UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO FÍSICA
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA AMBIENTAL

IMPLEMENTAÇÃO E ANÁLISE DE DESEMPENHO DE UM SISTEMA DE TRANSMISSÃO DE DADOS A LONGAS DISTÂNCIAS UTILIZANDO TECNOLOGIA LORA

PAULO HENRIQUE CORRÊA DE MORAIS

PROF. DR. PETER ZEILHOFER
ORIENTADOR

PROF. DR. ROBSON ROGÉRIO DUTRA PEREIRA
COORIENTADOR

Cuiabá, MT
05/2021

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO FÍSICA
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA AMBIENTAL

IMPLEMENTAÇÃO E ANÁLISE DE DESEMPENHO DE UM SISTEMA DE TRANSMISSÃO DE DADOS A LONGAS DISTÂNCIAS UTILIZANDO TECNOLOGIA LORA

PAULO HENRIQUE CORRÊA DE MORAIS

Dissertação apresentada ao Programa de Pós-Graduação em Física Ambiental da Universidade Federal de Mato Grosso, como parte dos requisitos para obtenção do título de Mestre em Física Ambiental.

PROF. DR. PETER ZEILHOFER
ORIENTADOR

PROF. DR. ROBSON ROGÉRIO DUTRA PEREIRA
COORIENTADOR

Cuiabá, MT
05/2021

Dados Internacionais de Catalogação na Fonte.

C824i Corrêa de Morais, Paulo Henrique.
IMPLEMENTAÇÃO E ANÁLISE DE DESEMPENHO DE UM SISTEMA DE TRANSMISSÃO DE DADOS A LONGAS DISTÂNCIAS UTILIZANDO TECNOLOGIA LORA / Paulo Henrique Corrêa de Morais. -- 2021
99 f. : il. color. ; 30 cm.

Orientador: Peter Zeilhofer.
Co-orientador: Robson Rogério Dutra Pereira.
Dissertação (mestrado) - Universidade Federal de Mato Grosso, Instituto de Física, Programa de Pós-Graduação em Física Ambiental, Cuiabá, 2021.
Inclui bibliografia.

1. Sistema embarcado. 2. Monitoramento ambiental. 3. Transmissão de dados. 4. IoT. 5. LoRa. I. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DE MATO GROSSO

PRÓ-REITORIA DE ENSINO DE PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM FOLHA DE APROVAÇÃO

FOLHA DE APROVAÇÃO

TÍTULO: **IMPLEMENTAÇÃO E ANÁLISE DE DESEMPENHO DE UM SISTEMA DE TRANSMISSÃO DE DADOS A LONGAS DISTÂNCIAS UTILIZANDO TECNOLOGIA LORA**

AUTOR : **MESTRANDO PAULO HENRIQUE CORRÊA DE MORAIS**

Dissertação defendida e aprovada em **24 DE MAIO DE 2021**.

COMPOSIÇÃO DA BANCA EXAMINADORA

Prof. Dr. Peter Zeilhofer – Orientador - Instituto de Geografia, História e Documentação IGHD

Prof. Dr. Paulo Henrique Zanella de Arruda - Examinador Interno - Instituto de Física - UFMT

Prof. Dr. Ronan Marcelo Martins - Examinador Externo - Instituto Federal de Mato Grosso – IFMT

Cuiabá-MT, 24/05/2021.

Documento assinado eletronicamente por **Ronan Marcelo martins**, **Usuário Externo**, em 25/05/2021, às 18:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **PAULO HENRIQUE ZANELLA DE ARRUDA**, **Docente da Universidade Federal de Mato Grosso**, em 26/05/2021, às 01:02, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **PETER ZEILHOFER**, **Docente da Universidade Federal de Mato Grosso**, em 26/05/2021, às 14:12, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do

E8gLSBGb2xoYSBkZSBBcHJvdmHn428gSUYgLSBQUEcgZW0gRu1zaWNhIEFtYmllbnRhbCAzNTIyMTc1 SEI 23108.037899/20

eletrônica

Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufmt.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3522175** e o código CRC **DC637CB2**.

E8gLSBGb2xoYSBkZSBBcHJvdmHn428gSUYgLSBQUEcgZW0gRu1zaWNhIEFtYmllbnRhbCAzNTIyMTc1 SEI 23108.037899/20

DEDICATÓRIA

À Deus, aos meus Alunos, aos meus pais Eraldo e Marta, as minhas irmãs Bianca e Vivi e a todos os brasileiros.

AGRADECIMENTOS

À Deus, que habitou entre nós, e mostrou como agir com o próximo e seguir em frente perante os desafios da vida.

Ao Prof. Dr. Peter Zeilhofer, pela orientação, a oportunidade, a inserção nos projetos, os trabalhos de campo, pelo incentivo e pela confiança neste trabalho.

Ao Prof. Msc. Robson Rogério Dutra Pereira, pela coorientação deste trabalho, a ajuda nos trabalhos de campo, a parceria, a troca de ideias, pelo incentivo e apoio.

Ao Prof. Dr. Ronan Martins e ao Prof. Dr. Paulo Arruda, pela disponibilidade, e as considerações que contribuíram com este trabalho.

Ao prof. Roberto Souto, pelo incentivo, apoio e parceria, a ajuda nos trabalhos de campo e o empenho para a realização do meu mestrado.

Aos professores Ruan Silva e Carlos Borges, meus colegas e companheiros, agradeço a parceria, apoio e ajuda com minhas turmas e aulas.

A minha querida amiga Isabella Silva, pela parceria desde a graduação, pela disponibilidade e ajuda com este trabalho.

Aos meus parceiros de campanhas de campo, Rubia, Antônio e Valdeci, com quem aprendi qualidades ímpares.

A todos os meus colegas do programa de pós em especial a prof.ª Luciana e o Charles, pelas reuniões de estudos e os grupos de trabalhos.

Ao programa de Pós-Graduação em Física Ambiental da Universidade Federal de Mato Grosso, ao Prof. Serginho e ao Prof. Paraná, os Técnicos e todos os professores, que compartilharam seus conhecimentos.

Aos meus alunos pela compreensão e incentivo nesta minha caminhada e ao Instituto Federal de Mato Grosso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog-to-Digital Converter
API	Application Programming Interface
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System for Mobile Communications
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LoRa	Long Range
LPWAN	Low Power Wide Area Network
MCL	Módulo Controlador LoRa (nome dado ao protótipo)
MCL-Rx	Módulo Controlador LoRa modo Receptor
MCL-Tx	Módulo Controlador LoRa modo Transmissor
NB-IoT	Narrow Band – IoT
OOK	On-Off-Keying
OTA	Over-The-Air
PAN	Personal Area Network
PCB	Printed Circuit Board
PHY	Physical layer
RF	Radiofrequência
RSSI	Received Signal Strength Indicator
RTC	Real Time Clock
Rx	Receptor
SF	Spreading Factor
SNR	Signal to Noise Ratio
SoC	System-on-a-chip
SPI	Serial Peripheral Interface
TES	Tempo de Espera do Sinal

TOA	Time on Air
Tx	Transmissor
USB	Universal Serial Bus
VSAT	Very Small Aperture Terminal
WLAN	Wireless Local Area Network

LISTA DE SÍMBOLOS

dBi	Antena isotrópica em decibéis
b	Bit
b/s	Bits por segundo
bps	Bits por segundo
B	Byte
CR	Código de redundância, onde $CR \in \{1,2,3,4\}$.
dBm	Decibel miliwatt
SF	Fator de espalhamento (7 a 12)
Hz	Hertz
BW	Largura de banda (Hz)
mA	Miliampère
mAh	Miliampère hora
mW	Miliwatt
Ω	Ohm
T_S	Período do símbolo
R_b	Taxa de bits
R_c	Taxa de chip
rc	Taxa de código
R_S	Taxa do símbolo
V	Volt

RESUMO

MORAIS, P. H. C. **Implementação e análise de desempenho de um sistema de transmissão de dados a longas distâncias utilizando tecnologia LoRa**. Cuiabá, 2021, 99p. Dissertação (Mestrado em Física Ambiental) - Instituto de Física, Universidade Federal de Mato Grosso.

A implementação de projetos de monitoramento ambiental remoto, subsidiado por componentes de transmissão de baixo custo é um desafio expressivo, principalmente em regiões periféricas sem cobertura de telefonia móvel, disponibilidade limitada de energia e difícil acesso para coleta de dados e manutenção dos equipamentos, situação típica para vastas regiões do território brasileiro. Assim, este trabalho propõe a implementação e validação de um sistema de coleta e transmissão de dados em raios até cerca de 5 km via LoRa. O protótipo desenvolvido neste trabalho é capaz de coletar e armazenar dados, enviá-los via radiofrequência LoRa e se conectar à internet via Wi-Fi. A funcionalidade dos componentes de hardware de baixo custo foi programada em código aberto que pode ser ajustado para os mais diversos tipos de sensores ou aplicações. Foram realizados testes de interação com sensores, transmissão de dados e teste de alcance de transmissão em uma região urbana e em uma região rural. Nos testes realizados os resultados o protótipo obteve desempenho satisfatório condizentes com a capacidade de alcance anunciada pelo fabricante, alcançando distâncias de 3 km em regiões urbanas e 4 km em regiões rurais.

Palavras-chave: sistema embarcado, monitoramento ambiental, transmissão de dados, IoT, LoRa.

ABSTRACT

MORAIS, P. H. C. **Implementation and performance analysis of a data transmission system over long distances using LoRa technology**. Cuiabá, 2021, 99p. Dissertation (Master's degree in Environmental Physics) - Institute of Physics, Federal University of Mato Grosso.

The implementation of remote environmental monitoring projects, subsidized by low-cost transmission components is a significant challenge, especially in peripheral regions without mobile phone coverage, limited energy availability and difficult access for data collection and equipment maintenance, a typical situation to vast regions of the Brazilian territory. Thus, this work proposes the implementation and validation of a data collection and transmission system in rays up to about 5 km via LoRa. The prototype developed in this work can collect and store data, send it via LoRa radio frequency and connect to the internet via Wi-Fi. The functionality of the low-cost hardware components was programmed in open code that can be selected for the most different types of sensors or applications. Interaction tests with sensors, data transmission and transmission range tests were carried out in an urban region and in a rural region. In the tests performed, the prototype results of satisfactory performance consistent with a reach capacity announced by the manufacturer, reaching distances of 3 km in urban regions and 4 km in rural regions.

Keywords: embedded system, environmental monitoring, data transmission, IoT, LoRa.

1. INTRODUÇÃO

1.1. Problemática

Nos mais diversos âmbitos industriais e científicos, a *Internet of Things* (IoT) experimenta um avanço vertiginoso. Conceituada como a interconexão digital de objetos, a IoT configura-se como uma rede de objetos físicos (veículos, prédios, ou sensores de monitoramento ambiental) e sua capacidade de reunir e de transmitir dados. Possibilita assim que objetos com capacidade computacional e de comunicação, se conectem à Internet para coletar e transmitir dados, além de possibilitar que sejam controlados remotamente e que os próprios objetos sejam usados como provedores de serviços (ZANJIREH & LARIJANI, 2015).

Nas ciências ambientais o uso dos conceitos de IoT é acompanhado e fomentado pelo aumento da disponibilidade de sensores de baixos custos e de fácil implementação, habilitados de coletar, em tempo real, dados dos mais diversos parâmetros que caracterizam o meio físico, tais como, a temperatura, humidade, velocidade de vento entre outros (MONTORI et al., 2018).

Mato Grosso, o terceiro maior estado do Brasil com mais do que 903.000 km² (IBGE, 2019) possui acesso a telefonia móvel, principal meio de conectividade, em todas as suas sedes municipais (ANATEL, 2020). Apesar da importância extrema da atividade agrícola para a economia do Estado e do alto grau de mecanização, um levantamento realizado pelo projeto AgriHub (2020), mostrou que, em 2018, somente 21% das sedes das propriedades rurais possuíam cobertura 3G/4G. Das propriedades, 12% não possuem acesso à internet. Outros meios de acesso utilizados nas propriedades são os seguintes: a) por sistema de rádio (41%), b) banda larga (19%) e c) satélite (8%). O mesmo estudo mostra ainda que a cobertura de conectividade nas lavouras e pastagens das propriedades alcança somente 14% (LATORRACA & SILVA, 2018). Não sendo disponíveis dados quantitativos sobre a conectividade em áreas de limitado interesse econômico, comumente objeto de estudos científicos. Estes trabalhos citados indicam a extrema dificuldade em implementar projetos de monitoramento ambiental automatizado remoto em regiões remanescentes dos principais biomas brasileiras que possuem importância primordial para a manutenção de serviços ecossistêmicos em termos globais.

1.2. Justificativa

O presente estudo se justifica frente aos desafios enfrentados por pesquisas ambientais na maior parte do território nacional que visam a implantação de sistemas de monitoramento ambiental automatizadas baseados em IoT, atividades que frequentemente não dispõem de financiamentos para a aquisição de infraestrutura comercial de comunicação que pudesse garantir conectividade em regiões remotas.

Neste contexto, este trabalho buscou estudar a aplicabilidade da tecnologia LoRa de baixo custo, baixo consumo de energia e de longo alcance de comunicação, para subsidiar estudos ambientais *in loco*, visando a integração de diversos dispositivos, tanto sensores como atuadores.

1.2.1. Objetivo Geral

Desenvolver um módulo universal de baixo custo, e avaliar o desempenho na aquisição e transmissão de dados no âmbito da IoT para aplicações em estudos ambientais.

1.2.2. Objetivos Específicos

- Priorizar a utilização de dispositivos massivos[1];
- Utilizar a tecnologia LoRa (SEMTECH, 2019) para a aquisição remota de dados e conexões de longo alcance;
- Identificar e implementar soluções de hardware e software adequadas para confecção de plataformas LoRa de baixo custo;
- Avaliar a estabilidade da comunicação LoRa e o desempenho da transmissão LoRa em relação ao distanciamento dos rádios para diferentes protótipos de comunicação.

1.3. Estrutura da Dissertação

Esta dissertação está organizada da seguinte forma:

- No Capítulo 2 é apresentada uma visão geral sobre a IoT e revisadas possíveis soluções de conectividade com ênfase na tecnologia LoRa e em aplicações de monitoramento ambiental.

[1] Dispositivos massivos possuem um baixo custo na produção, alta eficiência energética e transmitem poucos dados.

- No Capítulo 3 são abordadas as técnicas utilizadas para comunicação LoRa, o hardware empregado e detalhados os testes durante a confecção de dois protótipos.

- O Capítulo 4 mostra a concepção dos módulos desenvolvidos e sua programação e discute os resultados dos testes realizados.

- No Capítulo 5 são apresentadas as considerações finais que levantam possíveis melhorias no sistema desenvolvido e apresentam sugestões para possíveis trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

2.1. Internet das Coisas

Internet das Coisas (*Internet of Things* – IoT) é um conceito que se refere a conexão de dispositivos a partir da internet, proporcionando a interação entre o usuário e seus respectivos dispositivos "coisas". A expansão vertiginosa da IoT é fomentada pelo aperfeiçoamento e a difusão crescente de sistemas embarcados, disponíveis a preços cada vez mais acessíveis (SANTOS et al., 2016).

> A Internet das Coisas descreve a rede de objetos físicos "coisas" que são incorporados a sensores, software e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos. e sistemas pela internet. Esses dispositivos variam de objetos domésticos comuns a ferramentas industriais sofisticadas. Com mais de 7 bilhões de dispositivos IoT conectados atualmente, os especialistas esperam que esse número aumente para 10 bilhões até 2020 e 22 bilhões até 2025 (ORACLE BRASIL, 2020).

Santos et al. (2016, p. 3) enfatizam que: "É importante notar que um dos elementos cruciais para o sucesso da IoT encontra-se na padronização das tecnologias. Isto permitirá que a heterogeneidade de dispositivos conectados à Internet cresça, tornando a IoT uma realidade.".

O conceito, formalizado por Ashton (2009), em uma apresentação na Procter & Gamble (P&G) em 1999, ganhou grande amplitude nas últimas décadas, onde a utilização da IoT foi impulsionada por amplos avanços recentes no desenvolvimento tecnológico (ORACLE BRASIL, 2020):

- Acesso à tecnologia de sensores de baixo custo e baixa potência;
- Conectividade;
- Plataformas de processamento em nuvem;
- *Machine learning* e análise avançada e;
- Inteligência artificial conversacional (AI).

Dispositivos IoT possuem algumas características básicas e recorrentes. São compostos por um microcontrolador, um rádio para interface de comunicação, sensores e/ou atuadores e uma grande eficiência energética (CIRANI et al., 2014). Conforme Wi-Fi Alliance (2020), a IoT tem seu próprio conjunto de requisitos como: dispositivos com baixo consumo de energia, conexões de longo alcance, penetração

através de materiais de construção e outras obstruções e suporte para um número maior de dispositivos finais.

Os dispositivos IoT podem ser classificados conforme as suas funções, tarefas onde são empregados, prioridade de conectividade e duração de bateria. Dispositivos conectados à internet em uma quantidade expressiva, são conhecidos como dispositivos IoT Massivos. Possuem, geralmente, baixo custo na produção, alta eficiência energética e transmitem poucos dados. Já os dispositivos com alta transferência de dados, que exercem tarefas de alta confiabilidade são classificados como dispositivos IoT Críticos. A Figura 1 mostra alguns exemplos das categorias de dispositivos IoT massivos e críticos (5G AMERICAS, 2019).

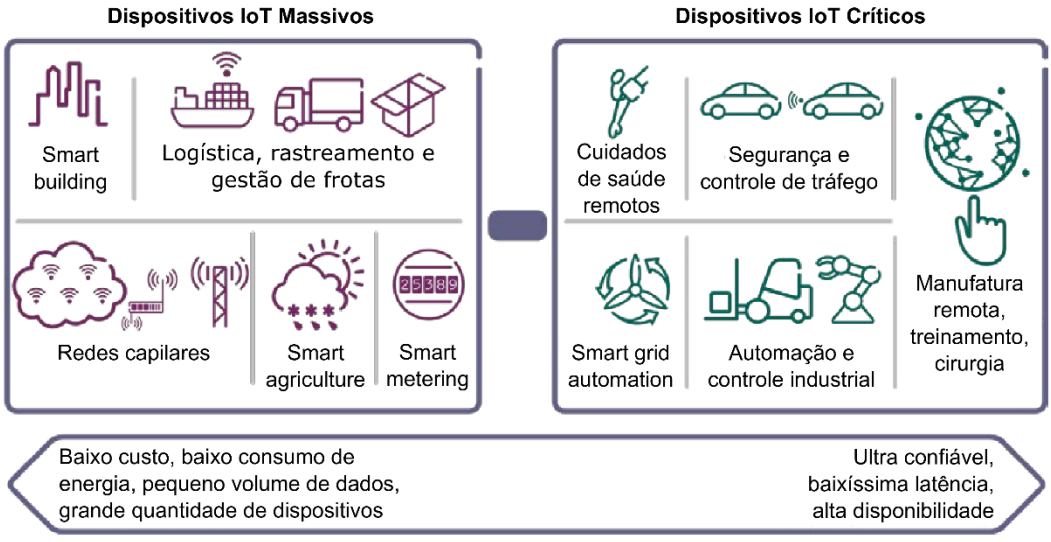

Figura 1 – Classificação dos dispositivos IoT.
FONTE: Adaptado de 5G Americas (2019)

2.1.1. Soluções para Conectividade IoT

Tecnologias sem fio possuem relevância primordial na implementação de soluções IoT, devido à flexibilidade e fácil implementação em locais remotos e de difícil acesso. As tecnologias disponíveis diferenciam-se pela capacidade na transmissão dos dados e no alcance de transmissão. Entretanto, considera-se importante avaliar a autonomia energética do dispositivo caso o sistema esteja instalado em lugares remotos (SANTOS et al., 2016).

A implementação de soluções baseadas em tecnologias sem fio, como redes Wi-Fi (IEEE 802.11), dados moveis (GPRS) ou telefonia celular (GSM) demandam,

entretanto, a disponibilidade de *hardware* com maior capacidade e sofisticação e possuem maior demanda energética (SILVA, 2019).

Estão disponíveis diversas tecnologias de comunicação sem fio que em tese podem ser utilizadas para a camada de rede de aplicações de IoT. Tecnologias como Bluetooth, Wi-Fi e Telefonia Celular, estão adequando seu portifólio para oferecer opções de conectividade para IoT. E a tecnologia LoRa surge inserida neste ecossistema IoT.

O Bluetooth, de amplo uso para aplicações de cunho cotidiano, possui como novo padrão de mercado a tecnologia Bluetooth *Low Energy* (LE) BLUETOOTH SIG (2020). É caracterizada por baixo consumo de energia e possui várias opções de camadas físicas (PHY) e comunicação em malha com vários dispositivos. Os dispositivos que utilizam a tecnologia Bluetooth possuem uma faixa de alcance de até 100 metros caracterizado com uma *Personal Area Network* (PAN), porém a tecnologia pode chegar na faixa de 1 km com rádios específicos (BLUETOOTH SIG, 2020).

Outra tecnologia amplamente utilizada para conexão entre dispositivos com altas taxas de transferência de dados é a tecnologia Wi-Fi. A tecnologia Wi-Fi classificada como *Wireless Local Area Network* (WLAN), possui vários padrões. Visando um cenário futuro para a comunicação da IoT, o Grupo de Trabalho IEEE 802.11 introduziu o Wi-Fi HaLow baseado no padrão IEEE 802.11ah. (BAÑOS-GONZALEZ et al., 2016). A tecnologia Wi-Fi HaLow opera em faixa de frequência menores que 1 GHz (sub-1 GHz) capaz de oferecer conectividade de 50 m a 1 km de distância (WI-FI ALLIANCE, 2020).

Já a conectividade por telefonia móvel, que teve início no chamado Sistema Global de Comunicações Móveis (*Global System for Mobile Communications* – GSM) possui ampla cobertura nos grandes centros populacionais. (ETSI, 2020). Em sua evolução com a expansão do acesso à internet, a tecnologia móvel é impulsionada pelo aumento de dispositivos e tráfego de dados cada vez maior com aplicativos cada vez mais complexos (5G AMERICAS, 2019). O padrão 5G será impulsionado principalmente por aplicativos de IoT, que serão tratados como dispositivos em massa e dispositivos críticos para IoT.

Um outro padrão de telefonia móvel, o *Narrow Band* – IoT (NB-IoT) é uma alternativa para grandes extensões territoriais, onde suas principais características são:

a) cobertura estendida (40 km); b) enquadra-se como *Low-Power Wide-Area Network* (LPWAN), com perda de sinal próximo a 164 dB; c) longa duração de bateria de 10 ano, e; d) suporte para aproximadamente 50.000 dispositivos por célula (FLORE, 2016).

Para locais remotos, sem acesso às tecnologias citadas anteriormente, podem ser adotadas soluções de monitoramento via Satélite utilizando redes *Very Small Aperture Terminal* (VSAT), para upload dos dados. O Sistema Brasileiro de Coleta de Dados (SBCD) utiliza satélites e uma rede de Plataformas de Coleta de Dados (PCDs) instaladas no território brasileiro. Por exemplo, as Estações de Recepção de Cuiabá e/ou Alcântara que recebem dados dos satélites e são enviados para o Sistema Integrado de Dados Ambietais (SINDA) (INPE, 2021).

A solução particular ou autônoma para coleta de dados via satélite é onerosa em planos mensais com franquias que podem variar de US$ 50 à US$ 150. Além disso, nem sempre há suporte técnico para todas as regiões do país.

Já o sistema LoRa (*Long Range*) é uma tecnologia de radiofrequência que, possui topologia de rede do tipo LPWAN, que permite, a transmissão e a recepção de dados em distâncias de até 5 km em áreas urbanas e até 15 km ou mais, em áreas rurais com visada direta. O baixo consumo de energia elétrica permite a criação de dispositivos com baterias que podem durar até 10 anos viabilizando o uso nas redes para IoT (SEMTECH CORPORATION, 2019).

2.1.2. IoT nas Ciências Ambientais

García et al. (2020) constataram, um aumento expressivo em projetos de automatização na agricultura, área que demanda funcionalidades similares como as ciências ambientais, tais como, o monitoramento de parâmetros meteorológicos e pedológicos.

Fang et al. (2014) apresentaram uma estrutura conceitual de sistemas integrados de informação no monitoramento e manejo ambiental baseados em IoT. Identificaram quatro camadas que podem compor tais sistemas, sendo as primeiras, a camada de percepção (*perception layer*) que inclui os mais diversos tipos de sensores para aquisição automática de dados e a camada de rede (*network layer*) que garante a transmissão e controle da camada de percepção.

