_T{6,2};

% Faz as velocidades a partir das posições
PosData_Identify_F{8,2} = [0 ; ...
    diff(PosData_Identify_F{6,2})./diff(PosData_Identify_F{1,2})];
PosData_Identify_F{4,2} = [0 ; ...
    diff(PosData_Identify_F{2,2})./diff(PosData_Identify_F{1,2})];
PosData_Identify_T{8,2} = [0 ; ...
    diff(PosData_Identify_T{6,2})./diff(PosData_Identify_T{1,2})];
PosData_Identify_T{4,2} = [0 ; ...
    diff(PosData_Identify_T{2,2})./diff(PosData_Identify_T{1,2})];

% Leva os tempos pra zero
PosData_Identify_F{1,2} = PosData_Identify_F{1,2} - PosData_Identify_F{1,2}(1);
PosData_Identify_T{1,2} = PosData_Identify_T{1,2} - PosData_Identify_T{1,2}(1);

fit(PosData_Identify_T{1,2}(1:25), PosData_Identify_T{8,2}(1:25), ...
    '(b)*(1-exp(-a*x))','Start',[10 62]);

%Fazer uns gráficos
close all;
figure('units','normalized','outerposition',[0.01 0.01 0.95 0.95]);
subplot(2,4,2)
hold on;
plot(ParseVars{1,2}(  1:100)./(10^6),ParseVars{5,2}(  1:100),'b');
plot(ParseVars{1,2}(101:200)./(10^6),ParseVars{5,2}(101:200),'r');
hold off;
subplot(2,4,3)
hold on;
plot(ParseVars{1,2}(  1:100)./(10^6),ParseVars{4,2}(  1:100),'b');
plot(ParseVars{1,2}(101:200)./(10^6),ParseVars{4,2}(101:200),'r');
```

```
99  hold off;
100 subplot(2,4,5)
101 hold on;
102 plot(PosData_Identify_F{1,2},PosData_Identify_F{6,2},'r');
103 hold off;
104 subplot(2,4,6)
105 hold on;
106 plot(PosData_Identify_T{1,2},PosData_Identify_T{6,2},'r');
107 hold off;
108 subplot(2,4,7)
109 hold on;
110 plot(PosData_Identify_F{1,2},PosData_Identify_F{2,2},'b');
111 hold off;
112 subplot(2,4,8)
113 hold on;
114 plot(PosData_Identify_T{1,2},PosData_Identify_T{2,2},'b');
115 hold off;
116
117 % Salva
118 Tmp1 = 'TesteNormal_M2_10_08';
119 save(Tmp1,'PosData_Identify_F','PosData_Identify_T');
120 save(['Workspace_' Tmp1]);
121
122 %% Fecha a comunicação
123 fclose(S);
124 while MyStringComparator(S.Status,'closed') == 0
125     %;
126 end
127 clear S;
```

Arduino_SS_GeraControle

Função que gera os arquivos de controle para o Arduino.

Retorne à tabela clicando nesta frase

```
1 %% Prepara o nome
2
3 ScriptName = '// Arduino_SS_GeraControle \n';
4
5 %% Achar as velocidades aqui mesmo, por segundo
```

```
6
7  if (exist('PosData_Identify_F','var') == 1 &&...
8    exist('PosData_Identify_T','var') == 1)
9      TimerF = PosData_Identify_F{1,2}/10^6;
10     TimerT = PosData_Identify_T{1,2}/10^6;
11     Data_EncDD_F   = PosData_Identify_F{4,2};
12     Data_EncED_F_V = PosData_Identify_F{8,2};
13     Data_EncDD_T   = PosData_Identify_T{4,2};
14     Data_EncED_T_V = PosData_Identify_T{8,2};
15
16     if VelGrouping > 0
17         for n = 2:size(PosData_Identify_F{8,2},1)
18             PosData_Identify_F{4,2}(n) = ...
19                 PosData_Identify_F{4,2}(n-1)*VelGrouping + ...
20                 (1-VelGrouping)*PosData_Identify_F{4,2}(n);
21             PosData_Identify_F{8,2}(n) = ...
22                 PosData_Identify_F{8,2}(n-1)*VelGrouping + ...
23                 (1-VelGrouping)*PosData_Identify_F{8,2}(n);
24             PosData_Identify_T{4,2}(n) = ...
25                 PosData_Identify_T{4,2}(n-1)*VelGrouping + ...
26                 (1-VelGrouping)*PosData_Identify_T{4,2}(n);
27             PosData_Identify_T{8,2}(n) = ...
28                 PosData_Identify_T{8,2}(n-1)*VelGrouping + ...
29                 (1-VelGrouping)*PosData_Identify_T{8,2}(n);
30         end
31     end
32 else
33     Timer = (0:0.01:3-0.01)';
34     A = (100)*Timer;
35     F = str2func('@(x)(150/13)*(13*x - 1   +   exp(-13*x))');
36     PosData_Identify_F = {'Timer',Timer,1};
37     PosData_Identify_F = [ PosData_Identify_F ;{'EncDD',F(Timer),2}];
38     PosData_Identify_F = [ PosData_Identify_F ;{'PosD' ,A,3}];
39     PosData_Identify_F = [ PosData_Identify_F ;{'VelD' ,[0 ;...
40         diff(PosData_Identify_F{2,2})./diff(PosData_Identify_F{1,2})],4}];
41     PosData_Identify_F = [ PosData_Identify_F ;{'EPosD',zeros(10000,1),5}];
42     PosData_Identify_F = [ PosData_Identify_F ;{'EncED',F(Timer),6}];
43     PosData_Identify_F = [ PosData_Identify_F ;{'PosE' ,A,7}];
44     PosData_Identify_F = [ PosData_Identify_F ;{'VelE' ,[0 ;...
45         diff(PosData_Identify_F{6,2})./diff(PosData_Identify_F{1,2})],8}];
46     PosData_Identify_F = [ PosData_Identify_F ;{'EPosE',zeros(10000,1),9}];
47
48
```

```
    PosData_Identify_T = {'Timer',Timer,1};
    PosData_Identify_T = [PosData_Identify_T ; {'EncDD',F(Timer),2}];
    PosData_Identify_T = [PosData_Identify_T ; {'PosD' ,A,3}];
    PosData_Identify_T = [PosData_Identify_T ; {'VelD' ,[0 ;...
        diff(PosData_Identify_T{2,2})./diff(PosData_Identify_T{1,2})],4}];
    PosData_Identify_T = [PosData_Identify_T ; {'EPosD',zeros(10000,1),5}];
    PosData_Identify_T = [PosData_Identify_T ; {'EncED',F(Timer),6}];
    PosData_Identify_T = [PosData_Identify_T ; {'PosE' ,A,7}];
    PosData_Identify_T = [PosData_Identify_T ; {'VelE' ,[0 ;...
        diff(PosData_Identify_T{6,2})./diff(PosData_Identify_T{1,2})],8}];
    PosData_Identify_T = [PosData_Identify_T ; {'EPosE',zeros(10000,1),9}];
end

%% Configurações gerais dos gráficos, figuras e dados



FScale        = 2*pi*r/NFuros;%Passa de furos pra metros
YSize_Vel_I   = [0  5];%Pro Ylabel das velocidades na identificação
YSize_Pos_I   = [0  2];%Pro Ylabel das posições na identificação
YSize_Aci_S   = [0  8];%Pro Ylabel dos acionamentos na simulação
YSize_Pos_S   = [0  8];%Pro Ylabel das posições na simulação
YSize_Vel_S   = [0  5];%Pro Ylabel das velocidades na simulação
This_SimData = {'[PosCon,CVer]','ts (EF ET DF DT)','mp (EF ET DF DT)';...
    [PosCon,CVer],[inf inf inf inf],[0 0 0 0]...
    };
switch CVer
    case 0
        CName = 'Desligado';
        FName = 'Desligado';
    case 1
        CName = 'Deadbeat (Tipo 1 Normal';
        FName = 'Fig_DB1N';
        YSize_Aci_S = [0  30];%Deadbeat tem uma excursão maior
    case 2
        CName = 'Deadbeat (Tipo 2 Normal';
        FName = 'Fig_DB2N';
        YSize_Aci_S = [0  30];%Deadbeat tem uma excursão maior
    case 3
        CName = 'Deadbeat (Tipo 1 Amortecido';
        FName = 'Fig_DB1A';
        YSize_Aci_S = [0  15];%Deadbeat tem uma excursão maior
```

```
92      case 4
93          CName = 'Deadbeat (Tipo 2 Amortecido';
94          FName = 'Fig_DB2A';
95          YSize_Aci_S = [0  15];%Deadbeat tem uma excursão maior
96      case 5
97          CName = 'Proporcional Integral (Com Cancelamento';
98          FName = 'Fig_PIC';
99      case 6
100         CName = 'Proporcional Integral (Sem Cancelamento';
101         FName = 'Fig_PIN';
102     case 7
103         CName = 'Proporcional (Subamortecido';
104         FName = 'Fig_P';
105     case 8
106         CName = 'Lag (2.5x ';
107         FName = 'Fig_Lag';
108     case 9
109         CName = 'Avanço (Com Cancelamento';
110         FName = 'Fig_Lead';
111     case 10
112         CName = 'Proporcional-Integral-Derivativo (Zieger-Nichols';
113         FName = 'Fig_PIDZH';
114         %Tmp1 = ['$$C = \\frac{1-P}{N(z-1)}+\\frac{1}{N} $$'];
115         %Ctrl_Formulas = [Ctrl_Formulas; {[CName,') '],Tmp1,5}];
116 end
117 YName_Vel   = 'Velocidade (m/s)';
118 YName_Pos   = 'Posiçao (m)';
119 MaxY1       = 0;
120 MaxY2       = 0;
121 if CVer ~= 0
122     XName = 'Tempo(Segundos)';
123     CL1Name = 'Estimado';
124     CE1Name = 'Dados';
125     CL2Name = 'Ideal';
126     CE2Name = 'Id.';
127     YAName = 'Tensão (Volts)';
128     if PosCon
129         %YRName = 'Resposta (Furos do Encoder)';
130         YRName = 'Posiçao (m)';
131         CName = [CName ', Posição'];
132         FName = [FName '_Pos'];
133     else
134         %YRName = 'Resposta (Furos por Segundo)';
```

```
135             YRName = 'Velocidade (m/s)';
136             CName = [CName ', Velocidade'];
137             FName = [FName '_Vel'];
138         end
139         CName = [CName ', Amostragem de ',num2str(T),'s)'];
140         FName = [FName '_T',int2str(1000*T)];
141     end
142     FName = [FName '.png'];
143
144     %% Vetor de fórmulas pra pôr no LateX depois
145
146
147     Ctrl_Formulas = {};%Nome, fórmula em .tex, identificador
148     Tmp1 = ['$$C = \\frac{1}{G}\\frac{1}{(z-1)} $$'];
149     Ctrl_Formulas = [Ctrl_Formulas; {'DeadBeat Tipo 1',Tmp1,1,1}];
150     Tmp1 = ['$$C = \\frac{1}{G}\\frac{2z -1}{(z-1)^2} $$'];
151     Ctrl_Formulas = [Ctrl_Formulas; {'DeadBeat Tipo 2',Tmp1,2,2}];
152     Tmp1 = ['$$C = \\frac{1}{G}\\frac{0.6z+0.4}{(z-1)(z+0.4)} $$'];
153     Ctrl_Formulas = [Ctrl_Formulas; {'DeadBeat Amortecido Tipo 1',Tmp1,3,3}];
154     Tmp1 = ['$$C = \\frac{1}{G}\\frac{1.1z^3+0.5975z^2-0.495z-0.2025}'...
155         '{(z-1)^{2}(z+0.45)^{2}} $$'];
156     Ctrl_Formulas = [Ctrl_Formulas; {'DeadBeat Amortecido Tipo 2',Tmp1,4,4}];
157     Tmp1 = ['$$C = \\frac{1-P}{N(z-1)}+\\frac{1}{N} $$'];
158     Ctrl_Formulas = [Ctrl_Formulas; {'PI com cancelamentos',Tmp1,5,5}];
159     Tmp1 = ['$$C = \\frac{1}{N(z-1)}+\\frac{1+P}{N} $$'];
160     Ctrl_Formulas = [Ctrl_Formulas; {'PI sem cancelamentos',Tmp1,6,6}];
161     Tmp1 = ['$$C = K $$'];
162     Ctrl_Formulas = [Ctrl_Formulas; {'Proporcional',Tmp1,7,7}];
163     Tmp1 = ['$$C = \\frac{K(z-A)}{(z-P)} $$'];
164     Ctrl_Formulas = [Ctrl_Formulas; {'Atraso',Tmp1,8,8}];
165     Tmp1 = ['$$C = \\frac{0.5}{GK(1-GN_1)}*'...
166         '\\frac{z - GD_2}{z-\\frac{GN_1}{1-GN_1}} $$'];
167     Ctrl_Formulas = [Ctrl_Formulas; {'Avanço',Tmp1,9,9}];
168     %C = zpk(P(2),Z(1)/(1-Z(1)),(1.0/K)*(1/(1-Z(1)))/1.0,T);%1.6
169
170     %% Outros pontos de setup
171
172     if (exist('Freq','var') ~= 1)
173         Freq = 1;
174     end
175
176     if ~SkipControl
177         %% Testes para frente Direito
```

```
178     aux1 = PosData_Identify_F;
179     [GDDF1,GCDF1,GDDF2,GCDF2,CDF] =...
180         Temp_Identifier(aux1{4,2},aux1{1,2},PosCon+1,Freq,T,CVer,MaxVel);
181
182     if PosCon
183         GDDF = GDDF2;
184         GCDF = GCDF2;
185     else
186         GDDF = GDDF1;
187         GCDF = GCDF1;
188     end
189
190     [Sim_GC_Num,Sim_GC_Den] = tfdata(GCDF,'v');
191     [Sim_GD_Num,Sim_GD_Den] = tfdata(GDDF,'v');
192     if (isa(CDF,'tf') || isa(CDF,'zpk'))
193         [Sim_C_Num,Sim_C_Den] = tfdata(CDF,'v');
194         %display TransferFunction
195         while length(Sim_C_Num) < 6
196             Sim_C_Num = [Sim_C_Num 0];
197         end
198         while length(Sim_C_Den) < 6
199             Sim_C_Den = [Sim_C_Den 0];
200         end
201     else
202         Sim_C_Num = [C 0 0 0 0 0 0];
203         Sim_C_Den = [0 0 0 0 0 0 1];
204     end
205
206         [Sim_C_Num,Sim_C_Den] = tfdata(CDF,'v');
207         CDF_Fim = length(Sim_C_Num);
208         CDF_NumVec_Val = Sim_C_Num;
209         CDF_DenVec_Val = Sim_C_Den(2:end);
210
211
212
213
214     %% Testes para frente Esquerdo
215
216
217     [GDEF1,GCEF1,GDEF2,GCEF2,CEF] =...
218         Temp_Identifier(aux1{8,2},aux1{1,2},PosCon+1,Freq,T,CVer,MaxVel);
219
220     if PosCon
```

```
221             GDEF = GDEF2;
222             GCEF = GCEF2;
223         else
224             GDEF = GDEF1;
225             GCEF = GCEF1;
226         end
227
228
229         [Sim_GC_Num,Sim_GC_Den] = tfdata(GCEF,'v');
230         [Sim_GD_Num,Sim_GD_Den] = tfdata(GDEF,'v');
231
232         if (isa(CEF,'tf') || isa(CEF,'zpk'))
233             [Sim_C_Num,Sim_C_Den] = tfdata(CEF,'v');
234             %display TransferFunction
235             while length(Sim_C_Num) < 6
236                 Sim_C_Num = [Sim_C_Num 0];
237             end
238             while length(Sim_C_Den) < 6
239                 Sim_C_Den = [Sim_C_Den 0];
240             end
241         else
242             Sim_C_Num = C;
243             Sim_C_Den = 1;
244         end
245
246             [Sim_C_Num,Sim_C_Den] = tfdata(CEF,'v');
247             CEF_Fim = length(Sim_C_Num);
248             CEF_NumVec_Val = Sim_C_Num;
249             CEF_DenVec_Val = Sim_C_Den(2:end);
250
251
252
253         %% Testes para trás Direito
254         aux1 = PosData_Identify_T;
255         [GDDT1,GCDT1,GDDT2,GCDT2,CDT] =...
256             Temp_Identifier(aux1{4,2},aux1{1,2},PosCon+1,Freq,T,CVer,MaxVel);
257
258         if PosCon
259             GDDT = GDDT2;
260             GCDT = GCDT2;
261         else
262             GDDT = GDDT1;
263             GCDT = GCDT1;
```

