



UFC

UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

HENRIQUE MOREIRA DE ALBUQUERQUE

CONTROLE DE UM TRICICLO ASSISTIDO ESTÁTICO PARA REABILITAÇÃO

FORTALEZA

2024

HENRIQUE MOREIRA DE ALBUQUERQUE

CONTROLE DE UM TRICICLO ASSISTIDO ESTÁTICO PARA REABILITAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Jarbas Silveira

CIDADE

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas

Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A31c Albuquerque, Henrique Moreira de.

Controle de um triciclo assistido estático para reabilitação / Henrique Moreira de Albuquerque. – 2024.

100 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2024.

Orientação: Prof. Dr. Jarbas Aryel Nunes da Silveira.

1. Tecnologias assistivas. 2. Reabilitação. 3. Automação. 4. Protocolo Modbus. 5. Sistema de comunicação. I. Título.

CDD 621.39

HENRIQUE MOREIRA DE ALBUQUERQUE

CONTROLE DE UM TRICICLO ASSISTIDO ESTÁTICO PARA REABILITAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 24/09/2024.

BANCA EXAMINADORA

Prof. Dr. Jarbas Silveira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Jardel Silveira
Universidade Federal do Ceará (UFC)

Prof. Dr. Alexandre Coelho
Universidade Federal do Ceará (UFC)

A Deus.

A minha amada família, especialmente à
minha mãe, Aurinez, e em memória do
meu amado pai, Carlos.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus, por me conceder diariamente forças e motivação para seguir adiante.

À minha família, minha eterna gratidão, em especial à minha mãe, Aurinez, que sempre cuidou de mim com amor e dedicação, sendo fundamental na construção da pessoa que sou hoje. Agradeço também ao meu querido pai, Carlos, que sempre me inspirou a correr atrás de meus sonhos e me superar cada vez mais. Ambos foram pilares em cada uma de minhas decisões e conquistas.

À Universidade Federal do Ceará, pela estrutura oferecida, pelo apoio incondicional durante minha formação, e por proporcionar momentos inesquecíveis que levarei para toda a vida.

Aos meus colegas universitários Arthur, André, Fernando e Hemerson, agradeço por trazerem leveza, alegria e companheirismo aos meus dias de graduação, tornando essa jornada muito mais agradável.

Ao Professor Dr. Jarbas Silveira, expresso minha gratidão pela orientação, oferecendo suporte e ensinamentos ao longo deste caminho. Muito obrigado por todo o conhecimento transmitido.

Aos professores da banca examinadora, agradeço pelo tempo dedicado e pelas valiosas contribuições e sugestões, que enriqueceram ainda mais este trabalho.

Ao técnico Acácio, meu sincero agradecimento pela disposição em me auxiliar, pelos conselhos e pela ajuda indispensável para a conclusão deste projeto.

Ao meu colega de laboratório, Caio, sou grato pelo suporte no desenvolvimento do trabalho.

Por fim, aos colegas do Grupo de Robótica Competitiva (GRC), agradeço pelos momentos de descontração e pelas importantes trocas de conhecimento na área de computação.

“A necessidade é a mãe da inovação.”
(Platão)

RESUMO

O LESC (Laboratório de Engenharia de Sistemas de Computação) é utilizado por alunos, professores e pesquisadores para desenvolvimento de atividades experimentais de ensino, pesquisa e extensão. Dentre os equipamentos disponibilizados, encontra-se um triciclo utilizado para reabilitação dos movimentos das pernas. No entanto, a falta de equipamentos tecnológicos para o controle preciso e centralizado do triciclo faz com que o equipamento não seja aproveitado ao máximo. Dessa forma, este projeto tem como principal motivação utilizar dos conhecimentos da área de automação e controle. Portanto, este trabalho visa desenvolver um sistema de controle de um triciclo assistido estático para reabilitação que utiliza inversor de frequência, modelo CFW100 da fabricante WEG, que é responsável pelo acionamento e controle de um motor AC trifásico da WEG, um microcontrolador ESP32 para realizar a comunicação entre um aplicativo mobile e o inversor de frequência. A comunicação utiliza o protocolo Modbus RTU e o padrão de meio físico RS485. O objetivo deste projeto é disponibilizar um aplicativo mobile que possa operar o triciclo de forma remota com facilidade de uso pelos usuários com interface intuitiva e que o sistema tenha alta performance com respostas dos comandos de tempo satisfatório e com alto nível de segurança e confiabilidade. O projeto uniu conhecimentos de automação, controle e engenharia de sistemas embarcados. Pelas pesquisas, os resultados obtidos indicam que o sistema atende os objetivos propostos e pode considerar que o sistema é fácil de usar, seguro e confiável, tornando uma tecnológica eficiente, segura e acessível para a reabilitação. O sistema desenvolvido oferece uma plataforma inovadora combinando o controle preciso de um equipamento de reabilitação por um aplicativo móvel. Destaca-se ainda melhorias futuras como a otimização dos códigos, melhorias contantes na interface, uso de baterias para melhorar a eficiência energética e integração com IoT.

Palavras-chave: Tecnologia Assistiva; Reabilitação; Automação;

ABSTRACT

The LESC (Computer Systems Engineering Laboratory) is used by students, professors and researchers to develop experimental teaching, research and extension activities. Among the equipment provided is a tricycle used for rehabilitation of leg movements. However, the lack of technological equipment for precise and centralized control of the tricycle means that the equipment is not used to its full potential.

Therefore, the main motivation of this project is to use knowledge from the area of automation and control. Therefore, this work aims to develop a control system for a static assisted tricycle for rehabilitation that uses a frequency inverter, model CFW100 from the manufacturer WEG, which is responsible for driving and controlling a three-phase AC motor from WEG, and an ESP32 microcontroller to communicate between a mobile application and the frequency inverter. The communication uses the Modbus RTU protocol and the RS485 physical media standard. The objective of this project is to provide a mobile application that can operate the tricycle remotely, making it easy for users to use, with an intuitive interface, and for the system to have high performance, with satisfactory response times and a high level of safety and reliability. The project combined knowledge of automation, control, and embedded systems engineering. Based on the research, the results obtained indicate that the system meets the proposed objectives and can be considered easy to use, safe, and reliable, making it an efficient, safe, and accessible technology for rehabilitation. The developed system offers an innovative platform combining precise control of rehabilitation equipment through a mobile application. Future improvements include code optimization, constant improvements to the interface, use of batteries to improve energy efficiency, and integration with IoT.

Keywords: Assistive Technology; Rehabilitation; Automation.

LISTA DE FIGURAS

Figura 2.1 –	Requisição do cliente, resposta do servidor	18
Figura 2.2 –	Pilha de comunicação MODBUS	19
Figura 2.3 -	Tipos de Dados	20
Figura 2.4 -	Funções do MODBUS	20
Figura 2.5 -	RS-485	22
Figura 2.6 -	Sequência de Bits no modo RTU	23
Figura 2.7 -	Sequência de Bits no modo RTU (Sem paridade)	23
Figura 2.8 -	Quadro de Mensagem RTU	23
Figura 2.9 -	Quadro de Mensagem RTU, intervalo entre os quadros	24
Figura 2.10 -	Quadro de Mensagem RTU, tempo entre os caracteres	24
Figura 2.11 -	Sequência de Bits no modo ASCII	25
Figura 2.12 -	Sequência de Bits no modo ASCII (Sem paridade)	25
Figura 2.13 -	Quadro de Mensagem no modo ASCII	25
Figura 3.1 -	Triciclo com quadro	26
Figura 3.2 -	Módulo de Comunicação Serial TTL para RS485	27
Figura 3.3 -	CFW100-CRS485	28
Figura 3.4 -	Esp32	29
Figura 3.5 -	CFW100	30
Figura 3.6 -	Instalação do Inversor	31
Figura 3.7 -	Motor WEG W22 Premium	34
Figura 3.8 -	Proteus	35
Figura 3.9 –	Imagem do Software CadeSimu	36
Figura 3.10 –	Osciloscópio	39
Figura 3.11 –	Multímetro	40
Figura 3.12 –	Regulador de Tensão	41
Figura 3.13 –	Capacitor eletrolítico	42
Figura 3.14 –	LCD 16x4	43
Figura 4.1 -	Diagrama geral do sistema de controle do triciclo assistido estático para reabilitação	44
Figura 4.2 -	Esquema Elétrico no CADe Simu	45
Figura 4.3 -	Encaixe dos componentes	46

Figura 4.4 -	Vista de cima do PCB feita no Altium	47
Figura 4.5 –	Modelagem SolidWorks	48
Figura 4.6 -	Caixa PCB impressa	48
Figura 4.7 -	Esquema de ligação da Simulação Proteus	49
Figura 4.8 -	Realização da simulação no Proteus	50
Figura 4.9 -	Transmissão de dados vistos no osciloscópio	51
Figura 4.10 –	Esquema Elétrico PCB – ESP32 e portas de comunicação e alimentação	51
Figura 4.11 –	Esquema elétrico PCB – Regulador de Tensão	52
Figura 4.12 –	Código para recebimento e leitura de dados	53
Figura 4.13 –	Código para escrita em um único registrador	54
Figura 4.14 –	Código para escrita de múltiplos registradores	54
Figura 4.15 –	Código para leitura de um único registrador	55
Figura 4.16 –	Mapa de memória para o protocolo Modbus RTU	56
Figura 4.17 –	Definição das macros dos registradores	56
Figura 4.18 –	Funções dos bits para o parâmetro P682	57
Figura 4.19 –	Código do comando <i>Start</i>	57
Figura 4.20 -	Tela Home	60
Figura 4.21 -	Tela de Voltas	60
Figura 4.22 -	Tela de Hit	61
Figura 4.23 -	Tela de Configuração	62
Figura 4.24 -	Tela de Menu	62

LISTA DE GRÁFICOS

Gráfico 4.1	Classificação da facilidade de uso pelo aplicativo	63
Gráfico 4.2	Dificuldade de conexão do aplicativo com o triciclo pelos usuários	64
Gráfico 4.3	Classificação ao clicar no aplicativo se caso responder corretamente	64
Gráfico 4.4	Intuitividade da interface do aplicativo segundo os usuários	65
Gráfico 4.5	Clareza e compreensibilidade das informações no aplicativo segundo os usuários	65
Gráfico 4.6	Dificuldade no entendimento do aplicativo segundo os usuários	65
Gráfico 4.7	Satisfação no tempo de resposta do aplicativo segundos os usuários	66
Gráfico 4.8	Falhas ou travamentos durante o uso	67
Gráfico 4.9	Segurança que os usuários sentiram ao controlar o triciclo	68
Gráfico 4.10	Situações em que o aplicativo não respondeu como esperado segundo os usuários	68
Gráfico 4.11	Recomendação dos usuários	69
Gráfico 4.12	Classificação da satisfação da solução proposta pelo sistema segundo os usuários	69

LISTA DE TABELAS

Tabela 3.1 - CFW100-CRS485	29
Tabela 3.2 - ESP32	30

LISTA DE ABREVIATURAS E SIGLAS

TPV	Tempo de Primeira Volta
TVR	Tempo de Voltas Restantes
NV	Número de Voltas
IA	Inteligência Artificial
IoT	Internet da Coisas
RS	Recommendad Standart
EIA	Eletronic Industries Alliance
LCD	Liquid Crystal Display

LISTA DE SÍMBOLOS

%	Porcentagem
Ms	Millessegundos
Hz	Hertz
V	Voltz
A	Àmpere
F	Faraday

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Objetivo do Projeto	15
2	REVISÃO BIBLIOGRÁFICAS	16
2.1	Fisioterapia Ortopédica e Movimentos das Pernas	16
2.2	Automação e Tecnologia	17
2.3	Protocolo Modbus	18
2.3.1	Tipo de Dados	19
2.3.2	Código da Função	20
2.3.3	Meios de Comunicação	21
2.3.4	Modos de Transmissão	22
2.3.4.1	<i>RTU</i>	22
2.3.4.2	<i>ASCII</i>	24
2.1.4.1	<i>TCP/IP</i>	26
3	MATERIAIS E MÉTODOS	27
3.1	Materiais	27
3.1.1	Triciclo	27
3.1.2	Módulo Conversor TTL para RS485	28
3.1.3	Módulo de Comunicação CFW100 CRS485	28
3.1.4	ESP32	30
3.1.5	Inversor de Frequência CFW100	31
3.1.6	Linguagem de Programação C++	33
3.1.7	SolidWorks	34
3.1.8	Motor WEG W22 Premium	35
3.1.9	Software de Simulação Proteus	36
3.1.10	Visual Studio Code	36
3.1.11	Framework Flutter	37
3.1.12	Software CAdE Simu 4.0	37
3.1.13	Osciloscópio	38
3.1.14	Multímetro	39
3.1.15	Fonte de Alimentação de 9V	40
3.1.16	Regulador de Tensão 78L05ACD	41

3.1.17	Capacitores Eletrolíticos	42
3.1.18	LCD 16x4 – WH-1604A	43
4	RESULTADOS	44
4.1	Esquema elétrico do Quadro	45
4.2	Modelagem e Esquema elétrico do Sistema Embarcado	46
4.3	Comunicação	48
4.4	Desenvolvimento do Firmware	52
4.5	Interface Gráfica	59
4.6	Testes no Sistema	63
4.6.1	Análise da Usabilidade do Sistema	63
4.6.2	Discussão dos Resultados	70
5	CONCLUSÃO	71
5.1	Trabalhos Futuros	72
	REFERÊNCIAS	73
	APÊNDICE A – PARAMETRIZAÇÃO DO INVERSOR DE	
	FREQUÊNCIA	76
	APÊNDICE B – ARQUIVO main.cpp	77
	APÊNDICE C – ARQUIVO REG_CFW100.h	92
	ANEXO A – DADOS TÉCNICOS CFW100	95

1 INTRODUÇÃO

Uma das primeiras evidências na história das tecnologias de reabilitação surgiu no antigo Egito com uma evidência de uma prótese de um dedão do pé em uma múmia (FINCH, 2011). Nos dias atuais, é possível verificar uma gama de dispositivos desenvolvidos e em evolução, demonstrando uma tendência no aprimoramento e na diversidade de dispositivos para auxiliar os seres humanos.

Durante a Primeira Guerra Mundial, muitos soldados retornaram com amputações, o que impulsionou o desenvolvimento de próteses e dispositivos ortopédicos, trazendo avanços importantes para outras áreas, como a engenharia mecânica (PERIUS, 2014).

Com o avanço da eletrônica, dispositivos como próteses mioelétricas têm utilizado sinais musculares para controlar movimentos, oferecendo maior inclusão e independência para pessoas com deficiências motoras (BRASIL, 2014).

Após o avanço da eletrônica, houve o surgimento das tecnologias digitais que proporcionaram várias inovações na reabilitação, trazendo exoesqueletos robóticos, dispositivos de realidade virtual e sistemas de biofeedback. Essas tecnologias permitem uma reabilitação mais precisa, personalizada e motivadora, adaptando-se às necessidades específicas de cada paciente. Por exemplo, dispositivos como o HandMATE, um exoesqueleto de mão robótico, têm demonstrado eficácia em reabilitações domiciliares, fornecendo assistência motorizada e feedback em tempo real (Sandison, 2020).

O tratamento das doenças neuromusculares mudou nos últimos anos, do manejo paliativo dos sintomas aos métodos preventivos de capacitação com foco na funcionalidade, participação e atividade (Cândido, 2022). Com pesquisas focando em biotecnologia, neuroestimulação e inteligência artificial, teremos mais avanços na recuperação funcional e melhora na qualidade de vida dos pacientes. O triciclo assistido proposto neste trabalho é um exemplo dessa evolução, que combina princípios de engenharia e fisioterapia para oferecer uma solução inovadora e eficaz para reabilitação dos membros inferiores.

Esta monografia tem como objetivo apresentar o desenvolvimento de um sistema de controle de um triciclo assistido estático para reabilitação. Ele visa proporcionar uma alternativa viável e eficiente para a reabilitação dos membros inferiores. O triciclo é controlado por um aplicativo mobile, permitindo o controle de velocidade e outros comandos. Ferramentas de reabilitação, como exercícios terapêuticos, terapia manual, modalidades físicas e tecnologias assistivas, desempenham um papel vital na recuperação dos pacientes (Silveira et al., 2024). Espera-se que este dispositivo contribua significativamente para a melhoria da mobilidade e da qualidade de vida dos pacientes, alinhando-se às tendências contemporâneas de reabilitação que combinam tecnologia e prática clínica (Carbone & Gonçalves, 2022).

1.1 Objetivo do Projeto

Desenvolver um sistema de controle de um triciclo estático de modo que possa utilizar um aplicativo mobile para operar o triciclo de forma remota, visando proporcionar uma plataforma acessível e eficaz para a reabilitação com facilidade de uso pelos usuários que irão operar o sistema.

- ***Objetivos específicos***

- Implementar um sistema de comunicação sem fio (Bluetooth) entre um dispositivo móvel (celular) e o inversor de frequência, através do microcontrolador ESP32.
- Desenvolver interface de aplicativo para o dispositivo móvel poder realizar o controle do sistema.
- Estudar os princípios de funcionamento do inversor de frequência WEG CFW100, ESP32 e do protocolo Modbus.
- Projetar e implementar uma interface de comunicação serial com o inversor de frequência.
- Analisar os resultados obtidos e propor possíveis melhorias e otimização do projeto, visando aprimorar sua funcionalidade e usabilidade na prática clínica.

2 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica será subdividida em: Fisioterapia Ortopédica e Movimentos das Pernas, Automação e Tecnologia e, por fim, sobre o Protocolo Modbus.

2.1 Fisioterapia Ortopédica e Movimentos das Pernas

A reabilitação é importante para recuperação funcional de pacientes com diversas condições ortopédicas (Melvin, 2020). Dentre elas, crianças com disfunções musculoesqueléticas tem apresentado dificuldades para movimentar as pernas. As limitações físicas com mobilidade reduzida, isso incluindo as pernas, podem afetar negativamente as relações sociais ao longo da vida, impactando a escolaridade e as oportunidades de emprego devido à necessidade de assistência nas atividades da vida diária, acessibilidade reduzida e falta de tecnologia assistiva (Cândido, 2022).

A tecnologia assistiva desempenha um papel muito importante para a reabilitação de pacientes. Dentre elas, podemos destacar o HandMate que é um exoesqueleto de mão robótico integrado com um aplicativo Android para reabilitação domiciliar de pacientes pós-AVC, fornecendo assistência motorizada e feedback em tempo real para otimizar os exercícios terapêuticos (Sandison, 2020).

Além desse projeto, podemos destacar também o projeto de Sophie, 2022 que desenvolveu um projeto de bicicleta assistida ajustável via celular. Ela destaca que as bicicletas elétricas podem atender objetivos de recuperação de doenças crônicas e têm mostrado aceitabilidade entre os pacientes, embora exijam um bom controle do nível de atividade para garantir a segurança do paciente. Além disso, a criação de interfaces gráficas intuitivas e fáceis de usar, como a desenvolvida por Ricarte (2021), facilita a interação e o monitoramento dos parâmetros durante o uso, garantindo uma experiência de usuário simplificada e eficiente (Ricarte, 2021).

Pesquisas recentes destacam a importância de bicicletas assistidas por motor para melhorar a função motora e reduzir os sintomas em pacientes com Parkinson, demonstrando a eficácia dessas intervenções em facilitar o exercício assistido e promover a neuroplasticidade (Z. Meihuan, 2020).

- **Importância da Reabilitação de Movimentos das Pernas**

Pessoas com condições com a distrofia muscular de Duchenne, os exercícios aeróbios de baixa intensidade com bicicletas assistidas mostrou ser eficazes na melhoria da qualidade de vida e das funções pulmonares dos pacientes (Melvin, 2020). A reabilitação dos movimentos das pernas é importante pra fisioterapia ortopédica, assim podendo ajudar a restaurar a força motora, melhorar a coordenação. No entanto, são necessários mais esforços para desenvolver soluções robóticas para reabilitação neurológica que sejam acessíveis a um grande número de pacientes (Carbone & Gonçalves, 2022).

2.3 AUTOMAÇÃO E TECNOLOGIA

Automação é o uso de sistemas de controle para operar equipamentos com intervenção humana mínima ou reduzida (Pearson, 2014). Isso pode ser aplicado em diversas áreas, como na indústria, Tecnologia da Informação (TI), residencial e entre outras áreas em que se tem a oportunidade de automatizar algum processo. Tecnologia pode ser definida como a aplicação de conhecimento científico para propósitos práticos. Envolve o uso de ferramentas, máquinas, materiais e processos para resolver problemas e melhorar a vida humana (Worth Publishers, 2017). Dentre elas, existe um vasto campo em que podemos destacar como a TI, Biotecnologia, Bioengenharia e entre outros. Ambas, podem caminhar lado a lado para criar sistemas avançados como a Inteligência Artificial (IA) que é frequentemente utilizado para automatizar tarefas complexas, como diagnóstico médico, Internet das Coisas (IoT) que são dispositivos utilizados para automatizar tarefas cotidianas, como o controle de temperatura.