A implementação de sistemas de monitoramento ambiental, principalmente em países em desenvolvimento, depende comumente de soluções de baixo custo. Velásquez et al. (2017) apresentaram um sistema de monitoramento atmosférico urbano composto por um nó e uma plataforma com preço abaixo de 100 US$, capaz de fornecer informações ambientais pré-processados. Consiste em uma solução que disponibiliza, por sua concepção, flexibilidade e capacidade de escalonamento na implementação de sistemas com representatividade espacial no monitoramento ambiental.

2.2. Tecnologia LoRa

A tecnologia LoRa desenvolvida pela Semtech está em amplo uso mundialmente, com mais de 100 milhões de dispositivos implantados. Seus dispositivos podem ser encontrados em diversas áreas de aplicação como automação residencial, automação industrial, logística, agricultura e meteorologia, entre outros (SEMTECH, 2019).

No Brasil as faixas de 902-907,5 MHz e 915-928 MHz está liberada pela Anatel no subitem 10.2.5 do Ato nº 14448, de 04 de dezembro de 2017, incluindo equipamentos utilizando tecnologia de espalhamento espectral ou outras tecnologias de modulação digital. Tantos dispositivos finais, quanto gateway LoRa, devem ser homologados na Anatel para seu uso comercial (ANATEL, 2018).

2.2.1. Modulação LoRa

LoRa se baseia em uma técnica de modulação espectro de espalhamento derivada da tecnologia *chirp spread spectrum* (CSS) (SEMTECH, 2019).

A técnica de modulação CSS foi usada no passado, em meados de 1940, conhecida até então como *Chirp Pulse*, em aplicações de radar, com desenvolvimento consecutivo por Sidney Darlington (NANOTRON, 2019). Porém a tecnologia LoRa é a primeira implementação de baixo custo baseada em CSS, já que o seu longo alcance pode utilizar um único gateway ou estação base para cobrir centenas de quilômetros quadrados (LORA ALLIANCE, 2015).

A técnica *Chirp Pulse* é uma frequência modulada em pulsos, com duração “T”. Neste período, a frequência muda de maneira monotônica de um valor mais baixo

para um valor mais alto *Upchirp* ou reverso *Downchirp* (Figura 2) (LAMPE & IANELLI, 2003).

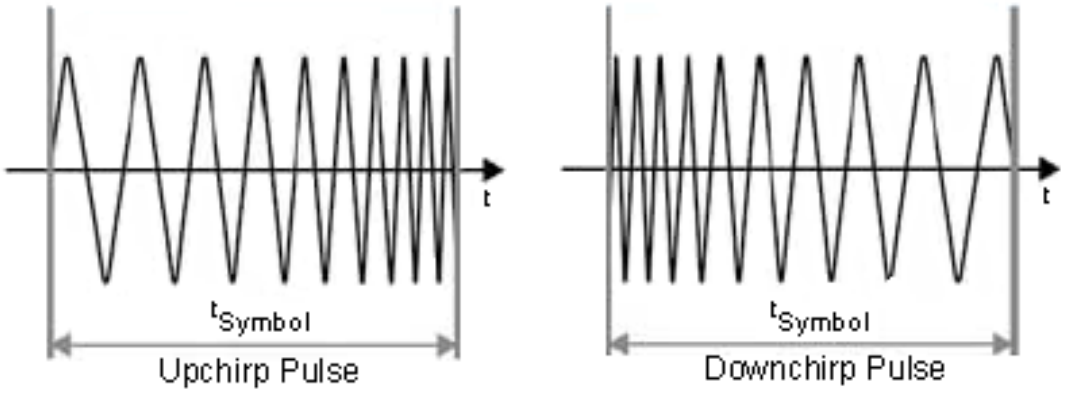

Figura 2 – Chirp pulse no domínio do tempo - Upchirp e Downchirp
FONTE: Adaptado de Nanotron (2019)

Lampe & Ianelli (2003) apresentam um exemplo da técnica de modulação *On-Off-Keying* (OOK) que utiliza o CSS (Figura 3), na qual *Upchirp* representa o nível alto e nulo o nível baixo, permitindo duas redes independentes coexistentes.

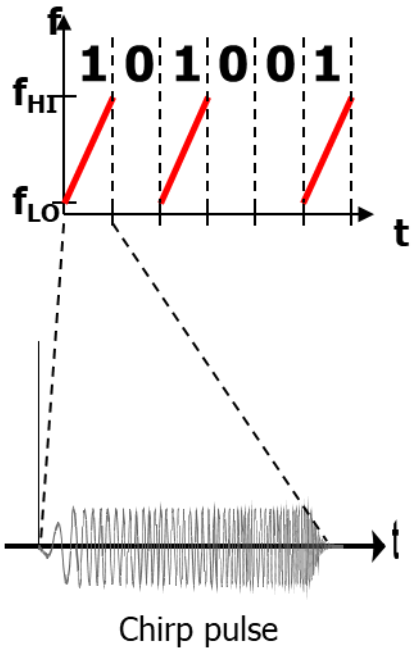

Figura 3 – Modulação OOK usando CSS.
FONTE: Lampe & Ianelli (2003)

Na modulação LoRa um formato de *frame* para a camada física é especificado e implementado nos transmissores e receptores (Figura 4). A primeira especificação é

o preâmbulo, onde se inicia uma sequência de *upchirp* constantes que cobre toda a banda de frequência. Após o preâmbulo inicia-se o cabeçalho acompanhado de um teste antes do envio efetivo do código *Cyclic Redundancy Check* (CRC), funcionando como um sincronismo para os dispositivos. Depois dos símbolos de cabeçalho a carga útil é codificada e checada com o CRC no final do frame (AUGUSTIN et al., 2016).

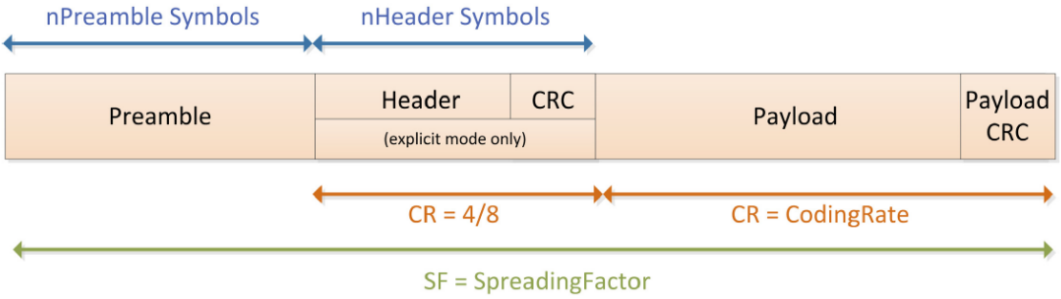

Figura 4 – Estrutura do pacote LoRa.
FONTE: SEMTECH CORPORATION (2013)

A modulação LoRa inclui um esquema de correção de erros variável que melhora a robustez do sinal transmitido. Para cada quatro bits de informação enviados, um quinto bit de informação de paridade é enviado (SEMTECH CORPORATION, 2019).

Baseado no manual de modulações básicas LoRa AN1200.22 (SEMTECH CORPORATION, 2015), a relação entre a taxa de bits de dados desejada, a taxa de símbolos e a taxa de chips para a modulação LoRa pode ser expressa conforme a equação (1), a taxa de bits (R_b) modulada definida como:

$$R_b = SF \times \frac{1}{\left[\frac{2^{SF}}{BW}\right]} \; [\, bit/s \,] \qquad (1)$$

Onde:

$R_b = taxa\ de\ bits$;

$SF = fator\ de\ espalhamento\ (7\ a\ 12)$;

$BW = Largura\ de\ banda\ (Hz)$.

Para encontrar o período do símbolo (T_S), utiliza-se a equação (2):

$$T_S = \frac{2^{SF}}{BW}\ [\ s\ ] \tag{2}$$

Onde:

$T_S = Perído\ do\ símbolo$;

$SF = fator\ de\ espalhamento\ (7\ a\ 12)$;

$BW = Largura\ de\ banda\ (Hz)$.

Portanto, a taxa de símbolos (R_S) pode ser obtido, utilizando o "T_S" da equação (2):

$$R_S = \frac{1}{T_S} = \frac{BW}{2^{SF}}\ [\ símbolos\ /\ s\ ] \tag{3}$$

Onde:

$R_S = Taxa\ do\ símbolo$;

$T_S = Perído\ do\ símbolo$;

$SF = fator\ de\ espalhamento\ (7\ a\ 12)$;

$BW = Largura\ de\ banda\ (Hz)$.

E, assim, a taxa de chip (R_c), resulta em:

$$R_c = R_S \times 2^{SF}\ [\ chips\ /\ s\ ] \tag{4}$$

Onde:

$R_c = Taxa\ de\ chip$;

$R_S = Taxa\ do\ símbolo$;

$SF = fator\ de\ espalhamento\ (7\ a\ 12)$;

Ao substituir a equação (3) na equação (4), percebe-se que na modulação LoRa, a taxa de chip depende apenas da largura de banda, ou seja, a taxa de chip é igual à largura de banda (um chip é enviado por segundo por Hertz da largura de banda), obtendo-se a equação (5):

$$R_c = \frac{BW}{2^{SF}} \times 2^{SF} \rightarrow\ R_c = BW[\ chips\ /\ s\ ] \tag{5}$$

Além disso, o LoRa inclui um código de correção direta de erros. A taxa de código (rc) é igual a equação (6):

$$rc = \frac{4}{4 + CR} \tag{6}$$

Onde:

$rc = Taxa\ de\ código$;

$CR\ =\ Código\ de\ redundância$, onde $CR \in \{1,2,3,4\}$.

Considerando isso, além da informação útil de bits transmitidas por símbolo, mais o código de correção, podemos encontrar a taxa de bits nominal, de acordo com a equação (7):

$$R_b = SF \times \frac{rc}{\left[\frac{2^{SF}}{BW}\right]}\ [\ bits/s\ ] \tag{7}$$

A tecnologia LoRa realiza um ajuste entre sensibilidade e taxa de dados em seus dispositivos, enquanto opera em um canal de largura de banda fixa de 125 KHz ou 500 KHz para canais de *uplink* e 500 KHz para canais de *downlink*. (SEMTECH CORPORATION, 2019). Analisando a equação (2) é possível observar que, quanto maior o fator de espalhamento (*Spreading Factor* – SF), maior é o tempo de permanência do espectro no ar e quanto menor o SF, maior a taxa de dados. (Figura 5)

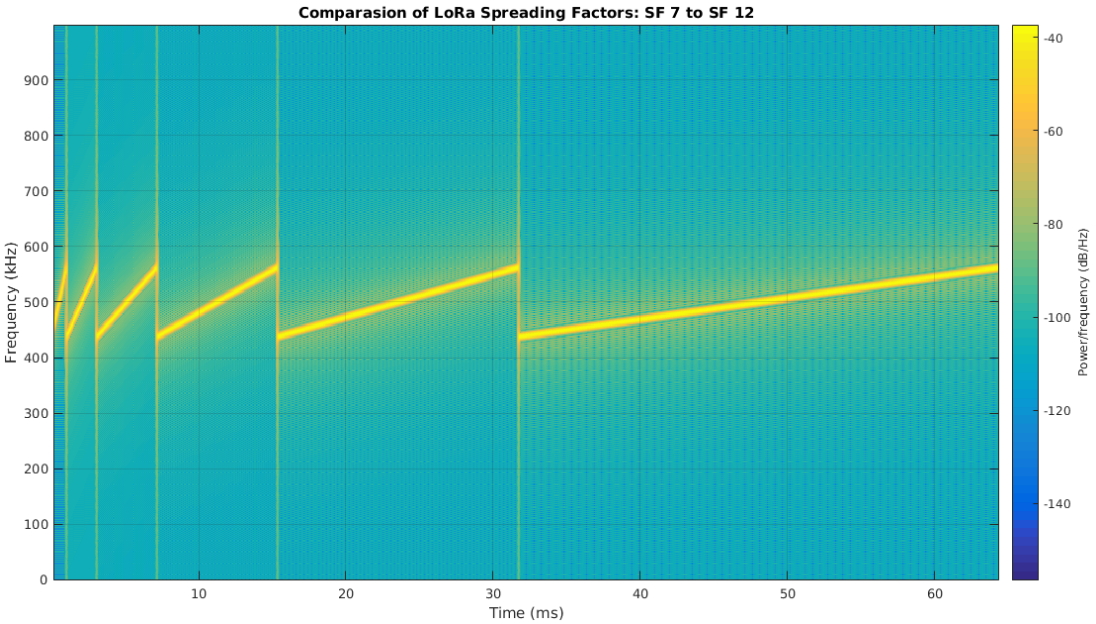

Figura 5 – Comparação dos fatores de espalhamento SF (7-12).
FONTE: Ghoslya (2020)

A Tabela 1 mostra quatro fatores de espalhamento para uma mensagem de *uplink* (UL) em um canal de 125 KHz, com a taxa de bits suportada, a distância que pode causar variações conforme o terreno (em áreas rurais pode-se alcançar maiores distancias em relação a áreas urbanas) e o tempo de permanência no ar (TOA), para uma carga útil de 11 Bytes para cada um dos quatro fatores de espalhamento.

Tabela 1 – Variação do fator de espalhamento (SF).

Fator de espalhamento (UL em 125 kHz)	**Taxa de bits**	**Distância**	***Time-on-Air* para 11 Bytes uteis**
SF10	980 bps	8 km	371 ms
SF9	1760 bps	6 km	185 ms
SF8	3125 bps	4 km	103 ms
SF7	5470 bps	2 km	61 ms

FONTE: Adaptado de SEMTECH CORPORATION (2019)

A sensibilidade do receptor à temperatura ambiente é dada pela equação (8):

$$S = -174 + 10\ \log_{10} BW + NF + SNR \qquad (8)$$

Onde:

1º Termo da equação (8): O valor numérico de -174, é devido ao ruído térmico em 1 Hz de largura de banda e só pode ser influenciado pela alteração da temperatura do receptor;

2º Termo da equação (8): Relacionado a largura de banda (BW) do receptor;

3º Termo da equação (8): É o valor do ruído do receptor (NF) e é fixo para uma determinada implementação de hardware;

4º Termo da equação (8): Representa a relação sinal / ruído (SNR), quanto menor esse número, mais sensível será o receptor.

A Tabela 2 apresenta alguns valores típicos de SNR para algumas configurações de modulação (SEMTECH CORPORATION, 2013).

Tabela 2 – SNR para várias configurações de modulação.

Modulação	SNR Típicos
LoRa SF12	-20 dB
LoRa SF10	-15 dB
GMSK	9 dB

FONTE: Adaptado de SEMTECH CORPORATION (2013)

Conforme SEMTECH CORPORATION (2015), as principais propriedades da modulação LoRa são:

- Largura de banda escalável: a modulação LoRa é escalável em largura de banda e frequência., adaptando-se facilmente a outro modo de operação através de alterações no registro de configuração.
- Baixo consumo de energia.
- Alta robustez a mecanismos de interferência.
- Resistente a múltiplos caminhos, tornando-o ideal para uso em ambientes urbanos e suburbanos, onde ambos os mecanismos dominam.
- Resistente ao efeito Doppler: o deslocamento Doppler causa uma pequena mudança de frequência no pulso LoRa, que introduz um deslocamento relativamente insignificante no eixo do tempo do sinal da banda base. Essa tolerância de desvio de frequência atenua o requisito de um oscilador de alta precisão.
- O LoRa é ideal para links de comunicação de dados em dispositivos em movimento, como sistemas de monitoramento sem fio da pressão dos pneus, aplicativos de direção, como pedágio e leitores de *tag's*.

- Capacidade de longo alcance, até 5 km em áreas urbanas e de até 15 km ou mais em áreas rurais.
- A modulação Semtech LoRa emprega fatores de propagação ortogonais que permitem que vários sinais de propagação sejam transmitidos ao mesmo tempo e no mesmo canal.
- LoRa é a modulação ideal para aplicações de radar e, portanto, é ideal para aplicações de alcance e localização, como serviços de localização em tempo real.

2.2.2. Protocolo LoRaWAN

LoRa é a tecnologia empregada no protocolo LoRaWAN que utiliza redes do tipo LPWAN, cuja padronização é de responsabilidade de uma associação conhecida como LoRa Alliance. Sua arquitetura de rede utiliza a topologia estrela (Figura 6), onde os dispositivos finais se comunicam com os *Gateway* por RF utilizando a tecnologia LoRa (LORA ALLIANCE, 2020).

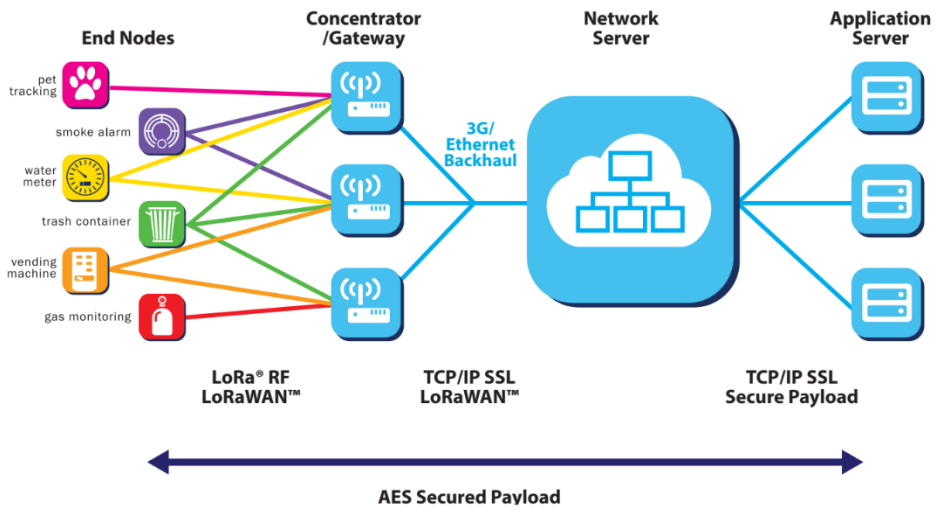

Figura 6 – Topologia do protocolo LoRaWAN.
FONTE: LoRa Alliance (2015)

Os *Gateways* são responsáveis por estabelecer a comunicação entre os dispositivos finais e o servidor, convertendo pacotes RF em pacotes IP e vice-versa. A comunicação entre o servidor e os *Gateway* é dado por meio de um *link* confiável de alta velocidade. Os dispositivos possuem comunicação bidirecional e são habilitados

para realizar atualização do *Firmware* utilizando a próprio canal RF técnica conhecida com *Over-The-Air* (OTA) (LORA ALLIANCE, 2020).

O protocolo LoRaWAN classifica seus dispositivos finais em três classes em função de suas aplicações (Figura 7):

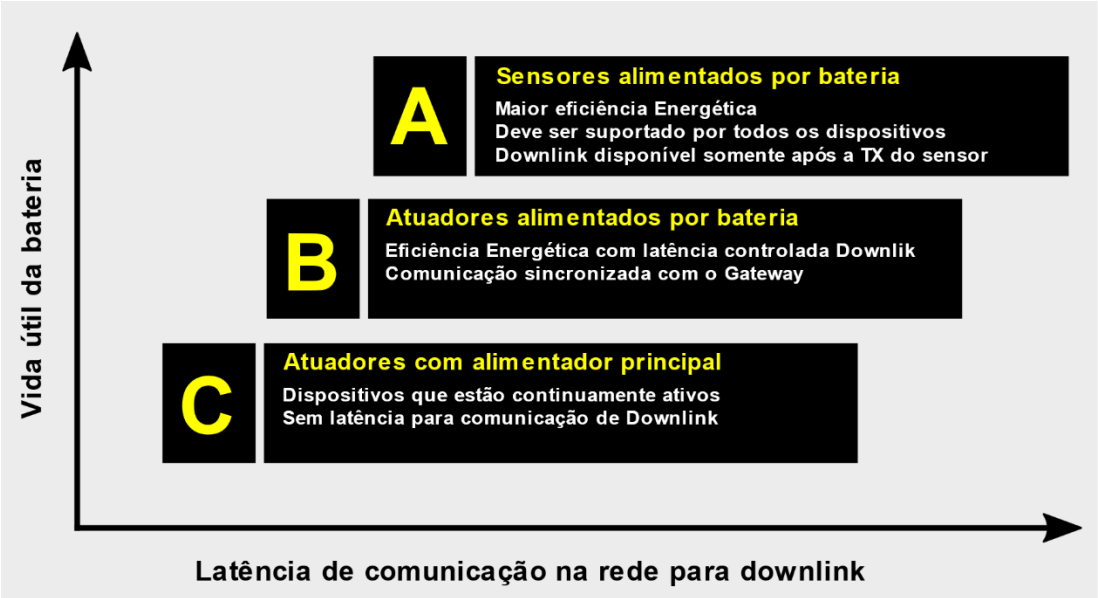

Figura 7 – Classificação dos dispositivos no protocolo LoRaWAN.
FONTE: Adaptado de LoRa Alliance (2015)

- **Classe A – Dispositivos bidirecionais de baixo consumo de energia:** É a classe padrão que deve ser suportada por todos os dispositivos finais do protocolo LoRaWAN. A comunicação sempre é iniciada pelo dispositivo final e é totalmente assíncrona. Após estabelecer o *Uplink*, duas janelas curtas de *Downlink* são abertas, disponibilizando uma comunicação bidirecional, se necessário. O dispositivo final pode entrar no modo de economia de energia, sendo ativado conforme sua aplicação (LORA ALLIANCE, 2020).

- **Classe B – Dispositivos bidirecionais com latência determinística para *downlink*:** Além da função de *Uplink* da classe A, os dispositivos da classe B são sincronizados à rede, utilizando sinais periódicos e permitem o *Downlink* em horários programados. Isso permite a rede enviar comunicação de *Downlink* com latência determinística, porém isso custa um consumo adicional de energia no dispositivo final (LORA ALLIANCE, 2020).

- **Classe C – Dispositivos bidirecionais com menor latência:** Utiliza a estrutura padrão da classe A, porém na classe C a latência é ainda mais reduzida, mantendo o dispositivo receptor aberto o tempo todo quando não está transmitindo (*half duplex*). Por isso, o servidor pode iniciar o

> *Downlink* a qualquer momento em que a via não esteja ocupada. Assim o consumo de energia é maior, portanto, a classe C é indicada para dispositivos com energia contínua disponível (LORA ALLIANCE, 2020).

Os pacotes enviados entre os dispositivos finais e o *Gateway*, possuem uma configuração variável para a taxa de dados o que permite uma troca dinâmica entre o alcance da comunicação e a duração da mensagem. Para aumentar a vida útil da bateria e a capacidade geral da rede, o servidor de rede LoRaWAN gerencia a configuração de taxa de dados e a potência de saída RF de cada dispositivo final individualmente. As taxas de transmissão variam de 0,3 kbps a 50 kbps. O protocolo LoRaWAN possui duas camadas de criptografia, uma senha exclusiva de 128 bits compartilhada entre os dispositivos finais e o servidor LoRaWAN. A senha para sessão de aplicativos (*AppSKey*) é compartilhada ponta a ponta no nível do aplicativo (LORA ALLIANCE, 2020).

2.2.3. Aplicações LoRa nas Ciências Ambientais

2.2.3.1. Caso 01

URSALINK (2020) com seus parceiros implementaram um projeto piloto de agricultura digital na Áustria. O projeto consiste na coleta, leitura e visualização de dados adquiridos de vários sensores que utilizam o protocolo LoRaWAN. Com os dados obtidos verifica-se as condições fitossanitárias de culturas, pois as plantas precisam ser protegidas de condições meteorológicas prejudiciais principalmente no início do desenvolvimento quando ainda são particularmente propensas a doenças. O monitoramento inclui a temperatura , umidade (controle de bactérias, vírus e fungos) e radiação solar (URSALINK, 2020). Foi instalado um gateway Ursalink UG87 LoRaWAN no topo de um prédio com os sensores colocados em locais equidistantes dentro dos blocos de manejo. O gateway coleta os dados a cada 30 minutos e disponibiliza em uma plataforma na internet (Figura 8).