```
264     end
265
266     [Sim_GC_Num,Sim_GC_Den] = tfdata(GCDT,'v');
267     [Sim_GD_Num,Sim_GD_Den] = tfdata(GDDT,'v');
268     if (isa(CDT,'tf') || isa(CDT,'zpk'))
269         [Sim_C_Num,Sim_C_Den] = tfdata(CDT,'v');
270         %display TransferFunction
271         while length(Sim_C_Num) < 6
272             Sim_C_Num = [Sim_C_Num 0];
273         end
274         while length(Sim_C_Den) < 6
275             Sim_C_Den = [Sim_C_Den 0];
276         end
277     else
278         Sim_C_Num = C;
279         Sim_C_Den = 1;
280     end
281
282         [Sim_C_Num,Sim_C_Den] = tfdata(CDT,'v');
283         CDT_Fim = length(Sim_C_Num);
284         CDT_NumVec_Val = Sim_C_Num;
285         CDT_DenVec_Val = Sim_C_Den(2:end);
286
287
288
289
290
291     %% Testes para trás Esquerdo
292
293     [GDET1,GCET1,GDET2,GCET2,CET] =...
294         Temp_Identifier(aux1{8,2},aux1{1,2},PosCon+1,Freq,T,CVer,MaxVel);
295
296     if PosCon
297         GDET = GDET2;
298         GCET = GCET2;
299     else
300         GDET = GDET1;
301         GCET = GCET1;
302     end
303     %T = 1;
304
305     [Sim_GC_Num,Sim_GC_Den] = tfdata(GCET,'v');
306     [Sim_GD_Num,Sim_GD_Den] = tfdata(GDET,'v');
```

```
307
308     if (isa(CET,'tf') || isa(CET,'zpk'))
309         [Sim_C_Num,Sim_C_Den] = tfdata(CET,'v');
310         %display TransferFunction
311         while length(Sim_C_Num) < 6
312             Sim_C_Num = [Sim_C_Num 0];
313         end
314         while length(Sim_C_Den) < 6
315             Sim_C_Den = [Sim_C_Den 0];
316         end
317     else
318         Sim_C_Num = C;
319         Sim_C_Den = 1;
320     end
321
322         [Sim_C_Num,Sim_C_Den] = tfdata(CET,'v');
323         CET_Fim = length(Sim_C_Num);
324         CET_NumVec_Val = Sim_C_Num;
325         CET_DenVec_Val = Sim_C_Den(2:end);
326
327     %% Simulação pra ver se a identificação ficou boa
328     figure('units','normalized','outerposition',[0.01 0.01 0.95 0.95])
329
330     subplot(2,4,1); %Velocidades, EF
331     hold on;
332     [Tmp1,Tmp2] = step(MaxVel*GCEF1,PosData_Identify_F{1,2}(end));
333     plot(Tmp2,FScale*Tmp1);
334     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
335     [Tmp1,Tmp2] = step(MaxVel*GDEF1,PosData_Identify_F{1,2}(end));
336     plot(Tmp2,FScale*Tmp1);
337     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
338     plot(PosData_Identify_F{1,2},FScale*PosData_Identify_F{8,2});
339     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_F{8,2})]);
340     grid;
341     title('Velocidade (EF)');
342     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthEast');
343     ylabel(YName_Vel);
344     hold off;
345     xlim([PosData_Identify_F{1,2}(1) PosData_Identify_F{1,2}(end)]);
346     %ylim(YSize_Vel_I);
347
348
349     subplot(2,4,2); %Velocidades, ET
```

```
350     hold on;
351     [Tmp1,Tmp2] = step(MaxVel*GCET1,PosData_Identify_T{1,2}(end));
352     plot(Tmp2,FScale*Tmp1);
353     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
354     [Tmp1,Tmp2] = step(MaxVel*GDET1,PosData_Identify_T{1,2}(end));
355     plot(Tmp2,FScale*Tmp1);
356     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
357     plot(PosData_Identify_T{1,2},FScale*PosData_Identify_T{8,2});
358     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_T{8,2})]);
359     grid;
360     title('Velocidade (ET)');
361     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthEast');
362     hold off;
363     xlim([PosData_Identify_T{1,2}(1) PosData_Identify_T{1,2}(end)]);
364     %ylim(YSize_Vel_I);
365
366
367     subplot(2,4,3); %Velocidades, DF
368     hold on;
369     [Tmp1,Tmp2] = step(MaxVel*GCDF1,PosData_Identify_F{1,2}(end));
370     plot(Tmp2,FScale*Tmp1);
371     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
372     [Tmp1,Tmp2] = step(MaxVel*GDDF1,PosData_Identify_F{1,2}(end));
373     plot(Tmp2,FScale*Tmp1);
374     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
375     plot(PosData_Identify_F{1,2},FScale*PosData_Identify_F{4,2});
376     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_F{4,2})]);
377     grid;
378     title('Velocidade (DF)');
379     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthEast');
380     hold off;
381     xlim([PosData_Identify_F{1,2}(1) PosData_Identify_F{1,2}(end)]);
382     %ylim(YSize_Vel_I);
383
384
385
386     subplot(2,4,4); %Velocidades, DT
387     hold on;
388     [Tmp1,Tmp2] = step(MaxVel*GCDT1,PosData_Identify_T{1,2}(end));
389     plot(Tmp2,FScale*Tmp1);
390     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
391     [Tmp1,Tmp2] = step(MaxVel*GDDT1,PosData_Identify_T{1,2}(end));
392     plot(Tmp2,FScale*Tmp1);
```

```
393     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
394     plot(PosData_Identify_T{1,2},FScale*PosData_Identify_T{4,2});
395     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_T{4,2})]);
396     grid;
397     title('Velocidade (DT)');
398     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthEast');
399     hold off;
400     xlim([PosData_Identify_T{1,2}(1) PosData_Identify_T{1,2}(end)]);
401     ylim(YSize_Vel_I);
402
403
404     %Ajeitar os tamanhos dos plots de velocidade
405     %MaxY1 = max([MaxY1 3]);
406     MaxY1 = 1.5*round(MaxY1,1);
407     subplot(2,4,1)
408     ylim([0 MaxY1]);
409     subplot(2,4,2)
410     ylim([0 MaxY1]);
411     subplot(2,4,3)
412     ylim([0 MaxY1]);
413     subplot(2,4,4)
414     ylim([0 MaxY1]);
415     MaxY1 = 0;
416
417
418
419
420
421     subplot(2,4,5); %Posição, EF
422     hold on;
423     [Tmp1,Tmp2] = step(MaxVel*GCEF2,PosData_Identify_F{1,2}(end));
424     plot(Tmp2,FScale*Tmp1);
425     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
426     [Tmp1,Tmp2] = step(MaxVel*GDEF2,PosData_Identify_F{1,2}(end));
427     plot(Tmp2,FScale*Tmp1);
428     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
429     plot(PosData_Identify_F{1,2},FScale*PosData_Identify_F{6,2});
430     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_F{6,2})]);
431     grid;
432     title('Posição (EF)');
433     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthWest');
434     ylabel(YName_Pos);
435     xlabel(XName);
```

```
436     hold off;
437     xlim([PosData_Identify_F{1,2}(1) PosData_Identify_F{1,2}(end)]);
438     %ylim(YSize_Pos_I);
439
440
441     subplot(2,4,6); %Posição, ET
442     hold on;
443     [Tmp1,Tmp2] = step(MaxVel*GCET2, PosData_Identify_T{1,2}(end));
444     plot(Tmp2, FScale*Tmp1);
445     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
446     [Tmp1,Tmp2] = step(MaxVel*GDET2, PosData_Identify_T{1,2}(end));
447     plot(Tmp2, FScale*Tmp1);
448     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
449     plot(PosData_Identify_T{1,2}, FScale*PosData_Identify_T{6,2});
450     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_T{6,2})]);
451     grid;
452     title('Posição (ET)');
453     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthWest');
454     xlabel(XName);
455     hold off;
456     xlim([PosData_Identify_T{1,2}(1) PosData_Identify_T{1,2}(end)]);
457     %ylim(YSize_Pos_I);
458
459
460     subplot(2,4,7); %Posição, DF
461     hold on;
462     [Tmp1,Tmp2] = step(MaxVel*GCDF2, PosData_Identify_F{1,2}(end));
463     plot(Tmp2, FScale*Tmp1);
464     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
465     [Tmp1,Tmp2] = step(MaxVel*GDDF2, PosData_Identify_F{1,2}(end));
466     plot(Tmp2, FScale*Tmp1);
467     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
468     plot(PosData_Identify_F{1,2}, FScale*PosData_Identify_F{2,2});
469     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_F{2,2})]);
470     grid;
471     title('Posição (DF)');
472     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthWest');
473     xlabel(XName);
474     hold off;
475     xlim([PosData_Identify_F{1,2}(1) PosData_Identify_F{1,2}(end)]);
476     %ylim(YSize_Pos_I);
477
478
```

```
479     subplot(2,4,8); %Posição, DT
480     hold on;
481     [Tmp1,Tmp2] = step(MaxVel*GCDT2,PosData_Identify_T{1,2}(end));
482     plot(Tmp2,FScale*Tmp1);
483     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
484     [Tmp1,Tmp2] = step(MaxVel*GDDT2,PosData_Identify_T{1,2}(end));
485     plot(Tmp2,FScale*Tmp1);
486     MaxY1 = max([MaxY1 max(FScale*Tmp1)]);
487     plot(PosData_Identify_T{1,2},FScale*PosData_Identify_T{2,2});
488     MaxY1 = max([MaxY1 max(FScale*PosData_Identify_T{2,2})]);
489     grid;
490     title('Posição (DT)');
491     legend([CE2Name '(s)'],[CE2Name '(z)'],CE1Name,'LOCATION','NorthWest');
492     xlabel(XName);
493     hold off;
494     xlim([PosData_Identify_T{1,2}(1) PosData_Identify_T{1,2}(end)]);
495     %ylim(YSize_Pos_I);
496
497     %Ajeitar os tamanhos dos plots de posição
498     MaxY1 = ceil(MaxY1);
499     subplot(2,4,5)
500     ylim([0 MaxY1]);
501     subplot(2,4,6)
502     ylim([0 MaxY1]);
503     subplot(2,4,7)
504     ylim([0 MaxY1]);
505     subplot(2,4,8)
506     ylim([0 MaxY1]);
507     MaxY1 = 0;
508
509
510
511     suptitle('Verificação da Identificação do Sistema');
512     set(gcf,'PaperPositionMode','auto');
513     if VelGrouping > 0
514         saveas(gcf,['Identificacao_VG',int2str(VelGrouping),'.png']);
515     else
516         saveas(gcf,['Identificacao_Nrm.png']);
517     end
518
519
520     %% Gera os defines aqui no final para ajeitar os tamanhos
521
```

```
522
523     MaxLength = max([CDF_Fim, CEF_Fim, CDT_Fim, CET_Fim]);
524     LengthVec =  -1:1:(MaxLength - 2);
525
526     %% Faz as simulações aqui
527     %Toda vez, antes da simulação, uma função ajeita os tamanhos
528     %dos vetores
529     %Menor que 3 no denominador e buga. Aí tem que mudar o numerador pra
530     %combinar.
531
532     figure('units','normalized','outerposition',[0.01 0.01 0.95 0.95])
533     if Simulate
534         MinY1 = 0;
535         %FScale = 1;
536         FScale = FScale;
537         Sim_M = Sim_M/(FScale);
538         CScale = 9/255;
539         % Motor Direito Frontal
540         [Sim_C_Num ,Sim_C_Den ] = Custom_MatchVecs(...
541             4,0,CDF_NumVec_Val,[1 CDF_DenVec_Val]);
542         [Sim_GC_Num,Sim_GC_Den] = tfdata(GCDF,'v');
543         Sim_G_Delay = GCDF.InputDelay;
544         sim('ArduinoSS_TestSubMdl.slx',Duracao)
545
546
547         X = AcScopeData(:,1);%Tempo
548         subplot(2,4,3);
549         Y = AcScopeData(:,2);%Referencia
550         %plot(X,Y,'k');
551         hold on;
552         Y = AcScopeData(:,3);%Estimado
553         plot(X,CScale*Y,'r');
554         MaxY1 = max([MaxY1 max(CScale*Y)]);
555         MinY1 = min([MinY1 min(CScale*Y)]);
556         Tmp1 = stepinfo(CScale*Y,X);
557         if Tmp1.SettlingTime < 4
558             This_SimData{2,2}(3) = Tmp1.SettlingTime + T;
559             This_SimData{2,3}(3) = Tmp1.Overshoot;
560         else
561             This_SimData{2,2}(3) = inf;
562             This_SimData{2,3}(3) = inf;
563         end
564         Y = AcScopeData(:,4);%Ideal
```

```
565         plot(X,CScale*Y,'b');
566         MaxY1 = max([MaxY1 max(CScale*Y)]);
567         MinY1 = min([MinY1 min(CScale*Y)]);
568         %pause();
569         hold off;
570         grid;
571         title('Acionamento (DF)')
572         %xlabel(XName);
573         %ylabel(YAName);
574         legend(CL1Name,CL2Name,'LOCATION','SouthEast');
575         xlim([X(1) X(end)]);
576         %ylim(YSize_Aci_S);
577
578         X = PosScopeData(:,1);%Tempo
579         subplot(2,4,7);
580         Y = PosScopeData(:,2);%Referencia
581         plot(X,FScale*Y,'k');
582         MaxY2 = max([MaxY2 max(FScale*Y)]);
583         hold on;
584         Y = PosScopeData(:,3);%Arduino
585         plot(X,FScale*Y,'r');
586         MaxY2 = max([MaxY2 max(FScale*Y)]);
587         Y = PosScopeData(:,4);%Contínuo
588         plot(X,FScale*Y,'b');
589         MaxY2 = max([MaxY2 max(FScale*Y)]);
590         hold off;
591         grid;
592         title('Resposta (DF)')
593         xlabel(XName);
594         %ylabel(YRName);
595         if PosCon
596             legend('Referência',CL1Name,CL2Name,'LOCATION','NorthWest');
597             %ylim(YSize_Pos_S);
598         else
599             legend('Referência',CL1Name,CL2Name,'LOCATION','SouthEast');
600             %ylim(YSize_Vel_S);
601         end
602         xlim([X(1) X(end)]);
603
604
605
606         % Motor Esquerdo Frontal
607         [Sim_C_Num ,Sim_C_Den ] = Custom_MatchVecs(...
```

```
608             4,0,CEF_NumVec_Val,[1  CEF_DenVec_Val]);
609         [Sim_GC_Num,Sim_GC_Den] = tfdata(GCEF,'v');
610         Sim_G_Delay = GCEF.InputDelay;
611         sim('ArduinoSS_TestSubMdl.slx',Duracao)
612
613         X = AcScopeData(:,1);%Tempo
614         subplot(2,4,1);
615         Y = AcScopeData(:,2);%Referencia
616         %plot(X,Y,'k');
617         hold on;
618         Y = AcScopeData(:,3);%Estimado
619         plot(X,CScale*Y,'r');
620         MaxY1 = max([MaxY1 max(CScale*Y)]);
621         MinY1 = min([MinY1 min(CScale*Y)]);
622         Tmp1 = stepinfo(CScale*Y,X);
623         if Tmp1.SettlingTime < 4
624             This_SimData{2,2}(1) = Tmp1.SettlingTime + T;
625             This_SimData{2,3}(1) = Tmp1.Overshoot;
626         else
627             This_SimData{2,2}(1) = inf;
628             This_SimData{2,3}(1) = inf;
629         end
630         Y = AcScopeData(:,4);%Ideal
631         plot(X,CScale*Y,'b');
632         MaxY1 = max([MaxY1 max(CScale*Y)]);
633         MinY1 = min([MinY1 min(CScale*Y)]);
634         hold off;
635         grid;
636         title('Acionamento (EF)')
637         %xlabel(XName);
638         ylabel(YAName);
639         legend(CL1Name,CL2Name,'LOCATION','SouthEast');
640         xlim([X(1) X(end)]);
641         ylim(YSize_Aci_S);
642
643         X = PosScopeData(:,1);%Tempo
644         subplot(2,4,5);
645         Y = PosScopeData(:,2);%Referencia
646         plot(X,FScale*Y,'k');
647         MaxY2 = max([MaxY2 max(FScale*Y)]);
648         hold on;
649         Y = PosScopeData(:,3);%Discreto
650         plot(X,FScale*Y,'r');
```

