UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA

Eduardo Guasso Tier

DESENVOLVIMENTO DE SISTEMAS DE MONITORAMENTO EMPREGANDO TECNOLOGIA LORA EM UM PROTÓTIPO TIPO FORMULA SAE

Santa Maria, RS

2019

Eduardo Guasso Tier

DESENVOLVIMENTO DE SISTEMAS DE MONITORAMENTO EMPREGANDO TECNOLOGIA LORA EM UM PROTÓTIPO TIPO FORMULA SAE

Trabalho de Conclusão de Curso apresentado ao do curso de Engenharia de Controle e Automação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para obtenção do grau de **Engenheiro de Controle e Automação.**

Orientador: Prof. Frederico Menine Schaf

Santa Maria, RS

2019

Eduardo Guasso Tier

DESENVOLVIMENTO DE SISTEMAS DE MONITORAMENTO EMPREGANDO TECNOLOGIA LORA EM UM PROTÓTIPO TIPO FORMULA SAE

Trabalho de Conclusão de Curso apresentado ao do curso de Engenharia de Controle e Automação da Universidade Federal de Santa Maria (UFSM), como requisito parcial para obtenção do grau de **Engenheiro de Controle e Automação.**

Aprovado em 18 de julho de 2019:

Frederico Menine Schaf, Dr. (UFSM)

(Presidente / Orientador)

Carlos Henrique Barriquello, Dr. (UFSM)

Claiton Moro Franchi, Dr. (UFSM)

DEDICATÓRIA

Dedico este estudo a todos os futuros engenheiros e estudantes da área, no intuito de ajuda-los a colocar em prática a tecnologia LoRa de radiofrequência.

Aos futuros integrantes da equipe Formula UFSM, como forma de agradecimento ao projeto que me ajudou a crescer como profissional e como pessoa.

Por fim, não poderia deixar de dedica-lo à minha família, que me apoiou e me proporcionou o necessário para tornar possível este e outros trabalhos durante os anos de graduação.

AGRADECIMENTOS

Agradeço a equipe Formula UFSM por todo aprendizado e por todos os momentos de superação e conquista durante os quatro anos em que estive presente.

Agradeço ao professor Frederico pela orientação neste trabalho e pela disponibilidade e interesse em ajudar a equipe Formula UFSM sempre que possível.

Agradeço ao professor Natanael Rodrigues pela orientação durante o Projeto Integrador, etapa inicial no desenvolvimento deste trabalho.

Agradeço aos meus pais pelo apoio e confiança depositados em mim para que eu conquistasse a graduação.

Agradeço aos meus amigos e colegas de subsistema Daniel Schreiner, Eugênio Pozzobon e Caroline dos Santos, pela companhia e ajuda diária para que este trabalho fosse realizado.

Agradeço aos meus colegas de graduação pela união durante os momentos difíceis e pela companhia durante os momentos de lazer.

Por fim, agradeço a empresa Mazza G-Tec apoiadora da equipe Formula UFSM e que proporciona a fabricação das tão necessárias placas de circuito impresso.

Santa Maria, RS

2019

“Não podemos resolver nossos problemas com o mesmo pensamento que usamos quando os criamos” (novamente Einstein).

(Claude Rouelle)

Santa Maria, RS

2019

RESUMO

DESENVOLVIMENTO DE SISTEMAS DE MONITORAMENTO EMPREGANDO TECNOLOGIA LORA EM UM PROTÓTIPO TIPO FORMULA SAE

AUTOR: Eduardo Guasso Tier

ORIENTADOR: Frederico Menine Schaf

Hoje em dia, a informação em tempo real é muito importante para acompanharmos o comportamento de sistemas que fazem parte do nosso dia a dia. Podemos vê-lo na indústria, onde uma empresa pode monitorar seus veículos via satélite ou, como neste trabalho, em um carro de corrida. Os sistemas de telemetria permitem que os sinais vitais e o desempenho dinâmico do veículo sejam monitorados remotamente, de forma constante. Assim, o presente trabalho descreve um estudo sobre o emprego de tecnologia LoRa no desenvolvimento de um sistema de telemetria com interface para computador para o protótipo 2019 da Equipe Formula UFSM, e envolve projeto eletrônico, desenvolvimento de software, programação, testes e manufatura dos componentes do sistema. Informações como temperatura do motor, temperatura do óleo, tensão da bateria e entre outros dados são enviados para um computador portátil onde são decodificados e mostrados em uma interface de monitoramento customizada que contém gráficos, medidores e indicadores. Além disso, optou-se por utilizar a comunicação Bluetooth presente em smartphones para desenvolver um aplicativo com uma tela de display para volante e uma tela de monitoramento para telemetria auxiliar. Sendo assim, apesar de o sistema não ter usufruído do desempenho máximo permitido pelo módulo LoRa devido a implementação de altas taxas de transmissão de dados e grandes pacotes, o objetivo principal de coletar dados de múltiplos sensores do protótipo e comunicar a um computador e a um smartphone em tempo real foi alcançado de forma satisfatória, atingindo os requisitos exigidos pelo projeto.

Palavras-chave: Telemetria. LoRa. Supervisório. Análise de dados. Aplicativo. Bluetooth. Formula SAE.

ABSTRACT

DEVELOPMENT OF MONITORING SYSTEMS USING LORA TECNOLOGY ON A FORMULA SAE PROTOTYPE

AUTHOR: Eduardo Guasso Tier

ADVISOR: Frederico Menine Schaf

Nowadays, the real time information is very important to follow the behavior of the systems that are part of our day to day. We can see in the industry, where a company can monitor its vehicles by satellite or, like in this work, in a race car. Telemetry systems allows vital signs and dynamic vehicle performance to be monitored remotely, in a constant way. Thus, the present work describes a study about the use of LoRa technology in the development of a computer interface telemetry system for the 2019 prototype of the UFSM Formula student team, and involves software design, programming, testing and manufacturing of system components. Information such as engine temperature, oil temperature, battery voltage, among others, are sent to a computer screen and displayed on a custom monitoring interface containing charts, gauges and indicators. In addition, was chose to use Bluetooth communication present in smartphones, to develop an application with a display to steering wheel screen and a monitoring screen for auxiliary telemetry. In this way, although the system has not enjoyed the maximum performance allowed by the LoRa module due to the implementation of high rates of data transmission and large packets, the main objective of collect data from multiple sensors of the prototype and communicate to a computer and a smartphone in real-time was achieved in a satisfactory manner, reaching the requirements requested by the project.

Keywords: Telemetry. LoRa. Supervisory. Data Analysis. Application. Bluetooth. Formula SAE.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	3 Dimension
ASCII	American Standard Code For Information Interchange
CAD	Computer-Aided Design Cad
CAN	Controller Area Network
CPU	Central Processing Unit
CR	Coding Rate
CRC	Cyclic Redundancy Checking
SF	Spreading Factor
CSMA	Carrier Sense Multiple Access
CSS	Chirp Spread Spectrum
CTISM	Colégio Técnico Industrial De Santa Maria
DIP	Dual In-Line Package
ECU	Engine Control Unit
EEPROM	Electrically-Erasable Programmable Read-Only Memory
EHF	Extra High Frequency
EUA	Estados Unidos Da América
FEC	Forward Error Correction
FIFO	First In, First Out
FSK	Frequency Shift Keying
FTDI	Future Technology Devices Internacional
GFSK	Gaussian Frequency Shift Keying
GNU	General Public License
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System For Mobile Communication
HF	High Frequency
IDE	Integrated Development Environment
IoT	Internet Of Things
IP	Internet Protocol
ISM	Industrial Sientific And Medical
L2CAP	Logical Link Control And Adaptation
LED	Light Emitting Diode
LF	Low Frequency

LoRa	Long Range
LPWAN	Low Power Wide Area Network
M2M	Machine To Machine
MAC	Media Access Control
MAP	Manifold Air Pressure
MF	Medium Frequency
MIT	Massachusetts Institute Of Technology
NRZ	Non-Return To Zero
NTC	Negative Temperature Coeficient
OOK	On-Off Keying
PAN	Personal Area Networks
PPP	Point-Topoint Protocol
PTC	Positeve Temperature Coeficient
PWM	Pulse Width Modulation
RPM	Rotações Por Minuto
RSSI	Received Signal Strength Indicator
RTU	Remote Terminal Unit
SAE	Society Of Automotive Engineers
SHF	Super High Frequency
SMA	Subminiature Version A
SMD	Surface Mounting Device
SNR	Signal-To-Noise Ratio
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
TCP	Trasmission Control Protocol
TDD	Time Division Duplex
TPS	Throttle Position Sensor
TTL	Transistor-Transitor Logic
UART	Universal Asynchrounous Receiver/Transmiter
UFSM	Universidade Federal De Santa Maria
UHF	Ultra High Frequecy
USB	Universal Serial Bus
VHF	Very High Frequency
WAP	Wireless Application Protocol

SUMÁRIO

1 INTRODUÇÃO

Hoje em dia, a informação em tempo real é muito importante para o acompanhamento do comportamento de sistemas que fazem parte do nosso dia a dia. Estes sistemas podem ser vistos na indústria, onde uma empresa pode monitorar seus veículos via satélite ou, como neste trabalho, em um carro de corrida. A competição Fórmula SAE Brasil, tem como objetivo propiciar aos estudantes de engenharia a oportunidade de aplicar os conhecimentos adquiridos em sala de aula, no projeto e na construção de um protótipo do tipo Formula SAE.

> Uma das armas mais importantes que uma equipe de corrida pode empregar é informação. Quanto mais informações ela puder reunir (e processar), melhor será seu julgamento na tomada de decisões importantes. A aquisição de dados fornece aos engenheiros a informação que eles e a equipe necessitam para avaliar o desempenho do veículo (SEGERS, 2008).

Sendo assim, o presente trabalho propõe a utilização das tecnologias de radiofrequência LoRa (*Long Range*) e Bluetooth para o desenvolvimento de um sistema de telemetria com interface para computador e para smartphone, juntamente com um *display* para volante eletrônico para o protótipo 2018 da Equipe Formula UFSM, e envolve projeto eletrônico, desenvolvimento de *software*, desenvolvimento de aplicativo, programação, testes e manufatura dos componentes do sistema.

Desta forma, espera-se que este novo sistema de telemetria resolva as limitações encontradas no sistema anterior e atenda as exigências da equipe para o novo protótipo.

1.1 OBJETIVOS

O sistema de telemetria tem como objetivo coletar dados de múltiplos sensores do protótipo e comunicar a um computador e a um *smartphone* em tempo real, a fim de apresentar para a equipe técnica informações relevantes sobre o comportamento do veículo. Além disso, este trabalho também tem como objetivo realizar um estudo sobre a aplicação e o desempenho da tecnologia LoRa para este sistema em particular.

Para o sistema de telemetria, dados como rotação do motor (RPM), velocidade do carro, posição da borboleta (TPS), pressão no coletor de admissão (MAP), temperatura do motor, do

óleo, da ECU (*Engine Control Unity*) e do ar, quantidade de oxigênio no escapamento (Lambda), pressão na linha de combustível, pressão do óleo e tensão da bateria, devem ser enviados para um receptor localizado no box da equipe, para que possam ser lidos em uma interface de monitoramento.

O aplicativo para volante tem como objetivos mostrar ao piloto as informações necessárias para uma pilotagem hábil e segura, e facilitar o monitoramento dos parâmetros do carro após cada teste em pista. Assim, através de comunicação Bluetooth, almeja-se o desenvolvimento de um aplicativo que contenha uma tela de display para volante e uma tela de monitoramento para telemetria auxiliar.

A interface do *display* para volante deve conter informações de rotação do motor e marcha atual, e indicadores de temperatura alta do motor e do óleo, enquanto que o *display* para telemetria auxiliar deve conter informações de rotação do motor (RPM), posição da borboleta (TPS), pressão no coletor de admissão (MAP), temperatura do motor e do óleo, quantidade de oxigênio no escapamento (Lambda), pressão na linha de combustível, pressão do óleo e tensão da bateria.

1.2 JUSTIFICATIVA

Em veículos tipo Formula SAE, este não é um sistema obrigatório a ser desenvolvido, mas possui um papel importante durante os testes e competições, pois ele permite que os sinais vitais e o desempenho dinâmico do veículo sejam monitorados remotamente de forma constante. Com o objetivo de, ao informar ao piloto, evitar possíveis falhas que possam prejudicar o desempenho do carro durante a competição.

Durante os testes dinâmicos do protótipo, realizados para a competição, a equipe encontrava dificuldades para monitorar os parâmetros do carro após os testes, uma vez que era preciso conectar a ECU do protótipo a um computador através de cabos, tornando este procedimento muito demorado.

Além disso, um *display* para volante torna possível que o piloto tenha acesso a dados instantâneos do veículo, essenciais para que este efetue ações necessárias que contribuam para a melhor performance quanto à estratégia de pista e pilotagem.

1.3 A COMPETIÇÃO FORMULA SAE

A competição Fórmula SAE BRASIL tem como objetivo propiciar aos estudantes de Engenharia a oportunidade de aplicar na prática os conhecimentos adquiridos em sala de aula, desenvolvendo um projeto completo e construindo um carro tipo Fórmula. Criada em 2004, a Fórmula SAE BRASIL está a caminho da sua 15ª edição, onde as duas equipes melhor classificadas ganham o direito de representar o Brasil em duas competições internacionais realizadas nos EUA.

Durante três dias de evento, os carros passam por provas estáticas e dinâmicas, avaliando a performance de cada projeto na pista, assim como as apresentações técnicas das equipes, que inclui projeto, custo, e uma apresentação de *marketing*. Todas as provas são pontuadas de maneiras diferentes, de maneira a garantir que o melhor conjunto de projeto e carro vença a competição.

A competição é uma oportunidade de crescimento, onde os trabalhos são realizados em grupo com intuito de estimular e desenvolver o trabalho em equipe. Além disso, os estudantes também ganham visibilidade por parte das grandes empresas e oportunidade de reconhecimento (CRUZ, 2018).

1.4 A EQUIPE FORMULA UFSM

Inicialmente denominada Bombaja Racing, a equipe Formula UFSM surgiu no ano de 2010 para representar a Universidade Federal de Santa Maria no projeto Formula SAE. Atualmente conta com cerca de 25 estudantes e é organizada de forma semelhante à uma empresa.

É composta pelo professor orientador Mario Martins, ex-participante da equipe da Brunel University e ex-juiz da competição Formula SAE Brasil, pelos professores colaboradores Fernando Bayer (CTISM), Cristiano Roos e Roberto Hausen, pelo colaborador Joelson Bilhão e por estudantes das mais diversas áreas de engenharia, sendo um deles o capitão e os demais estudantes compondo os subsistemas da equipe. Os subsistemas possuem estudantes responsáveis por organizar e delegar as funções entre os outros estudantes, de modo a organizar e manter o fluxo de trabalho.

Para a fabricação de um protótipo tipo Formula SAE, a equipe faz o projeto informacional e conceitual do veículo, seguindo o regulamento da competição, simulando o modelo em *softwares* de engenharia. Após a fase de projeto, é realizada uma listagem de componentes necessários para a montagem, onde posteriormente realiza-se o planejamento da produção para então, iniciar a fase de fabricação.

1.5 ORGANIZAÇÃO DO TRABALHO

Neste capítulo foram apresentados os objetivos, a justificativa, a competição Formula SAE, a equipe Formula UFSM e a organização do trabalho. No capítulo 2 são apresentadas as definições e fundamentações teóricas dos dispositivos e dos sistemas de comunicação que foram utilizados para o desenvolvimento deste sistema. Neste mesmo capítulo também consta uma explicação sobre o funcionamento e parametrização da tecnologia LoRa. No capítulo 3 é demonstrado a definição conceitual do projeto, os métodos de utilização e manufatura dos componentes físicos (*hardware*) e lógicos (*software*) do sistema e o processo de desenvolvimento do sistema de telemetria e do aplicativo, bem como um estudo sobre a escolha dos parâmetros LoRa. O capítulo 4 apresenta os resultados e discussões sobre cada setor do projeto de telemetria (transmissor, receptor e interface supervisória) e do aplicativo, além da descrição do teste de validação de parâmetros elaborado. Para finalizar há as considerações finais sobre o processo de desenvolvimento do sistema no capítulo 5, e os códigos elaborados para o correto funcionamento do sistema encontram-se como apêndices no final deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Um sistema de telemetria é um sistema elétrico utilizado para medir uma quantidade de uma fonte de informação e transmitir a quantidade medida para uma estação receptora distante onde estas medidas podem ser exibidas e/ou gravadas (RCC, 2006).

O desenvolvimento de um sistema de telemetria envolve a utilização de diferentes tecnologias durante o processo de aquisição, transmissão e amostragem dos dados. A seguir, são definidos as tecnologias de comunicação e os dispositivos utilizados no sistema de telemetria desenvolvido.

2.1 AQUISIÇÃO DE DADOS

Um sistema de aquisição de dados de carro de corrida é uma unidade eletrônica que adquire parâmetros definidos pelo usuário em função do tempo enquanto o carro está na pista. Os dados adquiridos podem ser transferidos para um computador onde possam ser monitorados e analisados, geralmente com pacotes de software especializados.

A análise de dados pode ser dividida nas seguintes categorias:

- Performance do veículo;
- Performance do piloto;
- Desenvolvimento do veículo;
- Confiabilidade e segurança;
- Determinação de parâmetros do veículo;
- Registro de corridas.

Embora muitos dos sinais estejam inter-relacionados, os dados que o sistema mede podem ser divididos nas seguintes categorias:

- Funções vitais do veículo;
- Atividades do piloto;
- Parâmetros do chassi.

Sistemas de aquisição de dados existem para quase todas as aplicações. Uma configuração tradicional para o início da aquisição de dados consiste de uma unidade de registro adequada que mede os seguintes sinais para análise de chassi e piloto:

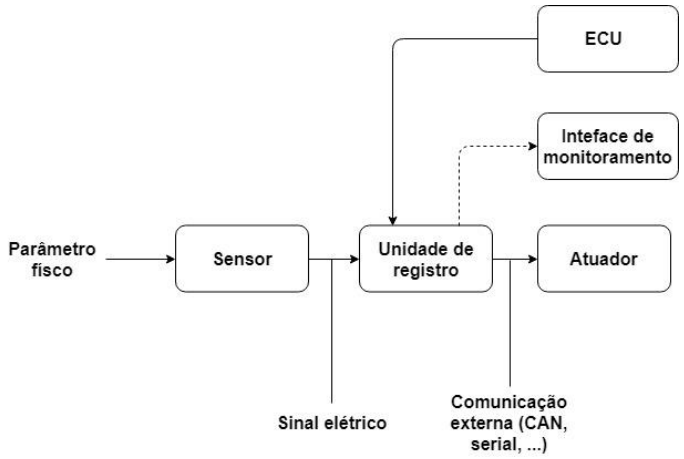
- RPM do motor;
- Velocidade do veículo;
- Posição do acelerador;
- Esterçamento do volante;
- Aceleração lateral e longitudinal.

Em adicional a estes sinais, as funções vitais do veículo (temperatura de componentes, pressão de fluídos, tensão da bateria, etc.) também devem ser registradas. Um canal de indicação deve ser providenciado para indicar o início e o fim da volta. Medindo estes seis sinais iniciais já possibilita aos engenheiros uma grande quantidade de dados para analisar.

Análises dinâmicas mais aprofundadas do veículo exigem uma quantidade maior de sinais a serem registrados. Uma longa lista pode ser mencionada, sendo que os engenheiros sempre se perguntam o que exatamente eles querem medir. Necessidades específicas requerem sinais específicos. A maioria das equipes inicia a aquisição de dados com os seis sinais básicos e estendem a lista conforme ganham mais experiência durante a análise de dados. Por isso, ao investir em um sistema de aquisição de dados, é preciso considerar que o número de sinais pode aumentar no futuro, impactando no chicote elétrico, memória disponível e dispositivos embarcados.

Sistemas de aquisição de dados estão disponíveis em várias configurações, mas normalmente possuem componentes básicos em comum, conforme a Figura 2.1.

Figura 2.1- Configuração geral de um sistema de aquisição de dados



Fonte: Traduzido de (SEGERS, 2008).

Um parâmetro físico (pressão, temperatura, velocidade, força, etc.) de interesse é capturado por um sensor que transforma a medida em um sinal elétrico proporcional a este parâmetro e entendível pela unidade de registro. Um dispositivo externo (computador por exemplo) pode se comunicar com a unidade de registro através de uma conexão externa. Esta conexão é normalmente bidirecional, pois a maioria dos sistemas atuais oferece a possibilidade de configurar alguns parâmetros pela escolha do usuário. Além disso, a comunicação CAN (*Controller Area Network*) vem se popularizando como um substituto para comunicações seriais ou paralelas devido à alta velocidade de comunicação e facilidade para se adicionar diferentes dispositivos a mesma rede (SEGERS, 2008).

2.2 SENSORES

Em sistemas de automação onde é preciso determinar as condições do ambiente a ser controlado e/ou monitorado, o processo de aquisição de dados é fundamental para o funcionamento correto do sistema. Para isso, utilizam-se instrumentos de medição para obter os valores das variáveis físicas deste ambiente, informando a um controlador sobre os eventos externos.

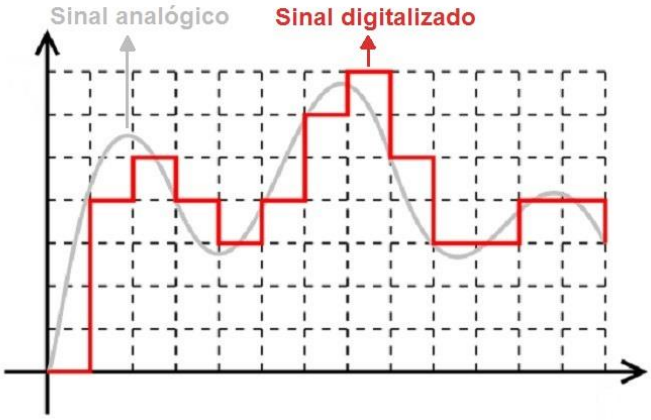
De uma forma simplificada, um instrumento de medição é um dispositivo que transforma uma variável física de interesse (pressão, temperatura, etc.) em um formato passível de medição por um instrumento. O elemento chave desse processo de medição é o sensor, que

tem a função de converter o sinal da variável física em um sinal da variável de saída apropriado (FRANCHI, 2015).

Sendo assim, os sensores são componentes que convertem grandezas físicas de diversas naturezas em variações de natureza elétrica (resistência, tensão ou corrente), podendo transmitir sinais analógicos ou digitais.

Sensores analógicos podem assumir qualquer valor no seu sinal de saída, desde que esteja dentro de sua faixa de operação. Alguma das grandezas que podem assumir sinais analógicos ao longo do tempo são: pressão, temperatura, velocidade, vazão, força, entre outros, enquanto que sensores digitais podem assumir apenas valores de um conjunto finito de valores no seu sinal de saída ao longo do tempo, que podem ser interpretados como zero ou um. Não existem naturalmente grandezas físicas que assumam esses valores, mas eles são assim mostrados ao sistema de controle após serem convertidos por um circuito eletrônico. (THOMAZINI; ALBUQUERQUE, 2005).

Figura 2.2- Comparação entre sinal analógico e digital



Fonte: techtudo.com.br.

A seguir, são tratados os sensores presentes no protótipo que são utilizados no sistema de telemetria.

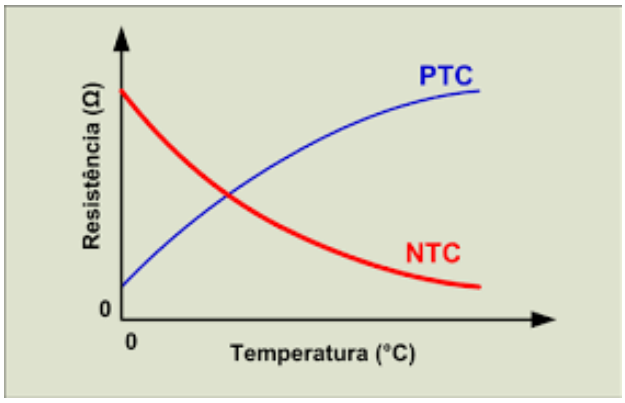
2.2.1 Temperatura

Os sensores de temperatura mais utilizados em indústrias, veículos e instalações prediais são os termistores, termopares e os termoresistores. Os termistores são resistores termicamente

sensíveis. São semicondutores eletrônicos cuja resistência elétrica varia com mudanças relativamente pequenas de temperatura, havendo duas variedades básicas deste tipo sensor: os de coeficiente positivo de temperatura (PTC) e os de coeficiente negativo de temperatura (NTC) (THOMAZINI; ALBUQUERQUE, 2005).

Nos termistores PTC, a resistência elétrica do material aumenta com o aumento da temperatura, enquanto que nos termistores NTC, a resistência elétrica diminui com o aumento da temperatura. Esta relação pode ser vista na Figura 2.3 a seguir.

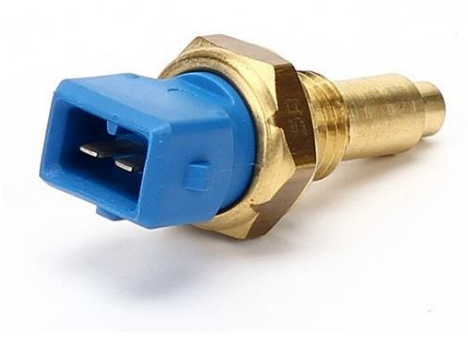
Figura 2.3- Comparação entre termistor PTC e NTC



Fonte: (PEREA; ANDRÉA; GONÇALVES VIANNA, 2010).

Os sensores de temperatura presentes no protótipo são termistores do tipo NTC (Figura 2.4) e são utilizados para medir a temperatura do óleo do motor, do líquido de arrefecimento, do ar da admissão e dos gases do escapamento.

Figura 2.4- Sensor de temperatura NTC

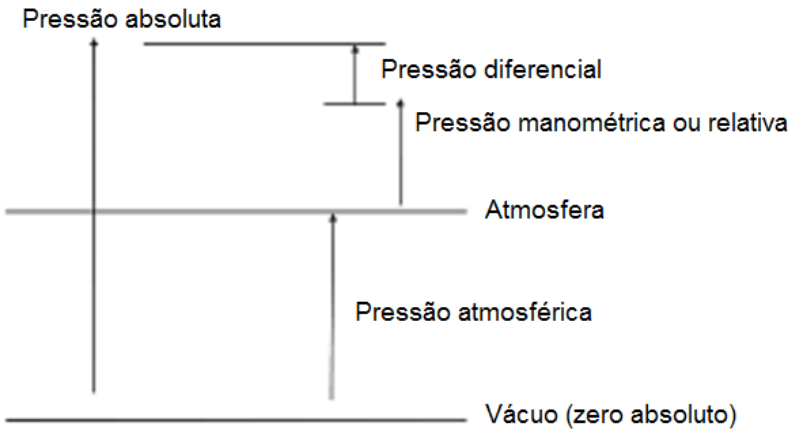


Fonte: lojapandoo.com.br.

2.2.2 Pressão

A medição e controle de pressão é a variável de processo mais usada na indústria de controle de processos nos seus mais diversos segmentos. Além disso, através da pressão é facilmente possível inferir uma série de outras variáveis de processo, tais como nível, volume, vazão e densidade. Em função da referência pode-se classificar a medição de pressão como: manométrica, absoluta e diferencial ou relativa (CASSIOLATO, 2018).

Figura 2.5- Referências de Pressão



Fonte: (JUNIOR, 2017).

As definições para os diferentes tipos de pressão mencionados na Figura 2.5 estão listadas abaixo:

- Pressão absoluta: é medida com relação ao vácuo perfeito, ou seja, é a diferença da pressão em um determinado ponto de medição pela pressão do vácuo (zero absoluto).
- Pressão diferencial: é a diferença de pressão medida entre dois pontos que não sejam os pontos de vácuo ou atmosfera.
- Pressão manométrica: é a diferença entre a pressão absoluta medida em um ponto qualquer e a pressão atmosférica. A pressão manométrica também é dada pela diferença entre a absoluta e a atmosférica.

Em geral, os sensores são classificados conforme a técnica usada na conversão mecânica da pressão em um sinal eletrônico proporcional. Estes sensores podem ser de capacitância variável, piezo-resistivos (*strain gauge*), potenciométricos, piezo-elétricos, entre outros.

Os sensores de pressão presentes no protótipo são utilizados para medir a pressão do óleo do motor, das linhas de fluído de freio, das linhas de combustível e do sistema de admissão de ar (MAP).