Então, o uso da automação nas tecnologias que existem e que ainda estão por vir tem o potencial de transformar a economia, o mercado de trabalho e a vida cotidiana, dessa forma otimizando uma tecnologia e criando novas oportunidades permitindo operar de maneira mais eficiente, produzir de maneira mais acelerada e também auxiliar na supervisão de modo que possa minimizar a intervenção humana nesses meios e também servir como ferramenta de trabalho de algum profissional específico no seu cotidiano.

2.4 PROTOCOLO MODBUS

Modbus é um protocolo de mensagens da camada de aplicação, posicionada no nível 7 do modelo OSI, fornecendo comunicação cliente/servidor entre dispositivos conectados em diferentes tipos de barramentos ou redes (MODBUS). Ele é um protocolo aberto, desenvolvido pela Modicon Industrial Automation Systems (atualmente Schneider Electric) em 1979. A sua aplicabilidade é de larga escala, diversas ferramentas de sistemas supervisórios e controladores utilizam este protocolo sendo a principal justificativa para seu uso a simplicidade e facilidade de implementação (NASCIMENTO; LUCENA, 2003). Na figura 2.1, podemos observar um exemplo de comunicação cliente servidor entre um computador (cliente) enviando uma solicitação de dados para um CLP (servidor).

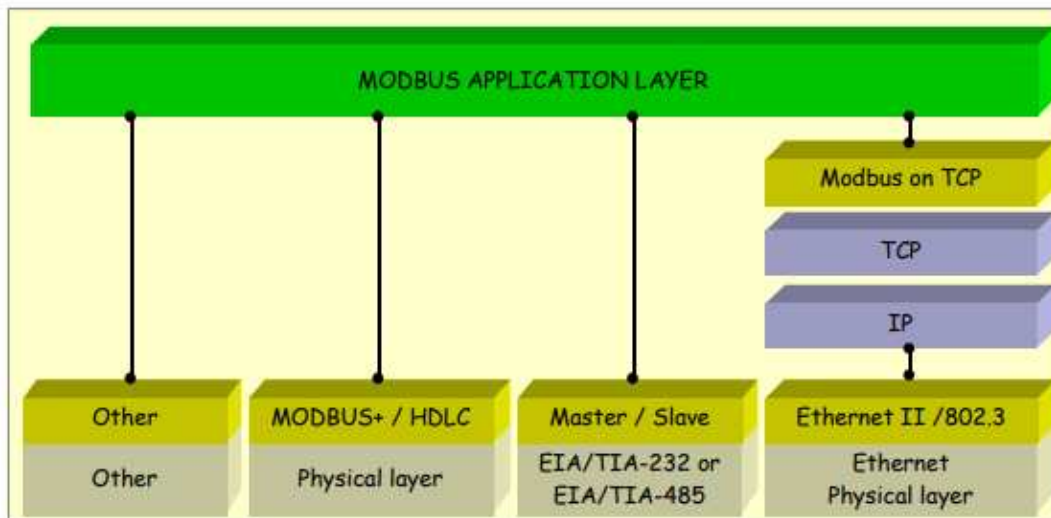
Figura 2.1: Requisição do cliente, resposta do servidor



Fonte: Elaborada pelo próprio autor

Na figura 2.2, está ilustrado a pilha de comunicação modbus. Como podemos observar ele está na camada de aplicação e pode usar várias camadas físicas. Dessa forma, se for utilizar o padrão de meio físico RS-232 ou RS485, será usado o mecanismo mestre-escravo (comunicação serial). No caso da rede Wi-fi ou Ethernet será usado o CSMA/CD ou outros meios.

Figura 2.2: Pilha de comunicação MODBUS



Fonte: Modbus (2024).

- **Troca de dados entre cliente e servidor**

Na troca de dados entre o cliente e o servidor, o cliente envia uma requisição e logo após a requisição o servidor pode ou não responder a solicitação. Caso o servidor tenha aceitado o comando, o servidor envia uma resposta com o mesmo código da função em que o cliente enviou. Caso o servidor não tenha aceitado, ele responderá a requisição com uma resposta de exceção, identificando a causa. Por fim, caso o servidor não responder significa que houve erro de comunicação ou o cliente enviou o endereço broadcast na requisição.

2.4.1 Tipos de Dados

Os dois principais tipos são as variáveis que tem acesso de bit e words (16 bits). Os tipos bits são dados que fornecem aos sistemas de entradas digitais (Discretes Input) e saídas digitais (Coils). As words são fornecidos para Input Registers que são as entradas analógicas e Holding Register que são as saídas analógicas que representa a maioria dos dados. Na Figura 2.3 podemos ver uma tabela que descreve os tipos de dados.

Figura 2.3: Tipos de Dados

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Fonte: Modbus (2024).

2.4.1 Código da Função

No protocolo Modbus, cada função é utilizada para acessar um tipo específico de dados como escrever em uma saída analógica ou ler uma entrada analógica. Na Figura 2.2, está a tabela que mostra a definição dos códigos das funções e podemos observar as funções que são de acesso a Bit e as funções de acesso a Words de 16 bits.

Figura 2.4: Funções do MODBUS

Data Access	Access Type	Physical Discrete Inputs	Read Discrete Inputs	02	02	6.2
			Bit access	Internal Bits Or Physical coils	Read Coils	01
Write Single Coil	05	05			6.5	
Write Multiple Coils	15	0F			6.11	
16 bits access	Physical Input Registers	Read Input Register	04	04	6.4	
		Read Holding Registers	03	03	6.3	
	Internal Registers Or Physical Output Registers	Write Single Register	06	06	6.6	
		Write Multiple Registers	16	10	6.12	
		Read/Write Multiple Registers	23	17	6.17	
		Mask Write Register	22	16	6.16	
		Read FIFO queue	24	18	6.18	
		Read File record	20	14	6.14	
	File record access	Write File record	21	15	6.15	
		Read Exception status	07	07	6.7	
Diagnostics	Diagnostics	08	00-18,20	08	6.8	
	Get Com event counter	11	0B	6.9		
	Get Com Event Log	12	0C	6.10		
	Report Server ID	17	11	6.13		
	Read device Identification	43	14	2B	6.21	
Other	Encapsulated Interface Transport	43	13,14	2B	6.19	
	CANopen General Reference	43	13	2B	6.20	

Fonte: Modbus (2024).

No projeto, as funções utilizadas foram a Write Single Register que é usado para escrever em um único registrador do tipo Holding, o Write Multiple Registers que é usado para escrever em um bloco contínuo de registradores de 1 a 123 registradores e Read Holding Register.

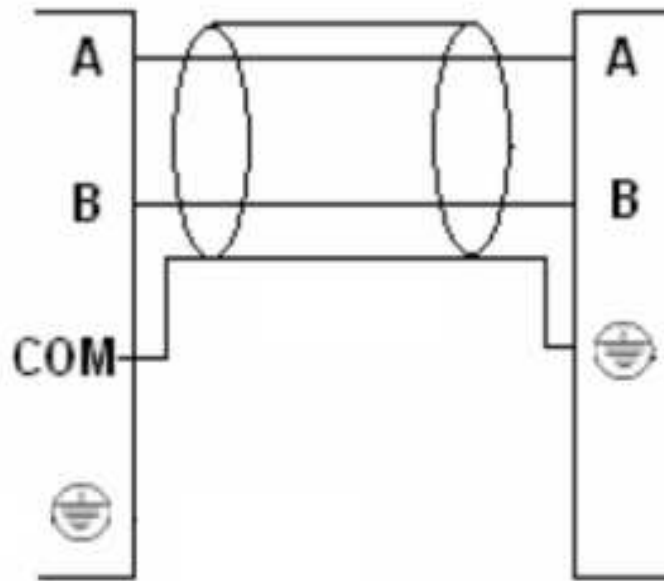
2.4.2 Meios de Comunicação

O protocolo ModBus pode ser utilizado em diversos padrões de meio físico, como: RS-232, RS-485 e Ethernet TCP/IP (MODBUS TCP). Assim, variando a velocidade de comunicação, comprimento máximo da rede e o número máximo de dispositivos conectados.

O padrão RS-232 (Recommendad Stardart-232) ou EIA-232 (Eletronic Industries Alliance-232) é utilizado apenas em comunicações ponto a ponto, ou seja, só admite dois dispositivos na rede, que no caso do protocolo Modbus representa o mestre e 1 escravo. A velocidade máxima desse padrão está em torno de 115Kbps, mas em alguns casos podem ser encontrados taxas um pouco maiores, a distância máxima entre os dispositivos da rede está em torno de 30m. Então, ele é simples e adequado para comunicações de curto alcance ponto a ponto, mas limitado em termos de distância e número de dispositivos

O padrão RS-485 (Recommendad Stardart-485) ou EIA-485 (Eletronic Industries Alliance-485) é muito utilizado na indústria. Sua distância máxima pode chegar até 1200 metros, dependendo da taxa de transmissão. Lembrando que quanto maior o comprimento da rede menor será a velocidade comunicação. O número máximo de dispositivos no barramento é 32. Esse padrão é ideal para redes industriais que exigem comunicação confiável entre múltiplos dispositivos em distâncias maiores, com alta imunidade de ruído. Na Figura 2.5, podemos observar uma ilustração que mostra como funciona as conexões do padrão RS485.

Figura 2.5: RS-485



Fonte: Nascimento e Lucena (2003).

O padrão Ethernet no protocolo Modbus podem chegar a 100 Mbps ou até 10 Gbps. A distância máxima pode variar de 100m até próximo de 200m dependendo do tipo de cabo utilizado e das condições de instalação do mesmo. Nesse padrão oferece maior flexibilidade e velocidade, permitindo integração de dispositivos em redes IP modernas, incluindo comunicação remota e suporte a IoT.

2.4.3 Modos de Transmissão

2.4.3.1 RTU

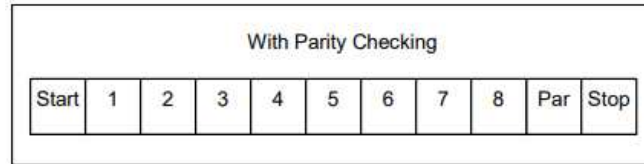
Nesse modo de transmissão os dispositivos se comunicam em uma linha serial usando o modo RTU (Remote Terminal Unit), cada byte de 8 bits em uma mensagem contém dois caracteres hexadecimais de 4 bits. A principal vantagem desse modo é que sua maior densidade de caracteres permite melhor transferência de dados do que o modo ASCII para a mesma taxa de transmissão. Cada mensagem deve ser transmitida em um fluxo contínuo de caracteres.

- **Os caracteres são transmitidos serialmente**

Cada caractere ou byte são transmitidos da esquerda para direita, como podemos observar na Figura 2.6 que ilustra a sequência de bits, contendo 1 start bit,

8 bits de dados (bit menos significativo enviado primeiro, 1 bit para conclusão da paridade e 1 stop bit).

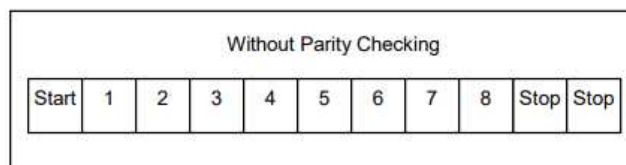
Figura 2.6: Sequência de Bits no modo RTU



Fonte: (MODBUS, 2024)

Os dispositivos podem ser configurados com paridade par, ímpar ou sem paridade. Se não for implementada paridade, um stop bit é adicionado para preencher o quadro de caracteres, como podemos ver na Figura 2.7 para um carácter assíncrono completo de 11-bits.

Figura 2.7: Sequência de Bits no modo RTU (Sem paridade)



Fonte: (MODBUS, 2024)

- **Descrição do quadro:**

Na figura 2.8 está a descrição do quadro, sendo que seu tamanho máximo do quadro do MODBUS RTU é 256 bytes, em que 1 byte é para o endereço do sevidor, 1 byte para o código da função, 252 bytes para os dados e 2 bytes para o CRC. O campo de checagem de erros é baseado no método CRC (Cyclical Redundancy Checking).

Figura 2.8: Quadro de Mensagem RTU

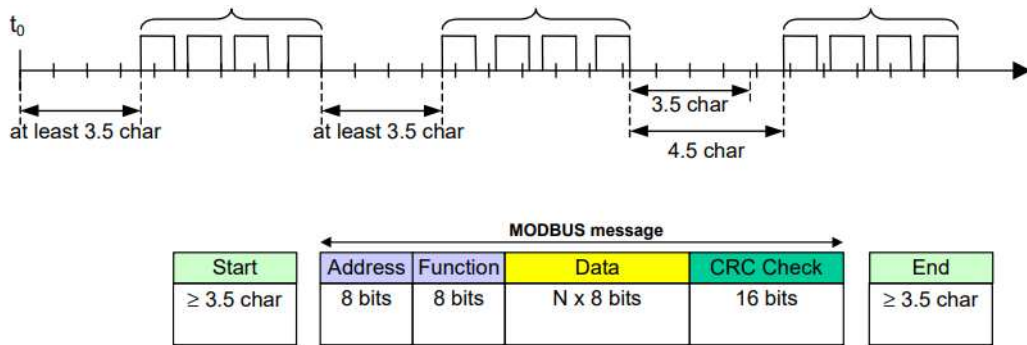
Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Fonte: (MODBUS, 2024)

- **Enquadramento de mensagens**

No modo RTU, os quadros são separados por um intervalo de pelo menos 3,5 tempos de caracteres. Na figura Figura 2.9 está ilustrando o tempo entre os quadros de mensagens.

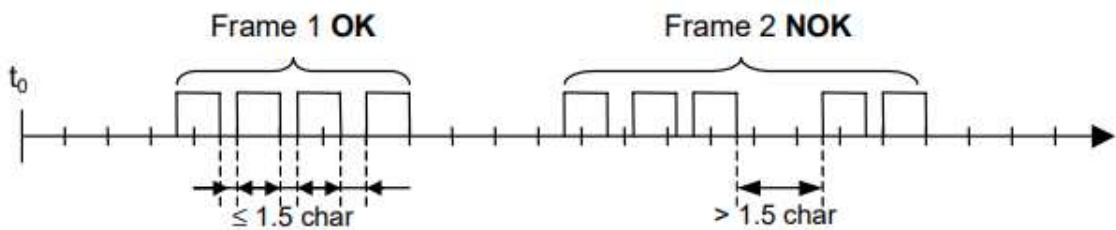
Figura 2.9: Quadro de Mensagem RTU, intervalo entre os quadros



Fonte: (MODBUS, 2024)

Todo o quadro de mensagem deve ser transmitido como um fluxo contínuo de caracteres. Se um intervalo de mais de 1,5 tempos de caracteres ocorrer entre dois caracteres, o quadro de mensagem será declarado incompleto e deve ser descartado pelo receptor. Na Figura 2.10 está ilustrando o tempo permitido entre os caracteres de modo que a mensagem seja completa.

Figura 2.10: Quadro de Mensagem RTU, tempo entre os caracteres



Fonte: (MODBUS, 2024)

2.4.3.2 ASCII

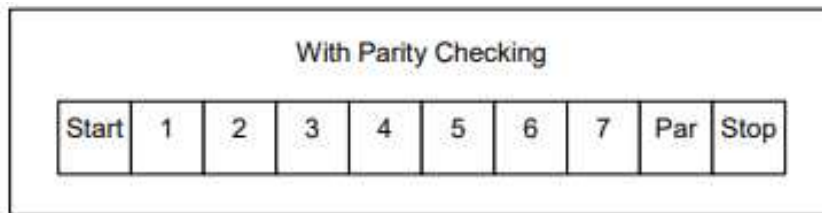
Nesse modo de transmissão é utilizado o ASCII (American Standard Code for Information Interchange), cada byte de 8 bits em uma mensagem é enviado como dois caracteres ASCII. Este modo é usado quando o link de comunicação não permite o modo RTU. Esse modo é menos eficiente que o RTU, pois cada byte precisa de dois caracteres. Exemplo: O byte 0x5B é codificado como dois caracteres: 0x35 e 0x42 (0x35 ="5" e 0x42 ="B" em ASCII).

O formato de 10 bits para cada byte modo em ASCII é 1 start bit, 7 bits de dados (bit menos significativo enviado primeiro), 1 bit para conclusão de paridade e 1 stop bit.

- **Paridade**

Cada caractere ou byte é enviado na ordem da esquerda para direita, como podemos observar na Figura 2.11 que está ilustrando o quadro de mensagem com um start bit, 7 bits de dados, 1 bit para conclusão de paridade e 1 stop bit.

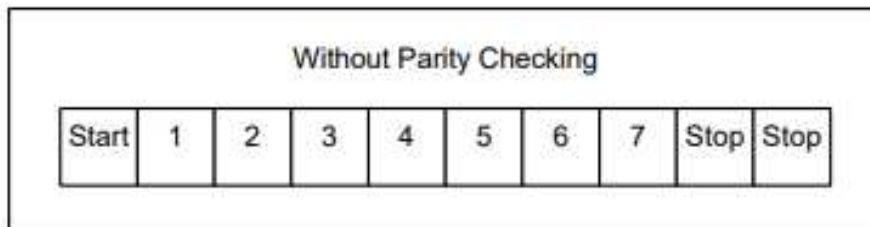
Figura 2.11: Sequencia de Bits no modo ASCII



Fonte: (MODBUS, 2024)

Assim como o RTU, os dispositivos podem ser configurados com paridade par, ímpar ou sem paridade. Se não for implementada paridade, um stop bit é adicionado para preencher o quadro de caracteres para um caractere assíncrono completo de 11-bits. Na Figura 2.12, ilustra o quadro de caracteres com dois stop bits.

Figura 2.12: Sequencia de Bits no modo ASCII (Sem paridade)



Fonte: (MODBUS, 2024)

- **Descrição do quadro:**

Na Figura 2.13 está a descrição do quadro no modo ASCII. No modo ASCII a mensagem é delimitada pelos caracteres Start e End. O tamanho máximo do quadro no ASCII é de 513 caracteres. Cada campo do quadro são caracteres ASCII e não bits.

Figura 2.13: Quadro de Mensagem no modo ASCII

Start	Address	Function	Data	LRC	End
1 char	2 chars	2 chars	0 up to 2x252 char(s)	2 chars	2 chars CR,LF

Fonte: (MODBUS, 2024)

2.4.3.3 TCP/IP

O modo TCP/IP é uma variante moderna do protocolo que utiliza redes Ethernet para comunicação. Permite a integração de dispositivos Modbus em redes IP, facilitando a conectividade com sistemas de TI e a integração com a Internet da Coisas (IoT).

O Modbus é baseado na comunicação cliente/servidor. O mestre faz a solicitação da mensagem, ou seja, inicializa a comunicação, também conhecida como request, enquanto que os demais dispositivos escravos respondem a solicitação com envio de dados ou realizando alguma ação, esta mensagem é chamada de reply ou response. O mestre pode trabalhar em dois modos: modo requisição e resposta, modo descrito anteriormente, na qual o mestre envia uma mensagem para um escravo e este o retorna com uma resposta; outro modo é chamado de difusão ou broadcast, na qual o mestre pode enviar uma mensagem comum a todos os escravos. A Figura 2.6 ilustra estes dois modos de comunicação realizados pelo mestre.

3 MATERIAIS E MÉTODOS

O capítulo abordará os principais materiais utilizados e uma breve descrição de suas características.

3.1 Materiais

3.1.1 Triciclo

Um triciclo é um veículo de três rodas que pode ser projetado para diversas finalidades, incluindo transporte, recreação e reabilitação. Em comparação com bicicletas tradicionais, os triciclos oferecem maior estabilidade e equilíbrio devido à sua configuração de três rodas, o que os torna especialmente adequados para pessoas com dificuldades de mobilidade ou equilíbrio. No contexto da fisioterapia, eles podem ser usados para exercícios terapêuticos, melhorando a força muscular, a coordenação e a resistência física dos pacientes. Na Figura 3.1, temos a foto do triciclo com o quadro elétrico. Na foto o triciclo estava finalizando o processo de montagem do suporte do quadro.