```
651         MaxY2 = max([MaxY2 max(FScale*Y)]);
652         Y = PosScopeData(:,4);%Contínuo
653         plot(X,FScale*Y,'b');
654         MaxY2 = max([MaxY2 max(FScale*Y)]);
655         hold off;
656         grid;
657         title('Resposta (EF)')
658         xlabel(XName);
659         ylabel(YRName);
660         if PosCon
661             legend('Referência',CL1Name,CL2Name,'LOCATION','NorthWest');
662             %ylim(YSize_Pos_S);
663         else
664             legend('Referência',CL1Name,CL2Name,'LOCATION','SouthEast');
665             %ylim(YSize_Vel_S);
666         end
667         xlim([X(1) X(end)]);
668
669
670         % Motor Direito Traseiro
671         [Sim_C_Num ,Sim_C_Den ] = Custom_MatchVecs(...
672             4,0,CDT_NumVec_Val,[1 CDT_DenVec_Val]);
673         [Sim_GC_Num,Sim_GC_Den] = tfdata(GCDT,'v');
674         Sim_G_Delay = GCDT.InputDelay;
675         sim('ArduinoSS_TestSubMdl.slx',Duracao)
676
677         X = AcScopeData(:,1);%Tempo
678         subplot(2,4,4);
679         Y = AcScopeData(:,2);%Referencia
680         %plot(X,Y,'k');
681         hold on;
682         Y = AcScopeData(:,3);%Estimado
683         plot(X,CScale*Y,'r');
684         MaxY1 = max([MaxY1 max(CScale*Y)]);
685         MinY1 = min([MinY1 min(CScale*Y)]);
686         Tmp1 = stepinfo(CScale*Y,X);
687         if Tmp1.SettlingTime < 4
688             This_SimData{2,2}(4) = Tmp1.SettlingTime + T;
689             This_SimData{2,3}(4) = Tmp1.Overshoot;
690         else
691             This_SimData{2,2}(4) = inf;
692             This_SimData{2,3}(4) = inf;
693         end
```

```
694         Y = AcScopeData(:,4);%Ideal
695         plot(X,CScale*Y,'b');
696         MaxY1 = max([MaxY1 max(CScale*Y)]);
697         MinY1 = min([MinY1 min(CScale*Y)]);
698         hold off;
699         grid;
700         title('Acionamento (DT)')
701         %xlabel(XName);
702         %ylabel(YAName);
703         legend(CL1Name,CL2Name,'LOCATION','SouthEast');
704         xlim([X(1) X(end)]);
705         %ylim(YSize_Aci_S);
706
707         X = PosScopeData(:,1);%Tempo
708         subplot(2,4,8);
709         Y = PosScopeData(:,2);%Referencia
710         plot(X,FScale*Y,'k');
711         MaxY2 = max([MaxY2 max(FScale*Y)]);
712         hold on;
713         Y = PosScopeData(:,3);%Arduino
714         plot(X,FScale*Y,'r');
715         MaxY2 = max([MaxY2 max(FScale*Y)]);
716         Y = PosScopeData(:,4);%Contínuo
717         plot(X,FScale*Y,'b');
718         MaxY2 = max([MaxY2 max(FScale*Y)]);
719         hold off;
720         grid;
721         title('Resposta (DT)')
722         xlabel(XName);
723         %ylabel(YRName);
724         if PosCon
725             legend('Referência',CL1Name,CL2Name,'LOCATION','NorthWest');
726             %ylim(YSize_Pos_S);
727         else
728             legend('Referência',CL1Name,CL2Name,'LOCATION','SouthEast');
729             %ylim(YSize_Vel_S);
730         end
731         xlim([X(1) X(end)]);
732
733
734         % Motor Esquerdo Traseiro
735         [Sim_C_Num ,Sim_C_Den ] = Custom_MatchVecs(...
736             4,0,CET_NumVec_Val,[1 CET_DenVec_Val]);
```

```
737         [Sim_GC_Num,Sim_GC_Den] = tfdata(GCET,'v');
738         Sim_G_Delay = GCET.InputDelay;
739         sim('ArduinoSS_TestSubMdl.slx',Duracao)
740
741         X = AcScopeData(:,1);%Tempo
742
743         subplot(2,4,2);
744         Y = AcScopeData(:,2);%Referencia
745         %plot(X,Y,'k');
746         hold on;
747         Y = AcScopeData(:,3);%Estimado
748         plot(X,CScale*Y,'r');
749         MaxY1 = max([MaxY1 max(CScale*Y)]);
750         MinY1 = min([MinY1 min(CScale*Y)]);
751         Tmp1 = stepinfo(CScale*Y,X);
752         if Tmp1.SettlingTime < 4
753             This_SimData{2,2}(2) = Tmp1.SettlingTime + T;
754             This_SimData{2,3}(2) = Tmp1.Overshoot;
755         else
756             This_SimData{2,2}(2) = inf;
757             This_SimData{2,3}(2) = Tmp1.Overshoot;
758         end
759         Y = AcScopeData(:,4);%Ideal
760         plot(X,CScale*Y,'b');
761         MaxY1 = max([MaxY1 max(CScale*Y)]);
762         MinY1 = min([MinY1 min(CScale*Y)]);
763         hold off;
764         grid;
765         title('Acionamento (ET)')
766         %xlabel(XName);
767         %ylabel(YAName);
768         legend(CL1Name,CL2Name,'LOCATION','SouthEast');
769         xlim([X(1) X(end)]);
770         %ylim(YSize_Aci_S);
771
772         X = PosScopeData(:,1);%Tempo
773         subplot(2,4,6);
774         Y = PosScopeData(:,2);%Referencia
775         plot(X,FScale*Y,'k');
776         MaxY2 = max([MaxY2 max(FScale*Y)]);
777         hold on;
778         Y = PosScopeData(:,3);%Discreto
779         plot(X,FScale*Y,'r');
```

```
780         MaxY2 = max([MaxY2 max(FScale*Y)]);
781         Y = PosScopeData(:,4);%Contínuo
782         plot(X,FScale*Y,'b');
783         MaxY2 = max([MaxY2 max(FScale*Y)]);
784         hold off;
785         grid;
786         title('Resposta (ET)');
787         xlabel(XName);
788         if PosCon
789             legend('Referência',CL1Name,CL2Name,'LOCATION','NorthWest');
790             %ylim(YSize_Pos_S);
791         else
792             legend('Referência',CL1Name,CL2Name,'LOCATION','SouthEast');
793             %ylim(YSize_Vel_S);
794         end
795         xlim([X(1) X(end)]);
796
797         %Ajeitar os tamanhos dos plots de acionamento
798         MaxY1 =  ceil(MaxY1);
799         MinY1 = floor(MinY1);
800         subplot(2,4,1)
801         ylim([MinY1 MaxY1]);
802         subplot(2,4,2)
803         ylim([MinY1 MaxY1]);
804         subplot(2,4,3)
805         ylim([MinY1 MaxY1]);
806         subplot(2,4,4)
807         ylim([MinY1 MaxY1]);
808         MaxY1 = 0;
809         MinY1 = 0;
810         %Ajeitar os tamanhos dos plots de resposta
811         MaxY2 = 2*round(MaxY2,1);
812         subplot(2,4,5)
813         ylim([0 MaxY2]);
814         subplot(2,4,6)
815         ylim([0 MaxY2]);
816         subplot(2,4,7)
817         ylim([0 MaxY2]);
818         subplot(2,4,8)
819         ylim([0 MaxY2]);
820         MaxY2 = 0;
821
822
```

```
823             xlabel(XName);
824             xlim([X(1) X(end)]);
825             %ylabel(YRName);
826             suptitle(CName);
827 
828             set(gcf,'PaperPositionMode','auto');
829             saveas(gcf,FName);
830 
831             Sim_M = Sim_M*(FScale);
832         end
833     else
834         %load(LastData);
835     end
836 
837     %% Gera os arquivos do controlador
838 
839     if GenerateFiles
840         FileCell = [FileCell ;...
841             {'modules','Setup_Controle_Defines_Coeficientes.h',NFiles+1}];
842         NFiles = NFiles + 1;
843         fid = fopen(FileCell{NFiles,2},'w');
844         Tmp1 = [ScriptName,...
845         Custom_Files_SeqAssist('#define Controle_Amostragem ',T*10^6,' \n')...
846         Custom_Files_SeqAssist('#define C_DD_Frente_K_Num ',...
847         CDF_NumVec_Val(1))...
848         Custom_Files_SeqAssist('#define C_DD_Frente_',LengthVec(2:end),...
849         '_Num ',CDF_NumVec_Val(2:end))...
850         Custom_Files_SeqAssist('#define C_DD_Frente_',LengthVec(2:end),...
851         '_Den ',CDF_DenVec_Val)...
852         Custom_Files_SeqAssist('#define C_ED_Frente_K_Num ',...
853         CEF_NumVec_Val(1))...
854         Custom_Files_SeqAssist('#define C_ED_Frente_',LengthVec(2:end),...
855         '_Num ',CEF_NumVec_Val(2:end))...
856         Custom_Files_SeqAssist('#define C_ED_Frente_',LengthVec(2:end),...
857         '_Den ',CEF_DenVec_Val)...
858         Custom_Files_SeqAssist('#define C_DD_Tras_K_Num ',...
859         CDT_NumVec_Val(1)),...
860         Custom_Files_SeqAssist('#define C_DD_Tras_',LengthVec(2:end),...
861         '_Num ',CDT_NumVec_Val(2:end)),...
862         Custom_Files_SeqAssist('#define C_DD_Tras_',LengthVec(2:end),...
863         '_Den ',CDT_DenVec_Val),...
864         Custom_Files_SeqAssist('#define C_ED_Tras_K_Num ',...
865         CET_NumVec_Val(1)),...
```

```
866       Custom_Files_SeqAssist('#define C_ED_Tras_',LengthVec(2:end),...
867       '_Num ',CET_NumVec_Val(2:end)),...
868       Custom_Files_SeqAssist('#define C_ED_Tras_',LengthVec(2:end),...
869       '_Den ',CET_DenVec_Val)...
870       ];
871       %Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
872       fprintf(fid,Tmp1);
873       fclose(fid);
874   end
875
876   %% Gerar o arquivo de setup do controlador normal
877
878
879
880   if GenerateFiles
881       FileCell = [FileCell ; {'modules','Setup_Controle_Reset.h',NFiles+1}];
882       NFiles = NFiles + 1;
883       fid = fopen(FileCell{NFiles,2},'w');
884       Tmp1 = ScriptName;
885       Tmp1 = [Tmp1 '#include "subrotinas/Subrotinas_Controle_Reset.h" \n'];
886       fprintf(fid,Tmp1);
887
888       fclose(fid);
889   end
890
891   if GenerateFiles
892       FileCell = [FileCell...
893       ; {'subrotinas','Subrotinas_Controle_Reset.h',NFiles+1}];
894       NFiles = NFiles + 1;
895       fid = fopen(FileCell{NFiles,2},'w');
896       Tmp1 = [ScriptName...
897       'VarControle_KpE       = 0;@'...
898       'VarControle_KpD       = 0;@'...
899       ];
900       if MaxLength < 3
901           if MaxLength == 2
902               Tmp1 = [Tmp1...
903               'VarControle_DDA    = 0;@'...
904               'VarControle_EDA    = 0;@'...
905               'VarControle_DDS    = 0;@'...
906               'VarControle_EDS    = 0;@'...
907               'VarControle_CDNum = 0;@'...
908               'VarControle_CDDen = 0;@'...
```

```
909                 'VarControle_CENum = 0;@'...
910                 'VarControle_CEDen = 0;@'...
911                 ];
912             end
913         else
914             Tmp1 = [Tmp1...
915             Custom_Files_SeqAssist(...
916             'VarControle_DDA[',LengthVec(2:end),']    = 0;')...
917             Custom_Files_SeqAssist(...
918             'VarControle_EDA[',LengthVec(2:end),']    = 0;')...
919             Custom_Files_SeqAssist(...
920             'VarControle_DDS[',LengthVec(2:end),']    = 0;')...
921             Custom_Files_SeqAssist(...
922             'VarControle_EDS[',LengthVec(2:end),']    = 0;')...
923             Custom_Files_SeqAssist(...
924             'VarControle_CDNum[',LengthVec(2:end),'] = 0;')...
925             Custom_Files_SeqAssist(...
926             'VarControle_CDDen[',LengthVec(2:end),'] = 0;')...
927             Custom_Files_SeqAssist(...
928             'VarControle_CENum[',LengthVec(2:end),'] = 0;')...
929             Custom_Files_SeqAssist(...
930             'VarControle_CEDen[',LengthVec(2:end),'] = 0;')...
931             ];
932         end
933
934         Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
935         fprintf(fid,Tmp1);
936         fclose(fid);
937     end
938
939
940
941     %% Gera a subrotina de decisão de controle
942
943     if GenerateFiles
944         FileCell = [FileCell ;{'subrotinas','Subrotinas_Controle.h',NFiles+1}];
945         NFiles = NFiles + 1;
946         fid = fopen(FileCell{NFiles,2},'w');
947
948         Tmp1 = [ScriptName...
949         'if (Tempo - amostra >= Controle_Amostragem){@'...
950         '    //Serial.print("IniciaControle");@'...
951         '    #include "Subrotinas_ML_PrintAll.h"@'...
```

```
952       '    #include "Subrotinas_Controle_TF.h"@'...
953       '    amostra = Tempo;@'...
954       '}else{@'...
955       '    if (amostra > Tempo || amostra ≤ 0){@'...
956       '        amostra = Tempo;@'...
957       '    }@'...
958       '}\n@'...
959       ];
960
961       Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
962       fprintf(fid,Tmp1);
963       fclose(fid);
964   end
965
966   %% Gera a subrotina do controlador
967
968   if GenerateFiles
969       FileCell = [FileCell ; {'subrotinas','Subrotinas_Controle_TF.h',NFiles+1}];
970       NFiles = NFiles + 1;
971       fid = fopen(FileCell{NFiles,2},'w');
972
973       Tmp1 = ScriptName;
974       if CVer ≠ 0
975           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '// Atualiza a posição')];
976           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '    #include ...
                  "Subrotinas_Controle_GenerateTargetPosition.h"')];
977           %Sepranando as partes:
978           %Atualização da posição/velocidade
979           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '// Faz a leitura dos contadores ...
                  via desativador de interrupção (Testanto sem)')];
980           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '// Agora segue a vida')];
981           %Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_DA    = ...
                  analogRead(Enc_DA);')];
982           %Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_EA    = ...
                  analogRead(Enc_EA);')];
983           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '    Vel_D    = float(C_DD);')];
984           Tmp1 = [Tmp1 Custom_Files_SeqAssist( '    Vel_E    = float(C_ED);')];
985
986           Tmp1 = [Tmp1 Custom_Files_SeqAssist('    StartV_D = Vel_D;')];
987           Tmp1 = [Tmp1 Custom_Files_SeqAssist('    StartV_E = Vel_E;')];
988           if ¬PosCon
989               Tmp1 = [Tmp1 Custom_Files_SeqAssist('    aux1  = float(Tempo − ...
                      amostra)/',10^6,'.0;')];
```

```
990             Tmp1 = [Tmp1 Custom_Files_SeqAssist('     Vel_D = (Vel_D - ...
                    VENC_D)/aux1;')];
991             Tmp1 = [Tmp1 Custom_Files_SeqAssist('     Vel_E = (Vel_E - ...
                    VENC_E)/aux1;')];
992         end
993         Tmp1 = [Tmp1 Custom_Files_SeqAssist('     VENC_D = StartV_D;')];
994         Tmp1 = [Tmp1 Custom_Files_SeqAssist('     VENC_E = StartV_E;')];
995         %O resto
996         if MaxLength < 3
997             Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Motor Direito')];
998             Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Realimentação de Estados')];
999             Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_DDS = PosD - ...
                    Vel_D;')];
1000            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Erro_D = VarControle_DDS;')];
1001            if MaxLength > 1
1002                Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'VarControle_DDS = ...
                        VarControle_DDS - VarControle_DDA*VarControle_CDDen;')];
1003            end
1004            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Caminho Direto Principal')];
1005            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Dear God, its empty!')];
1006
1007            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Cálculo da Ação de ...
                    Controle')];
1008            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = ...
                    VarControle_DDS*VarControle_KpD;')];
1009            if MaxLength > 1
1010                Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'Controle_ACD = ...
                        Controle_ACD + VarControle_DDA*VarControle_CDNum;')];
1011                Tmp1 = [Tmp1 Custom_Files_SeqAssist( '// Atualização das ...
                        Variáveis de Estado')];
1012                Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'VarControle_DDA = ...
                        VarControle_DDS;')];
1013            end
1014            %fprintf(fid,Tmp1);
1015
1016
1017            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Motor Esquerdo')];
1018            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Realimentação de Estados')];
1019            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDS = PosE - ...
                    Vel_E;')];
1020            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Erro_E = VarControle_EDS;')];
1021            if MaxLength > 1
1022                Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDS = ...
```