O sensor de pressão absoluta (Figura 2.6), ou MAP (*Manifold Absolute Pressure*), tem a função de informar as variações de pressão existentes no coletor de admissão e a pressão atmosférica local. Estas são informações importantes para o cálculo da massa de ar admitida e consequentemente para o controle de injeção de combustível (CAPELLI, 2010).

Figura 2.6- Sensor de pressão MAP

Fonte: boonstraparts.com.

2.2.3 Indutivo

Sensores indutivos (Figura 2.7) atuam com base no princípio da variação da indutância de uma bobina, quando um elemento metálico ou condutivo passa nas suas proximidades. O objeto metálico absorve parte do campo magnético e a variação da indutância da bobina é detectada pelo circuito do sensor que produz um sinal de saída, podendo ser a atuação de um contato NA (Normalmente Aberto) ou NF (Normalmente Fechado) para corrente alternada ou contínua, um transistor ou ainda um sinal variável de tensão ou de corrente (saída analógica) (FRANCHI, 2015).

Quando a atuação ocorre em um transistor, o sinal de saída é negativo em dispositivos NPN e positivo nos dispositivos PNP.

O protótipo da equipe conta com quatro sensores indutivos NPN que detectam os dentes de rodas fônicas feitas de material ferromagnético. Os sensores e as rodas fônicas são instalados nas rodas e são utilizados para calcular a velocidade do protótipo.

Figura 2.7- Sensor indutivo

Fonte: comatreleco.com.br

2.2.4 Posição

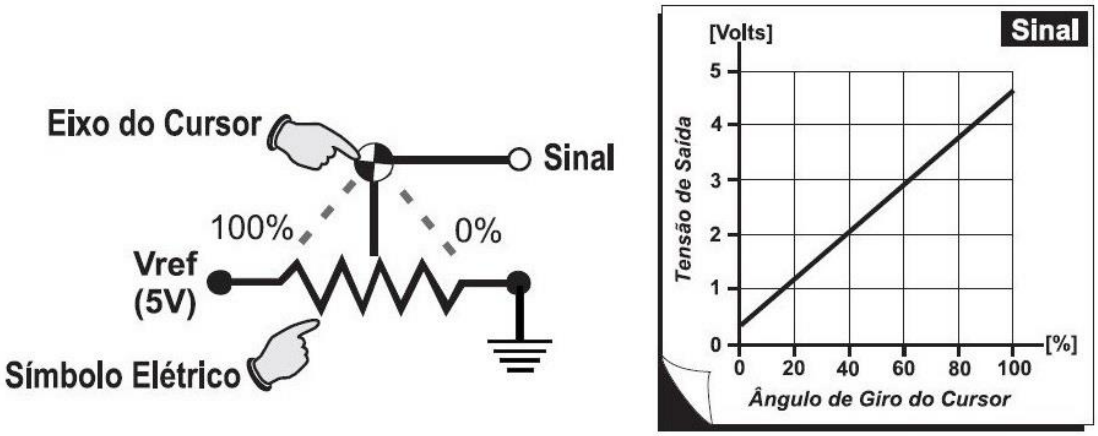
Dois métodos principais são utilizados para se detectar a posição de um objeto: por contato, através de chaves limitadoras e potenciômetros; ou sem contato com o objeto, através de sensores magnéticos, por efeito Hall, por ultrassom, entre outros.

No protótipo são utilizados sensores de posição por potenciômetro e por efeito Hall.

2.2.4.1 Potenciômetros

Potenciômetros são sensores que produzem uma resistência proporcional ao deslocamento ou posição, ou seja, são resistores variáveis de três terminais, sendo que dois são os extremos fixos, e o terceiro é o central (cursor) que pode deslocar-se de um extremo ao outro do resistor. Assim, alimentando os terminais extremos com uma tensão de referência, é possível medir, entre o terminal do cursor e qualquer um dos extremos, uma tensão variável que depende da posição do cursor (MTE-THOMSON, 2018).

Figura 2.8- Funcionamento de sensor de posição potenciométrico



Fonte: (MTE-THOMSON, 2018).

Os sensores de posição potenciométricos podem ser lineares, que medem o deslocamento linear de do dispositivo, ou circulares, que servem para medir o deslocamento angular de um eixo. No protótipo da equipe, é utilizado um sensor de posição linear instalado no setor de direção do veículo para medir o ângulo de esterçamento do volante, e um sensor de posição angular para medir a posição da borboleta do acelerador (TPS).

O sensor de posição da borboleta (Figura 2.9), ou TPS (*Throttle Position Sensor*), tem a função de traduzir o ângulo de abertura da borboleta de aceleração em um sinal elétrico que possa ser lido pela central eletrônica (ECU). Através deste sensor, a central eletrônica obtém informações de acelerações ou desacelerações realizadas pelo piloto, as quais são utilizadas no cálculo do tempo de injeção instantâneo e, consequentemente, no controle das condições de funcionamento do motor (CAPELLI, 2010).

Figura 2.9- Sensor de posição TPS

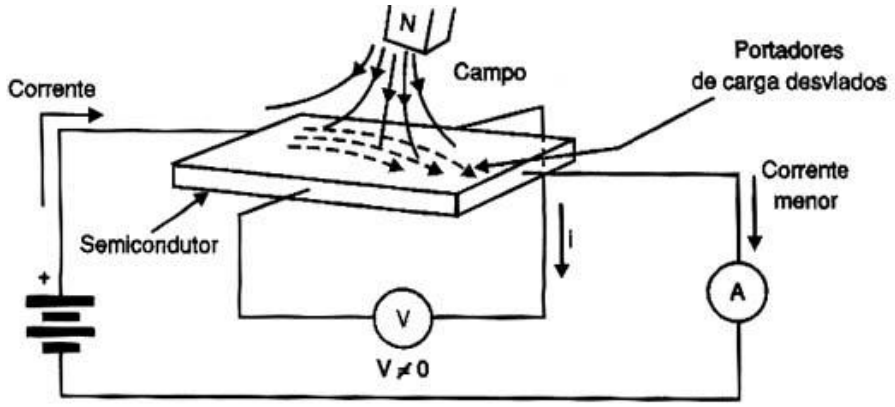
Fonte: boschautoparts.com.

2.2.4.2 Efeito hall

Os sensores de efeito Hall são muitos utilizados em sistemas embarcados para detectar posicionamento, velocidade, corrente, ponto de ignição, comutação, entre outras utilizações.

Uma camada semicondutora percorrida por corrente elétrica, imersa em um campo magnético normal (linhas de força perpendiculares à direção da corrente), gera nas suas extremidades uma diferença de potencial chamada de *Tensão Hall.* Se a intensidade da corrente permanece constante, a tensão gerada depende somente da intensidade do campo magnético. Para obter uma variação do campo magnético, o sensor é posicionado através de uma roda dentada. No movimento da roda dentada, o dente metálico da roda cobre o sensor, bloqueando o campo magnético e provocando uma redução do sinal de saída. Ao contrário, quando estiver junto a abertura, com o campo magnético presente, o sensor gera um nível de sinal alto na saída (CAPELLI, 2010).

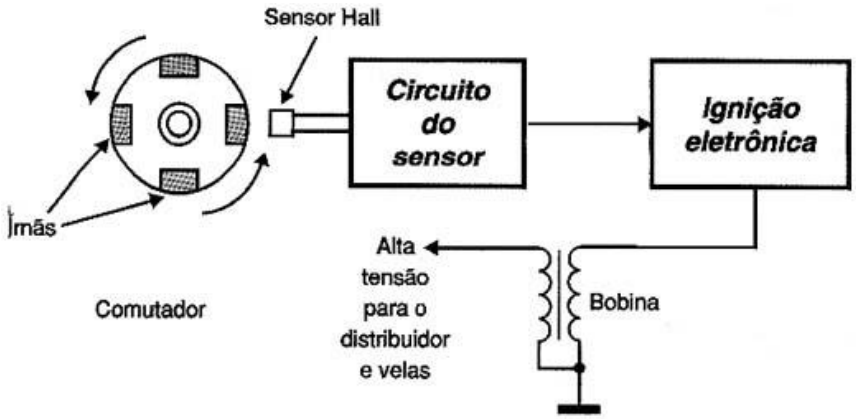
Figura 2.10- Semicondutor sob ação de campo magnético



Fonte: (BRAGA, 2018a).

Prendendo-se um ímã em qualquer peça móvel, pode-se detectar o movimento desta peça e medir a sua rotação ou ainda verificar a sua posição com a utilização de sensores magnéticos. Em um carro, por exemplo, um sensor magnético mede com precisão a rotação do motor fazendo o acionamento das velas (BRAGA, 2018a).

Figura 2.11- Utilização de sensor de efeito Hall em um motor



Fonte: (BRAGA, 2018a).

No motor utilizado pela equipe, internamente são utilizados dois sensores de efeito Hall: o sensor de referência do ponto morto superior (*Ref*) que tem a função de informar a rotação do motor, e o sensor de fase (*Sync*) que tem a função de informar a referência de fase do primeiro cilindro para determinar o momento de injeção e/ou ignição (sequencial). Com estes sensores é possível saber o ângulo do virabrequim e, consequentemente, a posição de cada pistão.

2.2.5 Lambda

A mistura entre o ar e o combustível é chamada de ideal ou estequiométrica quando a quantidade de ar é a mais adequada para prover a queima completa do combustível. Todo o gerenciamento eletrônico do motor tem como principal objetivo atingir a estequiometria. O fator lambda (λ) é a relação entre a quantidade de mistura aspirada pela quantidade de mistura necessária para que se tenha a relação ideal. No caso da gasolina, a relação ideal é de 14,7 partes de ar para 1 de combustível (CAPELLI, 2010).

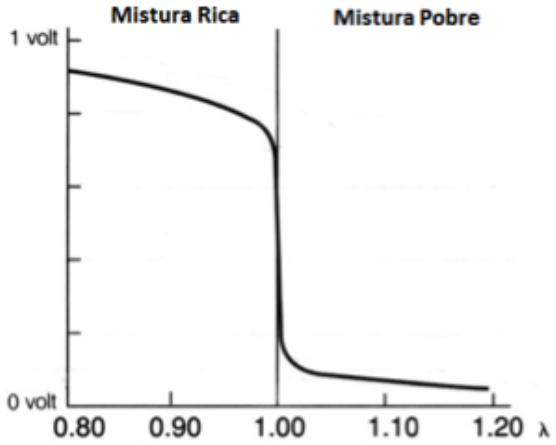
$$\lambda = \frac{ar/combustível}{ar/combustível\ estequiométrico} \quad (1)$$

- $\lambda = 1$: mistura ideal ou estequiométrica;
- $\lambda < 1$: mistura rica (mais combustível e menos ar);

- $\lambda > 1$: mistura pobre (mais ar e menos combustível).

Uma relação entre o fator lambda e a tensão gerada pelo sensor pode ser visualizada graficamente na Figura 2.12.

Figura 2.12- Fator lambda (relação da mistura ar/combustível)



Fonte: protuneelectronics.com.br.

A sonda lambda (Figura 2.13), ou sensor de oxigênio, tem como principal elemento o zircônio e sua função é medir o conteúdo de oxigênio nos gases de descarga e informar a central eletrônica para que ela possa corrigir a quantidade de combustível injetado no motor. A superfície externa do elemento de zircônio está em contato com o gás de escape enquanto que a interna está em contato com o ar atmosférico. Ambas estão revestidas por uma camada fina de platina, que age como eletrodo para conduzir o sinal do sensor até os fios condutores (CAPELLI, 2010).

Figura 2.13- Sensor lambda

Fonte: protuneelectronics.com.br.

O sensor lambda pode ser do tipo *narrowband,* onde a mistura é identificada apenas como rica ou pobre, ou do tipo *wideband*, onde o sensor pode ser usado para medir o fator lambda em outras faixas além do estequiométrico, proporcionando a possibilidade de controlar a mistura para que o motor trabalhe com misturas mais pobres ou mais ricas. Este tipo de controle é chamado de controle em malha fechada.

Além disso, visto que o elemento zircônio apenas produz tensão a temperaturas superiores a 300°C, o gás de escape demora alguns minutos para aquecer o sensor a esta temperatura após o acionamento do motor. Para reduzir este tempo de espera, muito sensores possuem aquecedores internos de cerâmica.

2.2.6 Acelerômetro e giroscópio

O acelerômetro é um dispositivo utilizado para medir a aceleração própria de um sistema, fornecendo uma saída que é proporcional à aceleração a ser medida. Acelerômetros, tipicamente, convertem energia mecânica em um sinal elétrico mensurável, onde geralmente a sensibilidade é dada em mV por g (unidade de medida de força de aceleração).

Enquanto que o giroscópio mede a alteração da taxa angular em torno de um único eixo do veículo. Sendo assim, o posicionamento do sensor é de extrema importância, devendo ser instalado o mais próximo possível do centro de gravidade do veículo, de modo que a parte superior do sensor esteja paralela ao solo para uma aquisição de dados com elevada exatidão.

Estes dois sensores, vide Figura 2.14, são utilizados para adquirir dados angulação e de aceleração lateral sofrida pelo protótipo durante as curvas, ajudando a desenvolver o projeto do chassi e configurar o melhor *setup* de suspensão do veículo, por exemplo.

Figura 2.14- Sensores acelerômetro e giroscópio, respectivamente

Fonte: me-mo-tec.com.

2.2.7 GPS

Os sensores de GPS são receptores com antenas que usam um sistema de navegação baseado em uma rede de 24 satélites em órbita ao redor da Terra para fornecer informações de posição, velocidade e tempo.

Em corridas de circuito, a unidade de GPS é uma boa alternativa ao tradicional sistema de cronometragem de voltas. A posição da linha de partida / chegada pode ser marcada para calcular o tempo de volta e fornecer informações de velocidade para fins de exibição e registro.

A unidade GPS utilizada pela equipe é o GPS-L10 (Figura 2.15) desenvolvido pela MoTeC, mesma empresa que desenvolve a ECU utilizada pela equipe, e trata-se de uma unidade GPS de 10 Hz, adequada apenas para uso com produtos MoTeC (MOTEC, 2011).

Figura 2.15- GPS-L 10 MoTeC

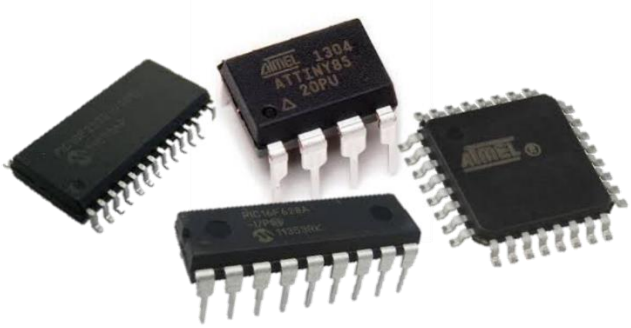
Fonte: (MOTEC, 2011).

2.3 MICROCONTROLADORES

Um microcontrolador (Figura 2.16) é um componente eletrônico capaz de efetuar processos lógicos com extrema rapidez e precisão. A grande vantagem deste componente é a sua possibilidade de programação, o que o torna adaptável à finalidade desejada, e que possibilita seu ajuste de acordo com a tarefa que deverá executar. Utilizada no gerenciamento de processos lógicos, esse gerenciamento pode ser entendido como o controle de periféricos, como: sensores, relês, resistências, *displays*, LED's, entre outros.

Um microcontrolador se diferencia de um microprocessador pela sua funcionalidade. Quanto a um microprocessador, para que ele possa ser usado, outros componentes devem ser adicionados, tais como memória e componentes para receber e enviar dados. Já o microcontrolador foi projetado para ter tudo isso em um só componente. Nenhum outro componente externo é necessário nas aplicações, uma vez que todos os periféricos necessários já estão contidos nele (ASSIS, 2004).

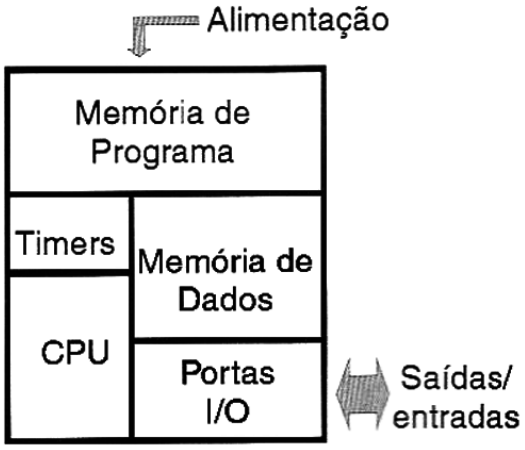
Figura 2.16- Microcontroladores



Fonte: vandertronic.com.

Para entender melhor o funcionamento do microcontrolador, é interessante analisar sua estrutura interna, mostrada na Figura 2.17.

Figura 2.17- Estrutura interna dos microcontroladores



Fonte: (BRAGA, 2018b).

O bloco mais importante é a Unidade Central de Processamento ou CPU (*Central Processing Unit*) que consiste num processador capaz de processar, calcular e realizar operações lógicas com sinais digitais. A CPU faz isso através de um programa próprio. Os microcontroladores podem processar informações através de um conjunto de instruções que são reconhecidas quando as informações chegam. As informações devem ser enviadas na forma digital para que a CPU entenda e execute as instruções que foram programadas.

Os microcontroladores possuem duas memórias. A primeira é a memória de programa que serve para armazenar as instruções ou programa que dizem o que o microcontrolador deve fazer. Essas memórias são do tipo EEPROM ou *Flash*, ou seja, memórias que retém a informação mesmo quando o microcontrolador é desligado e podem ser reprogramadas quando o ligamos a um computador para programar uma nova aplicação. A segunda é a memória de dados, onde ficam as informações que o microcontrolador recebe durante seu funcionamento, como por exemplo, as leituras dos sensores.

Um outro bloco importante a ser analisado é o dos *timers* ou temporizadores. Muitas das funções exercidas pelos microcontroladores exigem a utilização de intervalos de tempo precisos. Para esta finalidade, existe um bloco especial de programação em que isso pode ser feito uma vez que o processamento do microcontrolador é muito rápido determinado pelo *clock*. Este *clock* comanda as operações que podem chegar a milhares ou milhões por segundo. Com a ajuda do *timer*, podem ser obtidos ciclos de controle muito mais lentos.

As portas estão no bloco seguinte da estrutura básica consistindo no modo que o microcontrolador tem para se comunicar com o mundo exterior. As portas de entrada e saída (I/O = *Input/output*) são acessadas através de pinos do circuito integrado do microcontrolador e nas placas através de conectores onde pode-se fazer as ligações dos circuitos externos.

Estes microcontroladores para poderem ser usados e programados precisam ser montados em placas apropriadas e a sua programação se faz com a conexão num computador. Assim, estes microcontroladores podem ser adquiridos montados em pequenas placas que já possuem todos os recursos para a entrada de programação através de um simples cabo e dotados de conectores para se ligar os circuitos que devem ser controlados, os sensores ou dispositivos de controle e a fonte de alimentação (BRAGA, 2018b).

2.3.1 Arduino

O Arduino é uma placa microcontrolada de *hardware* livre composta por um microcontrolador Atmel, circuitos de entrada/saída e que pode ser facilmente conectada à um computador e programada através de um Ambiente de Desenvolvimento Integrado, ou IDE (*Integrated Development Environment),* utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB (THOMSEN, 2014).

O Arduino UNO R3 (Figura 2.18) é um modelo de placa baseada no microcontrolador ATmega328 e conta com 32 kB de memória *Flash* (0,5 kB são utilizados para o *bootloader*), 2 kB de memória SRAM e 1 kB de memória EEPROM. Possui 14 pinos de entrada e saída digital, onde 6 destes podem ser utilizados como saídas PWM (*Pulse Width Modulation*) e 6 entradas analógicas. Sua alimentação é via USB ou fonte de alimentação externa, funcionando de 6 V a 20 V. Seus pinos digitais operam com 5 V, fornecendo uma corrente de no máximo 40 mA, enquanto que os pinos analógicos variam de 0 V a 5 V.

Figura 2.18- Arduino UNO R3

Fonte: (THOMSEN, 2014).

2.4 COMUNICAÇÃO

Nesta seção, são apresentadas as tecnologias e protocolos de comunicação utilizados para o desenvolvimento do sistema de telemetria e aplicativo.

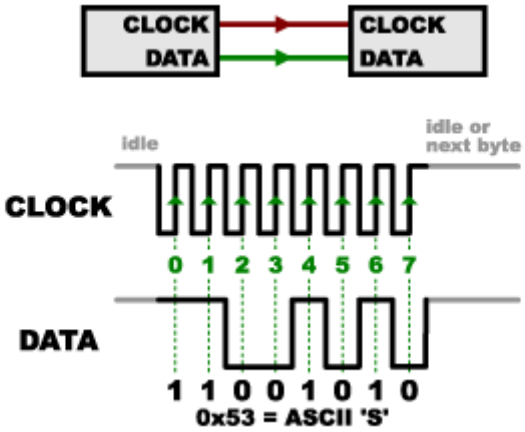
2.4.1 Serial

Comunicação serial, ou série, é o processo de enviar dados um *bit* de cada vez, sequencialmente, num canal de comunicação ou barramento. É diferente da comunicação paralela em que todos os *bits* de cada símbolo são enviados juntos. A comunicação serial é usada em toda comunicação de longo alcance e na maioria das redes de computadores, onde o custo de cabos e as dificuldades de sincronização tornam a comunicação paralela impraticável.

A comunicação serial é utilizada largamente e tem grande importância na automação, pois grande parte dos equipamentos ainda utilizam essa tecnologia para comunicação seja através dos padrões RS-232, RS-485, Ethernet, USB, entre vários outros. A comunicação é realizada através da transmissão de informação *bit* a *bit* sequencialmente, normalmente de caracteres ASCII, através de uma única via (condutor) ou por vias de transmissão e recepção chamados de RX e TX (MECAWEB, 2016).

A transmissão de dados pode ser feita por dois métodos diferentes: síncrono ou assíncrono. O método síncrono de comunicação depende de um sinal de *clock*, ou seja, cada *bit* ou conjunto de *bits* enviado depende de um pulso do *clock*, tendo como principal vantagem sua velocidade de transmissão de dados, sendo necessário um condutor extra para o *clock*.

Figura 2.19- Comunicação serial síncrona

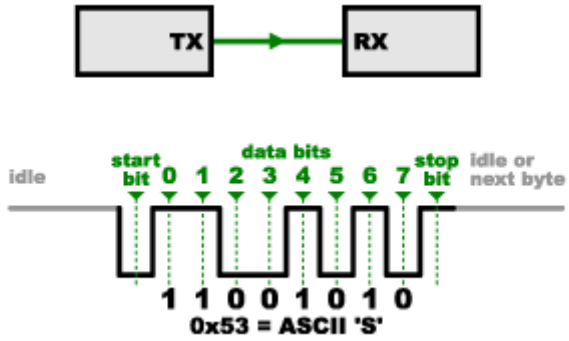


Fonte: (ROBOCORE, 2016).

O método assíncrono não precisa de um sinal de *clock*, portanto o número de condutores necessários é menor. Contudo, o envio dos dados é mais complicado e susceptível a erros, por isso alguns parâmetros são necessários para garantir o envio sem erros, como o *Baud Rate* que

especifica a velocidade de recepção e envio. É importante que os dois dispositivos utilizem a mesma taxa (ROBOCORE, 2016).

Figura 2.20- Comunicação serial assíncrona



Fonte: (ROBOCORE, 2016).

2.4.1.1 CAN

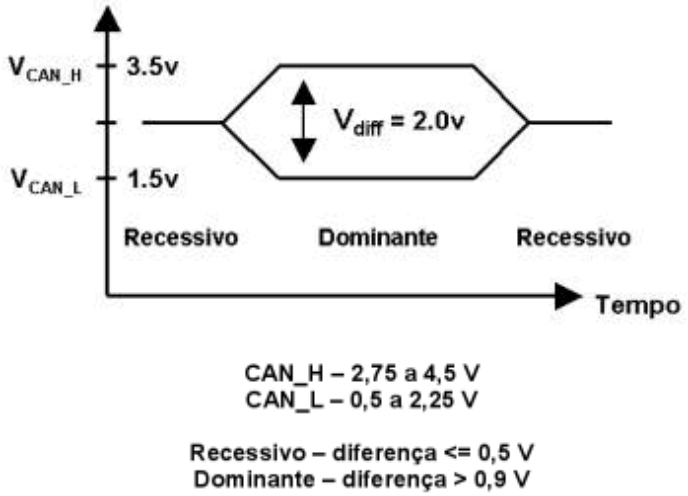
O CAN é um padrão de comunicação serial síncrono que utiliza uma topologia de rede do tipo barramento (CANbus). Baseado no conceito multimestre, onde todas as estações (nodos) são consideradas mestres, com publicação de mensagens no barramento em caráter *broadcast/multicast*, ou seja, não há um destinatário único ao qual a mensagem é enviada. O protocolo CAN utiliza o algoritmo de acesso ao meio (MAC) com contagem regressiva binária, também chamado de *CSMA/CA with NDA* (*Non Destructive Arbitration*) ou ainda *CSMA/AMP* (*Arbitration on Message Priority*), onde todas as estações disputam o barramento sempre que este estiver livre, comparando prioridades e não havendo colisão entre as mensagens. Além disso, a taxa de transmissão de dados é inversamente proporcional ao comprimento do barramento, tendo um máximo de 1 Mbps com até 40m e um mínimo de 5 kbps com até 1500m.

Geralmente são utilizados cabos de par trançado sem blindagem como meio físico, contudo, o protocolo CAN possibilita a utilização de outros meios físicos, como o uso de fibra óptica, ou ainda, radiofrequência (RF). A comunicação CAN é diferencial com sinalização entre condutores chamados de CAN_High e CAN_Low. Os dados são transmitidos com *bits* chamados de dominantes (nível lógico 0) e recessivos (nível lógico 1), onde esta característica é chamada de *Wired AND* devido a operação binária "E" intrínseca. *Bits* dominantes são

transmitidos com diferença de tensão entre CAN_High e CAN_Low, enquanto nos *bits* recessivos a tensão é aproximadamente igual.

A Figura 2.21 a seguir ilustra os níveis de sinal do barramento CAN.

Figura 2.21- Níveis de sinal do barramento CAN

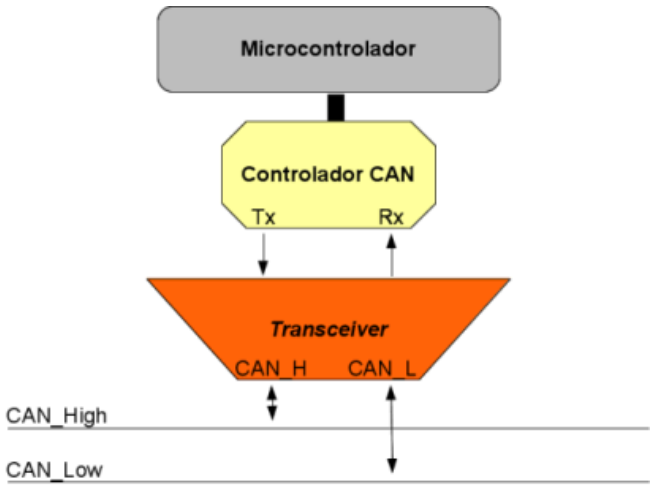


Fonte: (SCHAF, 2013).

Por ser um protocolo onde não há endereçamento, cada estação terá um filtro de identificador de mensagens que as interessam, uma vez que todas as mensagens de todos os nodos estão normalmente disponíveis, mas nem sempre todas as estações se interessam por todas as mensagens. Esta é a essência da característica *publisher/subscriber* (publicador e leitor).

A Figura 2.22 abaixo mostra os três elementos básicos que fazem o interfaceamento com dos serviços do protocolo CAN.

Figura 2.22- Elementos de interfaceamento do protocolo CAN



Fonte: (SCHAF, 2013).

