UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

EDUARDO DOS SANTOS JUNIOR

MONITORAMENTO REMOTO APLICADO A UM VEÍCULO ELÉTRICO

PATO BRANCO

2023

EDUARDO DOS SANTOS JUNIOR

MONITORAMENTO REMOTO APLICADO A UM VEÍCULO ELÉTRICO

Remote monitoring applied to an electric vehicle

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do Curso de Bacharelado em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Gustavo Weber Denardin

PATO BRANCO

2023

EDUARDO DOS SANTOS JUNIOR

MONITORAMENTO REMOTO APLICADO A UM VEÍCULO ELÉTRICO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do Curso de Bacharelado em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 26/junho/2023

Prof. MsC. André Macário Barros
Universidade Tecnológica Federal do Paraná

Prof. Dr. Giovanni Alfredo Guarneri
Universidade Tecnológica Federal do Paraná

Prof. Dr. Gustavo Weber Denardin
Universidade Tecnológica Federal do Paraná

PATO BRANCO
2023

Dedico este trabalho à minha família, por todo suporte, amor e carinho. Dedico também este trabalho aos amigos que me apoiaram e incentivaram nesta jornada.

AGRADECIMENTOS

Agradeço à minha família por todo suporte, incentivo, amor e carinho. Mesmo estando longe de casa me apoiaram e me incentivaram durante todos esses anos de graduação.

Agradeço ao meu orientador Prof. Dr. Gustavo Weber Denardin por todo suporte, ajuda e sabedoria ao me guiar durante o processo de desenvolvimento deste trabalho.

Agradeço ao Prof. Dr. Diogo Ribeiro Vargas pelos ensinamentos durante as orientações de projetos de extensão e iniciação científica.

Agradeço aos amigos da Equipe Tubarão Branco pelas manhãs de sábado de muito aprendizado, companheirismo e amizade.

Agradeço aos professores da UTFPR por toda sustentação e ensinamentos.

Agradeço à UTFPR por proporcionar toda a estrutura para me desenvolver pessoalmente e profissionalmente.

RESUMO

Este documento apresenta o desenvolvimento e a implementação de um sistema de monitoramento remoto aplicado a um protótipo de veículo elétrico, desenvolvido pela equipe de eficiência energética Tubarão Branco. O objetivo do sistema é coletar os dados gerados pelo protótipo durante o seu funcionamento e transmiti-los via comunicação sem fio para um computador, onde serão exibidos em forma gráfica. A comunicação entre os dispositivos do protótipo é realizada por meio de uma rede CAN e neste trabalho foi proposto o desenvolvimento de um dispositivo interno ao veículo para ler os dados da rede CAN e transmiti-los utilizando a tecnologia LoRa para um dispositivo externo ao veículo. Esse dispositivo externo, por sua vez, realiza a transmissão dos dados para um computador, onde serão processados e exibidos em gráficos utilizando a linguagem de programação Python. A geração de gráficos a partir dos dados coletados tem como objetivo fornecer uma análise detalhada do desempenho do veículo durante os trajetos percorridos durante competições. Isso proporciona maior segurança ao piloto e auxilia a equipe na busca por melhorias e otimizações do veículo. Vale ressaltar que, caso sejam adicionados mais dispositivos ao veículo, o projeto pode ser facilmente expandido pelos membros da equipe. Ao longo deste documento, serão apresentados os dispositivos utilizados no sistema, detalhes sobre sua implementação e funcionamento, além da metodologia adotada para a coleta e análise dos dados.

Palavras-chave: monitoramento remoto veicular; can; lora; sistemas embarcados; iot.

ABSTRACT

This document presents the development of a remote monitoring system applied to an electric vehicle prototype, developed by the Tubarão Branco energy efficiency team. The goal of the system is to collect data generated by the prototype during its operation and wirelessly transmit it to a computer, where it will be displayed graphically. Communication between the prototype devices is performed through a CAN bus, and this work proposes the development of an internal device within the vehicle to read the data from the CAN bus and transmit it using LoRa technology to an external device. This external device, in turn, transmits the data to a computer, where it will be processed and displayed in graphs using the Python programming language. The generation of graphs from the collected data aims to provide a detailed analysis of the vehicle's performance during the journeys undertaken in competitions. This provides greater safety for the pilot and assists the team in seeking improvements and optimizations for the vehicle. It is worth noting that, if more devices are added to the vehicle, the project can be easily expanded by the team members. Throughout this document, the devices used in the system, details of their implementation and operation, and the methodology adopted for data collection and analysis will be presented.

Keywords: vehicle remote monitoring; can; lora; embedded systems; iot.

LISTA DE FIGURAS

Figura 1 – Camadas do modelo OSI 17
Figura 2 – Topologia da Rede CAN 18
Figura 3 – Camadas da CAN 19
Figura 4 – Codificação/decodificação de bits recessivo e dominante 20
Figura 5 – Segmentos do tempo de bit 21
Figura 6 – Tempo quanta e tempo de bit 21
Figura 7 – Quadro de dados com identificador de de 11 *bits* 23
Figura 8 – Arbitração dos *nós* 24
Figura 9 – camadas da LoRa 26
Figura 10 – Chirp Ascendente e chirp descendente 26
Figura 11 – Símbolos LoRa 27
Figura 12 – Estrutura do pacote LoRa 28
Figura 13 – Arquitetura da solução proposta 30
Figura 14 – Microcontrolador ESP32 TTGO T-Beam V1.1 32
Figura 15 – Microcontrolador ESP32 TTGO LORA V1.0 32
Figura 16 – Conversor USB Serial FTDI FT232RL 33
Figura 17 – Transceptor CAN SN65HVD230 33
Figura 18 – Fluxograma das tarefas executadas pelos dispositivos 36
Figura 19 – Conexão dos pinos do dispositivo interno 38
Figura 20 – Fluxograma do gerenciamento das tarefas do dispositivo interno 39
Figura 21 – Monitor dispositivo interno 41
Figura 22 – Conexão entre a placa FT232RL e o kit de desenvolvimento 42
Figura 23 – Display dispositivo externo 43
Figura 24 – Fluxograma do gerenciamento das tarefas do dispositivo externo . . . 45
Figura 25 – Monitor dispositivo externo 46
Figura 26 – IPython console da IDE Spyder com os dados lidos e escritos no arquivo csv 47
Figura 27 – Visualização gráfica dos dados 47
Figura 28 – Distância do experimento de validação 49
Figura 29 – Distância do experimento de validação 49

Figura 30 – Distância das extremidades da pista na competição 50
Figura 31 – Distâncias do experimento de validação das configurações 50

LISTA DE TABELAS

Tabela 1 – Bandas de frequência regulamentadas 29

Tabela 2 – Mapeamento da conexão dos pinos do módulo LoRa e do kit de desenvolvimento 37

Tabela 3 – Mapeamento da conexão dos pinos do transceptor CAN do kit de desenvolvimento 37

Tabela 4 – Configuração dos parâmetros da rede CAN 38

Tabela 5 – Configuração dos parâmetros da LoRa 39

Tabela 6 – Identificadores dos dispositivos 40

Tabela 7 – Mapeamento da conexão dos pinos do módulo LoRa e do kit de desenvolvimento 42

Tabela 8 – Mapeamento da conexão dos pinos da placa FT232RL e do kit de desenvolvimento 42

Tabela 9 – Mapeamento da conexão dos pinos do display OLED no kit de desenvolvimento 43

Tabela 10 – Configuração dos parâmetros da LoRa 44

Tabela 11 – Configuração da comunicação UART 44

Tabela 12 – Configuração dos parâmetros fixos no experimento 51

Tabela 13 – Configuração dos parâmetros variáveis no experimento 51

LISTA DE ABREVIATURAS E SIGLAS

Siglas

t_{OSC}	período do oscilador
t_{SF1}	segmento de fase 1
t_{SF2}	segmento de fase 2
$t_{SegProp}$	segmento de propagação
$t_{SegSinc}$	segmento de sincronização
ACK	*acknowledges*
Anatel	Agência Nacional de Telecomunicações
BW	*Bandwidth*
CAN	*Controller Area Network*
CCITT	Comitê Consultivo Internacional de Telégrafo e Telefone
CR	*Coding Rating*
CRC	*cyclic redundancy check*
CSS	*Chirp Spread Spectrum*
CSV	*Comma-Separated Values*
DLC	*data length code*
EOF	*end-of-frame*
EV	*Electric vehicles*
FIFO	*First In First Out*
GPS	*Global Positioning System*
GUI	*Graphical User Interface*
I2C	*Inter-Integrated Circuit*
IDE	*identifier extension*

IFS	*interframe space*
IHM	Interação homem-máquina
IoT	*Internet of Things*
ISO	Organização Internacional de Padronização
LLC	*Logical Link Control*
LoRa	*Long Range*
MAC	*Medium Access Control*
NRZ	*Non Return to Zero*
OSI	*Open System Interconnection*
r0	bit reservado
RTOS	*Real-time operating system*
RTR	*remote transmission request*
SF	*Spreading Factor*
SOF	*start of frame*
SPI	*Serial Peripheral Interface*
TCC	Trabalho de conclusão de curso
TQ	Tempo Quanta
UART	*Universal Asynchronous Receiver / Transmitter*
USB	*Universal Serial Bus*
UTFPR	Universidade Tecnológica Federal do Paraná
VSCode	*Visual Studio Code*

SUMÁRIO

1 INTRODUÇÃO 13

1.1 Objetivos 14

1.1.1 Objetivo geral 14

1.1.2 Objetivos específicos 14

1.2 Estrutura do trabalho 14

2 REFERENCIAL TEÓRICO 16

2.1 Camadas *Open System Interconnection* (OSI) 16

2.2 Protocolo de comunicação CAN 17

2.2.1 Camadas do protocolo de comunicação CAN 18

2.2.2 Camada Física CAN 19

2.2.3 Sincronização, taxa de transmissão de *bits* e tempo de *bit* 20

2.2.4 Tempo quanta 22

2.2.5 Subcamada Media Access Control (MAC) 22

2.2.6 Quadros de dados 22

2.2.7 Arbitração dos *Nós* 23

2.2.8 Subcamada Logical Link Control (LLC) 24

2.2.9 Filtros de aceitação 24

2.2.10 Considerações finais sobre o protocolo CAN 25

2.3 Tecnologia LoRa 25

2.3.1 Camadas da tecnologia LoRa 25

2.3.2 Parâmetros da LoRa 26

2.3.3 Símbolos LoRa 27

2.3.4 Pacote de dados LoRa 27

2.3.5 Regulamentação das bandas de frequências da LoRa 28

2.3.6 Considerações finais sobre a tecnologia LoRa 29

3 ARQUITETURA E MATERIAIS 30

3.1 Arquitetura da solução proposta 30

3.2 Materiais 31

3.2.1 Hardware 32

3.2.2 Software 33

3.2.3 Dispositivo LoRa Interno 34
3.2.4 Dispositivo LoRa Externo 34
3.2.5 Computador 35
4 DESENVOLVIMENTO E RESULTADOS 36
4.1 Implementação do dispositivo LoRa interno 37
4.1.1 Implementação do hardware do dispositivo interno 37
4.1.2 Configuração da rede CAN no dispositivo interno 38
4.1.3 Configuração da LoRa no dispositivo interno 39
4.1.4 Implementação das tarefas do dispositivo interno 39
4.1.5 Resultado do dispositivo LoRa interno 40
4.2 Implementação do dispositivo externo 41
4.2.1 Implementação do hardware do dispositivo externo 41
4.2.2 Configuração da LoRa no dispositivo externo 43
4.2.3 Configuração da UART no dispositivo externo 44
4.2.4 Implementação do software do dispositivo externo 44
4.2.5 Resultado do dispositivo LoRa Externo 45
4.3 Implementação do tratamento de dados por meio do computador 45
4.3.1 Considerações finais da implementação 48
4.4 Validação do sistema e discussões 48
4.4.1 Teste 1: Validação para a configuração apresentada o documento 48
4.4.2 Teste 2: Validação para distintas configurações 50
4.4.3 Considerações finais do projeto 51
5 CONCLUSÃO 52
REFERÊNCIAS 53

1 INTRODUÇÃO

Os veículos elétricos (*Electric vehicles* (EV)), do tipo totalmente elétricos ou híbridos, têm recebido destaque na indústria e na academia. O primeiro EV data de 1899 tendo baixa autonomia o que limitava seu alcance e seu uso fora dos grandes centros urbanos. O Toyota Prius, um EV híbrido lançado em 1997, considerado o primeiro veículo comercial de sucesso econômico (SANTIAGO *et al.*, 2012). Atualmente é possível encontrar vários veículos elétricos comerciais sendo utilizados nas estradas. No ano de 2020 a empresa americana *Tesla*, fabricou 499.600 carros, e a alemã *Volkswagen* 421.600 unidades, considerando totalmente elétricos e híbridos (ZSW, 2021). O número de veículos movidos totalmente a eletricidade vem crescendo continuamente no mercado, sendo que o total de EV no mundo foi 1.398.050 (em 2015), 2.155.410 (em 2016), 3.408.670 (em 2017), 5.606.490 (em 2018), 7.860.690 (em 2019) e 10.907.150 (em 2020) (ZSW, 2018; ZSW, 2021).

Visando o estudo e a implementação de tecnologias utilizadas na indústria e na academia, a equipe *Tubarão Branco*[1] da Universidade Tecnológica Federal do Paraná (UTFPR) Campus Pato Branco, realiza o desenvolvimento de um protótipo de veículo elétrico de alta eficiência energética. Os alunos participantes da equipe desenvolveram, até o momento da publicação deste documento, o acionamento de um motor sem escovas, o dispositivo de Interação homem-máquina (IHM) no volante do veículo e um joulímetro para realizar a medição de consumo de energia elétrica do veículo.

O dispositivo de IHM localizado no volante do veículo desempenha um papel fundamental na interação do piloto com o veículo, permitindo o controle do motor de tração. Esta interação é realizada por meio da leitura de um potenciômetro e estes dados são enviados ao sistema de acionamento do motor para controlar sua potência.

Para viabilizar a transmissão dos dados provenientes do dispositivo IHM ao sistema de acionamento do motor, foi implementado um sistema de comunicação baseado no protocolo *Controller Area Network* (CAN). O desenvolvimento desse sistema teve início durante o período de iniciação científica, realizado entre os anos de 2020 e 2021, pelo autor deste Trabalho de conclusão de curso (TCC) (JUNIOR; SOUZA; VARGAS, 2021).