```
                    VarControle_EDS - VarControle_EDA*VarControle_CEDen;')];
1023            end
1024            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Caminho Direto Principal')];
1025            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Dear God, its empty!')];
1026
1027            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Cálculo da Ação de ...
                    Controle')];
1028            Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'Controle_ACE = ...
                    VarControle_EDS*VarControle_KpE;')];
1029            if MaxLength > 1
1030                Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'Controle_ACE = ...
                        Controle_ACE + VarControle_EDA*VarControle_CENum;')];
1031                Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Atualização das ...
                        Variáveis de Estado')];
1032                Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDA = ...
                        VarControle_EDS;')];
1033            end
1034        else
1035            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Motor Direito')];
1036            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Realimentação de Estados')];
1037            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_DDS[0] = PosD - ...
                    Vel_D;')];
1038            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Erro_D = VarControle_DDS[0];')];
1039            Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'VarControle_DDS[0] = ...
                    VarControle_DDS[0] - VarControle_DDA[' , LengthVec(2:end) ...
                    ,']*VarControle_CDDen[',LengthVec(2:end),'];')];
1040
1041            Tmp1 = [Tmp1 Custom_Files_SeqAssist( '// Caminho Direto Principal')];
1042            Tmp1 = [Tmp1 Custom_Files_SeqAssist( 'VarControle_DDS[', ...
                    LengthVec(3:end) ,'] = VarControle_DDA[', LengthVec(2:end-1) ...
                    ,'];')];
1043
1044            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Cálculo da Ação de ...
                    Controle')];
1045            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = ...
                    VarControle_DDS[0]*VarControle_KpD;')];
1046            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = Controle_ACD + ...
                    VarControle_DDA[', LengthVec(2:end) ...
                    ,']*VarControle_CDNum[',LengthVec(2:end),'];')];
1047            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Atualização das Variáveis ...
                    de Estado')];
1048            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_DDA[', ...
                    LengthVec(2:end) ,'] = VarControle_DDS[', ...
```

```
                LengthVec(2:end),'];')];
1049            %fprintf(fid,Tmp1);
1050
1051
1052            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Motor Esquerdo')];
1053            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Realimentação de Estados')];
1054            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDS[0] = PosE - ...
                    Vel_E;')];
1055            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Erro_E = VarControle_EDS[0];')];
1056
1057            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDS[0] = ...
                    VarControle_EDS[0] - VarControle_EDA[',LengthVec(2:end) ...
                    ,']*VarControle_CEDen[', LengthVec(2:end) ,'];')];
1058
1059            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Caminho Direto Principal')];
1060            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDS[', ...
                    LengthVec(3:end),'] = VarControle_EDA[' , LengthVec(2:end-1) ...
                    ,'];')];
1061
1062            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Cálculo da Ação de ...
                    Controle')];
1063            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACE = ...
                    VarControle_EDS[0]*VarControle_KpE;')];
1064            Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACE = Controle_ACE + ...
                    VarControle_EDA[', LengthVec(2:end) ,']*VarControle_CENum[', ...
                    LengthVec(2:end) ,'];')];
1065            Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Atualização das Variáveis ...
                    de Estado')];
1066            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_EDA[', ...
                    LengthVec(2:end) ,'] = VarControle_EDS[', LengthVec(2:end) ...
                    ,'];')];
1067            Tmp1 = [Tmp1 Custom_Files_SeqAssist('\n// Prints')];
1068        end
1069    else
1070        Tmp1 = [Tmp1 Custom_Files_SeqAssist('// Controle Desligado')];
1071        Tmp1 = [Tmp1 Custom_Files_SeqAssist(' ')];
1072        Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (MEP > 0){')];% Acionou
1073        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if (MEN > 0){')]; %Esquerdo p trás
1074        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if (MDN > 0){')]; %Direito p ...
                trás
1075        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACD = ...
                ',-MaxVel,';')];
1076        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACE = ...
```

```
            ',-MaxVel,';')];
1077        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito p frente
1078        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACD =  ...
            ',MaxVel*Sim_TurnAdj,';')];
1079        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACE = ...
            ',-MaxVel*Sim_TurnAdj,';')];
1080        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
1081        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];%Esquerdo p frente
1082        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if (MDN > 0){')]; %Direito p ...
            trás
1083        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACD = ...
            ',-MaxVel*Sim_TurnAdj,';')];
1084        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACE =  ...
            ',MaxVel*Sim_TurnAdj,';')];
1085        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito p frente
1086        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACD =  ...
            ',MaxVel,';')];
1087        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      Controle_ACE =  ...
            ',MaxVel,';')];
1088        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
1089        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
1090        Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];%Aqui ele para os 2
1091        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Controle_ACD =  ',0,';')];
1092        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Controle_ACE =  ',0,';')];
1093        Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
1094    end
1095
1096    fprintf(fid,Tmp1);
1097
1098    fclose(fid);
1099 end
```

Arduino_SS_GeraSetup

Função que gera a maioria dos arquivos que precedem à função loop() do Arduino, de declaração de variáveis até inicialização dos pinos e canais de comunicação.

Retorne à tabela clicando nesta frase

```
1 %% Prepara o nome
2     ScriptName = '// Arduino_SS_GeraSetup \n';
```

```
3
4
5  %% Declara as variáveis
6  %formato {'Tipo','Nome',Valor(es) Inicial(is), Índice}
7
8  VarsProg = [];
9  %VarsProg = {'','',0,n};
10 VarsProg = [VarsProg ; {'char','C',0,1}];
11 VarsProg = [VarsProg ; {'int','AnalogAux',0,VarsProg{end,4}+1}];
12 VarsProg = [VarsProg ; {'int','Sensor',0,VarsProg{end,4}+1}];
13 VarsProg = [VarsProg ; {'int','Operation',0,VarsProg{end,4}+1}];
14 VarsProg = [VarsProg ; {'int','cont',0,VarsProg{end,4}+1}];
15 VarsProg = [VarsProg ; {'int','Case',0,VarsProg{end,4}+1}];
16 VarsProg = [VarsProg ; {'float','PosE',0,VarsProg{end,4}+1}];
17 VarsProg = [VarsProg ; {'float','PosD',0,VarsProg{end,4}+1}];
18 VarsProg = [VarsProg ; {'unsigned long','Read',0,VarsProg{end,4}+1}];
19 VarsProg = [VarsProg ; {'volatile byte','Counter_DDP',0,VarsProg{end,4}+1}];
20 VarsProg = [VarsProg ; {'volatile byte','Counter_EDP',0,VarsProg{end,4}+1}];
21 VarsProg = [VarsProg ; {'volatile byte','Counter_DDN',0,VarsProg{end,4}+1}];
22 VarsProg = [VarsProg ; {'volatile byte','Counter_EDN',0,VarsProg{end,4}+1}];
23 VarsProg = [VarsProg ; {'int','Counter_DA',0,VarsProg{end,4}+1}];
24 VarsProg = [VarsProg ; {'int','Counter_EA',0,VarsProg{end,4}+1}];
25 VarsProg = [VarsProg ; {'long','C_DD',0,VarsProg{end,4}+1}];
26 VarsProg = [VarsProg ; {'long','C_ED',0,VarsProg{end,4}+1}];
27 VarsProg = [VarsProg ; {'float','DistVec_F',0,VarsProg{end,4}+1}];
28 VarsProg = [VarsProg ; {'float','DistVec_T',0,VarsProg{end,4}+1}];
29 VarsProg = [VarsProg ; {'unsigned long','amostra',0,VarsProg{end,4}+1}];
30 VarsProg = [VarsProg ; {'unsigned long','Timer',0,VarsProg{end,4}+1}];
31 VarsProg = [VarsProg ; {'volatile unsigned ...
       long','Timer_AuxD',0,VarsProg{end,4}+1}];
32 VarsProg = [VarsProg ; {'volatile unsigned ...
       long','Timer_AuxE',0,VarsProg{end,4}+1}];
33 VarsProg = [VarsProg ; {'unsigned long','Tempo',0,VarsProg{end,4}+1}];
34 VarsProg = [VarsProg ; {'float','aux1',0,VarsProg{end,4}+1}];
35 VarsProg = [VarsProg ; {'float','StartV_E',0,VarsProg{end,4}+1}];
36 VarsProg = [VarsProg ; {'float','StartV_D',0,VarsProg{end,4}+1}];
37 VarsProg = [VarsProg ; {'float','VENC_E',0,VarsProg{end,4}+1}];
38 VarsProg = [VarsProg ; {'float','VENC_D',0,VarsProg{end,4}+1}];
39 VarsProg = [VarsProg ; {'float','Vel_E',0,VarsProg{end,4}+1}];
40 VarsProg = [VarsProg ; {'float','Vel_D',0,VarsProg{end,4}+1}];
41 VarsProg = [VarsProg ; {'float','Controle_ACD',0,VarsProg{end,4}+1}];
42 VarsProg = [VarsProg ; {'float','Controle_ACE',0,VarsProg{end,4}+1}];
43 VarsProg = [VarsProg ; {'float','MEP',0,VarsProg{end,4}+1}];
```

```
44 VarsProg = [VarsProg ; {'float','VMEP',0,VarsProg{end,4}+1}];
45 VarsProg = [VarsProg ; {'float','MEN',0,VarsProg{end,4}+1}];
46 VarsProg = [VarsProg ; {'float','VMEN',0,VarsProg{end,4}+1}];
47 VarsProg = [VarsProg ; {'float','MDN',0,VarsProg{end,4}+1}];
48 VarsProg = [VarsProg ; {'float','VMDN',0,VarsProg{end,4}+1}];
49 VarsProg = [VarsProg ; {'float','MDP',0,VarsProg{end,4}+1}];
50 VarsProg = [VarsProg ; {'float','VMDP',0,VarsProg{end,4}+1}];
51 VarsProg = [VarsProg ; {'float','V_Start_EF',0,VarsProg{end,4}+1}];
52 VarsProg = [VarsProg ; {'float','V_Start_DF',0,VarsProg{end,4}+1}];
53 VarsProg = [VarsProg ; {'float','ErrorVector',[0 0 0 0],VarsProg{end,4}+1}];
54 VarsProg = [VarsProg ; {'char','aux_i',0,VarsProg{end,4}+1}];
55 VarsProg = [VarsProg ; {'int','Section',0,VarsProg{end,4}+1}];
56 VarsProg = [VarsProg ; {'int','FT_TPFDE',0,VarsProg{end,4}+1}];
57 VarsProg = [VarsProg ; {'int','PrevFT_TPFDE',0,VarsProg{end,4}+1}];
58 VarsProg = [VarsProg ; {'unsigned long','SensorTimer',0,VarsProg{end,4}+1}];
59 VarsProg = [VarsProg ; {'int','SensorSwitch',0,VarsProg{end,4}+1}];
60 VarsProg = [VarsProg ; {'float','DistT',0,VarsProg{end,4}+1}];
61 VarsProg = [VarsProg ; {'float','DistF',0,VarsProg{end,4}+1}];
62 VarsProg = [VarsProg ; {'float','Erro_E',0,VarsProg{end,4}+1}];
63 VarsProg = [VarsProg ; {'float','Erro_D',0,VarsProg{end,4}+1}];
64 VarsProg = [VarsProg ; {'float','VarControle_KpE',0,VarsProg{end,4}+1}];
65 VarsProg = [VarsProg ; {'float','VarControle_KpD',0,VarsProg{end,4}+1}];
66 if MaxLength > 2 %Denominador de mais de 1 termo (... + z^2 + 3z + 4)
67     VarsProg = [VarsProg ; ...
           {'float','VarControle_DDA',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
68     VarsProg = [VarsProg ; ...
           {'float','VarControle_EDA',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
69     VarsProg = [VarsProg ; ...
           {'float','VarControle_DDS',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
70     VarsProg = [VarsProg ; ...
           {'float','VarControle_EDS',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
71     VarsProg = [VarsProg ; ...
           {'float','VarControle_CDNum',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
72     VarsProg = [VarsProg ; ...
           {'float','VarControle_CDDen',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
73     VarsProg = [VarsProg ; ...
           {'float','VarControle_CENum',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
74     VarsProg = [VarsProg ; ...
           {'float','VarControle_CEDen',zeros(1,MaxLength-1),VarsProg{end,4}+1}];
75 else
76     if MaxLength == 2 %Denominador de 1 termo (z + 1)
77         VarsProg = [VarsProg ; {'float','VarControle_DDA',0,VarsProg{end,4}+1}];
78         VarsProg = [VarsProg ; {'float','VarControle_EDA',0,VarsProg{end,4}+1}];
```

```
79          VarsProg = [VarsProg ; {'float','VarControle_DDS',0,VarsProg{end,4}+1}];
80          VarsProg = [VarsProg ; {'float','VarControle_EDS',0,VarsProg{end,4}+1}];
81          VarsProg = [VarsProg ; ...
                {'float','VarControle_CDNum',0,VarsProg{end,4}+1}];
82          VarsProg = [VarsProg ; ...
                {'float','VarControle_CDDen',0,VarsProg{end,4}+1}];
83          VarsProg = [VarsProg ; ...
                {'float','VarControle_CENum',0,VarsProg{end,4}+1}];
84          VarsProg = [VarsProg ; ...
                {'float','VarControle_CEDen',0,VarsProg{end,4}+1}];
85      else %Controlador é proporcional apenas
86          VarsProg = [VarsProg ; {'float','VarControle_DDS',0,VarsProg{end,4}+1}];
87          VarsProg = [VarsProg ; {'float','VarControle_EDS',0,VarsProg{end,4}+1}];
88      end
89  end
90
91  %% Gera o arquivo que declara elas tudo
92
93  if GenerateFiles
94      FileCell = [FileCell ;...
95          {'modules','Setup_Declaracao_Variaveis.h',NFiles+1}];
96      NFiles = NFiles + 1;
97      fid = fopen(FileCell{NFiles,2},'w');
98
99      Tmp1 = ScriptName;
100     for n = 1:length(VarsProg)
101         if length(VarsProg{n,3}) == 1
102             Tmp1 = [Tmp1 Custom_Files_SeqAssist(VarsProg{n,1},' ',...
103                 VarsProg{n,2},';// Variavel Numero ',n)];
104         else
105             Tmp1 = [Tmp1 Custom_Files_SeqAssist(VarsProg{n,1},' ',...
106                 VarsProg{n,2},'[',length(VarsProg{n,3}),...
107                 '];// Variavel Numero ',n)];
108         end
109     end
110     Tmp1;
111     fprintf(fid,Tmp1);
112
113     fclose(fid);
114 end
115
116 %% Gera o arquivo que inicializa tudo
117
```

```
118 if GenerateFiles
119     FileCell = [FileCell ; {'modules','Setup_Inicializacao.h',NFiles+1}];
120     NFiles = NFiles + 1;
121     fid = fopen(FileCell{NFiles,2},'w');
122 
123     Tmp1 = ScriptName;
124     for n = 1:length(VarsProg)
125         if length(VarsProg{n,3}) == 1
126             Tmp1 = [Tmp1 Custom_Files_SeqAssist(VarsProg{n,2},...
127                 ' = ',VarsProg{n,3},';')];
128         else
129             A = 0:length(VarsProg{n,3})-1;
130             Tmp1 = [Tmp1 Custom_Files_SeqAssist(VarsProg{n,2},...
131                 '[',A,'] = ',VarsProg{n,3},';')];
132         end
133     end
134     Tmp1;
135     fprintf(fid,Tmp1);
136     fclose(fid);
137 end
138 
139 
140 %% Gera os arquivos que juntam os setups
141 
142 if GenerateFiles
143     FileCell = [FileCell ; {'modules','SetupGeral_Global.h',NFiles+1}];
144     NFiles = NFiles + 1;
145     fid = fopen(FileCell{NFiles,2},'w');
146 
147     Tmp1 = ScriptName;
148     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include <SoftwareSerial.h>')];
149     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pra função que desliga ...
        interrupções, deve resolver os saltos')];
150     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//#include <util/atomic.h>')];
151     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Pinagens.h"')];
152     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Configuracoes.h"')];
153     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Defines.h"')];
154     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
        "Setup_Controle_Defines_Coeficientes.h"')];
155     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
        "Setup_Declaracao_Variaveis.h"')];
156     Tmp1 = [Tmp1 Custom_Files_SeqAssist('SoftwareSerial bluetooth(P_rx, P_tx); ...
        ')];
```