Figura 3.1: Triciclo com quadro

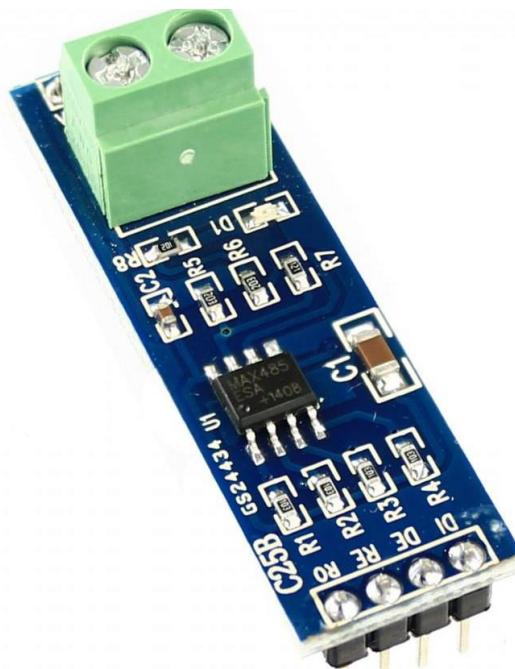


Fonte 1: Elaborada Pelo Próprio Autor

3.1.2 Módulo Conversor TTL para RS485

O módulo funciona como um transceptor, em que ele transmite e recebe dados dessa forma estabelecendo uma comunicação. Na transmissão dos dados, ele converte os dados em TTL para RS485 e no recebimento ocorre a conversão de RS485 para TTL. O módulo utiliza o CI MAX485 para realizar as conversões dos dados. Uma imagem ilustrativa é retratada na Figura 3.4.

Figura 3.2: Modelo de comunicação Serial TTL para RS485.



Fonte: (USINAINFO, 2024)

Este conversor possui uma placa de 8 pinos com terminais que facilita a conexão por meio de jumpers.

3.1.3 Módulo de comunicação CFW100 CRS485

O módulo de comunicação CFW100 CRS485 é um acessório utilizado para adicionar funcionalidade de comunicação serial ao inversor de frequência CFW100, da WEG. Este módulo permite a integração do inversor em redes de comunicação que utilizam o padrão físico RS-485. Na Figura 3.3, temos a foto do módulo de

comunicação, na foto é possível ver os terminais de 6 e 7 que foram utilizados para comunicação no e a porta 8 para o GND do circuito.

Figura 3.3: CFW100-CRS485.



Fonte 2: (WEG, 2019)

Na Tabela 3.1 temos as principais características do CFW100 CRS485:

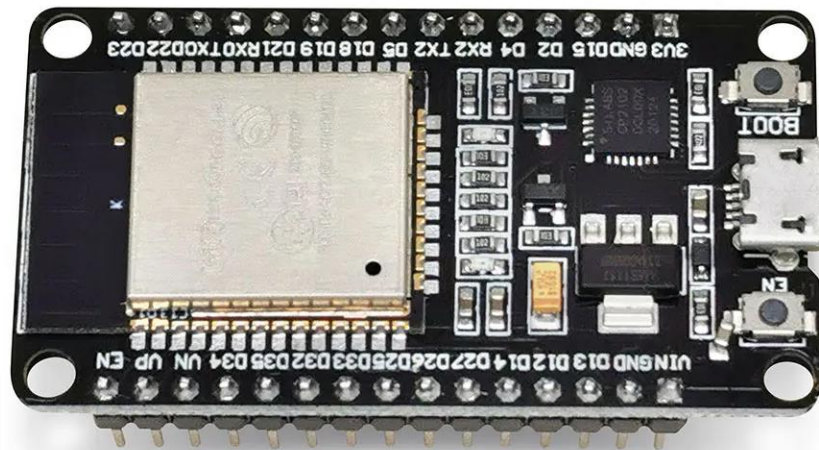
Tabela 3.1 – CFW100 CRS485

Informações	Descrição
Comunicação	Comunicação a longas distâncias e em ambientes industriais ruidosos.
Redes Industriais	Permite que o inversor de frequência CFW100 seja integrado a sistemas de automação e controle
Conectividade	Suporta múltiplos dispositivos conectados na mesma rede, possibilitando a criação de sistemas de controle distribuído.
Configuração e Monitoramento	Facilita a configuração dos parâmetros do inversor e o monitoramento de seu desempenho através de comandos e consultas via rede RS-485.

3.1.4 ESP32

O ESP32 é um microcontrolador de alto desempenho desenvolvido pela Espressif Systems, com suporte integrado para Wi-Fi e Bluetooth. Equipado com um processador dual-core Tensilica Xtensa LX6 de até 240 MHz. Ele é amplamente utilizado em aplicações de IoT e automação, o ESP32 é compatível com plataformas de desenvolvimento populares. Neste trabalho, foi integrado para controlar o inversor de frequência WEG CFW100 e o motor AC do triciclo assistido, permitindo comunicação via protocolo Modbus e operação eficiente do sistema. Na Figura 3.4, temos a foto do ESP32 que foi utilizado para realizar a comunicação entre o aplicativo móvel e o inversor de frequência.

Figura 3.4: Esp32



Fonte: (NETDRINKS, 2024)

Tabela 3.2 temos as principais informações técnicas:

Tabela 3.2 – ESP32

Informações	Descrição
Processador	Dual-core Tensilica Xtensa LX6, até 240 MHz.
Memória	Até 520 KB de SRAM.
Conectividade	Wi-Fi 802.11 b/g/n e Bluetooth v4.2 (BLE).

GPIO	Mais de 30 pinos configuráveis para PWM, ADC, DAC, I2C, SPI, UART, etc.
Periféricos	ADC de 12 bits, DAC, sensores de toque, interfaces SPI, I2C, I2S, UART.
Segurança	Criptografia de hardware e armazenamento seguro de chaves.
Consumo de Energia	Modo deep sleep com consumo muito baixo.

3.1.5 Inversor de Frequência CFW100

Neste projeto é empregado o inversor de frequência modelo CFW100 da fabricante WEG. Uma imagem ilustrativa é retratada na Figura 3.5 que remonta o inversor. O inversor foi acoplado dentro do quadro junto com as boteiras, os fios de energia, fonte de alimentação de 9V e contatora.

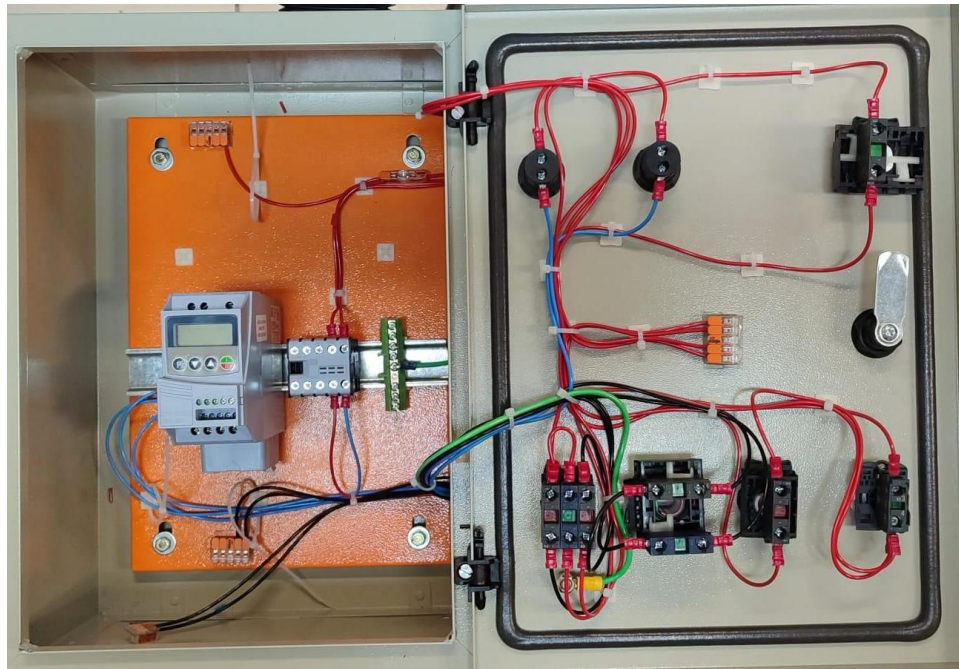
Figura 3.5: CFW100



Fonte: (WEG. Manual do Usuário Inversor de Frequência CFW100, 2017)

Na Figura 3.6 é apresentado o CFW100 instalado no laboratório.

Figura 3.6: Instalação do Inversor



Fonte: Elaborada pelo Próprio Autor

Na Figura 3.6 temos uma foto que mostra o local de instalação do inversor de frequência junto com os outros componentes do quadro.

As especificações técnicas do inversor de frequência estão no Anexo A, com informações diretamente da Folha de Dados.

O modelo CFW100 da WEG possibilita a utilização do protocolo de comunicação Modbus RTU, além de aceitar o padrão físico RS485, com a utilização do módulo CRS485.

Os parâmetros configurados para utilização podem ser analisados no Apêndice A.

3.1.6 Linguagem de Programação C++

Em 1980, Bjarne Stroustrup, da Bell Labs, iniciou o desenvolvimento de uma nova linguagem com bases na linguagem C. Esta, incrementava, em relação à linguagem C, conceitos de OOP (*Object-Oriented Programming*) com o uso das classes. Em 1983 recebeu o nome formal de C++ (++ é chamado de pós-incremento na linguagem C), quando da publicação do seu primeiro manual. Em 1985 nasceu a primeira versão comercial da C++ e a primeira edição do livro “The C++ Programming Language” de Bjarne Stroustrup (Zamboni, 2006).

- **A linguagem C++ tem características superiores às de outras linguagens:**
 - **Orientação a objeto:** Permite ao programador projetar aplicações sob a ótica do encapsulamento, da hereditariedade e do polimorfismo, possibilitando o reuso de uma maneira mais lógica e produtiva;
 - **Portabilidade:** Pode-se praticamente compilar um código C++ em quase todos os tipos de computadores e sistemas operacionais;
 - **Sintaxe compacta:** O código escrito em C++ é mais curto do que o escrito em outras linguagens;
 - **Compatibilidade com C:** Qualquer código escrito em C pode ser incluído em um programa C++ sem duras penas;
 - **Velocidade:** O código gerado por um compilador C++ é muito eficiente e adequado no que tange a dualidade da C++ com uma linguagem de alto nível e como uma linguagem de baixo nível;
 - **Compiladores e ambientes gratuitos:** É o caso do Borland C++ Compiler (BCC), do Bloodshed Dev-C++; do GCC (compilador multi-linguagem), do Digital Mars C++, do DJGPP, do Mingw32, do Intel C++ Compiler, etc. Esses e diversos outros compiladores e ambientes integrados podem ser adquiridos em revistas especializadas ou baixados da Internet;
 - **Bibliotecas genéricas:** É o caso da STL (Standart Template Library), uma coleção de contêineres, algoritmos e iteradores muitíssimo utilizados em computação.

3.1.7 SolidWorks

O SolidWorks é um software de modelagem 3D baseado em CAD (Computer-Aided Design), utilizado para projetar e desenvolver produtos em diversas indústrias, como engenharia mecânica, automotiva, aeroespacial e manufatura. Desenvolvido pela Dassault Systèmes, o SolidWorks permite a criação de modelos tridimensionais

detalhados, simulações e análises de desempenho, além de gerar desenhos técnicos para a fabricação.

Além disso, integra ferramentas de simulações que permitem que os engenheiros analisem o comportamento de suas criações sob diferentes condições, como estresse, fluxo de fluido, e transferência de calor, antes de produzir protótipos físicos. Ademais, oferece recursos colaborativos, facilitando o trabalho em equipe e a gestão de projetos complexos.

3.1.7 Motor WEG W22 Premium

O motor W22 Premium da WEG é um motor elétrico trifásico, projetado para diversas aplicações industriais. Destaca-se pela sua construção robusta, eficiência energética e ampla faixa de potências e tamanhos. Ele oferece diversas proteções e recursos adicionais, sendo compatível com normas internacionais de qualidade e segurança. Na Figura 3.7, temos uma ilustração do motor Weg W22 Premium de cor azul que foi usado para fazer o controle do sistema.

Figura 3.7: Motor Weg W22 Premium

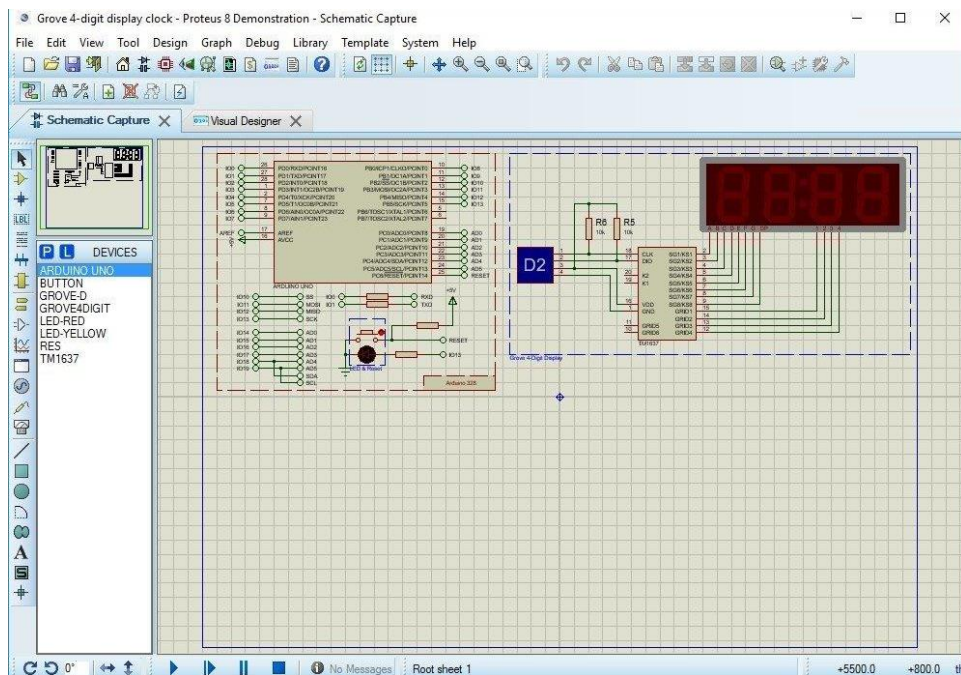


Fonte: (WEG. Motor W22 Plus Premium, 2024)

3.1.8 Software de Simulação Proteus

Desenvolvido pela Labcenter Electronics, o Proteus oferece uma plataforma integrada para a criação, simulação e teste de circuitos. Na Figura 3.8, temos uma *printscreen* da tela do software demonstrando os dispositivos e ferramentas que o mesmo permite utilizar nos projetos. Ele foi bastante utilizado para realizar os testes de comunicação do aplicativo.

Figura 3.8: Proteus



Fonte: Elaborada Pelo Próprio Autor

3.1.9 Visual Studio Code

O Visual Studio Code (VS Code) tem suporte para uma ampla gama de linguagens de programação, ferramentas integradas e uma comunidade ativa, o VS Code se tornou uma das ferramentas preferidas de desenvolvedores em todo o mundo.

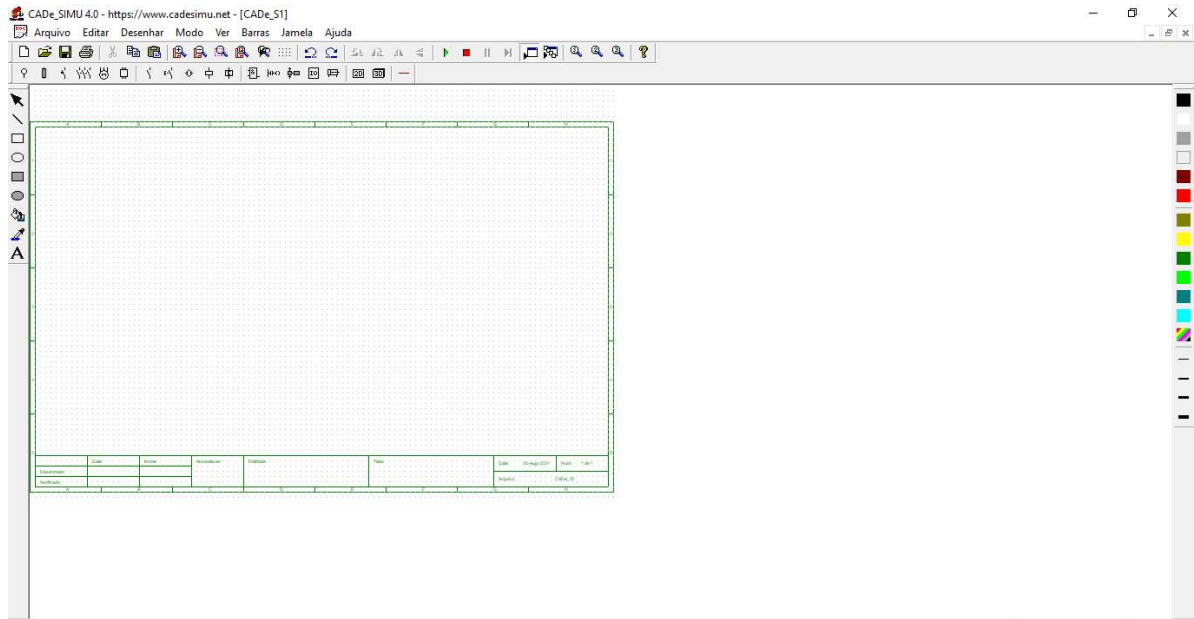
3.1.10 Framework Flutter

O Flutter é um framework de desenvolvimento de aplicativos móveis multiplataforma, desenvolvido pelo Google, que tem ganhado destaque na indústria devido à sua eficiência e flexibilidade. Este framework permite criar aplicativos nativos para dispositivos móveis, web e desktop a partir de um único código-base, tornando-o uma escolha atraente para desenvolvedores em todo o mundo.

3.1.11 Software CADe Simu 4.0

O CADe SIMU 4.0 permite a criação e simulação de circuitos de comando e potência, facilitando a análise e a verificação de funcionamento dos sistemas antes de sua implementação física. Com uma interface intuitiva, o CADe SIMU 4.0 suporta a inserção de diversos componentes elétricos, como relés, contadores, chaves, motores, e elementos de controle, possibilitando a simulação de circuitos complexos de maneira eficiente. Além disso, permite a geração de relatórios e documentação técnica para a apresentação e validação dos projetos. Na Figura 3.9, temos uma *printscreen* do software que foi utilizado para realizar as simulações do circuito do quadro elétrico.

Figura 3.9: Imagem do Software CadeSimu



Fonte: Elaborado pelo Próprio Autor

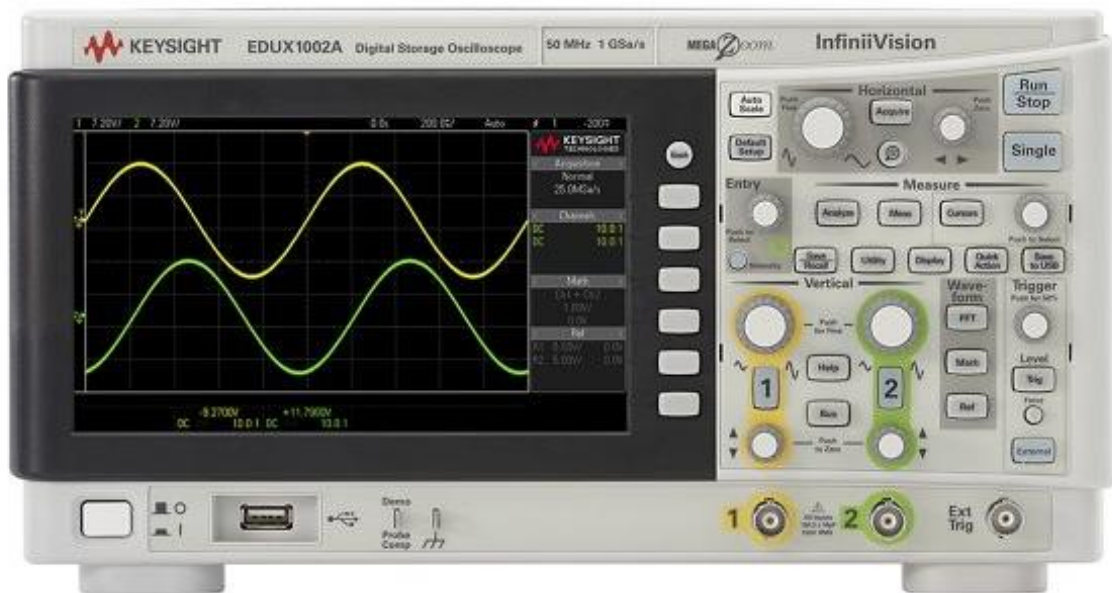
3.1.11 Osciloscópio

O osciloscópio é um equipamento eletrônico para a medição e análise de sinais elétricos variáveis no tempo, permitindo a visualização gráfica de tensões em função do tempo, exibidas na forma de ondas. Este instrumento oferece essenciais, como amplitude, frequência, período, forma de onda e a presença de distorções ou ruídos em sinais elétricos.