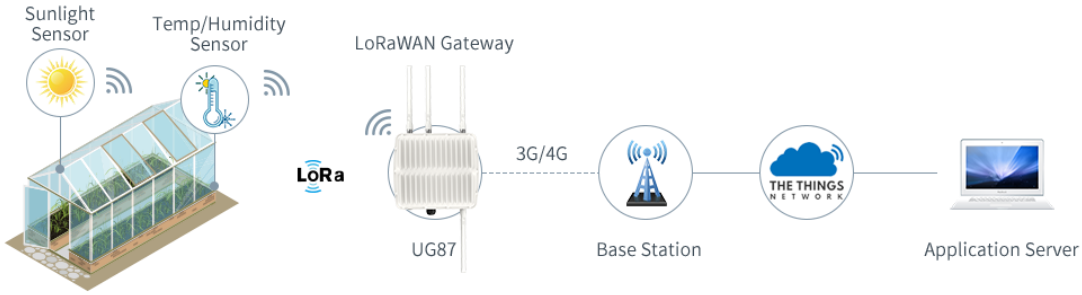

Figura 8 – Esquema de monitoramento utilizado pela URSALINK.
FONTE: Ursalink (2020)

2.2.3.2. Caso 02

Avaliando um projeto de monitoramento ambiental baseados em LoRa com integração em nuvem para área rural realizado na Universidade de Gadjah Mada, Hidayata et al. (2019) salientam a utilidade do uso da IoT para observação e análise das condições das plantas e do meio ambiente. Salientam a importância de uma concepção eficiente na transmissão e o gerenciamento de dados.

O projeto consiste em um Nó de Detecção (integrado com um sensor de temperatura e umidade e um sensor de radiação solar, o rádio LoRa e baterias), Gateway (integrado com um RTC, um SD Card, um rádio LoRa e baterias) e Servidor em Nuvem. O sistema usa duas conexões para transmissão de dados, que é a conexão LoRa e a conexão com a internet.

O sistema transfere com sucesso os dados desde o nó de detecção para o gateway até o servidor em nuvem com uma conexão Wi-Fi. Melhores resultados foram obtidos para uma linha de visada de 800m de distância com apenas 20% de perda de pacote (HIDAYATA et al., 2019).

2.2.3.3. Caso 03

Um experimento desenvolvido por Husein et al. (2019) trata de um sistema de baixo custo para o monitoramento da poluição do Ar, utilizando a tecnologia LoRa.

O sistema é baseado na implementação de uma rede de sensores sem fio. Consiste em três nós distribuídos em uma distância de 900m até o Gateway para medir a concentração de monóxido de carbono, dióxido de carbono e óxido de nitrogênio. Os dados obtidos podem ser visualizados por meio de um sistema Web.

O experimento mostrou que a recepção dos pacotes de dados foi de 100% a uma distância de 0 a 900 m. Houve primeiras perdas após 900 m e uma perda de 95%

a 1 km e 1,1 km. O sistema também comprovou sua capacidade de detectar poluição e classificar seu nível de poluição (HUSEIN et al., 2019).

Husein et al. (2019, tradução nossa[2]), concluem que “O experimento com a transmissão LoRa mostrou que a tecnologia LoRa é muito adequada para o sistema de poluição do ar, especialmente na transmissão de longo alcance, em comparação com outras técnicas de transmissão sem fio”. O sistema consiste em sensores de baixo custo (MQ7-MQ2- MQ135) conectados ao microcontrolador ATmega328, que segundo os autores se mostrou eficaz aos testes.

[2] “The experiment with LoRa transmission has shown that the LoRa technology is very suitable for the air pollution system especially in long range transmission compared to other wireless transmission techniques.” (HUSEIN et al., 2019)

3. MATERIAIS E MÉTODOS

3.1. Materiais

Neste trabalho foram implementados dois protótipos. Inicialmente, para o primeiro protótipo foi utilizado o Módulo WiFi LoRa 32 da Heltec Automation, que já possui o microcontrolador ESP32 e o Chip LoRa SX1276 integrados na mesma placa. No segundo protótipo foi utilizado a placa de desenvolvimento DOIT Kit V1 junto com um rádio LoRa da Ai-Thinker.

Para os dois protótipos foram utilizados dois rádios RF LoRa em transmissão ponto a ponto, sendo um utilizado como transmissor (Tx) e outro como receptor (Rx). Ambos os protótipos foram projetados para executar tarefas remotas por tecnologia LoRa, que permitem a adequação dos sistemas às necessidades de aplicações variadas, com ajustes na programação e incremento do hardware.

Para simular condições reais de monitoramento ambiental com geração de pacotes de dados a serem transmitidos e testar a comunicação entre o microcontrolador e os sensores foi desenvolvida uma camada de percepção, composta por um conjunto de sensores e, para auxiliar os testes *in situ*, um módulo GPS. Para a alimentação do sistema de monitoramento foi ainda confeccionado um componente de alimentação fotovoltaico.

3.1.1. Protótipo 1 – WiFi LoRa 32

O primeiro protótipo foi baseado no Módulo WiFi LoRa da Heltec Automation que integra o microcontrolador ESP32 e o chip LoRa SX1276 da SEMTECH com frequência central de 915 MHz. Este módulo também possui Wi-Fi e Bluetooth integrado, e tela OLED de 128x64 pontos de 0,96 polegadas. Acompanha um cabo com conector Pigtail – SMA e uma antena para frequência escolhida (HELTEC AUTOMATION, 2019) (Figura 9).

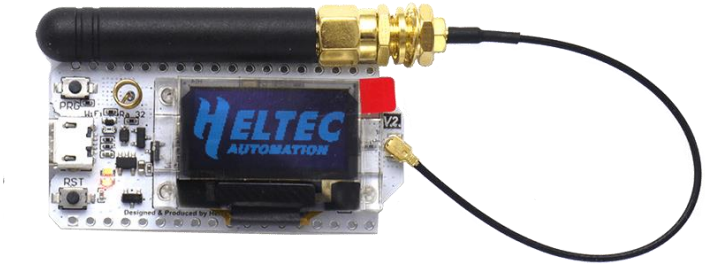

Figura 9 – Módulo WiFi LoRa 32 da Heltec Automation
FONTE: Heltec Automation (2019)

3.1.1.1. Microcontrolador SoC ESP32

A placa de desenvolvimento da Heltec possui design compacto pois seus componentes são projetados e integrados. O microcontrolador utilizado é o ESP32 com encapsulamento tipo SoC (*System-on-a-chip*), onde os demais componentes eletrônicos necessários para o funcionamento do microcontrolador são projetados e integrados na placa da WiFi LoRa 32 pela própria Heltec.

O ESP32 integra em um único chip combinado de Wi-Fi e Bluetooth de 2,4 GHz projetado com a tecnologia TSMC[3] *ultra-low-power* 40 nm, é capaz de funcionar de maneira confiável em ambientes industriais, com uma temperatura operacional que varia de -40 °C a + 125 °C. Alimentado por circuitos avançados de calibração, o ESP32 pode remover, dinamicamente, as imperfeições do circuito externo e se adaptar às mudanças nas condições externas (ESPRESSIF SYSTEMS, 2019).

3.1.1.2. Chip LoRa SX1276

O chip LoRa SX1276 é integrado na placa de desenvolvimento WiFi LoRa ESP 32 e fica localizado fisicamente na placa atras do display OLED. Suas trilhas e conexões já são conectados as GPIO do microcontrolados. As GPIO utilizadas são: 05, 14, 18, 19, 26, 27, 34 e 35.

Os transceptores SX1276 podem atingir sensibilidade acima de -148 dBm usando um cristal e lista de materiais de baixo custo, potência de + 20 dBm integrado, tornando-o ideal para qualquer aplicação que requeira alcance ou robustez.

[3] Taiwan Semiconductor Manufacturing Company

3.1.2. Protótipo 2 – DOIT DevKit V1 e Ai-Thinker Ra-02

O segundo protótipo foi baseado na placa de desenvolvimento DOIT ESP32 DevKit V1 (Figura 10) que possui o ESP32 WROOM integrado como componente principal. Contém 30 pinos, incluindo as entradas, saídas e alimentação. A placa possui uma saída de 3,3V para alimentação de circuitos externos e duas entradas de alimentação, uma entrada é o próprio conector micro-USB e a outra entrada para fontes externas é o pino "Vin[4]". Com um botão de *reset*, para reiniciar o micro e um botão *boot* interligado a GPIO_0 do microcontrolador.

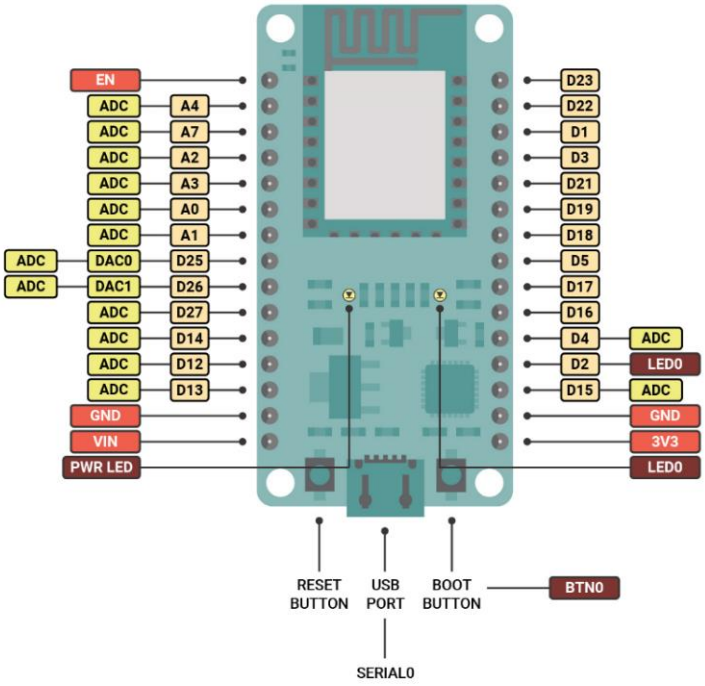

Figura 10 – Diagrama de pinos da placa ESP 32 DevKit V1.
FONTE: Adaptado de ZERYNTH (2020).

3.1.2.1. Microcontrolador ESP32 WROOM

A placa de desenvolvimento DOIT também utiliza como microcontrolador da série ESP32, porém o chip não é integrado direto na placa de circuito impresso (*Printed Circuit Board* - PCB). O microcontrolador compõe um módulo chamado ESP32 WROOM. O módulo ESP32 WROOM já possui todos os componentes eletrônico necessários para o funcionamento do microcontrolador. O módulo possui uma blindagem em torno dos componentes e possui uma antena em trilha de PCB.

[4] Devido à queda de tensão do regulador de tensão utilizado na placa a tensão de entrada de ser no mínimo de 5V e máxima de 12V recomendado por (ZERYNTH, 2020) para evitar aquecimento causado pelo regulador se a tensão for superior a 12V.

A PCB do módulo ESP32 WROOM é soldada na superfície superior da PCB da placa de desenvolvimento DOIT ESP32 DevKit, responsável pela disposição das GPIO, interface de comunicação e fonte de alimentação.

A Tabela 3 apresenta um resumo das especificações do ESP32 SERIES, onde as informações são válidas tanto para a placa de desenvolvimento da Heltec quanto para DOIT. Demais características podem ser observadas no *datasheet* da ESPRESSIF (ESPRESSIF - ESP32 SERIES DATASHEET, 2019).

Tabela 3 – Resumo das especificações do ESP32.

1 MCU	Xtensa® dual-core 32-bit LX6 microprocessador
448 kB	Memória flash ROM
520 kB	SRAM
16 kB	SRAM para o RTC interno
Wi-Fi: 802.11 b/g/n	Conexão sem fio
Bluetooth: v4.2 BR/EDR e BLE	Conexão sem fio
8 canais	SAR ADC até 18 canais[5]
2 canais	8-bit DAC
10 entradas	Sensor de Toque (Detecção Capacitiva)
4 x SPI	Serial Peripheral Interface
2 x I2C	Comunicação de periféricos usando mesmo barramento
2 x I2S	Comunicação interna para sistemas de áudio
3 x UART	Comunicação Serial
1 host	Controladores de host SD/SDIO/CE-ATA/MMC/eMMC
1 controlador escravo	Controlador escravo SDIO/SPI
1 Interface Ethernet MAC	Interface Ethernet MAC da com DMA dedicado e Suporte Protocolo IEEE 1588
1 Barramento CAN	CAN 2.0
1 PWM	PWM Motor
16 canais	PWM
Função Sleep	Ligar através de interrupções no GPIO, timer, através de medições no ADC ou pelo sensor de interrupção de toque capacitivo.

FONTE: Retirado de ESPRESSIF - ESP32 SERIES DATASHEET (2019)

3.1.2.2. Módulo LoRa Ra-02

[5] Durante testes constatou-se a não linearidade do conversor ADC, sendo necessárias à aplicação de correção feita pelo usuário ou fornecida pela própria Espressif em < https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html#adc-calibration >.

Para o segundo protótipo fez-se necessária a conexão dos terminais da placa de desenvolvimento DOIT ESP32 DevKit com os terminais correspondentes do módulo Ra-02 da Ai-Thinker.

O módulo utilizado é fornecido pela empresa Ai-Thinker (2017) (Figura 11), tecnologia que utiliza o chip SX1278 da SEMTECH , possui uma alta sensibilidade de mais de -148dBm, uma potência de +20dBm, consumo de transmissão de 93mA, recepção de 12,15mA, consumo *Standby* 1,5mA e alta confiabilidade (AI-THINKER TECHNOLOGY CO., LTD, 2017).

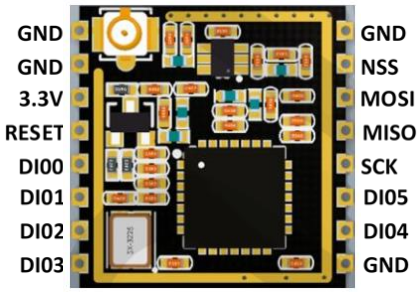

Figura 11 – Pinagem do módulo LoRa Ra-02.
FONTE: Adaptado de Ai-Thinker Technology CO., LTD (2017).

A frequência de operação usada foi de 433MHz com potência de Transmissão de 18±1 dBm com isso uma variação de 60 mW a 100 mW. A comunicação entre o ESP32 e o módulo LoRa Ra-02 é feita por SPI, cuja sensibilidade de recepção de dados é de -141 dBm. Possui uma taxa de bits de até 300 kbps para pacotes enviado com até 256 Bytes e inclui um método de detecção de erros (CRC) usado em redes digitais.

A saída para antena é dada pelo conetor Jack IPEX UFL para SMA Fêmea, possibilitando a ligação externa com a antena. Pode ser ligado também a uma antena dimensionada na própria placa de circuito impresso. A sua largura de banda padrão é de 125 kHz, entretanto pode ser ajustada.

3.1.3. Componentes Auxiliares

Estes componentes podem ser ligados em ambos os protótipos, desde que sejam feitos os devidos ajuste para a programação e as portas GPIO correspondentes.

3.1.3.1. Armazenamento de Dados

O módulo de cartão de memória utiliza cartões micro SD, sua comunicação com o micro é feita por SPI (Figura 12). Contém também um regulador de tensão para 3,3V e uma interface de ajuste para a comunicação de nível lógico diferente.

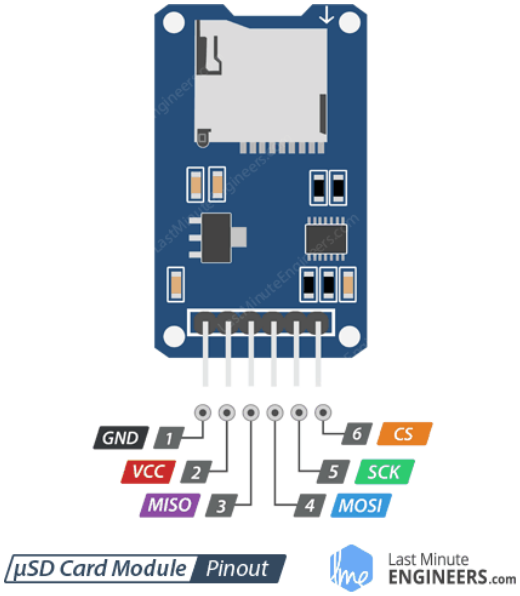

Figura 12 – Pinout módulo micro SD.
FONTE: Last Minute ENGINEERS (2020)

3.1.3.2. Calendário e Relógio (DS3231)

O RTC externo utilizado é o DS3231 (Figura 13). Trata-se de um relógio de tempo real de alta precisão e baixo consumo de energia. Possui um sensor de temperatura integrado a sua placa e um cristal oscilador para melhorar sua exatidão não contagem do tempo. Este módulo DS3231 é capaz de fornecer informações como segundo, minutos, horas, dia, mês e ano. Correções como meses com menos de 31 dias e anos bissextos são corrigidos automaticamente e pode operar tanto no formato 12 horas como 24 horas (MAXIM INTEGRATED, 2015).

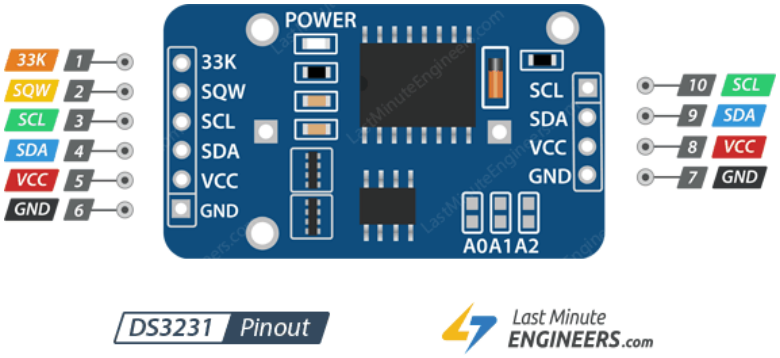

Figura 13 – Pinout módulo DS3231
FONTE: Last Minute ENGINEERS (2020)

3.1.4. Camada de Percepção

Para o segundo protótipo foi confeccionada uma completa camada de percepção. O Presente estudo visou primordialmente avaliar a funcionalidade da camada de rede implementada pela tecnologia LoRa para aplicações típicas do monitoramento ambiental. Para este fim foi configurada uma camada de percepção composta por sensores para monitoramento de temperatura e umidade, porém sem aferimento específico da acurácia das medições obtidas pelos mesmos.

3.1.4.1. Termohigrômetro (Dht22)

Nos sensores de temperatura e umidade relativa do ar, foi utilizado o termohigrômetro AM2302 (Figura 14) produzido pela empresa Aosong Electronics. Conhecido também como DHT22, trata-se de um módulo digital de temperatura e umidade com a detecção capacitiva para umidade. Este modulo é calibrado de fábrica com uma saída de sinal digital com alta confiabilidade, estabilidade e precisão com um microcontrolador de 8 bits integrado (AOSONG, 2020).

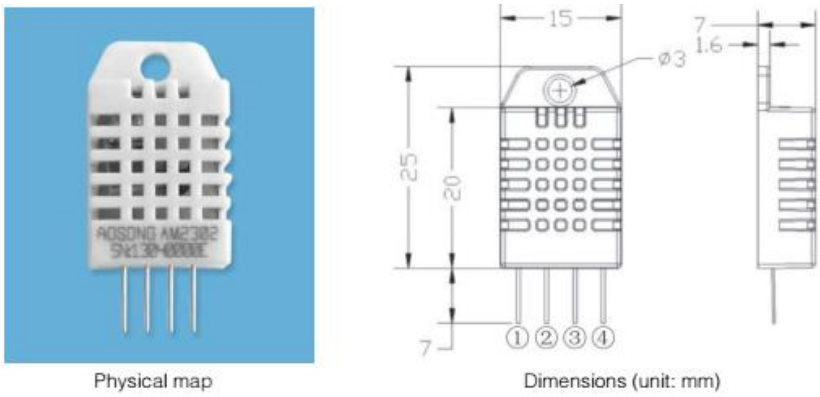

Figura 14 – Sensor de temperatura e humidade relativa do ar AM2302.
FONTE: Aosong (2020)

A resolução de medida do termohigrômetro é de 1/10, com precisão de ±0.5% de temperatura e ±2% umidade relativa. Cada medição deve respeitar no mínimo 2 segundos de intervalo para evitar erros de comunicação ou perda dos dados.

3.1.4.1.1. Estrutura e Abrigo dos Sensores

A estrutura onde serão fixadas as células solares, o case de proteção dos circuitos, o case de bateria e os sensores é formada por tubos de alumínio de ¼" dobrados em 45º e ficados a haste tubular central de ¼" com abraçadeiras em carbono zincado com rosca sem fim.

Para proteger o sensor termohigrômetro contra a exposição ao tempo e a radiação solar direta, foram confeccionados abrigos de proteção constituído de seis pratos material tipo melamina (Figura 15), sobrepostos com espaçamento de 1 cm entre eles, uma abertura central para o alojamento do sensor, 01 prensa cabo na parte inferior para fixação do sensor, tornando possível a integridade do sensor.

Figura 15 – Detalhe da confecção do abrigo para o termohigrômetro.
FONTE: Morais (2021)

3.1.4.2. Termorresistor (PT100)

O sensor utilizado para medir a temperatura do solo é do tipo termorrestivo, muito usado industrialmente, conhecido como PT100 (Figura 16), seu princípio de medição é verificar a variação da resistência elétrica em função da temperatura, utilizando o coeficiente da temperatura da platina pura que é 0,385 Ohm/K, de acordo com IEC 751 (DIN 43760). Quando a temperatura atinge o valor de °0 C o valor de resistência nominal do PT100 é de 100 Ohms. Sua precisão é de 0,03ºC com excelente estabilidade (ELECTRON TECNOLOGIA DIGITAL, 2014).

Figura 16 – Imagem ilustrativa PT100.
FONTE: (NOVUS PRODUTOS ELETRÔNICOS, 2020)

Os termorresistores são confeccionados em corpo cerâmico de alta alumina contendo capilares onde é inserido o filamento de platina, preenchidos com finíssimo pó cerâmico mantendo o filamento livre de vibrações sem causar “stress” mecânico em altas temperaturas. O cabeçote em alumínio injetado (Ip65), bulbo e haste em aço inoxidável, contribui para a utilização em contato com o solo, protegido a exposição ao tempo (NOVUS PRODUTOS ELETRÔNICOS, 2020).

3.1.4.2.1. Conversor Analógico Digital (ADS1115)

No caso do ESP32 não foi utilizado o seu conversor ADC (Conversor Analógico Digital) integrado ao chip, devido a sua não linearidade. Para tal a sua fabricante Espressif Systems Co, fornece uma API para calibração e configuração do ADC do ESP32. Esta API fornece funções para corrigir diferenças nas tensões medidas causadas pela variação das tensões de referência ADC (Vref) entre os chips (ESPRESSIF, 2020).

Para garantir a precisão nas leituras foi adotado um conversor ADC externo, o ADS1115, que possui resolução de 16 bits a 860 S/s (amostras/segundo) com 4

canais e um amplificador interno para ganho do sinal. O protocolo de comunicação utilizada é o I2C e é compatível com a tensão do sistema implantado. O ADS1115 é utilizado para a leitura do sensor termorresistivo.

3.1.4.3. Módulo GPS (NEO-6M)

Para os testes de deslocamento foram incluídos módulos GPS (*Global Positioning System*) em cada protótipo a fim de avaliar a posição do transmissor e do receptor o modelo utilizado foi o GY-GPS6MV2. Sua interface serial é de 3,3V e não tolerante a 5V. Tensão de alimentação: 3V-5V. Com a bateria para backup de dados.

Uma antena externa com conector SMA e uma base magnética foi conectada ao GPS, com frequência central de 1575,42 MHz, ganho de 28 dB, fator de ruído 1,5 dB e tensão nominal de 3 a 5,5V. A base magnética facilitou a ficção da antena durante os testes.

3.1.5. Fonte Energética

O sistema pode ser alimentado de duas maneiras, através do conector micro-USB do módulo ESP32, ou através da ligação direta no pino Vin do módulo ESP32. O regulador de tensão integrado no módulo reduz a tensão para a tensão nominal de 3,3 V do circuito. O esquema do arranjo de alimentação do sistema é mostrado na Figura 17.

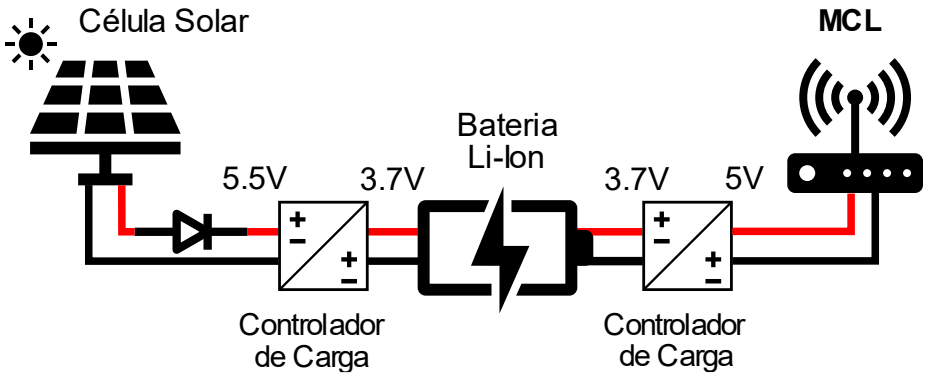

Figura 17 – Esquema de ligação do sistema de alimentação
FONTE: Morais (2021)

3.1.5.1. Células Fotovoltaicas

Para a recarga das baterias foi utilizado um conjunto de células solares com potência de pico de 6 W e tensão nominal de 6 V. As duas células são encapsuladas

em resina epóxi de alta transparência e placa PCB em fibra de vidro como rodapé. Esse conjunto é encontrado no mercado como minicélula solar, com um valor de US$ 15.