```
157     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Sim_Mag 1 ')];
158     if Run_Chao == 1
159         Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Serial.print ...
                //Serial.print ')];
160     end
161     Tmp1 = [Tmp1 Custom_Files_SeqAssist('void inter_DDCounter_FWD(){ ')];
162     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if((millis() - Timer_AuxD) > ...
        ',EncoderDelay,'){')];
163     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Timer_AuxD = millis();')];
164     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_DDP = Counter_DDP + 1;')];
165     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
166     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
167     Tmp1 = [Tmp1 Custom_Files_SeqAssist('void inter_EDCounter_FWD(){  ')];
168     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if((millis() - Timer_AuxE) > ...
        ',EncoderDelay,'){')];
169     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Timer_AuxE = millis();')];
170     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_EDP = Counter_EDP + 1;')];
171     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
172     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
173     Tmp1 = [Tmp1 Custom_Files_SeqAssist('void inter_DDCounter_BCK(){  ')];
174     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if((millis() - Timer_AuxD) > ...
        ',EncoderDelay,'){')];
175     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Timer_AuxD = millis();')];
176     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_DDN = Counter_DDN + 1;')];
177     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
178     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
179     Tmp1 = [Tmp1 Custom_Files_SeqAssist('void inter_EDCounter_BCK(){  ')];
180     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if((millis() - Timer_AuxE) > ...
        ',EncoderDelay,'){')];
181     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Timer_AuxE = millis();')];
182     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_EDN = Counter_EDN + 1;')];
183     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
184     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
185     fprintf(fid,Tmp1);
186     fclose(fid);
187 end
188
189 if GenerateFiles
190     FileCell = [FileCell ; {'modules','SetupGeral_Setup.h',NFiles+1}];
191     NFiles = NFiles + 1;
192     fid = fopen(FileCell{NFiles,2},'w');
193
194     Tmp1 = ScriptName;
```

```
195     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Setup_Inicializacao.h"')];
196     Tmp1 = [Tmp1 Custom_Files_SeqAssist('\n')];
197     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.begin(',S_BaudRate,');')];
198     Tmp1 = [Tmp1 Custom_Files_SeqAssist('bluetooth.begin(',B_BaudRate,');')];
199     Tmp1 = [Tmp1 Custom_Files_SeqAssist('\n')];
200     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Motores')];
201     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(EP, OUTPUT);')];
202     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(EN, OUTPUT);')];
203     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(DN, OUTPUT);')];
204     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(DP, OUTPUT);')];
205     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Ultrassom')];
206     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(TrigF, OUTPUT);')];
207     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(EchoF,  INPUT);')];
208     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//pinMode(TrigT, OUTPUT);')];
209     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(EchoT,  INPUT);')];
210     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Encoders')];
211     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(Enc_DD,  INPUT);')];
212     %Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(Enc_DA,  INPUT);')];
213     Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(Enc_ED,  INPUT);')];
214     %Tmp1 = [Tmp1 Custom_Files_SeqAssist('pinMode(Enc_EA,  INPUT);')];
215     if Run_Chao == 0
216         Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("\\n Waiting \\n");')];
217     end
218     fprintf(fid,Tmp1);
219     fclose(fid);
220 end
```

Arduino_SS_GeraAcionamento

Função que gera os arquivos para acionar os motores

Retorne à tabela clicando nesta frase

```
1 %% Prepara o nome
2
3 ScriptName = '// Arduino_SS_GeraAcionamento \n';
4
5
6 %% Gerar as subrotinas de acionamento do controlador
7
8 if GenerateFiles
```

```
9       FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_Controle_SetupTf_DF.h',NFiles+1}];
10      NFiles = NFiles + 1;
11      fid = fopen(FileCell{NFiles,2},'w');
12      Tmp1 = ScriptName;
13      Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_KpD    = ...
            C_DD_Frente_K_Num;')];
14      if MaxLength > 2
15          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CDNum[',LengthVec(2:end),'] = ...
                C_DD_Frente_',LengthVec(2:end),'_Num;')];
16          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CDDen[',LengthVec(2:end),'] = ...
                C_DD_Frente_',LengthVec(2:end),'_Den;')];
17      else
18          if MaxLength > 1
19              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CDNum = ...
                    C_DD_Frente_0_Num;')];
20              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CDDen = ...
                    C_DD_Frente_0_Den;')];
21          end
22      end
23      fprintf(fid,Tmp1);
24      fclose(fid);
25
26      FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_Controle_SetupTf_DT.h',NFiles+1}];
27      NFiles = NFiles + 1;
28      fid = fopen(FileCell{NFiles,2},'w');
29      Tmp1 = ScriptName;
30      Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_KpD     = ...
            C_DD_Tras_K_Num;')];
31      if MaxLength > 2
32          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CDNum[',LengthVec(2:end),'] = ...
                C_DD_Tras_',LengthVec(2:end),'_Num;')];
33          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CDDen[',LengthVec(2:end),'] = ...
                C_DD_Tras_',LengthVec(2:end),'_Den;')];
34      else
35          if MaxLength > 1
36              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CDNum = ...
                    C_DD_Tras_0_Num;')];
```

```
37              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CDDen = ...
                    C_DD_Tras_0_Den;')];
38          end
39      end
40      fprintf(fid,Tmp1);
41      fclose(fid);
42
43      FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_Controle_SetupTf_EF.h',NFiles+1}];
44      NFiles = NFiles + 1;
45      fid = fopen(FileCell{NFiles,2},'w');
46      Tmp1 = ScriptName;
47      Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_KpE     = ...
            C_ED_Frente_K_Num;')];
48      if MaxLength > 2
49          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CENum[',LengthVec(2:end),'] = ...
                C_ED_Frente_',LengthVec(2:end),'_Num;')];
50          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CEDen[',LengthVec(2:end),'] = ...
                C_ED_Frente_',LengthVec(2:end),'_Den;')];
51      else
52          if MaxLength > 1
53              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CENum = ...
                    C_ED_Frente_0_Num;')];
54              Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CEDen = ...
                    C_ED_Frente_0_Den;')];
55          end
56      end
57      fprintf(fid,Tmp1);
58      fclose(fid);
59
60      FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_Controle_SetupTf_ET.h',NFiles+1}];
61      NFiles = NFiles + 1;
62      fid = fopen(FileCell{NFiles,2},'w');
63      Tmp1 = ScriptName;
64      Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_KpE     = C_ED_Tras_K_Num;')];
65      if MaxLength > 2
66          Tmp1 = [Tmp1 ...
                Custom_Files_SeqAssist('VarControle_CENum[',LengthVec(2:end),'] = ...
                C_ED_Tras_',LengthVec(2:end),'_Num;')];
67          Tmp1 = [Tmp1 ...
```

```
            Custom_Files_SeqAssist('VarControle_CEDen[',LengthVec(2:end),'] = ...
            C_ED_Tras_',LengthVec(2:end),'_Den;')];
    else
        if MaxLength > 1
            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CENum = ...
                C_ED_Tras_0_Num;')];
            Tmp1 = [Tmp1 Custom_Files_SeqAssist('VarControle_CEDen = ...
                C_ED_Tras_0_Den;')];
        end
    end
    fprintf(fid,Tmp1);
    fclose(fid);

    FileCell = [FileCell ; ...
        {'subrotinas','Subrotinas_Controle_SetupTf_Mover_Frente.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = ScriptName;
    Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_EF.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_DF.h"')];
    fprintf(fid,Tmp1);
    fclose(fid);

    FileCell = [FileCell ; ...
        {'subrotinas','Subrotinas_Controle_SetupTf_Mover_Tras.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = ScriptName;
    Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_ET.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_DT.h"')];
    fprintf(fid,Tmp1);
    fclose(fid);

    FileCell = [FileCell ; ...
        {'subrotinas','Subrotinas_Controle_SetupTf_Mover_Direita.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = ScriptName;
    Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
```

```
        "Subrotinas_Controle_SetupTf_EF.h"')];
100     Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_DT.h"')];
101     fprintf(fid,Tmp1);
102     fclose(fid);
103
104     FileCell = [FileCell ; ...
        {'subrotinas','Subrotinas_Controle_SetupTf_Mover_Esquerda.h',NFiles+1}];
105     NFiles = NFiles + 1;
106     fid = fopen(FileCell{NFiles,2},'w');
107     Tmp1 = ScriptName;
108     Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_ET.h"')];
109     Tmp1 = [Tmp1 Custom_Files_SeqAssist( '#include ...
        "Subrotinas_Controle_SetupTf_DF.h"')];
110     fprintf(fid,Tmp1);
111     fclose(fid);
112
113     FileCell = [FileCell ; {'subrotinas', ...
        'Subrotinas_Controle_SetupTf_Mover_Parar.h',NFiles+1}];
114     NFiles = NFiles + 1;
115     fid = fopen(FileCell{NFiles,2},'w');
116     Tmp1 = ScriptName;
117     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
        '//detachInterrupt(digitalPinToInterrupt(Enc_DD));')];
118     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
        '//detachInterrupt(digitalPinToInterrupt(Enc_ED));')];
119     fprintf(fid,Tmp1);
120     fclose(fid);
121
122     FileCell = [FileCell ; {'subrotinas', ...
        'Subrotinas_Controle_AtivaInterrupts_DF.h',NFiles+1}];
123     NFiles = NFiles + 1;
124     fid = fopen(FileCell{NFiles,2},'w');
125     Tmp1 = ScriptName;
126     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
        'detachInterrupt(digitalPinToInterrupt(Enc_DD));')];
127     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
        'attachInterrupt(digitalPinToInterrupt(Enc_DD), inter_DDCounter_FWD, ...
        InterruptType);')];
128     fprintf(fid,Tmp1);
129     fclose(fid);
130
```

```
131     FileCell = [FileCell ; {'subrotinas', ...
            'Subrotinas_Controle_AtivaInterrupts_DT.h',NFiles+1}];
132     NFiles = NFiles + 1;
133     fid = fopen(FileCell{NFiles,2},'w');
134     Tmp1 = ScriptName;
135     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'detachInterrupt(digitalPinToInterrupt(Enc_DD));')];
136     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'attachInterrupt(digitalPinToInterrupt(Enc_DD), inter_DDCounter_BCK, ...
            InterruptType);')];
137     fprintf(fid,Tmp1);
138     fclose(fid);
139
140     FileCell = [FileCell ; {'subrotinas', ...
            'Subrotinas_Controle_AtivaInterrupts_EF.h',NFiles+1}];
141     NFiles = NFiles + 1;
142     fid = fopen(FileCell{NFiles,2},'w');
143     Tmp1 = ScriptName;
144     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'detachInterrupt(digitalPinToInterrupt(Enc_ED));')];
145     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'attachInterrupt(digitalPinToInterrupt(Enc_ED), inter_EDCounter_FWD, ...
            InterruptType);')];
146     fprintf(fid,Tmp1);
147     fclose(fid);
148
149     FileCell = [FileCell ; {'subrotinas', ...
            'Subrotinas_Controle_AtivaInterrupts_ET.h',NFiles+1}];
150     NFiles = NFiles + 1;
151     fid = fopen(FileCell{NFiles,2},'w');
152     Tmp1 = ScriptName;
153     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'detachInterrupt(digitalPinToInterrupt(Enc_ED));')];
154     Tmp1 = [Tmp1 Custom_Files_SeqAssist( ...
            'attachInterrupt(digitalPinToInterrupt(Enc_ED), inter_EDCounter_BCK, ...
            InterruptType);')];
155     fprintf(fid,Tmp1);
156     fclose(fid);
157
158 end
159
160 %% Gera a subrotina de configuração de acionamento
161
```

```
162 if GenerateFiles
163     FileCell = [FileCell ; {'subrotinas', 'Subrotinas_ConfigVel.h',NFiles+1}];
164     NFiles = NFiles + 1;
165     fid = fopen(FileCell{NFiles,2},'w');
166     Tmp1 = ScriptName;
167
168     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP = min(MEP*(MEP > 0) + MEN*(MEN > ...
            0),MaxVel);')];
169     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN = (MEN > 0);')];
170     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP = min(MDP*(MDP > 0) + MDN*(MDN > ...
            0),MaxVel);')];
171     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN = (MDN > 0);\n ')];
172
173     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if (EngineTest == 1 || EngineTest == ...
            2)')];
174     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACE = MaxVel;')];
175     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = MaxVel;')];
176     Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (MEP == 0){')];
177     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACE = 0;')];
178     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
179     Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (MDP == 0){')];
180     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = 0;')];
181     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
182     Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (MEN > 0){')];
183     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACE = -Controle_ACE;')];
184     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
185     Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (MDN > 0){')];
186     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Controle_ACD = -Controle_ACD;')];
187     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
188     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif\n')];
189
190     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Atualizar os contadores aqui')];
191     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    aux_i       = Counter_DDP - ...
            Counter_DDN;')];
192     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_DDP = 0;')];
193     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_DDN = 0;')];
194     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    C_DD        = C_DD + ...
            long(aux_i);')];
195     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    aux_i       = Counter_EDP - ...
            Counter_EDN;')];
196     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_EDP = 0;')];
197     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Counter_EDN = 0;')];
198     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    C_ED        = C_ED + ...
```

```
            long(aux_i);')];
199         Tmp1 = [Tmp1 Custom_Files_SeqAssist('Tempo = micros();')];
200         Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if (EngineTest == 3 || EngineTest == ...
            4 || MainProg == 1)')];
201         Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Controle.h"')];
202         Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif\n ')];
203
204
205         Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(Controle_ACE < 0){')];
206         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMEP = 255 + ...
            max(Controle_ACE,-MaxVel);')];
207         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMEN = 1;')];
208         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include ...
            "Subrotinas_Controle_AtivaInterrupts_ET.h"')];
209         Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];
210         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMEP = min(Controle_ACE,MaxVel);')];
211         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMEN = 0;')];
212         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include ...
            "Subrotinas_Controle_AtivaInterrupts_EF.h"')];
213         Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
214         Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(Controle_ACD < 0){')];
215         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMDP = 255 + ...
            max(Controle_ACD,-MaxVel);')];
216         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMDN = 1;')];
217         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include ...
            "Subrotinas_Controle_AtivaInterrupts_DT.h"')];
218         Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];
219         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMDP = min(Controle_ACD,MaxVel);')];
220         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  VMDN = 0;')];
221         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include ...
            "Subrotinas_Controle_AtivaInterrupts_DF.h"')];
222         Tmp1 = [Tmp1 Custom_Files_SeqAssist('} \n')];
223         fprintf(fid,Tmp1);
224
225         fclose(fid);
226     end
227
228     %% Gera as subrotinas dos movimentos
229
230     if GenerateFiles
231         FileCell = [FileCell ; {'subrotinas','Subrotinas_Mover_Normal.h',NFiles+1}];
232         NFiles = NFiles + 1;
233         fid = fopen(FileCell{NFiles,2},'w');
```

```
234     Tmp1 = ScriptName;
235     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_ConfigVel.h"')];
236     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP, VMEP);//EP')];
237     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN, VMEN);//EN')];
238     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN, VMDN);//DN')];
239     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP, VMDP);//DP')];
240     fprintf(fid,Tmp1);
241     fclose(fid);
242
243     FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_MoverR_Direita.h',NFiles+1}];
244     NFiles = NFiles + 1;
245     fid = fopen(FileCell{NFiles,2},'w');
246     Tmp1 = ScriptName;
247     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP,        MaxVel);//EP')];
248     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN,             0);//EN')];
249     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN,             1);//DN')];
250     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP, 255 - MaxVel);//DP')];
251     fprintf(fid,Tmp1);
252     fclose(fid);
253
254     FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_MoverR_Esquerda.h',NFiles+1}];
255     NFiles = NFiles + 1;
256     fid = fopen(FileCell{NFiles,2},'w');
257     Tmp1 = ScriptName;
258     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP, 255 - MaxVel);//EP')];
259     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN,             1);//EN')];
260     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN,             0);//DN')];
261     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP,        MaxVel);//DP')];
262     fprintf(fid,Tmp1);
263     fclose(fid);
264
265     FileCell = [FileCell ; {'subrotinas','Subrotinas_MoverR_Frente.h',NFiles+1}];
266     NFiles = NFiles + 1;
267     fid = fopen(FileCell{NFiles,2},'w');
268     Tmp1 = ScriptName;
269     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP,        MaxVel);//EP')];
270     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN,             0);//EN')];
271     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN,             0);//DN')];
272     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP,        MaxVel);//DP')];
273     fprintf(fid,Tmp1);
274     fclose(fid);
```