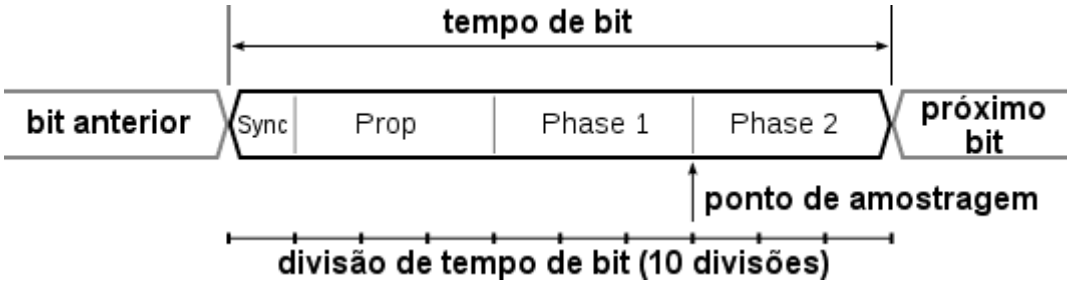
O *transceiver* é o elemento que faz a interface direta com o barramento CAN. Pode-se dizer que é nele que está implementada a camada física do barramento, possuindo normalmente saídas/entradas CAN_High e CAN_Low. Logo acima do *transceiver* está o controlador CAN, responsável pelos serviços das camadas de transferência e de objeto onde são implementados o MAC, o filtro, o encapsulamento de mensagens, etc. O controlador CAN se comunica com o *transceiver* via canais de recepção e transmissão (Rx e Tx), recebendo informações do barramento através do Rx vindas do *transceiver* enquanto que envia quadros para serem transmitidos através do Tx. Acima do controlador CAN está o microcontrolador que faz o processamento das mensagens e dos dados transmitidos no barramento. É neste elemento que são programadas as regras de controle do sistema de automação ligado a camada de aplicação.

Os quadros (mensagens) do CAN possuem um campo de dados flexível de 0 até 8 *bytes* com um identificador de mensagens de 11 ou 29 *bits*, dependendo da especificação, diretamente ligado a prioridade da mensagem.

O CAN utiliza a codificação de linha NRZ (*non-return to zero*), ou seja, o sinal não volta a um nível zero após o tempo de *bit*. Sendo assim, a sincronia é obtida conforme a aquisição dos *bits* da mensagem, onde cada nodo possui seu próprio *clock*, não havendo um sinal exclusivo de sincronia.

A Figura 2.23 a seguir mostra como é feita a sincronia com base nos tempos de *bit*.

Figura 2.23- Sincronização com base nos tempos de bit

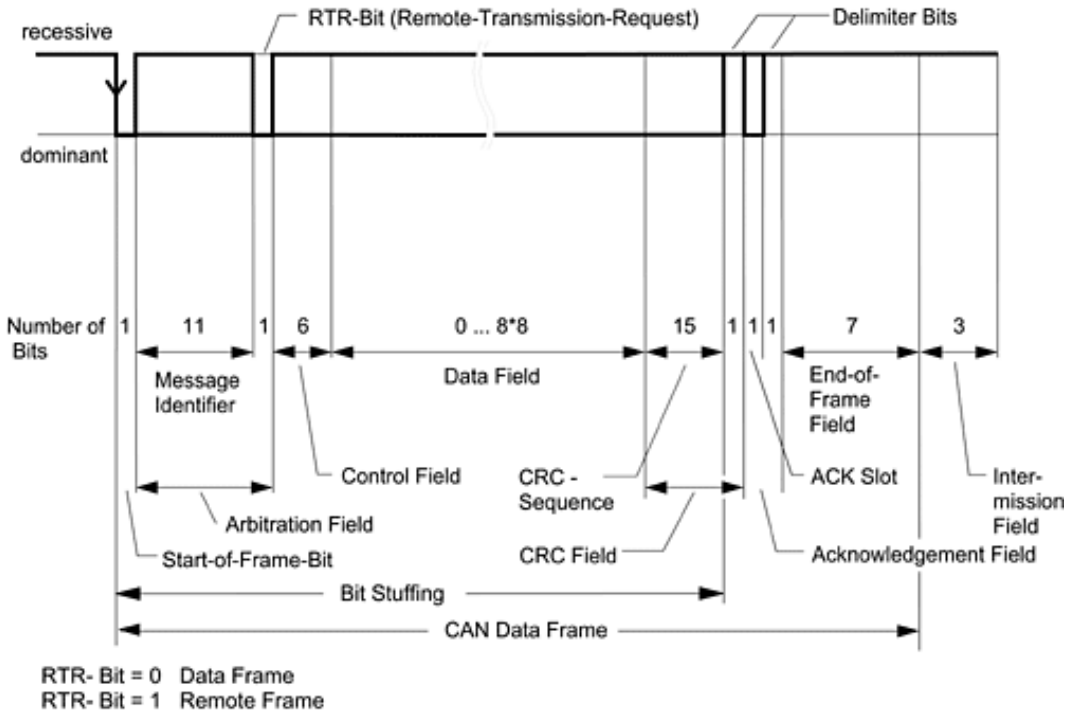


Fonte: (SCHAF, 2013).

Nota-se que o tempo de *bit* é dividido em 10 partes e há 4 fases no tempo de *bit*: sincronia (Sync), propagação (Prop), e as fases 1 e 2. Entre as fases 1 e 2, na 7ª divisão do tempo de *bit*, é feita uma amostragem que possibilita a sincronização dos nodos.

O protocolo CAN de baixo nível estabelece 4 tipos de quadros: de dados, remoto, de erro e de sobrecarga. O quadro de dados (*data frame*) é o quadro específico do nodo quando este deseja "publicar" seus dados (ações de controle, medidas de sensores, etc.). O quadro na especificação CAN2.0A é ilustrado na Figura 2.24 a seguir.

Figura 2.24- Quadro de dados do protocolo CAN



Fonte: (SCHAF, 2013).

Este quadro é composto pelos seguintes campos:

SOF *(Start of Frame*): sinaliza o início do quadro e é sempre um *bit* dominante (0).

Message Identifier: identificador da mensagem.

RTR (*Remote Transmission Request*): *flag* de solicitação de transmissão remota.

IDE (*Identifier Extension bit*): identificador de extensão do campo identificador.

R0 (*Reserved bit*): *bit* reservado e sempre dominante (0).

DLC (*Data Length Code*): comprimento do campo de dados em *bytes*.

Data field: campo de dados do nodo.

CRC: código de Hamming que possibilita detectar até 6 erros e corrigir 1.

CRC delimiter: delimitador do CRC. É sempre recessivo (1).

ACK slot: campo de confirmação.

ACK delimiter: delimitador do campo de confirmação, sendo sempre recessivo (1).

EOF (*End of Frame*): sinalizador do término do quadro que possui sempre bits recessivos (1).

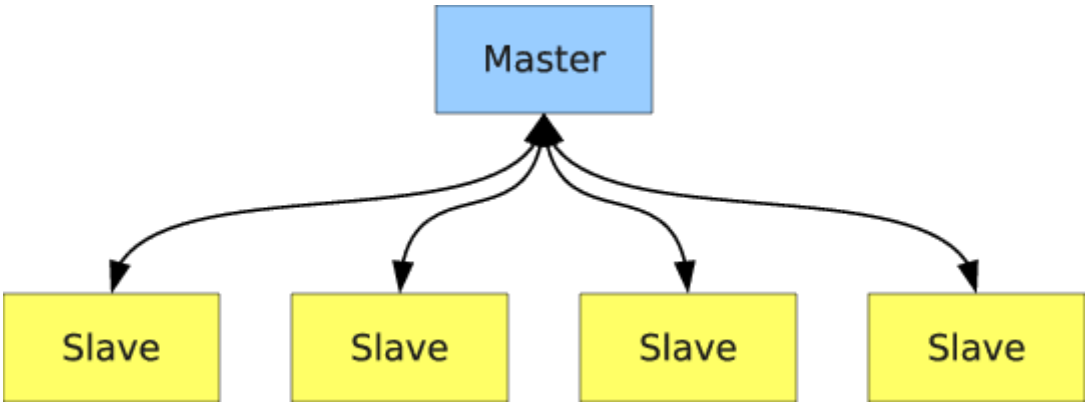
IFS (*Inter Frame Space*): tempo requerido pelo controlador CAN para processamento.

Além disso, toda rede CAN possui 2 terminadores. Estes terminadores nada mais são que resistores entre 120 e 124 ohms, conectados as pontas da rede para garantir a perfeita propagação dos sinais elétricos pelos fios da mesma (SCHAF, 2013).

2.4.1.2 SPI

A comunicação SPI é uma comunicação serial síncrona, onde é definido o conceito de Mestre-Escravo. Normalmente o gerador do sinal de sincronismo é definido como o mestre (*Master*) da comunicação. Para os dispositivos que utilizam do sinal de sincronismo gerado damos a definição de escravo (*Slave*). A ligação mais comum desse tipo de comunicação é um mestre com vários escravos. Essa ligação é representada na Figura 2.25 a seguir.

Figura 2.25- Conceito Mestre-Escravo

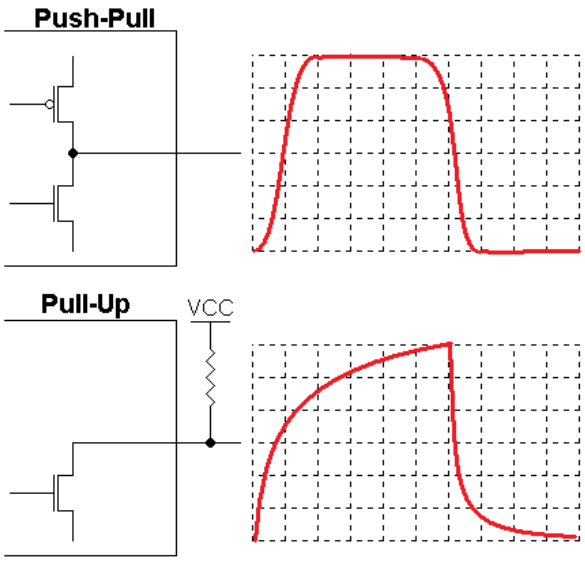


Fonte: (SACCO, 2014).

A comunicação SPI possui algumas características básicas. Primeiramente os sinais de comunicação possuem uma direção fixa e definida. Isso significa que sempre existem dois transistores definindo o estado de um pino (*Push-Pull*). Essa característica é uma das grandes diferenças entre outras comunicações seriais como I2C e OneWire, que possuem um mesmo barramento de dados para os sinais de entrada e saída através do esquema de dreno-aberto (*Pull-Up*).

Apesar de utilizar dois sinais de comunicação de dados em vez de um, é possível atingir velocidades maiores de comunicação, uma vez que há pouca deformação do sinal, como pode ser observado na Figura 2.26 abaixo.

Figura 2.26- Comparativo entre sinal Push-Pull e Pull-Up



Fonte: (SACCO, 2014).

Outra característica da comunicação SPI é que toda troca de dados acontece sempre em ambas as direções. Dessa forma, a comunicação é sempre *full-duplex*, onde os dois dispositivos transmitem dados simultaneamente em ambos os sentidos, mas em canais diferentes.

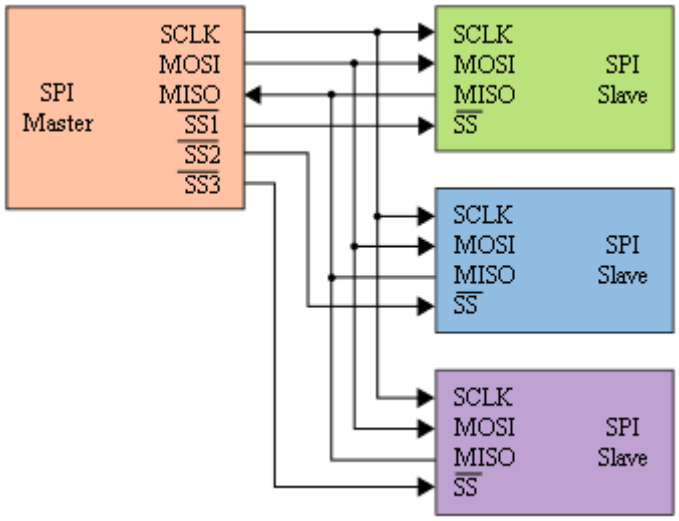
A Tabela 1 representa os pinos básicos de comunicação entre dispositivos SPI, enquanto que a Figura 2.27 demonstra o esquema padrão de ligação dos dispositivos com comunicação SPI.

Tabela 1- Pinos para comunicação SPI

Pino	**Nome padrão**	**Significado**	**Nomes alternativos**
Do Master para o Slave	MOSI	Master Output Slave Input	SDO, DO, SO
Do Slave para o Master	MISO	Master Input Slave Output	SDI, DI, SI
Clock	SCLK	Serial Clock	SCK, CLK
Seleção de Slave	SS	Slave Select	CS, nSS, nCS

Fonte: (SACCO, 2014).

Figura 2.27- Esquema de ligação dos dispositivos SPI



Fonte: (SACCO, 2014).

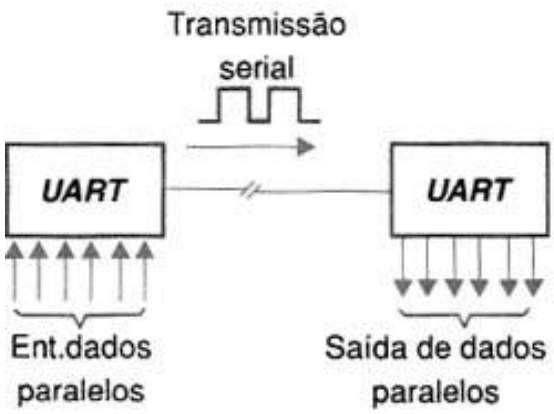
O pino SS funciona como seleção de escravo (*Slave Select*) e é um sinal ativo em nível baixo, ou seja, o dispositivo é selecionado quando este pino se encontra em nível baixo. No

entanto, muitos dispositivos utilizam este sinal como sincronismo de *frame*. Dessa forma, é um sinal importante que deve ser respeitado (SACCO, 2014).

2.4.1.3 UART

A comunicação UART (*Universal Asynchrounous Receiver/Transmiter*) é um elemento básico de qualquer sistema de comunicação serial e tem como finalidade possibilitar a transmissão e a recepção de dados originalmente disponíveis na forma paralela, convertendo os dados de modo que eles sejam transmitidos sequencialmente por uma linha de dados (transmissão serial) recuperando-os em sequência para serem apresentados de forma paralela na saída, de acordo com a Figura 2.28.

Figura 2.28- Transmissão de dados UART

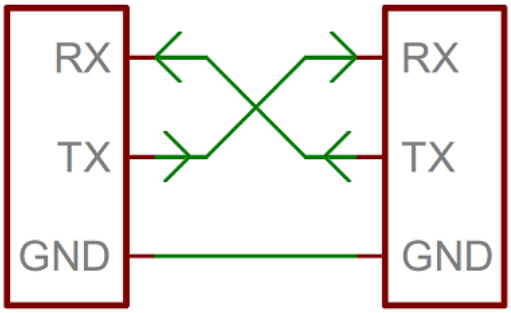


Fonte: (BRAGA, 2018c).

No entanto, precisa-se levar em conta que essa transmissão deve ser feita de modo seguro, com um controle de paridade para garantir a integridade dos dados e formas de sinalização do lado que transmite para que o lado que recebe saiba onde começa e onde termina uma transmissão (BRAGA, 2018c).

Sendo assim, o pino de transmissão (Tx) do protocolo envia um pacote de *bits*, chamado de caractere, que será interpretado *bit* a *bit* pelo receptor. Cada pacote enviado contém 1 *start bit* que indica o início da mensagem, 1 ou 2 *stop bit*s para indicar o final da mensagem, 5 a 9 *bits* de informação e 1 *bit* de paridade para evitar a recepção de erros. Além disso, por ser uma comunicação assíncrona, a comunicação é feita por dois pinos Rx/Tx que dependem da taxa de transmissão (*baud rate)* como referência.

Figura 2.29- Ligação para conexão UART



Fonte: (ROBOCORE, 2016).

A comunicação UART é um protocolo utilizado por muitos microcontroladores, pois é responsável pela conversão da comunicação paralela em serial, que na maioria das vezes é convertida em outro protocolo como por exemplo o controlador da placa Arduino Uno, que utiliza o protocolo UART mas tem o protocolo convertido para USB (ROBOCORE, 2016).

2.4.1.4 USB

USB (*Universal Serial Bus*) trata-se de uma tecnologia que tornou mais simples, fácil e rápida a conexão de diversos tipos de aparelhos (câmeras digitais, HDs externos, *pendrives*, *mouses*, teclados, impressoras, *scanners*, leitor de cartões, etc.) ao computador e a dispositivos móveis, evitando assim o uso de um tipo específico de conector para cada equipamento.

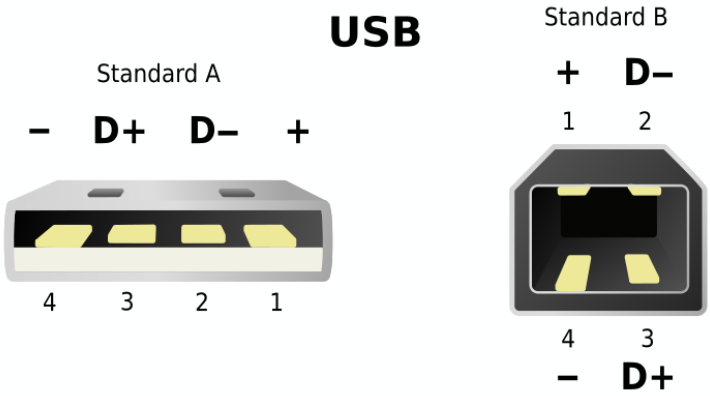
É possível conectar até 127 dispositivos ao mesmo tempo em uma única porta USB. Isso pode ser feito, por exemplo, por meio de *hubs*, dispositivos que utilizam uma única conexão USB para oferecer um número maior delas.

Os cabos USB podem ter até 5 metros de comprimento. Isso é necessário porque, em cabos maiores, o tempo de transmissão dos dados pode exceder o limite de 1.500 ns, onde a informação é considerada perdida. Esse limite pode ser aumentado com uso de *hubs* ou de equipamentos capazes de repetir os sinais da comunicação.

Além disso, o barramento USB pode ser utilizado para prover energia elétrica a determinados dispositivos. Para que isso seja possível, os cabos USB contam com pelo menos quatro fios internos: Vcc, D+, D- e GND. O primeiro é o responsável pela alimentação elétrica,

o segundo e o terceiro são utilizados na transmissão de dados, e o quarto, por sua vez, atua no controle elétrico, servindo como "fio terra".

Figura 2.30- Ligação de conectores USB



Fonte: (ROBOCORE, 2016).

A comunicação entre dispositivos conectados via USB é feita por meio de um protocolo. Nele, o *host* (computador ou equipamento que recebe as conexões) emite um sinal para encontrar os dispositivos conectados e estabelece um endereço para cada um deles. Uma vez estabelecida a comunicação, o *host* recebe a informação sobre o tipo de conexão que o dispositivo conectado utiliza (ALECRIM, 2017).

Os dispositivos que usam este protocolo precisam enviar 3 pacotes para realizar o envio de dados. Primeiro é enviado um *Token Packet* que informa o que será realizado na comunicação, se a informação será escrita ou lida e o endereço do dispositivo ao qual a mensagem será direcionada. Segundo, um *Data Packet* que é o pacote de dados que será escrito ou lido dependendo do comando dado anteriormente. E por último um *Handshaking Packet* que informa se os dois primeiros pacotes foram enviados corretamente (ROBOCORE, 2016).

2.4.1.5 Modbus

O Modbus é um dos protocolos mais utilizados em automação industrial, graças à sua simplicidade e facilidade de implementação, podendo ser utilizado em diversos padrões de meio físico, como RS-232, RS-485 e *Ethernet* TCP/IP (MODBUS TCP). A velocidade de comunicação varia em cada um desses padrões, bem como o comprimento máximo da rede e o número máximo de dispositivos conectados.

O padrão RS-232 (*Recommendad Standart-232*) é utilizado apenas em comunicações do tipo ponto a ponto, ou seja, só admite dois dispositivos na rede, que no caso do protocolo Modbus representa o mestre e um escravo. A velocidade máxima desse padrão fica em torno de 115kbps, enquanto que a distância máxima entre os dispositivos da rede fica em torno de 30m.

O padrão RS-485 (*Recommendad Standart-485*) é muito utilizado na indústria e sem dúvida é um dos padrões mais utilizados pelo protocolo Modbus. Esse padrão permite trabalhar com taxas de comunicação que podem chegar a 12Mbps e em alguns casos até 50Mbps. Quanto maior o comprimento da rede menor será a velocidade de comunicação. Sendo assim, a distância máxima da rede está em torno de 1200m e o número máximo de dispositivos no barramento da rede é de 32.

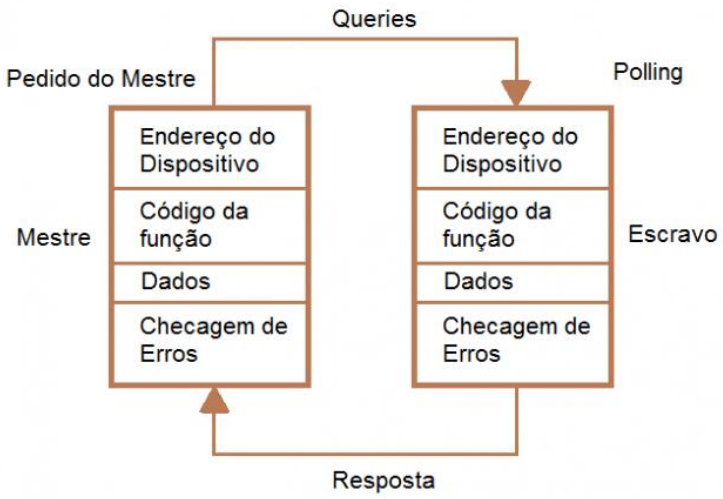
O padrão *Ethernet* no protocolo Modbus possui algumas variações, podendo chegar a 100Mbps ou até 10Gbps. A distância máxima pode variar de 100m até próximo de 200m dependendo do tipo de cabo utilizado e das condições de instalação do mesmo. Ao utilizar o meio físico *Ethernet*, o protocolo Modbus opera com o mecanismo de controle de acesso próprio da rede *Ethernet*, com mensagens no modelo cliente-servidor.

Em alguns casos é possível utilizar redes em fibra ótica, fato que permite alcançar distâncias maiores e maiores taxas de comunicação, bem como utilizar comunicação *wireless*.

A comunicação inicia com a estação mestre solicitando que os escravos enviem seus dados. Os escravos, por sua vez, recebem a requisição do mestre e retornam os dados solicitados. Os dados transmitidos podem ser discretos ou numéricos, ou seja, é possível enviar valores numéricos como temperatura e pressão ou enviar um bit para ligar e desligar um motor.

Na Figura 2.31 pode-se observar como é constituído o quadro de mensagens no protocolo Modbus.

Figura 2.31- Quadro de mensagem Modbus



Fonte: (FREITAS, 2014)

Na especificação do protocolo estão definidos dois modos de transmissão: ASCII e RTU (*Remote Terminal Unit*). Estes modos definem a forma como são transmitidos os *bytes* da mensagem e como a informação será empacotada na mensagem e descompactada no recebimento, não sendo possível utilizar os dois modos de transmissão na mesma rede.

Ao todo o protocolo Modbus possui 256 endereços onde 0 (Zero) é o endereço de *broadcast* (todos os escravos recebem a mensagem), 1 até 247 são os endereços disponíveis para os escravos e de 248 até 255 são os endereços reservados.

O código da função é onde o mestre especifica o tipo de serviço ou função solicitada ao escravo (leitura, escrita, etc). No protocolo Modbus, cada função é utilizada para acessar um tipo específico de dado de acordo com a Tabela 2.

Tabela 2- Funções no protocolo Modbus

Código da função	**Descrição**
1	Leitura de bloco de *bits* do tipo *coil* (saída discreta).
2	Leitura de bloco de *bits* do tipo entradas discretas.
3	Leitura de bloco de registradores do tipo *holding*.
4	Leitura de bloco de registradores do tipo *input*.

5	Escrita em um único *bit* do tipo *coil* (saída discreta).
6	Escrita em um único registrador do tipo *holding*.
7	Ler o conteúdo de 8 estados de exceção.
8	Prover uma série de testes para verificação da comunicação e erros internos.
11	Obter o contador de eventos.
12	Obter um relatório de eventos.
15	Escrita em bloco de *bits* do tipo *coil* (saída discreta).
16	Escrita em bloco de registradores do tipo *holding*.
17	Ler algumas informações do dispositivo.
20	Ler informações de um arquivo.
21	Escrever informações em um arquivo.
22	Modificar o conteúdo de registradores de espera através de operações lógicas.
23	Combina ler e escrever em registradores numa única transação.
24	Ler o conteúdo da fila FIFO (*First In, First Out*) de registradores.
43	Identificação do modelo do dispositivo.

Fonte: (FREITAS, 2014)

Quando os equipamentos são configurados para se comunicarem em uma rede Modbus usando ASCII (*American Standard Code for Information Interchange*), cada *byte* em uma mensagem é enviado como dois caracteres ASCII. Apesar de gerar mensagens legíveis pela tabela ASCII, esse modo consome mais recursos da rede. A principal vantagem dessa modalidade é que permite que os intervalos de tempo sejam cerca de um segundo para correr entre os caracteres sem causar erro.

Já no modo RTU, cada mensagem de 8 *bits* contém dois caracteres hexadecimais de 4 *bits*. A principal vantagem desse modo é que sua maior densidade de caracteres permite um

melhor processamento de dados do que o modo ASCII para a mesma velocidade de comunicação (*baudrate*), onde cada mensagem deve ser transmitida em um fluxo contínuo de caracteres. A composição da mensagem Modbus RTU é mostrada na Tabela 3.

Tabela 3- Composição da mensagem Modbus RTU

Endereço	**Função**	**Dados**	**Checagem CRC**
8 bits	8 bits	N x 8 bits	16 bits

Fonte: (FREITAS, 2014)

No modo RTU não existe um caractere específico que indique o início ou o fim de um telegrama. A indicação de quando uma nova mensagem começa ou quando ela termina é feita pela ausência de transmissão de dados na rede, por um tempo mínimo de 3,5 vezes o tempo de transmissão de um *byte* de dados. Sendo assim, caso um telegrama tenha iniciado após a decorrência desse tempo mínimo, os elementos da rede irão assumir que o primeiro caractere recebido representa o início de um novo telegrama.

Se durante a transmissão de um telegrama o tempo entre os *bytes* for maior que este tempo mínimo, o telegrama será considerado inválido, pois o controlador irá descartar os *bytes* já recebidos e montará um novo telegrama com os *bytes* que estiverem sendo transmitidos (FREITAS, 2014).

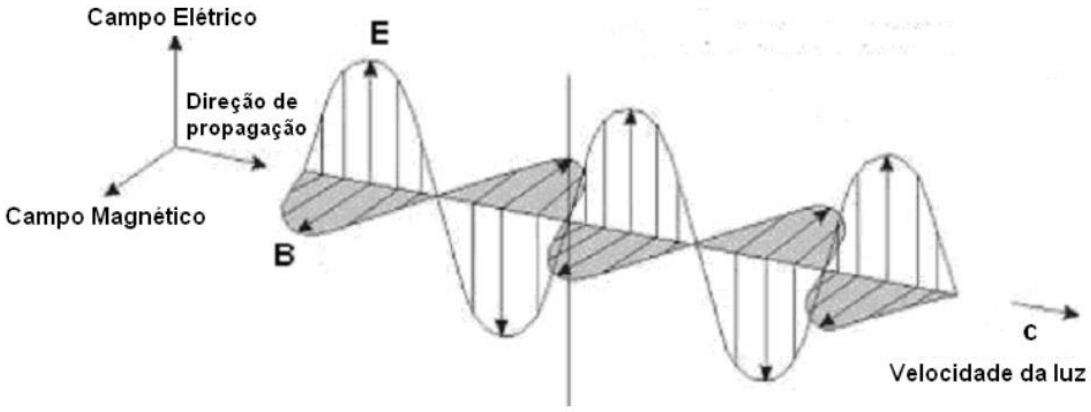
2.4.2 Radiofrequência

As ondas de radiofrequência são campos eletromagnéticos utilizados nas comunicações sem fio que levam energia de um ponto ao outro, permitindo a comunicação sem a necessidade de condutores elétricos. Sinais de radiofrequência são sinais que se propagam por um condutor cabeado e são irradiados no ar através de uma antena que converte um sinal do meio cabeado em um sinal *wireless* (sem fio) e vice-versa. Estes sinais irradiados no ar livre, em forma de ondas eletromagnéticas, propagam-se em linha reta e em todas as direções.

Uma onda é uma perturbação ou variação que transfere energia progressivamente de um ponto a outro em um meio e pode sofrer uma deformação elástica, de pressão, intensidade elétrica ou magnética, potência elétrica ou de temperatura. A radiofrequência utiliza ondas

eletromagnéticas (Figura 2.32) que não requerem meio material para se propagar, ou seja, podem se propagar no vácuo, ar, água e alguns sólidos.