3.1.5.2. Baterias de Lítio

Para autonomia do MCL utilizou-se uma bateria de lítio (Li-Ion) de 10000 mAh tensão de 3,7V, reaproveitada de um Power Bank portátil.

3.1.5.3. Controlador de Carga

No controlador de carga da bateria e do sistema foram utilizadas duas placas de proteção de carregamento modelo 134N3P, fabricante Si Tai&SH. Uma placa foi utilizada para efetuar a recarga da bateria e outra para converter 3,7V para 5V mantendo o sistema alimentado. O controlador de carga para baterias de Li-Ion evita sobretensão e eventuais aquecimentos por sobrecorrente. O custo de cada placa controladora de carga foi de US$ 0,5.

Antes de ligar a saída de tensão gerada pela célula solar ao controlador de tensão, foi inserido um diodo em série para auxiliar na queda de tensão reduzindo para 5,4V e evitando fluxo contrário de carga.

3.1.6. Esquema de Conexão dos Módulos com a Placa de Desenvolvimento

O esquema de conexão eletrônica dos módulos com a placa de desenvolvimento DOIT é apresentado na Figura 18.

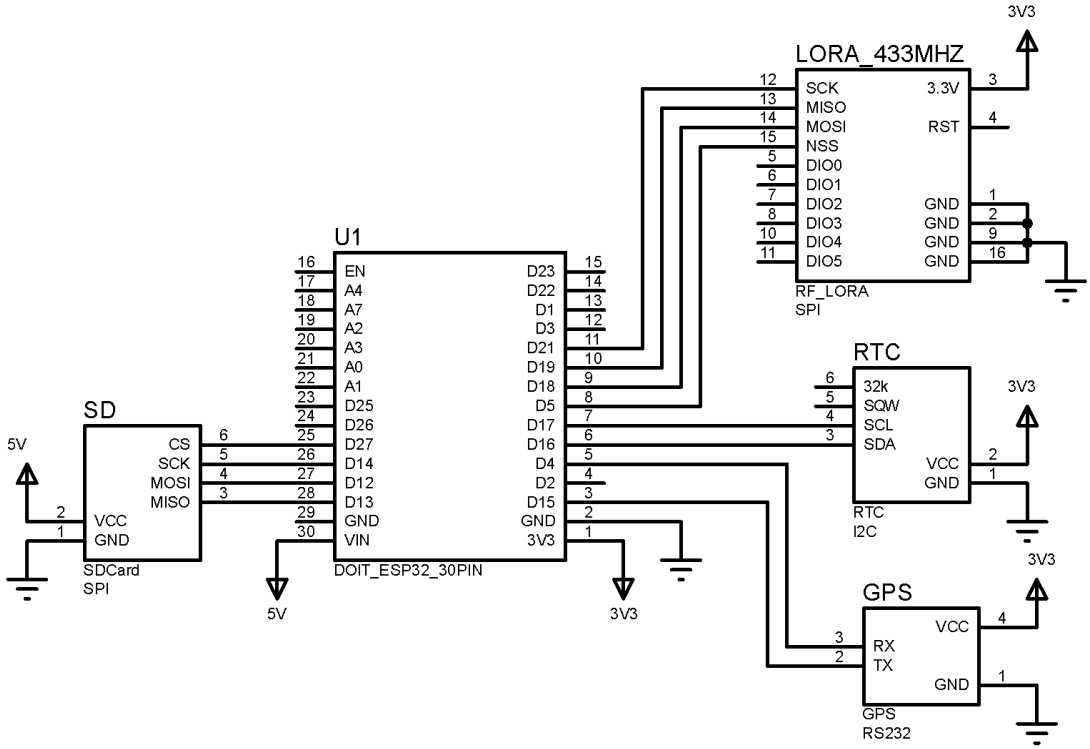

Figura 18 – Esquema de ligação eletrônica do protótipo.
FONTE: Morais (2021)

O esquema de ligação dos sensores no protótipo, é apresentado na Figura 19.

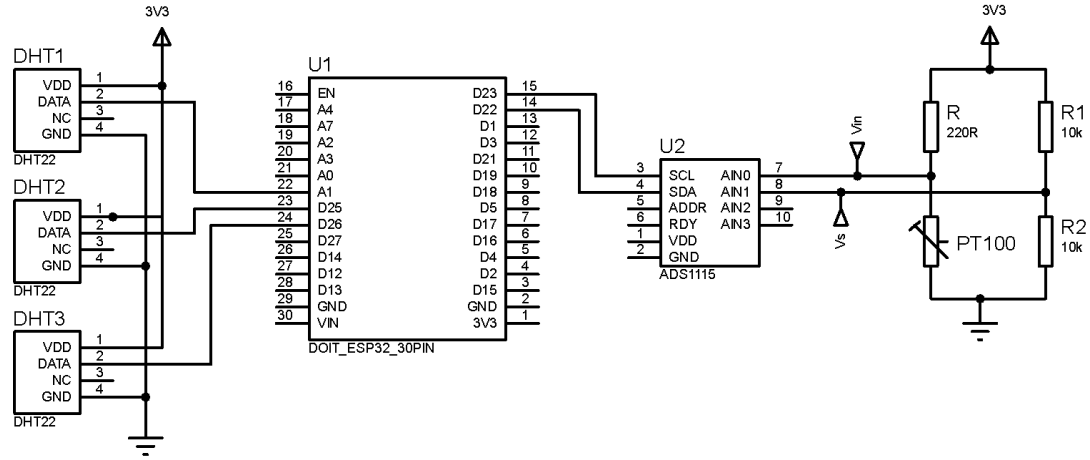

Figura 19 – Esquema eletrônico de ligação dos sensores.
FONTE: Morais (2021)

A conexão do sensor termohigrômetro (Dht22) foi feita por 3 vias, sendo uma via de comunicação com interface de barramento único e duas vias para alimentação elétrica do sensor, suportando uma distância de transmissão de até 20 metros.

Para leitura do sensor PT100 utilizou-se um ADC externo ADS1115. O valor da temperatura foi obtido através da interpretação de dois sinais obtidos pelo ADS1115

e enviado ao microcontrolador para processá-los. Um dos valores é a tensão do sistema (V_s) e o outro é a tensão do divisor de tensão do conjunto resistor de precisão em série com o PT100 (V_{in}).

3.1.7. Programação da Placa de Desenvolvimento

Na programação do microcontrolador utilizou a interface de desenvolvimento *Integrated Development Environment* (IDE) do Arduino. Para que a IDE reconheça o microcontrolador são necessários os *drivers* do *hardware* fornecidos pela Espressif. São disponibilizados no Github.com, com os passos para a instalação (ESPRESSIF SYSTEMS, 2020).

No projeto utilizou-se uma biblioteca para o RTC externo denominada *"RtcDS3231.h"* desenvolvida por MILLER (2019) instalado pela interface do compilador do Arduino, juntamente com mais duas bibliotecas de manipulação do tempo, para integração com o RTC interno do ESP32 a biblioteca *"time.h"* e a *"systime.h"*. Para a comunicação do RTC externo com o microcontrolador foi utilizada a biblioteca *"Wire.h"* responsável pela comunicação I2C, em que um barramento é responsável pelos dados e outro a sincronização do *clock* do microcontrolador.

Na comunicação com o módulo de microSD foi necessária a inclusão da biblioteca *"SPI.h"* responsável pela comunicação que utiliza um protocolo de dados seriais síncronos usado pelos microcontroladores para se comunicar com um ou mais dispositivos periféricos rapidamente em curtas distâncias. A biblioteca *"SD.h"*, responsável pelos comandos de escrita e leitura do cartão de memória com auxílio da biblioteca *"FS.h"* que permite usar o sistema de arquivos para armazenar dados de esboço, arquivos de configuração (ESP8266 ARDUINO CORE, 2017).

O módulo LoRa utiliza o protocolo de comunicação do tipo SPI, logo a biblioteca *"SPI.h"* é requisitada novamente, sendo necessário a indicação dos pinos para diferenciá-los do módulo de microSD. Para os comandos de transmissão e recepção e detalhes da conexão foi incluído a biblioteca *"LoRa.h"* desenvolvida por Aaron.Lee (2018) da Heltec Automation.

Ao utilizar a comunicação sem fio Wi-Fi: 802.11 b/g/n integrada ao microcontrolador fez-se uso da biblioteca *"Wi-Fi.h"* ofertada pela Espressif Systems.

3.2. Campanhas de Campo

Durante a evolução do projeto foram realizados cinco tipos de teste em campo afim de aferir a estabilidade da tecnologia fora do ambiente laboratorial. Os testes visaram avaliar:

i) o funcionamento da camada de percepção, i.e. a comunicação dos sensores escolhidos para validação via microcontrolador do módulo LoRa Heltec (3.2.1);

ii) a integridade da camada de rede, i.e., a comunicação e alcance entre dois módulos LoRa Heltec (3.2.2);

iii) a integração das camadas de percepção e rede, i.e., a aquisição e armazenamento dos dados medidos pelos sensores e sua correta transmissão entre dois módulos LoRa Heltec (3.2.3);

iv) o alcance da camada de rede, i.e., a distância máxima da transmissão dos dados entre dois módulos LoRa (DOIT) em ambiente rural / urbano misto sem módulos GPS acoplados (3.2.4) e;

v) o alcance da camada de rede em ambientes urbano e rural entre um módulo LoRa fixo e outro transportado em veículo (DOIT) com módulos GPS acoplados (3.2.5).

3.2.1. Teste 1: Funcionamento da Camada de Percepção

O primeiro teste da camada de percepção foi realizado no campo experimental da Unidade de Pesquisa e Centro Tecnológico do Instituto Mato-grossense de Algodão de Sorriso localizado na BR163, km 712 s/n - Mato Grosso, Brasil. Os testes visaram validar o funcionamento do microcontrolador, bem como a integração do mesmo com outros periféricos como sensores e módulo de armazenamento de dados (Figura 20).

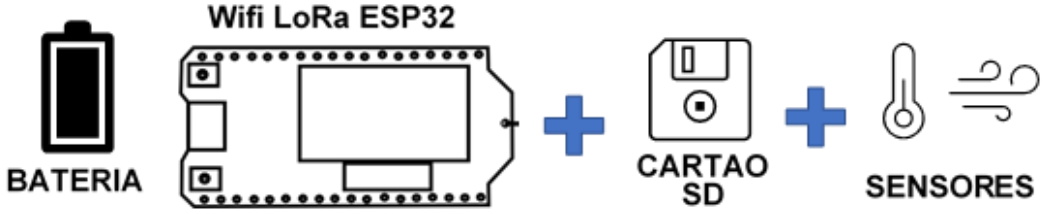

Figura 20 – Camada de percepção: sensores + armazenamento.
FONTE: Morais (2021)

Como preparativo foi programado um sensor termohigrômetro DHT11 para aferição de medições meteorológicas em uma plantação de algodão. Nos testes, os valores instantâneos foram verificados na tela OLED do módulo WiFi LoRa 32 da Heltec e salvos no cartão de memória.

3.2.2. Teste 2: Integridade da Camada de Rede

O teste da camada de rede utilizando a tecnologia LoRa foi realizado em uma região rural com relevo ondulado nas proximidades de duas Pequenas Centrais Hidroelétricas (PCH), a PCH José Gelázio da Rocha e a PCH Rondonópolis no município de Rondonópolis-MT, localizadas na Rodovia BR 163 - km 102, s/nº Ribeirão Ponte de Pedra.

Foram empregados dois módulos WiFi LoRa 32 da Heltec para transmissão ponto a ponto (Figura 21). A potência do rádio de 20 dBm, a frequência de 915 MHz. Utilizou-se a configuração padrão da biblioteca LoRa com o fator de espalhamento igual a onze (FS = 11). Além disso, foram usadas antenas externas modelo CM-907 fabricante AQUÁRIO, com um ganho de 5 a 7 dBi.

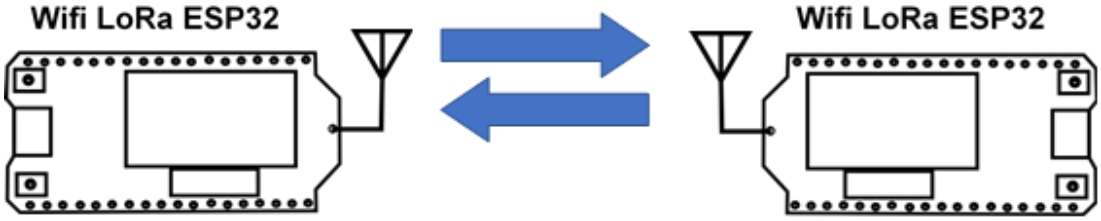

Figura 21 – Camada de Rede, comunicação ponto a ponto.
FONTE: Morais (2021)

Os módulos estavam programados para enviar/receber os valores de um contador implementado no microcontrolador, uma vez que era necessário reconhecer a máxima distância de recepção alcançada.

3.2.3. Teste 3: Integração das Camadas de Percepção e Rede

Esse teste avaliou a integração da camada de percepção com a camada de rede. Para a camada de percepção foi utilizado um sensor termohigrômetro interligado à placa de desenvolvimento WiFi LoRa 32 da Heltec (Transmissor), que é responsável pela comunicação na camada de rede com o outro dispositivo WiFi LoRa 32 da Heltec (Receptor). O dispositivo receptor por sua vez é responsável por armazenar os dados recebidos em um cartão de memória conforme o esquema apresentado na Figura 22. O teste foi realizado no Núcleo Avançado do Pantanal do Instituto Federal de Mato Grosso em Poconé (NAP IFMT - 102 km de Cuiabá)

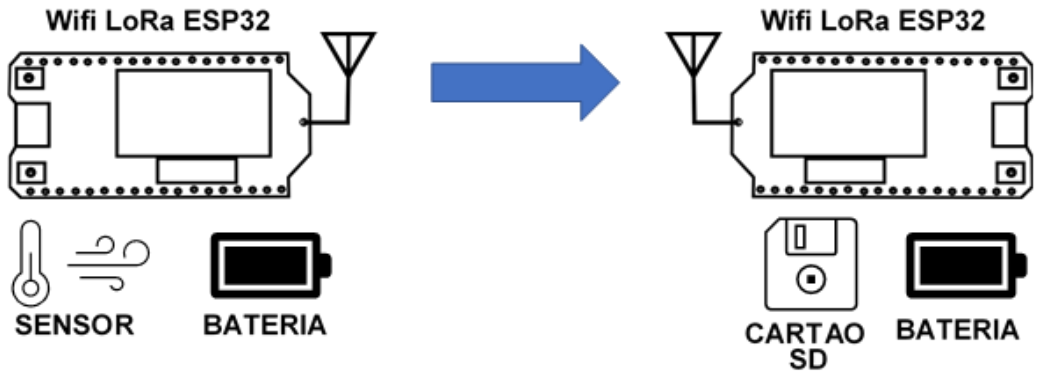

Figura 22 – Integração da camada de percepção com a camada de rede.
FONTE: Morais (2021)

O protótipo foi montado em *protoboards* e ligados em bancos de bateria, a frequência de transmissão era de 915 MHz, FS = 11. Foram utilizadas as antenas de fábrica que acompanham a o módulo WiFi LoRa 32.

3.2.4. Teste 4: Alcance da Camada de Rede (DOIT/LoRa)

O teste de alcance da camada de rede foi realizado na Fazenda Experimental da UFMT, Localizada em Santo Antônio do Leverger (36 Km de Cuiabá). O teste é baseado em transmitir os dados dos sensores medição *in situ* e verificar o máximo alcance de transmissão utilizando um receptor móvel.

Os protótipos possuíam as seguintes configurações: Placa de desenvolvimento DOIT ESP32 DevKit V1; módulo LoRa da Ai-Thinker com o chip SX1278 da SEMTECH com potência de +20 dBm; módulo RTC; módulo SD Card, duas antenas omnidirecional de 433 Mhz modelo YNX-433-A002, abrigados em case de PVC para proteção. No transmissor foram acoplados os sensores com suporte para os sensores termohigrômetro e respectivos abrigos conforme a Figura 23.

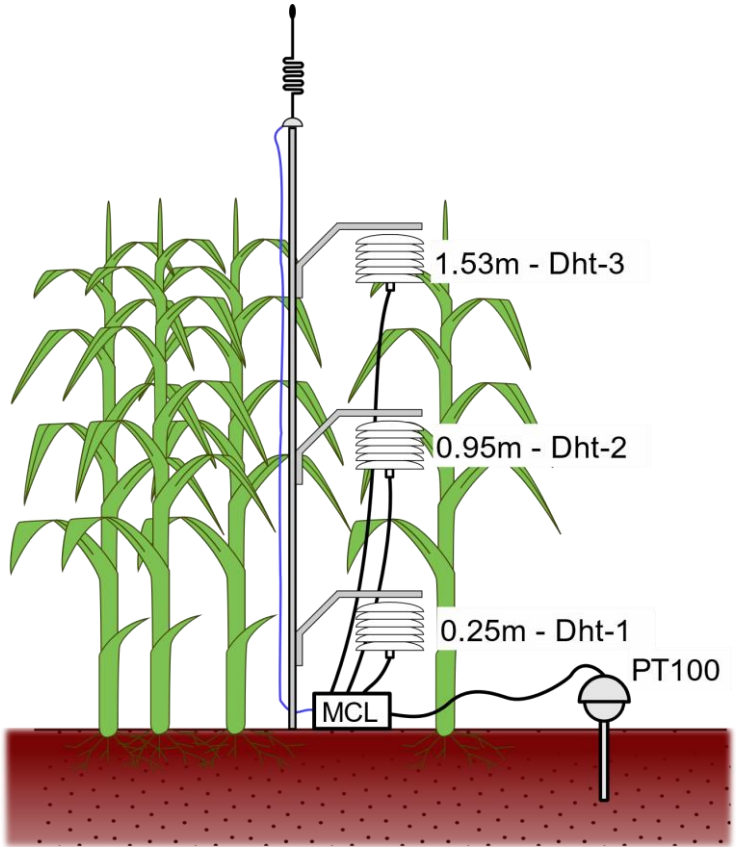

Figura 23 – Arranjo utilizado em teste com cultivo.
FONTE: Morais (2021)

O Módulo Controlador LoRa (MCL) utilizado como transmissor realiza a amostragem e envia os dados via LoRa para o Receptor. O receptor foi conectado ao laptop utilizado como interface visual entre protótipo e usuário, onde eram exibidos os dados recebidos. A antena do receptor foi fixada no teto do veículo por sua base magnética, sendo assim a qualidade e o alcance ficam em função da distância e característica do relevo entre a estação e o veículo com o receptor. Para coleta da respectiva localização do veículo utilizando o GPS (*Global Positioning System*) de celular.

3.2.5. Teste 5: Alcance da Camada de Rede em Ambientes Urbano e Rural entre Rádios LoRa Fixo e Outro Transportado em Veículo com Módulos GPS Acoplados (DOIT/LoRa)

Em uma segunda campanha para avaliação do alcance da camada de rede foram acoplados um módulo GPS interligado as ESP32, para realizar a captação da localização de forma autônoma com o transmissor fixo e o receptor em deslocamento.

A posição do transmissor (MCL-Tx) é enviada como pacote de dados, sendo assim o receptor (MCL-Rx) está programado para salvar sua posição no instante que recebe o pacote. Dessa forma, é possível verificar a distância do deslocamento.

Foi registrado o valor da intensidade do sinal recebido RSSI (*Received Signal Strength Indicator*), medido pelo próprio receptor, a quantidade de bytes recebidos, falhas nas mensagens enviadas nos pacotes de dados que eram enviados e os dados de localização do GPS acoplado ao sistema.

A antena utilizada no rádio LoRa foi a YNX-433-A0022 Omnidirectional com base magnética, frequência central de 433 MHz, ganho de 3~5 dBi. O modulo GPS utilizado foi o GY-GPS6MV2. No transmissor foi utilizado a antena de fábrica do módulo GPS. No receptor utilizou-se uma antena externa com ganho de 30±3 dBi, tensão de trabalho de 3 a 5 volts e frequência de 1575,42 MHz, com base magnética. A antena omnidirecional e a antena GPS externa foram posicionadas no teto do veículo o arranjo pode ser visto na Figura 24.

Figura 24 – Sistema LoRa Transmissor à esquerda e Sistema LoRa Receptor à direita.
FONTE: Morais (2021)

4. RESULTADOS E DISCUSSÕES

4.1. Avaliação da Funcionalidade dos Protótipo Implementados

A placa de desenvolvimento WiFi LoRa 32 da Heltec, utilizada no início do projeto foi a base dos testes da implementação da tecnologia LoRa, tanto para a programação, quanto para incremento de *hardware*.

Com os resultados não completamente satisfatórios obtidos na avaliação de alcance nos três primeiros testes, optou-se pela troca de *hardware* para um sistema modular com fabricantes distintos. Todos os detalhamentos a seguir se referem a este sistema, cujas diferenças com o protótipo 1 baseado no WiFi LoRa 32 da Heltec são expostos na Tabela 4.

Tabela 4 – Comparação entre Sistema Compacto (Heltec) e Sistema Modular (DOIT/Ai-Thinker).

Parâmetro	WiFi LoRa 32 da Heltec	DOIT ESP32 + Ai-Thinker LoRa
Corrente com micro programado como RX p/ Vin=5V	76.5 mA	85,7 mA
Tensão Nominal	3.3 V	3.3 V
Tipo	Compacto	Modular
Dimensões (mm)	26x51	(Doit) 28x52 (Ai-Thinker) 28x21
Área total da placa	1326 mm²	2044 mm²
Microprocessador	ESP32 SoC (Xtensa® dual-core 32-bit LX)	ESP32 WROOM (Xtensa® dual-core 32-bit LX)
Comunicação Wireless	Wi-Fi, Bluetooth	Wi-Fi, Bluetooth
Chip LoRa	SX1276	SX1278
Frequência Central	915 MHz	433 MHz
Potência	20 dBm	20 dBm
Tela OLED	128x64 0.96"	NA
Antena SMA	CM-907 G: 5 a 7 dBi	YNX-433-A0022 G: 3 a 5 dBi

FONTE: Morais (2021)

Na configuração adotada, uma placa de desenvolvimento DOIT ESP32 DevKit V1 foi interligada com o módulo LoRa da Ai-Thinker.

Para dispor nos testes da camada de rede de um sistema de monitoramento ambiental completo, foi confeccionada uma camada de percepção funcional incluindo sensores e estrutura para sua fixação. Os componentes eletrônicos foram alojados em case de PVC e fixados a uma estrutura de alumínio (Figura 25).