```
275
276     FileCell = [FileCell ; {'subrotinas','Subrotinas_MoverR_Tras.h',NFiles+1}];
277     NFiles = NFiles + 1;
278     fid = fopen(FileCell{NFiles,2},'w');
279     Tmp1 = ScriptName;
280     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP, 255 - MaxVel);//EP')];
281     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN,             1);//EN')];
282     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN,             1);//DN')];
283     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP, 255 - MaxVel);//DP')];
284     fprintf(fid,Tmp1);
285     fclose(fid);
286
287     FileCell = [FileCell ; {'subrotinas','Subrotinas_Parar.h',NFiles+1}];
288     NFiles = NFiles + 1;
289     fid = fopen(FileCell{NFiles,2},'w');
290     Tmp1 = ScriptName;
291     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_MoverS_Parar.h"')];
292     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_Controle_Reset.h"')];
293     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( EP,             0);//EP')];
294     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(EN,             0);//EN')];
295     Tmp1 = [Tmp1 Custom_Files_SeqAssist('digitalWrite(DN,             0);//DN')];
296     Tmp1 = [Tmp1 Custom_Files_SeqAssist('analogWrite( DP,             0);//DP')];
297     fprintf(fid,Tmp1);
298     fclose(fid);
299
300     FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_MoverS_Direita.h',NFiles+1}];
301     NFiles = NFiles + 1;
302     fid = fopen(FileCell{NFiles,2},'w');
303     Tmp1 = ScriptName;
304     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP = 100;')];
305     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN =   0;')];
306     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN = 100;')];
307     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP =   0;')];
308     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_Controle_SetupTf_Mover_Direita.h"')];
309     fprintf(fid,Tmp1);
310     fclose(fid);
311
312     FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_MoverS_Esquerda.h',NFiles+1}];
313     NFiles = NFiles + 1;
```

```
314     fid = fopen(FileCell{NFiles,2},'w');
315     Tmp1 = ScriptName;
316     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP =   0;')];
317     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN = 100;')];
318     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN =   0;')];
319     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP = 100;')];
320     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_Controle_SetupTf_Mover_Esquerda.h"')];
321     fprintf(fid,Tmp1);
322     fclose(fid);
323
324     FileCell = [FileCell ; {'subrotinas','Subrotinas_MoverS_Frente.h',NFiles+1}];
325     NFiles = NFiles + 1;
326     fid = fopen(FileCell{NFiles,2},'w');
327     Tmp1 = ScriptName;
328     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP = 100;')];
329     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN =   0;')];
330     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN =   0;')];
331     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP = 100;')];
332     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_Controle_SetupTf_Mover_Frente.h"')];
333     fprintf(fid,Tmp1);
334     fclose(fid);
335
336     FileCell = [FileCell ; {'subrotinas','Subrotinas_MoverS_Tras.h',NFiles+1}];
337     NFiles = NFiles + 1;
338     fid = fopen(FileCell{NFiles,2},'w');
339     Tmp1 = ScriptName;
340     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP =   0;')];
341     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN = 100;')];
342     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN = 100;')];
343     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP =   0;')];
344     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_Controle_SetupTf_Mover_Tras.h"')];
345     fprintf(fid,Tmp1);
346     fclose(fid);
347
348     FileCell = [FileCell ; {'subrotinas','Subrotinas_MoverS_Parar.h',NFiles+1}];
349     NFiles = NFiles + 1;
350     fid = fopen(FileCell{NFiles,2},'w');
351     Tmp1 = ScriptName;
352     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEP =   0;')];
353     Tmp1 = [Tmp1 Custom_Files_SeqAssist('MEN =   0;')];
```

```
354         Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDN =    0;')];
355         Tmp1 = [Tmp1 Custom_Files_SeqAssist('MDP =    0;')];
356         fprintf(fid,Tmp1);
357         fclose(fid);
358
359
360     end
```

Arduino_SS_GeraPosicao

Função que gera os arquivos de posição/velocidade alvo, dependendo do controlador escolhido.

Retorne à tabela clicando nesta frase

```
1   %% Perpara o a string com o nome do script
2       ScriptName = '// Arduino_SS_GeraPosicao \n';
3
4   %% Gera a subrotina de geração de posição
5
6   if GenerateFiles
7       FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_Controle_GenerateTargetPosition.h',NFiles+1}];
8       NFiles = NFiles + 1;
9       fid = fopen(FileCell{NFiles,2},'w');
10      Tmp1 = ScriptName;
11
12
13      if PosCon
14          Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(MEP > 0){')];%Não preciso ...
                testar o MDP porque os motores SEMPRE acionam juntos
15          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if(MEN > 0){')];%Esquerdo pra trás
16          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if(MDN > 0){')];%Direito pra ...
                trás
17          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = PosE - ...
                ',Sim_M*T/FScale,';')];
18          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = PosD - ...
                ',Sim_M*T/FScale,';')];
19          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito pra frente ...
                e Esquerdo pra trás, ele tá virando
20          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = PosE - ...
```

```
            ',Sim_M*T*Sim_TurnAdj/FScale,';')];
21          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = PosD + ...
            ',Sim_M*T*Sim_TurnAdj/FScale,';')];
22          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
23          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];%Esquerdo pra frente
24          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if(MDN > 0){')];%Direito pra ...
            trás, ele tá virando
25          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = PosE + ...
            ',Sim_M*T*Sim_TurnAdj/FScale,';')];
26          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = PosD - ...
            ',Sim_M*T*Sim_TurnAdj/FScale,';')];
27          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito pra frente ...
            e Esquerdo pra frente
28          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = PosE + ...
            ',Sim_M*T/FScale,';')];
29          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = PosD + ...
            ',Sim_M*T/FScale,';')];
30          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
31          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
32          Tmp1 = [Tmp1 Custom_Files_SeqAssist('}\n')];
33      else
34          Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(MEP > 0){')];%Não preciso ...
            testar o MDP porque os motores SEMPRE acionam juntos
35          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if(MEN > 0){')];%Esquerdo pra trás
36          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if(MDN > 0){')];%Direito pra ...
            trás
37          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = ',-Sim_M/FScale,';')];
38          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = ',-Sim_M/FScale,';')];
39          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito pra frente ...
            e Esquerdo pra trás, ele tá virando
40          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE = ' ...
            ,-Sim_M*Sim_TurnAdj/FScale,';')];
41          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD =  ...
            ',Sim_M*Sim_TurnAdj/FScale,';')];
42          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
43          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];%Esquerdo pra frente
44          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if(MDN > 0){')];%Direito pra ...
            trás, ele tá virando
45          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE =  ...
            ',Sim_M*Sim_TurnAdj/FScale,';')];
46          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD = ...
            ',-Sim_M*Sim_TurnAdj/FScale,';')];
47          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];%Direito pra frente ...
```

```
            e Esquerdo pra frente
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosE =  ',Sim_M/FScale,';')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('      PosD =  ',Sim_M/FScale,';')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  PosE = 0;')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('  PosD = 0;')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('}\n')];
    end
    fprintf(fid,Tmp1);

    fclose(fid);
end
```

Arduino_SS_MLPrintFiles

Função que gera os arquivos que enviam dados pelos canais de comunicação.

Retorne à tabela clicando nesta frase

```
%% Prepara o nome

ScriptName = '// Arduino_SS_MLPrintFiles \n';

%% Gera o vetor de tempo

if GenerateFiles
    FileCell = [FileCell ;...
        {'subrotinas','Subrotinas_ML_Temporizacao.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = [ScriptName...
    'Tempo = micros();\n'...
    'Serial.print("Tempo: ");\n'...
    'Serial.println(Tempo);\n'...
    ];
    fprintf(fid,Tmp1);
    fclose(fid);
end

```

```
%% Generates the prints Arduino Proximity Sensors and engines

if GenerateFiles
    FileCell = [FileCell ; {'subrotinas','Subrotinas_ML_PrintAll.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    A = ArduinoSS_DT1RTVecSetup;
    Tmp1 = ScriptName;
    if TestMode
        for n = A
            Tmp2 = size(VarsProg{n,3},2);
            if Tmp2 > 1
                Tmp1 = [Tmp1 Custom_Files_SeqAssist(...
                'Serial.print("',VarsProg{n,2},'[',0:Tmp2-1,']: ");\n',...
                'Serial.println(',VarsProg{n,2},'[',0:Tmp2-1,']);')];
                %Tmp1 = [Tmp1 ...
                    Custom_Files_SeqAssist('Serial.println(',VarsProg{n,2}, ...
                    '[',0:Tmp2-1,']: ");')];
            else
                Tmp1 = [Tmp1 ...
                    Custom_Files_SeqAssist('Serial.print("',VarsProg{n,2},': ...
                    ");')];
                Tmp1 = [Tmp1 ...
                    Custom_Files_SeqAssist('Serial.println(',VarsProg{n,2},');')];
            end
        end
    end
    fprintf(fid,Tmp1);
    fclose(fid);
end

%% Same, but sends it via bluetooth
if GenerateFiles
    FileCell = [FileCell ; {'subrotinas','Subrotinas_ML_PrintAllBL.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    A = ArduinoSS_DT1RTVecSetup;
    Tmp1 = ScriptName;
    if TestMode
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('String Str;')];
        Tmp1 = [Tmp1 Custom_Files_SeqAssist('Str = ...
            String(',VarsProg{A(1),2},') + " ";')];
        for n = A(min([2 end]):end)
```

```
58              Tmp2 = size(VarsProg{n,3},2)-1;
59              if Tmp2 ≥ 1
60                  Tmp1 = [Tmp1 Custom_Files_SeqAssist('Str = Str + ...
                        String(',VarsProg{n,2},'[',0:Tmp2,']) + " ";')];
61              else
62                  Tmp1 = [Tmp1 Custom_Files_SeqAssist('Str = Str + ...
                        String(',VarsProg{n,2},') + " ";')];
63              end
64          end
65          Tmp1 = [Tmp1(1:end-10) ';\n'];
66          Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (Serial.availableForWrite()){')];
67          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Serial.println(Str);')];
68          Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
69          Tmp1 = [Tmp1 Custom_Files_SeqAssist('bluetooth.println(Str);')];
70      end
71      fprintf(fid,Tmp1);
72      fclose(fid);
73  end
74
75  %% Print every variable in the arduino for debugging the program flow
76
77  if GenerateFiles
78      FileCell = [FileCell ; {'subrotinas','Subrotinas_DumpVariables.h',NFiles+1}];
79      NFiles = NFiles + 1;
80      fid = fopen(FileCell{NFiles,2},'w');
81      if (DumpVars && TestMode)
82          Tmp1 = ScriptName;
83          Tmp1 = [Tmp1 Custom_Files_SeqAssist('String ML_Str;')];
84          for n = 1:length(VarsProg)
85              A = length(VarsProg{n,3});
86              if A > 1
87                  %Tmp1 = [Tmp1 ...
                        Custom_Files_SeqAssist('Serial.print("',VarsProg{n,2},'[',0:(A-1),']: ...
                        "); \nSerial.println(',VarsProg{n,2},'[',0:(A-1),']);')];
88                  Tmp1 = [Tmp1 Custom_Files_SeqAssist('Str = Str + ...
                        "',VarsProg{n,2},'[',0:(A-1),']: " + ...
                        String(',VarsProg{n,2},'[',0:(A-1),']) + ";\n"')];
89              else
90                  %Tmp1 = [Tmp1 ...
                        Custom_Files_SeqAssist('Serial.print("',VarsProg{n,2},': ...
                        "); \nSerial.println(',VarsProg{n,2},');')];
91                  Tmp1 = [Tmp1 Custom_Files_SeqAssist('Str = Str + ...
                        "',VarsProg{n,2},': " + String(',VarsProg{n,2},') + ";\n"')];
```

```
92                  end
93              end
94              Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Str)')];
95              fprintf(fid,Tmp1);
96          else
97              Tmp1 = [Tmp1 Custom_Files_SeqAssist('// No Dump Activated')];
98          end
99
100         fclose(fid);
101     end
```

Arduino_SS_GenerateData

Geram arquivos intermediários de gerenciamento no Arduino.

Retorne à tabela clicando nesta frase

```
1   %% Prepara o nome
2
3   ScriptName = '// Arduino_SS_GenerateData \n';
4
5   %% Gera o MainProg
6   %  Ele equivale ao 'main', em que funciona tudo junto
7
8   if GenerateFiles
9       FileCell = [FileCell ; {'data','MainProg.h',NFiles+1}];
10      NFiles = NFiles + 1;
11      fid = fopen(FileCell{NFiles,2},'w');
12      Tmp1 = ScriptName;
13      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Global.h" \n\n')];
14      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void setup() {')];
15      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Setup.h"')];
16      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
17      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void loop() {')];
18      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "modules/LoopVariables.h"')];
19      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Bluetooth_LimpaOverflow.h"')];
20      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if ShowMarks')];
21      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Mark1");')];
```

```
22      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
23      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
24      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Bluetooth_Handler.h"')];
25      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if ShowMarks //nada no Bluetooth')];
26      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Mark11");')];
27      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
28      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
29      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "modules/Sensors_Geral.h"')];
30      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if ShowMarks //nada no Bluetooth')];
31      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Mark13");')];
32      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
33      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
34      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Bluetooth_Decisao.h"')];
35      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if ShowMarks')];
36      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Mark2");')];
37      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
38      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
39      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Subrotinas/Subrotinas_Mover_Normal.h"')];
40      if DumpVars
41          Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
                "modules/Subrotinas/Subrotinas_ML_PrintAll.h"')];
42      end
43      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
44      fprintf(fid,Tmp1);
45      fclose(fid);
46  end
47
48  %% Gera o programa de teste dos motores
49
50  if GenerateFiles
51      FileCell = [FileCell ; {'data','ML_Motores.h',NFiles+1}];
52      NFiles = NFiles + 1;
53      fid = fopen(FileCell{NFiles,2},'w');
54      Tmp1 = ScriptName;
55      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Global.h" \n\n')];
56      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void setup() {')];
57      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Setup.h"')];
58      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println(EngineTest);')];
```

```
59      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
60      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void loop() {')];
61      if DumpVars
62          Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println("Running");')];
63      end
64      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "modules/LoopVariables.h"')];
65      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if (EngineTest == 1)')];
66      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Subrotinas/SubrotinaTeste_Motores.h"')];
67      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
68      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if (EngineTest == 2 || EngineTest == ...
            3)')];
69      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Subrotinas/SubrotinaTeste_Controle.h"')];
70      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
71      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if (EngineTest == 4)')];
72      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/Subrotinas/SubrotinaTeste_Chao.h"')];
73      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
74      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
75      fprintf(fid,Tmp1);
76      fclose(fid);
77  end
78
79
80  %% Gera o programa de teste dos sensores
81
82
83  if GenerateFiles
84      FileCell = [FileCell ; {'data','ML_Sensors.h',NFiles+1}];
85      NFiles = NFiles + 1;
86      fid = fopen(FileCell{NFiles,2},'w');
87      Tmp1 = ScriptName;
88      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Global.h"')];
89      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void setup() {')];
90      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "modules/SetupGeral_Setup.h"')];
91      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
92      Tmp1 = [Tmp1 Custom_Files_SeqAssist('void loop() {')];
93      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "modules/LoopVariables.h"')];
94      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "modules/Sensors_Geral.h"')];
95      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
```

```
96      fprintf(fid,Tmp1);
97      fclose(fid);
98  end
```

Arduino_SS_GeraBluetooth

Gera os arquivos para a interação do Arduino com o módulo bluetooth.

Retorne à tabela clicando nesta frase

```
1   %% Nome
2
3   ScriptName = '// Arduino_SS_GeraBluetooth \n';
4
5
6   %% Bluetooth
7   if GenerateFiles
8       FileCell = [FileCell ; {'modules','Bluetooth_LimpaOverflow.h',NFiles+1}];
9       NFiles = NFiles + 1;
10      fid = fopen(FileCell{NFiles,2},'w');
11      Tmp1 = ScriptName;
12      Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(bluetooth.overflow()){')];
13      Tmp1 = [Tmp1 Custom_Files_SeqAssist('  while(bluetooth.available() > 0){')];
14      Tmp1 = [Tmp1 Custom_Files_SeqAssist('    C = bluetooth.read();')];
15      Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
16      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
17      fprintf(fid,Tmp1);
18      fclose(fid);% Here
19  end
20
21  if GenerateFiles
22      FileCell = [FileCell ; {'modules','Bluetooth_Handler.h',NFiles+1}];
23      NFiles = NFiles + 1;
24      fid = fopen(FileCell{NFiles,2},'w');
25      Tmp1 = ScriptName;
26      Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(!bluetooth.available()){')];
27      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode //Nada no Bluetooth')];
28      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Waiting");')];
29      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
30      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
31      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode //Nada no Bluetooth')];
```