Figura 2.32- Onda eletromagnética



Fonte: (VALLE, 2013).

O espectro eletromagnético é o intervalo completo da radiação eletromagnética (ondas de rádio, micro-ondas, infravermelho, luz visível, ultravioleta, raios x, até à radiação gama). No espectro eletromagnético, as diferentes bandas de radiofrequência são constituídas por diferentes intervalos de frequência (largura de banda).

Estes intervalos são representados na Figura 2.33.

Figura 2.33- Bandas de radiofrequência

NOME	SIGLA	BANDA DE FREQUÊNCIA	COMPRIMENTO DE ONDA	UTILIZAÇÕES
Frequências extra-baixas	ELF	30 Hz-3 kHz	10 000-1 000 000 m	Comunicação com submarinos.
Frequências muito baixas	VLF	3-30 kHz	100 000-10 000 m	Navegação e fins militares.
Frequências baixas	LF	30-300 kHz	10 000-1000 m	Navegação e estações de rádio internacionais.
Frequências médias	MF	300-3000 kHz	1000-100 m	Estações de rádio nacionais.
Frequências altas	HF	3-30 MHz	100-10 m	Estações de rádio e radiotelefone.
Frequências muito altas	VHF	30-300 MHz	10-1 m	Estações de rádio em FM e estações de televisão.
Frequências ultra-altas	UHF	300-3000 MHz	1-0,1 m	Estações de televisão, telemóveis e controlo aéreo por radar.
Frequências superaltas	SHF	3-30 GHz	0,1-0,01 m	Telemóveis, radar, satélites de comunicação e GPS.
Frequências extra-altas	EHF	30-300GHz	0,01-0,001 m	Estações espaciais.

Fonte: (VALLE, 2013).

As ondas de rádio de médias (MF) e baixas frequências (LF), possuem grandes comprimentos de onda, sendo por isso designadas por ondas médias e longas. Elas são as que melhor se difratam na atmosfera, contornando facilmente obstáculos de grandes dimensões, e acompanhando a curvatura terrestre até alguns milhares de quilômetros. À medida que a frequência aumenta, diminui a capacidade de transmitir para distâncias muito longas ao nível da superfície terrestre. As ondas de rádio de elevadas frequências, HF e VHF, possuem pequenos comprimentos de ondas, geralmente sofrendo múltiplas reflexões na ionosfera e na superfície terrestre, além de não conseguirem acompanhar a curvatura da Terra. Desta forma, elas são utilizadas em comunicações que não exigem grande alcance, mas, por serem geralmente moduladas em frequência (FM), são muito utilizadas quando é exigida a alta qualidade de som e imagem.

As ondas eletromagnéticas com frequências ultraelevadas (UHF), superaltas (SHF) e extra-altas (EHF), como as micro-ondas, são pouco absorvidas e/ou refletidas na atmosfera e praticamente não sofrem difração, propagando-se em linha reta. Uma vez que podem atravessar a ionosfera, são utilizadas nas comunicações via satélite (VALLE, 2013).

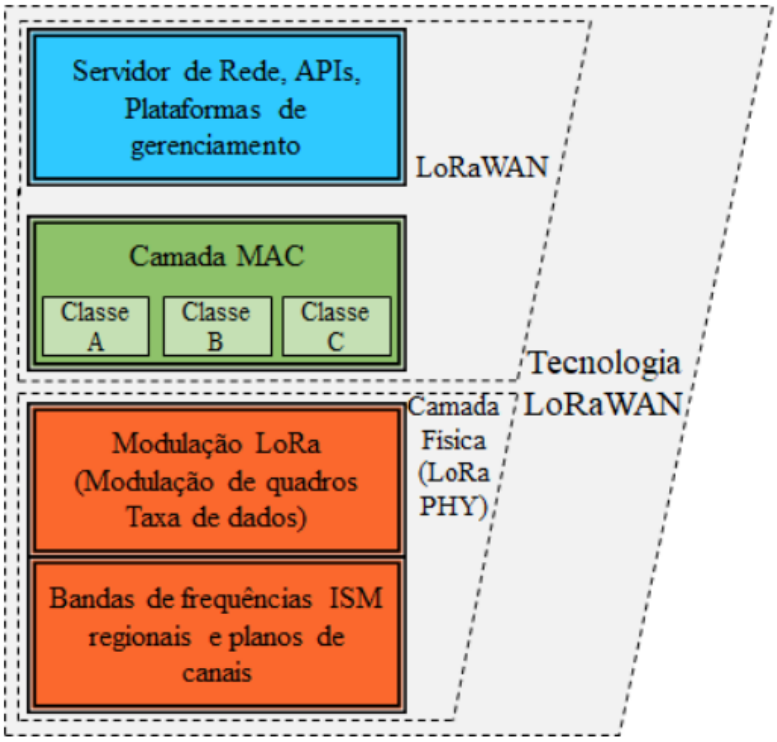
2.4.2.1 LoRa

LoRa (*Long Range*) é um sistema de comunicações sem fio promovido pela LoRa Alliance com uma especificação proprietária para redes de grande alcance e de baixa potência, focada em dispositivos presentes no cotidiano dos usuários e que seguem o paradigma da internet das coisas (*IoT)*. Enquadrado na categoria de redes LPWAN (*Low Power Wide Area Networks*), o LoRa procura atender aos requisitos de baixo consumo de energia, operando em frequências não licenciadas ISM, com o intuito de reaproveitar a operação em frequência em redes de telefonia legadas (GPRS, GSM, entre outras).

Baseado em um modelo de camadas de funções, mostrado na Figura 2.34 a tecnologia LoRaWAN implementa uma pilha de protocolos que define a comunicação entre os componentes da arquitetura da tecnologia em um único salto. Na prática, a tecnologia LoRa é simplesmente a estrutura física/eletrônica que permite a modulação. Para criar uma rede, é necessária uma camada lógica de controle de acesso ao meio (camada MAC, Servidor de Rede e Servidor de Aplicação), que recebe o nome de LoRaWAN. A associação da camada física LoRa com a camada lógica da rede LoRaWAN constituem a rede LoRaWAN. Desse modo,

dispositivos terminais conectam-se na rede, sendo estabelecidos os parâmetros de frequência e taxas de transmissão através da camada lógica da rede (ORTIZ, 2018).

Figura 2.34- Estrutura em camadas da tecnologia LoRaWAN



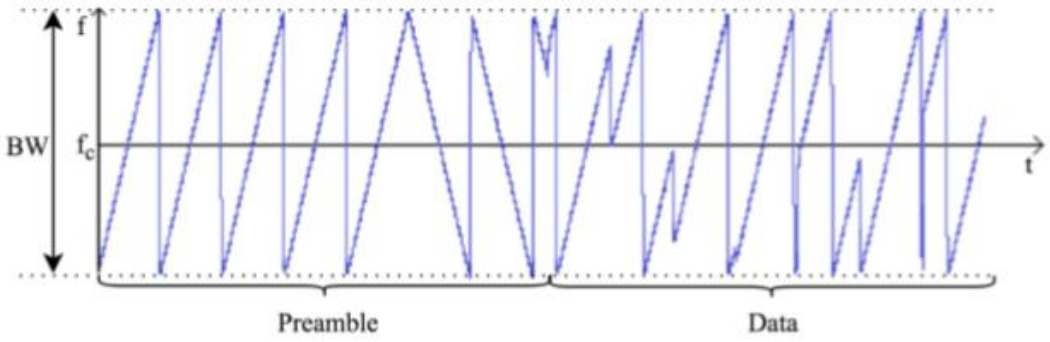
Fonte: (ORTIZ, 2018).

O LoRaWAN é o protocolo padrão aberto desenvolvido pela *LoRa Alliance* e define a arquitetura do sistema e parâmetros de comunicação e de acesso ao meio (MAC – *Medium Access Control*) que utiliza a camada física LoRa, permitindo que muitos dispositivos finais se comuniquem com um *gateway* usando a modulação LoRa. O LoRaWAN define taxas de transmissão de dados, o suporte à comunicação bidirecional e a oferta de serviços de mobilidade e localização dos nós da rede.

A camada física do LoRa (LoRa *PHY*) é uma tecnologia proprietária da empresa Semtech, baseada em uma técnica de modulação de espalhamento espectral proprietária, ou seja, uma variante do espalhamento espectral por *chirp* (*Chirp Spread Spectrum* – CSS). Nesta técnica, cada símbolo é enviado em um sinal de banda estreita propagada sobre uma banda de frequência mais ampla, com a mesma densidade de potência. A Figura 2.35 mostra o comportamento do sinal resultante da técnica de modulação usada na tecnologia LoRa. Além da Correção Adiantada de Erros (*Forward Error Correction* - FEC), e do CSS, o LoRa consegue incrementar a sensibilidade do receptor, aumentar a faixa de cobertura, prover alta robustez,

resistência ao efeito de múltiplas rotas e ao efeito Doppler, além de permitir o uso de baixa potência para a transmissão (ORTIZ, 2018).

Figura 2.35- Comportamento do sinal resultante da modulação CSS



Fonte: (AUGUSTIN et al., 2016).

LoRa é uma modulação do espectro de *Chirp*, que usa um sinal sinusoidal e saltos de frequência com uma variação linear de frequência ao longo do tempo para codificar informações.

Chirps são basicamente uma rampa de frequência mínima para máxima (*up-chirp*) ou de frequência máxima para mínima (*down-chirp*) em que cada símbolo é um *Chirp*. Para gerar símbolos / *Chirps*, o modem modula a fase de um oscilador. O número de vezes por segundo que o modem ajusta a fase é chamado de taxa de *chip* e define a largura de banda de modulação.

A modulação LoRa cria um sinal de *Chirp* para cada símbolo enviado, enquanto que cada *bit* é enviado através de vários *chips* de informação, como é chamado. Estes *chips* são variações de frequência que modulam a informação de cada *bit*, onde a quantidade de *chips* por símbolo é definida pelo fator de espalhamento (SF).

Devido à linearidade dos pulsos de *Chirp*, as frequências de *offsets* entre o receptor e o transmissor são facilmente eliminadas no decodificador. Isso também torna essa modulação imune ao efeito Doppler, equivalente a um deslocamento de frequência. O desvio de frequência entre o transmissor e o receptor pode atingir 20% da largura de banda sem afetar o desempenho da decodificação. (AUGUSTIN et al., 2016).

O LoRa substitui um bit por múltiplos *chips* de informação, fazendo com que o Fator de espalhamento (*Spreading Factor* - SF) tenha influencia na duração em tempo no ar de um pacote. Os múltiplos *chips* de informação precisam ser transmitidos tão rápido quanto a taxa de

bits original. Portanto, os dados são enviados com uma taxa de *chip* (*chips per second* - cps) igual à largura de banda (125 kHz = 125 kcps).

Para que os dispositivos terminais se comuniquem com os *gateways*, a tecnologia LoRa se baseia na adaptação da modulação CSS. O consumo de energia, a faixa de transmissão e a resistência `a interferência do ruído podem ser determinados a partir de quatro parâmetros de configuração da camada física do LoRa: a frequência da portadora, que define a frequência central para a banda de transmissão; a largura de banda, que define o tamanho da faixa de frequências utilizada; a taxa de codificação, que define a taxa de FEC; e o fator de espalhamento, que define o espalhamento espectral (ORTIZ, 2018).

Outro parâmetro da modulação LoRa que é implementado nos transceptores da Semtech é a otimização da baixa taxa de dados. Este parâmetro é obrigatório no LoRa ao usar fatores de espalhamento de 11 e 12 com uma largura de banda de 125 kHz ou inferior. O efeito deste parâmetro não está documentado, porém, sabe-se que reduz o número de *bits* transmitidos por símbolo em dois.

2.4.2.1.1 Frequência de portadora

A frequência portadora (*Carrier Frequency* - CF) é a frequência central usada para a banda de transmissão e é definida de acordo a região de operação dos equipamentos.

Os sistemas de salto em radiofrequência do Brasil abrangem as faixas de 902-907,5 MHz e 915-928 MHz, regidos segundo a Resolução nº680 de 27 de junho de 2017 e o Ato nº14.448 de 04 de dezembro de 2017 (Seção 10.2) publicados pela ANATEL (Agencia Nacional de Telecomunicações).

2.4.2.1.2 Largura de banda

A largura de banda (*Bandwidth* - BW) é o intervalo de frequências na faixa de transmissão. Se a largura de banda for baixa, a sensibilidade será alta, mas a taxa de transmissão será baixa; quer dizer que a potência do sinal que o receptor vai detectar pode estar por baixo do nível de ruído, requerendo um maior ganho de processamento na recepção para a decodificação do sinal. Por outro lado, uma largura de banda alta leva a uma alta taxa de

transferência, mas com maior tempo de transmissão e menor sensibilidade. Com a largura de banda maior, mais sinais ortogonais serão usados, resultando em uma capacidade da rede mais alta, mas a potência do sinal terá uma sensibilidade mais baixa por conta da integração adicional de ruído. Para o chip SX1276 são definidas três possíveis configurações de largura de banda: 500 kHz, 250 kHz e 125 kHz.

2.4.2.1.3 Taxa de codificação

A taxa de codificação TC é relacionada com a técnica de FEC (código de correção contra ruídos que permite adicionar margens de até 3 dB´s na relação sinal / ruído) usada pelo modem LoRa e oferece proteção contra rajadas de interferência. O CR (*Coding Rate*) define quantos *bits* são utilizados para dados de redundância na mensagem, a fim de realizar a recuperação de erros. O CR da carga útil é armazenado no cabeçalho do pacote onde são definidos quatro valores de TC para serem implementados: 4/5, 4/6, 4/7 e 4/8, sendo que um CR maior oferece maior proteção, porém, aumenta o tempo no ar (ORTIZ, 2018).

A CR define a taxa de codificação TC segundo a Equação (2).

$$TC = \frac{4}{4+CR}, com\ CR \in \{1, 2, 3, 4\} \quad (2)$$

2.4.2.1.4 Fator de espalhamento

O fator de espalhamento (*Spreading Factor* – SF) é a razão entre a taxa de símbolos e a taxa de *chips*, onde grandes sequências de *bits* são codificadas em um único símbolo, reduzindo assim a relação sinal ruído (*Signal-to-Noise Ratio* - SNR) e a interferência de outras frequências nas transmissões de dados. Um fator de espalhamento mais alto aumenta a Relação Sinal / Ruído (SNR) e, portanto, sensibilidade e alcance, mas também aumenta o tempo no ar do pacote.

A taxa de modulação é definida pelo SF onde o número de *chips* por símbolo é calculado como 2^{SF}. Por exemplo, com um SF igual a 12, são utilizados 4096 *chips* por símbolo.

O fator de espalhamento pode ser selecionado de 6 a 12, sendo que cada aumento em SF reduz a taxa de transmissão e, portanto, dobra a duração da transmissão. Com a maior taxa de transmissão, um SF igual a 6 torna-se um caso especial e requer operações especiais onde cabeçalhos implícitos são obrigatórios.

Sendo assim, é possível calcular uma taxa de transmissão (*bitrate* - *Br*) teórica, definida em função do fator de espalhamento, segundo a Equação (3):

$$Br = SF * \frac{TC*BW}{2^{SF}} \quad (3)$$

O tempo requerido para enviar um símbolo LoRa está influenciado pelo tipo de modulação e pelo fator de espalhamento, em função da largura de banda do canal, segundo a Equação (4):

$$Ts = \frac{2^{SF}}{BW} \quad (4)$$

2.4.2.1.5 Formato da carga útil

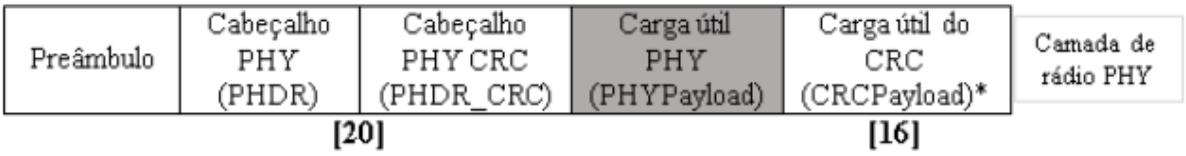
Embora a modulação LoRa possa ser usada para transmitir quadros arbitrários, um formato de quadro físico é especificado e implementado nos transmissores e receptores da Semtech. A largura de banda e o fator de espalhamento são constantes para um quadro, sendo que um quadro LoRa começa com um preâmbulo, e este preâmbulo começa com uma sequência de saltos constantes que cobrem toda a banda de frequência. Os últimos dois saltos codificam a palavra de sincronização, onde a palavra de sincronização é um *byte* de avaliação que mostra como diferenciar as redes que usam as mesmas bandas de frequência.

Após o preâmbulo, existe um cabeçalho opcional. Quando está presente, este cabeçalho é transmitido com uma taxa de código de 4/8. Isso indica o tamanho da carga útil (em *bytes*), a taxa de código usada para o final da transmissão e se o método de detecção de erros CRC de 16 *bits* para a carga útil está ou não presente no final do quadro. O cabeçalho também inclui um CRC para permitir que o receptor descarte pacotes com cabeçalhos inválidos. O tamanho da carga útil é armazenado usando um *byte*, limitando o tamanho da carga útil a 256 *bytes*. O cabeçalho é opcional para permitir sua desativação em situações em que não é necessário, por

exemplo, quando o comprimento da carga útil, a taxa de codificação e a presença de CRC são conhecidos antecipadamente. A carga útil é enviada após o cabeçalho e, no final do quadro, há o CRC opcional (AUGUSTIN et al., 2016).

A Figura 2.36 a seguir mostra o formato do quadro de mensagem da camada física do LoRa. O tamanho dos campos está indicado entre colchetes na parte inferior das camadas, em byes.

Figura 2.36- Formato das mensagens do LoRa



Fonte: (ORTIZ, 2018).

O preâmbulo indica o esquema de modulação dos pacotes, estabelecendo um mesmo fator de espalhamento para cada pacote enviado. Os cabeçalhos PHY e PHY CRC indicam a taxa de código para executar a FEC a ser implementada na mensagem. A Carga útil PHY será então a concatenação das informações contidas na camada MAC e no cabeçalho do quadro, completando assim o formato da carga útil na camada física PHY (ORTIZ, 2018).

A Equação (5) fornece o número de símbolos (*Ns*) requeridos para transmitir uma carga útil em função de todos esses parâmetros. Este número deve ser adicionado ao número de símbolos do preâmbulo, para calcular o tamanho total do pacote em símbolos (AUGUSTIN et al., 2016).

$$Ns = 8 + \left(\left[\frac{8\,x\,PL - 4\,x\,SF + 8 + CRC + H}{4\,x\,(SF - DE)} \right] x\,(CR + 4) \right) \quad (5)$$

Esta equação também mostra que o tamanho mínimo de um pacote é oito símbolos e possui as seguintes dependências:

- PL é o tamanho da carga útil em *bytes*;
- CRC é 16 se o CRC estiver ativado e zero caso contrário;
- H é 20 quando o cabeçalho é ativado e zero caso contrário;
- DE é 2 quando a otimização da taxa de dados baixa está habilitada e zero caso contrário.

2.4.2.2 Bluetooth

Bluetooth é uma tecnologia sem fio usada para conectar e transmitir dados entre dispositivos em redes pessoais (*Personal Area Networks* - PANs) e surgiu como resposta para a necessidade de conectar dispositivos sem a utilização de cabos, visando a economia de energia, a fácil operação e a comodidade. Com velocidades de transmissão que podem se aproximar de 24 Mbps, o bluetooth é o padrão em comunicação de curta distância.

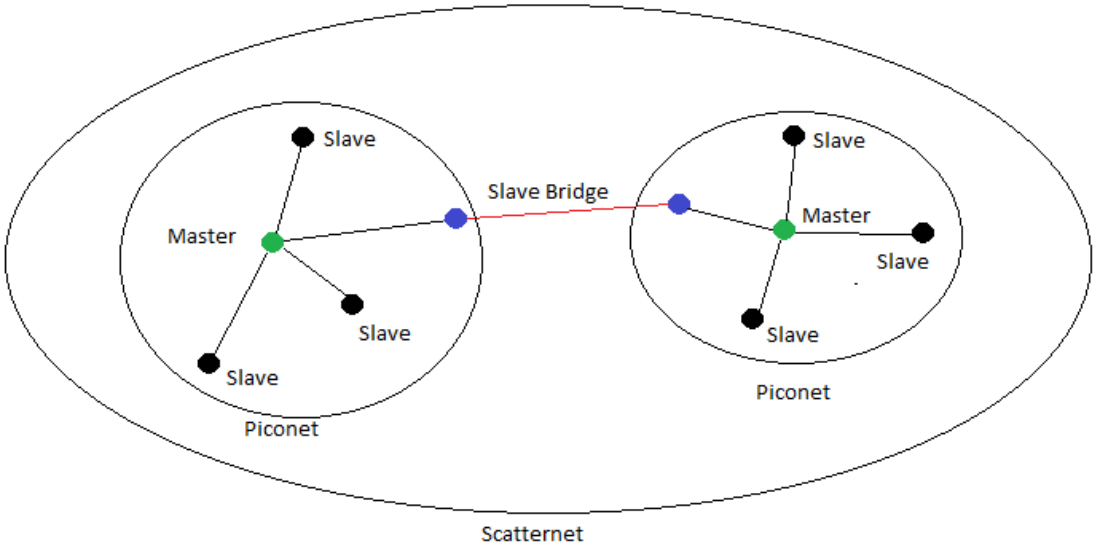
O bluetooth utiliza ondas de rádio de baixa potência, operando em frequências que vão de 2.4GHz a 2.5 GHz, na faixa de frequência ISM (*Industrial, Scientific, Medical*). O uso da baixa potência aumenta a economia de energia das baterias e limita o alcance a um máximo de 100m.

A tecnologia é dividida em três classes levando em conta o alcance das ondas de rádio, são elas:

- Classe 3 de 1 mW de potência: Alcança distâncias de até 1m;
- Classe 2 de 10 mW potência: Alcança distâncias de até 10m;
- Classe 1 de 100 mW potência: Alcança distâncias de até 100m.

O conceito de *Bluetooth Wireless Personal Area Network* (BT-WPAN) é o nome dado a área onde dispositivos bluetooth formam uma rede baseada em *Piconets* e *Scatternets*. Uma *piconet* consiste de um conjunto de até oito dispositivos conectados, onde o dispositivo que iniciou a conexão é marcado como mestre e os demais como escravos. Duas *piconets* podem se conectar através de dispositivos comuns em ambas as redes, sendo a única restrição que estes dispositivos não sejam os mestres de suas *piconets*. Esta união de *piconets* recebe o nome de *Scatternet*, conforme exemplificado na Figura 2.37 a seguir (DEVMEDIA, 2013).

Figura 2.37- Bluetooth Wireless Personal Area Network (BT-WPAN)

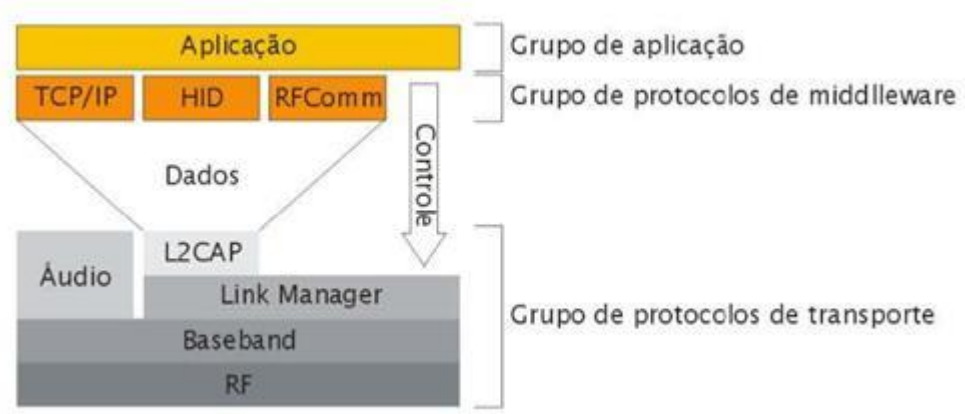


Fonte: (DEVMEDIA, 2013).

As transmissões ocorrem no modo *Full Duplex*, onde os dispositivos transmitem e recebem dados por um esquema de divisão de tempo chamado TDD (*Time Division Duplex*). Além disso, a pilha de protocolos Bluetooth é dividida em três camadas: camada de transporte; camada *middleware* e camada de aplicação.

Os protocolos de transporte são responsáveis por localizar os dispositivos e gerenciar os *links* físicos e lógicos entre eles. Suportam tanto conexões síncronas quanto assíncronas e englobam as camadas de rádio frequência (RF), *Baseband*, *Link Manager* e *Logical Link Control and Adaptation* (L2CAP). Os protocolos de *Middleware* são responsáveis por permitir a interação entre aplicações antigas e novas. Padrões como *Point-toPoint Protocol* (PPP), *Wireless Application Protocol* (WAP), *Internet Protocol* (IP) e *Trasmission Control Protocol* (TCP) fazem parte desta camada. A camada de aplicação faz referência aos aplicativos que podem usufruir da especificação bluetooth. A Figura 2.38 representa as camadas de protocolos bluetooth (DEVMEDIA, 2013).

Figura 2.38- Camadas de protocolos bluetooth



Fonte: (DEVMEDIA, 2013).

As siglas representadas na Figura 2.38 são explicadas a seguir:

RF ou RFCOM: Protocolo que habilita a comunicação serial através da emulação de portas na camada L2CAP;

Baseband: Controlador de Link que define conexão, endereçamento, formato dos pacotes, temporização e potência;

Link Manager: Define aspectos de segurança e gerencia os links entre os dispositivos;

Áudio: Para manter a alta qualidade de serviço para aplicações de áudio, o tráfego do mesmo é tratado como alta prioridade e é direcionado diretamente para a *baseband*, o qual é transmitido em pequenos pacotes diretamente para a interface RF;

L2CAP: Responsável pela transparência de comunicação dos diferentes dispositivos;

TCP/IP: Protocolo utilizado pela internet para a transmissão confiável de dados;

HID: *Human Interface Device Profile* é um perfil que define protocolos, procedimentos e características para utilizar dispositivos Bluetooth como teclados, controle de jogos ou dispositivos para monitoramento remoto;

RFCOM: Protocolo que habilita a comunicação serial através da emulação de portas na camada L2CAP;

Aplicação: A camada de aplicação consiste das próprias aplicações que utilizam links Bluetooth, incluindo aplicações legadas ou orientadas a este.

3 MATERIAL E MÉTODOS

Neste capítulo serão apresentados de forma específica todos os passos seguidos para a execução de projeto, desenvolvimento e manufatura do sistema de telemetria e do aplicativo para volante.

3.1 PROJETO

Inicialmente é feito um levantamento de conceitos e requisitos para o novo projeto através da análise do sistema anterior, das lições aprendidas que foram registradas e de um *benchmarking* de outras equipes. A partir destas análises são definidas especificações básicas como: comunicações confiáveis entre o protótipo e as interfaces de monitoramento; alcance mínimo de 350m para o sistema de telemetria principal; interface nítida e confiável do *display* para o volante; e possuir os seguintes requisitos técnicos básicos: Coletar dados de múltiplos sensores do protótipo e comunicar a um computador em tempo real; e fornecer ao piloto informações básicas para pilotagem.

Considerando que o novo sistema de telemetria tem como principal objetivo aumentar o alcance de transmissão, o primeiro passo foi pesquisar, entre as equipes de Formula SAE, quais eram as tecnologias de transmissão de dados mais utilizadas, e que cobriam o alcance mínimo requisitado. Dentre as tecnologias encontradas, pode-se citar a comunicação ZigBee como a mais utilizada entre as equipes.

Cogitando a hipótese de utilizar a comunicação ZigBee para o novo sistema, foi realizado testes de campo com módulos da linha XBee, chegando a conclusão de que esta tecnologia não atenderia as exigências do projeto devido ao baixo alcance da transmissão de sinal, e a complexidade de manipulação dos dados recebidos. Assim, pesquisando por novas tecnologias de comunicação, decidiu-se por testar e utilizar uma nova tecnologia de comunicação por rádio frequência chamada LoRa, que promete ser um sistema de comunicação confiável, de longo alcance e com uma configuração e manipulação mais simples e custo mais baixo com relação aos dispositivos de comunicação ZigBee.

Feita a análise inicial dos sistemas, são definidos os materiais e os métodos a serem utilizados para o desenvolvimento do novo sistema.

3.2 HARDWARE

Nesta seção são abordados os módulos comerciais utilizados, bem como a metodologia de desenvolvimento das placas e circuitos para o funcionamento do sistema de telemetria.