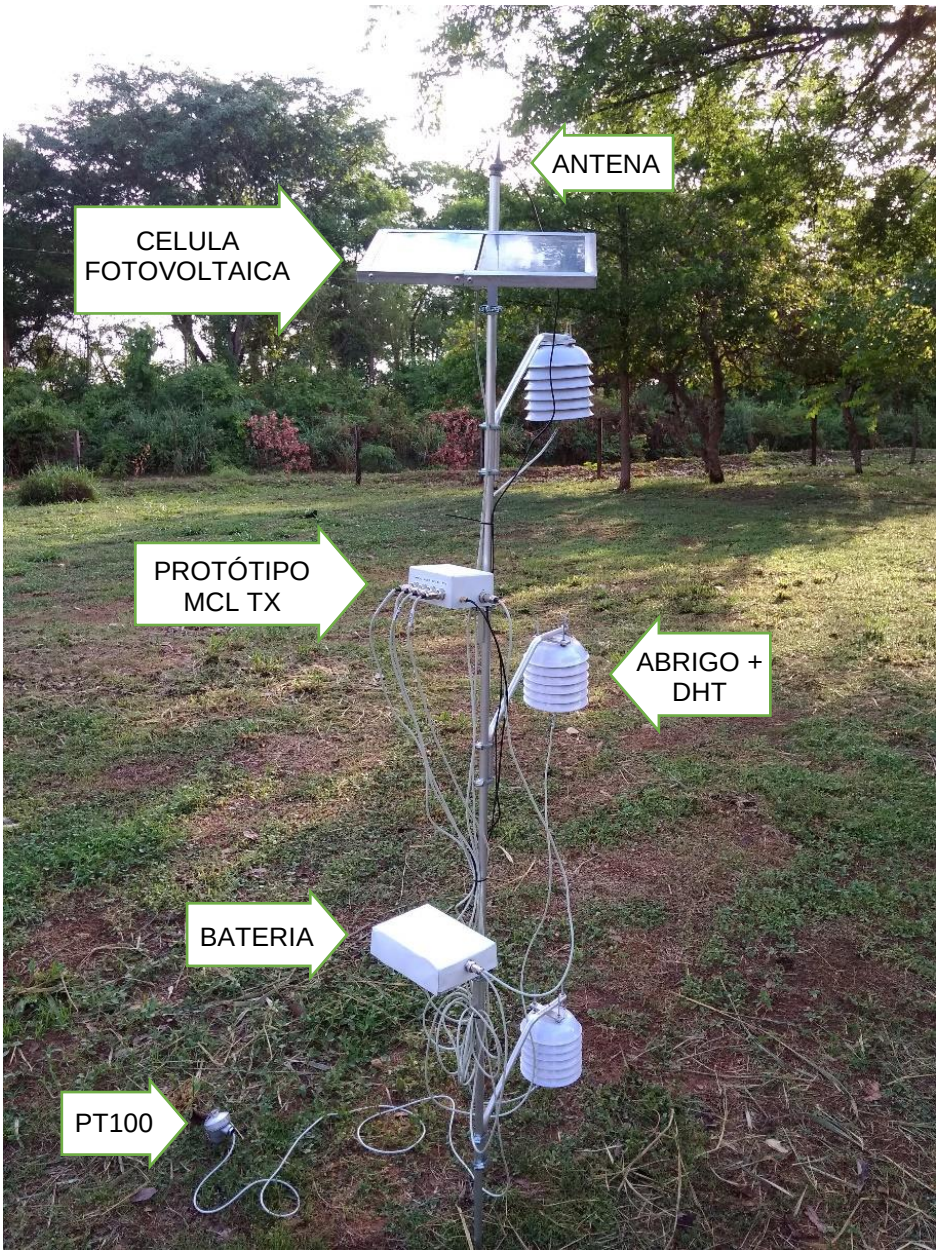

Figura 25 – Estrutura proposta para acoplamento dos componentes.
FONTE: Morais (2021)

O protótipo 2 na sua configuração final é denominado de Módulo Controlador LoRa (MCL), é constituído por uma configuração básica de *hardware* comum, tanto para o transmissor como para o receptor. A configuração envolve: o microcontrolador

(ESP32), o RF LoRa (SX1278), modulo de armazenamento (μSD) e o calendário externo (RTC), conforme a Figura 26.

Figura 26 – *Hardware* operacional básico.
FONTE: Morais (2021)

Pensando em possíveis perdas de comunicação o sistema foi composto por um cartão de memória micro SD, que independente da transmissão salva os dados lidos ou o último comando enviado. São salvos também os eventos e possíveis erros dos sistemas em um arquivo de texto separado identificado como “log.txt”. O calendário e relógio o RTC externo, é responsável pela sinalização do tempo dos eventos ocorridos.

O ESP32 com seus dois núcleos consegue realizar duas tarefas distintas ao mesmo tempo. No sistema desenvolvido, as prioridades nas tarefas foram: uma tarefa (rotina) fica responsável pela coleta dos dados, bem como salvá-los e a outra tarefa ficou responsável exclusivamente para a transmissão ou recepção dos dados.

4.1.1. Confecção do Sistema e Testes Funcionais

O MCL (protótipo 2) utiliza uma placa de fenolite cobreada e perfurada, para soldar os componentes que compõem o sistema (Figura 27). Suas ligações são feitas por condutores soldados na parte inferior da placa.

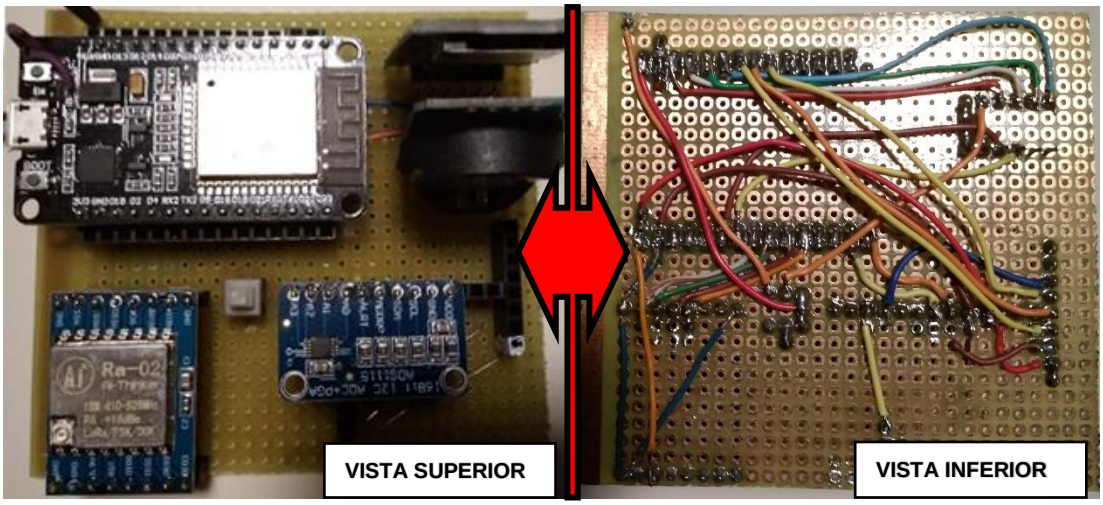

Figura 27 – Vista superior (à esquerda) e inferior (à direita) da montagem do MCL.
FONTE: Morais (2021)

Para a confecção do protótipo utilizando a configuração básica de *hardware*, foram gastos 38 US$, especificado na Tabela 5.

Tabela 5 – Custo dos componentes utilizados no protótipo 2 (MCL).

Tipo	Modelo	Valor	
Placa de Desenvolvimento	DOIT ESP32 DevKit V1	$	7.50
RF LoRa	SX1278 (Ai-Thinker Ra-02)	$	10.80
RTC	DS3231	$	5.40
SDCard	Catalex	$	2.70
Antena Externa	YNX-433-A0022	$	6.50
Case proteção	PVC PATOLA 12cmX8cm	$	5.50
	Total	**$**	**38.40**

FONTE: Morais (2021)

Equipamentos comerciais com a mesma finalidade de transmissão de dados, e seus respectivos valores são apresentados na Tabela 6.

Tabela 6 – Equipamentos comerciais utilizados na transmissão de dados.

Modelo	Consumo máx.	Alcance máx.	Taxa de Transferência	Valor	
INT700 ZB Serial SMA Adapter (Low Power)	50 mA; 12V; 600 mW	120 m	250 Kbps	$	178.00

Ebyte® E32-DTU-170L30 SX1278 170 MHz	624 mA; 12V; 7488 mW	8000 m	2.4 Kbps	$	51.00
WiFi LoRa 32 (V2) Heltec Automation	130 mA; 5V; 650 mW	4000 m	---	$	26.00
Protótipo Módulo Controlador LoRa	155 mA; 5V; 775 mW	4000 m	300 Kbps	$	38.40

FONTE: Morais (2021)

Apesar do valor de aquisição não ser expressivo, o protótipo apresenta ainda funcionalidades estendidas em relação aos equipamentos comerciais, como: a possibilidade de salvar os dados em um cartão de memória, um calendário para registrar data e a hora de possíveis eventos e possibilidade de comunicação Wi-Fi, além da interface serial padrão ofertado pelos equipamentos comerciais.

Uma das características de um sistema IoT é o baixo consumo de energia. A Tabela 7 resume o consumo nos principais modos de operação do protótipo.

Tabela 7 – Consumo de energia do protótipo, para vários tipos de arranjos.

Arranjo	Modo	Potência
Hardware básico	Ligado	$500\ mW$
Hardware básico	Transmitindo (TX)	$775\ mW$
Hardware básico	Recebendo (RX)	$525\ mW$
Hardware básico + GPS	Leitura do GPS + TX	$950\ mW$
Hardware básico + 3xDht22 + PT100	Leitura dos sensores + TX	$820\ mW$
Hardware básico + 3xDht22 + PT100 + GPS	Leitura dos sensores + TX	$1000\ mW$

FONTE: Morais (2021)

4.1.2. Adequação das Bibliotecas de Programação e Soluções Adotadas

Por ser um sistema que reúne vários componentes em um único protótipo e o compilador Arduino já possuir algumas bibliotecas nativas, quando foi instalado pacote de bibliotecas e de reconhecimento de hardware fornecido pela Espressif Systems, houve uma série de duplicidade de bibliotecas e várias incompatibilidades entre módulos que utilizavam a mesma interface de comunicação.

Sendo assim, foram restados todos os componentes individualmente, verificando as funções características de cada biblioteca. Após essa verificação eram sequencialmente acrescidos os módulos, verificando possíveis incompatibilidades. Os erros mais frequentes eram duplicidade de bibliotecas. Estes conflitos foram resolvidos a partir da análise dos arquivos em questão, verificando se no cabeçalho ("NomeDoArquivo.h" característico pela extensão ".h") da referida biblioteca já havia a inclusão de bibliotecas anteriores usadas. Eventuais duplicações foram removidas manualmente. Outro tipo de erro ocorreu na configuração de periféricos com a mesma interface de comunicação (ex.: SPI e I2C), sendo necessário o ajuste do endereço do barramento ou escolha de GPIO exclusivas para os módulos em conflito.

A última versão operacional testada do código, com as respectivas bibliotecas utilizadas foi disponibilizada em um repositório no GitHub (https://github.com/pauloeng28) Morais (2021). Foi criada também uma biblioteca para leitura dos dados dos sensores termohigrômetro, baseada na biblioteca do Winlin (2016-2017) e no datasheet do AOSONG (2020).

Quando ligado à bateria, o sistema aplica as configurações iniciais do ESP32, testa a comunicação com o cartão de memória, configura a data e a hora do sistema e inicia o rádio LoRa. Um fluxograma com a lógica de programação para ambos os MCL é apresentado na Figura 28.

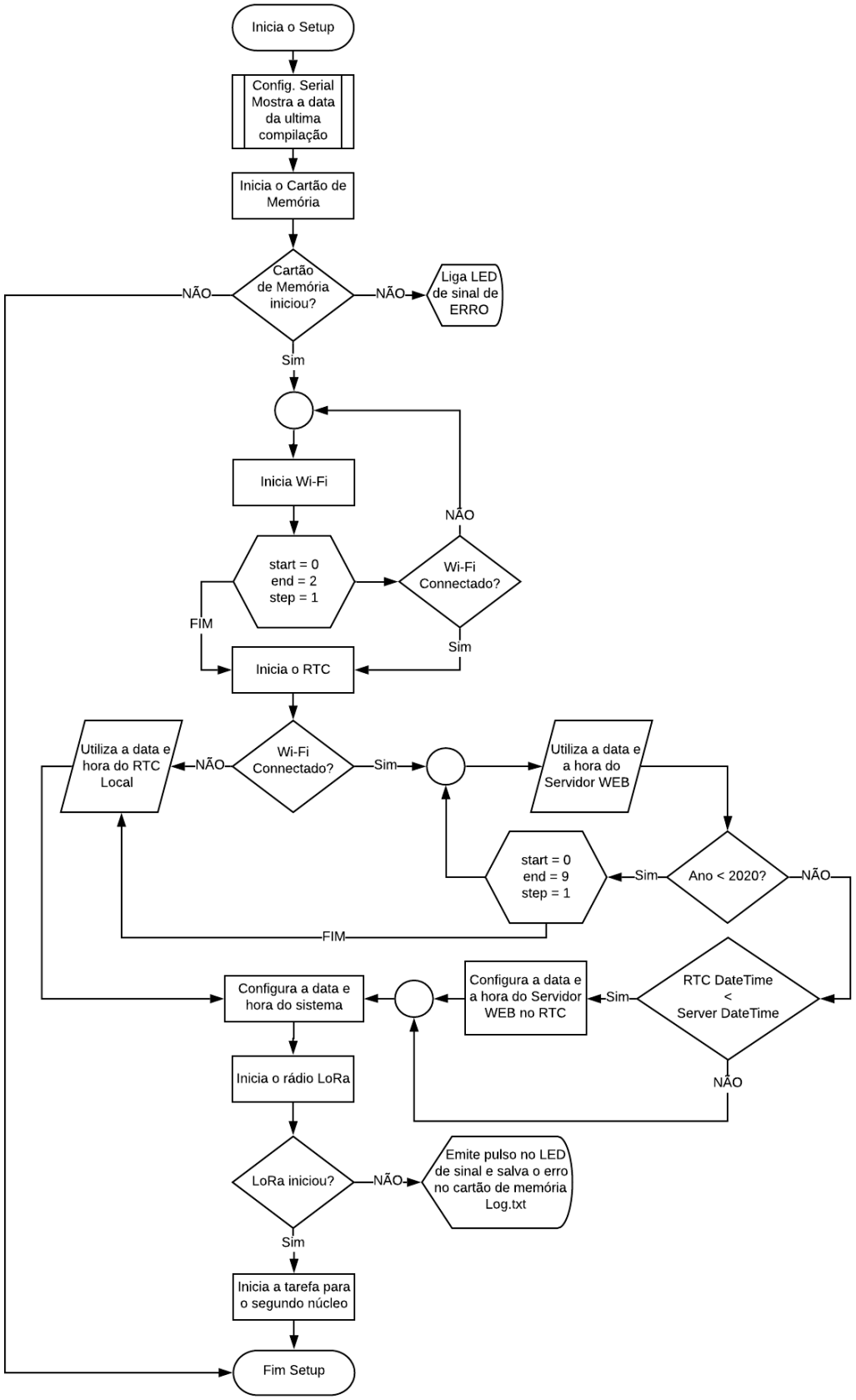

Figura 28 – Lógica de configuração inicial do sistema.
FONTE: Morais (2021)

O ESP32 possui RTC interno, porém se houver corte no fornecimento de energia, ou pane no sistema, sua contagem é perdida e a data-hora deve ser programada novamente. Portanto, todas as vezes que o ESP32 é reiniciado, executa-se a aquisição da data-hora por dois modos: no primeiro modo são realizadas três tentativas de conexão com a internet para aquisição da data-hora por um servidor Web. Se o servidor web estiver indisponível ou não houver conexão com a internet o segundo modo é acionado e o ESP32 buscará a data-hora em um RTC externo, previamente programado com data-hora atual.

Durante os trabalhos de campo, com aquisição de dados medidos pelo PT100 foram observadas fortes oscilações nos valores de temperatura gerados pelo ADC. Para filtragem destes valores foi criada uma rotina, pela qual o microcontrolador seleciona os valores de resistência entre 90 Ω que equivale aproximadamente -24º C e 140 Ω que equivale aproximadamente 104º C (NOVUS PRODUTOS ELETRÔNICOS, 2020).

Salienta-se que no fluxograma (Figura 28) não está inclusa a rotina de inicialização dos sensores, nem suas rotinas de tratamento. O fluxograma também não apresenta a rotina de transmissão/recepção LoRa e a rotina de armazenamento de dados. O código da programação completo pode ser consultado em Morais (2021) ou nos anexos: ANEXO A, ANEXO B e ANEXO C.

4.2. Desempenho do Sistema LoRa nas Campanhas de Campo

4.2.1. Teste 1: Funcionamento da Camada de Percepção

A placa de desenvolvimento WiFi LoRa 32 da Heltec realizou com êxito as aquisições do sensor termohigrômetro em campo. O sistema ficou instalado por 24 horas, neste período de teste não houve erros durante a sua leitura. Os dados foram armazenados no cartão de memória com sucesso.

4.2.2. Teste 2: Integridade da Camada de Rede

Neste teste da camada de rede implementada utilizou-se a placa de desenvolvimento WiFi LoRa 32 da Heltec. Foi usada uma antena externa omnidirecional sem visada direta. Com essa configuração foi obtido um alcance máximo de 1,4 km (Figura 29) entre o receptor PCH_RX2 e o transmissor, localizado próximo a entrada da área de influência (PCH_TX).

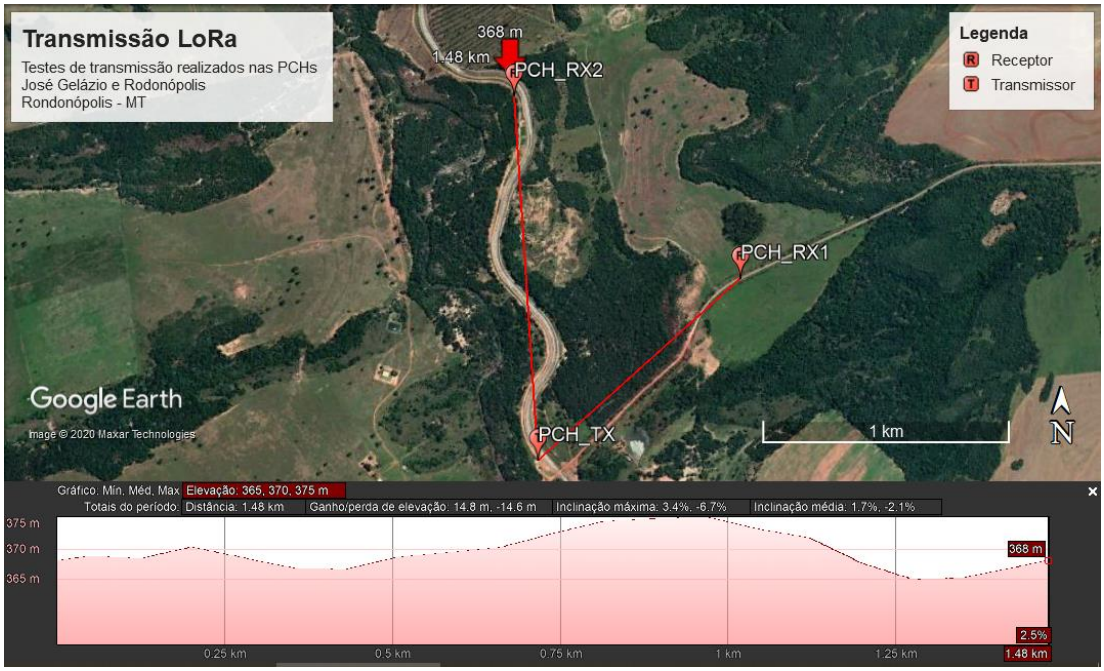

Figura 29 – Transmissão da entrada da PCH até a estrada interna da PCH.
FONTE: Adaptado de GOOGLE EARTH PRO (2020)

Em um segundo ensaio com o receptor (PCH_RX1) instalado em área externa da PCH, o máximo alcance foi de 1,1 km entre os rádios (Figura 30).

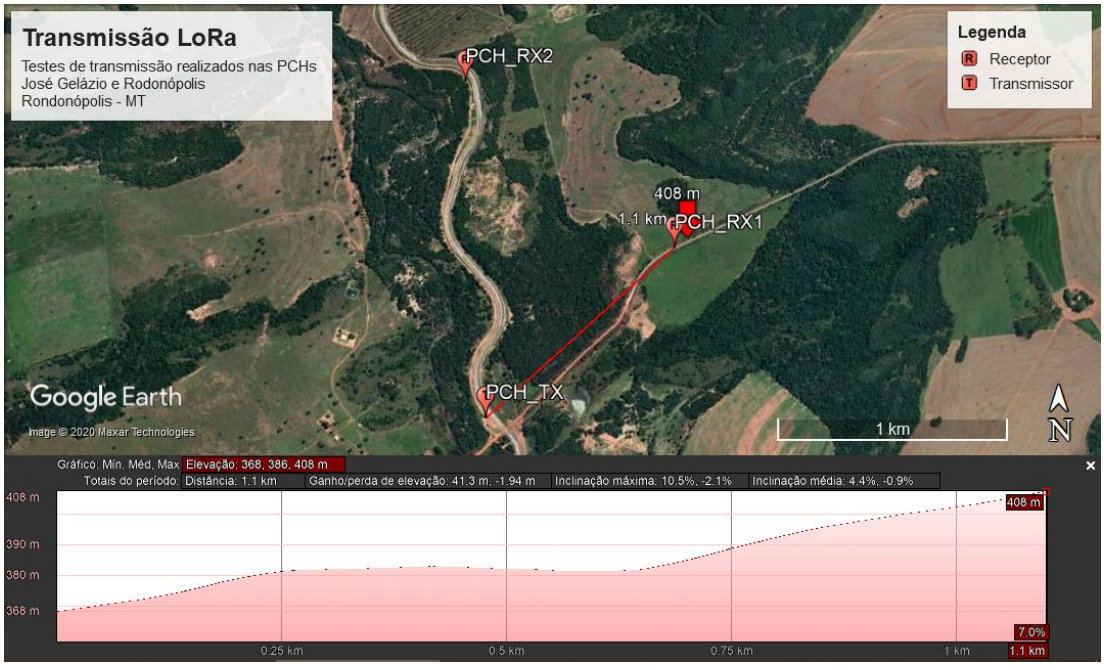

Figura 30 – Transmissão da entrada da PCH até a MT-471.
FONTE: Adaptado de GOOGLE EARTH PRO (2020)

Nota-se que neste ensaio não houve visada entre os rádios e que o trecho é coberto por vegetação de Cerrado denso.

4.2.3. Teste 3: Integração das Camadas de Percepção e Rede

Na integração da camada de percepção (sensores) com a camada de rede (LoRa) foram transmitidos 38 Bytes via LoRa a uma distância de 500 metros. Os dados recebidos ficam armazenados no cartão de memória em um arquivo ".txt". Os dados são tratados e separados por vírgulas pelo microcontrolador. A sequência das informações enviadas foi: temperatura, umidade, data, horário e temperatura do módulo RTC, exemplificado a seguir: "24.20,61.90,08/08/2019,08:00:49,25.25↲".

Analisando a cadência dos dados recebidos na Figura 31 é possível verificar a perda do link da comunicação LoRa nos intervalos de: 8:14 à 8:16; 8:17 à 8:20; e 8:34 à 8:39; sendo necessário a reinicialização do sistema para restabelecer a comunicação.

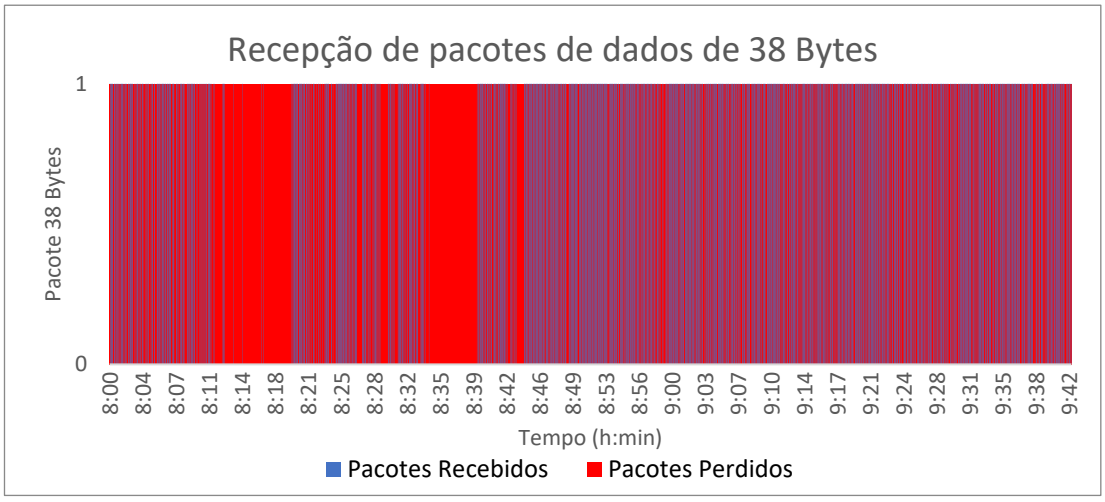

Figura 31 – Pacotes recebidos em um período.
FONTE: Morais (2021)

A cadência de transmissão programada foi de 3 segundos. Durante a transmissão houve também seis falhas de 1 minuto no recebimento dos pacotes, porém sem necessidade de reinicializar o sistema. Os sensores realizaram as leituras com sucesso.

Perante as perdas de link na comunicação LoRa, observou-se que o link só é restabelecido se reinicializar as configurações iniciais do módulo LoRa.

4.2.4. Teste 4: Alcance da Camada de Rede (DOIT/LoRa)

No teste de alcance via comunicação LoRa foi utilizado o protótipo final (MCL) composto por componentes modulares projetado baseado nas necessidades encontradas nos testes anteriores.

Neste teste foram enviados pacotes de 53 Bytes com o fator de espalhamento igual à onze. O pacote era formado pelos dados dos sensores instalados na plantação de milho mais a data e hora da aquisição dos dados. A cadência de transmissão programada foi de 5 segundos. Ao iniciar o deslocamento com o receptor houve alguns pontos cegos durante a rota. O sistema é equipado com módulo de armazenamento micro SD tanto no transmissor quanto no receptor, caso haja perdas de comunicação é possível adquirir a série de dados completa no transmissor onde estão acoplados os sensores.