No contexto deste trabalho, o osciloscópio foi utilizado para verificar o comportamento dos dados em forma de sinais elétricos gerados pelo ESP32 e o Conversor TTL para RS485. Ele permitiu validar que os dados definidos no código estavam sendo enviados antes de conectar os fios de comunicação diretamente módulo CRS485 acoplado no inversor de frequência.

Na Figura 3.10, temos uma imagem do osciloscópio utilizado que foi da marca Keysight do modelo EDUX1002A de cor branca.

Figura 3.10: Osciloscópio



Fonte: (Keysight. EDUX1002A, 2024)

3.1.12 Multímetro

O multímetro é um instrumento eletrônico portátil usado para medir diversas grandezas elétricas, como tensão (V), corrente (A) e resistência (Ω). Ele pode operar em modo digital ou analógico, sendo amplamente utilizado para a verificação, diagnóstico e manutenção de circuitos eletrônicos. No projeto em questão, o multímetro desempenhou um papel essencial durante o processo de montagem e testes do sistema de controle do triciclo assistido, permitindo medições precisas de diferentes parâmetros elétricos.

O multímetro foi utilizado principalmente para verificar a tensão de alimentação do ESP32, garantindo que a fonte de 9V estivesse fornecendo a energia necessária de maneira estável. Além disso, foi utilizado para medir a resistência em componentes do circuito, como os atuadores, garantindo que esses componentes estivessem funcionando corretamente. Além disso, foi utilizado para fazer testes de continuidade no circuito para assegurar possíveis falhas nos fios e componentes do circuito do sistema embarcado e do quadro elétrico.

Graças à sua versatilidade e precisão, o multímetro permitiu uma avaliação detalhada de possíveis falhas no circuito, facilitando ajustes durante o processo de prototipagem. Ele foi um instrumento importante para garantir a integridade elétrica e

funcional de todo o sistema, desde a etapa de montagem até os testes finais de desempenho.

Figura 3.11: Multímetro



Fonte: Elaborada Pelo Próprio Autor

3.1.13 Fonte de Alimentação de 9V

A fonte de alimentação de 9V é um dispositivo responsável por converter a tensão da rede elétrica para uma tensão contínua adequada para alimentar circuitos eletrônicos. No caso deste projeto, a fonte de 9V foi utilizada para fornecer energia ao módulo ESP32, um microcontrolador que requer alimentação estável para o seu correto funcionamento.

Essa fonte de alimentação oferece uma saída de 9V em corrente contínua (DC), que é posteriormente regulada por um regulador de tensão para 5V que por sua vez irá alimentar o ESP32, conforme necessário para operar seus componentes. Uma tensão estável é crucial para evitar falhas de operação, interferências ou danos aos componentes sensíveis do microcontrolador. Além disso, a escolha de uma fonte de 9V garante que o sistema tenha uma margem de segurança na alimentação, evitando quedas de tensão que poderiam prejudicar a comunicação ou o controle dos atuadores do projeto.

3.1.14 Regulador de Tensão 78L05ACD

O 78L05ACD é um regulador de tensão linear de três terminais que fornece uma saída estável de 5V DC a partir de uma entrada de tensão maior, geralmente entre 7V e 35V. Ele é amplamente utilizado em projetos eletrônicos para garantir uma alimentação estável e filtrada, protegendo os componentes sensíveis contra variações de tensão que poderiam causar mau funcionamento ou danos.

No contexto deste projeto, o regulador 78L05ACD foi utilizado para reduzir e estabilizar a tensão fornecida pela fonte de alimentação de 9V, de modo a alimentar de maneira segura o ESP32 e outros componentes que requerem uma tensão de operação de 5V. Ele é capaz de fornecer uma corrente de até 100 mA, o que é suficiente para muitos microcontroladores e circuitos de baixa potência. Além disso, o 78L05ACD oferece proteção contra sobreaquecimento e curto-circuito, proporcionando uma camada adicional de segurança ao sistema.

Sua implementação no projeto foi importante para garantir que, mesmo com variações na entrada de tensão, o circuito recebesse uma alimentação constante de 5V, evitando falhas na comunicação entre os sensores e o microcontrolador. Na Figura 3.13, temos uma foto do regulador de tensão utilizado.

Figura 3.12: Regulador de Tensão



Fonte: (MiniKits. 78L05ACD, 2024)

3.1.15 Capacitores Eletrolíticos

Os capacitores eletrolíticos são componentes passivos utilizados em circuitos eletrônicos para armazenar energia elétrica temporariamente e suavizar flutuações de tensão. Diferentemente de outros tipos de capacitores, eles possuem uma capacitância relativamente alta, o que os torna ideais para aplicações que requerem filtragem de baixas frequências ou armazenamento de grandes quantidades de carga em pequenos volumes.

Esses capacitores utilizam um eletrólito como meio de armazenamento e possuem polaridade, ou seja, devem ser conectados com cuidado para que a corrente flua na direção correta, sob o risco de danificação. São amplamente empregados em fontes de alimentação para suavizar picos de corrente e tensão, em circuitos de desacoplamento para eliminar ruídos de baixa frequência, e em sistemas de temporização.

No projeto, os capacitores eletrolíticos foram usados em conjunto com o regulador de tensão LD1117S33 para filtrar e estabilizar a tensão de alimentação do ESP32. Os capacitores utilizados foram de $10\mu\text{F}$ 16V, 100nF 25V com 10% de tolerância, e $47\mu\text{F}$ 16V. Na imagem a seguir podemos ver a imagem de um capacitor. Na Figura 3.14, temos exemplo de um capacitor eletrolítico de $10\mu\text{F}$ de 16V que foi utilizado no projeto em conjunto com o regulador de tensão.

Figura 3.13: Capacitor eletrolítico



Fonte: (AutoCore. Capacitor Eletrolítico $10\mu\text{F}$ 16V, 2024)

3.1.16 LCD 16x4 – WH-1604A

O display LCD 16x4 é uma interface visual de cristal líquido, com capacidade para exibir até 16 caracteres por linha em 4 linhas. No projeto, o LCD 16x4 foi utilizado para exibir informações dos feedbacks dos comandos enviados pelo aplicativo.

Figura 3.14: LCD 16x4



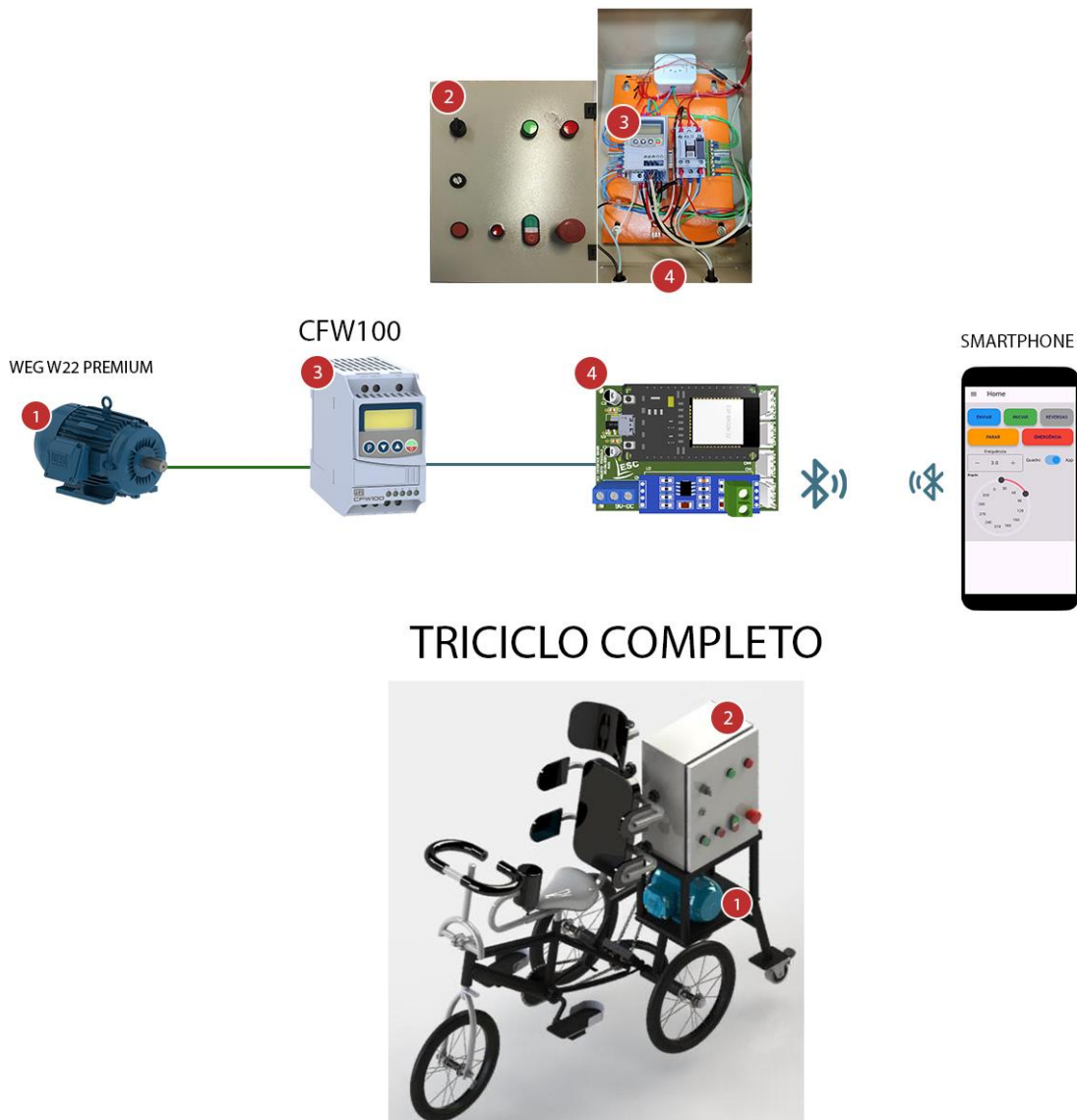
Fonte: (WINSTAR. Display LCD de 16 caracteres por 4 linhas, 2024)

4 RESULTADOS

Este capítulo tem como objetivo apresentar, detalhadamente, o desenvolvimento do presente projeto e os resultados obtidos. Em suma, o projeto que teve como objetivo desenvolver um sistema de controle de um triciclo assistido estático no laboratório do LESC por meio de um dispositivo mobile, agregou conhecimentos da área de redes industriais, através do protocolo de comunicação Modbus RTU, implementado na linguagem C++, reabilitação assistida e automação.

O projeto completo desenvolvido pode ser observado pela ilustração da Figura 4.1 em que se pode observar como funciona a comunicação e o controle dos principais componentes do projeto.

Figura 4.1: Diagrama geral do sistema de controle do triciclo assistido estático para reabilitação



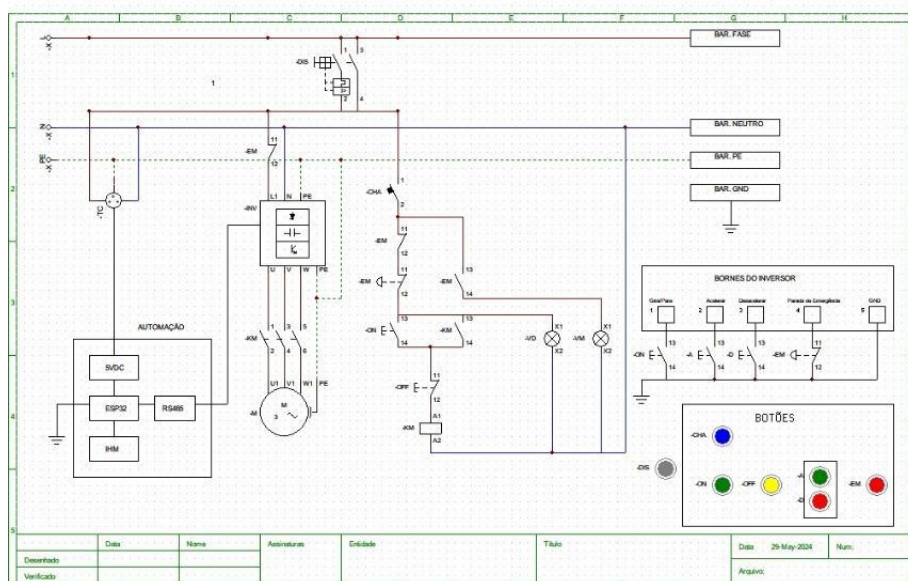
Fonte: Elaborada pelo Próprio Autor

Para facilitar o entendimento do capítulo, foi dividido o desenvolvimento do projeto em 6 partes: a primeira corresponde ao esquema elétrico do quadro; a segunda parte descreve a modelagem e esquema elétrico do sistema embarcado; a terceira descreve a comunicação entre aplicativo e o Esp32 e também a comunicação Cliente/Servidor entre inversor e Esp32; a quarta o desenvolvimento do Firmware; a quinta o desenvolvimento da interface gráfica e, por fim, a sexta apresenta os testes realizado no laboratório com os participantes.

4.1 Esquema elétrico do Quadro

Para o desenvolvimento do quadro instalado no triciclo, foi usado o simulador CAde Simu para simular o esquema elétrico de ligação dos fios, inversor de frequência, contator, botoeiras, motor e Leds. Com o simulador foi possível testar o funcionamento do quadro antes de fazer as ligações e ter uma maior garantia de funcionamento. Na Figura 4.2 está ilustrado o esquema desenvolvido para a montagem do circuito. No esquema podemos notar as caixas com título de “AUTOMAÇÃO” que representa o esquema do sistema de controle do inversor representam os botões, a caixa com título de “BOTÕES” que representa os botões de controle presente no quadro.

Figura 4.2: Esquema elétrico no CAde Simu



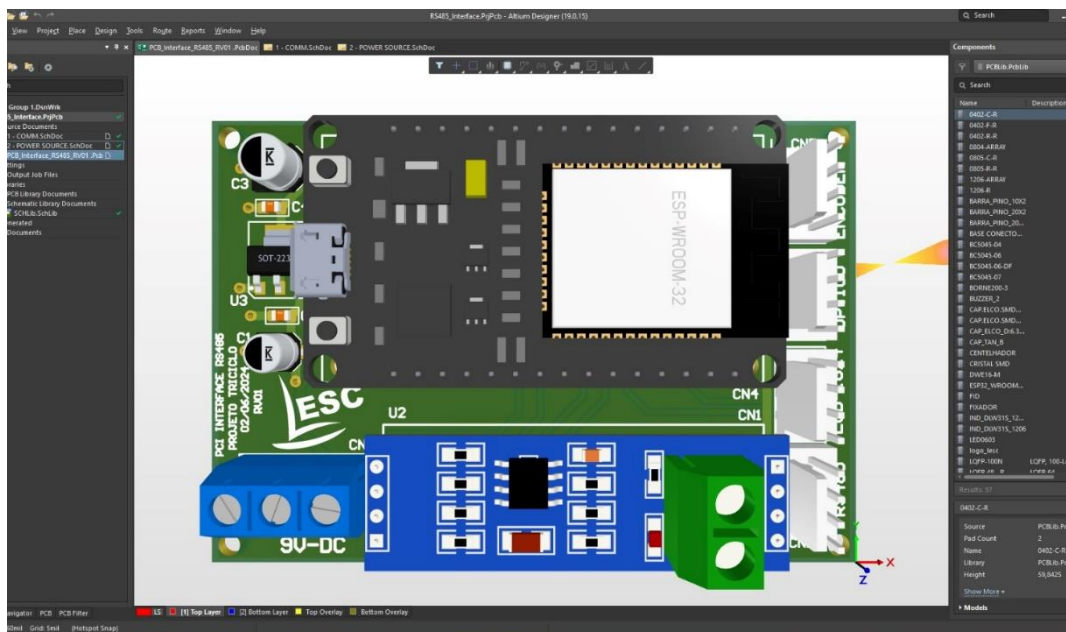
Fonte: Elaborada pelo Próprio Autor

4.2 Modelagem e Esquema elétrico do Sistema Embarcado

- **Modelagem**

Com o intuito de criar um dispositivo compacto e fácil de transportar e realizar as conexões do RS485 e da fonte, foi feito a modelagem de uma pcb para acomodar o Esp32 e o módulo conversor TTL para RS485, fazendo as conexões dos circuitos conforme o esquema elétrico. Dessa maneira, o posicionamento dos componentes da pcb podem ser vistas de acordo com a imagem da Figura 4.3. Na imagem, mostram-se o Esp32, módulo conversor, plug de encaixa da fonte e plugs de encaixe das conexões do RS485.

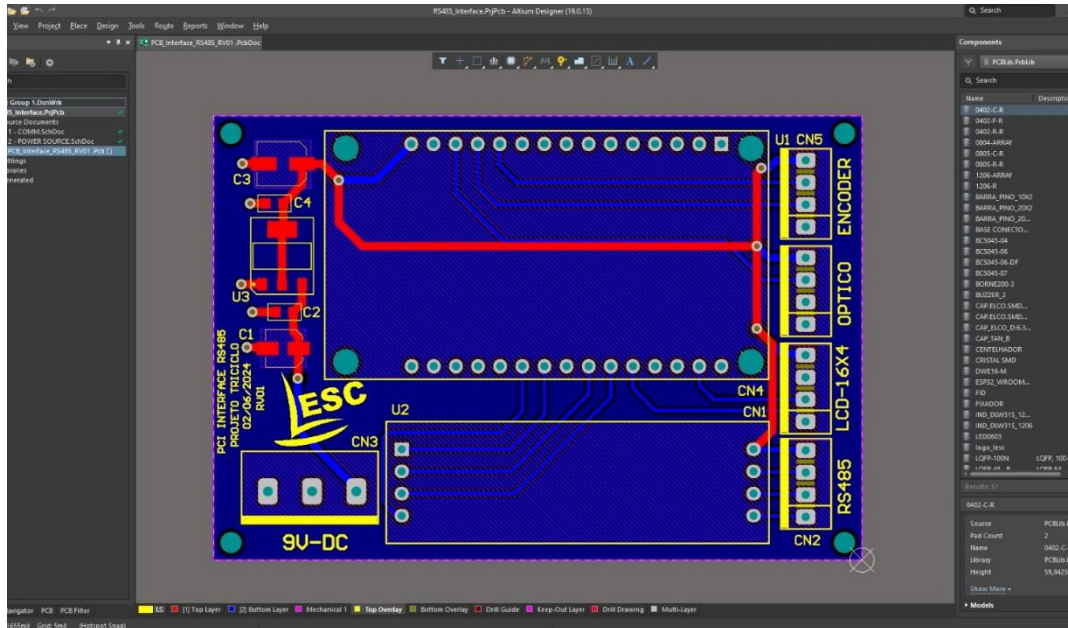
Figura 4.3: Encaixe dos componentes



Fonte: Elaborada pelo próprio autor

Nesta imagem da Figura 4.4, mostra-se a pcb sem os componentes. Na imagem é possível ver a litografia, demonstrando onde vai ser encaixado cada componente e também é possível ver as trilhas das conexões.

Figura 4.4: Vista de cima do PCB feita no Altium

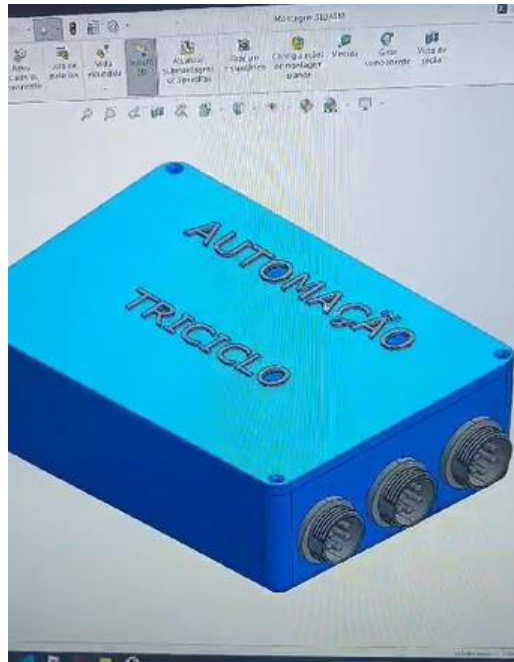


Fonte: Elaborada pelo próprio autor

- **Modelagem da caixa do sistema de automação**

Além da modelagem do pcb, foi feita a modelagem da caixinha em que foi acoplado o pcb junto com os componentes da automação. A caixinha foi modelada no software SolidWorks, permitindo a entrada da fonte, e do RS485. Na Figura 4.5, temos a modelagem completa da caixinha feita no SolidWorks. Na Figura 4.6, temos a caixinha impressa na impressora 3D.

Figura 4.5: Modelagem SolidWorks



Fonte: Elaborada pelo próprio autor

Figura 4.6: Caixa PCB impressa.