Posteriormente, um joulímetro foi integrado ao protótipo do veículo elétrico, utilizando o sistema de comunicação existente, com o objetivo de realizar a medição da tensão e corrente da bateria. Essa integração permitiu a coleta automatizada dos valores de tensão e corrente da bateria, proporcionando medições mais precisas e confiáveis. A adição do joulímetro ao sistema de comunicação possibilitou a obtenção de informações mais detalhadas sobre o consumo de energia do veículo.

Com a integração do joulímetro ao sistema, foi identificada a necessidade de coletar os dados em tempo real para análises do comportamento do veículo. Para atender a essa de-

[1] Mais informações sobre a equipe Tubarão Branco podem ser encontradas no site da UTFPR e também no LinkedIn

manda, foi decidido implementar um sistema de transmissão de dados sem fio de longo alcance. Essa escolha se baseou na necessidade de obter as informações dos dispositivos presentes no protótipo de forma contínua e remota, permitindo uma monitorização mais eficaz e em tempo real do desempenho do veículo. A utilização de um sistema de transmissão sem fio proporciona maior flexibilidade e mobilidade, permitindo o acesso e a análise dos dados em qualquer lugar dentro do alcance da rede sem fio.

1.1 Objetivos

As atividades para o desenvolvimento do monitoramento remoto do protótipo de veículo elétrico foram desdobradas e seus objetivos são apresentados neste capítulo.

1.1.1 Objetivo geral

Desenvolver um sistema de telemetria para um protótipo de veículo elétrico utilizando uma rede CAN, para realizar a comunicação entre os diversos módulos do veículo, e um enlace sem fio para transmitir os dados de *duty cyle*, tensão e corrente da bateria para um computador que estará executando o *software* de telemetria.

1.1.2 Objetivos específicos

As atividades para o desenvolvimento deste trabalho estão descritas a seguir.

1. Desenvolver um gateway capaz de traduzir as mensagens da rede de comunicação CAN para *Long Range* (LoRa).
2. Implementar o dispositivo capaz de receber as mensagens LoRa e encaminhar ao *software* de telemetria por uma conexão *Universal Serial Bus* (USB).
3. Efetuar o tratamento dos dados recebidos utilizando gráficos gerados por *software*.

1.2 Estrutura do trabalho

Nos capítulos subsequentes deste trabalho, serão apresentados os conceitos teóricos e práticos fundamentais para compreensão e desenvolvimento do projeto.

No Capítulo 2, intitulado como "Referencial Teórico", serão abordados os conceitos do modelo *Open System Interconnection* (OSI), o protocolo CAN e a tecnologia LoRa. Será fornecido um embasamento teórico sobre esses temas, a fim de proporcionar uma melhor compreensão das bases utilizadas no projeto.

Em seguida, no Capítulo 3, intitulado como "Arquitetura e Materiais", será descrita a metodologia adotada para a realização do projeto, bem como a arquitetura de solução proposta. Serão apresentados os passos e procedimentos seguidos na implementação do sistema de monitoramento remoto para o veículo elétrico.

Por fim, no Capítulo 4, intitulado "Resultados", será apresentada a implementação do projeto, descrevendo as soluções técnicas adotadas, os resultados obtidos e as análises realizadas.

2 REFERENCIAL TEÓRICO

A seguir serão apresentados alguns dos principais conceitos e características do protocolo CAN e da tecnologia LoRa. O protocolo CAN é um padrão de comunicação serial utilizado em redes veiculares, industriais e de automação, que oferece alta confiabilidade e tolerância a falhas. A tecnologia LoRa é uma alternativa de comunicação sem fio de longa distância e baixo consumo de energia, que tem sido amplamente utilizada em aplicações de *Internet of Things* (IoT).

2.1 Camadas *Open System Interconnection* (OSI)

Segundo (KUMAR; DALAL; DIXIT, 2014), o modelo OSI é um modelo conceitual que padroniza os sistemas computacionais de comunicação, tendo seu início no final decada de 70 pela Organização Internacional de Padronização (ISO) e pelo Comitê Consultivo Internacional de Telégrafo e Telefone (CCITT) (que vem da tradução do título em francês). Este modelo de referência é composto por sete diferentes camadas de abstração, sendo as camadas: Física, Enlace de Dados, Rede, Transporte, Sessão, Apresentação e Aplicação.

Cada camada possui uma função específica na transmissão de dados entre dispositivos de rede.

- Camada Física: Esta camada lida com a transmissão física dos dados através do meio de comunicação. Ela define as especificações elétricas, mecânicas e funcionais da conexão física entre dispositivos de rede.
- Camada de Enlace de Dados: Responsável pela comunicação confiável entre dispositivos adjacentes. Ela segmenta os dados em quadros, adiciona informações de controle de erro e controle de fluxo, e lida com o acesso ao meio compartilhado.
- Camada de Rede: Gerencia o roteamento dos dados através de uma rede, selecionando caminhos eficientes e encaminhando pacotes entre redes diferentes.
- Camada de Transporte: Garante a entrega confiável dos dados entre os dispositivos finais. Ela segmenta os dados em unidades menores, adiciona informações de controle de fluxo e controle de erros.
- Camada de Sessão: Estabelece, gerencia e finaliza as conexões entre aplicativos. Ela sincroniza as comunicações, permite a recuperação de falhas e o reestabelecimento de sessões interrompidas.
- Camada de Apresentação: Lida com a formatação e representação dos dados para que possam ser compreendidos pelos aplicativos. Ela realiza tarefas como tradução de formatos, compressão e criptografia.

- Camada de Aplicação: Fornece serviços de rede diretamente aos aplicativos e usuários finais, permitindo a comunicação entre os aplicativos em diferentes dispositivos.

Figura 1 – Camadas do modelo OSI

7	Aplicação
6	Apresentação
5	Sessão
4	Transporte
3	Rede
2	Enlace
1	Física

Fonte: (KUMAR; DALAL; DIXIT, 2014).

De acordo com (BOSCH, 1991) a CAN utiliza apenas as camadas Física e Enlace e de acordo com (AUGUSTIN *et al.*, 2016) a LoRa também utiliza apenas as camadas Física e Enlace, portanto neste referencial teórico serão apresentadas apenas as duas camadas de forma específica para cada tecnologia de comunicação nas seções 2.2 para a CAN e 2.3 para a LoRa.

2.2 Protocolo de comunicação CAN

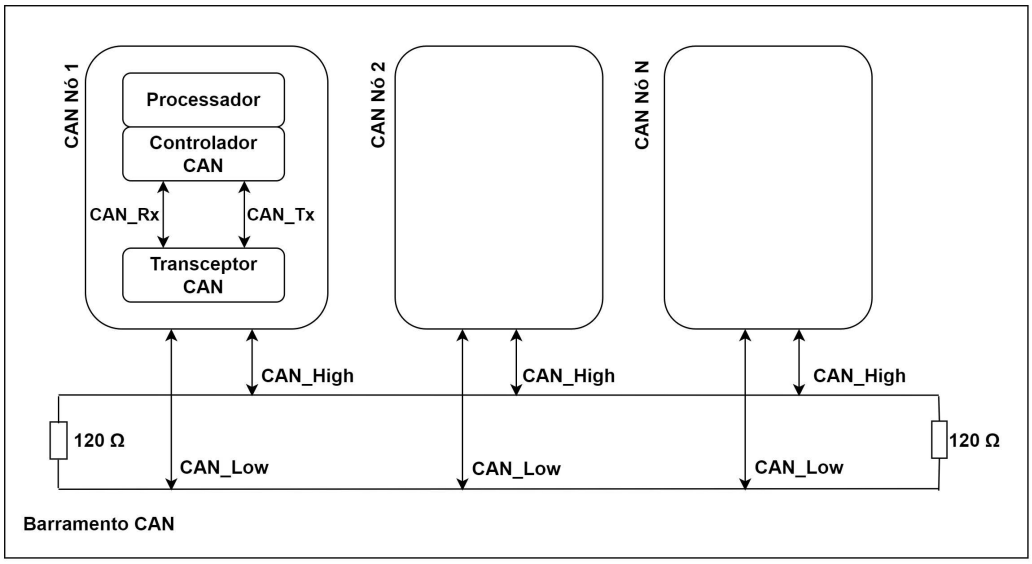
A CAN utiliza um par trançado de cabos elétricos para transmitir as informações. Os cabos *CAN_High*, *CAN_Low* e um resistor de 120 Ω conectado em cada extremidade formam o barramento CAN (TI, 2016).

Os sistemas responsáveis por enviar e receber pacotes de dados através do barramento CAN são denominados *Nós*. De acordo com Bosch (1991), um *Nó* é composto por um processador, um controlador CAN e um transceptor. Os *Nós* se conectam ao barramento CAN através dos cabos *CAN_High* e *CAN_Low*. O barramento CAN e os *Nós* estão representados na Figura 2.

O processador é o componente central da rede, responsável por executar as tarefas de processamento de dados e gerenciar o fluxo de informação entre os dispositivos da rede CAN.

O controlador CAN é responsável por gerenciar a comunicação entre os dispositivos na rede, realizando a filtragem e o armazenamento de mensagens que estão sendo transmitidas pelo barramento, detecta erros e implementa o padrão CAN nas informações tratadas anteriormente pelo processador (BOSCH, 1991).

Figura 2 – Topologia da Rede CAN



Fonte: TI (2016).

No armazenamento, o controlador identifica todos os pacotes de dados que passam pelo barramento e os guarda em uma pequena memória *First In First Out* (FIFO). Em seguida, notifica o processador sobre a disponibilidade dos dados para que seja possível tratar as informações (BOSCH, 1991).

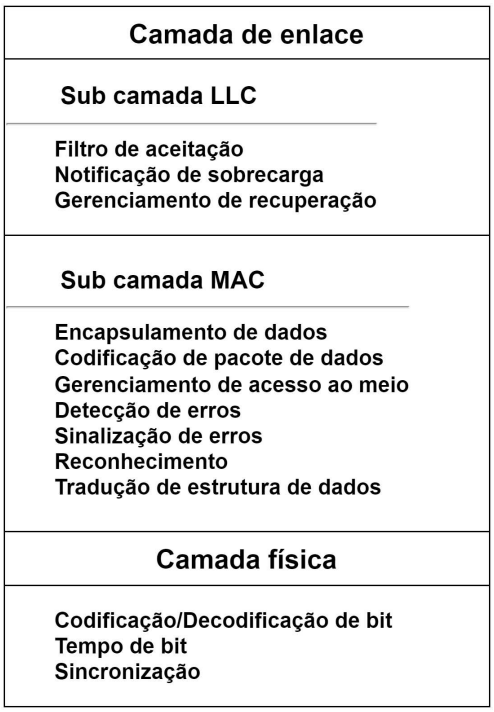
Na filtragem de mensagens, o filtro no *hardware* do controlador é configurado para que selecione as mensagens desejadas e descarte as indesejadas, sendo assim, o processador trata apenas as informações relevantes (BOSCH, 1991).

O transceptor converte um sinal digital em um sinal diferencial e está diretamente ligado ao barramento CAN através dos pinos *CAN_Low* e *CAN_High*. O terminal *CAN_High* é conectado ao barramento *CAN Bus High Line* e o terminal *CAN_Low*, à linha do barramento *CAN Bus Low Line*. Para adicionar outros nós ao barramento é necessário conectá-los da mesma maneira (BOSCH, 1991).

2.2.1 Camadas do protocolo de comunicação CAN

A rede CAN utiliza uma arquitetura em camadas que segue o modelo de referência OSI. De acordo com Bosch (1991), a CAN utiliza apenas as duas camadas mais baixas do modelo de referência OSI: camada física e camada de enlace de dados, conforme apresentado na Figura 3. Sendo a camada de enlace dividida em duas subcamadas: Controle de acesso ao meio (do inglês, *Medium Access Control* (MAC)) e Controle de Link Lógico (do inglês, *Logical Link Control* (LLC)). As outras camadas podem ser implementadas por protocolos de mais alto nível, como por exemplo o *CANopen*.

Figura 3 – Camadas da CAN



Fonte: Bosch (1991).

2.2.2 Camada Física CAN

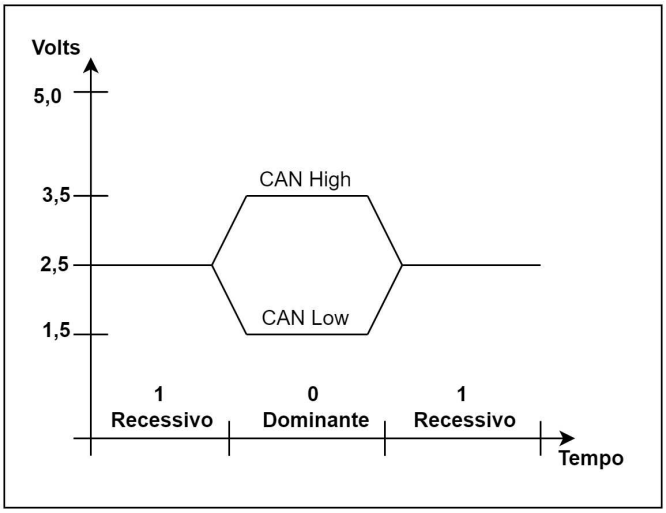
A camada física define o modo com que os dados serão transmitidos. De acordo com (BOSCH, 1991) é na camada física que são definidas as tensões para os *bits* dominante e recessivo, impedância do cabo e o esquema de sincronização. O barramento é composto por um par de cabos trançados, podendo ser também, um par liso. Porém, no segundo caso o barramento se torna mais sensível a ruídos.

O barramento é compartilhado com todos os nós, o tempo para transmissão de *bit* deve ser suficientemente grande para que a transmissão da mensagem chegue ao destinatário e volte ao remetente a notificação de que a mensagem foi recebida.

O protocolo CAN é um barramento serial assíncrono com codificação de *bits Non Return to Zero* (NRZ), isto é, os *bits* são representados por dois níveis de diferença de tensão entre o *CAN_High* e o *CAN_Low*. O *bit 0* (um) é representado por uma diferença de tensão positiva e é denominado como *bit* dominante. O *bit 1* (zero) é representado pela diferença de tensão zero e é denominado como *bit* recessivo. Dessa forma, não existe uma condição neutra, conforme apresentado na Figura 4 (RICHARDS, 2002).

A rede CAN, segundo (BOSCH, 1991), é uma rede de comunicação serial que utiliza a estratégia de transmissão de dados *multicast* com sincronização de tempo, isto é, a mensagem é entregue a todos os destinatários simultaneamente. Caso haja algum erro no barramento, nenhuma mensagem será recebida.

Figura 4 – Codificação/decodificação de bits recessivo e dominante



Fonte: (CORRIGAN, 2016).