3.2.1 Módulos comerciais

A fim de facilitar e possibilitar a comunicação entre alguns dispositivos, optou-se pela utilização de alguns módulos comerciais com circuitos e protocolos de comunicação pré-programados, demonstrados a seguir.

3.2.1.1 Módulo LoRa

A alternativa comercial encontrada para a utilização da tecnologia LoRa foi o módulo RF LoRa1276 (Figura 3.1). Este é um módulo transceptor de rádio frequência (RF) com tecnologia LoRa (*Long Range*) baseado no *chip* SX1276 da empresa Semtech, que oferece uma mistura de comunicação de longo alcance, baixo consumo de energia e transmissão segura de dados, visando atender uma enorme variedade de projetos voltados a Internet das Coisas (IoT – *Internet of Things*) e M2M (*Machine to Machine*).

O *chip* SX1276 pode atingir uma sensibilidade de -139dBm e potência de transmissão de +20dBm, além de oferecer vantagens significativas tanto no bloqueio quanto na seletividade em relação a técnicas convencionais de modulação, resolvendo o comprometimento entre o longo alcance, imunidade às interferências e o consumo de energia, garantindo uma excelente robustez na comunicação.

Dessa forma o módulo RF LoRa1276, com apenas 100mW de potência e frequência de 915MHz, torna possível comunicações sem fio de longo alcance, podendo chegar à 4km em ambiente suburbano e 2km em ambiente urbano, sendo comunicação ponto a ponto ou multipontos (utilizando um *gateway*) (CURTO CIRCUITO, 2018).

Figura 3.1- Módulo RF LoRa1276

Fonte: (CURTO CIRCUITO, 2018).

O módulo LoRa é alimentado com uma Tensão de 1,8 a 3,7V e possui um consumo de corrente de 10,8mA para recepção de sinal e 120mA para transmissão de sinal. Também possui modos de modulação LoRa TM, FSK, GFSK e OOK e uma taxa de transferência de dados de 0,018 a 37,5 Kbps para a modulação do tipo LoRa TM que foi utilizada neste trabalho.

A conexão com o Arduino é realizada utilizando a interface de comunicação SPI e uma biblioteca específica desenvolvida para esta tecnologia.

3.2.1.2 Módulo CAN

Para receber os dados da central eletrônica (ECU) através do protocolo CAN, utilizou-se o Módulo CAN BUS MCP2515 (Figura 3.2) desenvolvido especialmente para integrar projetos eletrônicos e microcontroladores Arduino a uma rede CAN. Este módulo conta com um controlador CAN MCP2515 e um transceptor TJA1050 integrado à placa.

Figura 3.2- Módulo CAN BUS MCP2515

Fonte: filipeflop.com.

Este módulo suporta a especificação CAN V2.0B com uma velocidade de 1 Mbps, chamada de *High Speed*, possui uma tensão de alimentação de 5V e corrente de operação de 5mA, além de resistores de terminação de 120 ohms.

A conexão com o Arduino é realizada utilizando a interface de comunicação SPI e uma biblioteca desenvolvida para ser utilizada com este modelo. Já a comunicação com a rede CAN é feita através de uma derivação do barramento até o módulo.

3.2.1.3 Módulo Bluetooth

Para fazer a comunicação Bluetooth com o smartphone, foi utilizado o módulo Bluetooth HC-05 (Figura 3.3), que difere do modelo HC-06 por suportar o modo escravo (*slave*) e mestre (*master*).

Figura 3.3- Módulo Bluetooth HC-05

Fonte: eletrogate.com.

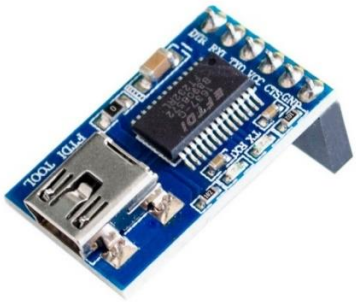
Este módulo possui uma tensão de funcionamento de 3,3 a 5V, taxa de transmissão de 2Mbps e opera na frequência de 2,4 GHz. A conexão com o Arduino é feita através da comunicação serial pelas portas RX e TX criadas a partir de um código multiserial.

3.2.1.4 Modulo FTDI

O módulo FTDI (Figura 3.4) é baseado no CI FT232RL e funciona como um conversor de USB para Serial, que permite comunicação de dispositivos TTL (*Transistor-Transitor Logic*)

com computadores através da porta USB. Possui tensão de entrada e saída de 5V e conector mini USB, além de LEDs indicadores para RX e TX.

Figura 3.4- Módulo FTDI com conector USB mini



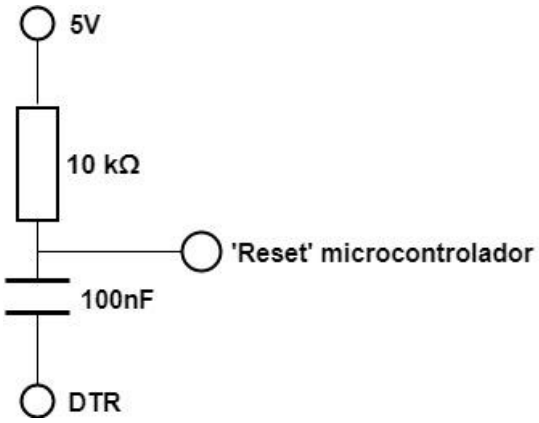
Fonte: pt.aliexpress.com.

A pinagem do módulo FTDI utilizado está listada abaixo:

- DTR (*Data Terminal Ready control output*)
- RX (*Receive Asynchronous data input*)
- TX (*Transmit Asynchronous data input*)
- CTS (*Clear to Send control input*)
- VCC
- GND

A utilização do módulo FTDI possibilita a programação do microcontrolador sem que seja necessário retira-lo da placa. Para isso, é necessário “resetar” o microcontrolador para carregar o programa no mesmo. Assim, deve-se implementar o circuito da Figura 3.5 para que quando o código seja gravado, o pino DTR envie 0V para o terminal do capacitor, fechando o circuito no pino ‘reset’ do microcontrolador ATmega328P, resetando o mesmo (BAÚ DA ELETRÔNICA, 2018).

Figura 3.5- Circuito para "reset" do microcontrolador



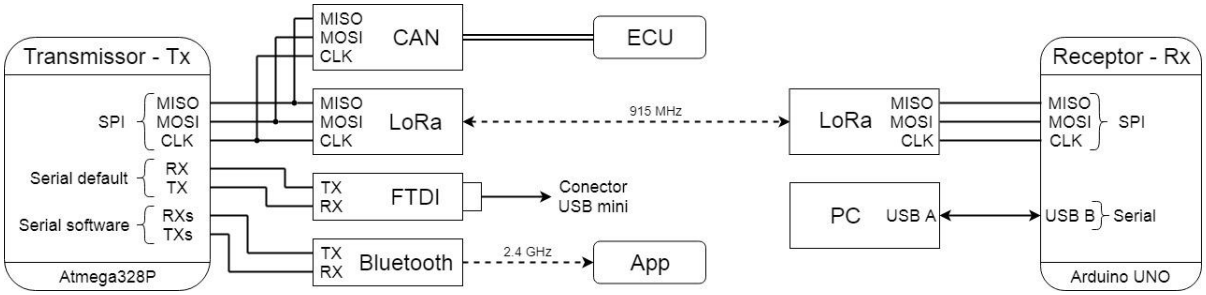
Fonte: Próprio autor.

3.2.2 Placas

O desenvolvimento das placas customizadas inicia-se com o desenho dos diagramas elétricos no *software* Proteus Professional, seguido de simulações e testes destes diagramas em bancada. Após a verificação do correto funcionamento dos circuitos elétricos e da definição dos componentes a serem utilizados, é realizado o desenho dos *layouts* finais das placas no mesmo *software*. Em seguida, os arquivos *Gerber* são gerados e enviados para uma empresa de prototipagem de placas, parceira da equipe.

Um diagrama de comunicação entre os dispositivos do sistema pode ser visto na Figura 3.6 a seguir.

Figura 3.6- Diagrama de comunicação entre os dispositivos

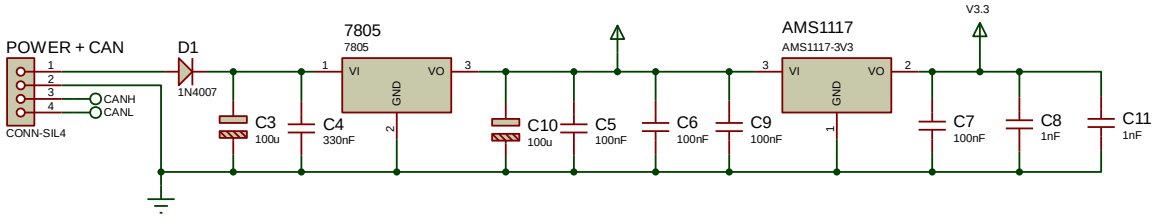


Fonte: Próprio autor.

3.2.2.1 Diagramas elétricos

O circuito elétrico da placa do emissor conta com capacitores de filtro e reguladores de tensão (LM7805 e AMS1117 3,3) para abaixar a tensão da bateria (12 volts), uma vez que o microcontrolador deve ser alimentado com 5 volts e o módulo LoRa com 3,3 volts.

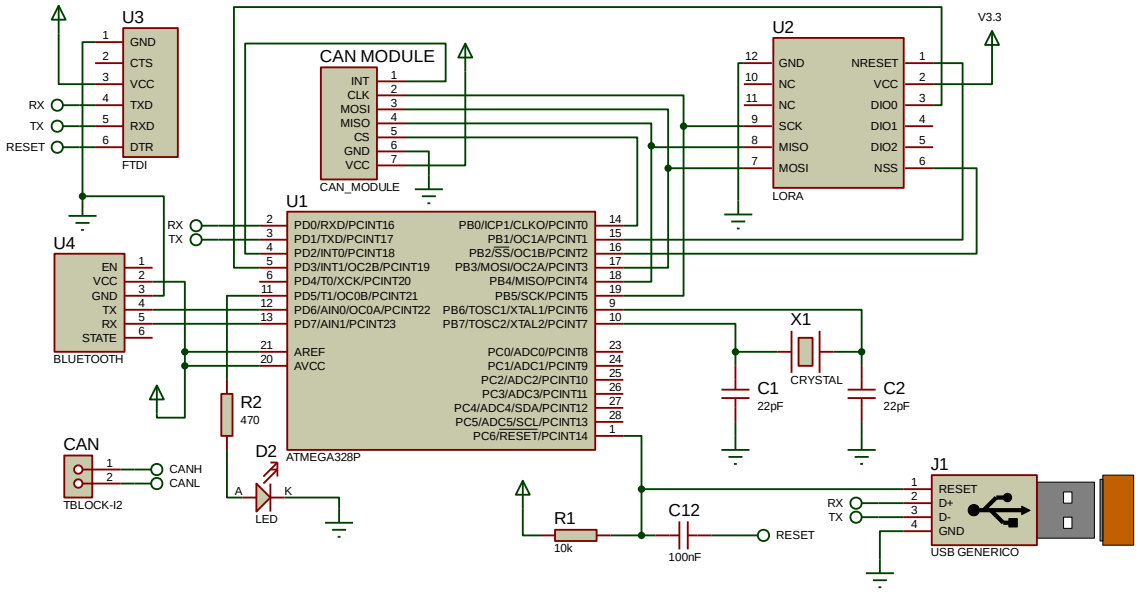
Figura 3.7- Circuito elétrico de alimentação do transmissor



Fonte: Próprio autor.

Segundo a folha de dados, o regulador de tensão AMS1117 3,3 exige uma tensão de entrada fixa de 5 volts. Para que este não receba uma tensão de 12 volts diretamente da bateria, os dois reguladores devem ser conectados em série.

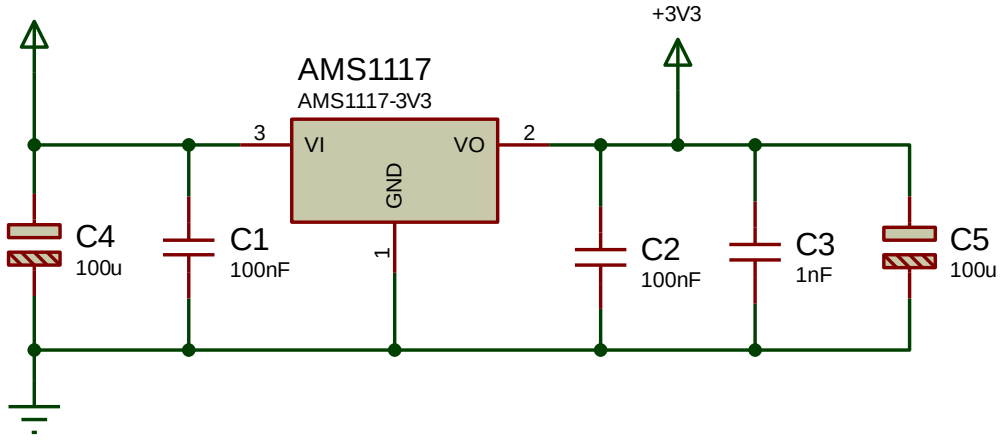
Figura 3.8- Circuito elétrico de operação do transmissor



Fonte: Próprio autor.

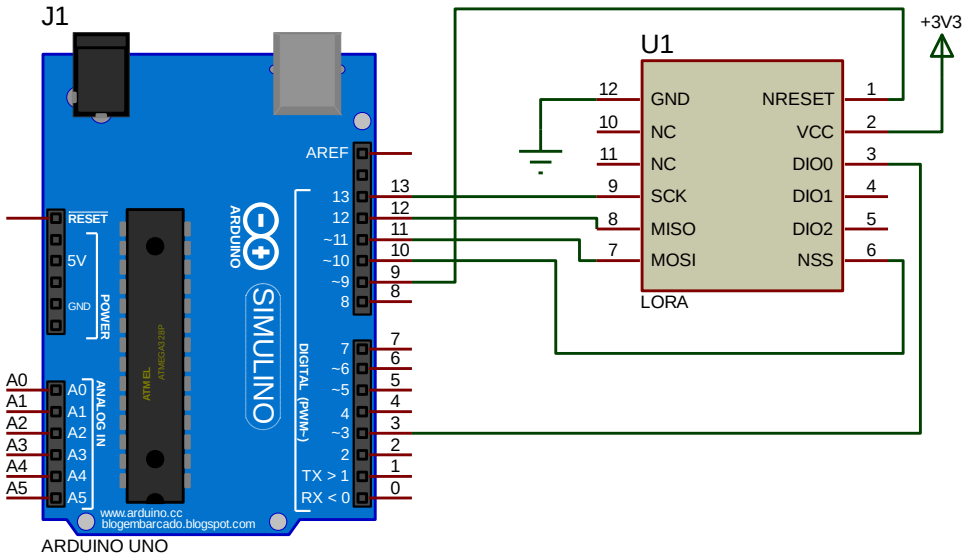
Já o circuito elétrico do *shield* receptor conta com um regulador de tensão para abaixar a tensão da fonte USB do Arduino (5 volts) para 3.3 volts, pois a porta de 3.3 volts do Arduino não fornece a corrente suficiente para alimentar o módulo LoRa.

Figura 3.9- Circuito elétrico de alimentação do receptor



Fonte: Próprio autor.

Figura 3.10- Circuito elétrico de operação do receptor



Fonte: Próprio autor.

3.2.2.2 Dispositivos e componentes eletrônicos

Optou-se pela utilização de resistores, capacitores e um regulador SMD a fim de reduzir o tamanho total da placa e facilitar o roteamento das trilhas. A lista de componentes com informações básicas e comerciais encontra-se na Tabela 4, sendo que alguns produtos e serviços tiveram custo nulo devido ao patrocínio de empresas privadas que apoiam a equipe formula UFSM.

Tabela 4- Lista de dispositivos e componentes eletrônicos

Produto	**Tipo**	**Quantidade**	**Custo**	**Fornecedor**
Antena 915Mhz	SMA	2	R$ 14,08	AliExpress
Arduino UNO	Placa	1	0	Eletrodex
Atmega328P	DIP	1	0	LCSC
Borne 2 pinos	DIP	1	0	LCSC
Caixa receptor	PB-204	1	R$ 5,07	Patolas
Caixa transmissor	Impressão 3D	1	0	Particular
Capacitor 100nF	SMD	7	0	LCSC
Capacitor 100uF	DIP	4	0	LCSC
Capacitor 1nF	SMD	3	0	LCSC
Capacitor 22pF	SMD	2	0	LCSC
Capacitor 330nF	SMD	1	0	LCSC
Conector 4 vias	MIKE	1	R$ 8,00	Eletrodex
Conector USB fêmea	DIP	1	0	LCSC
Cristal 16MHz	DIP	1	0	LCSC
Diodo 1N4007	SMD	1	0	LCSC
FTDI Basic	DIP	1	R$ 8,80	AliExpress
Header fêmea 1,27mm	DIP 1/2'	1	0	LCSC
Header macho 1,27mm	DIP 1/2'	1	0	LCSC
Header fêmea 7 pinos	DIP	1	0	LCSC
Header fêmea 6 pinos	DIP	1	0	LCSC
Header fêmea 4 pinos	DIP	1	0	LCSC
Header macho 6 pinos	DIP	1	0	LCSC
Header pino longo 10 pinos	DIP	1	R$ 1,95	Eletrodex
Header pino longo 6 pinos	DIP	1	R$ 1,85	Eletrodex
Header pino longo 8 pinos	DIP	2	R$ 1,45	Eletrodex

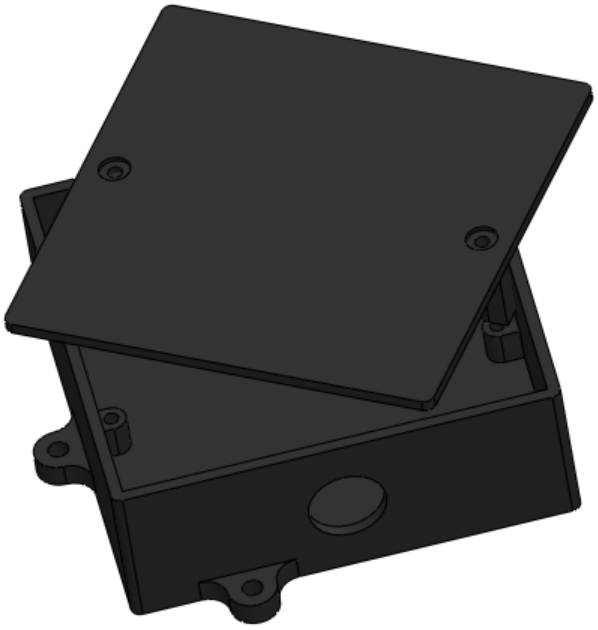
Led vermelho 3mm	DIP	1	0	LCSC
Modulo Bluetooth HC-05	DIP	1	R$ 39,00	Webtrônico
Módulo CAN MCP2515	DIP	1	R$ 5,46	AliExpress
Modulo LoRa	DIP 1/2'	2	R$ 50,56	Curto Circuito
Regulador de tensão 7805	DIP	1	0	LCSC
Regulador de tensão AMS1117 3.3V	SMD	2	0	LCSC
Resistor 470Ω	SMD	1	0	LCSC
Resistor 8,2KΩ	SMD	1	0	LCSC
Soquete estampado 28 pinos	DIP	1	0	LCSC

Fonte: Próprio autor.

3.2.2.3 Layout

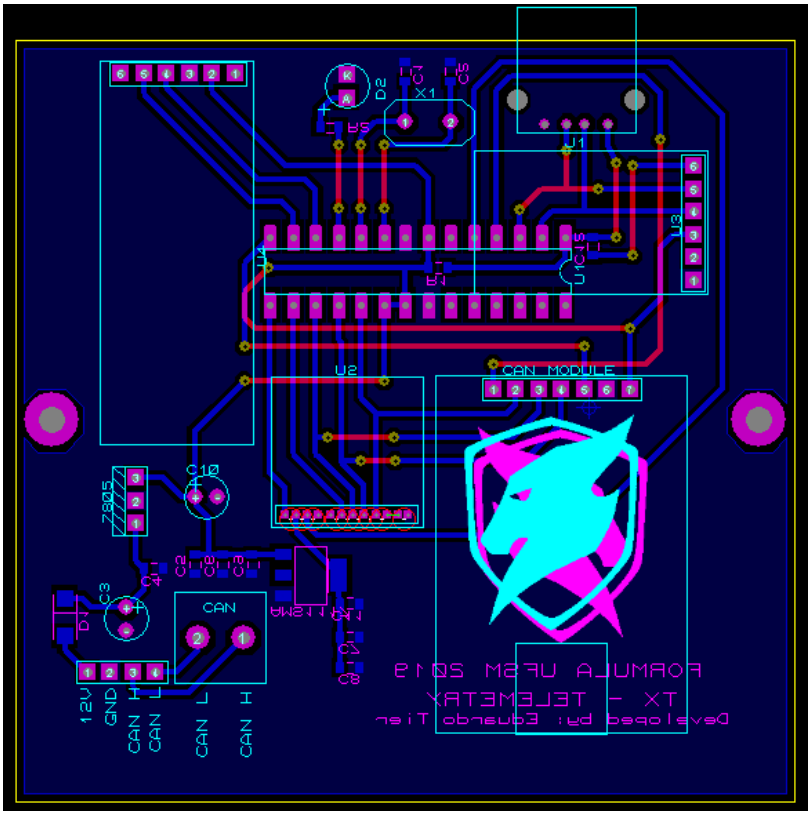
O *layout* das placas (Figura 3.12 e Figura 3.13) e a posição dos componentes são definidos a partir de um espaço limite que deve ser respeitado. No caso da placa do transmissor, o espaço é definido pelo formato de uma caixa desenhada em CAD (Figura 3.11) e fabricada a partir de impressão 3D, projetada especificamente para ser instalada na parte traseira do painel do veículo (85x87mm), enquanto que a placa do receptor (*shield*) deve respeitar o formato e as dimensões de um Arduino UNO R3.

Figura 3.11- CAD da caixa do módulo transmissor



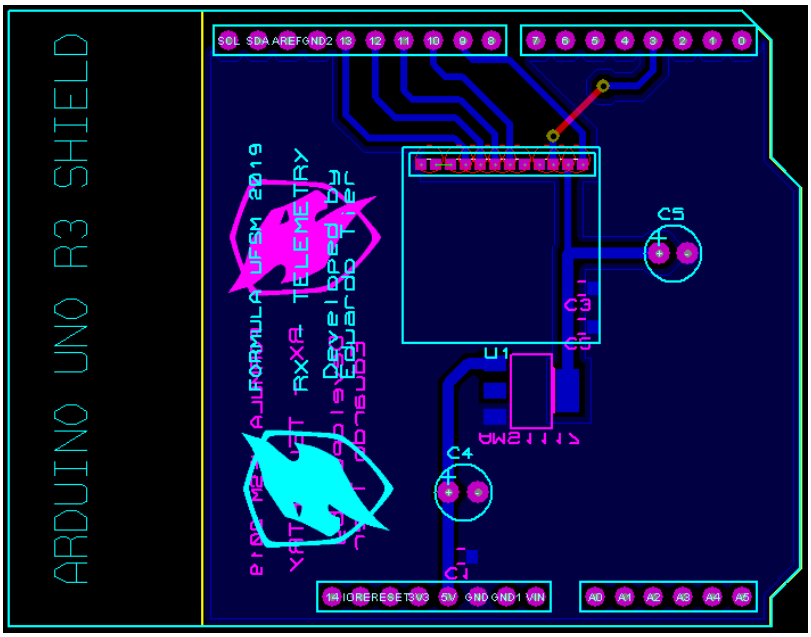
Fonte: Próprio autor.

Figura 3.12- Layout da placa do transmissor



Fonte: Próprio autor.

Figura 3.13- Layout da placa do receptor



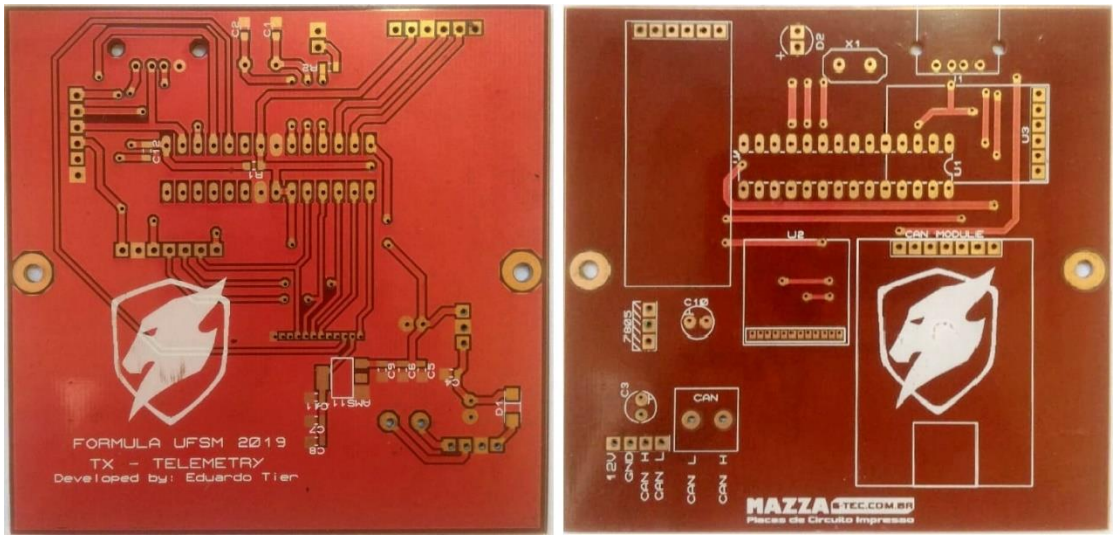
Fonte: Próprio autor.

3.2.2.4 Manufatura

A prototipagem das placas foi terceirizada, feita pela empresa Mazza G-Tec, patrocinadora da equipe Formula UFSM.

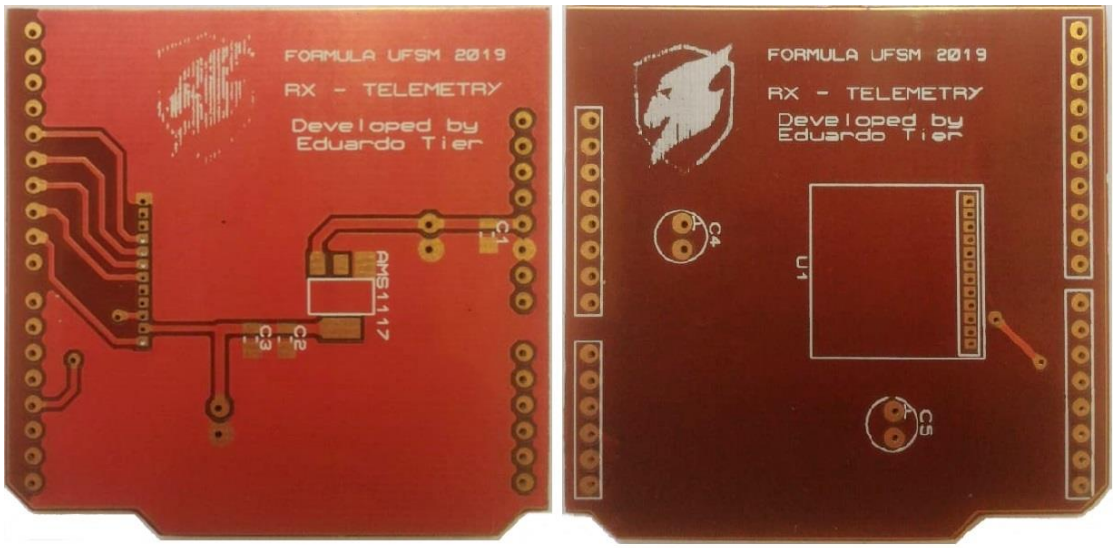
As placas de circuito impresso prototipadas podem ser vistas na Figura 3.14 e Figura 3.15.

Figura 3.14- Placa de circuito impresso do transmissor



Fonte: Próprio autor.