Os dados transmitidos do plantio são recebidos no receptor localizado na sede da fazenda Experimental da UFMT com uma distância de 770 metros sem visada direta utilizando uma antena omnidirecional. Após esta constatação foram realizados outros deslocamentos fora dos limites da fazenda, foram captados alguns pontos com o GPS do celular conforme o deslocamento e a recepção do sinal, para auxílio na localização as principais distâncias podem ser vistas na Figura 32 a seguir. A maior distância alcançada foi de 2 quilômetros, no interior da cidade. A altura da antena do transmissor era de 1,80 metros do solo.

Figura 32 – Principais pontos alcançados com a transmissão LoRa
FONTE: Adaptado de GOOGLE EARTH PRO (2020)

Como o auxílio da ferramenta de perfil de elevação ofertado pelo Google Earth foram gerados o perfil dos três principais pontos obtidos, que serão mostrados na Figura 33. O ponto MCL-RX1 com a maior distância de transmissão/recepção e o ponto onde as condições do relevo são favoráveis a visada das antenas, com uma pequena estrutura seria possível ajustar essa visada.

Os pontos MCL-RX2 e MCL-RX3 possuem uma elevação entre o transmissor/receptor, onde o sinal pode ter sofrido difração na sua propagação fenômeno em que uma onda passa tangente a um obstáculo e muda a sua trajetória.

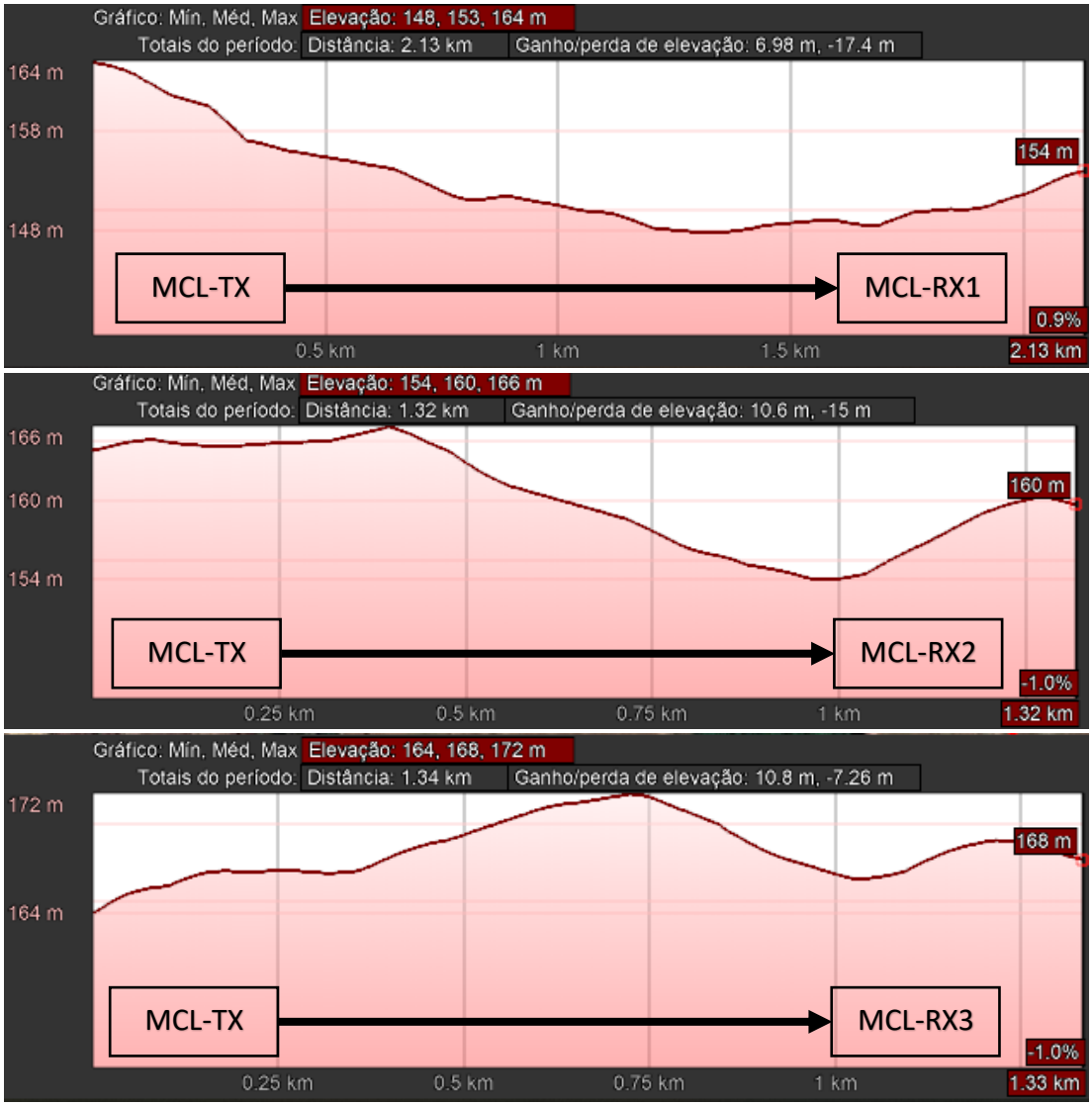

Figura 33 – Perfil de elevação das visadas entre o MCL-TX e o MCL-RX.
FONTE: Adaptado de GOOGLE EARTH PRO (2020)

Após os testes verificou-se a necessidade de correção na rotina de programação para a captura de amostras para o sensor termorresistivo, correção na

programação do módulo LoRa para que perante perda de comunicação RF LoRa o link é restabelecido automaticamente, adicionar a captação autônoma da localização por GPS integrado ao sistema e salvar os parâmetros do GPS e do sinal recebido no cartão de memória, para isso foram realizados de testes adicionais cujo os resultados serão apresentados no próximo tópico.

4.2.5. Teste 5: Alcance da Camada de Rede em Ambientes Urbano e Rural Entre Rádios LoRa Fixo e Outro Transportado em Veículo com Módulos GPS Acoplados (DOIT)

Nos testes anteriores verificou-se que quando ocorre a perda do link na comunicação LoRa, ele só restabelecido após a reinicialização do sistema. Também foi constatado que este erro só ocorre na recepção dos dados, ou seja, o transmissor continua enviando normalmente os dados. Aparentemente o problema de comunicação não está no rádio LoRa e sim na comunicação entre o microcontrolador e módulo LoRa.

Tendo em vista que uma análise na biblioteca "*LoRa.h*" não foi realizada, no entanto foi implementado uma rotina de tempo de espera do sinal (TES), deve ser maior que o tempo de cadência de transmissão. Se o sistema atingir o TES, a rotina aplica as configurações iniciais para o sistema LoRa e a sim o sistema LoRa volta ao seu funcionamento sem a necessidade de reiniciar o microcontrolador, tornando essa verificação de perda de link automática.

A metodologia deste teste foi aplicada em dois ambientes diferentes, uma região urbana e uma região suburbana. Na região urbana o pacote de dados transmitido foi de 55 Bytes a cada 5 segundos, com FS=11. A Figura 34 apresenta a distribuição dos pontos onde o receptor LoRa recebeu o pacote de dados. Não havia visada direta entre o transmissor e o receptor.

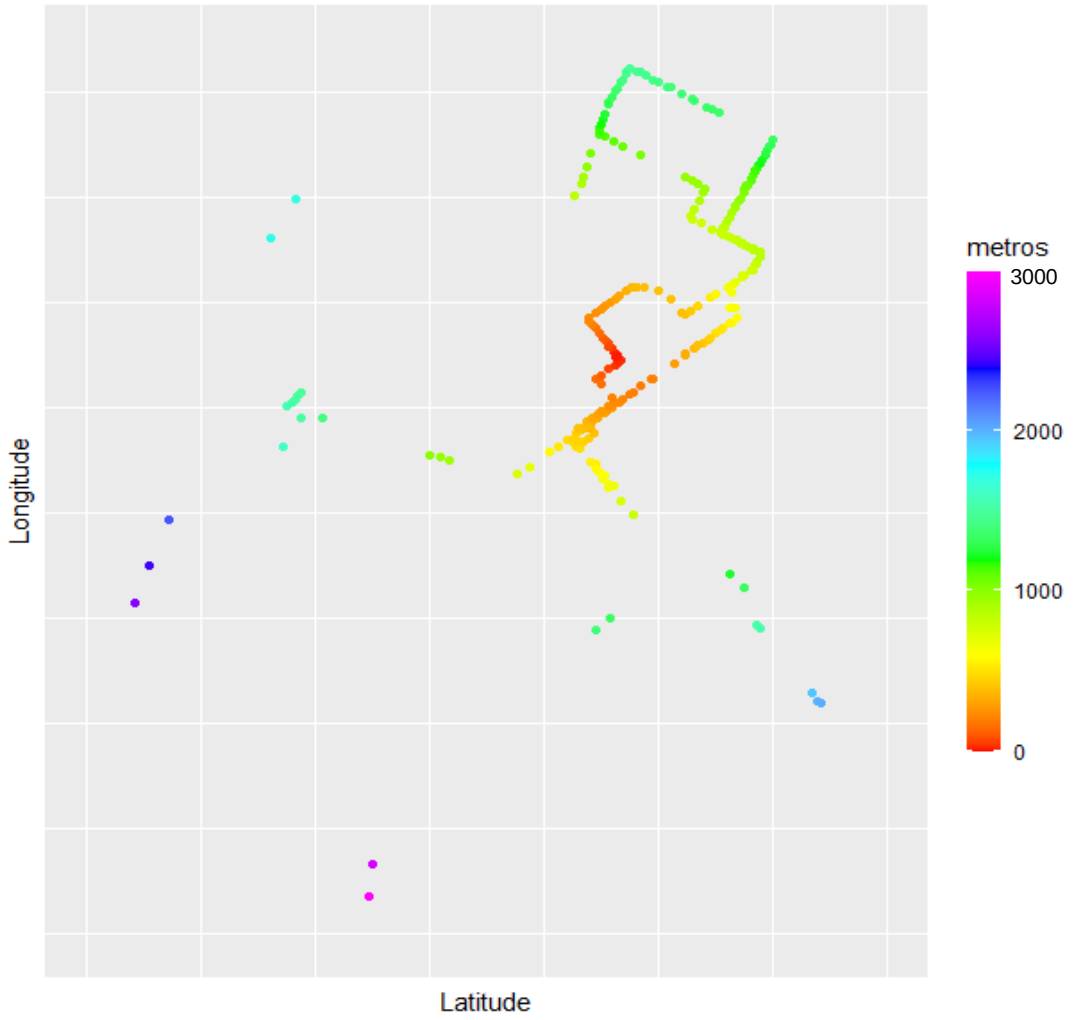

Figura 34 – Distribuição dos pontos conforme o deslocamento do receptor LoRa quando há recepção de sinal.
FONTE: Morais (2021)

Na Figura 34 é possível verificar uma quantidade de pontos maior próximo ao transmissor com um raio de 1 km, a partir desta distância começam a surgir lacunas na transmissão a distância máxima alcançada foi de 3 km. Houve várias perdas de link devido a quantidade de obstruções provenientes de regiões urbanas. A velocidade média de deslocamento do receptor era de 20 km/h.

O próximo resultado apresenta um cenário de campo aberto com pouca ou nenhuma obstrução entre o rádio LoRa transmissor e o receptor, porém além dos pacotes de dados são observados a potência do sinal recebido pela distância percorrida, nesta ocasião o pacote de dados possuía 22 Bytes era enviado a cada 5 segundos, o FS = 11, a frequência central de 433 Mhz. A Figura 35 apresenta o trecho percorrido

durante o teste do ponto zero (transmissor) ao ponto de maior distância captado à 4 km (receptor).

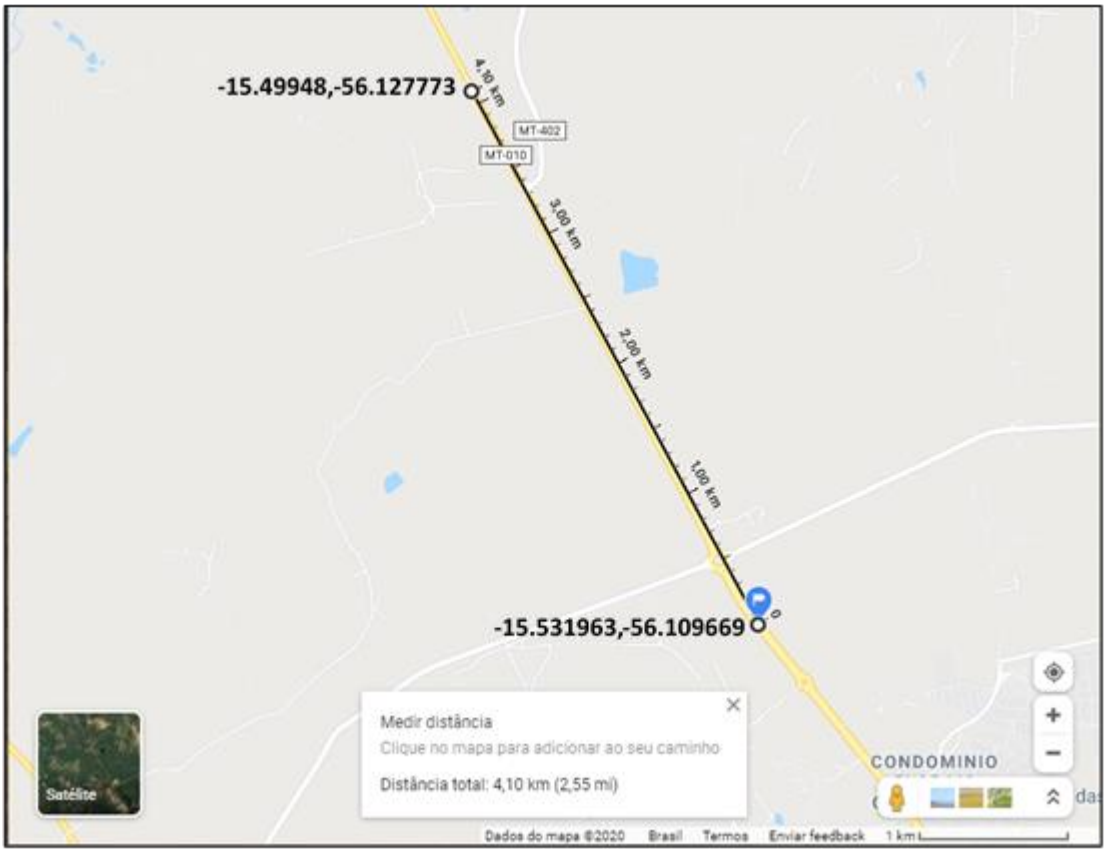

Figura 35 – Transmissão LoRa em campo aberto, com baixa obstrução.
FONTE: Adaptado de Google Maps (2020).

O teste realizado em linha com deslocamento gradativo com velocidade[6] média de 25 km/h, é apresentado no gráfico da Figura 36, que plota a taxa da potência da intensidade do sinal recebido (RSSI) em relação ao deslocamento do Receptor LoRa. Também é adicionado um segundo dado de elevação do relevo.

[6] Velocidade adotada pelo condutor do veículo com pouca variação, pois o tráfego da rodovia estava baixo na ocasião. Referência da velocidade observada no hodômetro do veículo.

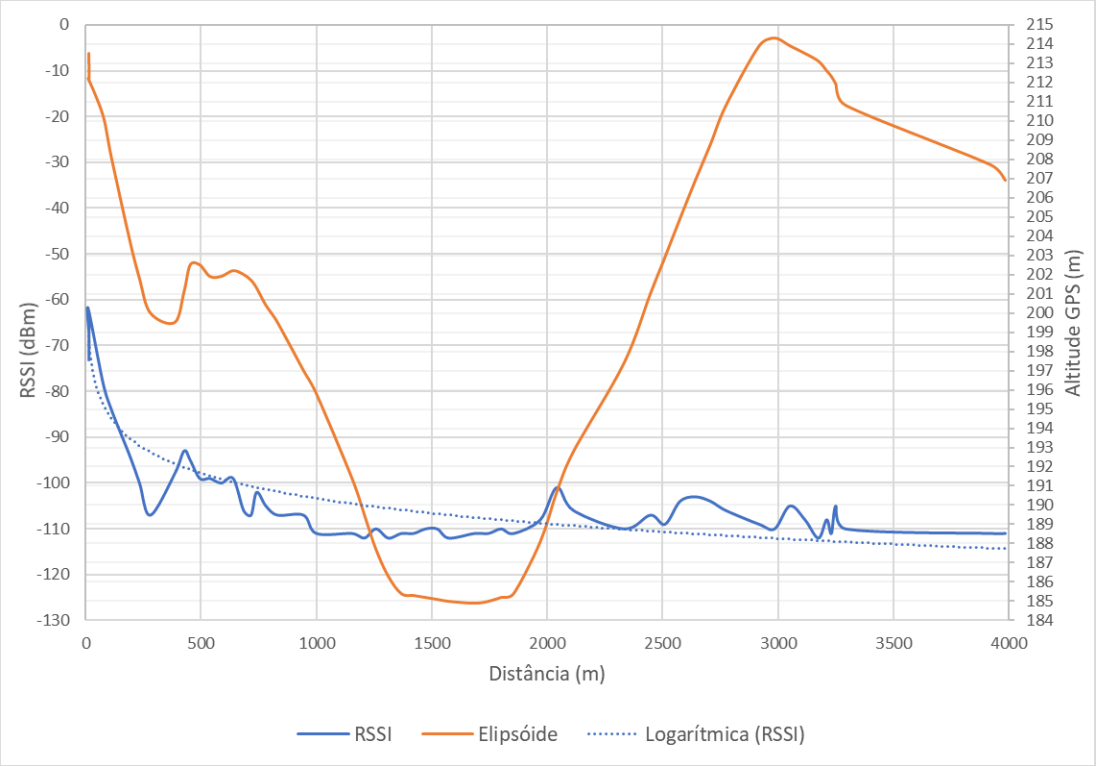

Figura 36 – Transmissão LoRa, relação entre distância e nível do sinal (RSSI).
FONTE: Morais (2021)

Analisando a Figura 36 destacam-se as seguintes características: a intensidade do sinal após 1 km oscila entre -109 dBm à -111 dBm, a intensidade do sinal oscila conforme a característica do perfil do relevo como nos trecho de 0 m a 500 m com diminuição no RSSI e em 2000 m com um ganho de sinal. A linearidade ao final do gráfico representa uma perda de recepção do sinal entre o trecho de 3,3 km a 3,9 km.

5. CONSIDERAÇÕES FINAIS

Na implementação dos dois protótipos LoRa e sua validação, permitem diversas pontuações diante da sua aplicabilidade para estudo de monitoramento ambiental de baixo custo.

A partir da revisão bibliográfica efetuada observou-se que há uma forte tendência ao crescimento em aplicações e projetos envolvendo a IoT, em todos os campos de aplicação, mas também visando o monitoramento ambiental. Nota-se que grandes empresas de tecnologia estão atualizando seus portfolios, procurando em se adequar para aumentar a oferta de soluções IoT. Detectaram-se, porém limitações em termos de futuras padronizações ou integrações entre as diversas tecnologias relacionadas a conectividade para IoT. O baixo número de estudos disponíveis sobre aplicações de IoT com exposição da funcionalidade das suas camadas de rede no monitoramento ambiental ainda sugere, que existe ampla necessidade de melhor explorar as potencialidades e limitações desta abordagem nas ciências ambientais.

A boa disponibilidade de sistemas embarcados ou sistemas SoC de baixo custo foi de grande relevância para a realização deste trabalho. A praticidade em relação a gravação do código (USB), a compactação dos componentes eletrônicos, baixo consumo de energia e a capacidade de conectividade contribuem para criação e desenvolvimento de sistemas versáteis para IoT para o monitoramento ambiental. As duas placas de desenvolvimento utilizadas se mostraram eficientes perante as tarefas designadas e sempre estáveis durante os testes de validação.

A placa de desenvolvimento da HELTEC de design altamente compacto e já integrado com LoRa, utilizada no protótipo inicial, obteve um alcance de transmissão alcançando nos testes 1,5 km em área rural com relevo ondulado.

A placa de desenvolvimento DOIT DevKit V1 (sistema embarcado) que necessita o acoplamento de um RF-LoRa resultou em um design menos compacto. Seu alcance de transmissão foi de 3 a 4 km (dependendo do cenário), utilizando uma antena externa omnidirecional. A versatilidade do protótipo também foi documentada a partir da integração de um módulo GPS, que subsidiou os testes de RF LoRa.

O protótipo (DOIT/LoRa) manteve um *link* de comunicação bastante estável em situações de deslocamento; sendo as perdas de comunicação se deram perante obstruções de relevo ou construções densas.

Os custos para aquisição dos componentes principais do protótipo composto pelo microprocessador e o rádio LoRa foi de US$ 19. Para funcionamento *in situ*, ambas as soluções necessitam de proteção (case). Os custos do protótipo completo foram de US$ 38, incluindo o microprocessador, rádio LoRa, RTC, módulo de armazenamento, case de proteção e antena externa.

Uma vez que o desenvolvimento dos protótipos e sua funcionalidade nos testes apresentou resultado satisfatório, a associação bem-sucedida destes protótipos à tecnologia LoRa permite inferir que esta tecnologia está apta para aplicação em sistema embarcado. Esta associação evidencia que a tecnologia implementada favorece o desenvolvimento de inúmeras aplicações nas ciências ambientais.

5.1. Trabalhos Futuros

Visando a ampliação e aperfeiçoamento de sistemas de monitoramento ambiental de baixo custo sugere-se a realização de possíveis trabalhos futuros:

i) Adquirir módulos LoRa homologados na Anatel, para a possibilitar testes com o MCL com permanência em campo. E disponibilizar esses dados na internet através de rede de telefonia móvel ou rede Wi-Fi com acesso à internet.

ii) Adquirir, implementar e avaliar a comunicação com um Gateway para a implementação do protocolo LoRaWAN, possibilitando a associação de múltiplos sensores ambientais.

iii) Integrar sensores comerciais de baixo custo ao sistema embarcado e validar os dados coletados a partir de um sensor de referência.

iv) Realizar testes com variação no fator de espalhamento (FS) e verificar a influência no RSSI pela distância de recepção. Verificar o TOA na transmissão LoRa.

v) Estudar e desenvolver uma biblioteca LoRa baseada nas existentes, verificar a falha na retomada do *link* de comunicação entre os rádios.

vi) Desenvolver antenas direcionais para teste em pontos fixos, testar o alcance e analisar o sinal LoRa utilizando equipamentos para análise do espectro do sinal.

vii) Aplicar as correções da API ofertadas pela Espressif no conversor analógico/digital do ESP32 e avaliar os resultados.

6. REFERÊNCIAS BIBLIOGRÁFICAS

5G AMERICAS. 5G: The Future of IoT. **5G Americas**, July 2019. Disponivel em: <https://www.5gamericas.org/5g-the-future-of-iot/>. Acesso em: 10 Julho 2020.

AARON.LEE. HelTec AutoMation. LoRa.h, ChengDu, China, 2018. Disponivel em: <https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series>.

AGRIHUB. AGRIHUB. **AGRIHUB**, 2020. Disponivel em: <https://agrihub.com.br/>. Acesso em: 15 ago. 2020.

AI-THINKER TECHNOLOGY CO., LTD. Ai-Thinker. **Tópico Módulo LoRa Series**, 2017. Disponivel em: <https://docs.ai-thinker.com/lora>. Acesso em: 5 Março 2020.

ANATEL. Ato nº 14448, de 04 de dezembro de 2017. **Anatel.gov.br**, 21 Junho 2018. Disponivel em: <https://www.anatel.gov.br/legislacao/index.php/component/content/article?id=1139>. Acesso em: 24 Julho 2020.

ANATEL. Anatel - Telefonia Móvel. **Agência Nacional de Telecomunicações**, 2020. Disponivel em: <https://www.anatel.gov.br/dados/controle-de-qualidade/controle-telefonia-movel#cobertura>. Acesso em: 15 ago. 2020.

AOSONG. **Temperature and humidity module AM2302 Product Manual**. www.aosong.com. Guangzhou, p. 11. 2020.

ASHTON, K. That 'Internet of Things' Thing. **RFID Journal**, 2009. Disponivel em: <https://www.rfidjournal.com/that-internet-of-things-thing-3>. Acesso em: 12 Fevereiro 2021.