2.2.3 Sincronização, taxa de transmissão de *bits* e tempo de *bit*

A sincronização e a taxa de transmissão de *bits* são aspectos fundamentais para o funcionamento do protocolo CAN. Para estabelecer a sincronização entre transmissor e receptor, é necessário que a velocidade de transporte de dados, taxa de *bits*, seja homogênea e constante em todo o sistema, bem como, o envio de um *bit* por um dispositivo e recebido por todos os dispositivos na rede para sinalizar o início da transmissão de dados (BOSCH, 1991).

No protocolo CAN, a taxa de *bits* pode ser ajustada para diferentes valores, sendo que valores mais altos, com valor máximo de 1 Mbit/s, permitem uma transmissão mais rápida, porém com menor alcance. Por outro lado, valores mais baixos permitem uma transmissão mais lenta, mas com maior alcance (BOSCH, 1991).

A taxa de transmissão de *bits* é definido como o número de *bits* por segundo que foram transmitidos pelo barramento. O cálculo da taxa de *bits* está apresentado na Equação 1 (RICHARDS, 2002).

$$f_{bit} = \frac{1}{t_{bit}} \tag{1}$$

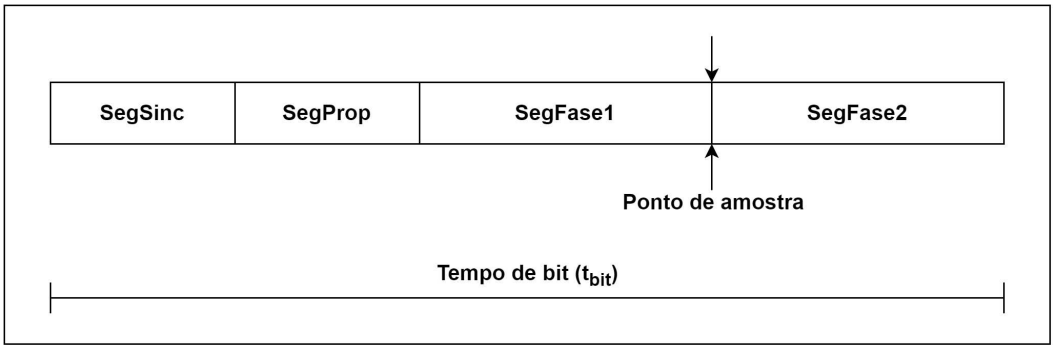
Sendo, f_{bit} a frequência nominal da taxa de *bits* e t_{bit} o período nominal do tempo de *bit*, que é calculado a partir da Equação 2 (RICHARDS, 2002).

$$t_{bit} = t_{SegSinc} + t_{SegProp} + t_{SF1} + t_{SF2} \tag{2}$$

O tempo de *bit* é dividido em quatro partes e o ponto de amostragem está localizado no final do segmento de fase 1 (RICHARDS, 2002). Na Figura 5 estão representados os segmentos do tempo de *bit*.

- O segmento de sincronização ($t_{SegSinc}$) é o primeiro segmento do tempo de *bit* e é responsável pela sincronização dos *Nós* no barramento.
- O segmento de propagação ($t_{SegProp}$) corresponde ao tempo que leva para o sinal elétrico percorrer a distância entre os dispositivos na rede.
- O segmento de fase 1 (t_{SF1}) corresponde ao tempo que leva para transmitir um bit de informação na rede.
- O segmento de fase 2 (t_{SF2}) é utilizado para garantir que haja tempo suficiente entre as transmissões de mensagens na rede.

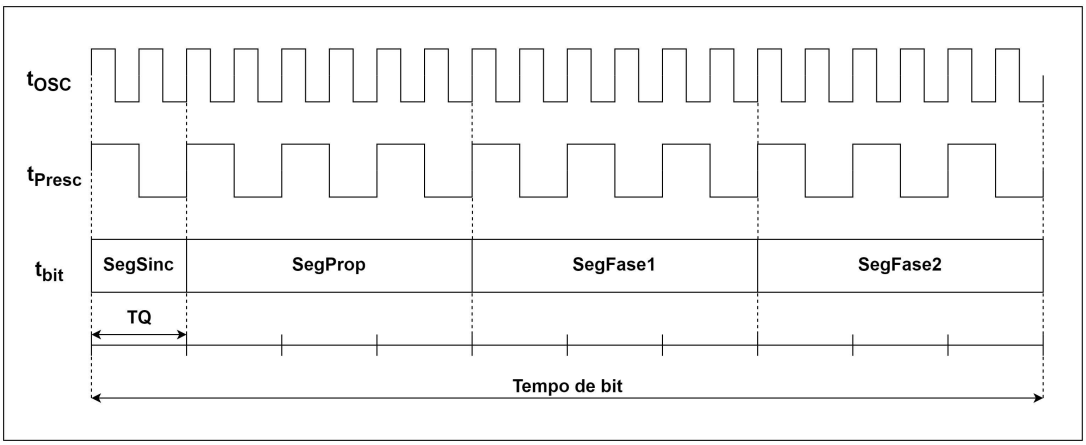
Figura 5 – Segmentos do tempo de bit



Fonte: (RICHARDS, 2002).

Os segmentos que constituem o tempo de *bit* são formados por unidades inteiras denominadas de Tempo Quanta (TQ). O segmento de sincronização possuí um valor fixo de 1 TQ. O segmento de propagação pode ser configurado de 1 a 8 TQ. O segmento de fase 1 pode ser configurado de 1 a 16 TQ e o segmento de fase 2 de 2 a 8 TQ (RICHARDS, 2002). A Figura 6 apresenta a relação entre o período do oscilador, o período do *prescaler*, o TQ e seus segmentos e o tempo de *bit*.

Figura 6 – Tempo quanta e tempo de bit



Fonte: (RICHARDS, 2002).

2.2.4 Tempo quanta

O TQ é um parâmetro importante no protocolo CAN que define a duração mínima de cada unidade de tempo na rede. Essa unidade de tempo é utilizada para determinar o tempo de transmissão de um *bit* de informação e é ajustada de acordo com a taxa de *bits* escolhida para a rede.

O tempo de duração de cada TQ é baseada no período do oscilador (t_{OSC}) e o *Prescaler*. A Equação 3 apresenta o cálculo do TQ (RICHARDS, 2002).

$$TQ = 2 * Prescaler * t_{OSC} \tag{3}$$

2.2.5 Subcamada Media Access Control (MAC)

A função da subcamada MAC é gerenciar o acesso ao meio físico de transmissão de dados da rede CAN. Ela realiza o controle dos quadros, a arbitragem dos quadros, a verificação de erros, a sinalização de erros e confinamento de falhas. Além disso, na subcamada MAC é realizada a verificação da disponibilidade de transmissão no barramento ou se foi iniciada uma recepção de dados (BOSCH, 1991).

2.2.6 Quadros de dados

No protocolo CAN existem quatro tipos de quadros que podem ser transmitidos pelo barramento, sendo eles: quadro de dados, quadro remoto, quadro de erro e quadro de sobrecarga.

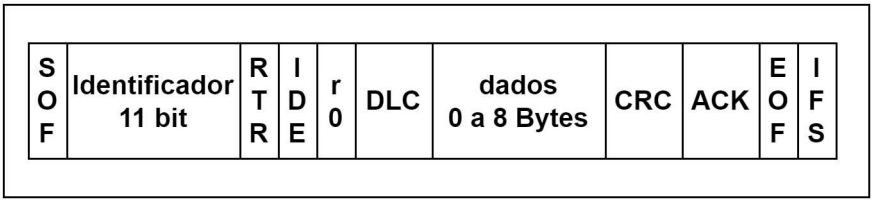
Também existem dois tipos de formatos de quadros, sendo o quadro padrão com um identificador de 11 *bits* e o quadro extendido com um identificador de 29 *bits* (BOSCH, 1991). A Figura 7 representa o quadro de dados com identificador de 11 *bits*.

De acordo com (CORRIGAN, 2016), o quadro de dados com o identificador de 11 *bits* possui diferentes campos tais como:

- O *bit* de inicio de quadro (do inglês, *Start of frame* (SOF)) é utilizado sinalizar o início de um quadro e sincroniza os nós do barramento.
- O Identificador de 11 *bits* é utilizado para estabelecer a prioridade da mensagem. Quanto menor o valor, maior é sua prioridade.
- O *bit* de solicitação de transmissão remota (do inglês, *Remote transmission request* (RTR)) é dominante quando as informações são solicitadas de outro nó.
- O *bit* de extensão de identificador único (do inglês, *Identifier extension* (IDE)) dominante determina que o quadro de dados utilizará o identificador de 11 *bits*.

- O *Bit reservado (r0)* é utilizado para futuras emendas padrão.
- O código de comprimento de dados (do inglês, *Data length code* (DLC)) contém o número de bytes que serão transmitidos.
- A verificação de redundância cíclica (do inglês, *Cyclic redundancy check* (CRC)) contém dados do segmento anterior para detecção de erro.
- O *Acknowledges* (ACK) é um bit para notificar o transmissor de que a mensagem foi recebida sem erros. Os *nós* que receberam a mensagem sobrescrevem este segmento com um *bit* dominante.
- O campo de fim de quadro (do inglês, *End-of-frame* (EOF)) de 7 *bits* indica o fim do quadro de dados.
- O espaço interquadro (do inglês, *Interframe space* (IFS)) é o tempo necessário para que a mensagem seja transmitida até o *buffer* do receptor.

Figura 7 – Quadro de dados com identificador de de 11 *bits*



Fonte: (CORRIGAN, 2016).

2.2.7 Arbitração dos *Nós*

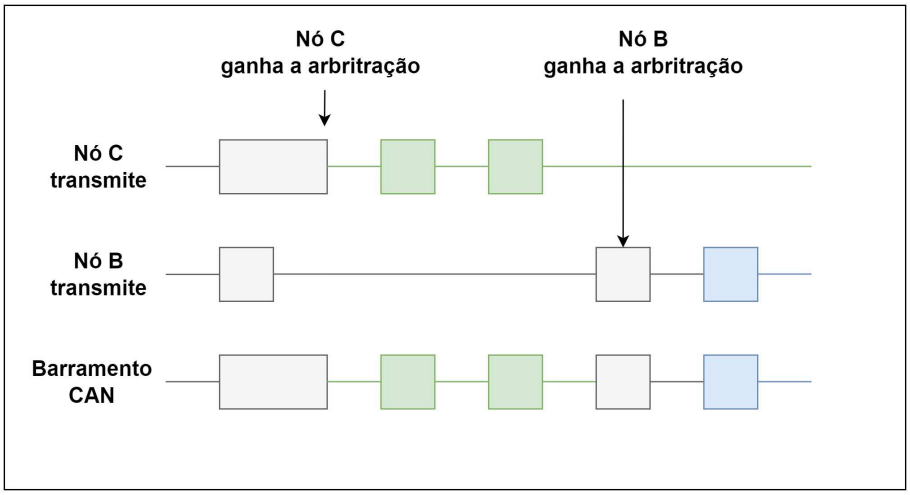
De acordo com (BOSCH, 1991), a CAN é um sistema multimestre, isto é, quando o barramento não está transmitindo dados, qualquer nó pode se tornar mestre e enviar uma mensagem pelo barramento, consequentemente, os outros nós tornam-se escravos e a recebem. Para isso, faz-se necessário um sistema de arbitração, que tem como base a detecção do *bit* dominante.

A arbitração é um processo importante no protocolo CAN que determina qual mensagem deve ser transmitida na rede em caso de conflito de mensagens. Quando vários dispositivos na rede tentam transmitir mensagens ao mesmo tempo, pode ocorrer uma colisão de mensagens. A arbitração é o processo pelo qual os dispositivos na rede CAN decidem qual mensagem deve ter prioridade de transmissão.

A Figura 8 exemplifica a arbitração. Cada *nó* monitora suas próprias transmissões, conforme o *bit* recessivo do *nó B* é substituído pelo *bit* dominante de prioridade mais alta do *nó C*, *B* detecta que o estado do barramento não corresponde ao *bit* que ele transmitiu, portanto,

o *nó B* interrompe a transmissão e o *nó C* continua a transmissão de sua mensagem. Outra tentativa de transmitir a mensagem é feita pelo *nó B* uma vez que o barramento é liberado pelo *nó C* (CORRIGAN, 2016).

Figura 8 – Arbitração dos *nós*



Fonte: (CORRIGAN, 2016).

2.2.8 Subcamada Logical Link Control (LLC)

A subcamada LLC é uma das subcamadas presentes na camada de enlace de dados do protocolo CAN e tem como função garantir que as mensagens transmitidas na rede sejam entregues corretamente e que a integridade dos dados seja mantida.

É de responsabilidade da subcamada LLC implementar os protocolos de controle de erro, como o CRC, que verifica se as mensagens foram transmitidas sem erros, e o ACK, que confirma que a mensagem foi recebida corretamente pelo destinatário (BOSCH, 1991).

Além disso, é responsável pela segmentação e reagrupamento de mensagens maiores em unidades menores, garantindo que as mensagens sejam transmitidas de forma eficiente e sem perda de informações.

2.2.9 Filtros de aceitação

O controlador CAN contém um filtro de aceitação de *hardware* (FIFO) que é utilizado para filtrar os pacotes enviados através do barramento CAN. Os filtros de aceitação tornam um *nó* mais eficiente, pois apenas os pacotes que são relevantes ao *nó* são utilizados pelo processador (BOSCH, 1991).

O filtro de aceitação do controlador CAN é configurado usando dois valores de 32 bits conhecidos como código de aceitação e máscara de aceitação (BOSCH, 1991).

O código de aceitação determina a sequência de *bits* do identificador que serão aceitos pelo filtro de aceitação. A máscara de aceitação é uma sequência de *bits* que determina quais *bits* do código de aceitação podem ser ignorados. Desta forma, pacotes com diferentes identificadores podem ser aceitos por um único código de aceitação (BOSCH, 1991).

2.2.10 Considerações finais sobre o protocolo CAN

Em suma, o protocolo CAN demonstrou ser uma solução eficiente e confiável para a comunicação entre dispositivos internos ao veículo. Sua capacidade de transmitir dados de forma robusta, mesmo em ambientes com interferências eletromagnéticas, torna-o amplamente adotado na indústria automotiva e em aplicações industriais críticas.

A arquitetura do protocolo CAN permite a conexão de múltiplos dispositivos em um barramento compartilhado, simplificando a comunicação e reduzindo a complexidade do sistema. Com sua ampla adoção e interoperabilidade entre diferentes fabricantes, o protocolo CAN se tornou um padrão global na comunicação de dados em sistemas embarcados, oferecendo confiabilidade, escalabilidade e eficiência para uma ampla gama de aplicações.