Fonte: Elaborada pelo próprio autor

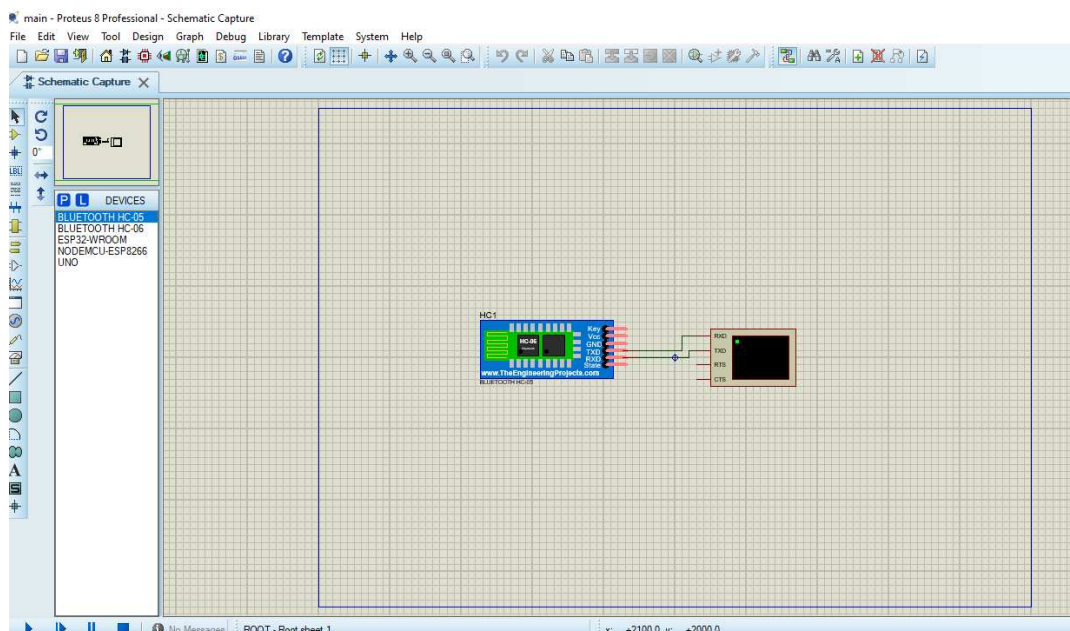
4.3 Comunicação

- **Simulação de comunicação bluetooth**

Foram realizadas simulações no software Proteus utilizando um módulo bluetooth HC-05 para simular a comunicação de envio de dados pelo bluetooth. Nas imagens das Figuras 4.7 e 4.8 estão representados o esquema elétrico do circuito e o

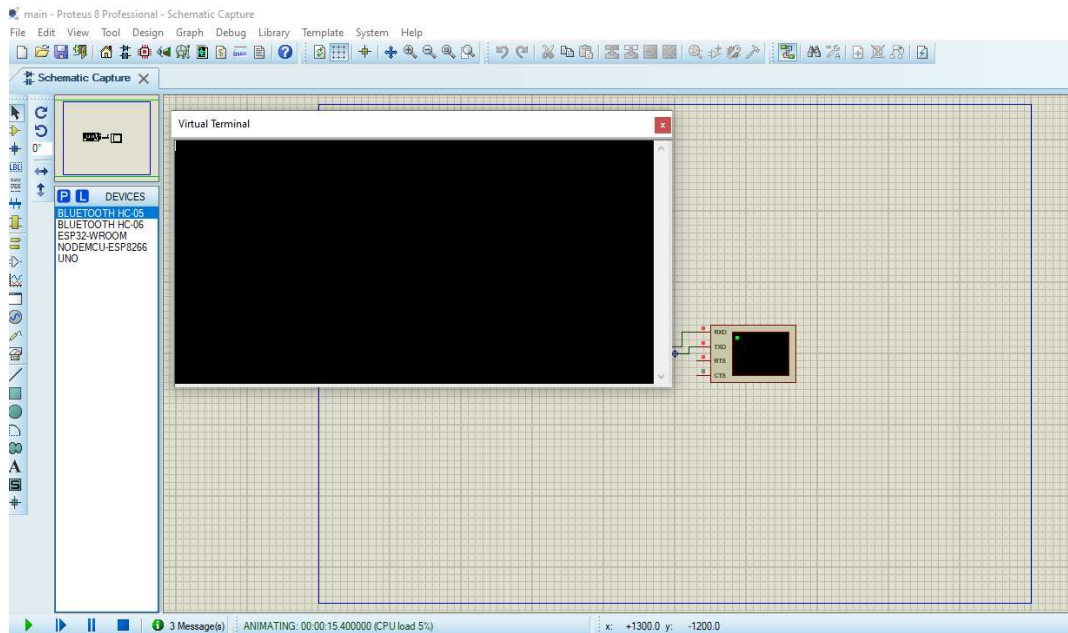
terminal utilizado para visualizar os comandos enviados pelo aplicativo, respectivamente.

Figura 4.7: Esquema de ligação da Simulação Proteus



Fonte: Elaborada pelo próprio autor

Figura 4.8: Realização da simulação no Proteus



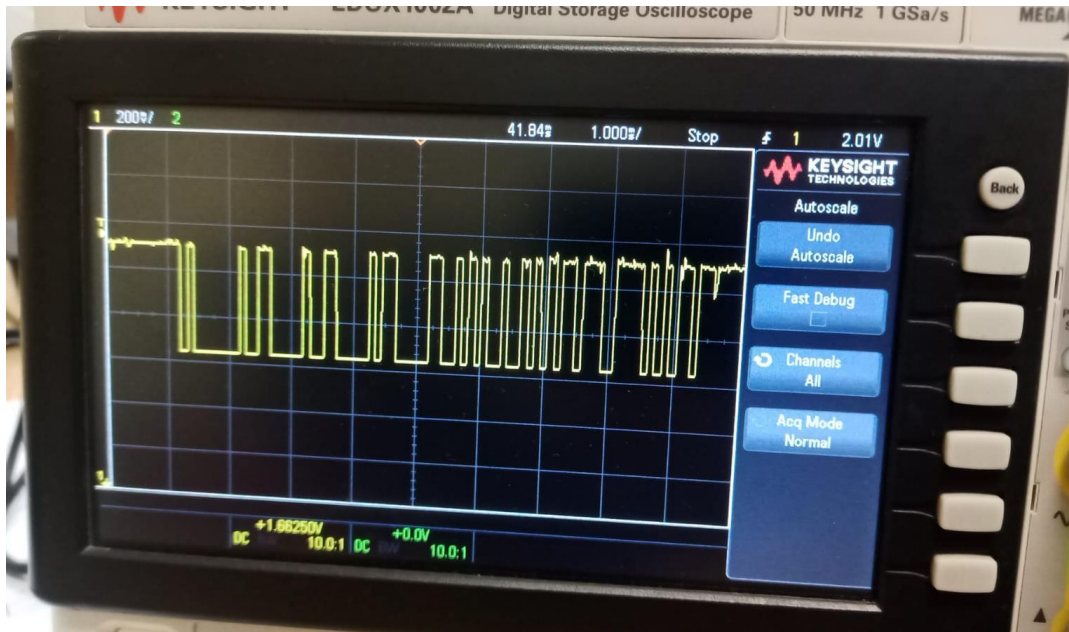
Fonte: Elaborada pelo próprio autor

- **Comunicação entre Esp32 e Inversor de Frequência**

Na comunicação entre o inversor e o Esp32, foi utilizado o módulo CRS485 conectado no inversor de frequência e o módulo conversor TTL para RS485 foi conectado Esp32. Após as conexões, foram feitos testes com um osciloscópio para testar o tráfego dos dados no padrão RS485.

Na Figura 4.9 podemos ver os dados sendo trafegado na rede, os pulsos representam os bits que estão sendo enviados do ESP32 para o inversor de frequência em formato hexadecimal.

Figura 4.9: Osciloscópio

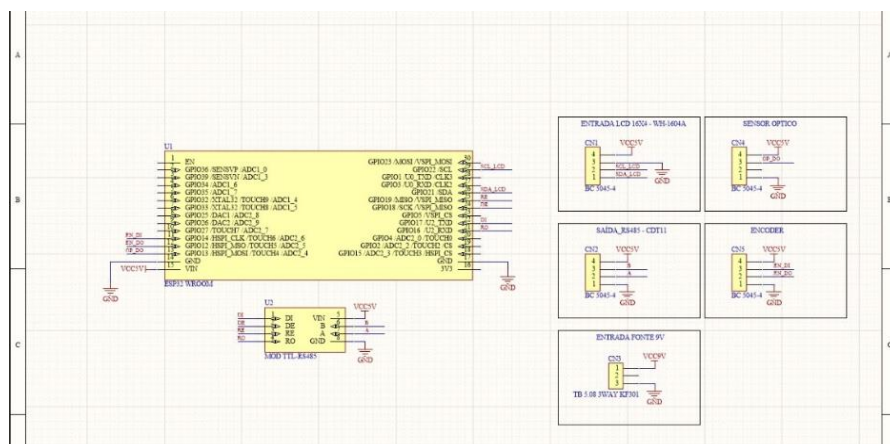


Fonte: Elaborada pelo próprio autor

- **Esquema Elétrico**

No esquema da Figura 4.10, podemos ver o esquema elétrico da pcb feito no Altium para as conexões do Esp32 com o conversor TTL para RS485, fonte de alimentação de 9V e LCD 16x4.

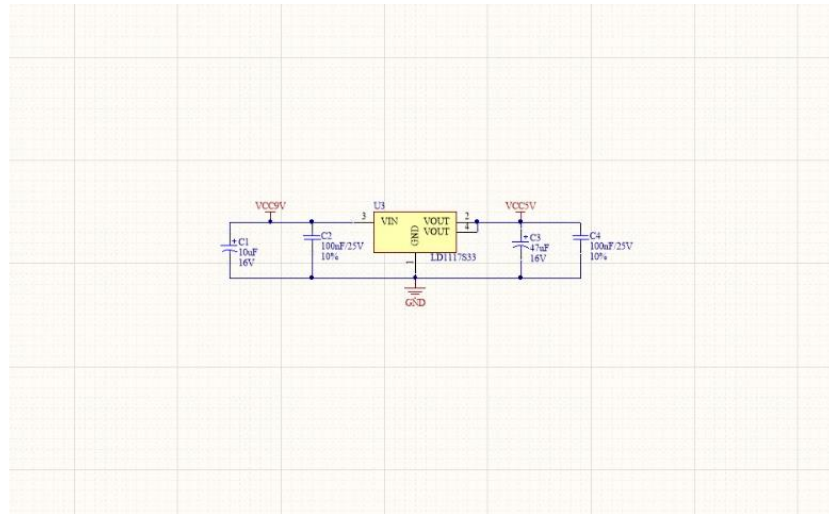
Figura 4.10: Esquema elétrico PCB – Esp32 e portas de comunicação e alimentação



Fonte: Elaborada pelo próprio autor

No esquema da Figura 4.11, podemos ver as conexões dos capacitores com o regulador de tensão que alimenta o Esp32 na porta Vin, assim convertendo a tensão de 9V da fonte de alimentação para saída de 5V.

Figura 4.11: Esquema elétrico PCB – Regulador de Tensão



Fonte: Elaborada pelo próprio autor

4.4 Desenvolvimento do Firmware

Para o desenvolvimento do Firmware foi utilizado a linguagem C++ junto com a IDE Visual Studio Code. A IDE possui uma extensão chamada de *PlatformIO* que oferece um ambiente de desenvolvimento voltado para o desenvolvimento de software embarcado. O primeiro arquivo principal criado foi o chamado de “*main.cpp*”. Foi importado a biblioteca “*BluetoothSerial.h*” para realizar a implementação do bluetooth no Esp32. Outra biblioteca utilizada, foi o *ModbusMaster.h* para criar uma instância do protocolo Modbus na rede RS-485. No código, foi adicionado duas funções importantes chamados de *preTransmission* e *postTransmission* em que ambas servem para funcionar conjunto com o módulo conversor TTL para RS485.

Na Figura 4.12, mostra a implementação das funções.

Figura 4.12: Código para recebimento e leitura de dados

```
void preTransmission()  
{  
    digitalWrite(MAX485_DE, 1);  
    digitalWrite(MAX485_RE, 1);  
}  
  
void postTransmission()  
{  
    digitalWrite(MAX485_DE, 0);  
    digitalWrite(MAX485_RE, 0);  
}
```

Fonte: Elaborada pelo próprio autor

O *preTransmission* faz um envio de um sinal digital de nível lógico alto para ativar as portas RE e DE do módulo e permitir o envio dos dados para o inversor, enquanto a função *postTransmission* envia um sinal de nível lógico baixo para permitir o recebimento dos dados.

Para este projeto, foram utilizadas as funções 03, 06 e 16 do protocolo Modbus que são para leitura e escrita para um registrador e para múltiplos registradores. Na Figura 4.13 temos o código para a escrita de um único registrador do tipo *holding*, na Figura 4.14 temos o código para escrita de múltiplos registradores do tipo *holding* e na Figura 4.15 temos a implementação do código para leitura em um único registrador do tipo *holding*.

Figura 4.13: Código para escrita em um único registrador.

```

void WriteSingleRegister(uint16_t REG, uint16_t value, String s)
{
  lcd.clear();
  lcd.print(s);

  uint8_t result;

  node.begin(ID_INVERSOR, Serial1);
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);
  result = node.writeSingleRegister(REG, value);
  delay(100);

  if (result != node.ku8MBSuccess)
  {
    Error(REG);
  }
}

```

Fonte: Elaborada pelo próprio autor.

Figura 4.14: Código para escrita de múltiplos registradores.

```

void Write_Multiple_Register(char addr, uint16_t REG)
{
  uint8_t result;
  uint16_t data[2];

  node.begin(ID_INVERSOR, Serial1);
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);

  result = node.writeMultipleRegisters(REG, 2);
  data[0] = 3;
  data[1] = 4096;

  node.setTransmitBuffer(0, data[0]);
  node.setTransmitBuffer(1, data[1]);

  delay(100);

  Serial.print(data[1]);

  if (result != node.ku8MBSuccess)
  {
    Error(REG);
  }
}

```

Fonte: Elaborada pelo próprio autor

Figura 4.15: Código para leitura de um único registrador.

```

float Read_Holding_Register(char addr, uint16_t REG)
{
    float i = 0;
    uint8_t result, j;

    uint16_t data[2];
    uint32_t value = 0;
    node.begin(ID_INVERSOR, Serial1);
    node.preTransmission(preTransmission);
    node.postTransmission(postTransmission);

    result = node.readHoldingRegisters(REG, 2);
    delay(500);

    if (result == node.ku8MBSuccess)
    {
        for (j = 0; j < 2; j++)
        {
            data[j] = (node.getResponseBuffer(j));
        }
        Serial.print(data[1], HEX);
        Serial.println(data[0], HEX);

        value = data[0];
        value = value << 16;
        value += data[1];

        return i;
    }
    else
    {
        Serial.print("Falha na Conexão modbus. REG >>> ");
        Serial.println(REG);
        delay(1000);
        return 0;
    }
}

```

Fonte: Elaborada pelo próprio autor.

Embora o código da Figura 4.15 não tenha sido utilizado, ele foi implementado em caso de uso futuro.

Além disso, foi criado um arquivo externo chamado de “REG_CFW100.h” para criar as macros que definem os parâmetros dos registradores que serão utilizados para leitura, configuração, controle e identificação do inversor de frequência. Para facilitar o endereçamento dos dados foram feitos com offset igual a zero, que significa que o número do parâmetro equivale ao endereço do registrador. A Figura 4.16 apresenta uma ilustração do esquema de endereçamento dos parâmetros do tipo holding.

Figura 4.16: Mapa de memória para o protocolo Modbus RTU

Parâmetro	Endereço do dado Modbus	
	Decimal	Hexadecimal
P000	0	0000h
P001	1	0001h
⋮	⋮	⋮
P100	100	0064h
⋮	⋮	⋮

Fonte: (*Manual WEG*).

Os registradores que foram utilizados podem ser vistos no Apêndice – A da dissertação. Na Figura 4.17 está a implementação em código dos endereços dos registradores no arquivo “*REG_CFW100.h*”.

Figura 4.17: Definição das macros dos registradores

```
#define ID_INVERSOR 1
#define TOTAL_DE_REG 2

// Registradores de controle e referência de velocidade
#define REG_CTRL 682
#define REG_VELOCIDADE 683

// Registradores de Comando
#define REG_CONFIG_220 220
#define REG_CONFIG_222 222
#define REG_CONFIG_226 226
#define REG_CONFIG_227 227
#define REG_CONFIG_308 308
#define REG_CONFIG_310 310
#define REG_CONFIG_263 263
#define REG_CONFIG_264 264
#define REG_CONFIG_265 265
#define REG_CONFIG_266 266

// Registradores de Configuração do motor
#define REG_CONFIG_120 120
#define REG_CONFIG_399 399
#define REG_CONFIG_400 400
#define REG_CONFIG_401 401
#define REG_CONFIG_402 402
#define REG_CONFIG_403 403
#define REG_CONFIG_404 404
#define REG_CONFIG_407 407
#define REG_CONFIG_134 134

// Registradores de leitura
#define REG_V 002
#define REG_A 003
```

Fonte: Elaborada pelo próprio autor.

- **Controle do Motor**

O parâmetro de referência da rotação do motor é o P683 – Referência de Rotação via serial. Ele recebe a referência em uma resolução de 13 *bits*. Considera-se a seguinte relação para obter a rotação na resolução de 13 bits:

$$\text{Referência em 13 bits} = \frac{\text{Rotação(Hz)} \times 8192}{60} \quad (4.1)$$

O parâmetro P682 foi utilizado para realizar os comandos de *Stop*, *Start* e *Reversão* do motor. Na Figura 4.18 pode ser observado as características do parâmetro P682.

Figura 4.18: Funções dos bits para o parâmetro P682

Bits	Valores
Bit 0 Gira/Para	0: Para motor por rampa de desaceleração. 1: Gira motor de acordo com a rampa de aceleração até atingir o valor da referência de velocidade.
Bit 1 Habilita Geral	0: Desabilita geral o drive, interrompendo a alimentação para o motor. 1: Habilita geral o drive, permitindo a operação do motor.
Bit 2 Sentido de Giro	0: Sentido de giro do motor oposto ao da referência (sentido reverso). 1: Sentido de giro do motor igual ao da referência (sentido direto).
Bit 3 JOG	0: Desabilita a função JOG. 1: Habilita a função JOG.
Bit 4 LOC/REM	0: Drive vai para o modo local. 1: Drive vai para o modo remoto.
Bit 5 Utiliza Segunda Rampa	0: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da primeira rampa, programada nos parâmetros P100 e P101. 1: Drive utiliza como rampa de aceleração e desaceleração do motor os tempos da segunda rampa, programada nos parâmetros P102 e P103.
Bit 6	Reservado.
Bit 7 Reset de Falhas	0: Sem função. 1: Se em estado de falha, executa o reset do drive.
Bits 8 a 15	Reservado.

Fonte: WEG

O parâmetro recebe como dado uma palavra de 16 bits, sendo que cada bit é um comando. Temos como exemplo o código na Figura 4.19 do comando de *Start* do motor.

Figura 4.19: Código do comando Start

```
void Start()
{
    tricicloAssistido.frequencia = 3;
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1], tricicloAssistido.frequencia);
    if (tricicloAssistido.sentidoGiro)
    {
        WriteSingleRegister(REG_ADDR_WRITE[0], 0x0007, "Start");
    }
    else
    {
        WriteSingleRegister(REG_ADDR_WRITE[0], 0x0003, "Start Reversao");
    }
}
```

Fonte: Elaborada pelo próprio autor

No código vemos a primeira chamada da função *WriteSingleRegister* que recebe como parâmetro um hexadecimal “0x0007”. Este hexadecimal convertendo em bits será “0000000000000111”, dessa forma de acordo com a tabela estaremos passando para os Bits 0, 1 e 2 o valor 1, assim de acordo com a tabela o motor irá girar conforme a rampa de aceleração, o drive será habilitado e o motor irá girar no sentido direto.

- **Funcionalidade de Controle de número de Voltas do motor**

Esta funcionalidade está associada com a tela de *Voltas* do aplicativo desenvolvido. Resumidamente, essa funcionalidade recebe pelo usuário o número de voltas que ele quer que o pedal do triciclo gire. Devido a utilização de engrenagens e correntes para transmitir a rotação do motor para os pedais do triciclo, foi necessário utilizar uma equação que relaciona o tempo necessário para concluir o número de voltas com o tempo da primeira volta, o tempo das voltas restantes e o número de voltas. A equação pode ser vista a seguir:

$$\textit{Tempo Total} = (TPV) + (NV - 1) \times (TVR) \quad (4.2)$$

O TPV e TVR, foram constantes definidas em Ms (Milissegundos) através de medições utilizando um cronômetro de celular com base em um valor de referência do pedal. Assim, os valores definidos foram:

$$TPV = 4840 \text{ Ms}$$

$$TVR = 4340 \text{ Ms}$$

Dessa forma a equação final ficou como:

$$\textit{Tempo Total} = 4840 + (NV - 1) \times 4340 \quad (4.3)$$

4.5 Interface Gráfica

Para desenvolvimento da interface gráfica, utilizou-se a linguagem Dart que foi criada para o framework Flutter. A seguir serão apresentadas as telas desenvolvidas e as suas funcionalidades. Por fim, no Apêndice B é possível analisar os registradores do inversor utilizado durante o desenvolvimento da aplicação.