AUGUSTIN, A.; YI, J.; CLAUSEN, T.; TOWNSLEY, W. M. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. **Sensors 2016, 16, 1466**, p. 4-7, Setembro 2016. Disponivel em: <www.mdpi.com/journal/sensors>.

BAÑOS-GONZALEZ, V.; AFAQUI, M. S.; LOPEZ-AGUILERA, E.; GARCIA-VILLEGAS, E. IEEE 802.11ah: A Technology to Face the IoT Challenge. **Sensors 2016**, 22 Novembro 2016. 21 p. Disponivel em: <https://doi.org/10.3390/s16111960>.

BLUETOOTH SIG. 2020 Bluetooth Market Update. **Bluetooth SIG**, 2020. Disponivel em: <https://www.bluetooth.com/bluetooth-resources/2020-bmu/?utm_campaign=bmu&utm_source=internal&utm_medium=web&utm_content=2020bmu-featured-content-cta-market-audioentertainment>. Acesso em: 8 Julho 2020.

CIRANI, S.; DAVOLI, L.; FERRARI, G.; LÉONE, R.; MEDAGLIANI, P.; PICONE, M.; VELTRI, L. A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things. **IEEE**, p. 14, 21 Outubro 2014. Disponivel em: <http://ieeexplore.ieee.org>.

ELECTRON TECNOLOGIA DIGITAL. **Sensor de Temperatura-Pt100 - Catálogo Rev. 2.1**. Itupeva,SP, p. 4. 2014.

ESP8266 ARDUINO CORE. Filesystem. **ESP8266 Arduino Core**, 2017. Disponivel em: <https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html>. Acesso em: 11 Junho 2020.

ESPRESSIF - ESP32 SERIES DATASHEET. **ESP32 Series Datasheet**. Espressif Inc. All rights reserved. Xangai, p. 61. 2019.

ESPRESSIF. Analog to Digital Converter. **ESP-IDF Programming Guide**, 2020. Disponivel em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/adc.html#adc-calibration>. Acesso em: 1 Maio 2020.

ESPRESSIF SYSTEMS. ESP32 SoC. **ESPRESSIF**, 2019. Disponivel em: <https://www.espressif.com/en/products/hardware/esp32/overview>. Acesso em: 15 outubro 2019.

ESPRESSIF SYSTEMS. Installation instructions using Arduino IDE Boards Manager. **GitHub**, 2020. Disponivel em: <https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md>. Acesso em: 22 Abril 2020.

ETSI. 2G - Global System for Mobile Communication (GSM). **ETSI**, 2020. Disponivel em: <https://www.etsi.org/technologies/mobile/2g>. Acesso em: 9 Julho 2020.

FANG, S.; XU, L. D.; ZHU, Y.; AHATI, J.; PEI, H.; YAN, J.; LIU, Z. An Integrated System for Regional Environmental Monitoring and Management Based on Internet of Things. **IEEE Transactions on Industrial Informatics**, v. 10, n. 2, p. 1596-1605, maio 2014. DOI: 10.1109/TII.2014.2302638.

FLORE, D. 3GPP Standards for the Internet-of-Things. **3GPP**, Feb. 2016. Disponivel em: <https://www.3gpp.org/images/presentations/3GPP_Standards_for_IoT.pdf>. Acesso em: 10 Julho 2020.

GARCÍA, L.; PARRA, L.; JIMENEZ, J. M.; LLORET, J.; LORENZ, P. IoT-Based Smart Irrigation Systems: An Overview on the Recent Trends on Sensors and IoT Systems for Irrigation in Precision Agriculture. **Sensors**, v. 20(4), p. 1042, 2020.

GHOSLYA, S. All About LoRa and LoRaWAN - LoRa: Symbol Generation. **Educational Blogger**, 2020. Disponivel em: <https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html>. Acesso em: 21 Julho 2020.

GOOGLE EARTH PRO. **Google Earth Pro versão 7.3.3.7786 (64-bit)**. [S.l.]: [s.n.], 2020.

HELTEC AUTOMATION. WiFi LoRa 32. **Heltec Automation**, 2019. Disponivel em: <https://heltec.org/project/wifi-lora-32>. Acesso em: 15 outubro 2019.

HIDAYATA, M. S.; NUGROHOA, A. P.; SUTIARSO, L.; OKAYASU, T. Development of environmental monitoring systems based on LoRa with cloud integration for rural area. **IOP Conference Series: Earth and Environmental Science**, 2019. Disponivel em: <https://doi.org/10.1088/1755-1315/355/1/012010>. Sci. 355 012010.

HUSEIN, N. A. A.; RAHMAN, A. H. A.; DAHNIL, D. P. Evaluation of LoRa-based Air Pollution Monitoring System. **International Journal of Advanced Computer Science and Applications**, Bandar Baru Bangi, v. 10, n. 7, p. 391-396, 2019.

IBGE. Mato Grosso | Cidades e Estados | IBGE. **Instituto Brasileiro de Geografia e Estatística**, 2019. Disponivel em: <https://www.ibge.gov.br/cidades-e-estados/mt/>. Acesso em: 31 jul. 2020.

INPE. Sistema Integrado de Dados Ambientais. **Instituto Nacional de Pesquisas Espaciais**, 2021. Disponivel em: <http://sinda.crn.inpe.br/PCD/SITE/novo/site/sobre.php>. Acesso em: 09 março 2021.

LAMPE, J.; IANELLI, Z. Introduction to Chirp Spread Spectrum (CSS) Technology. **IEEE802**, 11 November 2003. Disponivel em: <http://www.ieee802.org/802_tutorials/03-November/15-03-0460-00-0040-IEEE-802-CSS-Tutorial-part1.ppt>. Acesso em: 11 Setembro 2019.

LAST MINUTE ENGINEERS. Interface DS3231 Precision RTC Module with Arduino. **Last Minute Engineers**, 2020. Disponivel em: <https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/>. Acesso em: 20 Abril 2020.

LAST MINUTE ENGINEERS. Interfacing Micro SD Card Module with Arduino. **Last Minute Engineers**, 2020. Disponivel em: <https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/>. Acesso em: 20 Abril 2020.

LATORRACA, D.; SILVA, F. Onde estão as grandes oportunidades do Agro? Uma visão de dentro da porteira. **Instituto Mato-grossense de Economia Agropecuária (Imea)** , Cuiabá/MT, abr. 2018.

LORA ALLIANCE. **White Paper: A Technical Overview of Lora and Lorawan**. The LoRa Alliance: San Ramon. CA, USA. 2015.

LORA ALLIANCE. About LoRaWAN® | LoRa Alliance. **LoRa Alliance**, 2020. Disponivel em: <https://lora-alliance.org/about-lorawan>. Acesso em: 12 Julho 2020.

LORA ALLIANCE. Home page | LoRa Alliance®. **LoRa Alliance®**, 2020. Disponivel em: <https://lora-alliance.org/>. Acesso em: 11 Julho 2020.

MAXIM INTEGRATED. **DS3231 Datasheet**. San Jose, CA, p. 20. 2015.

MILLER, M. C. **RTC by Makuna**. Version 2.3.3, 16 jul. 2019.

MONTORI, F.; BEDOGNI, L.; BONONI, L. A Collaborative Internet of Things Architecture for Smart Cities and Environmental Monitoring. **IEEE INTERNET OF THINGS JOURNAL**, 5, n. 2, Abril 2018.

MORAIS, P. H. C. D. Repositories GitHub Paulo Morais. **GitHub**, 2021. Disponivel em: <https://github.com/pauloeng28>.

NANOTRON. Technology CSS. **nanotron Technologies**, 2019. Disponivel em: <https://nanotron.com/EN/CO_techn-css.php/>. Acesso em: 25 setembro 2019.

NOVUS PRODUTOS ELETRÔNICOS. Sensores de Temperatura. **Novus Produtos Eletrônicos**, 2020. Disponivel em: <http://www.novus.com.br/produtos/605211>. Acesso em: 28 Abril 2020.

ORACLE BRASIL. O que é Internet das Coisas. **Oracle**, 2020. Disponivel em: <https://www.oracle.com/br/internet-of-things/what-is-iot.html>. Acesso em: 25 março 2020.

SANTOS, B. P.; SILVA, L. A. M.; CELES, C. S. F. S.; NETO, J. B. B.; PERES, B. S.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. A. F. **Internet das Coisas da Teoria à Prática**. Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos. v. 31. 2016.

SEMTECH. What is LoRa? | Semtech. **Semtech**, 2019. Disponivel em: <https://www.semtech.com/lora/what-is-lora>. Acesso em: 14 set. 2019.

SEMTECH CORPORATION. SX1272/3/6/7/8: LoRa Modem Designer's Guide AN1200.13. **Semtech**, July 2013.

SEMTECH CORPORATION. **LoRa™ Modulation Basics AN1200.22**. Semtech Corporation. APPLICATION NOTE, p. 9-12. 2015.

SEMTECH CORPORATION. LoRa® and LoRaWAN®: A Technical Overview. **Semtech Corporation**, p. 26, 2019. Disponivel em: <https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/>. Acesso em: 16 Julho 2020.

SILVA, C. D. S. E. **Um estudo sobre tecnologias de comunicação sem fio para aplicações de IOT em agricultura de precisão**. Dissetação(Mestrado em Computação) - UNIVERSIDADE FEDERAL FLUMINENSE. Niterói, p. 108. 2019.

URSALINK. Digital Farming Is Creating a More Plentiful, Sustainable Food System in Austria. **Ursalink**, 2020. Disponivel em: <https://www.ursalink.com/en/success-stories/digital-farming/>. Acesso em: 25 Julho 2020.

VELÁSQUEZ, P.; VÁSQUEZ, L.; CORREA, C.; RIVERA, D. A low-cost IoT based environmental monitoring system. A citizen approach to pollution awareness. **2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)**, Pucon, Chile, 2017. 1-6. DOI: 10.1109/CHILECON.2017.8229599.

WI-FI ALLIANCE. Wi-Fi HaLow™: Wi-Fi® for IoT applications (2020). **Wi-Fi Alliance**, May 2020. Disponivel em: <https://www.wi-fi.org/downloads-registered-guest/Wi-Fi_HaLow_White_Paper_20200518_0.pdf/36881>. Acesso em: 9 Julho 2020.

WINLIN. SimpleDHT.h - libraries Arduino, 2016-2017. Disponivel em: <https://github.com/winlinvip/SimpleDHT>.

ZANJIREH, M. M.; LARIJANI, H. A Survey on Centralised and Distributed Clustering Routing Algorithms for WSNs. **IEEE 81st Vehicular Technology Conference (VTC Spring)**, Glasgow, maio 2015. 1-6.

ZERYNTH. Docs - DOIT Esp32 DevKit v1. © **Copyright Zerynth**, Pisa, 2020. Disponivel em: <https://docs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html>. Acesso em: 3 Fevereiro 2020.

ANEXOS

ANEXO A – Função para Leitura do Termohigrômetro Digital Desenvolvida em C++.

ANEXO B – Código de Programação Completo do MCL Transmissor.

ANEXO C – Código de Programação Completo do MCL Receptor.

ANEXO A – Função para Leitura do Termohigrômetro Digital Desenvolvida em C++.

```
////////////////////// DHT22 configuration //////////////////////
int Dht22Port; // usei com default
String DataDht22 = "\0";
String MsgError = "\0";
//////////////////////////// Functions ////////////////////////////
byte bits2byte(byte data[8]);
void Dht22();
//////////////////////////// Setup ////////////////////////////////
void setup() {
  ////////////////////////// UART Init ////////////////////////////
  Serial.begin(115200);
  Serial.flush();
  delay(50);
}
//////////////////////////// Main /////////////////////////////////
void loop()
{
  Dht22();
}
//////////////////////////DHT22 Function ////////////////////////
void Dht22()
{
  byte Dht22Data[40];
  unsigned long tempo = 0;
  float Temp = 0;
  float Hum = 0;
  bool flag = false;
  DataDht22 = "\0"; //correção dia 26/01/20 às 22:22
  for ( int dht = 1; dht < 4; dht++) {
    switch (dht) {
      case 1:
        Dht22Port = 26; // 33,32, obs. 35 e 34 não funcionam acredito q possa ser a função
VDET
        //Serial.println("Pot: 26"); 1,126 111.2
        break;
      case 2:
        Dht22Port = 25;
        //Serial.println("Pot: 25");
        break;
      case 3:
        Dht22Port = 33;
        //Serial.println("Pot: 33");
        break;
    }
    memset(Dht22Data, 0, 40);
    pinMode(Dht22Port, OUTPUT);
    digitalWrite(Dht22Port, LOW); //1.T(be), PULL LOW 1ms(0.8-20ms).
    delayMicroseconds(1000);
    digitalWrite(Dht22Port, HIGH);
    pinMode(Dht22Port, INPUT); //2.T(go), PULL HIGH 30us(20-200us).
    delayMicroseconds(40);
    if (digitalRead(Dht22Port) == LOW) {
      tempo = micros();
      while (digitalRead(Dht22Port) == LOW) {
        if ((tempo + 85) <= micros()) {
          MsgError = "ERR:RespLOWtime_S" + String(dht);
          flag = true;
          goto rpt;
        }
      }
    }
    tempo = micros() - tempo;
    if (tempo < 30) {
      MsgError = "ERR:RelLOWtime_S" + String(dht);//3.T(rel), PULL LOW 80us(75-85us).
      flag = true;
      goto rpt;
    }
    if (digitalRead(Dht22Port) == HIGH) {
      tempo = micros();
      while (digitalRead(Dht22Port) == HIGH) {
        if ((tempo + 85) <= micros()) {
```

```
          MsgError = "ERR:RelHIGHtime_S" + String(dht);
          flag = true;
          goto rpt;
        }
      }
    }
    tempo = micros() - tempo;
    if (tempo < 50) MsgError = "ERR:RehHIGHtime_S" + String(dht); //4.T(reh), PULL HIGH
80us(75-85us).
    for (int i = 0; i < 40; i++ ) {
      if (digitalRead(Dht22Port) == LOW) {
        tempo = micros();
        while (digitalRead(Dht22Port) == LOW) {
          if ((tempo + 85) <= micros()) {
            MsgError = "ERR:RehLOWtime_S" + String(dht);
            flag = true;
            goto rpt;
          }
        }
      }
      tempo = micros() - tempo;
      if (tempo < 24) MsgError = "ERR:TLow_S" + String(dht); //5.T(LOW), 1bit start, PULL LOW
50us(48-55us).
      if (digitalRead(Dht22Port) == HIGH) {
        tempo = micros();
        while (digitalRead(Dht22Port) == HIGH) {
          if ((tempo + 85) <= micros()) {
            MsgError = "ERR:TLow_S" + String(dht);
            flag = true;
            goto rpt;
          }
        }
      }
      tempo = micros() - tempo;
      if (tempo < 11) {
        MsgError = "ERR:DataRead_S" + String(dht); //6.T(H0), PULL HIGH 26us(22-30us), bit(0)
        flag = true;
        goto rpt;
      }
      Dht22Data[i] = (tempo > 40 ? 1 : 0); //7.T(H1), PULL HIGH 70us(68-75us), bit(1)
    }
    if (digitalRead(Dht22Port) == LOW) {
      tempo = micros();
      while (digitalRead(Dht22Port) == LOW) {
        if ((tempo + 85) <= micros()) {
          MsgError = "ERR:DataRead_S" + String(dht);
          flag = true;
          goto rpt;
        }
      }
    }
    tempo = micros() - tempo;
    if (tempo < 24) {
      MsgError = "ERR:DataEOF_S" + String(dht); //8.T(en), PULL LOW 50us(45-55us).
      flag = true;
      goto rpt;
    }
rpt:
    if (flag == true) {
      DataDht22 = DataDht22 + ",nan,nan";
      void
      flag = false;
    }
    else {
      short humidity = bits2byte(Dht22Data);
      short humidity2 = bits2byte(Dht22Data + 8);
      short temperature = bits2byte(Dht22Data + 16);
      short temperature2 = bits2byte(Dht22Data + 24);
      byte check = bits2byte(Dht22Data + 32);
      byte expect = (byte)humidity + (byte)humidity2 + (byte)temperature +
(byte)temperature2;
```

```
      if (check != expect)MsgError = "ErrDataCheck_S" + String(dht);
      temperature = temperature << 8 | temperature2;
      humidity = humidity << 8 | humidity2;
      String S_Temp = (String)Temp = (float)((temperature & 0x8000 ? -1 : 1) * (temperature &
0x7FFF)) / 10.0;
      String S_Hum = (String)Hum = (float)humidity / 10.0;
      DataDht22 = (DataDht22 + "," + S_Temp.substring(0, 4) + "," + S_Hum.substring(0, 4));
      //Serial.println(DataDht22);
    }
  }
}

byte bits2byte(byte data[8]) {
  byte v = 0;
  for (int i = 0; i < 8; i++) {
    v += data[i] << (7 - i);
  }
  return v;
}
```