2.3 Tecnologia LoRa

A tecnologia LoRa é uma tecnologia de comunicação sem fio de longo alcance e baixa potência, desenvolvida para conectar dispositivos à IoT. Ela utiliza um esquema de modulação de espectro espalhado para enviar dados de forma eficiente e confiável por longas distâncias, usando baixa potência (AUGUSTIN *et al.*, 2016).

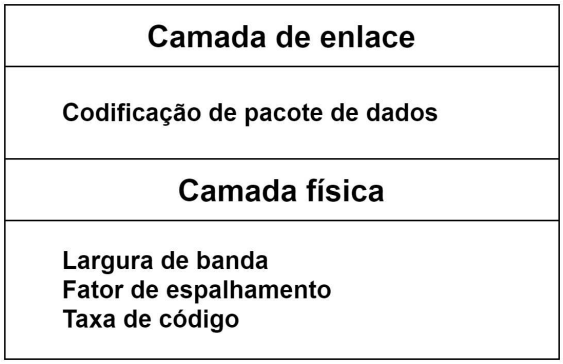
A LoRa oferece uma solução eficiente e escalável para conectar dispositivos IoT, com alcance de até vários quilômetros em ambientes urbanos e mais de dez quilômetros em áreas rurais(AUGUSTIN *et al.*, 2016).

2.3.1 Camadas da tecnologia LoRa

Na tecnologia LoRa é utilizado o modelo de referência de redes em camadas OSI. De acordo com (AUGUSTIN *et al.*, 2016) a LoRa utiliza apenas as duas camadas mais baixas do modelo de referência OSI: camada física e camada de enlace de dados, conforme apresentado na Figura 9. Protocolos como *LoRaWAN* podem implementar as camadas mais altas do modelo de referência OSI.

A camada física é responsável por transmitir e receber sinais de rádiofrequência, enquanto a camada de enlace de dados é responsável por fornecer um meio de acesso ao canal de comunicação para dispositivos finais e garantir a integridade dos dados transmitidos.

Figura 9 – camadas da LoRa



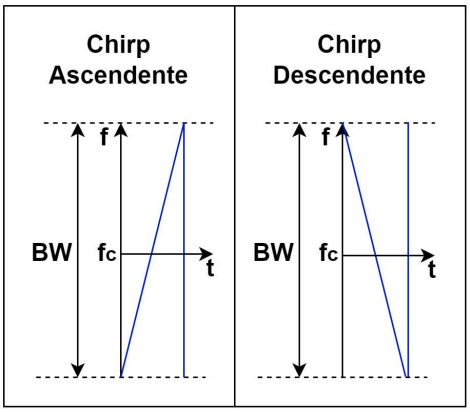
Fonte: (SEMTECH, 2019).

2.3.2 Parâmetros da LoRa

A LoRa utiliza a modulação de sinal *Chirp Spread Spectrum* (CSS), que é baseado em ondas senoidais que variam sua frequência ao longo do tempo de forma não linear, criando um sinal conhecido como *chirp*. Um pulso no qual a frequência aumenta é denominado de *chirp* ascendente e um pulso no qual a frequência diminui é denominado *chirp* descendente. A Figura 10 exemplifica os dois tipos de *chirps* (AUGUSTIN *et al.*, 2016).

A largura de banda (do inglês, *Bandwidth* (BW)), é a faixa de frequência disponível para uma transmissão de dados em um sistema de comunicação sem fio. Na tecnologia LoRa, a largura de banda está diretamente relacionada à taxa de dados e ao alcance da comunicação.

Figura 10 – Chirp Ascendente e chirp descendente



Fonte: (AUGUSTIN *et al.*, 2016).

O *Coding Rating* (CR), ou taxa de codificação, é uma medida da eficiência de transmissão de dados em um sistema de comunicação sem fio. É utilizado para indicar a quantidade de redundância adicionada aos dados antes da transmissão, a fim de melhorar a confiabilidade da comunicação.

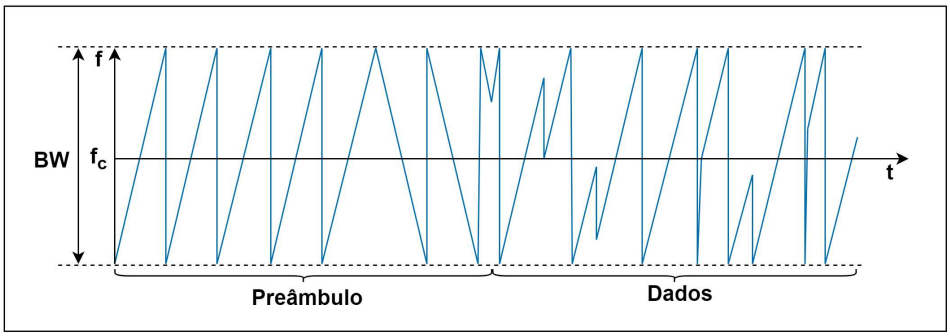
Um CR maior significa que mais redundância é adicionada aos dados, tornando-os mais resistentes a erros, mas também aumentando o tempo de transmissão e reduzindo a taxa de

dados. Por outro lado, um CR menor resulta em uma taxa de dados mais alta, mas pode tornar a comunicação mais suscetível a erros.

2.3.3 Símbolos LoRa

Para realizar a transmissão são enviados símbolos LoRa, estes símbolos são composto de 2^{SF} *chirps*. Sendo o fator de espalhamento (do inglês, *Spreading Factor* (SF)) a velocidade na qual a frequência do sinal muda ao longo da BW de um canal. Cada símbolo é determinado por suas descontinuidades no formato dos tipos de *chirps* (AUGUSTIN *et al.*, 2016). Tais descontinuidades estão exemplificadas na Figura 11.

Figura 11 – Símbolos LoRa



Fonte: (AUGUSTIN *et al.*, 2016).

A taxa de símbolos e a taxa de bits em um determinado SF são proporcionais à BW, a Equação 4 apresenta o cálculo do período de um símbolo (T_s).

$$T_s = \frac{2^{SF}}{BW} \tag{4}$$

2.3.4 Pacote de dados LoRa

Um pacote de dados é a unidade básica de informação transmitida entre os dispositivos de comunicação, nele contém informações sobre o transmissor, o receptor e os dados a serem transmitidos.

Cada pacote de dados inclui um cabeçalho, que contém informações sobre a origem e o destino do pacote, e os próprios dados a serem transmitidos, que são protegidos por uma técnica de codificação de correção de erro. A Figura 12 apresenta a estrutura de um pacote LoRa.

A taxa de transmissão de pacotes LoRa é ajustável, permitindo que os dispositivos sejam otimizados para diferentes aplicações.

O preâmbulo começa com uma sequência de *chirps* ascendentes constantes que cobrem toda a banda de frequência. Os dois últimos *chirps* ascendentes codificam a palavra de

sincronização. A duração total deste preâmbulo pode ser configurada entre 10,25 e 65.539,25 símbolos.Conforme apresentado na Figura 11.

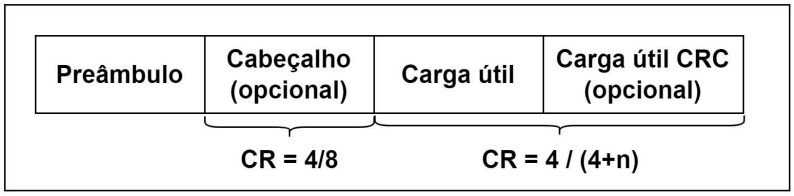
A palavra de sincronização é um valor de um *byte* usado para diferenciar as redes LoRa que usam as mesmas bandas de frequência. Pois o dispositivo apenas irá decodificar os sinais que correspondem à palavra de sincronização configurada.

Após a palavra de sincronização são enviados dois *chirps* descendentes e mais um quarto de um *chirp* descendente.

O cabeçalho é um elemento opcional presente apenas quando se deseja apresentar o tamanho da carga útil em *bytes* e o CRC. Quando habilitado é transmitido com uma CR de 4/8. O cabeçalho também inclui um CRC para permitir que o receptor descarte pacotes com cabeçalhos inválidos (AUGUSTIN *et al.*, 2016).

A carga útil é enviada após o cabeçalho, sendo esta limitada a 255 *bytes*. No final do quadro está o CRC que é um elemento opcional.

Figura 12 – Estrutura do pacote LoRa



Fonte: (AUGUSTIN *et al.*, 2016).

A quantidade de símbolos necessários para realizar a transmissão da carga útil do pacote (n_s) é calculada através da Equação 5. O *PL* é o tamanho da carga útil em bytes. O CRC é 16 se estiver habilitado e zero caso contrário. O *H* é 20 quando o cabeçalho está habilitado e zero caso contrário. O *DE* é 2 quando a otimização de baixa taxa de dados está habilitada e zero caso contrário (AUGUSTIN *et al.*, 2016).

$$n_s = 8 + max\left(\left[\frac{8PL - 4SF + 8 + CRC + H}{4 * (SF - DE)}\right] * \frac{4}{CR}, 0\right) \tag{5}$$

2.3.5 Regulamentação das bandas de frequências da LoRa

As especificações da LoRaWAN são documentos desenvolvidos e mantidos pela LoRa *Alliance*, e variam de acordo com as regiões globais, seguindo as restrições reguladas por essas regiões.

A Agência Nacional de Telecomunicações (Anatel) é responsável por regular as frequências de comunicação utilizadas pela tecnologia LoRa no Brasil. A faixa de frequência autorizada para LoRa é de 915-928 MHz, na qual são permitidas transmissões com potência máxima de até 500 mW. Além disso, a Anatel exige que os dispositivos que utilizam essa faixa de frequência estejam certificados pela agência, a fim de garantir a conformidade com as normas e evitar interferências prejudiciais a outros sistemas de comunicação. A regulamentação da Anatel busca

Tabela 1 – Bandas de frequência regulamentadas

Nomenclatura	Bandas de frequência (MHz)
AS923	923
AU915	915-928
CN470	470-510
CN779	779-787
EU433	433
EU868	863-870
KR920	920-923
IN865	865-867
RU864	864-870
US915	902-928

Fonte: (ALLIANCE, 2018b).

assegurar que a tecnologia LoRa seja utilizada de maneira segura e eficiente no Brasil (ANATEL, 2017).

2.3.6 Considerações finais sobre a tecnologia LoRa

Com base nos conceitos apresentados nesse capítulo, é viável estabelecer uma estratégia de desenvolvimento para o projeto do sistema de telemetria. Essa estratégia será fundamentada na aplicação do protocolo CAN e da tecnologia LoRa, os quais desempenham papéis essenciais na comunicação e transmissão de dados. A utilização do protocolo CAN permitirá a comunicação eficiente entre os dispositivos presentes no veículo, enquanto a tecnologia LoRa proporcionará a transmissão de dados de forma sem fio, viabilizando a monitorização remota. A combinação dessas duas tecnologias possibilitará o desenvolvimento de um sistema de telemetria robusto e abrangente.

Com a tecnologia LoRa é possível ter diversos dispositivos finais e um gateway que recebe todos os dados destes dispositivos, porém este tipo de arquitetura não é utilizado neste trabalho, pois este projeto será integrado e posteriormente evoluido pela equipe Tubarão Branco. Vale destacar que, como parte da estratégia de desenvolvimento do protótipo, a equipe opta por realizar o desenvolvimento de novos dispositivos de forma individual e posteriormente integrá-lo com os outros disposivos, se possível, posteriormente.

Com a utilização da tecnologia LoRa, é possível empregar múltiplos dispositivos finais e um *gateway* que recebe os dados provenientes desses dispositivos. No entanto, a arquitetura mencionada não é adotada neste projeto, pois este trabalho será integrado ao protótipo e futuramente será aprimorado pela equipe Tubarão Branco. Cabe ressaltar que, como parte da estratégia para desenvolvimento do protótipo, a equipe opta por realizar a elaboração de novos dispositivos à parte dos existentes, com a intenção de integrá-los posteriormente com os demais dispositivos, sempre que possível.

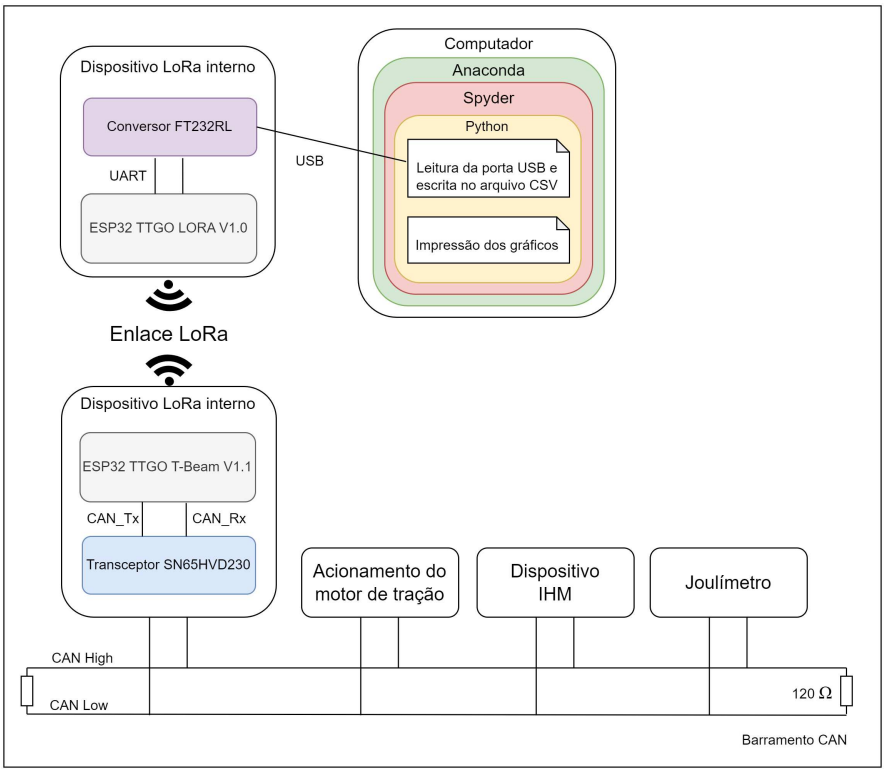
3 ARQUITETURA E MATERIAIS

A seguir serão apresentadas informações sobre os dispositivos que existem no protótipo do veículo elétrico, dispositivo interno, dispositivo externo e a apresentação dos dados. Também serão apresentados os microcontroladores, transceptor CAN, módulo LoRa e softwares utilizados durante o desenvolvimento do projeto.