```
32      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Froze 01 (");')];
33      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(bluetooth.overflow());')];
34      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(")");')];
35      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
36      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
37      Tmp1 = [Tmp1 Custom_Files_SeqAssist(';')];
38      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];
39      Tmp1 = [Tmp1 Custom_Files_SeqAssist('C = bluetooth.read();')];
40      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode //Alguma coisa no ...
            Bluetooth')];
41      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Froze 02");')];
42      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
43      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
44      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode //Diz o caractere lido')];
45      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Read:");')];
46      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(C);')];
47      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("\t");')];
48      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
49      Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(C == ''+''){//195 = + em ASCII')];
50      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(25);// Aguarda o resto da ...
            mensagem chegar')];
51      Tmp1 = [Tmp1 Custom_Files_SeqAssist('C = bluetooth.read();')];
52      Tmp1 = [Tmp1 Custom_Files_SeqAssist('cont = (C == 68)*14 + (C == ...
            67)*42;//68 = D e 67 = C em ASCII')];
53      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode //Diz o que foi ...
            ignorado no buffer')];
54      Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Usado pra pular as diretivas do ...
            módulo')];
55      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Skipping(");')];
56      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(cont);')];
57      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("): <");')];
58      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
59      Tmp1 = [Tmp1 Custom_Files_SeqAssist('while(cont > 0){')];
60      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
61      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(C);')];
62      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
63      Tmp1 = [Tmp1 Custom_Files_SeqAssist('C = bluetooth.read();')];
64      Tmp1 = [Tmp1 Custom_Files_SeqAssist('cont = cont - 1;')];
65      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
66      Tmp1 = [Tmp1 Custom_Files_SeqAssist('C = 80;//80 = P em ASCII')];
67      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
68      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(">");')];
69      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println("\t \t");')];
```

```
70      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("C <- P");')];
71      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Setup (+) ");')];
72      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Operation);')];
73      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("\t \t");')];
74      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
75      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
76      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Case = Operation + (C == 83)*2;// 83 ...
            = S em ASCII')];
77      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
78      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
79      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Case (");')];
80      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Case);')];
81      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("): (");')];
82      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(C);')];
83      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(") ->");')];
84      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
85      Tmp1 = [Tmp1 Custom_Files_SeqAssist('switch(Case){')];
86      Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 2:')];
87      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Operation = 1;')];
88      Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 0:')];
89      Tmp1 = [Tmp1 Custom_Files_SeqAssist(' C = 80;//80 = P em ASCII')];
90      Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 1:')];
91      Tmp1 = [Tmp1 Custom_Files_SeqAssist('default:')];
92      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Prev = C;')];
93      Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
94      Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 3:')];
95      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Operation = 0;')];
96      Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
97      Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
98      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
99      Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("(");')];
100     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(C);')];
101     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(",");')];
102     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Operation);')];
103     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(")");')];
104     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
105     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
106     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
107     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
108     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Sensor = ");')];
109     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Sensor);')];
110     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Serial.println();')];
111     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
```

```
112     Tmp1 = [Tmp1 Custom_Files_SeqAssist('switch (C){')];
113     Tmp1 = [Tmp1 Custom_Files_SeqAssist('default:')];
114     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 83://83 = S em ASCII')];
115     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 80://80 = P em ASCII')];
116     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE =  8;')];
117     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Counter_DD = 0;')];
118     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Counter_ED = 0;')];
119     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pos_D = 0;')];
120     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pos_E = 0;')];
121     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
122     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 70://70 = F em ASCII')];
123     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE =  4;')];
124     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = 0;')];
125     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
126     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 68://68 = D em ASCII')];
127     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE =  2;')];
128     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = 0;')];
129     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
130     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 69://69 = E em ASCII')];
131     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE =  1;')];
132     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = 0;')];
133     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
134     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 84://84 = T em ASCII')];
135     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE = 16;')];
136     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = 0;')];
137     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
138     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
139     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
140     fprintf(fid,Tmp1);
141     fclose(fid);
142 end
143
144
145 if GenerateFiles
146     FileCell = [FileCell ; {'modules','Bluetooth_Decisao.h',NFiles+1}];
147     NFiles = NFiles + 1;
148     fid = fopen(FileCell{NFiles,2},'w');
149     Tmp1 = ScriptName;
150     Tmp1 = [Tmp1 Custom_Files_SeqAssist('switch(FT_TPFDE){')];
151     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 1://E')];
152     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 33://E + T')];
153     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 65://E + F')];
154     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 97://E + T/F')];
```

```
155     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pra Esquerda')];
156     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "subrotinas/Subrotinas_MoverS_Esquerda.h"')];
157     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
158     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Turning Left");')];
159     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
160     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
161     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
162     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 2://D')];
163     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 34://D + T')];
164     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 66://D + F')];
165     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 98://D + T/F')];
166     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pra Direita')];
167     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "subrotinas/Subrotinas_MoverS_Direita.h"')];
168     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
169     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Turning Right");')];
170     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
171     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
172     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
173     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 4://F')];
174     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 36://F + T')];
175     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pra Frente')];
176     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode_VelEncs')];
177     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Vel (D)(E):");')];
178     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_DD);')];
179     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("/");')];
180     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_ED);')];
181     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("->");')];
182     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_DD - Enc_ED);')];
183     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
184     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
185     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "subrotinas/Subrotinas_MoverS_Frente.h"')];
186     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
187     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Moving Foward");')];
188     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
189     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
190     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
191     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 48://T e T')];
192     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 112://T e T+F')];
193     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 68://F e F')];
194     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 100://F e T+F')];
```

```
195     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 0://S')];
196     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 8://P')];
197     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 40://P e T')];
198     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 72://P e F')];
199     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 104://P e T+F')];
200     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Fica Parado')];
201     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Counter_DD = 0;')];
202     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Counter_ED = 0;')];
203     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "subrotinas/Subrotinas_MoverS_Parar.h"')];
204     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
205     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Stopping");')];
206     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
207     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
208     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
209     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 16://T')];
210     Tmp1 = [Tmp1 Custom_Files_SeqAssist('case 80://T + F')];
211     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pra Trás')];
212     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode_VelEncs')];
213     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Vel (D)(E):");')];
214     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_DD);')];
215     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("/");')];
216     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_ED);')];
217     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("->");')];
218     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Enc_DD - Enc_ED);')];
219     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
220     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
221     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "subrotinas/Subrotinas_MoverS_Tras.h"')];
222     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#if DebugMode')];
223     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Moving Reverse");')];
224     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
225     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#endif')];
226     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
227     Tmp1 = [Tmp1 Custom_Files_SeqAssist('default:')];
228     Tmp1 = [Tmp1 Custom_Files_SeqAssist('break;')];
229     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
230     Tmp1 = [Tmp1 Custom_Files_SeqAssist('FT_TPFDE =  (FT_TPFDE & 31);')];
231     fprintf(fid,Tmp1);
232     fclose(fid);
233
234 end
```

Arduino_SS_GeraSensores

Gera os arquivos que lêem dados dos 2 sensores de posição.

Retorne à tabela clicando nesta frase

```
%% Prepara o nome

ScriptName = '// Arduino_SS_GeraSensores.h \n';

%% Gera os arquivos relacionados aos sensores

if GenerateFiles

    FileCell = [FileCell ; {'modules','Sensors_Medicao.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = ScriptName;
    if UseSensors
        Tmp1 = [Tmp1...
        '@'...
        '// Minha medição de distância@'...
        '// Manda um pulso no Trig Compartilhado@'...
        'if (millis() - SensorTimer > 50){@'...
          'SensorTimer = millis();@'...
          'digitalWrite(TrigT,LOW);@'...
          'digitalWrite(TrigT,HIGH);@'...
          'delayMicroseconds(15);@'...
          'digitalWrite(TrigT,LOW);@'...
          'if (SensorSwitch){@'...
            'SensorSwitch = 0;@'...
            'DistF = pulseInLong(EchoF,HIGH,40000)*(Enc_SegToCm);@'...
         '}else{@'...
            'SensorSwitch = 1;@'...
            'DistT = pulseInLong(EchoT,HIGH,40000)*(Enc_SegToCm);@'...
          '}@'...
        '}@'...
        ];
    end
    Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
    fprintf(fid,Tmp1);
    fclose(fid);
end
```

```
38
39
40 if GenerateFiles
41     FileCell = [FileCell ; {'modules','Sensors_Geral.h',NFiles+1}];
42     NFiles = NFiles + 1;
43     fid = fopen(FileCell{NFiles,2},'w');
44     Tmp1 = ScriptName;
45     if UseSensors
46         Tmp1 = [Tmp1...
47         '@#include "Sensors_Medicao.h"@'...
48         '// Gera a detecção@'...
49         'if(DistT == 0){@'...
50         '  DistT = 40;@'...
51         '}@'...
52         'if(DistF == 0){@'...
53         '  DistF = 40;@'...
54         '}@'...
55         'Sensor = (Dist > DistT) | ((Dist > DistF) << 1);@'...
56         'FT_TPFDE = ((Sensor << 5) | FT_TPFDE);@'...
57         ];
58     end
59     Tmp1 = Custom_ReplaceStr(Tmp1,'@','\n');
60     fprintf(fid,Tmp1);
61     fclose(fid);
62 end
63
64 if GenerateFiles
65     FileCell = [FileCell ; {'modules','ML_WaitInput_SensorTest.h',NFiles+1}];
66     %FileCell = [FileCell ; {'modules','ML_SensorTest_Setup',NFiles+1}];
67     NFiles = NFiles + 1;
68     fid = fopen(FileCell{NFiles,2},'w');
69     Tmp1 = ScriptName;
70     if UseSensors
71         Tmp1 = [Tmp1 Custom_Files_SeqAssist('if (Read == 250){')];
72         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Read = 0;')];
73         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  float Temp = -1;')];
74         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  while(Temp < 0 || Temp > ...
               1000){')];
75         Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Serial.print("Waiting on ...
               FMMNP_D Value; 250");')];
76         Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Serial.println();')];
77         Tmp1 = [Tmp1 Custom_Files_SeqAssist('    while(Serial.available() == ...
               0){')];
```

```
78          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      ;')];
79          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
80          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Temp = Serial.parseInt();')];
81          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if (Temp ≤ 1000 || Temp ≥ ...
                0){')];
82          Tmp1 = [Tmp1 Custom_Files_SeqAssist('      FMMNP_D = Temp/1000;')];
83          Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
84          Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
85          Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
86      end
87      fprintf(fid,Tmp1);
88      fclose(fid);
89  end
```

Arduino_SS_ConfigGeral

Gera arquivos de configurações gerais que não se encaixam bem em outras categorias.

Retorne à tabela clicando nesta frase

```
1  %% Nome
2
3  ScriptName = '// Arduino_SS_ConfigGeral \n';
4
5
6  %% Variáveis Gerais
7
8  StopDist = 20;%Centímetros de distância pra parar o carrinho
9  aux1 = PosData_Identify_F;
10 aux2 = PosData_Identify_T;
11 A = abs([aux1{2,2};aux1{6,2};aux2{2,2};aux2{6,2}]);
12 clear aux1
13 clear aux2
14 if PosCon
15      Aceleracao = T*Sim_M/FScale;
16      Frenagem = T*Sim_M/FScale;
17 else
18     %Aceleracao = (MaxVel*Sim_M)/max(A);
19      Aceleracao = Sim_M/FScale;
20      Frenagem = Aceleracao;
21 end
```

```
22
23 %% Arquivos
24
25 if GenerateFiles
26     FileCell = [FileCell ; {'modules','Pinagens.h',NFiles+1}];
27     NFiles = NFiles + 1;
28     fid = fopen(FileCell{NFiles,2},'w');
29     Tmp1 = ScriptName;
30     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Pinos onde os fios estão ...
           conectados')];
31     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//RefreshRate')];
32     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Periodo 1')];
33     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Bluetooth')];
34     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define P_rx 4')];
35     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define P_tx 5')];
36     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//MEF')];
37     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define EP  9')];
38     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define EN 11')];
39     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DN 10')];
40     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DP  6')];
41     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Ultrassom')];
42     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define TrigF 7')];
43     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define EchoF A1')];
44     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define TrigT 7')];
45     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define EchoT A0')];
46     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Encoders')];
47     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Enc_DD 2')];
48     %Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Enc_DA A0')];
49     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Enc_ED 3')];
50     %Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Enc_EA A1')];
51     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Tipo de interrupção')];
52     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//RISING, CHANGE ou FALLING')];
53     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define InterruptType RISING')];
54     fprintf(fid,Tmp1);
55     fclose(fid);
56 end
57
58 if GenerateFiles
59     FileCell = [FileCell ; {'modules','LoopVariables.h',NFiles+1}];
60     NFiles = NFiles + 1;
61     fid = fopen(FileCell{NFiles,2},'w');
62     Tmp1 = ScriptName;
63     Tmp1 = [Tmp1 Custom_Files_SeqAssist('char Prev = 80;// 80 = P em ASCII')];
```

```
64      Tmp1 = [Tmp1 Custom_Files_SeqAssist('float temp;')];
65      fprintf(fid,Tmp1);
66      fclose(fid);
67  end
68
69  if GenerateFiles
70      FileCell = [FileCell ; {'modules','Defines.h',NFiles+1}];
71      NFiles = NFiles + 1;
72      fid = fopen(FileCell{NFiles,2},'w');
73      Tmp1 = ScriptName;
74      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define ShowMarks 0')];
75      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define TestMode 0')];
76      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DebugMode 0')];
77      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DebugMode_Dist 0')];
78      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DebugMode_Vel 0')];
79      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DebugMode_Encs 0')];
80      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define DebugMode_VelEncs 0')];
81      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define IgnoreCalibration 1')];
82      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define UseEngines 1')];
83      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Responses 1')];
84      fprintf(fid,Tmp1);
85      fclose(fid);
86  end
87
88  if GenerateFiles
89      FileCell = [FileCell ; {'modules','Configuracoes.h',NFiles+1}];
90      NFiles = NFiles + 1;
91      fid = fopen(FileCell{NFiles,2},'w');
92      Tmp1 = ScriptName;
93      Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Configurações dos valores')];
94      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define TrigSize 10')];
95      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Dist ',StopDist,' //Distância ...
            Pra parar o carro')];
96      Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Motores')];
97      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Aceleracao ',Aceleracao)];
98      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Frenagem ',Aceleracao)];
99      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define MaxVel ',MaxVel)];
100     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define ModVelNormal 1')];
101     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define ModVelGiro 0.25')];
102     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define V_Start_E_Start 50')];
103     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define V_Start_D_Start 50')];
104     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define CalibrationMargin 10')];
105     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Encoders')];
```

```
106     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define NPulses_45 16 // Número de ...
            pulsos pra 45 graus')];
107     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define VelToFuros 20 // ')];
108     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define VelFurosLim 100 // Mede a ...
            velocidade após n furos contados')];
109     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define FMMN 10// Número de medições ...
            no filtro de média')];
110     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define FMMNP_V 0.5// Coeficiente de ...
            peso dos valores antigos pra velocidade')];
111     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define FMMNP_D 0.9// Coeficiente de ...
            peso dos valores antigos pra distância')];
112     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Histerese 4 // Folga pro ...
            controle de velocidade')];
113     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define FimDeContagem 1000')];
114     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Sensores')];
115     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define WaitUSensor 12500')];
116     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#define Enc_SegToCm 0.017')];
117     fprintf(fid,Tmp1);
118     fclose(fid);
119 end
120
121
122 if GenerateFiles
123     FileCell = [FileCell ; ...
            {'subrotinas','Subrotinas_ML_WaitSerialInput.h',NFiles+1}];
124     NFiles = NFiles + 1;
125     fid = fopen(FileCell{NFiles,2},'w');
126     Tmp1 = ScriptName;
127     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
128     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Waiting: input");')];
129     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
130     Tmp1 = [Tmp1 Custom_Files_SeqAssist('while (Serial.available() == 0){')];
131     Tmp1 = [Tmp1 Custom_Files_SeqAssist(' #include "Subrotinas_Parar.h";')];
132     Tmp1 = [Tmp1 Custom_Files_SeqAssist(' #include ...
            "Subrotinas_MoverS_Parar.h";')];
133     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
134     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print("Received:");')];
135     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = Serial.parseInt();')];
136     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Section = Section - 48*(Section > 48 ...
            && Section < 57);')];
137     Tmp1 = [Tmp1 Custom_Files_SeqAssist('while (Serial.available() != 0){')];
138     Tmp1 = [Tmp1 Custom_Files_SeqAssist('char temp = Serial.read();')];
139     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
```

```
140     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.print(Section);')];
141     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Serial.println();')];
142     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//delay(2000);')];
143     fprintf(fid,Tmp1);
144     fclose(fid);
145 end
```

Arduino_SS_GeraTestes

Retorne à tabela clicando nesta frase