- **Tela de Home**

Na tela Home que pode ser visto na Figura 4.20, cada botão ao ser pressionado enviará dados no formato *String* para o Esp32 através do protocolo Bluetooth. Esses dados variam conforme cada botão foi programado para enviar. Cada rótulo do botão indica sua ação característica, como o botão INICIAR inicia a rotação do motor e o botão EMERGÊNCIA faz a parada de emergência do motor. O botão ENVIAR realiza o envio do valor da frequência do motor, o valor da frequência é somente alterado após o pressionamento dos botões “+” ou “-”, sendo 3.0 o valor mínimo e 15.0 seu valor máximo, e logo em seguida o pressionamento do botão ENVIAR. O botão SWITCH da cor azul executa a função de alternar o controle do motor entre o quadro do sistema ou pelo aplicativo. Ao ser alterado o aplicativo envia os dados que permite a condição de alteração de modo de controle e logo em seguida o Esp32 envia todos os dados de configuração para os registradores do inversor para permitir essa troca de controle. Devido ao tempo de gravação dos novos valores dos registradores, demora cerca de alguns segundos para realizar a alternância entre os modos e estar pronto para operação. O botão do tipo “slide” será uma implementação que fará o motor girar em um setor específico, de modo que o usuário possa variar a área conforme desejado.

Figura 4.20: Tela Home

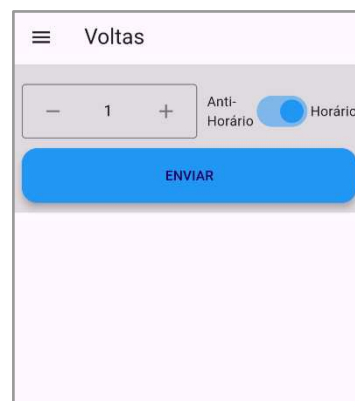


Fonte: Elaborada pelo Próprio Autor

- **Tela de Voltas**

Na tela *Voltas* que está na Figura 4.21, existirão três tipos de botões sendo um contador de voltas representado pelos ícones “+” e “-”, um botão *SWITCH* que será responsável pelo sentido de rotação das voltas e um botão convencional na cor azul com rótulo *ENVIAR*. Quando escolhido o valor de voltas e o sentido de rotação, ao pressionar o botão enviar os dados são enviados para o Esp32. No Firmware do Esp32 existe uma rotina que através de uma expressão matemática calcula o tempo de rotação necessário para um número de voltas específicas na frequência de rotação em 3 Hz.

Figura 4.21: Tela de Voltas



Fonte: Elaborada pelo Próprio Autor

- **Tela de Hit**

Nesta tela representada pela Figura 4.22, haverá etapas de treino para o usuário. Cada etapa é dividida por um tempo fixo, sendo assim alterando a frequência de rotação do motor de acordo com as especificações. Exemplo: No Status existe o Aquecimento leve, nessa etapa a rotação será de 4Hz, logo em seguida após 30 segundos o aplicativo altera a etapa para Treino Intenso, assim aumentando a frequência de rotação para 10 Hz. No momento, as etapas dos tempos estão fixas, mas nas implementações futuras poderá ser alterada pelo usuário. Então, ao pressionar o botão iniciar o aplicativo começa a contar o tempo até 1 minuto e 30 segundos. Nesse intervalo de tempo o aplicativo altera os Status de treino entre Aquecimento, Hit e Resfriamento e variando sua frequência de acordo com o Status em 4Hz, 10Hz e 4Hz, respectivamente. O botão *Parar* trava o processo do Hit no tempo determinado e o botão *Resetar* reinicia todo o processo.

Figura 4.22: Tela de Hit

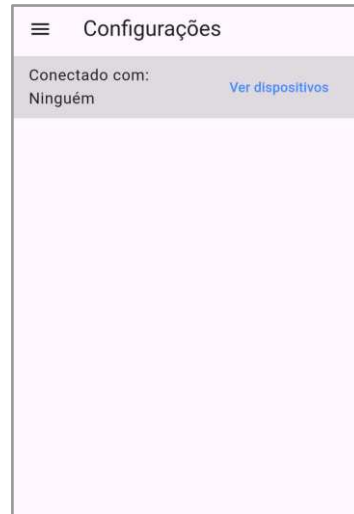


Fonte: Elaborada pelo Próprio Autor

- **Tela de Configuração**

Nesta outra tela ilustrada pela Figura 4.23 serão destinados todos os itens de configuração do aplicativo e do inversor. No momento, existe apenas um item responsável pela conectividade Bluetooth entre o aplicativo e o Esp32. Ao se conectar será possível realizar o controle do motor pelas três telas anteriores.

Figura 4.23: Tela de Configuração

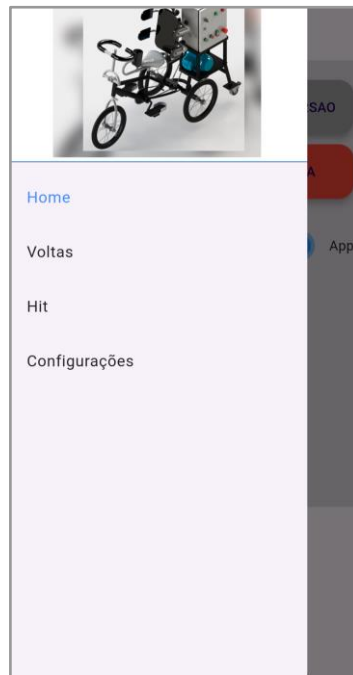


Fonte: Elaborada pelo Próprio Autor

- **Menu Lateral**

Na Figura 4.24, representa a imagem do menu lateral que faz a alternância entre as quatro telas do aplicativo.

Figura 4.24: Tela de Menu



Fonte: Elaborada pelo Próprio Autor

4.6 Testes no Sistema

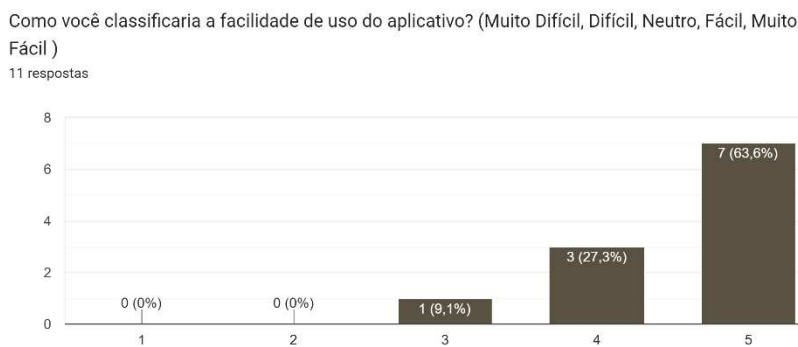
Foi aplicado um questionário de usabilidade com 11 participantes de alunos de bacharelado e mestrado do Centro de Tecnologia da UFC que testaram o sistema. Os principais resultados serão apresentados a seguir.

4.6.1 Análise da Usabilidade do Sistema

- **Facilidade de Uso**

Quando questionado sobre a facilidade de uso do aplicativo, 63,6% dos participantes classificaram como Muito Fácil, 27,3% classificaram como fácil e 9,1% classificaram como neutro. Sobre a dificuldade de conectar o aplicativo ao triciclo, 90,9% dos participantes responderam que não tiveram dificuldades. Quando questionando sobre responder corretamente ao pressionar os botões, 45,5% dos participantes responderam como Sempre, 45,5% responderam como frequentemente e 9,1% respondeu às vezes. Nos Gráficos 4.1, 4.2 e 4.3 podemos ver os valores exatos da pesquisa.

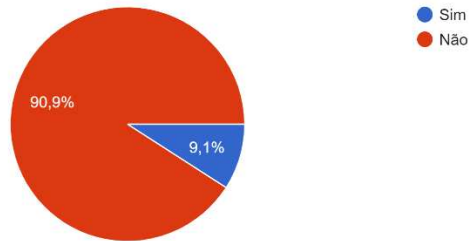
Gráfico 4.1 – Classificação da facilidade de uso pelo aplicativo



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.2 – Dificuldade de conexão do aplicativo com o triciclo pelos usuários

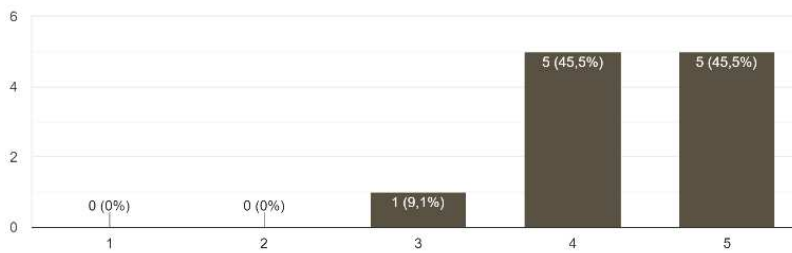
Você encontrou alguma dificuldade ao conectar o aplicativo ao triciclo?
11 respostas



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.3 – Classificação ao clicar no aplicativo se caso responder corretamente

Ao clicar em um dos botões o aplicativo respondeu corretamente? (Nunca, Raramente, Às vezes, Frequentemente, Sempre)
11 respostas



Fonte: Dados da pesquisa, gerados via Google Forms

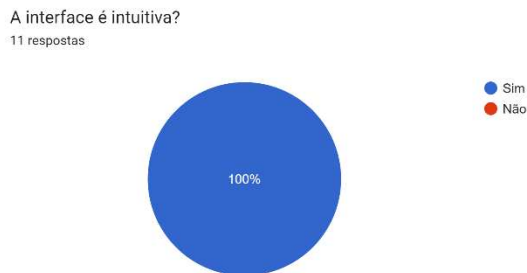
Esses resultados indicam que o aplicativo foi considerado pela maior parte dos participantes como intuitivo e fácil de manusear, além disso o aplicativo funcionou corretamente na maior parte do tempo.

- **Interface**

Quando questionado sobre a interface do aplicativo, 100% responderam que a interface é intuitiva e 100% dos participantes responderam que as informações do aplicativo foram claras e compreensíveis. No entanto, 90,9% dos participantes não

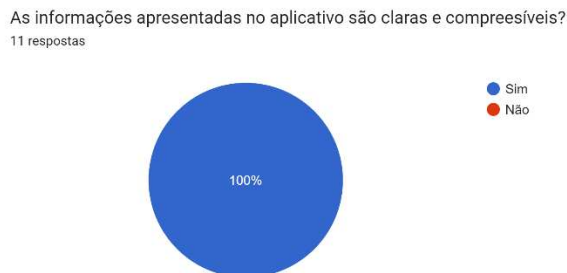
tiveram dificuldade em entender as funções do aplicativo. Nos Gráficos 4.4, 4.5 e 4.6 temos os resultados da pesquisa.

Gráfico 4.4 – Intuitividade da interface do aplicativo segundo os usuários



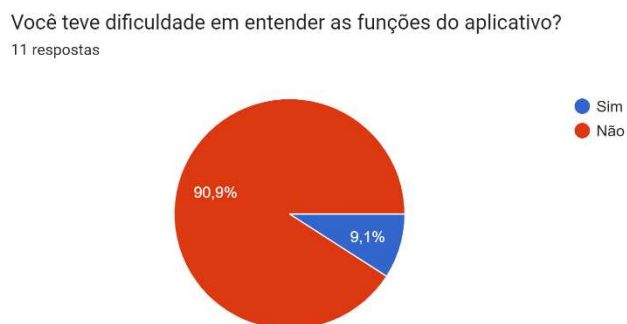
Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.5 – Clareza e compreensibilidade das informações no aplicativo segundo os usuários



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.6 – Dificuldade no entendimento do aplicativo segundo os usuários



Fonte: Dados da pesquisa, gerados via Google Forms

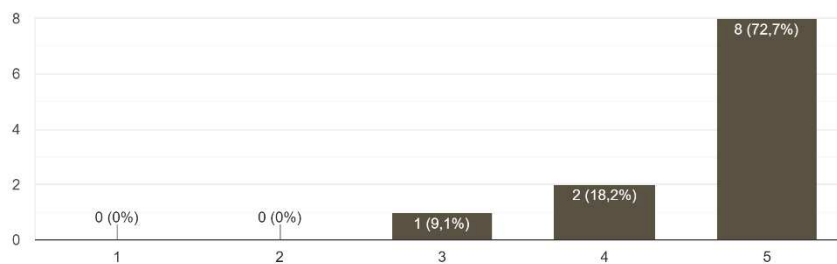
Pelos dados indica uma avaliação majoritária dos participantes de que a interface é intuitiva e com as informações claras e compreensíveis. Entretanto, 9,1% dos participantes tiveram dificuldades em entender as funções do aplicativo. Um participante relatou que: “Os padrões de cores poderiam melhorar”.

- **Desempenho e Resposta do Sistema**

Com relação ao tempo de resposta dos comandos do aplicativo, 72,7% dos participantes classificaram como satisfatório, 18,2% classificaram como pouco satisfatório, enquanto 9,1% dos participantes classificaram como neutro. Sobre o aplicativo apresentar falhas ou travamentos durante o uso, 100% dos participantes responderam que não houve falhas ou travamentos. Nos Gráficos 4.7 e 4.9 temos os dados exatos apurados na pesquisa.

Gráfico 4.7 – Satisfação no tempo de resposta do aplicativo segundos os usuários

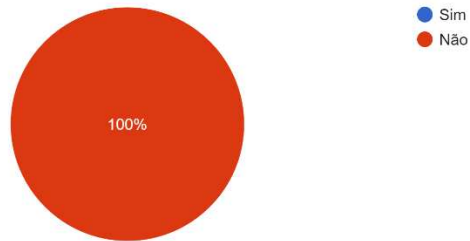
O tempo de resposta dos comandos do aplicativo foram satisfatórios? (Insatisfatório, Pouco Insatisfatório, Neutro, Pouco Satisfatório, Satisfatório)
11 respostas



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.8 – Falhas ou travamentos durante o uso

O aplicativo apresentou falhas ou travamentos durante o uso?
11 respostas



Fonte: Dados da pesquisa, gerados via Google Forms

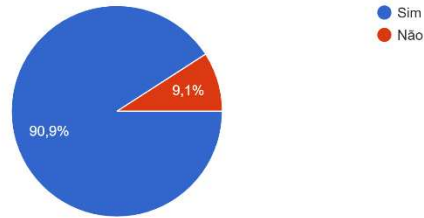
Com esses resultados demonstra que o sistema tem tempo de resposta ideal e condiz com os objetivos do projeto. Além disso, não houve falhas e travamentos no controle do sistema.

- **Segurança e Confiabilidade**

Com relação a segurança ao controlar o sistema, 90,9% dos participantes classificou que sentiu segurança, enquanto 9,1% classificou que não sentia segurança. Quando perguntado se houve alguma situação em que o aplicativo não respondeu como esperado, 81,8% dos participantes classificou que não ocorreram alguma situação que o aplicativo não respondeu como esperado. Nos Gráficos 4.9 e 4.10 podemos ver os dados exatos da pesquisa para este quesito.

Gráfico 4.9 – Segurança que os usuários sentiram ao controlar o triciclo

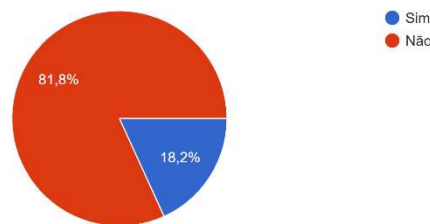
Você sentiu segurança ao usar o sistema para controlar o triciclo?
11 respostas



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.10 – Situações em que o aplicativo não respondeu como esperado segundo os usuários

Houve alguma situação que o aplicativo não respondeu como esperado?
11 respostas



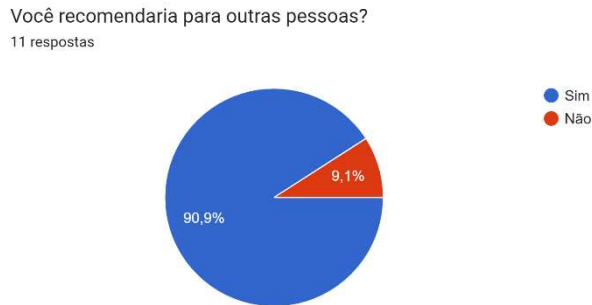
Fonte: Dados da pesquisa, gerados via Google Forms

Com os resultados majoritários de sentiu segurança ao usar o sistema e que não houve alguma situação em que o aplicativo não respondeu como esperado, isso indica que a segurança do sistema é alta. Um participante relatou que: “Houve um momento que ao aumentar a velocidade parecia ter diminuído”.

- **Satisfação Geral**

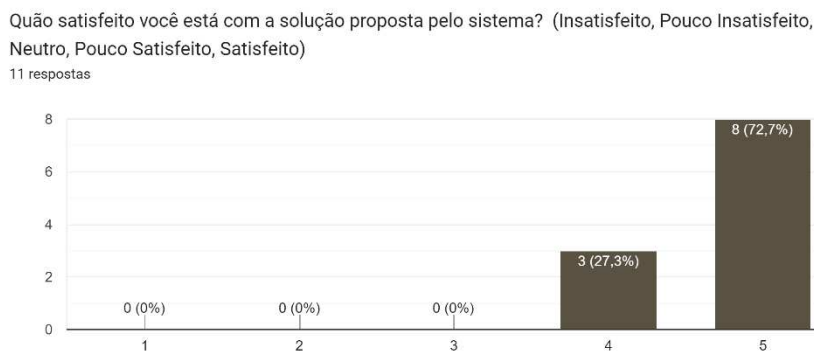
Quando perguntados sobre a satisfação geral do sistema, 72,2% responderam que estão satisfeitos e 27,3% responderam que estão pouco satisfeitos. Além disso, 90,9% dos participantes responderam que recomendariam para outras pessoas. Nos Gráficos 4.11 e 4.12 temos os valores ilustrado em forma de gráfico dos dados apurados.

Gráfico 4.11 – Recomendação dos usuários



Fonte: Dados da pesquisa, gerados via Google Forms

Gráfico 4.12 – Classificação da satisfação da solução proposta pelo sistema segundo os usuários



Fonte: Dados da pesquisa, gerados via Google Forms

Com os resultados dos dados, dessa forma corrobora uma avaliação positiva ao sistema desenvolvido, assim reforçando sua aplicabilidade em um contexto real.

- **Sugestões**

Quando perguntado aos participantes sobre sugestão de melhoria, tivemos alguns relatos. Um deles foi que *“Poderia atualizar a velocidade apenas ao realizar sua mudança, e não só depois de clicar no botão de enviar”*, e outro participante relatou que *“Centralizar os ícones do app. Retirar a função de conexão da tela de início e coloca-la na tela de configuração”*.

Com esses relatos, podemos considerar que existe espaço pra mais melhorias do projeto, como melhorar a interface, realinhar os elementos das telas, e reprogramar os comandos dos botões.

Quando perguntado sobre algum comentário adicional um respondeu que *“O Equipamento é profissional, que trará muitos benefícios nos tratamentos de pacientes com mobilidade. O projeto deve ser expandido para adultos também, pensando em escalabilidade”*, outro considerou que *“Acho uma ótima ideia de produto e recomendaria para clínicas de fisioterapia*, outro disse que *“Gostei muito do projeto e da iniciativa, creio que ele vai auxiliar muito no desenvolvimento da mobilidade das crianças em fisioterapia”*, e por fim um considerou *“Proposta inovadora e com ampla aplicação”*. Isso sugere que o projeto tem aceitação positiva entre os participantes, porém há espaço para melhorias e ampliar a aplicação em outras pessoas de diferentes faixas etárias além de crianças.

4.1 Discussão dos Resultados

Os dados coletados mostram que a maioria dos usuários considera o sistema fácil de usar e com interface intuitiva. Além disso, os dados demonstram que o sistema responde rapidamente e com alta segurança. No entanto, algumas funções do aplicativo não ficaram com fácil compreensão, assim sugere melhorias nas informações dispostas no aplicativo e textos presentes em cada tela do aplicativo para facilitar o entendimento das funcionalidades.

Além disso, a análise qualitativa dos comentários sugere ampliar o uso do triciclo para adultos, dessa forma incluindo mais grupos para reabilitação além de crianças e melhorias na interface e na funcionalidade do controle de velocidade do aplicativo.