3.1 Arquitetura da solução proposta

A arquitetura da solução proposta baseia-se na utilização de um sistema já presente no protótipo do veículo e está representada na Figura 13. Vale ressaltar que o desenvolvimento dos dispositivos Acionamento do motor de tração, dispotivo IHM e o Joulímetro não fazem parte do escopo do projeto.

Figura 13 – Arquitetura da solução proposta



Fonte: Autoria própria (2023).

Os dispositivos que estão instalado no veículo são: Joulímetro, dispositivo IHM e acionamento do motor de tração.

O joulímetro realiza a leitura da tensão e corrente da bateria do veículo e envia os dados através da rede CAN. O dispositivo IHM Realiza a leitura de um potênciometro, sendo este, o acelerador do veículo e envia os dados através da CAN para o acionamento do motor de tração. O acionamento do motor de tração realiza o controle do motor de tração do veículo através dos dados enviados pelo Dispositivo IHM.

Este projeto tem como objetivo o desenvolvimento de três componentes principais: o dispositivo LoRa interno, o dispositivo LoRa externo e um software para a representação gráfica dos dados.

Para efetuar o desenvolvimento do *software* proposto do dispositivo interno e dispositivo externo, emprega-se a IDE VsCode juntamente com a extensão *Espressif-IDF*, uma ferramenta utilizada para programação de software destinado aos microcontroladores ESP32. Os detalhes específicos da implementação do sistema podem ser encontrados no Capítulo 4.

O dispositivo LoRa interno tem a finalidade de ler os dados presentes no barramento CAN existente e transmiti-los através do enlace LoRa. O *software* proposto para o dispositivo interno é dividido em duas tarefas distintas. A primeira tarefa é responsável pela leitura do barramento CAN e identificação da origem de cada pacote de dados. A segunda tarefa é encarregada de transmitir os dados recebidos pelo barramento CAN utilizando o enlace LoRa. O tempo de execução das tarefas é gerenciado pelo sistema operacional de tempo real *FreeRTOS* que utiliza o buffer *Queue* para enviar os dados de uma tarefa para a outra.

O dispositivo LoRa externo, por sua vez, tem como objetivo receber os dados transmitidos pelo enlace LoRa e enviá-los para um computador utilizando a comunicação serial USB. O software proposto para o dispositivo externo também é dividido em duas tarefas. A primeira tarefa realiza a leitura dos dados transmitidos através da tecnologia LoRa. A segunda tarefa é responsável por transmitir os dados recebidos pela LoRa para o computador por meio da comunicação USB. Para isso, utiliza-se uma placa conversora serial USB, onde os dados são enviados do microcontrolador para a placa conversora através da comunicação *Universal Asynchronous Receiver / Transmitter* (UART) e, em seguida, são convertidos para a comunicação USB. O tempo de execução das tarefas é gerenciado pelo sistema operacional de tempo real *FreeRTOS* que utiliza o buffer *Queue* para enviar os dados de uma tarefa para a outra.

Por fim, o componente do *software* no computador consiste no desenvolvimento de uma interface que possibilitará a representação gráfica dos dados transmitidos. Essa etapa é dividida em dois códigos em *Python*. O primeiro código realiza a leitura da porta serial e armazena os dados em um arquivo *Comma-Separated Values* (CSV). O segundo código lê esse arquivo CSV e os apresenta de forma gráfica, possibilitando a visualização dos dados de maneira mais intuitiva.

3.2 Materiais

A seguir serão apresentadas algumas características dos *hardwares* e *softwares* utilizados durante o desenvolvimento do projeto.

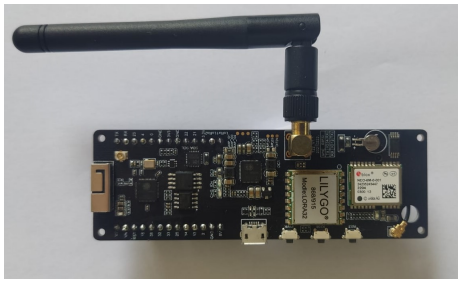
3.2.1 Hardware

1. **ESP32 TTGO T-Beam V1.1:** O kit de desenvolvimento *ESP32 TTGO T-Beam V1.1* é um dispositivo fabricado pela *LilYGO* que foi projetado para aplicações de IoT que exigem comunicação sem fio de longo alcance.

   O kit utiliza o microprocessador *MCU ESP32* e possui *Wi-Fi*, *Bluetooth* e um módulo LoRa *SX1276* integrado, que permite a comunicação de dados a longas distâncias. O dispositivo também possui um acelerômetro e um *Global Positioning System* (GPS), que permitem a coleta de dados de localização.

   O *T-Beam V1.1* pode ser alimentado por uma bateria recarregável de lítio e possui um carregador integrado, tornando-o adequado para uso em aplicações remotas ou móveis. Também possui uma variedade de interfaces, incluindo UART, *Inter-Integrated Circuit* (I2C) e *Serial Peripheral Interface* (SPI), que permitem a integração com outros dispositivos e sensores.

Figura 14 – Microcontrolador ESP32 TTGO T-Beam V1.1

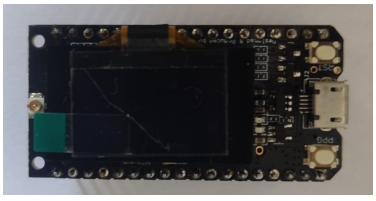


Fonte: Autoria própria (2023).

2. **ESP32 TTGO LORA V1.0:** O kit de desenvolvimento *ESP32 TTGO LORA V1.0* é um dispositivo fabricado pela *LilYGO* que permite a comunicação de longo alcance através da tecnologia LoRa.

   O kit utiliza o microprocessador *MCU ESP32* e possui *Wi-Fi*, *Bluetooth* e módulo de rádio LoRa *SX1276* integrado. O dispositivo também possui uma tela *OLED* integrada, permitindo a visualização de dados em tempo real. Também possui várias interfaces, incluindo UART, I2C e SPI, que permitem a integração com outros dispositivos e sensores.

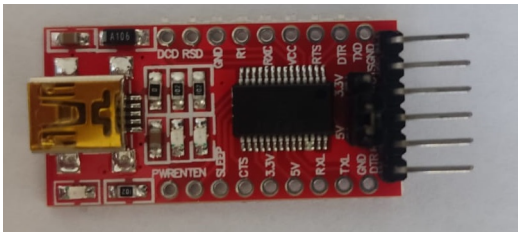
Figura 15 – Microcontrolador ESP32 TTGO LORA V1.0



Fonte: Autoria própria (2023).

3. **Conversor USB-serial FTDI FT232RL:** A placa *conversor USB-serial FTDI FT232RL* é um dispositivo que permite a comunicação serial entre um computador e um microcontrolador ou outro dispositivo eletrônico. O conversor é equipado com um chip *FTDI FT232RL* que converte o sinal USB em sinal UART, tornando-se ideal para programação e depuração de microcontroladores e dispositivos eletrônicos.

Figura 16 – Conversor USB Serial FTDI FT232RL



Fonte: Autoria própria (2023).

4. **Transceptor CAN SN65HVD230:** O transceptor CAN *SN65HVD230* é um dispositivo eletrônico que permite a transmissão e recepção de dados utilizando o protocolo CAN). O transceptor é projetado para suportar uma ampla gama de aplicações industriais e automotivas, oferecendo uma alta taxa de transmissão de dados de até 1 Mbps e uma ampla faixa de temperatura de operação de -40℃ a 125℃. O transceptor é alimentado por uma tensão de 3,3V ou 5V e apresenta proteção contra sobrecorrente, sobretensão e curto-circuito.

Figura 17 – Transceptor CAN SN65HVD230

Fonte: Autoria própria (2023).

3.2.2 Software

1. **VSCode:** O *Visual Studio Code* (VSCode) é um editor de código fonte desenvolvido pela *Microsoft*. É uma ferramenta que suporta uma grande variedade de linguagens de programação e *frameworks*. O VSCode tem se tornado cada vez mais popular entre desenvolvedores devido à sua interface intuitiva, facilidade de uso e uma ampla gama de extensões que adicionam recursos e funcionalidades ao editor.

2. **Extensão Espressif IDF:** A extensão *Espressif IDF* para o VSCode é uma ferramenta para desenvolvimento de *software* para os dispositivos da família *ESP32*. A extensão fornece recursos que permitem aos desenvolvedores criar, compilar e depurar seus

projetos diretamente no VSCode, além de suportar a configuração e a programação de dispositivos. A extensão também inclui a integração com as ferramentas de linha de comando do *Espressif* IoT *Development Framework (ESP-IDF)*, o que permite uma compilação rápida e fácil de projetos usando as bibliotecas e ferramentas fornecidas pela *Espressif*.

3. **Anaconda:** *Anaconda* é uma distribuição do *Python*, projetada para atender às necessidades de desenvolvedores de *software*. É uma plataforma de código aberto que inclui um gerenciador de pacotes e um IDE que facilita a instalação e gerenciamento de pacotes, bibliotecas e dependências.

4. **Spyder:** *Spyder* é um IDE que faz parte do pacote de *software Anaconda Python*. A IDE fornece uma interface gráfica de usuário (do inglês, *Graphical User Interface* (GUI)) para trabalhar com o *Python*, facilitando a escrita, depuração e teste de código. O *Spyder* inclui uma ampla gama de recursos, como realce de sintaxe, auto completar, inspeção de variáveis, plotagem de gráficos, suporte a múltiplos arquivos e integração com ferramentas de gerenciamento de pacotes, como o *Conda*.

A seguir serão descritos com maiores detalhes os materiais utilizados em cada disposito proposto.

3.2.3 Dispositivo LoRa Interno

Para o desenvolvimento do dispositivo interno, foi escolhido o kit de desenvolvimento *ESP32 TTGO T-Beam V1.1*, que possui diversos dispositivos integrados, como o microcontrolador *ESP32*, controlador CAN e o módulo LoRa *SX1276*. Essa escolha se deu principalmente pelo fato de que o kit já oferece suporte para as tecnologias necessárias ao projeto, além de contar com a certificação da Anatel para a utilização da frequência de 915 MHz, que é a faixa de operação permitida no Brasil para a tecnologia LoRa. Dessa forma, o kit se tornou uma opção conveniente e confiável para o desenvolvimento do dispositivo interno.

Além disso, o kit também possui um módulo *GPS NEO-6M*, que possibilita a realização de outras implementações além do escopo do projeto. A presença desse módulo pode ser utilizada para aprimorar a localização do dispositivo, o que pode ser útil em diversas aplicações.

3.2.4 Dispositivo LoRa Externo

Para o desenvolvimento do dispositivo externo, foi escolhido o kit de desenvolvimento *ESP32 TTGO LoRa V1.0*, assim como o *T-Beam V1.1* este kit também possui controlador CAN e o módulo LoRa *SX1276* certificado pela Anatel para a utilização da frequência de 915 MHz. Também está presente no kit um *display OLED* que permite a visualização dos dados.

Para realizar a transmissão dos dados recebidos através do enlace LoRa para o computador, optou-se por utilizar a placa conversora USB serial *FT232RL*. Essa placa permite a comunicação entre o microcontrolador do kit de desenvolvimento *ESP32 TTGO LoRa V1.0* e o computador através da porta USB, possibilitando a transferência de dados.

3.2.5 Computador

Para realizar o tratamento dos dados recebidos da placa conversora USB *FT232RL*, optou-se por utilizar a linguagem de programação *Python*. Essa escolha foi feita principalmente por causa da facilidade que a linguagem oferece para imprimir gráficos e também para realizar a leitura da porta USB.

Para desenvolver o código para o projeto em questão, optou-se por utilizar a IDE *Spyder*, que é uma das opções disponíveis na plataforma *Anaconda*. A escolha pela IDE *Spyder* se deu principalmente pela sua facilidade de uso e recursos que facilitam a escrita, depuração e teste de código *Python*. A plataforma *Anaconda* é amplamente utilizada em projetos de ciência de dados e aprendizado de máquina, além de fornecer um ambiente integrado para o gerenciamento de pacotes e dependências do *Python*.

4 DESENVOLVIMENTO E RESULTADOS

A seguir serão discutidos os principais aspectos relacionados à implementação do sistema, como a comunicação entre os dispositivos do veículo através do protocolo CAN, a transmissão remota dos dados coletados para um computador utilizando a tecnologia LoRa, além da impressão em forma gráfica dos dados recebidos.

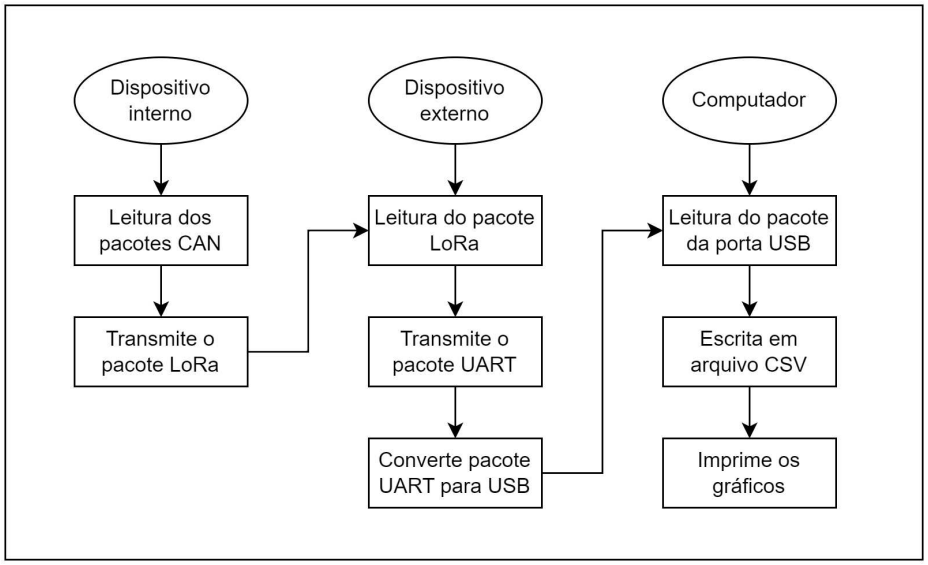
O dispositivo LoRa interno realiza a leitura dos dados através do barramento CAN. O dispositivo IHM fornece o valor do *duty cycle* em porcertagem. O Joulímetro fornece os dados de tensão e corrente da bateria. Após a leitura dos dados no barramento o dispositivo LoRa interno transmite as informações através da LoRa para o dispositivo LoRa externo.