Figura 3.15- Placa de circuito impresso do receptor

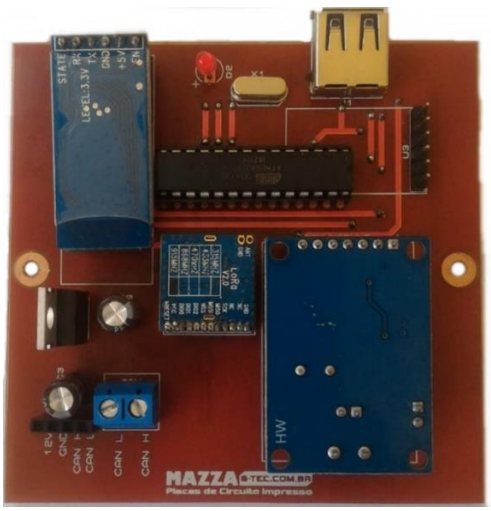


Fonte: Próprio autor.

Após a prototipagem, é realizada a soldagem dos componentes eletrônicos e dos conectores para os dispositivos, utilizando um ferro de solda com regulagem de temperatura ajustado para aproximadamente 220°C (um pouco abaixo da temperatura máxima para componentes SMD, segundo BRAGA, 2018d), fluxo de solda e estanho com proporção de 63% de estanho e 37% de chumbo e diâmetro de 0,5mm, a fim de proporcionar uma solda mais limpa e um processo de soldagem mais prático e eficiente.

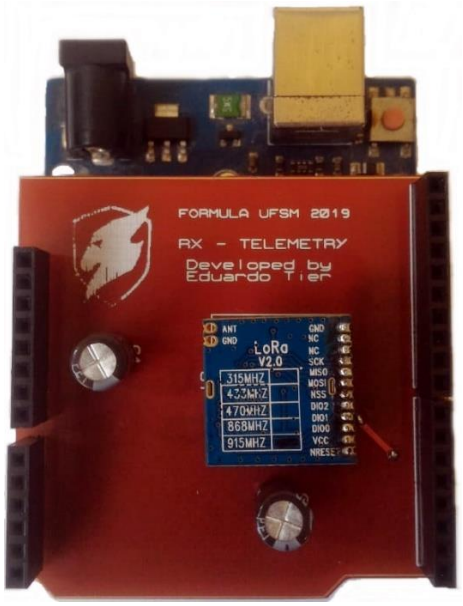
As placas com os componentes soldados podem ser vistas na Figura 3.16 e Figura 3.17.

Figura 3.16- Componentes eletrônicos soldados a placa do transmissor



Fonte: Próprio autor.

Figura 3.17- Componentes eletrônicos soldados a placa do receptor



Fonte: Próprio autor.

3.3 SOFTWARE

Nesta seção são abordados os *softwares* utilizados para o desenvolvimento do sistema de telemetria e aplicativo.

3.3.1 IDE Arduino

O IDE (*Integrated Development Environment*) do Arduino é um aplicativo de plataforma cruzada (para Windows, macOS, Linux) que é escrito na linguagem de programação Java e é usado para escrever e fazer *upload* de programas para a placa Arduino e, consequentemente, para o microcotrolador Atmega328p utilizado neste sistema.

O Arduino IDE suporta as linguagens C e C ++ usando regras especiais de estruturação de código, além de fornecer uma biblioteca de *software* do projeto *Wiring* e possuir várias funções específicas. O código escrito pelo usuário requer apenas duas funções básicas: para iniciar o rascunho (*setup*) e o *loop* principal do programa (*loop*), que são compilados e vinculados a um programa principal (*main*) cíclico e executável com um conjunto de ferramentas GNU. O Arduino IDE emprega o programa para converter o código executável em

um arquivo de texto em uma codificação hexadecimal que é carregada na placa Arduino por um programa carregador no *firmware* da placa.

3.3.2 Proteus

Proteus Design Suite é um programa proprietário, composto por uma variedade de ferramentas, incluindo captura esquemática, simulação e *design* de *layout* de placas de circuito impresso (PCBs) usadas principalmente para o projeto de circuitos integrados.

A captura esquemática no Proteus Design Suite é usada tanto para a simulação de circuitos como para a fase de esquematização de um projeto de *layout* de PCB.

A simulação de microcontroladores no Proteus funciona por meio da aplicação de um arquivo hexadecimal ou um arquivo de depuração na parte do microcontrolador no esquemático. Ele é então co-simulado juntamente com qualquer componente eletrônico analógico e digital conectados a ele. Isso permite seu uso em um amplo espectro de prototipagem de projeto.

O módulo de visualização 3D permite que a placa em desenvolvimento seja vista em 3D, juntamente com um plano de altura semi-transparente que representa o gabinete da placa. O formato de saída STEP pode então ser usado para montagem e posicionamento preciso da placa em sistemas CAD.

3.3.3 Motec ECU Manager

Para a configuração dos parâmetros do barramento CAN, definição das entradas e saídas da ECU e mapeamento de injeção e ignição, é utilizado o *software* Motec ECU Manager.

Este *software* é compatível com sistemas operacionais Windows e foi projetado para configuração, ajuste e diagnóstico do sistema de gerenciamento do motor, podendo o ajuste ser realizado *online* (com a ECU conectada) ou *off-line* (MOTEC, 2018).

O software está disponível em duas versões:

- Versão 2: com funcionalidades predefinidas, reduz a complexidade da configuração inicial. Isso resulta em uma configuração rápida, mantendo um alto nível de flexibilidade.

- Versão 3: com mais flexibilidade no número de parâmetros que podem ser configurados. Esta versão geralmente requer mais tempo de configuração inicial, mas permite um ajuste muito preciso para o aplicativo.

Algumas características do *software*:

- *Quick* Lambda - ajuste de combustível automatizado;
- Leituras de sensores e *status*, configurações de saída, compensações e erros de diagnóstico;
- *Layouts* de tela definidos pelo usuário;
- Gráficos 3D de tabelas de calibração;
- Teste de saída;
- Comparações de arquivos múltiplos;
- Interpolação de tabelas;
- Exportação de tabelas;
- Telas de ajuda.

3.3.4 Elipse SCADA

O Elipse SCADA é uma ferramenta para o desenvolvimento de sistemas de supervisão e controle de processos. O *software* possui diversos recursos que facilitam e agilizam a tarefa de desenvolvimento de aplicações. Totalmente configurável pelo usuário, permite a monitoração de variáveis em tempo real, através de gráficos e objetos que estão relacionados com as variáveis físicas de campo. Também é possível fazer acionamentos e enviar ou receber informações para equipamentos de aquisição de dados. Além disso, através de uma linguagem de programação exclusiva chamada Elipse Basic, é possível automatizar diversas tarefas específicas.

O Elipse SCADA está disponível em quatro versões, atendendo as demandas de personalização dos clientes. Estas versões se diferenciam na sua funcionalidade, cada uma acrescentando recursos em relação à versão anterior. São elas: *View*, MMI (*Man Machine*

Interface), Pro (*Professional*) e *Power*. Além destas, também há a versão Demo (demonstrativo) que limita a utilização de apenas 20 *tags*.

O Elipse SCADA possui três módulos para sua operação: Configurador, *Runtime* e *Master*. O módulo ativo é definido a partir de um dispositivo de proteção (*hardkey*) que é acoplado ao computador. Enquanto que os módulos Configurador e *Master* foram especialmente desenvolvidos para a criação e o desenvolvimento de aplicativos, o módulo *Runtime* permite apenas a execução destes. Neste módulo, não é possível qualquer alteração no aplicativo por parte do usuário (ELIPSESCADA HMI/SCADA SOFTWARE, 2005).

3.3.5 LoRa Modem Calculator Tool

A fim de simplificar as decisões de projeto usando o modem LoRa, há uma ferramenta de planejamento de *software* que permite a avaliação rápida da configuração do modem LoRa e o desempenho de tempo no ar e sensibilidade resultantes (SEMTECH, 2013).

Todos os parâmetros do modem LoRa, configurações do pacote e definições RF podem ser modificados segundo a aplicação, a fim de simular o desempenho RF, bem como os tempos de transmissão e consumo de corrente do sistema, sem que haja a necessidade de calcular manualmente as equações de projeto mencionadas na subseção 2.3.2.1.

Mais informações sobre as práticas de simulação podem ser encontradas na subseção 3.5.2.2 mais abaixo.

3.3.6 MIT App Inventor

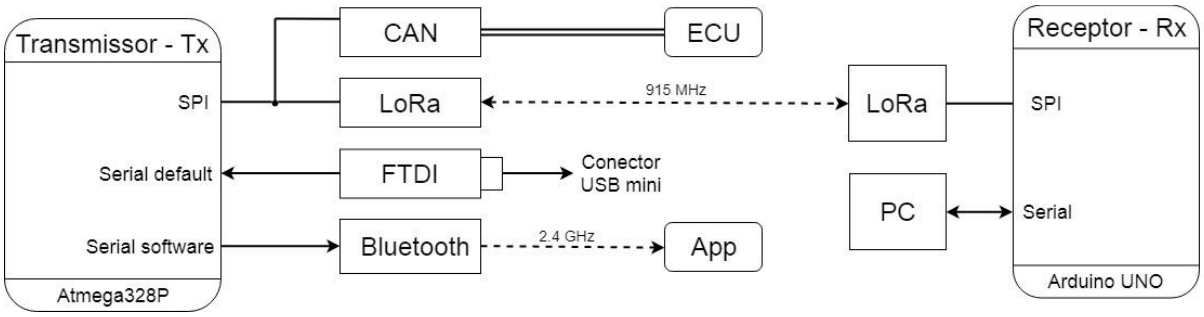
Para o desenvolvimento do aplicativo com *display* para volante e telemetria auxiliar, utilizou-se a aplicação "MIT AI2 Companion", também conhecido como App Inventor, que se encontra disponível na Google Play Store. Este aplicativo é uma aplicação de código aberto originalmente criada pela Google e atualmente mantida pelo Instituto de Tecnologia de Massachusetts (MIT) onde o usuário pode criar aplicativos de *software* para o sistema operacional Android através de uma interface gráfica intuitiva e visualmente programável, permitindo ao usuário arrastar e soltar objetos visuais em um esquema de diagrama de blocos.

3.4 REDE DE COMUNICAÇÃO

A rede de comunicação do sistema de telemetria e aplicativo inicia quando os dados gerados pelos sensores presentes no carro são enviados diretamente para a central eletrônica (ECU). Estes dados são transferidos, através do barramento CAN, para a placa transmissora do sistema. A partir disso, os dados são enviados para o aplicativo e para um módulo receptor localizado no *box* da equipe, através de tecnologia Bluetooth e LoRa, respectivamente. O modulo receptor é conectado a uma placa microcontrolada Arduino UNO através de um *shield* customizado onde, utilizando a porta Serial e um cabo USB, os dados são transferidos para um computador portátil onde são decodificados e mostrados na interface de monitoramento.

O princípio de funcionamento do sistema pode ser visto no fluxograma da Figura 3.18.

Figura 3.18- Rede de comunicação do sistema



Fonte: Próprio autor.

3.5 TELEMETRIA

O sistema de telemetria tem a função de coletar os dados gerados pelos sensores do protótipo e transmitir estes dados remotamente para um receptor conectado a uma interface de monitoramento, apresentando para a equipe as informações relevantes sobre o comportamento do veículo.

Uma lista das variáveis medidas pelo sistema pode ser vista na Tabela 5.

Tabela 5- Lista de variáveis medidas

Real	Tipo	Precisão	Identificador
Potência do sinal	Int	0	RSSI

Relação sinal / ruído	Float	2	SNR
RPM	Int	0	RPM
Velocidade	Byte	0	Speed
Temperatura do motor	Float	1	EngTemp
Marcha	Byte	0	Gear
Pressão na admissão	Float	1	MAP
Posição da borboleta	Float	1	TPS
Pressão do combustível	Float	1	FuelPress
Pressão do óleo	Float	1	OilPress
Pressão nos freios	Byte	0	BrakePress
Temperatura do ar	Float	1	AirTemp
Tensão da bateria	Float	2	BatV
Lambda	Float	2	Lambda
Temperatura do óleo	Float	1	OilTemp
Temperatura do pneu dianteiro direito	Byte	0	TempB1
Temperatura do pneu dianteiro esquerdo	Byte	0	TempB2
Temperatura do pneu traseiro direito	Byte	0	TempB3
Temperatura do pneu traseiro esquerdo	Byte	0	TempB4
Ângulo de esterçamento do volante	Int	0	SAngle
Acelerômetro	Float	1	Accel
Giroscópio	Int	0	Gyro
Latitude GPS	Float	6	GPSlat
Longitude GPS	Float	6	GPSlong
Tempo de volta	Long	0	Time
Número de cones derrubados	Byte	0	Cones
Estado do barramento CAN	Byte	0	CANstate

Identificador do pacote	Int	0	MsgCount

Fonte: Próprio autor.

Os dados dos sensores, juntamente com informações de endereçamento e um identificador de mensagem, são fragmentados *byte* a *byte* através de uma função *union,* de acordo com o tipo da variável declarada, para serem alocadas em um pacote, formando a carga útil da comunicação LoRa.

Os *bytes* do pacote recebido são decodificados pela mesma função *union*, e a atribuídos a suas respectivas variáveis referentes aos dados do veículo. Na Figura 3.19, pode-se observar um exemplo dos dados decodificados recebidos pelo receptor juntamente com as informações de pacotes perdidos, relação sinal / ruído e sensibilidade de transmissão em cada pacote.

Figura 3.19- Exemplo de recepção de pacote pelo módulo receptor

```
Received from: 0xaa
Sent to: 0xbb
Message ID: 3726
Lost packets: 0
RPM: 3254
Speed: 87
Engine Temp: 100.5
Gear: 2
MAP: 25.0
TPS: 97.7
Fuel pressure: 4.3
Oil Pressure: 3.2
Brake Pressure: 234
Air Temperature: 32.0
Battery Voltage: 12.40
Lambda: 1.00
Oil Temperature: 85.8
Steering Angle: 123
Accelerometer: 1.2
Gyroscope: 34
GPS latitude: 0.000000
GPS longitude: 0.000000
Tire 1 Temperature: 0
Tire 2 Temperature: 0
Tire 3 Temperature: 0
Tire 4 Temperature: 0
Lap Time(ms): 0
CAN state: 0
Cones hit: 1
RSSI: -70
Snr: 8.25
```

Fonte: Próprio autor.

Os dados recebidos de maior relevância são atribuídos as vinte *tags* disponíveis e enviados ao sistema supervisório pela porta serial do Arduino via cabo USB e protocolo de comunicação Modbus, onde as informações são mostradas na interface de monitoramento desenvolvida no *software* Elipse SCADA.

3.5.1 Antenas

Para estabelecer uma comunicação com o mínimo de perdas possíveis entre o transmissor e o receptor do sistema de telemetria, foram utilizados cabos coaxiais com conectores SMA para conectar os módulos a duas antenas lineares de borracha com ganho de 5dBi, projetadas para trabalhar na mesma frequência dos módulos LoRa, ou seja, 915Mhz.

Figura 3.20- Antena linear de borracha e cabo coaxial com conector SMA

Fonte: pt.aliexpress.com.

3.5.2 Transmissão de dados

Antes de determinar os parâmetros ideais da modulação LoRa, é preciso considerar algumas variáveis de transmissão de dados determinadas pela aplicação em questão, como o tamanho da carga útil (em *bytes*) e do preâmbulo (em símbolos/caracteres), a taxa de transmissão e a potência do módulo transmissor.

Na carga útil do pacote LoRa, são enviados 4 *bytes* iniciais referentes ao endereço de destino, endereço local e identificador do pacote. Segundo a Tabela 6, que demonstra o tamanho das variáveis (em *bytes*) da carga útil do pacote de acordo com o tipo das variáveis declaradas, o tamanho total da carga útil do pacote enviado pela modulação é composto por 70 *bytes* de dados.

Tabela 6- Tamanho das variáveis da carga útil do pacote

Tipo	Tamanho reservado	Quantidade	Tamanho total
Int	2	4	8
Float	4	12	48
Long int	4	1	4
Byte	1	10	10
Total	-	27	70

Fonte: Próprio autor.

O preâmbulo é usado para sincronizar o receptor com o fluxo de dados de entrada. Esta é uma variável programável para que o comprimento do preâmbulo possa ser estendido, para uso em aplicações específicas. O comprimento do preâmbulo transmitido pode ser alterado, gerando comprimentos de 6 a 65535 símbolos somados a mais 4, uma vez que a parte fixa dos dados de preâmbulo é considerada. Isso permite a transmissão de sequências arbitrariamente longas. Por padrão, o pacote é configurado com uma sequência de 8 + 4 símbolos, gerando um total de 12 símbolos.

Além disso, também é possível determinar a frequência de transmissão através da configuração do tempo de envio entre cada pacote. Levando em conta aplicações práticas descritas por Segers (2008), em seu livro sobre aquisição e análise de dados para carros de corrida, em que são utilizadas frequências de aquisição em torno dos 5 Hz, configurou-se um tempo de envio entre cada pacote de 200ms. Esta frequência de aquisição é alta o suficiente para a aplicação e baixa o suficiente para obter um bom desempenho para a modulação LoRa.

Através do tamanho da carga útil e da frequência de aquisição, é possível determinar a taxa de transmissão (*bitrate* - *Br*) mínima requisitada pela modulação LoRa. Considerando as variáveis mencionadas anteriormente e que cada *byte* é composto por 8 *bits*, a Equação (6) resulta em um *bitrate* igual a 2800 bps (bits por segundo), equivalente a 2,8 kbps.

$$Br = \frac{bytes\ x\ 8\ x\ 1000ms}{tempo\ de\ transmissão} = \frac{70\ x\ 8\ x\ 1000}{200} \qquad (6)$$

Outro parâmetro que o modem LoRa permite configurar é a potência de transmissão do módulo (*TX Power*). Este parâmetro varia de 2 a 20 (potência máxima de 100mW) e tem grande influência no consumo de energia do sistema, comprometendo também o alcance total deste. Como a fonte de energia do módulo transmissor é recarregada enquanto o veículo está ligado, o alto consumo de energia não é um quesito a ser considerado. Sendo assim, para proporcionar um maior alcance do sistema, configurou-se o sistema para utilizar a máxima potência de transmissão (20).

Cabe aqui ressaltar que quanto menor o valor da potência do sinal recebido (RSSI), maior é a sensibilidade do sistema e maior será o alcance do mesmo.

Levando em conta que um SF igual a 6 torna-se um caso especial e requer operações especiais onde cabeçalhos implícitos são obrigatórios, a utilização deste fator não se enquadra nos parâmetros necessários para esta aplicação.

Um resumo das variáveis fixas que devem ser consideradas pode ser visto a seguir:

- Tamanho da carga útil: 70 *bytes*;
- Tamanho do preâmbulo: 8 + 4 símbolos (padrão);
- *Bitrate* mínimo: 2,8 kbps;
- Potência do transmissor: 20 (100mW).

3.5.2.1 Parametrização teórica

Na folha de dados do dispositivo LoRa consta as informações da Tabela 7 que correlaciona a taxa de dados teórica para diferentes parâmetros de largura de banda, fator de espalhamento e sensibilidade. Para determinar os parâmetros mínimos de desempenho segundo a Tabela 7, a partir de uma taxa de transmissão maior ou igual a 2,8kbps, e com o objetivo de usufruir da potência máxima do modulo LoRa, a parametrização ideal para a modulação, que proporciona uma maior sensibilidade, seria:

- Largura de banda: BW = 500kHz;
- Fator de espalhamento: SF = 10;
- Sensibilidade: RSSI = -127 dBm;

- Taxa de transmissão: 2932 bps.

A tabela que correlaciona estes fatores e que está presente na folha de dados do dispositivo, pode ser observada na Tabela 7.

Tabela 7- Tabela de correlação de taxa de dados

Signal Band Width	Spreading Factor	Sensitivity (dBm)	Actual Band Rate (pbs)
62.5kHz	7	- 126	2169
62.5kHz	8	- 129	1187
62.5kHz	9	- 132	656
62.5kHz	10	- 135	296
62.5kHz	11	- 137	164
62.5kHz	12	- 139	91
125kHz	7	- 123	4338
125kHz	8	- 126	2375
125kHz	9	- 129	1312
125kHz	10	- 132	733
125kHz	11	- 133	328
125kHz	12	- 136	183
250kHz	7	- 120	8676
250kHz	8	- 123	4750
250kHz	9	- 125	2624
250kHz	10	- 128	1466
250kHz	11	- 130	778
250kHz	12	- 133	366
500kHz	7	- 118	17353
500kHz	8	- 121	9501
500kHz	9	- 124	5249
500kHz	10	- 127	2932

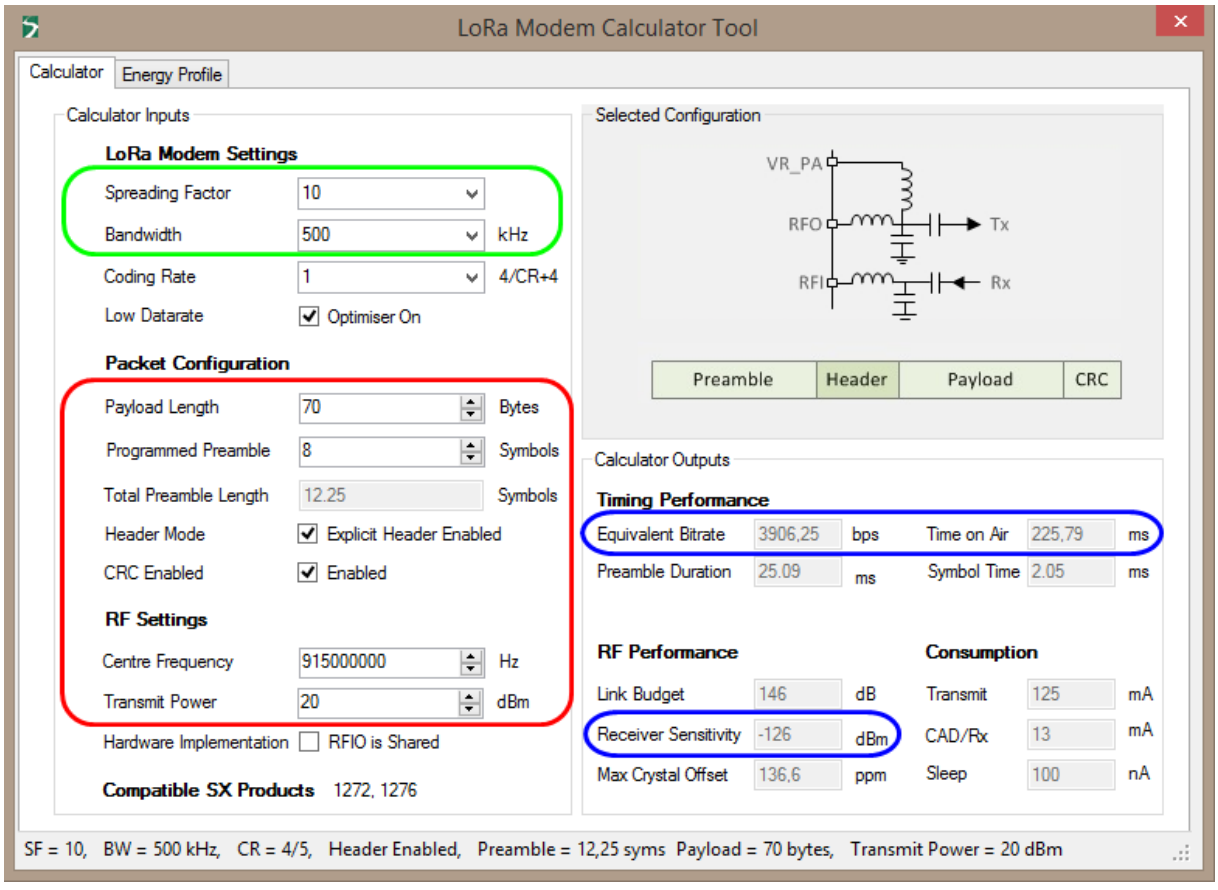
500kHz	11	- 129	1557
500kHz	12	- 130	830

Fonte: (NICERF, 2016).

3.5.2.2 Parametrização prática

Para a determinação prática dos parâmetros da modulação do tipo LoRa TM, foi utilizado o *software* LoRa Modem Calculator Tool (Figura 3.21) seguido de um teste de alcance a fim de definir a parametrização resultante que será utilizada.

Figura 3.21- Interface de simulação do software LoRa Modem Calculator Tool



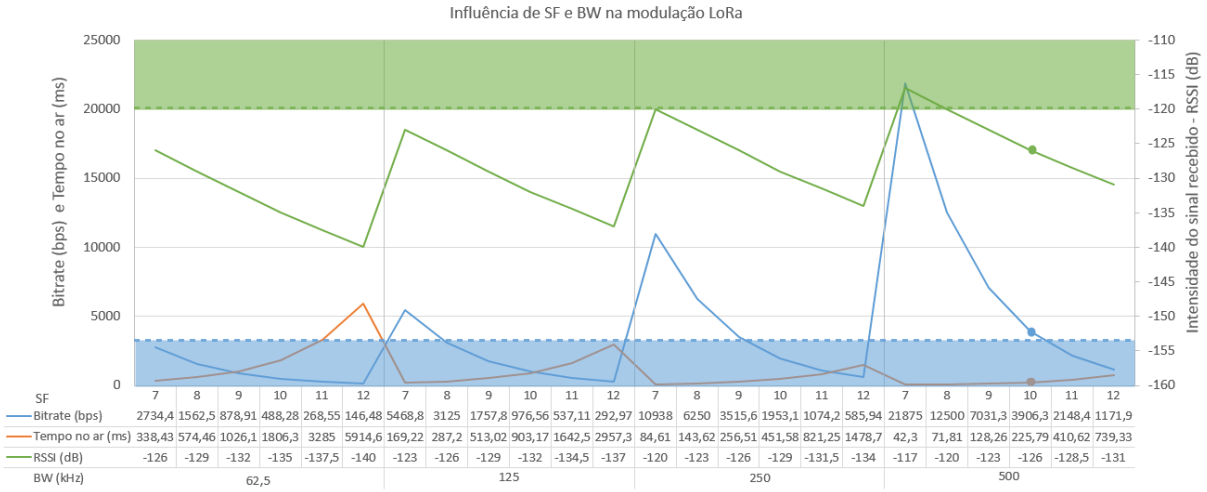
Fonte: Próprio autor.

A simulação foi realizada utilizando as variáveis fixas mencionados na subseção 3.5.2 destacadas em vermelho na Figura 3.21, considerando uma taxa de código (*Coding Rate*) igual a 1, que proporciona uma menor influência no *bitrate* e no tempo no ar. Além disso, foi habilitada a otimização de baixa taxa de dados (*Low Datarate*) para evitar problemas em torno da oscilação do oscilador de referência de cristal devido a mudança de temperatura e/ou

movimento. Variando-se entre os parâmetros de largura de banda e fator de espalhamento permitidos pelo módulo LoRa, destacados em verde na Figura 3.21, é possível obter os dados de taxa de transmissão (*Equivalent Bitrate*), tempo no ar (*Time on Air*) e sensibilidade de recepção (*Receiver Sensitivity*), marcadas em azul na Figura 3.21, para as diferentes combinações de parâmetros.

Os resultados da simulação podem ser vistos no gráfico da Figura 3.22.

Figura 3.22- Dados de resultado das simulações feitas para a modulação LoRa



SF	7	8	9	10	11	12	7	8	9	10	11	12	7	8	9	10	11	12	7	8	9	10	11	12
Bitrate (bps)	2734,4	1562,5	878,91	488,28	268,55	146,48	5468,8	3125	1757,8	976,56	537,11	292,97	10938	6250	3515,6	1953,1	1074,2	585,94	21875	12500	7031,3	3906,3	2148,4	1171,9
Tempo no ar (ms)	338,43	574,46	1026,1	1806,3	3285	5914,6	169,22	287,2	513,02	903,17	1642,5	2957,3	84,61	143,62	256,51	451,58	821,25	1478,7	42,3	71,81	128,26	225,79	410,62	739,33
RSSI (dB)	-126	-129	-132	-135	-137,5	-140	-123	-126	-129	-132	-134,5	-137	-120	-123	-126	-129	-131,5	-134	-117	-120	-123	-126	-128,5	-131
BW (kHz)	62,5						125						250						500					

Fonte: Próprio autor.

O gráfico da Figura 3.22 mostra os dados de taxa de transmissão (*bitrate*) em azul e tempo no ar em laranja (referentes ao eixo vertical esquerdo), e dados de sensibilidade (RSSI) em verde (referente ao eixo vertical direito).

Pela análise dos dados de simulação é possível perceber que a taxa de transmissão aumenta com o aumento da largura de banda, mas diminui com o aumento do fator de espalhamento. Já o tempo no ar diminui com o aumento da largura de banda e aumenta com o aumento do fator de espalhamento, enquanto que a sensibilidade do sistema diminui com o aumento da largura de banda e aumenta com o aumento do fator de espalhamento.