5 CONCLUSÃO

O presente projeto descreveu o desenvolvimento de um sistema de controle de um triciclo assistido estático para reabilitação. Para isso, foram utilizados componentes, destacando o uso, como microcontrolador ESP32 para gerenciar a comunicação entre o inversor de frequências CFW100 e o aplicativo móvel via protocolo Modbus RTU sobre RS485. O uso desses componentes foi importante por serem confiáveis e de baixo custo e fácil implementação, o que reforça o potencial de replicabilidade e escalabilidade da solução em diferentes contextos de reabilitação. Além disso, o uso da linguagem de programação C++ para o desenvolvimento do firmware do ESP32 garantiu a robustez e eficiência do código, além de permitir a fácil implementação de novas funcionalidades no futuro.

O sistema desenvolvido permitiu que o triciclo fosse controlado remotamente por meio de um dispositivo móvel, com a capacidade de ajustar a velocidade de rotação, enviar comandos de *Start* e *Stop* e definir número de voltas dos pedais. As funcionalidades foram implementadas para serem intuitivas. O aplicativo foi desenvolvido pelo framework Flutter que permitiu a criação de uma interface multiplataforma, que pode ser executada tanto em dispositivos Android quanto iOS, proporcionando maior flexibilidade e alcance do sistema. Durante o processo de design da interface, foi priorizada a usabilidade, com o intuito de garantir que o sistema pudesse ser operado facilmente por pessoas com diferentes níveis de familiaridade com a tecnologia. Os testes de usabilidade confirmaram que a interface do aplicativo foi bem avaliada pelos participantes, sendo considerada intuitiva e de fácil manuseio.

Os testes realizados com 11 participantes revelaram resultados bastante satisfatórios em termos de funcionalidade e eficiência do sistema. A maioria dos usuários relatou que não encontrou dificuldades para operar o aplicativo ou conectar o sistema ao triciclo. Além disso, os comandos enviados pelo aplicativo para o controle do motor trifásico do triciclo foram processados de maneira rápida e precisa, sem a ocorrência de falhas significativas. A capacidade de controle da velocidade, direção e número de voltas do pedal demonstrou o sucesso da integração entre o hardware e o software, validando a eficácia do sistema.

A confiabilidade do sistema também foi um dos pontos fortes do projeto. Durante os testes, não foram observados travamentos ou interrupções no funcionamento do sistema, o que é essencial para garantir a segurança dos pacientes

durante o processo de reabilitação. Além disso, o sistema foi projetado com mecanismos de segurança, como o botão de emergência no aplicativo, que permite a interrupção imediata do funcionamento do motor em caso de necessidade, oferecendo tranquilidade aos profissionais de saúde que operam o equipamento.

Do ponto de vista técnico, o sistema desenvolvido atingiu aos requisitos estabelecidos, com os resultados demonstram que o uso do aplicativo móvel é viável e eficiente. Além disso, indicam que o sistema é de fácil uso com uma interface intuitiva, tem alta performance e contém um bom nível de segurança.

Ainda, o presente projeto foi multidisciplinar, agregando conhecimentos da área de automação, sistemas embarcados e controle para uma aplicação de cunho fisioterapêutico em um laboratório de estudos dos cursos de Engenharia de Computação.

Em resumo, este projeto atingiu seus principais objetivos, entregando uma solução tecnológica eficiente, segura e acessível para a reabilitação de membros inferiores. O sistema desenvolvido oferece uma plataforma inovadora combinando o controle preciso de um equipamento de reabilitação com a conveniência de um aplicativo móvel. A combinação de automação, tecnologia assistiva e sistemas embarcados apresentada neste trabalho é um exemplo claro de como a engenharia pode transformar positivamente o campo da saúde.

5.1 Trabalhos Futuros

O projeto foi desenvolvido para ser escalável e passível de agregar funcionalidades futuras, assim o projeto pode ser expandido para adultos e outras faixas etárias. Logo, se tratando de trabalhos futuros, o aplicativo de controle poderá utilizar motores com menor CV e de corrente contínua. Além disso, pode-se utilizar baterias, tornando o sistema independente da rede elétrica durante o funcionamento. Possibilita também a utilização de IoT para armazenar dados em um banco de dados. O uso de sensores também pode agregar valor ao sistema, permitindo um monitoramento mais detalhado do desempenho durante o exercício.

Destaca-se ainda como melhorias futuras ao trabalho, a otimização dos códigos, utilizando código Assembly em algumas funções, e melhorias constantes na interface, em especial dos botões, objetivando atender as funcionalidades que serão aplicadas. Além disso, pode ser implementado informações nas telas explicando as funcionalidades presentes.

REFERÊNCIAS

Zamboni, Lincoln César; Pamboukian, Sergio Vicente D.; Barros, Edson A. R.; *C++ para Universitários*. 2. ed. São Paulo: Páginas & Letras, 2006.

Referência: MODBUS. Disponível em: <https://www.modbus.org>. Acesso em: 15 ago. 2024.

Groover, Mikell P. **Automation, Production Systems, and Computer-Integrated Manufacturing**. Pearson, 2014.

Volti, Rudi. *Society and Technological Change*. Worth Publishers, 2017.

USINAINFO. Conversor Serial TTL para RS485. Imagem. Disponível em: https://www.usinainfo.com.br/1019824-thickbox_default/conversor-serial-ttl-para-rs485.jpg. Acesso em: 21 ago. 2024.

WEG. Motor W22 Plus Premium. Imagem. Disponível em: https://static.weg.net/medias/images/hf0/h00/BRASIL_W22_Plus_Premium_63_132_IE3_B3Dnew_1200Wx1200H.jpg. Acesso em: 21 ago. 2024.

KEYSIGHT. EDUX1002A. Imagem. Disponível em: <https://www.keysight.com/br/pt/product/EDUX1002A/oscilloscope-50-mhz-2-analog-channels.html>. Acesso em: 22 set. 2024.

QuardonComponents. LTC1771IS8. Imagem. Disponível em: <https://www.quardoncomponents.com/shop/wp-content/uploads/LTC1771IS8.jpg>. Acesso em: 22 set. 2024.

AutoCore. Capacitor Eletrolítico 10uF 16V. Imagem. Disponível em: <https://cdn.awsli.com.br/2500x2500/78/78150/produto/69812319/26196f12bc.jpg>. Acesso em: 22 set. 2024.

WINSTAR. Display LCD de 16 caracteres por 4 linhas. Imagem. Disponível em: <https://www.winstar.com.tw/uploads/photos/character-lcd-display-modules/WH1604A-TMI-JT.jpg>. Acesso em: 22 set. 2024.

WEG. *Manual do Usuário Inversor de Frequência CFW100*. Edição 07. Jaraguá do Sul: WEG Equipamentos Elétricos S.A., 2017. 184 p. Multilíngue. (Código: 10005750207).

WEG. *Manual de Programação Inversor de Frequência CFW100.* Edição 10. Jaraguá do Sul: WEG Equipamentos Elétricos S.A., 2019. 100 p.

FINCH, Jacqueline. The ancient origins of prosthetic medicine. *The Lancet*, v. 377, p. 548, 12 fev. 2011.

NETDRINKS. Espumante Esp. 32.235. Disponível em: https://images.tcdn.com.br/img/img_prod/672486/esp_32_235_1_3bfd2b451c966d74d4057de594110c94_20211222202922.jpg. Acesso em: 21 ago. 2024.

GETROTECH. **O que é um inversor de frequência?**: Inversores de frequência. Getrotech, 2017. Acesso em 25 abr. 2019. Disponível em: <www.getrotech.com.br/Artigos/o-que-e-um-inversor-de-frequencia/>.

JUNIOR, M. C. da F. **Desenvolvimento de software para parametrização, monitoramento e supervisão de conversores de frequência.** 2011. 1-80 p. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Federal de Mato Grosso do Sul, Campo Grande, MS, Brasil, 2011.

NASCIMENTO, J. M.; LUCENA, P. B. **Protocolo Modbus.** LECA-DCA-UFRN, 2003. Acesso em 04 mai. 2019. Disponível em:<www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho3/trabalho3_13.pdf>.

Sophie Rey Barth, Nicolas Pinsault, Hugo Terrisse, Claire Eychenne, Carole Rolland, Eloi Gergelé, Alison Foote, Catherine Guyot, Jean-Luc Bosson, **Corrigendum to A program centered on smart electrically assisted bicycle outings for rehabilitation after breast cancer: A pilot study**, Medical Engineering & Physics, Volume 106, 2022, 103776, ISSN 1350-4533, <https://doi.org/10.1016/j.medengphy.2022.103776>. (<https://www.sciencedirect.com/science/article/pii/S1350453322000273>) (<https://www.sciencedirect.com/science/article/pii/S1350453322000273>)

SILVEIRA, Rafael Schimith da; CATELANO, Bruna de Albuquerque; SOARES, Hattos Paulo Mendes; MOLOSSI, Mayara. **Fisioterapia ortopédica e intensivismo.** Revista CPAQV – Centro de Pesquisas Avançadas em Qualidade de Vida, vol. 16, n. 2, 2024.

PERIUS, Tiago Faria. *Geração de diretrizes de projeto com o uso do Design for X para o desenvolvimento de prótese de membro inferior de baixo custo.* 2014. 174 f. Dissertação (Mestrado em Design) – Programa de Pós-Graduação em Design, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2014.

BRASIL. Ministério da Saúde. *Técnico em órteses e próteses: livro-texto.* Brasília: Ministério da Saúde, 2014. 318 p. Disponível em: <http://www.saude.gov.br/bvs>.

SANDISON, Melissa et al. **HandMATE: Wearable Robotic Hand Exoskeleton and Integrated Android App for At Home Stroke Rehabilitation.** 2020 42nd Annual

International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), 2020.

MELVIN, Joy et al. **Effectiveness of low-intensity aerobic exercise with assisted bicycle exercises on quality of life and lung functions in Duchenne muscular dystrophy.** International Journal of Disability, Development and Education, 2020.

CÂNDIDO, Amanda de Melo. **Reabilitação motora para crianças e jovens com distrofia muscular: Revisão de escopo.** Dissertação (Mestrado em Fisioterapia) - Universidade Federal do Rio Grande do Norte, Natal, 2022.

Ricarte, G. L. (2021). **Interface de Controle Portátil para Ciclismo por Estimulação Elétrica Funcional.** Dissertação de Mestrado, Universidade de Brasília.

Z. Meihuan, "**Embedded Design of Human Motion Physiological Parameter Collector,**" 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 2020, pp. 538-541, doi: 10.1109/ICPICS50287.2020.9202089.

CARBONE, Giuseppe; GONÇALVES, Rogério Sales. Editorial: **Robot-assisted rehabilitation for neurological disorders.** *Frontiers in Robotics and AI*, v. 9, 2022. Disponível em: <https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022.1014681>. Acesso em: 29 jun. 2024.

H. Neuber and M. Strube, "**Closed-Loop heartrate Control with Electric Bicycles for Cardiac Rehabilitation,**" 2020 21st International Conference on Research and Education in Mechatronics (REM), Cracow, Poland, 2020, pp. 1-5, doi: 10.1109/REM49740.2020.9313080.

Beeson, P. (2012). "**Joint mobilization for the treatment of osteoarthritis of the knee: a systematic review.**" *Clinical Rehabilitation*, 26(4), 315-326.

Dias, R., & Farias, M. (2018). "**Strength training for knee osteoarthritis: A systematic review.**" *Journal of Physical Therapy Science*, 30(5), 707-712.

Lee, J., & Kim, D. (2017). "**Proprioceptive exercises for patients with knee osteoarthritis: A systematic review.**" *Journal of Exercise Rehabilitation*, 13(4), 371-376.

Patterson, K., & Finestone, H. (2016). "**Gait retraining for stroke patients: a systematic review.**" *Journal of Rehabilitation Research & Development*, 53(1), 1-20.

Lake, D. A. (2013). "**Neuromuscular electrical stimulation in poststroke rehabilitation: systematic review and meta-analysis.**" *European Journal of Physical and Rehabilitation Medicine*, 49(3), 437-444.

Fransen, M., & McConnell, S. (2008). "**Exercise for osteoarthritis of the knee.**" *Cochrane Database of Systematic Reviews*, (4), CD004376.

Adams, D., & Logerstedt, D. (2012). "**Rehabilitation following ACL reconstruction: a systematic review.**" *Journal of Bone and Joint Surgery*, 94(19), 1737-1748.

APÊNDICE A – PARAMETRIZAÇÃO DO INVERSOR DE FREQUÊNCIA

A seguir, descreve-se os parâmetros necessários para que o inversor de frequência atue de forma serial, no modo remoto.

PARÂMETROS BÁSICOS DE CONFIGURAÇÃO

P220 (seleção fonte LOC/REM) - 1

P222 (Sel. Referência REM) - 7

P227 (seleção gira/para REM) – 2

P263 – 1

P264 – 11

P265 – 12

P266 - 5

PARÂMETROS RELACIONADOS A REDE DE COMUNICAÇÃO SERIAL

P308 (endereço serial) - 1

P310 (taxa de comunicação serial) - 1

CONFIGURAÇÕES DO MOTOR

P400 – 220

P401 – 1.77

P402 - 1700 rpm

P403- 60 Hz

P404 - 3

P407 - 0.7

APÊNDICE B – Arquivo main.cpp

```

#include "BluetoothSerial.h"
#include "REG_CFW100.h"
#include <ModbusMaster.h>
#include <LiquidCrystal_I2C.h>

#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

#define MAX485_RE 19 // RE
#define MAX485_DE 18 // DE
#define SERIAL_RX_PIN 16 // RO
#define SERIAL_TX_PIN 17 // DI

struct Automacao_Triciclo
{
    bool isConnected = false;
    bool isTest = false;
    bool isConfig = true;
    bool isReset = false;
    bool sentidoGiro = true;

    int typeConfig = 1;
    float frequencia = 0;
    int nVoltas = 0;

    unsigned long tempoPrimeiraVolta = 4840;
    unsigned long tempoVoltasRestantes = 4340;

    String strCMD = "";
    char c;
};

Automacao_Triciclo tricicloAssistido;

ModbusMaster node;

```

```

BluetoothSerial SerialBT;
LiquidCrystal_I2C lcd(0x27, 16, 4);

TaskHandle_t Task1;

void preTransmission()
{
  digitalWrite(MAX485_DE, 1);
  digitalWrite(MAX485_RE, 1);
}

void postTransmission()
{
  digitalWrite(MAX485_DE, 0);
  digitalWrite(MAX485_RE, 0);
}

void Error(uint16_t REG)
{
  Serial.print("Falha na Conexão modbus. REG >>> ");
  Serial.println(REG);
  delay(1000);
}

float Read_Holding_Register(char addr, uint16_t REG)
{
  float i = 0;
  uint8_t result, j;

  uint16_t data[2];
  uint32_t value = 0;
  node.begin(ID_INVERSOR, Serial1);
  node.preTransmission(preTransmission);
  node.postTransmission(postTransmission);

  result = node.readHoldingRegisters(REG, 2);
  delay(500);

  if (result == node.ku8MBSuccess)
  {
    for (j = 0; j < 2; j++)
    {

```

```

    data[j] = (node.getResponseBuffer(j));
}
Serial.print(data[1], HEX);
Serial.println(data[0], HEX);

value = data[0];
value = value << 16;
value += data[1];

return i;
}
else
{
    Serial.print("Falha na Conexão modbus. REG >>> ");
    Serial.println(REG);
    delay(1000);
    return 0;
}
}

void WriteSingleRegister(uint16_t REG, uint16_t value, String s)
{
    lcd.clear();
    lcd.print(s);

    uint8_t result;

    node.begin(ID_INVERSOR, Serial1);
    node.preTransmission(preTransmission);
    node.postTransmission(postTransmission);
    result = node.writeSingleRegister(REG, value);
    delay(100);

    if (result != node.ku8MBSuccess)
    {
        Error(REG);
    }
}

void WriteSingleRegister2(uint16_t REG, uint16_t value)
{
    float i = 0;

```

```

uint8_t result, j;
uint16_t data[2];

node.begin(ID_INVERSOR, Serial1);
node.preTransmission(preTransmission);
node.postTransmission(postTransmission);
result = node.writeSingleRegister(REG, value);
delay(100);

if (result != node.ku8MBSuccess)
{
    Error(REG);
}
}

void SetVelocity(char addr, uint16_t REG, float vel)
{
    float i = 0;
    uint8_t result, j;
    uint16_t data[2];
    uint32_t value = 0;

    node.begin(ID_INVERSOR, Serial1);
    node.preTransmission(preTransmission);
    node.postTransmission(postTransmission);

    int v = (vel * 8192) / 60;

    result = node.writeSingleRegister(REG, v);

    delay(100);

    if (result != node.ku8MBSuccess)
    {
        Error(REG);
    }
}

void Start()
{
    tricicloAssistido.frequencia = 3;
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1], tricicloAssistido.frequencia);
}

```

```
if (tricicloAssistido.sentidoGiro)
{
    WriteSingleRegister(REG_ADDR_WRITE[0], 0x0007, "Start");
}
else
{
    WriteSingleRegister(REG_ADDR_WRITE[0], 0x0003, "Start Reversao");
}
}

void Stop()
{
    WriteSingleRegister(REG_ADDR_WRITE[0], 0x0000, "Stop");
}

void Emergency_Stop()
{
    WriteSingleRegister(REG_ADDR_WRITE[0], 0x0040, "Emergency Stop");
}

void Reversao()
{
    tricicloAssistido.frequencia = 3;
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1], tricicloAssistido.frequencia);
    Emergency_Stop();
    tricicloAssistido.sentidoGiro = !tricicloAssistido.sentidoGiro;
}

void ConfigControlBornes()
{
    Stop();

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Configurando");
    lcd.setCursor(0, 1);
    lcd.print("Modo Serial");
    lcd.setCursor(0, 2);

    for (int i = 0; i < 6; i++)
    {
```

```

    WriteSingleRegister2(REG_ADDR_CONFIG_COMANDO[i],
REG_ADDR_CONFIG_VALUE_MOTOR[i]);
    lcd.setCursor(i, 2);
    lcd.print("|");
    lcd.setCursor(0, 3);
    lcd.print(String(i * 10) + "%");

    delay(50);
}

for (int i = 0; i < 9; i++)
{
    WriteSingleRegister2(REG_ADDR_CONFIG_COMANDO[i],
REG_ADDR_CONFIG_VALUE_QUADRO[i]);
    lcd.setCursor(i, 2);
    lcd.print("|");
    lcd.setCursor(0, 3);
    lcd.print(String(i * 10) + "%");

    delay(50);
}
}

void ConfigControlSerial()
{
    Stop();

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Configurando");
    lcd.setCursor(0, 1);
    lcd.print("Modo Serial");
    lcd.setCursor(0, 2);

    for (int i = 0; i < 6; i++)
    {
        WriteSingleRegister2(REG_ADDR_CONFIG_COMANDO[i],
REG_ADDR_CONFIG_VALUE_MOTOR[i]);
        lcd.setCursor(i, 2);
        lcd.print("|");
        lcd.setCursor(0, 3);
        lcd.print(String(i * 10) + "%");
    }
}

```

```

    delay(50);
}

for (int i = 0; i < 9; i++)
{
    WriteSingleRegister2(REG_ADDR_CONFIG_COMANDO[i],
REG_ADDR_CONFIG_VALUE_SERIAL[i]);
    lcd.setCursor(i, 2);
    lcd.print("|");
    lcd.setCursor(0, 3);
    lcd.print(String(i * 10) + "%");

    delay(50);
}

void Reset()
{
    Emergency_Stop();
    WriteSingleRegister(204, 5, "Reset");
}

void Config()
{
    Stop();

    lcd.clear();
    lcd.print("Configurando...");

    for (int i = 0; i < 6; i++)
    {
        WriteSingleRegister2(REG_ADDR_CONFIG_COMANDO[i],
REG_ADDR_CONFIG_VALUE_MOTOR[i]);
        lcd.setCursor(i, 2);
        lcd.print("|");
        lcd.setCursor(0, 3);
        lcd.print(String(i * 10) + "%");

        delay(50);
    }
}