O dispositivo LoRa externo realiza a leitura dos dados transmitidos pelo dispositivo LoRa interno via tecnologia LoRa. Em seguida transmite os dados utilizando comunicação UART para a placa *FT232RL*, esta por sua vez, transmite os dados utlizando a comunicação USB para um computador.

O computador, através da linguagem *Python*, realiza a leitura dos dados na porta serial USB transmitidos pelo dispositivo LoRa externo e os escreve em um arquivo CSV. Por fim, realiza a leitura do arquivo CSV e imprime os dados em um gráfico.

A Figura 18 apresenta o fluxograma das tarefas executadas pelos dispositivo LoRa interno, dispositivo LoRa externo e computador. As tarefas executadas pelos dispotivo interno e externo são gerenciadas pelo sistema operacional em tempo real *FreeRTOS*.

Figura 18 – Fluxograma das tarefas executadas pelos dispositivos



Fonte: Autoria própria (2023).

4.1 Implementação do dispositivo LoRa interno

A implementação do dispositivo LoRa interno foi dividida em cinco partes distintas. A primeira parte aborda a implementação do *hardware*, detalhando os componentes utilizados e a configuração física do dispositivo. A segunda parte apresenta as configurações da rede CAN no dispositivo. A terceira parte apresenta as configurações da LoRa. A quarta parte descreve as tarefas executadas pelo microcontrolador, incluindo a programação das funcionalidades e o controle do fluxo de dados. Por fim, a quinta parte apresenta a evidência da solução, demonstrando o resultado obtido e a validação do dispositivo LoRa.

Essa divisão estruturada do projeto permite uma análise mais detalhada de cada etapa e fornece uma visão abrangente da implementação do dipositivo LoRa interno.

4.1.1 Implementação do hardware do dispositivo interno

O dispositivo interno foi desenvolvido utilizando o kit de desenvolvimento *ESP32 TTGO T-Beam V1.1* que possui o módulo LoRa *SX1276* integrado ao kit. O mapeamento dos pinos utilizados pelo módulo LoRa *SX1276* no kit está detalhado na Tabela 2, que apresenta a correspondência entre os pinos físicos do kit e as funções específicas do módulo LoRa.

Tabela 2 – Mapeamento da conexão dos pinos do módulo LoRa e do kit de desenvolvimento

ESP32 TTGO T-Beam V1.1	Módulo LoRa SX1276
18	CS
14	RST
19	MISO
27	MOSI
5	SCK

Fonte: Autoria própria (2023).

Para realizar a comunicação com o barramento CAN, foi utilizado o transceptor CAN *SN65HVD230*. O mapeamento da conexão entre os pinos físicos do transceptor CAN e sua correspondência com os pinos específicos do kit de desenvolvimento está detalhado na Tabela 7.

Tabela 3 – Mapeamento da conexão dos pinos do transceptor CAN do kit de desenvolvimento

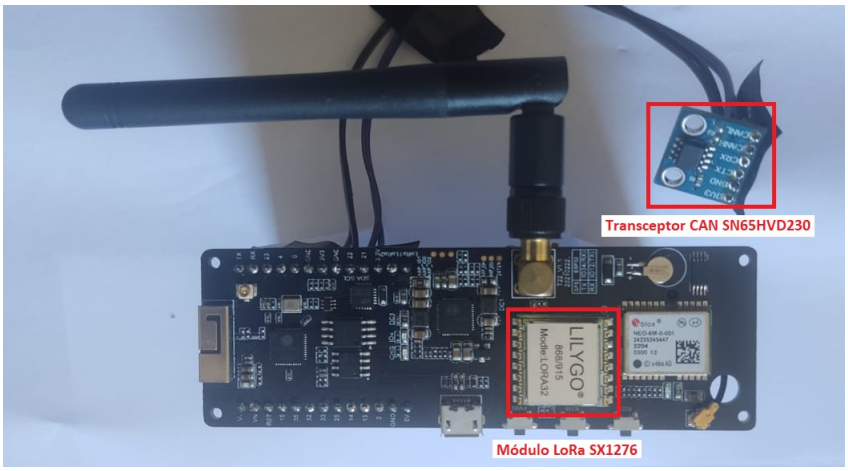
ESP32 TTGO T-Beam V1.1	Transceptor CAN SN65HVD230
3v3	3v3
GND	GND
21	CAN_Tx
22	CAN_Rx

Fonte: Autoria própria (2023).

A Figura 19 apresenta a montagem do dispositivo interno com o kit de desenvolvimento *ESP32 TTGO T-Beam V1.1*, módulo LoRa *SX1276* e o transceptor CAN *SN65HVD230*.

É importante ressaltar que este projeto não abrange a medição da posição do veículo por meio do GPS. O projeto se limita exclusivamente na leitura dos dados do barramento CAN e na transmissão desses dados para um computador através do enlace LoRa. A equipe Tubarão Branco poderá considerar a implementação do sistema de GPS em projetos futuros.

Figura 19 – Conexão dos pinos do dispositivo interno



Fonte: Autoria própria (2023).

4.1.2 Configuração da rede CAN no dispositivo interno

Para realizar a leitura do barramento CAN, é necessário configurar o dispositivo interno com os mesmos parâmetros utilizados no barramento CAN existente. Esses parâmetros são: o tempo quanta, o tamanho do segmento 1, o tamanho do segmento 2 e o ponto de amostragem.

O barramento opera com uma taxa de bits de $125kbit/s$. No microcontrolador *ESP32 TTGO T-Beam V1.1*, o clock dos periféricos é configurado para $80MHz$, portanto, para se obter um tempo quanta de $500ns$ é necessário que o Prescaler tenha o valor 40. Os valores dos segmentos 1 e 2 e o ponto de amostra são definidos a partir do barramento CAN existente. A Tabela 4 apresenta os valores utilizados na configuração da rede CAN.

Tabela 4 – Configuração dos parâmetros da rede CAN

Configuração	Valor
Clock	80 MHz
Tempo quanta	500 ns
Prescaler	40
Segmento 1	13 TQ
Segmento 2	2 TQ
Ponto de amotra	87,5%

Fonte: Autoria própria (2023).

4.1.3 Configuração da LoRa no dispositivo interno

Para realizar a configuração da tecnologia LoRa, é necessário definir os seguintes parâmetros: largura de banda, taxa de codificação, preâmbulo e fator de espalhamento. Os valores desses parâmetros estão especificados na Tabela 10.

Os parâmetros utilizados no projeto seguem as especificações padronizadas pela *LoRa Alliance* e os valores adotados são baseados na documentação de referência (ALLIANCE, 2018a).

Tabela 5 – Configuração dos parâmetros da LoRa

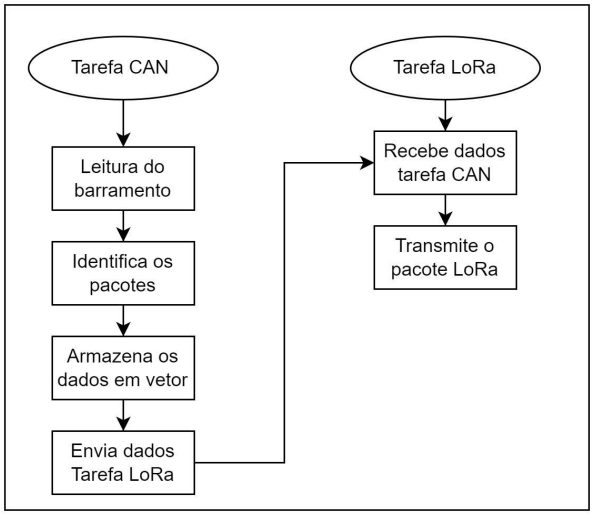
Configuração	Valor
Frequência de transmissão	915 MHz
Largura de banda	125 kHz
Taxa de dados	3
Taxa de bit	1760 bit/s
Taxa de codificação	4/5
Preâmbulo	8
Fator de espalhamento	9

Fonte: Autoria própria (2023).

4.1.4 Implementação das tarefas do dispositivo interno

O *software* do dispositivo interno é gerenciado pelo *Real-time operating system* (RTOS) *FreeRTOS* e é subdividido em duas tarefas distintas: tarefa CAN e tarefa LoRa. Essa abordagem permite uma organização eficiente e uma distribuição adequada das funcionalidades do sistema, garantindo a sincronização e o correto funcionamento das tarefas em paralelo. O fluxograma presente na Figura 20 apresenta a estrutura das tarefas.

Figura 20 – Fluxograma do gerenciamento das tarefas do dispositivo interno



Fonte: Autoria própria (2023).

A tarefa CAN é responsável por ler os dados do barramento CAN e, em seguida, identificar a origem dos pacotes de dados com base em seus identificadores, os valores estão apresentados na Tabela 6.

Os dados recebidos são armazenados em um vetor de tipo de dado $uint8$, sendo que a primeira posição do vetor é reservada para o valor do *duty cycle*. A segunda posição é destinada ao valor da tensão. Já a terceira e quarta posição são destinadas ao valor da corrente, sendo o valor inteiro armazenado na terceira posição e a parte não inteira na quarta posição do vetor.

Tabela 6 – Identificadores dos dispositivos

Dispositivo	Valor hexadecimal do ID
IHM	0x401
Joulímetro	0x402

Fonte: Autoria própria (2023).

Em seguida, os valores armazenados no vetor são enviados para a tarefa LoRa através do *buffer Queue*. O *buffer Queue* é uma estrutura de dados do tipo fila, onde os elementos são armazenados e acessados em uma ordem específica, chamada de FIFO.

Após receber os dados por meio do *buffer Queue*, a tarefa LoRa realiza a transmissão dos dados contidos no vetor por meio de pacotes LoRa. É importante ressaltar que o vetor de dados presente na tarefa LoRa é do tipo $uint8$.

As informações são armazenadas em uma única posição do vetor devido ao tipo de dado utilizado, que é $uint8\_t$. Esse tipo de dado permite armazenar valores decimais de 0 a 255. O valor do *duty cycle*, por sua vez, pode variar de 0 a 100 pois é um valor em porcentagem. O valor da tensão é de aproximadamente $48V$, pois é o valor da tensão nominal da bateria. O valor da corrente pode variar de 0 a 20, estes valores foram testados pela equipe Tubarão Branco durante os laboratórios.

4.1.5 Resultado do dispositivo LoRa interno

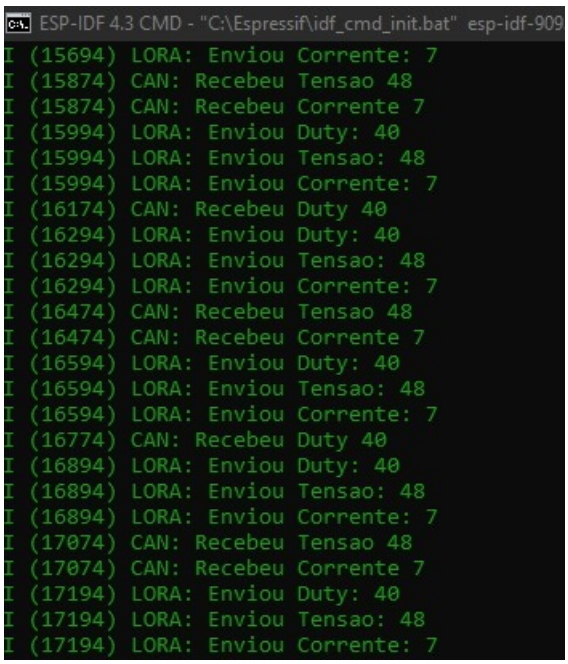
A Figura 21 é uma representação visual do log das informações recebidas através do barramento CAN e do envio dessas informações por meio do enlace LoRa.

Na Figura é possível observar alguns registros dos dados de *duty cycle* enviado pelo dispositivo IHM e os valores da tensão e da corrente da bateria, enviado pelo joulímetro.

O log contém dois identificadores: o identificador CAN e o identificador LoRa. O identificador CAN permite verificar os valores recebidos por meio do barramento CAN, enquanto o identificador LoRa permite verificar os valores enviados por meio do enlace LoRa. A imagem ilustra essa sequência de eventos, proporcionando uma visualização do fluxo das informações desde o barramento CAN até a transmissão via LoRa.

Para realizar esta depuração, utilizou-se a ferramenta *monitor* que pertence a extensão *Espressif-IDF* da IDE VsCode.

Figura 21 – Monitor dispositivo interno



Fonte: Autoria própria (2023).

O registro de log fornece evidências da bem-sucedida execução das tarefas, exibindo os valores do *duty cycle*, tensão e corrente recebidos através do barramento CAN e transmitidos por meio do enlace LoRa.

4.2 Implementação do dispositivo externo

A implementação do dispositivo LoRa externo será dividida em cinco partes distintas, visando uma abordagem estruturada e detalhada do projeto. A primeira parte compreenderá a implementação do *hardware*, abrangendo a descrição dos componentes utilizados e a configuração física do dispositivo. A segunda parte serão descritas as configurações da Lora. A terceira parte serão descritas as configurações da comunicação UART. A quarta parte abordará as tarefas executadas pelo microcontrolador, englobando a programação das funcionalidades e o controle do fluxo de dados. Por fim, a quinta parte consistirá na apresentação das evidências da solução, incluindo a demonstração dos resultados obtidos e a validação do dispositivo LoRa.

Essa divisão em etapas distintas proporcionará uma análise minuciosa de cada aspecto do projeto, permitindo uma compreensão abrangente e aprofundada da implementação do dispositivo LoRa externo.

4.2.1 Implementação do hardware do dispositivo externo

O dispositivo interno foi desenvolvido utilizando o kit de desenvolvimento *ESP32 TTGO LORA V1.0* que possui o módulo LoRa *SX1276* integrado ao kit. O mapeamento dos pinos

utilizados pelo módulo LoRa *SX1276* no kit está detalhado na Tabela 7, que apresenta a correspondência entre os pinos físicos do kit e as funções específicas do módulo LoRa.

Tabela 7 – Mapeamento da conexão dos pinos do módulo LoRa e do kit de desenvolvimento

ESP32 TTGO LORA V1.0	Módulo LoRa SX1276
18	CS
14	RST
19	MISO
27	MOSI
5	SCK

Fonte: Autoria própria (2023).

Para realizar a transmissão dos dados do dispositivo externo ao computador foi utlizado a placa conversora USB *FT232RL*. O mapeamento da conexão entre os pinos físicos da placa *FT232RL* e sua correspondência com os pinos específicos do kit de desenvolvimento está detalhado na Tabela 8.