O objetivo é escolher a parametrização ideal para a aplicação através da taxa de transmissão mínima (valores acima da linha tracejada em azul na Figura 3.22). Sendo assim, os resultados que possuem uma taxa de transmissão igual ou maior que 2,8kbps estão representados na Tabela 8.

Tabela 8- Parâmetros com taxa maior ou igual a 2,8kbps

***Bitrate* (bps)**	**Sensibilidade (dBm)**	**Tempo no ar (ms)**	**BW (kHz)**	**SF**
5468,75	-123	169,22	125	7
3125	-126	287,2	125	8
10937,5	-120	84,61	250	7
6250	-123	143,62	250	8
3515,63	-126	256,51	250	9
21875	-117	42,3	500	7
12500	-120	71,81	500	8
7031,25	-123	128,26	500	9
3906,25	-126	225,79	500	10

Fonte: Próprio autor.

Para determinar os valores teóricos de desempenho do sistema, faz-se necessária a realização de testes práticos a fim de obter o maior alcance possível e determinar a melhor parametrização para a aplicação, dentre as mencionadas na Tabela 8.

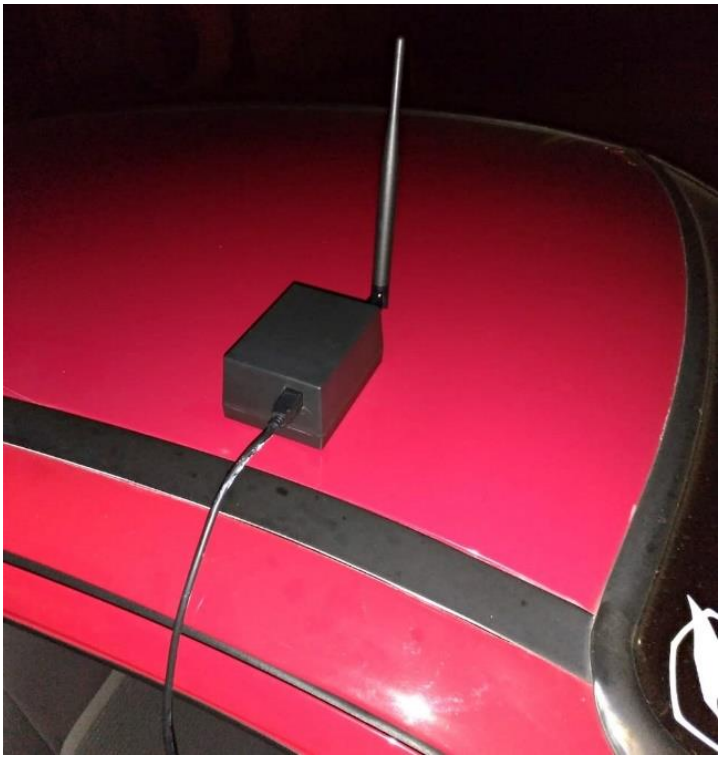
3.5.2.2.1 Teste de alcance

O ponto mais distante da pista de corrida utilizada durante a competição fica a uma distância de 350m do box da equipe, com pequenos aclives durante o percurso que podem causar pequenas intereferências na comunicação LoRa. Sendo assim, o teste realizado deve atingir um alcance de no mínimo 400m para que o o sistema tenha um desempenho suficiente e satisfatório para ser utilizado durante a competição.

O teste foi realizado em uma via com cerca de 1,6km de extensão da Universidade Federal de Santa Maria, posicionando-se o módulo transmissor em uma esquina enquanto o módulo receptor foi fixado no teto de um carro (Figura 3.23), que se afastava pela via até que a comunicação entre os módulos fosse interrompida por mais de 5 segundos, medindo assim a distância percorrida e consequentemente o alcance do sistema. Este procedimento repetiu-se por oito vezes, variando-se os parâmetros de modulação de acordo com a Tabela 8 e anotando-se os resultados obtidos.

As condições climáticas foram verificadas no dia e horário do teste, mostrando um céu limpo (sem chuva), com temperatura de 12°C, 63% de umidade no ar e ventos Sul de até 5 km/h.

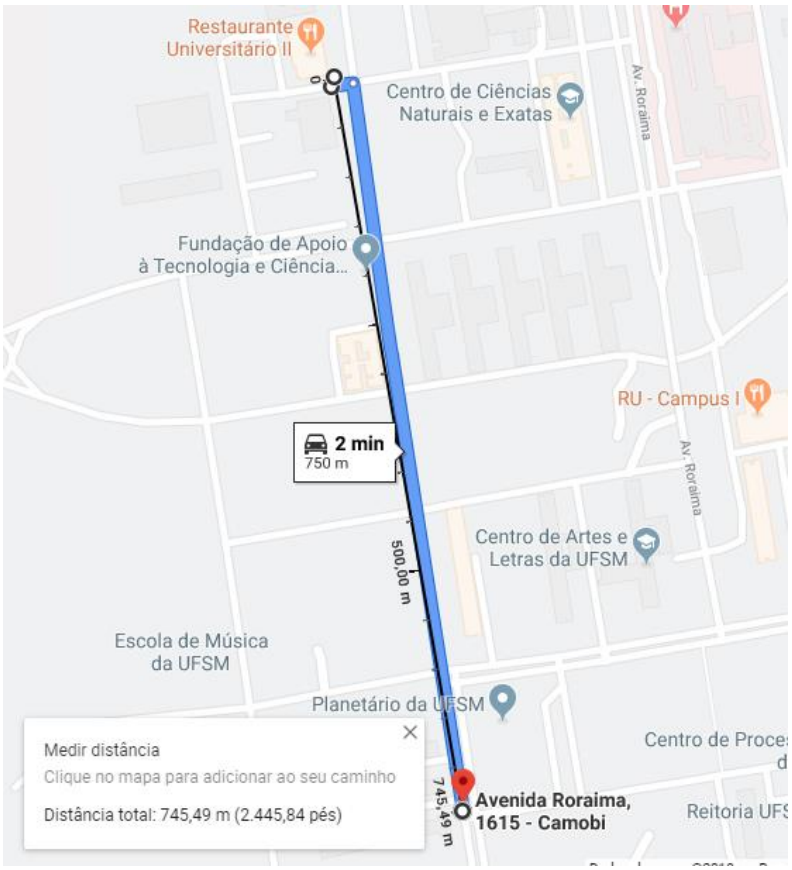
Figura 3.23- Fixação do módulo receptor para teste de alcance



Fonte: Próprio autor.

A trajetória do teste (linha azul), bem como a distância percorrida (linha preta) podem ser vistas na Figura 3.24, enquanto que os resultados de alcance (em metros) para cada configuração de parâmetros podem ser vistos na Tabela 9.

Figura 3.24- Trajetória dos testes de alcance



Fonte: Google maps.

Tabela 9- Resultados obtidos para o teste de alcance

BW (kHz)	SF	Alcance (m)
125	7	745
125	8	678
250	7	706
250	8	727
250	9	715
500	7	665
500	8	684
500	9	695
500	10	712

Fonte: Próprio autor.

Pela análise dos resultados obtidos, considerando que o maior alcance medido foi de aproximadamente 745m, a parametrização ideal para o sistema é:

- Largura de banda: BW = 125 kHz;
- Fator de espalhamento: SF = 7.

Este alcance é suficientemente grande para cobrir toda a pista durante a competição e ainda ter uma boa margem de alcance para a utilização do sistema em pistas maiores. Contudo, para obter resultados mais precisos e confiáveis sobre a parametrização ideal, é necessária a realização de uma série de repetições do teste para cada combinação de parâmetros e para diferentes condições climáticas, a fim de obter uma variância entre os resultados e verificar a influência das condições climáticas no desempenho do sistema.

3.6 APLICATIVO

Com a necessidade de mostrar ao piloto as informações necessárias para uma pilotagem hábil e segura, e de facilitar o monitoramento dos parâmetros do carro após cada teste em pista, decidiu-se pela criação de um aplicativo com três telas distintas: uma tela de *display* para volante; uma tela de monitoramento para telemetria auxiliar; e uma tela inicial que faz a conexão Bluetooth e dá acesso ás outras duas telas.

A placa transmissora utiliza um módulo Bluetooth que envia os dados para o *smartphone* onde os dados são alocados em um pacote e separados pelo caractere ‘|’ (barra), para serem decodificados através do aplicativo desenvolvido seguindo a mesma lógica. Estes dados então são mostrados no *display* para volante e na interface de telemetria auxiliar.

Os dados recebidos pela comunicação bluetooth podem ser vistos na Figura 3.25 através de um aplicativo de terminal para dispositivos seriais.

Figura 3.25- Exemplo de recepção de pacote pelo aplicativo

Fonte: Próprio autor.

3.6.1 Telas

Através de uma entrevista feita com os pilotos da equipe, definiu-se o melhor *layout* de *display* para volante de acordo com as sugestões dadas por eles. A interface do *display* para volante conta com uma barra crescente de LED's de acordo com a rotação do motor, indicador de marcha atual, indicador de temperatura do motor e indicador de temperatura do óleo. Além disso, a interface de *display* conta com avisos de momento ideal para troca de marcha (*shift light*) e de temperatura alta do motor e do óleo, através da alteração das cores dos indicadores.

A *Shift light* funciona da seguinte forma: quando a rotação do motor atinge 11000 rotações (ponto de torque máximo do motor), um temporizador (*clock*) de 500ms é habilitado, alterando a cor de fundo da tela, entre branco e preto, a cada sinal de *clock*.

O display para telemetria auxiliar conta com indicadores de rotação do motor (RPM), posição da borboleta (TPS), pressão no coletor de admissão (MAP), temperatura do motor e do óleo, quantidade de oxigênio no escapamento (Lambda), pressão na linha de combustível, pressão do óleo e tensão da bateria. Além disso, assim como no *display* para volante, o *display* para telemetria auxiliar também conta com avisos de temperatura alta do motor e do óleo, e tensão baixa da bateria, através da alteração das cores dos indicadores.

4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados e os modelos finais dos sistemas desenvolvidos, bem como algumas discussões e observações sobre o funcionamento destes.

4.1 CUSTOS DE FABRICAÇÃO

Para o cálculo do custo total do sistema são levados em conta os custos dos componentes e dispositivos utilizados, das caixas dos módulos transmissor e receptor, e do processo de prototipagem das placas, desconsiderando gastos com fretes de produtos. Além disso, levando em conta o custo zero na aquisição de alguns produtos e serviços, foi realizado um cálculo para o custo de fabricação real considerando apenas o que foi gasto e um cálculo simulado considerando o custo teórico de todos os produtos e serviços utilizados.

Segundo a Tabela 4, o valor total gasto no desenvolvimento e construção do sistema foi de R$92,15. Atribuindo valores teóricos aos componentes e dispositivos de custo zero (obtido das mesmas empresas patrocinadora) e ao processo de prototipagem das placas (através de um orçamento *online*) considerando um pedido mínimo de 5 placas, o valor gasto para o desenvolvimento do sistema seria de aproximadamente R$290,00.

4.2 TELEMETRIA

O estudo da parametrização teórica (subseção 3.5.2.1) e da parametrização prática (subseção 3.5.2.2) tem como objetivo a obtenção da configuração de parâmetros da modulação LoRa que proporciona o melhor desempenho de transmissão de dados para a aplicação em questão. Sendo assim, o teste de desempenho do sistema tem como objetivo obter a área de alcance máximo do mesmo, analisar o correto funcionamento do mesmo, e validar a escolha dos parâmetros de modulação para a aplicação.

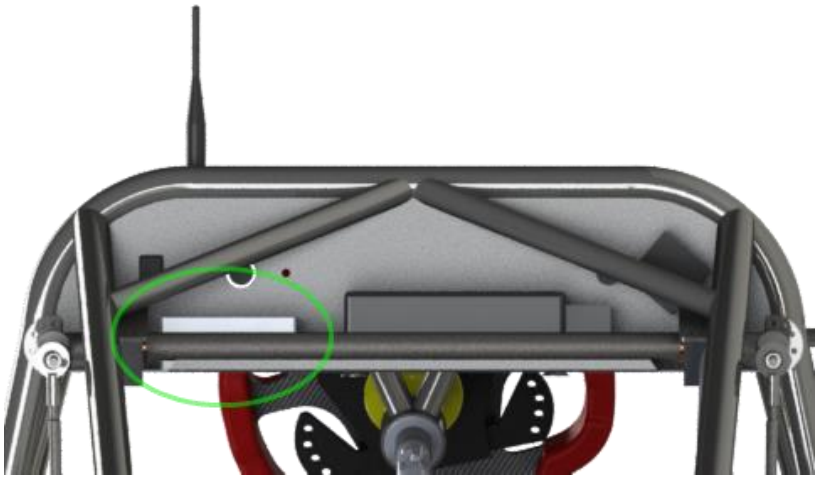
4.2.1 Parametrização

Comparando-se a parametrização ideal teórica (subseção 3.5.2.1) e prática (subseção 3.5.2.2), pode-se perceber discordâncias entre os resultados obtidos. Isso deve-se ao fato de que o teste prático foi realizado sem repetições para uma obtenção de resultados mais confiàveis, além de que o sistema pode sofrer influência de fatores como condições do ambiente e perdas físicas, fatores estes que não são considerados durante a análise teórica. Sendo assim, levando-se em conta o teste de alcance realizado, foi escolhido a utilização dos parâmetros de largura de banda (BW) igual a 125 kHz e fator de espalhamento (SF) igual a 7 para este sistema.

4.2.2 Transmissor

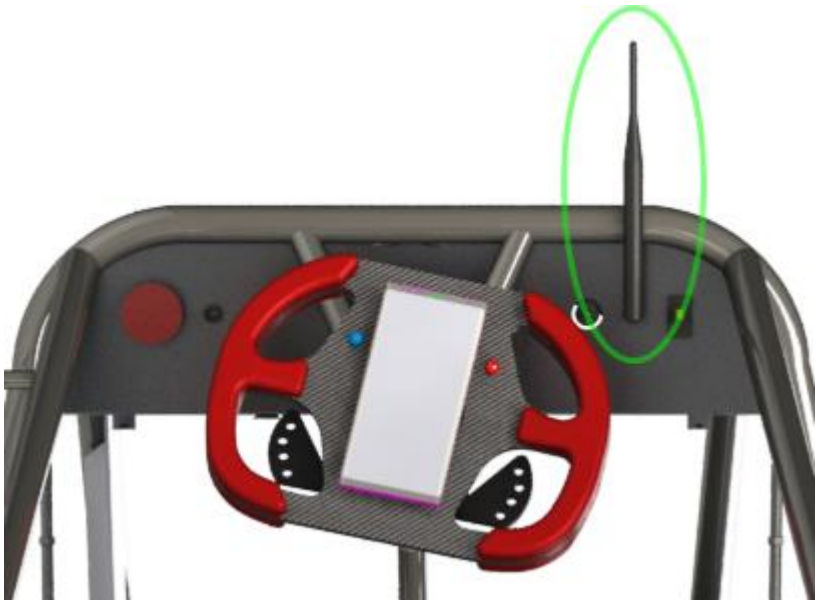
O módulo transmissor é acoplado a caixa impressa em 3D com furações para a saída da antena e do conector do chicote. A fixação do módulo transmissor é realizada através de parafusos na parte traseira do painel do veículo (Figura 4.1), enquanto que a fixação da antena é feita através de uma rosca diretamente no conector SMA do cabo coaxial que passa por um furo localizado na parte frontal do painel do veículo (Figura 4.2).

Figura 4.1- Modulo transmissor fixado atrás do painel



Fonte: Próprio autor.

Figura 4.2- Antena fixada no painel do veículo



Fonte: Próprio autor.

Este posicionamento da antena foi escolhido a fim de evitar possíveis interferências eletromagnéticas provenientes de outros dispositivos presentes no veículo e de não interferir no processo de manutenção dos outros sistemas do veículo.

4.2.3 Receptor

O *shield* receptor é conectado a placa Arduino UNO e fixado em uma caixa Patola PB-204 com furações para a saída da antena e do conector serial da placa. Além disso, a utilização de um cabo USB de 3m de comprimento e o grau de movimento da antena possibilita que este sistema receptor seja fixado em um ponto alto do box (Figura 4.3a) ou apoiado sobre uma superfície qualquer (Figura 4.3b).

Figura 4.3- Módulo receptor para fixação no box (a) ou apoiado em uma superfície (b)

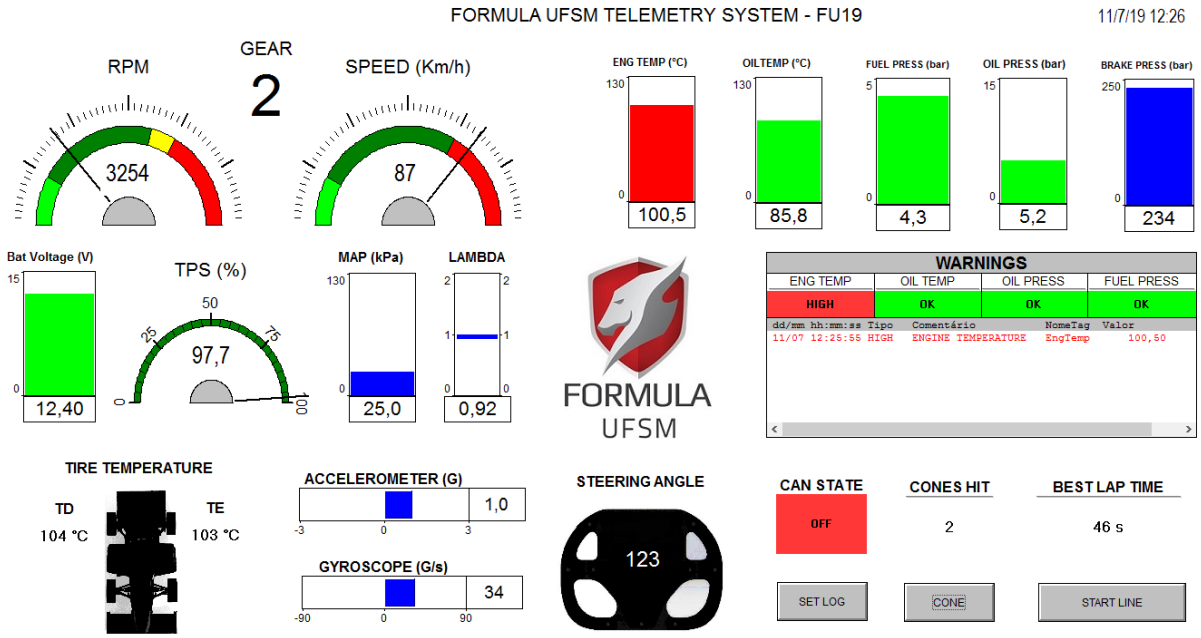
(a) (b)

Fonte: Próprio autor

4.2.4 Interface de monitoramento

A interface de monitoramento desenvolvida no *software* Elipse SCADA pode ser vista na Figura 4.4.

Figura 4.4- Interface de monitoramento.



Fonte: Próprio autor.

A interface conta com indicadores de rotação do motor (RPM), velocidade do carro, marcha, temperatura do motor, do óleo, do ar e dos pneus, pressão da linha de combustível, do óleo e da linha de freios, posição da borboleta (TPS), pressão no coletor de admissão (MAP), quantidade de oxigênio no escapamento (Lambda), tensão da bateria, ângulo de esterçamento do volante, acelerômetro e giroscópio. A interface também conta com alarmes de temperatura alta do motor e do óleo, de pressão baixa da linha de combustível e do óleo e de tensão baixa da bateria. Ainda, a interface conta indicadores de melhor tempo de volta do protótipo, número de cones derrubados e estado de funcionamento do barramento CAN.

O botão “CONE” é utilizado para registrar de forma manual cada cone que é derrubado durante o percurso do protótipo, enquanto que o botão “START LINE” registra o posicionamento da linha de largada pelos dados do GPS, enviando este dado ao módulo receptor, para que os tempos de volta possam ser calculados. Além disso, ao pressionar o botão “SET LOG”, todos os dados recebidos e mostrados até o momento são salvos em uma tabela do Excel. Esta tabela é utilizada posteriormente para análise de dados e de performance do veículo.

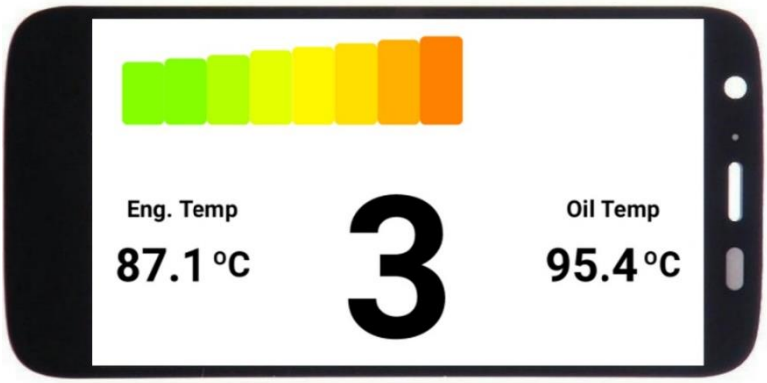
4.3 APLICATIVO

As telas do aplicativo podem ser vistas na Figura *4.5*, Figura *4.6* e Figura *4.7*.

Figura 4.5- Tela inicial do aplicativo

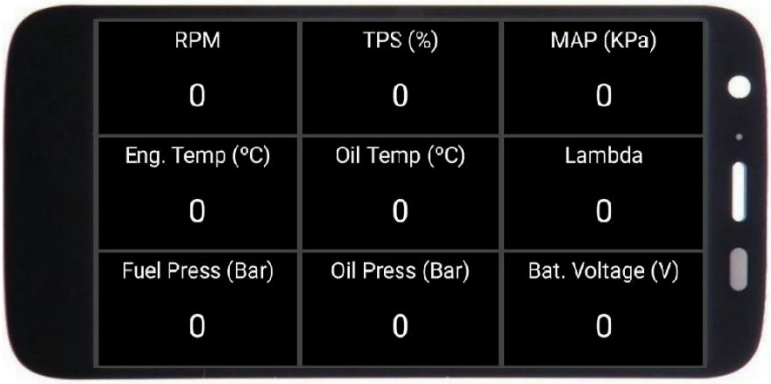
Fonte: Próprio autor.

Figura 4.6- Tela de display para volante



Fonte: Próprio autor.

Figura 4.7- Tela de telemetria auxiliar



Fonte: Próprio autor.

5 CONSIDERAÇÕES FINAIS

O objetivo principal de coletar dados de múltiplos sensores do protótipo e comunicar a um computador e a um *smartphone* em tempo real foi alcançado utilizando o sistema de telemetria desenvolvido neste trabalho. Com ele é possível obter os principais dados do veículo de forma a evitar problemas e riscos ao veículo e ao piloto. Sendo assim, o sistema mostrou-se eficiente, cobrindo uma área maior que o alcance mínimo desejado sem perda de sinal, e promissor para futuros aperfeiçoamentos quanto ao aumento do alcance de transmissão e quanto ao desenvolvimento interfaces de monitoramento mais completas.

Durante as etapas de desenvolvimento do projeto, foram encontrados problemas, como atrasos durante a entrega de alguns produtos e serviços, além de obstáculos como a dificuldade para encontrar materiais úteis sobre a tecnologia de comunicação LoRa, e contratempos com relação ao tempo de atualização de amostragem dos dados do aplicativo.

Apesar de o sistema não ter usufruído do desempenho de alcance máximo permitido pelo módulo LoRa devido a implementação de altas taxas de transmissão de dados e grandes pacotes, e considerando a distância máxima da pista da competição, o sistema desenvolvido atingiu os requisitos e resultados exigidos pelo projeto de forma satisfatória.

Através dos parâmetros de desempenho mínimo exigidos pela aplicação, dos estudos sobre os parâmetros de modulação e do resultado do teste de alcance, pode-se afirmar que para pacotes de dados menores e taxas de transmissão mais lentas, é possível obter uma área de cobertura ainda maior, com os mesmos dispositivos utilizados para a elaboração deste sistema. Ou seja, devido à alta confiabilidade e a grande variedade de parâmetros configuráveis, a modulação LoRa pode ser altamente recomendada principalmente para aplicações que não exijam altas taxas de transmissões.

É importante ressaltar que a parametrização escolhida é específica para a aplicação deste sistema, sendo necessário mais testes e/ou simulações para aplicações com diferentes requisitos de projeto. Além disso, vale destacar a importância dos testes práticos diante das simulações, utilizando os dados teóricos apenas como base para estudos mais completos.

Como recomendações para aperfeiçoamentos e otimizações futuras, sugere-se a utilização de antenas com maiores ganhos e módulos que proporcionam maior potência e sensibilidade durante a transmissão de dados. Ainda visando melhorar o desempenho e aumentar o alcance do sistema, também é possível otimizar ainda mais o envio dos dados da

telemetria através da manipulação de variáveis *float* em dois *bytes*, a fim de diminuir o tamanho dos pacotes enviados. Além disso, também é possível enviar pacotes de dados de diferentes tamanhos, agrupando-se as variáveis em pacotes determinados pela taxa de atualização das variáveis físicas medidas. A temperatura de um sistema varia lentamente enquanto que a pressão de uma linha de fluído, por exemplo, varia de forma mais rápida. Sendo assim, as variáveis de temperatura do protótipo podem ser agrupadas em pacotes menores, e enviadas com uma frequencia mais baixa que as variáveis críticas do sistema.

Outra recomendação é o estudo e implementação de uma variação de parâmetros LoRa em tempo real de acordo com o tamanho e frequência de envio das mensagens, possibilitando o envio de pacotes com diferentes taxas de transmissão. Este procedimento pode ser muito útil quando utilizada a comunicação *duplex* permitida pelo módulo LoRa.

Para este trabalho implementou-se uma comunicação ponto-a-ponto com a tecnologia LoRa. Contudo, os módulos também permitem uma comunicação em rede (LoRaWAN) muito confiável e completa, sendo possível desenvolver sistemas em rede para a mesma aplicação com a criação de mais pontos receptores, caso seja necessário. Este tipo de comunicação abre uma grande gama de possibilidades para diferentes aplicações.

Além disso, com a intenção de aperfeiçoar o projeto para o protótipo deste ano, o desenvolvimento deste sistema terá continuidade com a implementação da comunicação *duplex*, em que o módulo receptor envia ao transmissor, através da interface de monitoramento, dados como número de cones derrubados e posição da linha de largada. Estes dados são utilizados para informar ao piloto, pelo aplicativo, erros de percurso e tempos de volta.

Ainda para o protótipo deste ano, visando a adição de novas funções e dados de desempenho em pista no sistema de telemetria e de aplicativo, como tempos de volta, melhores tempos, traçado de pista, picos de medição, entre outros, sugere-se a utilização de um *software* não industrial (sugestão dos juízes durante a competição) e com uma maior aquisição de dados para o desenvolvimento da interface de monitoramento, uma vez que o *software* utilizado restringe a aquisição a apenas 20 *tags*, bem como a utilização de um *software* mais robusto e profissional para o desenvolvimento de um novo aplicativo.

Também é possível otimizar o envio dos dados para o aplicativo através da manipulação de variáveis em *bytes*, ao invés de ASCII, a fim de facilitar o envio e recebimento destas variáveis pelo aplicativo.

Para finalizar, com a intenção de proporcionar uma análise remota em tempo real dos dados obtidos do protótipo, é possível implementar um sistema de registro e uma interface de análise de dados padrão compatível com os dados gerados pela interface de monitoramento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. **O que é USB (Universal Serial Bus)?** 2017. Disponível em: <https://www.infowester.com/usb.php>. Acesso em: 5 dez. 2018.

ASSIS, Pietro Diovane Keoma Bergamaschi De. **MICROCONTROLADOR**. Barbacena. Disponível em: <http://www.unipac.br/site/bb/tcc/tcc-f6cceedfa3f6307211208b80c790c6e3.pdf>. Acesso em: 13 out. 2018.

AUGUSTIN, Aloÿs et al. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. **Sensors**, [s. l.], p. 18, 2016. Disponível em: <http://www.mdpi.com/1424-8220/16/9/1466>. Acesso em: 16 out. 2018.

BAÚ DA ELETRÔNICA, Blog. **ATMEGA328P : Aprenda a gravá-lo com Módulo Conversor FTDI USB/SERIAL**. 2018. Disponível em: <http://blog.baudaeletronica.com.br/utilizando-o-modulo-conversor-ftdi-usb-serial-para-gravar-o-microcontrolador-atmega328p/>. Acesso em: 5 jun. 2019.