```
1  %% Nome
2
3
4       ScriptName = '// Arduino_SS_GeraTestes \n';
5
6
7  %% Subrotinas Teste - Motores 1
8
9
10 if GenerateFiles
11      FileCell = [FileCell ; {'subrotinas','SubrotinaTeste_Motores.h',NFiles+1}];
12      NFiles = NFiles + 1;
13      fid = fopen(FileCell{NFiles,2},'w');
14      Tmp1 = ScriptName;
15      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(10000);')];
16      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_MoverR_Frente.h"')];
17      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
18      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Parar.h"')];
19      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
20      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_MoverR_Tras.h"')];
21      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
22      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Parar.h"')];
23      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
24      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
            "Subrotinas_MoverR_Direita.h"')];
25      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(1000);')];
26      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Parar.h"')];
27      Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
28      Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include ...
```

```
        "Subrotinas_MoverR_Esquerda.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(1000);')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Parar.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(2000);')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Parar.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('delay(200000);')];
    fprintf(fid,Tmp1);
    fclose(fid);
end

%% Subrotinas Teste - Motores 2
%% Subrotinas Teste - Motores 3


if GenerateFiles
    FileCell = [FileCell ; {'subrotinas','SubrotinaTeste_Controle.h',NFiles+1}];
    NFiles = NFiles + 1;
    fid = fopen(FileCell{NFiles,2},'w');
    Tmp1 = ScriptName;
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('Tempo = micros();')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('if(bluetooth.available()){')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  switch (bluetooth.parseInt()){')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('    case 1:')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      #include ...
        "Subrotinas_MoverS_Frente.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      break;')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('    case 2:')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      #include ...
        "Subrotinas_MoverS_Tras.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      break;')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('    case 3:')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      #include ...
        "Subrotinas_MoverS_Parar.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('      break;')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  bluetooth.read();')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  bluetooth.read();')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include ...
        "Subrotinas_ML_PrintAllBL.h"')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
    Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Mover_Normal.h"')];
    fprintf(fid,Tmp1);
    fclose(fid);
```

```
67 end
68
69 %% Subrotinas Teste - Motores 4
70
71
72 if GenerateFiles
73     FileCell = [FileCell ; {'subrotinas','SubrotinaTeste_Chao.h',NFiles+1}];
74     NFiles = NFiles + 1;
75     fid = fopen(FileCell{NFiles,2},'w');
76     Tmp1 = ScriptName;
77     Tmp1 = [Tmp1 Custom_Files_SeqAssist('if ((Tempo - Timer > ',6*10^6,' ) || ...
           Section == 0){')];
78     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//if ((Read > ',500,' ) || Section == ...
           0){')];
79     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  #include "Subrotinas_Parar.h"')];
80     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if (Section == 0){')];
81     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Read = 0;')];
82     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    PosD = 0;')];
83     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    PosE = 0;')];
84     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    //V_Start_DF = MaxVel;')];
85     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    //V_Start_EF = MaxVel;')];
86     if RunMain == -4
87         Tmp1 = [Tmp1 Custom_Files_SeqAssist('    Section = 1; ')];
88     end
89     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];
90     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Section = Section + 1;')];
91     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  delay(5000);')];
92     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
93     if RunMain == -4
94         Tmp1 = [Tmp1 Custom_Files_SeqAssist('  delay(10000);')];
95     end
96     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Timer = micros();')];
97     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}else{')];
98     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  Read = Read + 1;')];
99     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
100    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if (Section == 1){')];
101    Tmp1 = [Tmp1 Custom_Files_SeqAssist('    #include ...
           "Subrotinas_MoverS_Frente.h"')];
102    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];
103    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  if (Section == 2){')];
104    Tmp1 = [Tmp1 Custom_Files_SeqAssist('    #include ...
           "Subrotinas_MoverS_Tras.h"')];
105    Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }else{')];
```

```
106     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    if (Section == 3){')];
107     Tmp1 = [Tmp1 Custom_Files_SeqAssist('      #include ...
            "Subrotinas_MoverS_Esquerda.h"')];
108     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }else{')];
109     Tmp1 = [Tmp1 Custom_Files_SeqAssist('      if (Section == 4){')];
110     Tmp1 = [Tmp1 Custom_Files_SeqAssist('        #include ...
            "Subrotinas_MoverS_Direita.h"')];
111     Tmp1 = [Tmp1 Custom_Files_SeqAssist('      }else{')];
112     Tmp1 = [Tmp1 Custom_Files_SeqAssist('        if (Section == 5){')];
113     Tmp1 = [Tmp1 Custom_Files_SeqAssist('          #include ...
            "Subrotinas_MoverS_Parar.h"')];
114     Tmp1 = [Tmp1 Custom_Files_SeqAssist('          Section = 0;')];
115     Tmp1 = [Tmp1 Custom_Files_SeqAssist('        }else{')];
116     Tmp1 = [Tmp1 Custom_Files_SeqAssist('          if (Read > 500){')];
117     Tmp1 = [Tmp1 Custom_Files_SeqAssist('            Section = 0;')];
118     Tmp1 = [Tmp1 Custom_Files_SeqAssist('            #include ...
            "Subrotinas_MoverS_Parar.h"')];
119     Tmp1 = [Tmp1 Custom_Files_SeqAssist('          }else{')];
120     Tmp1 = [Tmp1 Custom_Files_SeqAssist('            if (Section == 6){')];
121     Tmp1 = [Tmp1 Custom_Files_SeqAssist('              #include ...
            "Subrotinas_MoverS_Frente.h"')];
122     Tmp1 = [Tmp1 Custom_Files_SeqAssist('            }else{')];
123     Tmp1 = [Tmp1 Custom_Files_SeqAssist('              if (Section == 7){')];
124     Tmp1 = [Tmp1 Custom_Files_SeqAssist('                #include ...
            "Subrotinas_MoverS_Tras.h"')];
125     Tmp1 = [Tmp1 Custom_Files_SeqAssist('              }')];
126     Tmp1 = [Tmp1 Custom_Files_SeqAssist('            }')];
127     Tmp1 = [Tmp1 Custom_Files_SeqAssist('          }')];
128     Tmp1 = [Tmp1 Custom_Files_SeqAssist('        }')];
129     Tmp1 = [Tmp1 Custom_Files_SeqAssist('      }')];
130     Tmp1 = [Tmp1 Custom_Files_SeqAssist('    }')];
131     Tmp1 = [Tmp1 Custom_Files_SeqAssist('  }')];
132     Tmp1 = [Tmp1 Custom_Files_SeqAssist('}')];
133     Tmp1 = [Tmp1 Custom_Files_SeqAssist('#include "Subrotinas_Mover_Normal.h"')];
134     Tmp1 = [Tmp1 Custom_Files_SeqAssist('Tempo = micros();')];
135     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Serial.print("Time: ");')];
136     Tmp1 = [Tmp1 Custom_Files_SeqAssist('//Serial.println(Tempo);')];
137     fprintf(fid,Tmp1);
138     fclose(fid);
139 end
```

Arduino_SS_MoveFiles

Os arquivos estão separados em pastas para organização. Esta função move os arquivos para as respectivas pastas para permitir o funcionamento.

Retorne à tabela clicando nesta frase

```
%% Move os arquivos para as respectivas pastas
for n = 1:size(FileCell,1)
    switch MyStringComparator(FileCell{n,1},...
            {'subrotinas';'modules';'data'})
        case 1
            TgtFolder = [Folders{1},'/',Folders{2},...
                '/',Folders{3},'/',Folders{4}];
        case 2
            TgtFolder = [Folders{1},'/',Folders{2},...
                '/',Folders{3}];
        case 3
            TgtFolder = [Folders{1},'/',Folders{2}];
        case 0
            TgtFolder = Folders{1};
    end
    movefile(FileCell{n,2},TgtFolder);
end
```

ArduinoSS_DT1

Função utilizada para a leitura de dados pelo canal serial diante de comandos do celular.

Retorne à tabela clicando nesta frase

```
%% Script de coleta de dados

%% Configurações
ParseVars = {};
Tmp1 = zeros(10^5,1);
A = ArduinoSS_DT1RTVecSetup;%Variáveis do VarsProg que o parser vai atrás

for n = 1:size(A,2)
    ParseVars = [ParseVars ; {VarsProg{A(n),2},Tmp1,n,0,VarsProg{A(n),2}(1), ...
        A(n)}];
```

```
10 end
11 ParseVars = [ParseVars ; {'Waiting',Tmp1,ParseVars{end,3}+1,0,'W', -1}];
12
13 % Vetor dos plots em tempo real
14 RT_SizeTracker = {};
15 for n = 1:ParseVars{end,3}
16     RT_SizeTracker = [RT_SizeTracker ; {ParseVars{n,1},0,ParseVars{n,3}}];
17 end
18
19 AskInput = 'Type value 5 or 6 for the Ardunio, 0 for exit \n';
20 %% Reinicia a Conexão
21
22 delete(instrfind);
23 S = serial('COM7','BaudRate',S_BaudRate);
24 fopen(S);
25 if Auto == 1
26     BL = bluetooth('Aurora',B_BaudRate);
27 end
28
29 %% Ler Dados
30 Case = 0;
31 Val = 0;
32 %Runs = 2;
33 Run = 0;
34 Step = 1;
35 Step_Vec = [4 8 16 8 1 2 -1];
36 OverflowDetector = 0;
37 if 1 %Runs == 0
38     while x ≥ 0
39         if S.BytesAvailable > 0
40             OverflowDetector = max([S.BytesAvailable OverflowDetector]);
41             Info = fscanf(S);
42             [Case , Val] = ArduinoSS_Parser_A(Info,ParseVars);
43             %{
44             if (Auto && (Case > Step*50))
45                 if Step* > 0
46                     if Step_Vec(Step) > 0
47                         s
48                     end
49                 end
50             end
51             %}
52             if (Case ≥ ParseVars{1,3} && Case ≤ ParseVars{end,3})
```

```
53                  ParseVars{Case,4} = ParseVars{Case,4} + 1;
54                  ParseVars{Case,2}(ParseVars{Case,4}) = Val;
55                  if RealTime
56                      RT_SizeTracker{Case,2} = RT_SizeTracker{Case,2} + 1;
57                  end
58                  if Case == ParseVars{end,3}
59                      if RealTime
60                          run Arduino_SS_DT1_RTSetup
61                      end
62                      if x == 0
63                          x = -1;
64                      else
65                          fprintf(S,int2str([x , ' 0']));
66                          x = 0;
67                      end
68                  end
69                  if OverflowDetector == 512 && RunMain == 1
70                      x = -1; %Encerra se der overflow no canal serial
71                  end
72                  if RealTime
73                      RT_SizeTracker = ...
                            Arduino_SS_RTPlot(ParseVars,RT_SizeTracker,PlotsVec);
74                  end
75              end
76          end
77      end
78      if RealTime
79          % Põe títulos e legendas nos plots
80          for n = 1:size(PlotsVec,1)
81              figure(n);
82              title(['Tempo Real: ' CName]);
83              legend({ParseVars{PlotsVec{n}(2:end),1}},'LOCATION','SouthEast');
84              grid;
85          end
86          % Refaz esses 2 gráficos no subplot
87          Tmp1 = size(PlotsVec,1);
88          figure('units','normalized','outerposition',[0.01 0.01 0.95 0.95])
89          for n = 1:size(PlotsVec,1)
90              subplot(1,size(PlotsVec,1),n);
91              Tmp2 = Conc(ones(1,size(PlotsVec{n}(2:end),2)),...
92                  PlotsVec{n}(2:end));
93              Tmp3 = min([ParseVars{Tmp2,4}]);
94              Data = ParseVars;
```

```
95              Tmp4 = sort([PlotsVec{:}]);
96              Tmp4 = Tmp4(Tmp1:end);
97              for k = Tmp4
98                  Data{k,2} = Data{k,2}(1:Tmp3);
99              end
100             Data{1,2} = (Data{1,2} - min(Data{1,2}))./10^6;
101             plot(Data{Tmp2,2});
102             legend(ParseVars{Tmp2(2:2:end),1},'LOCATION','SouthWest');
103             title(Plotnames{n});
104             xlabel('Tempo (Segundos)')
105         end
106         subplot(1,Tmp1,1);
107         if PosCon
108             ylabel('Furos do Encoder');
109         else
110             ylabel('Furos por Segundo');
111         end
112         suptitle(['Tempo Real: ' CName]);
113         set(gcf,'PaperPositionMode','auto');
114         saveas(gcf,['RT' FName]);%Salva a que vai pro Overleaf
115         movefile(['RT' FName(1:end-4) '.png'],'POverleaf');
116         saveas(gcf,['RT' FName]);%Salva outra de Backup
117     end
118 end
119
120
121
122 A = min([ParseVars{[1 2 3 4 5],4}]);
123
124 %VelD          = ParseVars{ 2,2}(1:A);
125 %VelE          = ParseVars{ 3,2}(1:A);
126 Timer          = ParseVars{ 1,2}(1:A);
127 PosD           = ParseVars{ 2,2}(1:A);
128 PosE           = ParseVars{ 3,2}(1:A);
129 EncDD          = ParseVars{ 4,2}(1:A);
130 EncED          = ParseVars{ 5,2}(1:A);
131 %Erro_D        = ParseVars{20,2}(1:A);
132 %Erro_E        = ParseVars{21,2}(1:A);
133 %Controle_ACD = ParseVars{22,2}(1:A);
134 %Controle_ACE = ParseVars{23,2}(1:A);
135 if RunMain == -2
136     Erro_D        = zeros(A,1);
137     Erro_E        = zeros(A,1);
```

```
138         Controle_ACD = zeros(A,1);
139         Controle_ACE = zeros(A,1);
140     end
141
142     Timer = Timer - min(Timer);
143
144     PosData = {'Timer',Timer(1:A)/10^6,1};
145     VelD = [0 ; diff(EncDD(1:A))./diff(PosData{1,2})];
146     VelE = [0 ; diff(EncED(1:A))./diff(PosData{1,2})];
147     PosData = [PosData ; {'EncDD',EncDD(1:A),PosData{end,3}+1}];
148     PosData = [PosData ; {'PosD',PosD(1:A),PosData{end,3}+1}];
149     PosData = [PosData ; {'VelD',VelD(1:A),PosData{end,3}+1}];
150     PosData = [PosData ; {'EPosD',PosD(1:A)-EncDD(1:A),PosData{end,3}+1}];
151     PosData = [PosData ; {'EncED',EncED(1:A),PosData{end,3}+1}];
152     PosData = [PosData ; {'PosE',PosE(1:A),PosData{end,3}+1}];
153     PosData = [PosData ; {'VelE',VelE(1:A),PosData{end,3}+1}];
154     PosData = [PosData ; {'EPosE',PosE(1:A)-EncED(1:A),PosData{end,3}+1}];
155
156     if RunMain == -3
157         A = min([ParseVars{[8 9 10 11],4}]);
158
159         ConData = {'Timer',Timer(1:A)/10^6,1};
160         ConData = [ConData ; {'ErroD',Erro_D(1:A),ConData{end,3}+1}];
161         ConData = [ConData ; {'ControleACD',Controle_ACD(1:A),ConData{end,3}+1}];
162         ConData = [ConData ; {'ErroE',Erro_E(1:A),ConData{end,3}+1}];
163         ConData = [ConData ; {'ControleACE',Controle_ACE(1:A),ConData{end,3}+1}];
164     end
165
166
167     %% Diz se teve overflow do canal
168
169     OverflowDetector = (OverflowDetector == 512)
170
171     %% Closes the connection
172     fclose(S);
173     while MyStringComparator(S.Status,'closed') == 0
174         %;
175     end
176     clear S;
```

Apêndice II

Seção dedicada a vídeos de testes relevantes e coleta de dados. Todos os vídeos se encontram no Youtube [43].

Ruído do Encoder

Clique em ’https://youtu.be/X8sZfX_B0_o’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Mostra da Interface

Clique em ’https://youtu.be/HUFrBEntgvc’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Teste dos Sensores

Clique em ’https://youtu.be/xbvhywuKFeI’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Mostra da Movimentação

Clique em ’https://youtu.be/_JkcaUEHzw0’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Movimentação por voz com internet

Clique em ’https://youtu.be/4HsgWWlGOoI’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Movimentação por voz sem internet

Clique em ’https://youtu.be/C7UTLbOTz7s’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.

Processo De medição do Atraso

Clique em ’https://youtu.be/E8SX5ShDi28’ para ver o vídeo no Youtube.

Clique em ’Retorno’ para retornar à menção deste link no documento.