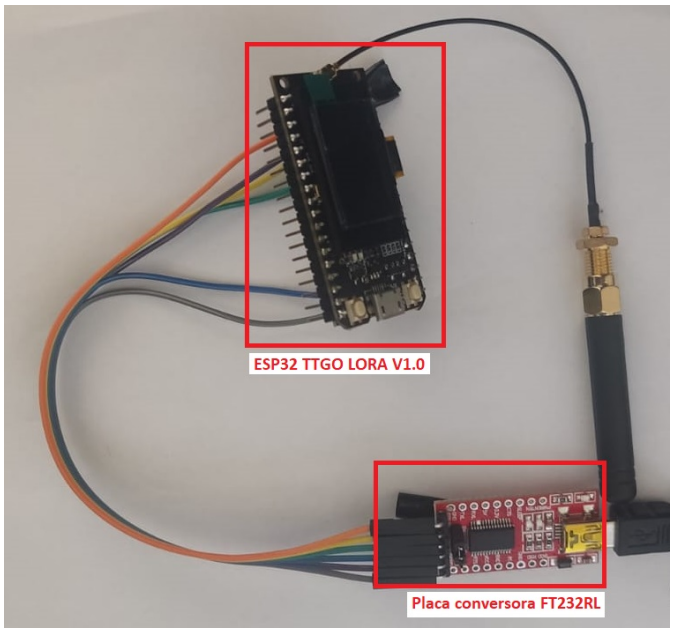
Tabela 8 – Mapeamento da conexão dos pinos da placa FT232RL e do kit de desenvolvimento

ESP32 TTGO LORA V1.0	Conversor serial USB FT232RL
5v	5v
GND	GND
1	UART_Tx
3	UART_Rx

Fonte: Autoria própria (2023).

A Figura 22 apresenta a montagem do dispositivo interno com o kit de desenvolvimento *ESP32 TTGO LORA V1.0* e a placa conversora USB *FT232RL*.

Figura 22 – Conexão entre a placa FT232RL e o kit de desenvolvimento



Fonte: Autoria própria (2023).

O Display OLED *SSD1308* também está integrado ao kit de desenvolvimento. O mapeamento dos pinos utilizados pelo OLED *SSD1308* no kit está detalhado na Tabela 9, que apresenta a correspondência entre os pinos físicos do kit e as funções específicas do *display*.

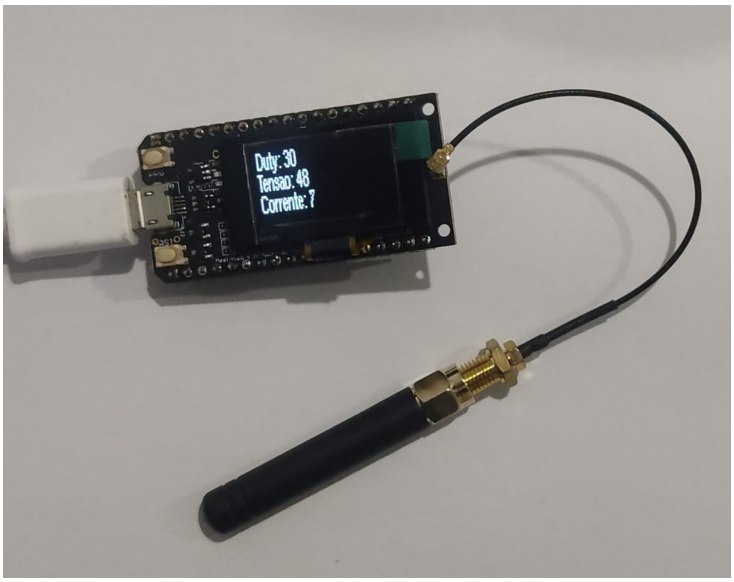
Tabela 9 – Mapeamento da conexão dos pinos do display OLED no kit de desenvolvimento

ESP32 TTGO LORA V1.0	Display OLED
4	SDA
15	SCL
16	RST

Fonte: Autoria própria (2023).

A Figura 23 apresenta o funcionamento do display OLED, que imprime no display as informações recebidas através da LoRa.

Figura 23 – Display dispositivo externo



Fonte: Autoria própria (2023).

4.2.2 Configuração da LoRa no dispositivo externo

Para realizar a configuração da tecnologia LoRa, é necessário definir os seguintes parâmetros: largura de banda, taxa de codificação, preâmbulo e fator de espalhamento. Os valores desses parâmetros estão especificados na Tabela 10.

Os parâmetros utilizados no dispositivo externo são identicos aos parâmetros da configuração do dispositivo interno para que seja possível realizar a comunicação através do enlace LoRa.

Tabela 10 – Configuração dos parâmetros da LoRa

Configuração	Valor
Frequência de transmissão	915 MHz
Largura de banda	125 kHz
Taxa de dados	3
Taxa de bit	1760 bit/s
Taxa de codificação	4/5
Preâmbulo	8
Fator de espalhamento	9

Fonte: Autoria própria (2023).

4.2.3 Configuração da UART no dispositivo externo

Os parâmetros de comunicação UART foram configurados de acordo com as especificações da placa *FT232RL*. Os valores utilizados estão descritos na Tabela 11. Esses parâmetros incluem a taxa de transmissão, o número de bits de dados, os bits de parada, o bit de paridade e o controle de fluxo. A configuração adequada desses parâmetros é fundamental para garantir uma comunicação correta e confiável entre os dispositivos conectados via UART.

Tabela 11 – Configuração da comunicação UART

Configuração	Valor
Clock	80 MHz
Taxa de transmissão	115200
Bits de dados	8
Bit de parada	1
Bit de paridade	Desabilitado
Controle de fluxo	Desabilitado

Fonte: Autoria própria (2023).

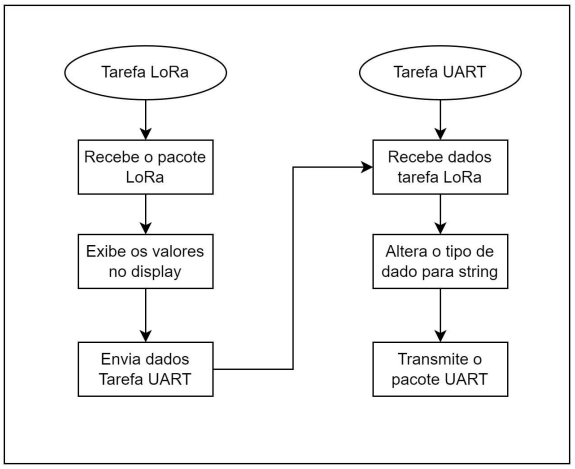
4.2.4 Implementação do software do dispositivo externo

O *software* do dispositivo externo é gerenciado pelo RTOS *FreeRTOS* e é subdividido em duas tarefas distintas: tarefa LoRa e tarefa UART. Essa abordagem permite uma organização eficiente e uma distribuição adequada das funcionalidades do sistema, garantindo a sincronização e o correto funcionamento das tarefas em paralelo. O fluxograma presente na Figura 24 apresenta a estrutura das tarefas.

A tarefa LoRa recebe os dados provenientes do enlace LoRa e os armazena em um vetor de dados do tipo $uint8$. Os dados recebidos através do enlace são exibidos em um *display* e por fim, são enviados para a tarefa UART através do *buffer Queue*.

O vetor de dados segue mesmo o padrão de armazenamento do dispositivo interno, sendo a primeira posição do vetor para o *duty cycle*, a segunda para a tensão e a terceira e quarta posições para a corrente.

Figura 24 – Fluxograma do gerenciamento das tarefas do dispositivo externo



Fonte: Autoria própria (2023).

Após receber os dados por meio do *buffer Queue*, a tarefa UART transmite os dados para a placa conversora *FT232RL*, que realiza a transmissão dos dados atrvés da comunicação USB para o computador.

4.2.5 Resultado do dispositivo LoRa Externo

A Figura 25 é uma representação visual do log das informações recebidas através do enlace LoRa e do envio dessas informações através da placa conversora USB *FT232RL*.

Na figura, é possível observar os dados de *duty cycle*, tensão e corrente da bateria. A imagem ilustra essa sequência de eventos, permitindo visualizar o fluxo das informações.

Para realizar esta depuração, utilizou-se a ferramenta *monitor* que pertence a extensão *Espressif-IDF* da IDE VsCode.

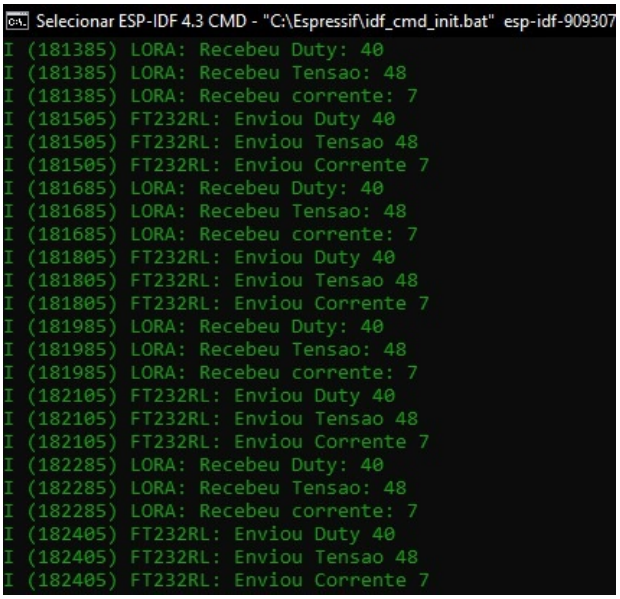
Por meio do registro de log, é possível verificar os valores recebidos por meio do enlace LoRa e a transmissão realizada pela UART. Além disso, é possível acompanhar a execução das tarefas do dispositivo externo.

4.3 Implementação do tratamento de dados por meio do computador

Para o tratamento dos dados através do computador foi utilizado o IDE *Spyder*, que suporta a linguagem de programação *Python*. Através do *Spyder*, foi realizado a implementação das etapas necessárias para processar os dados coletados.

A estratégia de dividir em duas etapas, a primeira para realizar a leitura da porta USB e a segunda para imprimir os gráficos, se deve ao fato da necessidade de realizar a comparação do desempenho do veículo em diversos trechos do percurso durante a competição, devido a mudança de estratégia, para melhor desempenho do veículo, por parte da equipe.

Figura 25 – Monitor dispositivo externo



Fonte: Autoria própria (2023).

Desta forma, a etapa de leitura da porta USB poderá ser executada durante todo o tempo da prova sem que haja conflito de permissões de acesso à porta USB. O código de impressão dos dados pode ser adaptado de acordo com a demanda da equipe, tornando possível imprimir diversos gráficos partindo da mesma fonte de dados (arquivo CSV) e pode ser executado tanto em tempo real quanto após o término do percursso do veículo. A leitura da porta e a impressão dos graficos são realizados a cada um segundo.

A primeira etapa consistiu na leitura dos dados provenientes da porta USB. A linguagem *Python* possui a biblioteca *serial*, que permite estabelecer a conexão com a porta USB, para obter os dados transmitidos pelo dispositivo LoRa externo. Em seguida, foi utilizada a biblioteca *CSV*, para armazenar os dados em um arquivo CSV. Também foi utilizada as bibliotecas *time* e *datetime* para realizar a contagem do tempo.

A Figura 26 apresenta o *IPython Console* com os dados recebidos e escritos no arquivo CSV. A primeira coluna apresenta um contador em segundos, a segunda coluna apresenta o valor do *duty cycle*, a terceira coluna apresenta tensão da bateria, a quarta coluna apresenta a corrente da bateria e a quinta coluna apresenta o tempo no formato *hh:mm:ss*.

Por fim, para realizar a manipulação e impressão dos dados em forma gráfica, foram utilizadas as bibliotecas *pandas* e *matplotlib*.

A biblioteca *pandas* foi empregada para a manipulação dos dados armazenados. Já a biblioteca *matplotlib* foi utilizada para a criação de gráficos a partir dos dados processados.

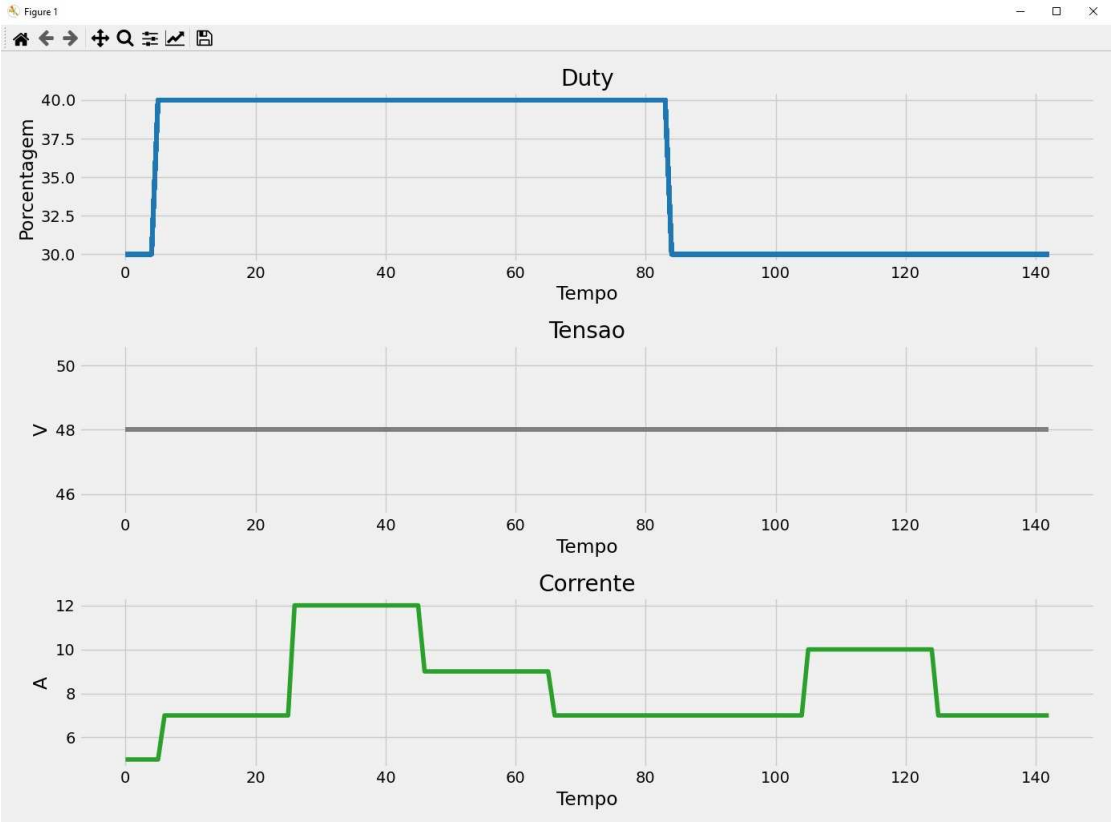
A Figura 27 apresenta o resultado final deste projeto, onde são exibidos os gráficos gerados a partir dos dados coletados. Esses gráficos proporcionam uma visualização clara e concisa do comportamento do veículo durante as competições, permitindo realizar análises e identificar possíveis melhorias para otimizar seu desempenho.