BRAGA, Newton C. **Como funcionam os sensores de Efeito Hall**. 2018a. Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/6640-como-funcionam-os-sensores-de-efeito-hall-art1050>. Acesso em: 13 out. 2018.

BRAGA, Newton C. **O básico sobre os Microcontroladores – parte 1**. 2018b. Disponível em: <http://newtoncbraga.com.br/index.php/eletronica/52-artigos-diversos/13263-o-basico-sobre-os-microcontroladores-parte-1-mic139>. Acesso em: 13 out. 2018.

BRAGA, Newton C. **Como funcionam as UARTs (TEL006)**. 2018c. Disponível em: <http://newtoncbraga.com.br/index.php/telecom-artigos/1709-tel006.html>. Acesso em: 5 dez. 2018.

BRAGA, Newton C. **Soldando componentes SMD (ART1328)**. 2018d. Disponível em: <https://www.newtoncbraga.com.br/index.php/novos-componentes/52-artigos-tecnicos/artigos-diversos/6177-art1328>. Acesso em: 30 jun. 2019.

CAPELLI, Alexandre. **Eletroeletrônica Automotiva - Injeção Eletrônica, Arquitetura do Motor e Sistemas Embarcados**. 1. ed. São Paulo: Érica Ltda., 2010.

CASSIOLATO, César. **SENSORES DE PRESSÃO | SMAR - Líder em Automação Industrial**. 2018. Disponível em: <http://www.smar.com/brasil/artigo-tecnico/sensores-de-pressao>. Acesso em: 12 out. 2018.

COUNCIL, Range Commanders (Telemetry Group). Telemetry Applications Handbook. [s. l.], p. 317, 2006. Disponível em: <http://www.dtic.mil/dtic/tr/fulltext/u2/1038198.pdf>

CRUZ, Mariana. **Fórmula SAE BRASIL**. 2018. Disponível em: <http://portal.saebrasil.org.br/programas-estudantis/formula-sae-brasil>. Acesso em: 11 nov. 2018.

CURTO CIRCUITO. **Módulo RF LoRa1276 - 915MHz - 100mW - 20dBm - SPI**. 2018. Disponível em: <https://www.curtocircuito.com.br/modulo-rf-lora1276-915mhz-100mw-20dbm.html>. Acesso em: 14 out. 2018.

DEVMEDIA. **Introdução à comunicação bluetooth no Android**. 2013. Disponível em: <https://www.devmedia.com.br/introducao-a-comunicacao-bluetooth-no-android/27636>. Acesso em: 11 nov. 2018.

ELIPSESCADA HMI/SCADA SOFTWARE. . [s.l: s.n.]. Disponível em: <http://www.politecnica.pucrs.br/~filipi/scada/scadatutorial_BR.pdf>. Acesso em: 6 dez. 2018.

FRANCHI, Claiton Moro. **Instrumentação de Processos Industriais - Princípios e Aplicações**. 1. ed. São Paulo: Érica Ltda., 2015.

FREITAS, Carlos Márcio. **Protocolo Modbus: fundamentos e aplicações - Embarcados**. 2014. Disponível em: <https://www.embarcados.com.br/protocolo-modbus/>. Acesso em: 6 dez. 2018.

JUNIOR, Sergio Luis Brockveld. **Aula - Medição de Pressão**. [s.l: s.n.]. Disponível em: <http://docente.ifsc.edu.br/sergio.brockveld/MaterialDidatico/Instrumentação/Aula - Medição de Pressão.pdf>. Acesso em: 13 out. 2018.

MECAWEB. **Porta Serial**. 2016. Disponível em: <http://www.mecaweb.com.br/eletronica/content/e_serial>. Acesso em: 12 nov. 2018.

MOTEC. **GPS-L5 / GPS-L10 User Manual**. [s.l: s.n.].

MOTEC. **MoTeC > M800 > Software**. 2018. Disponível em: <https://www.motec.com.au/m800/m800software/>. Acesso em: 6 dez. 2018.

MTE-THOMSON. **Aula 8 – SENSOR DE POSIÇÃO DA BORBOLETA – Oficina do Saber**. 2018. Disponível em: <https://cursosonline.mte-thomson.com.br/unit/aula-8-sensor-de-posicao-da-borboleta/>. Acesso em: 13 out. 2018.

NICERF. **LORA1276 100mW long range Spread Spectrum modulation wireless transceiver module V2.0**. [s.l: s.n.].

ORTIZ, Fernando Molano. **Análise de Desempenho de uma Rede Sem-fio de Baixa Potência e Longo Alcance para a Internet das Coisas**. [s.l: s.n.]. Disponível em: <http://www.pee.ufrj.br/index.php/pt/producao-academica/dissertacoes-de-mestrado/2018/2016033228--84/file>. Acesso em: 6 jun. 2019.

PARTS, Boonstra. **Honda CBR 600 RR 2003-2004 - Sensor MAP**. [s.d.]. Disponível em: <https://www.boonstraparts.com/pt/componente/honda-cbr-600-rr-2003-2004-cbr600rr-pc37-sensor-map-2004-079800-5710&ID=B1000258DG>. Acesso em: 13 out. 2018.

PEREA, João E. M.; ANDRÉA, Martins; GONÇALVES VIANNA, Carla. Análise sobre experimentos com potenciômetros para a introdução do uso de sensores em cursos de física. [s. l.], p. 23, 2010. Disponível em: <http://www.sinect.com.br/anais2010/artigos/EF/61.pdf>. Acesso em: 12 out. 2018.

ROBOCORE. **Comparação Entre Protocolos de Comunicação Serial**. 2016. Disponível em: <https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html>. Acesso em: 12 nov. 2018.

SACCO, Francesco. **Comunicação SPI – Parte 1 e 2**. 2014. Disponível em: <https://www.embarcados.com.br/spi-parte-1/>. Acesso em: 27 nov. 2018.

SCHAF, Frederico. **C:310/T:310/D:DPEE1052/A:2017/P:101: A rede CAN**. 2013. Disponível em: <https://ead06.proj.ufsm.br/moodle/mod/page/view.php?id=346884>. Acesso em: 12 nov. 2018.

SEGERS, Jorge. **Analysis techniques for racecar data acquisition**. 1. ed. Warrendale: SAE International, 2008.

SEMTECH. **SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200.13**. [s.l: s.n.]. Disponível em: <https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf>. Acesso em: 20 jun. 2019.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga De. **Sensores Industriais - Fundamentos e Aplicações**. 5. ed. São Paulo: Érica Ltda., 2005.

THOMSEN, Adilson. **O que é Arduino?** 2014. Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 13 out. 2018.

VALLE, Carlos Magno Catharino Olsson. **Comunicação por Radio Frequência para Controladores Lógicos Programáveis (CLP)**. 2013. [s. l.], 2013. Disponível em: <https://www.maxwell.vrac.puc-rio.br/21715/21715.PDF>. Acesso em: 13 nov. 2018.

APÊNDICE A – CÓDIGO DO MÓDULO TRANSMISSOR

```
#include <SPI.h>
#include <LoRa.h>
#include <mcp_can.h>
#include <SoftwareSerial.h>

typedef union
{
  unsigned long l;      // long type without signal (4 bytes / 32 bits)
  float f;              // floating point type with signal (4 bytes / 32
bits)
  int i;                // integer type with signal (2 bytes / 16 bits)
  byte b[4];            // 4 position vector of byte type (4 bytes / 32
bits)
} un32;                 // union name

SoftwareSerial mySerial(6, 7); // RX, TX

// CAN Variables
long unsigned int rxId;
unsigned char len = 0;
unsigned char rxBuf[8];
MCP_CAN CAN0(8);

// LoRa Module pin mapping
const int csPin = 10;          // LoRa radio chip select (NSS)
const int resetPin = 9;        // LoRa radio reset (NRESET)
const int irqPin = 3;          // Change for your board; must be a hardware
interrupt pin (D0)

// Telemetry variables
int RPM;
byte Speed;
float EngTemp;
byte Gear;
float MAP;
float TPS;
float FuelPress;
float OilPress;
byte BrakePress;
float AirTemp;
float BatV;
float Lambda;
float OilTemp;
int SAngle;
float Accel;
int Gyro;
float GPSlat;
float GPSlong;
byte TempB1;
byte TempB2;
byte TempB3;
byte TempB4;
unsigned long Time;   //send in ms (4 bytes)
byte CANstate;

// Union variables
un32 uRPM;
un32 uEngTemp;
un32 uMAP;
```

```
un32 uTPS;
un32 uFuelPress;
un32 uOilPress;
un32 uAirTemp;
un32 uBatV;
un32 uLambda;
un32 uOilTemp;
un32 uSAngle;
un32 uAccel;
un32 uGyro;
un32 uGPSlat;
un32 uGPSlong;
un32 uTime;
un32 uMsgCount;

// Manipulation variables
int GPSlatHW;
int GPSlatLW;
int GPSlongHW;
int GPSlongLW;
byte Cones;
byte AuxCAN;
int TimeCAN = 0;
unsigned long TimeCones = 0;
unsigned long TimeStart = 0;
float StartGPSlat = 0;
float StartGPSlong = 0;
int t = 0;
int ta = 0;

// LoRa packet variables
int MsgCount;                 // count of outgoing messages
byte localAddress = 0xAA;     // address of this device
byte destination = 0xBB;      // destination to send to (with 0xFF as the
broadcast address)
long lastSendTime = 0;        // last send time
int interval = 200;           // interval between sends

void setup() {

  // Serial parameters
  Serial.begin(115200);   //FTDI
  mySerial.begin(9600);   //Bluetooth

  digitalWrite (5, HIGH); // Turn on a led if the power is On

  // CAN parameters
  pinMode(2, INPUT_PULLUP);
  if (CAN0.begin(CAN_500KBPS, MCP_8MHz) == CAN_OK) {   // init can bus :
baudrate = 1000k
  }

  // LoRa parameters
  LoRa.setPins(csPin, resetPin, irqPin); // set CS, reset, IRQ pin

  if (!LoRa.begin(915E6)) {             // initialize ratio at 915 MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                       // if failed, do nothing
  }

  LoRa.setSignalBandwidth(125E3);
```

```
  LoRa.setSpreadingFactor(7);
  LoRa.setTxPower(20);

  Serial.println("LoRa init succeeded.");

}
void loop() {

  if (!digitalRead(2)) {                        // If pin 2 is low, read re-
ceive buffer
    for (int i = 0; i < 8; i++) {
      rxBuf[i] = 0;
    }

    CAN0.readMsgBuf(&len, rxBuf);               // Read data: len = data
length, buf = data byte(s)
    rxId = CAN0.getCanId();                     // Get message ID

    if (rxId == 2005) {
      TempB1     = rxBuf[1];
      TempB2     = rxBuf[5];
    }

    if (rxId == 2006) {
      TempB3     = rxBuf[1];
      TempB4     = rxBuf[5];
    }

    if (rxId == 2015) {
      RPM        = rxBuf[0] * 256 + rxBuf[1];
      Speed      = rxBuf[2] * 256 + rxBuf[3];
      EngTemp    = rxBuf[4] * 256 + rxBuf[5];
      Gear       = rxBuf[6] * 256 + rxBuf[7];
    }
    else if (rxId == 2016) {
      MAP        = rxBuf[2] * 256 + rxBuf[3];
      TPS        = rxBuf[4] * 256 + rxBuf[5];
      FuelPress  = rxBuf[6] * 256 + rxBuf[7];
    }

    if (rxId == 2017) {
      OilPress   = rxBuf[0] * 256 + rxBuf[1];
      BrakePress = rxBuf[2] * 256 + rxBuf[3];
      AirTemp    = rxBuf[4] * 256 + rxBuf[5];
      BatV       = rxBuf[6] * 256 + rxBuf[7];
    }

    if (rxId == 2018) {
      Lambda     = rxBuf[2] * 256 + rxBuf[3];
      OilTemp    = rxBuf[4] * 256 + rxBuf[5];
      SAngle     = rxBuf[6] * 256 + rxBuf[7];
    }

    if (rxId == 2019) {
      Accel      = rxBuf[0] * 256 + rxBuf[1];
      Gyro       = rxBuf[2] * 256 + rxBuf[3];
    }

    if (rxId == 2023) {
      GPSlatHW   = rxBuf[0] * 256 + rxBuf[1];
      GPSlatLW   = rxBuf[2] * 256 + rxBuf[3];
```

```
    GPSlongHW  = rxBuf[4] * 256 + rxBuf[5];
    GPSlongLW  = rxBuf[6] * 256 + rxBuf[7];
  }

  AuxCAN = 1;
  TimeCAN = millis();
  CANstate = 1;
}

// CAN bus verification
if (AuxCAN == 0) {
  if (millis() - TimeCAN > 1000) {
    CANstate = 0;
  }
}

// GPS coordinate calculation
GPSlat = ((GPSlatHW * 65536) + GPSlatLW) / 10000000;
GPSlong = ((GPSlongHW * 65536) + GPSlongLW) / 10000000;

// Lap time calculation here
//Time.l =

uRPM.i = RPM;
uEngTemp.f = EngTemp / 10;
uMAP.f = MAP / 10;
uTPS.f = TPS / 10;
uFuelPress.f = FuelPress / 1000;
uOilPress.f = OilPress / 1000;
uAirTemp.f = AirTemp / 10;
uBatV.f = BatV / 100;
uLambda.f = Lambda / 1000;
uOilTemp.f = OilTemp / 10;
uSAngle.i = SAngle / 10;
uAccel.f = Accel / 1000;
uGyro.i = Gyro / 10;
uGPSlat.f = GPSlat;
uGPSlong.f = GPSlong;
uTime.l = Time;
uMsgCount.i = MsgCount;
BrakePress = BrakePress / 1000;

// For tests only
//  uRPM.i = RPM;
//  uEngTemp.f = EngTemp;
//  uMAP.f = MAP;
//  uTPS.f = TPS;
//  uFuelPress.f = FuelPress;
//  uOilPress.f = OilPress;
//  uAirTemp.f = AirTemp;
//  uBatV.f = BatV;
//  uLambda.f = Lambda;
//  uOilTemp.f = OilTemp;
//  uSAngle.i = SAngle;
//  uAccel.f = Accel;
//  uGyro.i = Gyro;
//  uGPSlat.f = GPSlat;
//  uGPSlong.f = GPSlong;
//  uTime.l = Time;
//  uMsgCount.i = MsgCount;
```

```
  // Send LoRa packet
  if (millis() - lastSendTime > interval) {

    LoRa.beginPacket();                   // start packet
    LoRa.write(destination);              // add destination address
    LoRa.write(localAddress);             // add sender address
    LoRa.write(uMsgCount.b, 2);           // add message ID
    LoRa.write(uRPM.b, 2);                // start vehicle variables
    LoRa.write(Speed);
    LoRa.write(uEngTemp.b, 4);
    LoRa.write(Gear);
    LoRa.write(uMAP.b, 4);
    LoRa.write(uTPS.b, 4);
    LoRa.write(uFuelPress.b, 4);
    LoRa.write(uOilPress.b, 4);
    LoRa.write(BrakePress);
    LoRa.write(uAirTemp.b, 4);
    LoRa.write(uBatV.b, 4);
    LoRa.write(uLambda.b, 4);
    LoRa.write(uOilTemp.b, 4);
    LoRa.write(uSAngle.b, 2);
    LoRa.write(uAccel.b, 4);
    LoRa.write(uGyro.b, 2);
    LoRa.write(uGPSlat.b, 4);
    LoRa.write(uGPSlong.b, 4);
    LoRa.write(TempB1);
    LoRa.write(TempB2);
    LoRa.write(TempB3);
    LoRa.write(TempB4);
    LoRa.write(uTime.b, 4);
    LoRa.write(CANstate);
    LoRa.endPacket();                     // finish packet and send it
    MsgCount++;                           // increment message ID
    lastSendTime = millis();              // timestamp the message
    AuxCAN = 0;                           // clear CAN manipulation varia-
ble

    //Serial.println("Sending: " + String(MsgCount));   //for tests only
  }

  //Send Bluetooth packet
  t = millis();
  if ((t - ta) > 500) {
    ta = millis();
    mySerial.print (t);                     //1, Control variable
    mySerial.print ("|");
    mySerial.print (RPM);                   //2
    mySerial.print ("|");
    mySerial.print ((EngTemp / 10), 1);     //3
    mySerial.print ("|");
    mySerial.print (Gear);                  //4
    mySerial.print ("|");
    mySerial.print ((MAP / 10), 1);         //5
    mySerial.print ("|");
    mySerial.print ((TPS / 10), 1);         //6
    mySerial.print ("|");
    mySerial.print ((FuelPress / 1000), 1); //7
    mySerial.print ("|");
    mySerial.print ((OilPress / 1000), 1);  //8
    mySerial.print ("|");
    mySerial.print ((BatV / 100), 2);       //9
```

```
    mySerial.print ("|");
    mySerial.print ((Lambda / 1000), 2);    //10
    mySerial.print ("|");
    mySerial.print ((OilTemp / 10), 1);     //11
    mySerial.print ("|");
    mySerial.println (ta);                  //12, Control variable
  }

  // parse for a packet, and call onReceive with the result (for LoRa du-
plex only):
  //onReceive(LoRa.parsePacket());

}

void onReceive(int packetSize) {

  if (packetSize == 0) {
    return;          // if there's no packet, return
  }

  // read packet header bytes:
  int recipient = LoRa.read();          // recipient address
  byte sender = LoRa.read();            // sender address
  byte incomingMsgId = LoRa.read();     // incoming msg ID
  byte incomingLength = LoRa.read();    // incoming msg length

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {   // check length for error
    Serial.println("error: message length does not match length");
    return;                             // skip rest of function
  }

  if (recipient != localAddress) {  // if the recipient isn't this device
    Serial.println("This message is not for me.");
    return;                             // skip rest of function
  }

  // if message is for this device:
  if (incomingMsgId == 11) {
    Cones ++;
    // Send cone indicator to bluetooth here
  }

  if (incomingMsgId == 22) {
    StartGPSlat = GPSlat;
    StartGPSlong = GPSlong;
  }
}
```

APÊNDICE B – CÓDIGO DO MÓDULO RECEPTOR

```
#include <SPI.h>
#include <LoRa.h>
#include <SimpleModbusSlave.h>

typedef union
{
  unsigned long l;      // long type without signal (4 bytes / 32 bits)
  float f;              // floating point type with signal (4 bytes / 32
bits)
  int i;                // integer type with signal (2 bytes / 16 bits)
  byte b[4];            // 4 position vector of byte type (4 bytes / 32
bits)
} un32;                 // union name

// LoRa Module pin mapping
const int csPin = 10;          // LoRa radio chip select (NSS)
const int resetPin = 9;        // LoRa radio reset (NRESET)
const int irqPin = 3;          // change for your board; must be a hardware
interrupt pin (D0)

// Modbus variables
enum
{
  RPM1,
  Speed1,
  EngTemp1,
  Gear1,
  MAP1,
  TPS1,
  FuelPress1,
  OilPress1,
  BrakePress1,
  TempB31,
  BatV1,
  TempB41,
  Lambda1,
  OilTemp1,
  SAngle1,
  Accel1,
  Gyro1,
  Time1,
  CANstate1,
  HOLDING_REGS_SIZE
};

unsigned int holdingRegs[HOLDING_REGS_SIZE];

// Telemetry variables
un32 RPM;
byte Speed;
un32 EngTemp;
byte Gear;
un32 MAP;
un32 TPS;
un32 FuelPress;
un32 OilPress;
byte BrakePress;
un32 AirTemp;
un32 BatV;
```

```
un32 Lambda;
un32 OilTemp;
un32 SAngle;
un32 Accel;
un32 Gyro;
un32 GPSlat;
un32 GPSlong ;
byte TempB1;
byte TempB2;
byte TempB3;
byte TempB4;
un32 Time;
byte CANstate;
un32 IncomingMsgId;

// Packet variables
String outgoing;              // outgoing message
byte msgID;                   // ID message
byte localAddress = 0xBB;     // address of this device
byte destination = 0xAA;      // destination to send to

// Rx variables
int Cones = 0;
int Lost = 0;

// Manipulation variables
long AUX1 = 0;
long AUX2 = 0;

void setup() {

  Serial.begin(115200);

  // LoRa parameters
  LoRa.setPins(csPin, resetPin, irqPin); // set CS, reset, IRQ pin

  if (!LoRa.begin(915E6)) {             // initialize ratio at 915 MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                       // if failed, do nothing
  }

  LoRa.setSignalBandwidth(125E3);
  LoRa.setSpreadingFactor(7);
  //LoRa.setTxPower(20);      // for LoRa duplex only

  // Modbus parameters
  modbus_configure(&Serial, 115200, SERIAL_8N1, 1, 2, HOLDING_REGS_SIZE,
holdingRegs);
  modbus_update_comms(115200, SERIAL_8N1, 1);

  Serial.println("LoRa init succeeded.");
}

void loop() {

  // for LoRa duplex only:
  // if Cone Button is pressed
  //  Cones ++;
  //  msgID = 11;
  //  sendMessage(message);
```

```
  // if Start Button is pressed
  //  msgID = 22;
  //  sendMessage(message);

  // parse for a packet and call onReceive with the result
  onReceive(LoRa.parsePacket());

  holdingRegs[RPM1] = RPM.i;
  holdingRegs[Speed1] = Speed * 100;
  holdingRegs[EngTemp1] = EngTemp.f * 100;
  holdingRegs[Gear1] = Gear;
  holdingRegs[MAP1] = MAP.f * 100;
  holdingRegs[TPS1] = TPS.f * 100;
  holdingRegs[FuelPress1] = FuelPress.f * 100;
  holdingRegs[OilPress1] = OilPress.f * 100;
  holdingRegs[BrakePress1] = BrakePress;
  holdingRegs[BatV1] = BatV.f * 100;
  holdingRegs[Lambda1] = Lambda.f * 100;
  holdingRegs[OilTemp1] = OilTemp.f * 100;
  holdingRegs[TempB31] = TempB3;
  holdingRegs[TempB41] = TempB4;
  holdingRegs[SAngle1] = SAngle.i + 1000;
  holdingRegs[Accel1] = Accel.f + 1000;
  holdingRegs[Gyro1] = Gyro.i + 1000;
  holdingRegs[Time1] = Time.l;
  holdingRegs[CANstate1] = CANstate;
  modbus_update();                    // Interface atualization

}

void sendMessage(String outgoing) {
  LoRa.beginPacket();                 // start packet
  LoRa.write(destination);            // add destination address (with 0xFF
as the broadcast address)
  LoRa.write(localAddress);           // add sender address
  LoRa.write(msgID);                  // add message ID, acording to the
button pressed
  LoRa.write(outgoing.length());      // add payload length
  LoRa.print(outgoing);               // add payload
  LoRa.endPacket();                   // finish packet and send it
}

void onReceive(int packetSize) {

  if (packetSize == 0) {
    return;                          // if there's no packet, return
  }

  if (packetSize != 70) {           // check length for error
    Lost ++;
    Serial.println("error: message length does not match length");
    return;                          // skip rest of function
  }

  // Read packet bytes:
  int recipient = LoRa.read();          // recipient address
  byte sender = LoRa.read();            // sender address
  IncomingMsgId.b[0] = LoRa.read();     // incoming msg ID
  IncomingMsgId.b[1] = LoRa.read();
  RPM.b[0] = LoRa.read();
  RPM.b[1] = LoRa.read();
```

```
Speed = LoRa.read();
EngTemp.b[0] = LoRa.read();
EngTemp.b[1] = LoRa.read();
EngTemp.b[2] = LoRa.read();
EngTemp.b[3] = LoRa.read();
Gear = LoRa.read();
MAP.b[0] = LoRa.read();
MAP.b[1] = LoRa.read();
MAP.b[2] = LoRa.read();
MAP.b[3] = LoRa.read();
TPS.b[0] = LoRa.read();
TPS.b[1] = LoRa.read();
TPS.b[2] = LoRa.read();
TPS.b[3] = LoRa.read();
FuelPress.b[0] = LoRa.read();
FuelPress.b[1] = LoRa.read();
FuelPress.b[2] = LoRa.read();
FuelPress.b[3] = LoRa.read();
OilPress.b[0] = LoRa.read();
OilPress.b[1] = LoRa.read();
OilPress.b[2] = LoRa.read();
OilPress.b[3] = LoRa.read();
BrakePress = LoRa.read();
AirTemp.b[0] = LoRa.read();
AirTemp.b[1] = LoRa.read();
AirTemp.b[2] = LoRa.read();
AirTemp.b[3] = LoRa.read();
BatV.b[0] = LoRa.read();
BatV.b[1] = LoRa.read();
BatV.b[2] = LoRa.read();
BatV.b[3] = LoRa.read();
Lambda.b[0] = LoRa.read();
Lambda.b[1] = LoRa.read();
Lambda.b[2] = LoRa.read();
Lambda.b[3] = LoRa.read();
OilTemp.b[0] = LoRa.read();
OilTemp.b[1] = LoRa.read();
OilTemp.b[2] = LoRa.read();
OilTemp.b[3] = LoRa.read();
SAngle.b[0] = LoRa.read();
SAngle.b[1] = LoRa.read();
Accel.b[0] = LoRa.read();
Accel.b[1] = LoRa.read();
Accel.b[2] = LoRa.read();
Accel.b[3] = LoRa.read();
Gyro.b[0] = LoRa.read();
Gyro.b[1] = LoRa.read();
GPSlat.b[0] = LoRa.read();
GPSlat.b[1] = LoRa.read();
GPSlat.b[2] = LoRa.read();
GPSlat.b[3] = LoRa.read();
GPSlong.b[0] = LoRa.read();
GPSlong.b[1] = LoRa.read();
GPSlong.b[2] = LoRa.read();
GPSlong.b[3] = LoRa.read();
TempB1 = LoRa.read();
TempB2 = LoRa.read();
TempB3 = LoRa.read();
TempB4 = LoRa.read();
Time.b[0] = LoRa.read();
Time.b[1] = LoRa.read();
```

```
Time.b[2] = LoRa.read();
Time.b[3] = LoRa.read();
CANstate = LoRa.read();

if (recipient != localAddress) {  // if the recipient isn't this device
  Serial.println("This message is not for me.");
  return;                         // skip rest of function
}

// if message is for this device:
//Print details
Serial.println("Received from: 0x" + String(sender, HEX));
Serial.println("Sent to: 0x" + String(recipient, HEX));
Serial.print("Message ID: ");
Serial.println(IncomingMsgId.i);
Serial.print("Lost packets: ");
Serial.println(Lost);
Serial.print("RPM: ");
Serial.println(RPM.i);
Serial.print("Speed: ");
Serial.println(Speed);
Serial.print("Engine Temp: ");
Serial.println(EngTemp.f, 1);
Serial.print("Gear: ");
Serial.println(Gear);
Serial.print("MAP: ");
Serial.println(MAP.f, 1);
Serial.print("TPS: ");
Serial.println(TPS.f, 1);
Serial.print("Fuel pressure: ");
Serial.println(FuelPress.f, 1);
Serial.print("Oil Pressure: ");
Serial.println(OilPress.f, 1);
Serial.print("Brake Pressure: ");
Serial.println(BrakePress);
Serial.print("Air Temperature: ");
Serial.println(AirTemp.f, 1);
Serial.print("Battery Voltage: ");
Serial.println(BatV.f, 2);
Serial.print("Lambda: ");
Serial.println(Lambda.f, 2);
Serial.print("Oil Temperature: ");
Serial.println(OilTemp.f, 1);
Serial.print("Steering Angle: ");
Serial.println(SAngle.i);
Serial.print("Accelerometer: ");
Serial.println(Accel.f, 1);
Serial.print("Gyroscope: ");
Serial.println(Gyro.i);
Serial.print("GPS latitude: ");
Serial.println(GPSlat.f, 6);
Serial.print("GPS longitude: ");
Serial.println(GPSlong.f, 6);
Serial.print("Tire 1 Temperature: ");
Serial.println(TempB1);
Serial.print("Tire 2 Temperature: ");
Serial.println(TempB2);
Serial.print("Tire 3 Temperature: ");
Serial.println(TempB3);
Serial.print("Tire 4 Temperature: ");
Serial.println(TempB4);
```

```
  Serial.print("Lap Time(ms): ");
  Serial.println(Time.l);
  Serial.print("CAN state: ");
  Serial.println(CANstate);
  Serial.print("Cones hit: ");
  Serial.println(Cones);
  Serial.println("RSSI: " + String(LoRa.packetRssi()));
  Serial.println("Snr: " + String(LoRa.packetSnr()));
  Serial.println();
}
```