```

```
if (tricicloAssistido.typeConfig == 0)
{
    ConfigControlBornes();
}
else if (tricicloAssistido.typeConfig == 1)
{
    ConfigControlSerial();
}

tricicloAssistido.isConfig = false;

lcd.clear();

delay(1000);
}

void Write_Multiple_Register(char addr, uint16_t REG)
{
    uint8_t result;
    uint16_t data[2];

    node.begin(ID_INVERSOR, Serial1);
    node.preTransmission(preTransmission);
    node.postTransmission(postTransmission);

    result = node.writeMultipleRegisters(REG, 2);
    data[0] = 3;
    data[1] = 4096;

    node.setTransmitBuffer(0, data[0]);
    node.setTransmitBuffer(1, data[1]);

    delay(100);

    Serial.print(data[1]);

    if (result != node.ku8MBSuccess)
    {
        Error(REG);
    }
}
```



```
void GET_MEDIDA()
{
    delay(700);

    for (char i = 0; i < 2; i++)
    {
        DATA_MEDIDA[i] = Read_Holding_Register(ID_INVERSOR,
        REG_ADDR_READ[i]);
    }
}

void ExecMedida()
{
    GET_MEDIDA();

    Serial.println("Lendo as tensões do dispositivo");
    Serial.print("V = ");
    Serial.print(DATA_MEDIDA[0], 3);
    Serial.println(" V");
    Serial.print("A = ");
    Serial.print(DATA_MEDIDA[1], 3);
    Serial.println(" A");
    Serial.print("");

    delay(1000);
}

void Teste()
{
    Start();
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1], 10);
    delay(1000);
    Stop();
    delay(1000);
    Start();
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1], 15);
    delay(1000);
    Emergency_Stop();
    delay(1000);
}
```

```

void ControleVoltas(){
  unsigned long y = tricicloAssistido.tempoPrimeiraVolta + (tricicloAssistido.nVoltas -
1)*tricicloAssistido.tempoVoltasRestantes;

  Serial.print(String(millis()) + "\n");

  unsigned long time = millis() + y;

  Serial.print(String(time) + "\n");

  Serial.print("Inicio\n");
  Start();
  while(millis() < time){}
  Stop();
  Serial.print("Fim\n");

}

```

```

void ControleBluetooth()
{
  while (SerialBT.available())
  {
    tricicloAssistido.isConnected = SerialBT.connected();
    if (!tricicloAssistido.isConnected)
    {
      Emergency_Stop();
      break;
    }

    tricicloAssistido.c = (char)SerialBT.read();

    tricicloAssistido.strCMD += tricicloAssistido.c;
  }

  if (tricicloAssistido.strCMD[tricicloAssistido.strCMD.length() - 1] == 'A')
  {
    tricicloAssistido.nVoltas = tricicloAssistido.strCMD.substring(0,
(tricicloAssistido.strCMD.length() - 2)).toInt();
    Serial.print(tricicloAssistido.nVoltas);

    tricicloAssistido.sentidoGiro = true;
  }
}

```

```

NovaFuncionalidade();

triculoAssistido.strCMD = "";
}
else if (triculoAssistido.strCMD[triculoAssistido.strCMD.length() - 1] == 'H')
{
    triculoAssistido.nVoltas = triculoAssistido.strCMD.substring(0,
(triculoAssistido.strCMD.length() - 2)).toInt();
    Serial.print(triculoAssistido.nVoltas);

    triculoAssistido.sentidoGiro = false;

    NovaFuncionalidade();

    triculoAssistido.strCMD = "";

    triculoAssistido.sentidoGiro = true;
}
else
{

    if (triculoAssistido.strCMD.length() > 0)
    {
        if (triculoAssistido.strCMD == "START")
        {
            Start();
        }
        else if (triculoAssistido.strCMD == "STOP")
        {
            Stop();
        }
        else if (triculoAssistido.strCMD == "EMERGENCY_STOP")
        {
            Emergency_Stop();
        }
        else if (triculoAssistido.strCMD == "REVERSAO")
        {
            Reversao();
        }
        else if (triculoAssistido.strCMD == "MODO_QUADRO")
        {

```

```

    ConfigControlBornes();
}
else if (tricicloAssistido.strCMD == "MODO_APP")
{
    ConfigControlSerial();
}
else
{
    tricicloAssistido.frequencia = tricicloAssistido.strCMD.toFloat();
    SetVelocity(ID_INVERSOR, REG_ADDR_WRITE[1],
tricicloAssistido.frequencia);
}

    tricicloAssistido.strCMD = "";
}
}
}

```

```

void StatusCarregandoDisplay()

```

```

{
    Emergency_Stop();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Desconectado...");

    int i = 0;
    int l = 6;

    while (!tricicloAssistido.isConnected)
    {

        tricicloAssistido.isConnected = SerialBT.connected();
        if (i < l)
        {
            lcd.setCursor(i, 1);
            lcd.print("|");
        }
        else if (i < 16)
        {
            lcd.setCursor(i - l + 1, 1);
            for (int k = 0; k < l; k++)
            {

```

```
        lcd.print("|");
    }
    lcd.setCursor(i - l, 1);
    lcd.print(" ");
}
else
{
    lcd.setCursor(i - l, 1);
    lcd.print(" ");
}

delay(500);
i++;
if (i >= (15 + l + 1))
{
    i = 0;
}
}

lcd.clear();
lcd.print("Conexao Aceita");

delay(1000);

lcd.clear();
lcd.print("Controle");
lcd.setCursor(0, 1);
lcd.print("Liberado");
}

void StatusDisplay(String s)
{
    lcd.print(s);
}

void setup()
{
    Serial.begin(9600);
    Serial1.begin(19200, SERIAL_8E1, SERIAL_RX_PIN, SERIAL_TX_PIN);

    pinMode(MAX485_DE, OUTPUT);
    pinMode(MAX485_RE, OUTPUT);
```

```
SerialBT.begin("TRIARE");

digitalWrite(MAX485_DE, 0);
digitalWrite(MAX485_RE, 0);

lcd.init();
lcd.backlight();
}

void loop()
{

  if (tricicloAssistido.isReset)
  {
    Reset();
  }

  if (tricicloAssistido.isConfig)
  {
    Emergency_Stop();
    Config();
  }
  else
  {
    if (tricicloAssistido.isTest)
    {
      Teste();
    }
    else
    {
      tricicloAssistido.isConnected = SerialBT.connected();
      if (!tricicloAssistido.isConnected)
      {
        Emergency_Stop();
        StatusCarregandoDisplay();
      }
      else
      {
        ControleBluetooth();
      }
    }
  }
}
```

}
}

APÊNDICE C – Arquivo REG_CFW100.h

```
#define ID_INVERSOR 1
#define TOTAL_DE_REG 2

// Registradores de controle e referência de velocidade
#define REG_CTRL 682
#define REG_VELOCIDADE 683

// Registradores de Comando
#define REG_CONFIG_220 220
#define REG_CONFIG_222 222
#define REG_CONFIG_226 226
#define REG_CONFIG_227 227
#define REG_CONFIG_308 308
#define REG_CONFIG_310 310
#define REG_CONFIG_263 263
#define REG_CONFIG_264 264
#define REG_CONFIG_265 265
#define REG_CONFIG_266 266

// Registradores de Configuração do motor
#define REG_CONFIG_120 120
#define REG_CONFIG_399 399
#define REG_CONFIG_400 400
#define REG_CONFIG_401 401
#define REG_CONFIG_402 402
#define REG_CONFIG_403 403
#define REG_CONFIG_404 404
#define REG_CONFIG_407 407
#define REG_CONFIG_134 134

// Registradores de leitura
#define REG_V 002
#define REG_A 003

uint16_t REG_ADDR_READ[2] = {
    REG_V,
    REG_A,
};

uint16_t REG_ADDR_WRITE[2] = {
    REG_CTRL,
    REG_VELOCIDADE,
};

uint16_t REG_ADDR_CONFIG_MOTOR[8] = {
    REG_CONFIG_120,
    REG_CONFIG_134,
    REG_CONFIG_400,
```






```
REG_CONFIG_401,  
REG_CONFIG_402,  
REG_CONFIG_403,  
REG_CONFIG_404  
};  
  
uint16_t REG_ADDR_CONFIG_COMANDO[9] = {  
    REG_CONFIG_220,  
    REG_CONFIG_222,  
    REG_CONFIG_226,  
    REG_CONFIG_227,  
    REG_CONFIG_308,  
    REG_CONFIG_310,  
    REG_CONFIG_263,  
    REG_CONFIG_264,  
    REG_CONFIG_265  
};  
  
uint16_t REG_ADDR_CONFIG_VALUE_MOTOR[8] = {  
    0,  
    150,  
    220,  
    1,  
    1700,  
    60,  
    3,  
};  
  
uint16_t REG_ADDR_CONFIG_VALUE_SERIAL[9] = {  
    1,  
    9,  
    5,  
    2,  
    1,  
    1,  
    1,  
    8,  
    0  
};  
  
uint16_t REG_ADDR_CONFIG_VALUE_QUADRO[9] = {  
    1,  
    7,  
    4,  
    1,  
    1,  
    1,  
    1,  
    1,  
    11,  
    12
```

```
};
```

```
float DATA_MEDIDA[TOTAL_DE_REG];
```

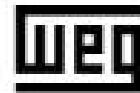
ANEXO A – DADOS TÉCNICOS CFW100

FOLHA DE DADOS		
Inversores de Frequência		
Características Principais		
	Referência	: CFW100B02PES120G2
	Código do produto	: 14248224
	Linha de produto	: CFW100
Dados básicos		
Tensão nominal de entrada		: 110-127 V
Tensão mínima - máxima de entrada		: 96-140 V
Número de Fases de entrada		: Monofásico
- De entrada		: 1
- De saída		: 3
		Pesada (HD)
Corrente nominal (HD)		: 2,6 A
Corrente de sobrecarga para 60 s (HD)		: 3,6 A
Corrente de entrada monofásica (HD) [1]		
Motor máximo aplicável:		
Tensão/Frequência	Sobrecarga Normal (HD)	Sobrecarga Pesada (HD)
220V / 60Hz	Não aplicável	0,75 / 0,55
220V / 60Hz	Não aplicável	0,5 / 0,37
230V / 60Hz	Não aplicável	0,75 / 0,55
230V / 60Hz	Não aplicável	0,5 / 0,37
Não aplicável	Não aplicável	Não aplicável
Não aplicável	Não aplicável	Não aplicável
Não aplicável	Não aplicável	Não aplicável
Não aplicável	Não aplicável	Não aplicável
Filtro RFI externo		: CFW100-KFABC-S1
Inibidor do Link		:
Cartão de memória		: Não incluído no produto
Porta USB		: Sim, via CFW100-CUSB
Frequência de rede		: 50/60Hz
Faixa de frequência de rede (mínima - máxima)		: 48-62 Hz
Desequilíbrio entre fases		: Menor ou igual a 3% da tensão de linha nominal de entrada
Tensões transitórias e sobretensões		: Categoria III
Fator de potência típico de entrada		: 0,70
Fator de deslocamento típico		: 0,98
Rendimento típico na condição nominal		: \geq 97%
Número máximo conexões (de rede) por hora		: 10 (1 a cada 6 minutos)
Alimentação da potência em corrente contínua		:
Frequência de chaveamento [3]		: 5 kHz
Frequência de chaveamento selecionáveis		: 2,5 e 15 kHz
Relógio de tempo real		: Não disponível
Função Copy		: Sim, via CFW100-CFW300-MMP
		: 30 W
Fonte disponível ao usuário		
Tensão de saída		: Não aplicável
Capacidade máxima		: Não aplicável
Dados de controle / desempenho		
Alimentação		: Fonte chaveada
Métodos de Controle - motor de indução		: V/F (escalar) e V/VW
Interface Encoder		: Não aplicável
Frequência de saída do controle [5]		: 0-400 Hz
Resolução de frequência		: 0,1 Hz
Controle V/F		
- Regulação de velocidade		: 1% da velocidade nominal
- Variação de velocidade		: 1:20
Controle V/VW		
- Regulação de velocidade		: 1% da velocidade nominal
- Variação de velocidade		: 1:30
Controle vetorial sensorless		
- Regulação de velocidade		: Não aplicável
- Variação de velocidade		: Não aplicável
Controle vetorial com Encoder		
- Regulação de velocidade		: Não aplicável
23/08/2023	As informações contidas são valores de referência. Sujeito a alterações sem aviso prévio. Imagem meramente ilustrativa.	1 / 4

FOLHA DE DADOS Inversores de Frequência		
Normas atendidas		
		<ul style="list-style-type: none"> - EN 61000-4-4 - Electromagnetic compatibility (EMC) - Part 4: Testing and measurement techniques - Section 4: Electrical fast transient/burst immunity test. - EN 61000-4-5 - Electromagnetic compatibility (EMC) - Part 4: Testing and measurement techniques - Section 5: Surge immunity test. - EN 61000-4-6 - Electromagnetic compatibility (EMC) - Part 4: Testing and measurement techniques - Section 6: Immunity to conducted disturbances, induced by radio-frequency fields. - Somente com filtro externo.
Construção Mecânica		<ul style="list-style-type: none"> - EN 60529 - degrees of protection provided by enclosures (IP code) - UL 50 - enclosure for electrical equipment. - IEC 60721-3-3 - classification of environmental conditions - part 3: classification of groups of environmental parameters and their severities - section 3: stationary use at weather protected locations level 3nd. - EN 60529 e UL 50
Certificações		
Notas		
<p>1) Considerado a potência de rede mínima 1%.</p> <p>2) Potências de motores orientativas, válidas para motores WEG standard de N pólos. O dimensionamento correto deve ser feito em função da corrente nominal do motor utilizado, que deve ser menor ou igual à corrente nominal de saída do inversor.</p> <p>3) Para operação com frequência de chaveamento acima da nominal, aplicar derating na corrente de saída (consultar o manual do usuário).</p> <p>4) Montagem em superfície, sobrecarga HD.</p> <p>5) Somente para proteção do circuito elétrico. Para proteção dos inversores, utilizar os fusíveis ultrarrápidos indicados.</p> <p>6) Somente com filtro externo.</p> <p>7) Para obter mais informações, consulte o manual do usuário do CPW100.</p> <p>8) Todas as imagens são meramente ilustrativas.</p>		
23/08/2023	As informações contidas são valores de referência. Sujeito a alterações sem aviso prévio. Imagem meramente ilustrativa.	4 / 4

FOLHA DE DADOS

Inversores de Frequência



Condições ambientais

Temperatura ao redor do inversor: de 0 °C a 50 °C. Para temperaturas acima do especificado é necessário aplicar redução de corrente de 2 % por °C de 50 a 60 °C.

Umidade relativa do ar: 5% a 95% sem condensação.

Altitude: até 1000 m (3281 ft) em condições normais. De 1000 m (3281 ft) a 4000 m (13123 ft) reduzir a corrente em 1% para cada 100 m acima (0,3% para cada 100 ft acima) de 1000 m (3281 ft). Reduzir a tensão máxima (127 V para modelos 110...127 V e 240 V para modelos 200...240 V) em 1,1% para cada 100 m acima (0,33% para cada 100 ft acima) de 2000 m.

Diretivas de sustentabilidade

RoHS : Sim
 Conformal Coating : 3C2 (IEC 60721-3-3:2002)

Dimensões e peso

- Tamanho : B
 - Altura : 117 mm / 4.6 in
 - Largura : 66 mm / 2.17 in
 - Profundidade : 129 mm / 5.08 in
 - Peso : 0,57 kg / 1.26 lb

Instalação Mecânica

Posição de montagem : Trilho DIN
 Parafuso para fixação : M4 com kit PLMP
 Torque de aperto : 2,5 N.m / 1.84 lb.ft
 Permite montagem lado-a-lado : Sim, sem derating
 Espaço mínimo ao redor do inversor:
 - Superior : 15 mm / 0.59 in
 - Inferior : 50 mm / 1.97 in
 - Frontal : 40 mm / 1.57 in
 - Entre inversores (IP20) : Não aplicável

Conexões elétricas

Bits e torques de aperto:

	Bit do cabo recomendada	Torque de aperto recomendado
Potência	2,5 mm ² (14 AWG)	1,4 N.m / 1.03 lb.ft
Frenagem	Não aplicável	1,4 N.m / 1.03 lb.ft
Aterramento	2,5 mm ² (14 AWG)	1,4 N.m / 1.03 lb.ft
Controle	0,5 A, 1,5 mm ² (20 a 14 AWG)	0,5 N.m / 0.37 lb.ft

Especificações complementares

SoftPLC : Sim, incorporado
 Corrente máxima de frenagem : Não disponível
 Resistência mínima para o resistor de frenagem : Não disponível
 Fusível recomendada : FNH00-20R-A
 MPW00-3-L010


Normas atendidas

Segurança	<ul style="list-style-type: none"> - UL 508C - Power conversion equipment. - UL 840 - Insulation coordination including clearances and creepage distances for electrical equipment. - EN 61800-5-1 - Safety requirements electrical, thermal and energy. - EN 50178 - Electronic equipment for use in power installations. - EN 60204-1 - Safety of machinery. Electrical equipment of machines. Part 1: General requirements. Nota: Para ser uma máquina em conformidade com essa norma, o fabricante da máquina é responsável pela instalação de um dispositivo de parada de emergência e um equipamento para seccionamento de rede. - EN 60148 (IEC 145) - Semiconductor converters. - EN 61800-2 - Adjustable speed electrical power drive systems - Part 2: General requirements - Rating specifications for low voltage adjustable frequency AC power drive systems. - UL 508C - Power conversion equipment.
Compatibilidade Eletromagnética [M]	<ul style="list-style-type: none"> - EN 61800-3 - Adjustable speed electrical power drive systems - Part 3: EMC product standard including specific test methods. - EN 55011 - Limits and methods of measurement of radio disturbance characteristics of industrial, scientific and medical (ISM) radio-frequency equipment. - CISPR 11 - Industrial, scientific and medical (ISM) radio-frequency equipment - Electromagnetic disturbance characteristics - Limits and methods of measurement. - EN 61000-4-2 - Electromagnetic compatibility (EMC) - Part 4: Testing and measurement techniques - Section 2: Electrostatic discharge immunity test. - EN 61000-4-3 - Electromagnetic compatibility (EMC) - Part 4: Testing and measurement techniques - Section 3: Radiated, radio-frequency, electromagnetic field immunity test.

23/08/2023

As informações contidas são valores de referência. Sujeito a alterações sem aviso prévio. Imagem meramente ilustrativa.

3 / 4

FOLHA DE DADOS Inversores de Frequência		
Controle V/F		
- Variação de velocidade		: Não aplicável
Entradas Analógicas		
Quantidade (padrão)		: Não disponível
Níveis		: Não aplicável
Impedância para entrada em tensão		: Não aplicável
Impedância para entrada em corrente		: Não aplicável
Função		: Não aplicável
Tensão máxima admissível		: Não aplicável
Entradas digitais		
Quantidade (padrão)		: Não disponível
Atribuição		: Ativo baixo e alto
Nível baixo máximo		: 5 V (baixo) e 10 V (alto)
Nível alto mínimo		: 10 V (baixo) e 20 V (alto)
Corrente de entrada		: 11 mA
Corrente de entrada máxima		: 20 mA
Função		: Programável
Tensão máxima admissível		: 30 Vcc
Saídas analógicas		
Quantidade (padrão)		: Somente com plug-in
Níveis		: Não aplicável
RL para saída em tensão		: Não aplicável
RL para saída em corrente		: Não aplicável
Função		: Não aplicável
Saídas digitais		
Quantidade (padrão) e tipo		: 3 relés NA e 1 transistor
Tensão máxima		: Não aplicável
Corrente máxima		: Não aplicável
Função		: Não aplicável
Comunicação		
- Modbus-RTU (com acessório: CPW100-CRB485, CPW100- CUSB ou CPW100-CBLT)		
- Modbus/TCP (Não disponível)		
- Profibus DP (Não disponível)		
- Profibus DPV1 (Não disponível)		
- Profinet (Não disponível)		
- CANopen (com acessório: CPW100-CCAN)		
- DeviceNet (com acessório: CPW100-CCAN)		
- Ethernet/IP (Não disponível)		
- EtherCAT (Não disponível)		
- Bluetooth (com acessório: CPW100-CBLT)		
- BACnet (Não disponível)		
Proteções disponíveis		
- Sobrecorrente/Curto fase-fase na saída		
- Não aplicável		
- Subtensão/tensão na potência		
- Sobretemperatura do dissipador		
- Sobrecarga no motor		
- Não aplicável		
- Falha / Alarme externo		
- Erro de programação		
- Falha na CPU ou memória		
Interface de operação (HMI)		
Disponibilidade		: Incluída no produto
Instalação HMI		: HMI fixo
Quantidade de teclas HMI		: 4
Display		: LCD Numérico
Exatidão de indicação de corrente		: 10% da corrente nominal
Resolução de velocidade		: 0,1 Hz
Grau de proteção da HMI padrão		: IP20
Tipo de bateria da HMI		: Não aplicável
Expectativa de vida da bateria da HMI		: Não aplicável
Tipo de HMI remota		: Acessório CPW100-KHMR
Moldura para a HMI remota		: Não aplicável
Grau de proteção da HMI remota		: IP54
Condições ambientais		
Grau de proteção		: IP20
Grau de poluição (EN50178 e UL508C)		: 2
23/08/2023	As informações contidas são valores de referência. Sujeito a alterações sem aviso prévio. Imagem meramente ilustrativa.	2 / 4