Figura 26 – IPython console da IDE Spyder com os dados lidos e escritos no arquivo csv

Fonte: Autoria própria (2023).

O primeiro gráfico apresenta os valores do *duty cycle* em porcentagem em função do tempo em segundos. O segundo gráfico apresenta a tensão da bateria em volts em função do tempo em segundos. O terceiro gráfico apresenta o comportamento da corrente em ampere em função do tempo em segundos.

Figura 27 – Visualização gráfica dos dados



Fonte: Autoria própria (2023).

Os trajetos percorridos pelo veículo durante as competições, até o momento, costumam ser em terrenos planos e a velocidade máxima da competição é de $40km/h$. O limite de velo-

cidade raramente é atingido, pois de acordo com (TRENTO, 2020) quanto menor o *duty cycle* maior é a eficiência do protótipo.

Portanto durante a competição o veículo não realiza mudanças abruptas em relação ao *duty cycle* e pelo trajeto ser em terrenos planos os valores da corrente também não possuem variações abruptas.

4.3.1 Considerações finais da implementação

O desenvolvimento deste projeto possui como objetivo principal o monitoramento remoto do veículo realizando a leitura dos dados dos dispositivos existentes no veículo. A partir desta primeira versão de monitoramento remoto a equipe Tubarão Branco poderá ampliar o projeto.

Como projetos futuros a equipe poderá implementar o GPS integrado ao kit de desenvolvimento *ESP32 TTGO T-Beam v1.1*, assim como o cálculo da velocidade do veículo. Também poderá ser implementado um sistema de monitoramento da temperatura das chaves do inversor trifásico presente no acionamento do motor, pois a temperatura elevada pode danificar os circuitos integrados.

Com a ampliação dos dispositivos do protótipo, também haverão mais dados a serem coletados. Portanto, o projeto deste trabalho de conclusão de curso poderá ser adaptado para as novas realidades vindouras da equipe Tubarão Branco, neste caso a estratégia de armazenamento dos dados em um único pacote LoRa poderá ser alterado, pois este projeto possui uma grande escalabilidade.

4.4 Validação do sistema e discussões

A fim de realizar a validação, foram conduzidos dois testes distintos. O primeiro teste teve como objetivo validar o envio dos pacotes de dados utilizando as configurações estabelecidas neste documento. O segundo teste teve como objetivo testar o envio de pacotes em diferentes configurações e compará-las entre si.

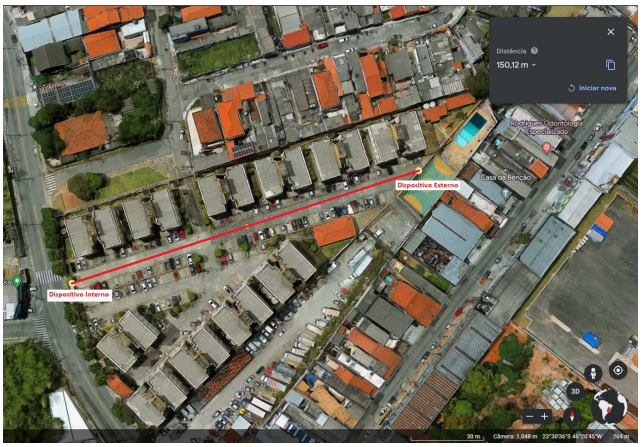
4.4.1 Teste 1: Validação para a configuração apresentada o documento

O experimento realizado para validar o enlace LoRa foi conduzido em um condomínio, onde não havia obstáculos entre o dispositivo interno e o dispositivo externo. A temperatura ambiente durante o experimento foi registrada em torno de $17°C$.

O dispositivo interno foi posicionado em uma elevação de aproximadamente $1{,}80$ metros em relação ao solo, enquanto o dispositivo externo foi colocado a uma altura de aproximadamente $1{,}50$ metros em relação ao solo. A distância máxima entre os dispositivos durante o experimento foi de 150 metros, conforme ilustrado na Figura 28. Essa distância foi selecionada

para avaliar o desempenho e a eficácia do enlace LoRa em condições de comunicação sem obstáculos diretos entre os dispositivos.

Figura 28 – Distância do experimento de validação



Fonte: Autoria própria (2023).

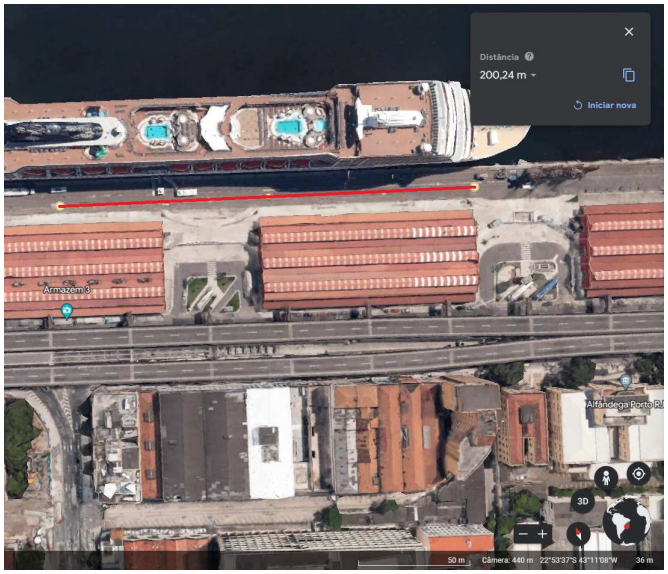
O experimento foi realizado por aproximadamente 3 minutos, sendo que foram enviados 192 pacotes pelo dispositivo interno, com o intervalo de 500 milissegundos entre cada pacote. O dispositivo externo recebeu 192 pacotes. Portanto, o experimento para a distância de 150 metros possui a eficiência de 100%. A Figura 29 apresenta a foto tirada do *display* no fim do experimento.

Figura 29 – Distância do experimento de validação

Fonte: Autoria própria (2023).

A distância do experimento realizado é considerada adequada, uma vez que atende aos requisitos estabelecidos pela última competição de eficiência energética em que a equipe participou. Essa competição ocorreu no Pier Mauá, localizado na cidade do Rio de Janeiro, e as extremidades da pista tinham uma distância aproximada de 200 metros, conforme ilustrado na Figura 30. Apesar das extremidades serem de 200 metros a área para os espectadores fica na região central da pista.

Figura 30 – Distância das extremidades da pista na competição



Fonte: Autoria própria (2023).

4.4.2 Teste 2: Validação para distintas configurações

O experimento foi conduzido na cidade de Suzano com o objetivo de determinar a distância máxima de transmissão dos pacotes para diferentes configurações na tecnologia LoRa. A validação da distância máxima foi baseada em um critério de taxa de sucesso de aproximadamente 90%, pois valores inferiores poderiam afetar o desempenho da monitoramento do veículo. A Figura

Figura 31 – Distâncias do experimento de validação das configurações

Fonte: Autoria própria (2023).

No experimento, foram utilizadas três configurações distintas, conforme apresentado na Tabela 12. Os valores da frequência de transmissão, largura de banda, taxa de codificação e preâmbulo foram mantidos constantes para todas as configurações.

Tabela 12 – Configuração dos parâmetros fixos no experimento

Configuração LoRa	Valor
Frequência de transmissão	915 MHz
Largura de banda	125 kHz
Taxa de codificação	4/5
Preâmbulo	8

Fonte: Autoria própria (2023).

No experimento foram utilizadas as taxas de transmissões DR2, DR3 e DR4. A Tabela 13 apresenta os valores das configurações da taxa de bit, fator de espalhamento, distância máxima e a taxa de sucesso de transmissão para as três taxas de transmissões.

Após a realização do experimento, observou-se uma relação inversa entre a taxa de transmissão e a distância máxima de transmissão. Verificou-se que quanto maior a taxa de transmissão, menor foi a distância máxima alcançada. Além disso, constatou-se que, para as competições da equipe, a taxa de transmissão de $3125bit/s$ é adequada, uma vez que a distância máxima percorrida pelo protótipo não ultrapassará os $725m$.

Tabela 13 – Configuração dos parâmetros variáveis no experimento

Configuração LoRa	DR 2	DR 3	DR 4
Taxa de bit	980 bit/s	1760 bit/s	3125 bit/s
Fator de espalhamento	10	9	8
Distância	950m	870m	725m
Taxa de sucesso	90%	88%	91%

Fonte: Autoria própria (2023).

4.4.3 Considerações finais do projeto

Existem estudos na literatura, como o realizado por (AUGUSTIN *et al.*, 2016), que apresentam experimentos com eficiência de transmissão superiores a 80% para distâncias de até 2800 metros.

No contexto atual da equipe, não são necessárias comunicações de distâncias tão longas, conforme apresentado por (AUGUSTIN *et al.*, 2016). No entanto, em projetos futuros, configurações diferentes do enlace LoRa podem ser consideradas, levando em conta a possibilidade de alcançar distâncias maiores de transmissão.

5 CONCLUSÃO

Este trabalho teve como objetivo a implementação de um sistema de monitoramento remoto aplicado em um protótipo de veículo elétrico. Durante o desenvolvimento, foi utilizado o protocolo CAN para estabelecer a comunicação com fio entre os dispositivos presentes no protótipo, permitindo a troca de informações de forma eficiente e confiável. Além disso, foi utilizada a tecnologia LoRa para realizar a transmissão sem fio dos dados coletados pelo veículo.

Para viabilizar a transmissão dos dados do veículo para um computador, foram utilizados os protocolos de comunicação UART e USB. Através desses protocolos, os dados transmitidos pela tecnologia LoRa são recebidos e processados pelo computador, possibilitando a sua visualização em forma gráfica. Essa abordagem permite uma análise mais detalhada do comportamento do veículo durante as competições, fornecendo *insights* valiosos para a equipe de eficiência energética Tubarão Branco.

Com a implementação do dispositivo LoRa interno, foi viabilizada a leitura do valor do *duty cycle* transmitido pelo dispositivo IHM, bem como a leitura dos valores de tensão e corrente da bateria, transmitidos pelo joulímetro. Essas leituras foram realizadas com sucesso, possibilitando a transmissão sem fio dos dados por meio do enlace LoRa.

Em relação à implementação do dispositivo LoRa externo, foi possível efetuar a leitura dos dados recebidos através do enlace LoRa e transmiti-los para o computador utilizando a comunicação USB.

Por fim, foi possível realizar a leitura da porta USB utilizando a linguagem *Python* e armazenar os dados em um arquivo CSV. Além disso, também foi viável a leitura dos dados do arquivo CSV e a sua impressão em forma de gráfico, proporcionando uma visualização clara e compreensível dos dados obtidos.

Em resumo, o principal objetivo deste trabalho foi aplicar o sistema de monitoramento remoto para realizar uma análise detalhada do comportamento do veículo durante as competições da equipe Tubarão Branco. Através da visualização dos dados em forma de gráficos, será possível identificar padrões, otimizar o desempenho e garantir a segurança do piloto. Com os resultados obtidos, espera-se contribuir para o aprimoramento contínuo do veículo elétrico e promover a eficiência energética nas competições.

REFERÊNCIAS

ALLIANCE, L. **LoRaWAN 1.0.3 Regional Parameters**. 2018. Disponível em: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_regional_parameters_v1.0.3reva_0.pdf. Acesso em: 08 mai. 2023.

ALLIANCE, L. **RP2-1.0.3 LoRaWAN® Regional Parameters**. 2018. Disponível em: https://hz137b.p3cdn1.secureserver.net/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf?time=1683287188. Acesso em: 08 mai. 2023.

ANATEL. **Resolução nº 680**. 2017. Disponível em: https://informacoes.anatel.gov.br/legislacao/resolucoes/2017/936-resolucao-680. Acesso em: 08 mai. 2023.

AUGUSTIN, A. *et al.* A study of lora: Long range amp; low power networks for the internet of things. **Sensors**, MDPI Open Access Journal, v. 16, maio 2016. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/16/9/1466. Acesso em: 17 feb. 2023.

BOSCH, R. **CAN Specification**. Version 2.0, 1991.

CORRIGAN, S. **Introduction to the Controller Area Network (CAN)**. [*S.l.*], 2016. Disponível em: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1678312246255. Acesso em: 08 mar. 2023.

JUNIOR, E. dos S.; SOUZA, G. M. D.; VARGAS, D. R. Rede can embarcada em um protótipo de veículo elétrico de alta efiência energética. *In*: **XXVI Seminário de Iniciação Científica e Tecnológica, SICITE**. Guarapuava: UTFPR, 2021.

KUMAR, S.; DALAL, S.; DIXIT, V. The osi model: Overview on the seven layers of computer networks. **International Journal of Computer Science and Information Technology Research**, set. 2014. ISSN 2348-120X. Acesso em: 01 mar. 2023.

RICHARDS, P. **Understanding Microchip's CAN Module Bit Timing**. [*S.l.*], 2002. Disponível em: https://ww1.microchip.com/downloads/en/Appnotes/00754.pdf. Acesso em: 08 mar. 2023.

SANTIAGO, J. de *et al.* Electrical motor drivelines in commercial all-electric vehicles: A review. **IEEE Transactions on Vehicular Technology**, v. 61, n. 2, p. 475–484, 2012. ISSN 1939-9359. Disponível em: https://ieeexplore.ieee.org/abstract/document/6093982.

SEMTECH. **LoRa® and LoRaWAN®: A Technical Overview**. 2019. Disponível em: https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan. Acesso em: 21 mar. 2023.

TI. **Introduction to the Controller Area Network (CAN)**. [*S.l.*], 2016. Disponível em: ftp://sundance.com/Pub/Support_Files/jp/ccs31/setup/docs/pdf/sloa101.pdf. Acesso em: 02 mar. 2023.

TRENTO, E. Estudo de ponte trifásica para acionamento de motor cc sem escovas para uso em veículos elétricos. *In*: **Monografia (Trabalho de conclusão do curso de graduação)**. Pato Branco: UTFPR, 2020.

ZSW. **Press Release 04/2018**. Center for Solar Energy and Hydrogen Research Baden-Württemberg (Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg, ZSW), 2018.

ZSW. **Data Service Renewable Energies**. Center for Solar Energy and Hydrogen Research Baden-Württemberg (Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg, ZSW), 2021.