ANEXO B – Código de Programação Completo do MCL Transmissor.

```
#include "RtcDS3231.h"
#include "Wire.h"
#include <time.h> // Estas bibliotecas são exclusivas do sistema. Estão presentes nos packets
hardware ESP32.
#include <sys/time.h> // Utilizam o contador interno do EPS32.
#include "WiFi.h"
#include "TinyGPS.h"
#include "FS.h"
#include "SD.h"
#include "SPI.h"
#include "LoRa.h"
#include "dht22.h"
#include "Adafruit_ADS1015.h" // modificado a bilioteca em #include <Wire.h> para #include
"Wire.h"
//ESP.restart();
////////////////////////////// Wifi configuration ////////////////////////////////
const char* ssid     = "Your_SSID"; // Insira aqui os dados de sua rede Wi-Fi
const char* password = "Your_PASSWORD";
////////////////////////////////// Time configuration ////////////////////////////////
long timezone = -4;
struct tm timeinfo;
unsigned long dly = 0 ;
////////////////////////////// SD Card configuration ////////////////////////////////
#define SD_CS 27 //default 15
#define SD_SCK 14 // default 14
#define SD_MOSI 12 // default 13
#define SD_MISO 13 // 12
SPIClass sd_spi(HSPI);
String SendMsgData = "\0";
String SaveMsgData = "\0";
String MsgPacket = "\0";
String DataDht22 = "\0";
String TimeLocal = "\0";
String MsgErro = "\0";
String Erro = "\0";
////////////////////////////////// Functions //////////////////////////////////////
void LoRaTask(void *Parametro);
void appendFile(fs::FS &fs, const char * path, const char * message);
void GetTimeLocal();
void ErrorMsg();
void cbk(int packetSize);
double temp(double x);
void Pt100();
void Dht22();
void GPSdata();
////////////////////////////// Lora configuration ////////////////////////////////
#define BAND    433E6
#define PABOOST true
#define LoRa_SS 5
#define LoRa_MOSI 18
#define LoRa_MISO 19
#define LoRa_SCK 21
#define LoRa_RST 26
#define Lora_DIO0 15
////////////////////////////// RTC configuration ////////////////////////////////
RtcDS3231<TwoWire> Rtc(Wire);
////////////////////////////// ADS configuration ////////////////////////////////
Adafruit_ADS1115 ads;  /* Use this for the 16-bit version */
float Vin, R1, Vref = 0;
float R2 = 214.9;//218.91;
int16_t adc0, adc1 = 0;
String temp100;
////////////////////////////// GPS configuration ////////////////////////////////
TinyGPS gps;
String SinalGPS = "\0";
////////////////////////////// DHT22 configuration ////////////////////////////////
dht22 dht01(26); // cria objeto dht01 e indica a porta para leitura do sensor Ex. dht22
NomeDoObjeto(nº_do_pino)
dht22 dht02(25);
dht22 dht03(33);
```

```
void setup() {
  ////////////////////////////////// UART Init ////////////////////////////////////
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, 4, 15); //rx , tx - GPS
  Serial.flush();
  delay(50);
  String DateCompile = ("Compiled in: " + String(__DATE__) + " " + String(__TIME__) + "\n");
  Serial.println(DateCompile);
  pinMode(2, OUTPUT);
  ////////////////////////////////// SD Card Init ////////////////////////////////////
  sd_spi.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
  if (!SD.begin(SD_CS, sd_spi)) {
    Serial.println("Card Mount Failed");
    digitalWrite(2, HIGH);
    return;
  }
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    digitalWrite(2, HIGH);
    return;
  }
  ////////////////////////////////// Wifi Init ////////////////////////////////////
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  delay(1000);
  for (int i = 0; i < 3; i++) {
    if (WiFi.status() == WL_CONNECTED) {
      digitalWrite(2, HIGH);
      delay(1000);
      digitalWrite(2, LOW);
      Serial.println("WiFi connected");
      MsgErro = "WiFi connected. ";
      appendFile(SD, "/logTX.txt", MsgErro.c_str());
      MsgErro = "\0";
      MsgErro = "IP address: ";
      MsgErro += WiFi.localIP().toString().c_str();
      MsgErro += "\n";
      Serial.print(MsgErro);
      appendFile(SD, "/logTX.txt", MsgErro.c_str());
      MsgErro = "\0";
      break;
    }
    Serial.print("Reconnecting to ");
    Serial.print(ssid);
    long inicio = (micros() + 10000000);
    while (inicio > micros()) {
      //if (wifi != "CONNECTED") {
      if (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      } else break;
    }
    Serial.println("\nConnection fail");
    MsgErro = "Wi-Fi Connection fail\n";
    appendFile(SD, "/logTX.txt", MsgErro.c_str());
    MsgErro = "\0";
  }
  ////////////////////////////////// RTC Init ////////////////////////////////////
  Wire.begin(16, 17); //SDA, SCL
  Rtc.Begin();
  if (WiFi.status() != WL_CONNECTED) {
jump:
    MsgErro = "RTC used time\n";
    appendFile(SD, "/logTX.txt", MsgErro.c_str());
    MsgErro = "\0";
    timeval tv;//Cria a estrutura temporaria para funcao abaixo.
    tv.tv_sec = (Rtc.GetDateTime() + 946684800);//Atribui a data atual.
    settimeofday(&tv, NULL);//Configura o RTC para manter a data atribuida
```

```
    if ((RtcDateTime(__DATE__, __TIME__) + 120) > Rtc.GetDateTime()) {
      appendFile(SD, "/logTX.txt", DateCompile.c_str());
      appendFile(SD, "/gpsTX.txt", "LAT,LOG,SAT,Precisão,Altura,Date GPS,Time GPS\n");
      appendFile(SD, "/dataTX.txt",
"Date,Time,DHT01Temp,DHT01Hum,DHT02Temp,DHT02Hum,DHT03Temp,DHT03Hum,PT100,LAT,LOG\n");
    }
  }
  else {
    Serial.println("Contacting Time Server");
    int cont = 0;
repeat:
    configTime(3600 * timezone, 3600, "pool.ntp.org");
    getLocalTime(&timeinfo, 5000);
    Serial.println(&timeinfo, "Now is: %A, %B %d %Y %H:%M:%S");
    int ano = timeinfo.tm_year + 1900;
    int mes = timeinfo.tm_mon + 1;
    int dia = timeinfo.tm_mday;
    int hora = timeinfo.tm_hour;
    int minuto = timeinfo.tm_min;
    int segundo = timeinfo.tm_sec;
    cont++;
    if (cont > 10)goto jump;
    if (ano < 2020) goto repeat;
    RtcDateTime HorarioServer = RtcDateTime(ano, mes, dia, hora, minuto, segundo);
    if (!Rtc.IsDateTimeValid()) {
      if (Rtc.LastError() != 0) {
        MsgErro = "RTC communications error = " + Rtc.LastError();
        appendFile(SD, "/logTX.txt", MsgErro.c_str());
        MsgErro = "\0";
        Serial.print("RTC communications error = ");
        Serial.println(Rtc.LastError());
      }
      else {
        MsgErro = "RTC lost confidence in the DateTime!";
        appendFile(SD, "/logTX.txt", MsgErro.c_str());
        MsgErro = "\0";
        Serial.println("RTC lost confidence in the DateTime!");
        Rtc.SetDateTime(HorarioServer);
      }
    }
    if (!Rtc.GetIsRunning()) {
      MsgErro = "RTC was not actively running, starting now";
      appendFile(SD, "/logTX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC was not actively running, starting now");
      Rtc.SetIsRunning(true);
    }
    if ( ((Rtc.GetDateTime() + 60) < HorarioServer) && (ano >= 2020))  {
      MsgErro = "RTC is older than server time!  (Updating DateTime)";
      appendFile(SD, "/logTX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC is older than server time!  (Updating DateTime)");
      Rtc.SetDateTime(HorarioServer);
    }
    if ( (Rtc.GetDateTime()  > (HorarioServer + 60)) && (ano >= 2020))  {
      MsgErro = "RTC is advanced than server time!  (Updating DateTime)";
      appendFile(SD, "/logTX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC is advanced than server time!  (Updating DateTime)");
      Rtc.SetDateTime(HorarioServer);
    }
    if ((RtcDateTime(__DATE__, __TIME__) + 120) > HorarioServer) {
      appendFile(SD, "/logTX.txt", DateCompile.c_str());
      appendFile(SD, "/gpsTX.txt", "LAT,LOG,SAT,Precisão,Altura,Date GPS,Time GPS\n");
      appendFile(SD, "/dataTX.txt",
"Date,Time,DHT01Temp,DHT01Hum,DHT02Temp,DHT02Hum,DHT03Temp,DHT03Hum,PT100,LAT,LOG\n");
    }
  }
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF);
  ////////////////////////////////// LoRa Init ////////////////////////////////////
```

```
  SPI.begin(LoRa_SCK, LoRa_MISO, LoRa_MOSI, LoRa_SS);
  LoRa.setPins(LoRa_SS, LoRa_RST, Lora_DIO0); // default (5,14,26)
  if (!LoRa.begin(BAND, PABOOST)) {
    Serial.print("Starting LoRa failed!\r\n");
    MsgErro = "Starting LoRa failed!";
    ErrorMsg();
    digitalWrite(2, HIGH);
    delay(10000);
    digitalWrite(2, LOW);
  }
  else {
    //LoRa.setSpreadingFactor(11); //default SF11 {6-12}
    Serial.print("LoRa Initial success!\r\n");
    MsgErro = "LoRa Initial success!";
    ErrorMsg();
  }
  ////////////////////////////////// Create task LoRa ////////////////////////////////
  xTaskCreatePinnedToCore (LoRaTask, "LoRa", 1024, NULL, 0, NULL, 0);
  ////////////////////////////////// Config ADS1115 //////////////////////////////////
  Wire.begin(22, 23);
  ads.setGain(GAIN_ONE);// 1x gain   +/- 4.096V  1 bit = 0.125mV
  ads.begin();
  ////////////////////////////////// END SETUP ///////////////////////////////////////
  GetTimeLocal();
  MsgErro = ("Start system in: " + TimeLocal + "\n");
  appendFile(SD, "/logTX.txt", MsgErro.c_str());
  MsgErro = "\0";
}
void loop()
{
  dly = micros();
  GPSdata();
  Pt100 ();
  Dht22();
  GetTimeLocal();
  SaveMsgData = (TimeLocal + "," + DataDht22 + "," + temp100 + "," + SinalGPS + "\n");
  appendFile(SD, "/DataTX.txt", SaveMsgData.c_str());
  SendMsgData = SaveMsgData;
  dly = (micros() - dly) / 1000;
  if (dly < 60000) delay(60000 - dly);
}

void LoRaTask(void *Parametro)
{
  (void) Parametro;
  int counter = 0;
  for (;;) // A Task shall never return or exit.
  {
    if (SendMsgData != MsgPacket)
    {
      digitalWrite(2, HIGH);
      LoRa.beginPacket();
      LoRa.print(SendMsgData);
      LoRa.endPacket();
      Serial.print(SendMsgData);
      MsgPacket = SendMsgData;
      digitalWrite(2, LOW);
    }
  }
}

void ErrorMsg() {
  if (MsgErro != Erro) {
    Erro = MsgErro;
    GetTimeLocal();
    TimeLocal += " => ";
    TimeLocal += MsgErro + "\n";
    appendFile(SD, "/logTX.txt", TimeLocal.c_str());
    TimeLocal = "\0";
  }
}
```

```
void appendFile(fs::FS &fs, const char * path, const char * message) {
  sd_spi.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
  if (!SD.begin(SD_CS, sd_spi)) {
    Serial.println("Card Mount Failed");
    digitalWrite(2, HIGH);
    while (true);
  }
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    digitalWrite(2, HIGH);
    while (true);
  }
  //Serial.printf("Appending to file: %s\n", path);
  File file = fs.open(path, FILE_APPEND);
  if (!file) {
    Serial.println("Failed to open file for appending");
    digitalWrite(2, HIGH);
    while (true);
  }
  if (file.print(message)) {
    //Serial.println("Message appended");
  } else {
    Serial.println("Append failed");
    digitalWrite(2, HIGH);
    while (true);
  }
  file.close();
}

void GetTimeLocal()
{
  if (!getLocalTime(&timeinfo)) {
    MsgErro = "Failed to obtain time";
    ErrorMsg();
    return;
  }
  String dia = String (timeinfo.tm_mday);
  if ((timeinfo.tm_mday) < 10) dia = + "0" + dia;
  String mes = String (timeinfo.tm_mon + 1);
  if ((timeinfo.tm_mon + 1) < 10) mes = "0" + mes;
  String ano = String (timeinfo.tm_year + 1900); //1900
  ano = ano.substring(2, 4);
  String hora = String (timeinfo.tm_hour);
  if ((timeinfo.tm_hour) < 10) hora = "0" + hora;
  String minuto = String (timeinfo.tm_min);
  if ((timeinfo.tm_min) < 10) minuto = "0" + minuto;
  String segundo = String (timeinfo.tm_sec);
  if ((timeinfo.tm_sec) < 10) segundo = "0" + segundo;
  TimeLocal = dia + "/" + mes + "/" + ano + "," + hora + ":" + minuto + ":" + segundo;
}

void Dht22() {
  dht01.dht22Data();
  DataDht22 = String(dht01.Temp) + ",";
  DataDht22 += String(dht01.Hum) + ",";
  if (dht01.MsgError != "SuccessData") {
    MsgErro = dht01.MsgError + ":dht01";
    ErrorMsg();
  }
  dht02.dht22Data();
  DataDht22 += String(dht02.Temp) + ",";
  DataDht22 += String(dht02.Hum) + ",";
  if (dht02.MsgError != "SuccessData") {
    MsgErro = dht02.MsgError + ":dht02";
    ErrorMsg();
  }
  dht03.dht22Data();
  DataDht22 += String(dht03.Temp) + ",";
  DataDht22 += String(dht03.Hum);
```

```
  if (dht03.MsgError != "SuccessData") {
    MsgErro = dht03.MsgError + ":dht03";
    ErrorMsg();
  }
}

void Pt100() {
  int cont = 0;
rpt:
  adc0 = ads.readADC_SingleEnded(0);
  adc1 = ads.readADC_SingleEnded(1);
  Vref = adc1;
  Vref = Vref * 0.0001309 * 2;
  Vin = adc0;
  Vin = Vin * 0.000125;
  R1 = (Vin * R2) / (Vref - Vin);
  cont++;
  if (cont > 50)goto nan;
  if ( R1 < 90 || R1 > 130 )goto rpt;
nan:
  if (cont > 50)temp100 = "nan";
  else temp100 = (String)temp(R1);
  //printf("Tensão: %.3fV - Tensão Ref: %.3fV - Resistência: %.2fOhms - Temperatura:
%.2fºC\n", Vin, Vref, R1, temp(R1));
  //printf("Tensão do Sistema: %.3fV\n", Vref);
}

// correção retirada de Copyright (c) 2019, P. Lutus -- http://arachnoid.com. All Rights
Reserved.
double temp(double x) {
  double terms[] = {
    -4.4277499439427243e+002,
    6.9012656109555630e+000,
    -5.4203211695891609e-004,
    -5.4242492920976264e-004,
    1.4493080701111702e-006,
    1.7821172291658785e-008,
    1.7227205127942579e-010,
    -1.8484057190693898e-012,
    -1.2045312329567013e-014,
    1.5940889037491267e-016,
    -6.8429376874253937e-019,
    2.5960304042778199e-021,
    -6.5642792240198388e-024
  };
  double t = 1;
  double r = 0;
  for (double c : terms) {
    r += c * t;
    t *= x;
  }
  return r;
}

void GPSdata()
{
  bool newData = false;
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (Serial2.available())
    {
      char c = Serial2.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }
  if (newData)
  {
    float flat, flon, alt;
    int year;
```

```
    byte month, day, hour, minute, second, hundredths, sat, prec;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
    sat = gps.satellites();
    prec = gps.hdop();
    alt = gps.f_altitude();
    if (hour > 3)hour = (hour - 4);
    else {
      hour = (hour + 20);
      day = (day - 1);
    }
    char sz[60];
    sprintf(sz, "%f,%f,%d,%d,%0.2f,%02d/%02d/%02d,%02d:%02d:%02d",
            flat, flon, sat, prec, alt, day, month, year, hour, minute, second);
    SinalGPS = sz;
    SinalGPS += "\n";
    appendFile(SD, "/gpsTX.txt", SinalGPS.c_str());
    char sz1[21];
    sprintf(sz1, "%f,%f",
            flat, flon);
    SinalGPS = sz1;
  }
}
```

ANEXO C – Código de Programação Completo do MCL Receptor.

```
#include "RtcDS3231.h"
#include "Wire.h"
#include <time.h> // Estas bibliotecas são exclusivas do sistema. Estão presentes nos packets
hardware ESP32.
#include <sys/time.h> // Utilizam o contador interno do EPS32.
#include "WiFi.h"
#include "TinyGPS.h"
#include "FS.h"
#include "SD.h"
#include "SPI.h"
#include "LoRa.h"
//ESP.restart();
////////////////////////////////// Wifi configuration ////////////////////////////////
const char* ssid     = "Your_SSID"; // Insira aqui os dados de sua rede Wi-Fi
const char* password = "Your_PASSWORD";
////////////////////////////////// Time configuration ////////////////////////////////
long timezone = -4;
struct tm timeinfo;
unsigned long dly = 0 ;
////////////////////////////////// SD Card configuration ////////////////////////////////
#define SD_CS 27 //default 15
#define SD_SCK 14 // default 14
#define SD_MOSI 12 // default 13
#define SD_MISO 13 // 12
SPIClass sd_spi(HSPI);
unsigned long recivedata = 0 ;
String DataMessage = "\0";
String MsgPacket = "\0";
String TimeLocal = "\0";
String MsgErro = "\0";
String Erro = "\0";
String Sinal = "\0";
////////////////////////////////// Functions ////////////////////////////////////////
void Task(void *Parametro);
void appendFile(fs::FS &fs, const char * path, const char * message);
void GetTimeLocal();
void ErrorMsg();
void cbk(int packetSize);
void GPSdata();
double calcDist(float ilat, float ilog, float flat, float flon);
double regress(double x);
////////////////////////////////// Lora configuration ////////////////////////////////
#define BAND    433E6
#define PABOOST true
#define LoRa_SS 5
#define LoRa_MOSI 18
#define LoRa_MISO 19
#define LoRa_SCK 21
#define LoRa_RST 26
#define Lora_DIO0 15
String packSize;
////////////////////////////////// RTC configuration ////////////////////////////////
RtcDS3231<TwoWire> Rtc(Wire);
////////////////////////////////// GPS configuration ////////////////////////////////
TinyGPS gps;
String SinalGPS = "\0";
float flat, flon, alt, ilat, ilog;
double dist;

////////////////////////////////////SETUP INIT/////////////////////////////////////////
void setup() {
  ////////////////////////////////////// UART Init ////////////////////////////////////
  Serial.begin(115200);
  Serial2.begin (9600, SERIAL_8N1, 4, 15); //rx , tx - GPS
  Serial.flush();
  delay(50);
  String DateCompile = ("Compiled in: " + String(__DATE__) + " " + String(__TIME__) + "\n");
  Serial.println(DateCompile);
  pinMode(2, OUTPUT);
  ////////////////////////////////// SD Card Init ////////////////////////////////////
  sd_spi.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
```

```
  if (!SD.begin(SD_CS, sd_spi)) {
    Serial.println("Card Mount Failed");
    digitalWrite(2, HIGH);
    return;
  }
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    digitalWrite(2, HIGH);
    return;
  }
  ////////////////////////////////// Wifi Init ////////////////////////////////////
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  delay(1000);
  for (int i = 0; i < 3; i++) {
    if (WiFi.status() == WL_CONNECTED) {
      digitalWrite(2, HIGH);
      delay(1000);
      digitalWrite(2, LOW);
      Serial.println("WiFi connected");
      MsgErro = "WiFi connected. ";
      appendFile(SD, "/logRX.txt", MsgErro.c_str());
      MsgErro = "\0";
      MsgErro = "IP address: ";
      MsgErro += WiFi.localIP().toString().c_str();
      MsgErro += "\n";
      Serial.print(MsgErro);
      appendFile(SD, "/logRX.txt", MsgErro.c_str());
      MsgErro = "\0";
      break;
    }
    Serial.print("Reconnecting to ");
    Serial.print(ssid);
    long inicio = (micros() + 10000000);
    while (inicio > micros()) {
      //if (wifi != "CONNECTED") {
      if (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
      } else break;
    }
    Serial.println("\nConnection fail");
    MsgErro = "Wi-Fi Connection fail\n";
    appendFile(SD, "/logRX.txt", MsgErro.c_str());
    MsgErro = "\0";
  }
  ////////////////////////////////// RTC Init ////////////////////////////////////
  Wire.begin(16, 17); //SDA, SCL
  Rtc.Begin();
  if (WiFi.status() != WL_CONNECTED) {
jump:
    MsgErro = "RTC used time\n";
    appendFile(SD, "/logRX.txt", MsgErro.c_str());
    MsgErro = "\0";
    timeval tv;//Cria a estrutura temporaria para funcao abaixo.
    tv.tv_sec = (Rtc.GetDateTime() + 946684800);//Atribui a data atual.
    settimeofday(&tv, NULL);//Configura o RTC para manter a data atribuida
    if ((RtcDateTime(__DATE__, __TIME__) + 120) > Rtc.GetDateTime()) {
      appendFile(SD, "/logRX.txt", DateCompile.c_str());
      appendFile(SD, "/gpsRX.txt", "LAT,LOG,SAT,Precisão,Altura,Date GPS,Time GPS\n");
      appendFile(SD, "/dataRX.txt",
"LocalDate,LocalTime,LocalLAT,LocalLOG,Distance,RSSI,SNR,PackSize,TxDate,TxTime,DHT01Temp,DHT
01Hum,DHT02Temp,DHT02Hum,DHT03Temp,DHT03Hum,PT100,LAT,LOG\n");
    }
  }
  else {
    Serial.println("Contacting Time Server");
    int cont = 0;
repeat:
```

```
    configTime(3600 * timezone, 3600, "pool.ntp.org");
    getLocalTime(&timeinfo, 5000);
    Serial.println(&timeinfo, "Now is: %A, %B %d %Y %H:%M:%S");
    int ano = timeinfo.tm_year + 1900;
    int mes = timeinfo.tm_mon + 1;
    int dia = timeinfo.tm_mday;
    int hora = timeinfo.tm_hour;
    int minuto = timeinfo.tm_min;
    int segundo = timeinfo.tm_sec;
    cont++;
    if (cont > 10)goto jump;
    if (ano < 2020) goto repeat;
    RtcDateTime HorarioServer = RtcDateTime(ano, mes, dia, hora, minuto, segundo);
    if (!Rtc.IsDateTimeValid()) {
      if (Rtc.LastError() != 0) {
        MsgErro = "RTC communications error = " + Rtc.LastError();
        appendFile(SD, "/logRX.txt", MsgErro.c_str());
        MsgErro = "\0";
        Serial.print("RTC communications error = ");
        Serial.println(Rtc.LastError());
      }
      else {
        MsgErro = "RTC lost confidence in the DateTime!";
        appendFile(SD, "/logRX.txt", MsgErro.c_str());
        MsgErro = "\0";
        Serial.println("RTC lost confidence in the DateTime!");
        Rtc.SetDateTime(HorarioServer);
      }
    }
    if (!Rtc.GetIsRunning()) {
      MsgErro = "RTC was not actively running, starting now";
      appendFile(SD, "/logRX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC was not actively running, starting now");
      Rtc.SetIsRunning(true);
    }
    if ( ((Rtc.GetDateTime() + 60) < HorarioServer) && (ano >= 2020))  {
      MsgErro = "RTC is older than server time!  (Updating DateTime)";
      appendFile(SD, "/logRX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC is older than server time!  (Updating DateTime)");
      Rtc.SetDateTime(HorarioServer);
    }
    if ( (Rtc.GetDateTime()  > (HorarioServer + 60)) && (ano >= 2020))  {
      MsgErro = "RTC is advanced than server time!  (Updating DateTime)";
      appendFile(SD, "/logRX.txt", MsgErro.c_str());
      MsgErro = "\0";
      Serial.println("RTC is advanced than server time!  (Updating DateTime)");
      Rtc.SetDateTime(HorarioServer);
    }
    if ((RtcDateTime(__DATE__, __TIME__) + 120) > HorarioServer) {
      appendFile(SD, "/logRX.txt", DateCompile.c_str());
      appendFile(SD, "/gpsRX.txt", "LAT,LOG,SAT,Precisão,Altura,Date GPS,Time GPS\n");
      appendFile(SD, "/dataRX.txt",
"LocalDate,LocalTime,LocalLAT,LocalLOG,Distance,RSSI,SNR,PackSize,TxDate,TxTime,DHT01Temp,DHT
01Hum,DHT02Temp,DHT02Hum,DHT03Temp,DHT03Hum,PT100,LAT,LOG\n");
    }
  }
  WiFi.disconnect(true);
  WiFi.mode(WIFI_OFF);
  ////////////////////////////////// LoRa Init //////////////////////////////////
  SPI.begin(LoRa_SCK, LoRa_MISO, LoRa_MOSI, LoRa_SS);
  LoRa.setPins(LoRa_SS, LoRa_RST, Lora_DIO0); // default (5,14,26)
  if (!LoRa.begin(BAND, PABOOST)) {
    Serial.print("Starting LoRa failed!\r\n");
    MsgErro = "Starting LoRa failed!";
    ErrorMsg();
    digitalWrite(2, HIGH);
    delay(10000);
    digitalWrite(2, LOW);
  }
```

```
  else {
    //LoRa.setSpreadingFactor(11); //default SF11 {6-12}
    Serial.print("LoRa Initial success!\r\n");
    MsgErro = "LoRa Initial success!";
    ErrorMsg();
  }
  ////////////////////////////////// Create Secondary Task ////////////////////////////////
  xTaskCreatePinnedToCore (Task, "Task", 8192, NULL, 0, NULL, 0);
  disableCore0WDT();
  ////////////////////////////////////// END SETUP ////////////////////////////////////////
  GetTimeLocal();
  MsgErro = ("Start system in: " + TimeLocal + "\n");
  appendFile(SD, "/logRX.txt", MsgErro.c_str());
  MsgErro = "\0";
}
void loop()
{
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    cbk(packetSize);
    recivedata = micros ();
  }
  if ((recivedata + 90000000) <= micros()) { // this delay must be synchronized with the
transmission interval
    recivedata = micros ();
    SPI.begin(LoRa_SCK, LoRa_MISO, LoRa_MOSI, LoRa_SS);
    LoRa.setPins(LoRa_SS, LoRa_RST, Lora_DIO0); // default (5,14,26)
    if (!LoRa.begin(BAND, PABOOST)) {
      Serial.print("Starting LoRa failed!\r\n");
      MsgErro = "Starting LoRa failed!";
      ErrorMsg();
      pinMode(2, OUTPUT);
      digitalWrite(2, HIGH);
      delay(10000);
      digitalWrite(2, LOW);
    }
    else {
      Serial.print("LoRa Initial success!\r\n");
      MsgErro = "LoRa Initial success!";
      ErrorMsg();
    }
  }
  if (DataMessage != MsgPacket) {
    //appendFile(SD, "/dataRX.txt", DataMessage.c_str());
    MsgPacket = DataMessage;
    if (flat != 0 ) {
      String latStr, logStr;
      latStr = DataMessage.substring(60, 70);
      logStr = DataMessage.substring(71, 81);
      ilat = latStr.toFloat();
      ilog = logStr.toFloat();
      dist = calcDist (ilat, ilog, flat, flon);
    }
    GetTimeLocal();
    Sinal = LoRa.packetRssi();
    Sinal = "RSSI: " + Sinal + " Packet Size: " + packSize + " Distance: " + String(dist);
    Serial.println(Sinal);
    Sinal = TimeLocal + "," + SinalGPS + "," + String(dist) + "," + LoRa.packetRssi() + "," +
LoRa.packetSnr() + "," + packSize + "," + DataMessage;
    appendFile(SD, "/DataRX.txt", Sinal.c_str());
    Serial.print(Sinal);//
  }
}

void cbk(int packetSize) {
  DataMessage = "\0";
  packSize = String(packetSize, DEC);
  //Serial.println("Received " + packSize + " bytes");
  for (int i = 0; i < packetSize; i++) {
    DataMessage += (char) LoRa.read();
  }
```

```
  //Serial.println (DataMessage);
}

void Task(void *Parametro) {
  delay(100);
  (void) Parametro;
  for (;;) { // A Task shall never return or exit.
    GPSdata();
  }
}

void ErrorMsg() {
  if (MsgErro != Erro) {
    Erro = MsgErro;
    GetTimeLocal();
    TimeLocal += " => ";
    TimeLocal += MsgErro + "\n";
    appendFile(SD, "/logRX.txt", TimeLocal.c_str());
    TimeLocal = "\0";
  }
}

void appendFile(fs::FS &fs, const char * path, const char * message) {
  sd_spi.begin(SD_SCK, SD_MISO, SD_MOSI, SD_CS);
  if (!SD.begin(SD_CS, sd_spi)) {
    Serial.println("Card Mount Failed");
    digitalWrite(2, HIGH);
    while (true);
  }
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    digitalWrite(2, HIGH);
    while (true);
  }
  //Serial.printf("Appending to file: %s\n", path);
  File file = fs.open(path, FILE_APPEND);
  if (!file) {
    Serial.println("Failed to open file for appending");
    digitalWrite(2, HIGH);
    while (true);
  }
  if (file.print(message)) {
    //Serial.println("Message appended");
  } else {
    Serial.println("Append failed");
    digitalWrite(2, HIGH);
    while (true);
  }
  file.close();
}

void GetTimeLocal()
{
  if (!getLocalTime(&timeinfo)) {
    MsgErro = "Failed to obtain time";
    ErrorMsg();
    return;
  }
  String dia = String (timeinfo.tm_mday);
  if ((timeinfo.tm_mday) < 10) dia = + "0" + dia;
  String mes = String (timeinfo.tm_mon + 1);
  if ((timeinfo.tm_mon + 1) < 10) mes = "0" + mes;
  String ano = String (timeinfo.tm_year + 1900); //1900
  ano = ano.substring(2, 4);
  String hora = String (timeinfo.tm_hour);
  if ((timeinfo.tm_hour) < 10) hora = "0" + hora;
  String minuto = String (timeinfo.tm_min);
  if ((timeinfo.tm_min) < 10) minuto = "0" + minuto;
  String segundo = String (timeinfo.tm_sec);
  if ((timeinfo.tm_sec) < 10) segundo = "0" + segundo;
```

```
  TimeLocal = dia + "/" + mes + "/" + ano + "," + hora + ":" + minuto + ":" + segundo;
  //Serial.println(TimeLocal);
}

void GPSdata()
{
  bool newData = false;
  for (unsigned long start = millis(); millis() - start < 1000;)
  {
    while (Serial2.available())
    {
      char c = Serial2.read();
      // Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) // Did a new valid sentence come in?
        newData = true;
    }
  }
  if (newData)
  {
    //float flat, flon, alt;
    int year;
    byte month, day, hour, minute, second, hundredths, sat, prec;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
    sat = gps.satellites();
    prec = gps.hdop();
    alt = gps.f_altitude();
    if (hour > 3)hour = (hour - 4);
    else {
      hour = (hour + 20);
      day = (day - 1);
    }
    char sz[60];
    sprintf(sz, "%f,%f,%d,%d,%0.2f,%02d/%02d/%02d,%02d:%02d:%02d",
            flat, flon, sat, prec, alt, day, month, year, hour, minute, second);
    SinalGPS = sz;
    SinalGPS += "\n";
    appendFile(SD, "/gpsRX.txt", SinalGPS.c_str());
    char sz1[21];
    sprintf(sz1, "%f,%f,%0.2f",
            flat, flon, alt);
    SinalGPS = sz1;
  }
}

double calcDist(float ilat, float ilog, float flat, float flon) {
  double dlong = (flon - ilog) ;
  //Serial.println(dlong, 6);
  double dlat = (flat - ilat) ;
  //Serial.println(dlat, 6);
  double c = sqrt(dlong * dlong  + dlat * dlat );
  //Serial.println (c, 7);
  return regress (c);
}

double regress(double x) {
  double terms[] = {
    -3.1689916842841064e+000,
    1.0901575726347862e+005,
    -4.6643053664161905e+005,
    5.7156360892116681e+007,
    -3.9168386961030402e+009,
    1.5607395647971548e+011,
    -3.4824757033661943e+012,
    3.2706372378635406e+013,
    2.9106720619946300e+014,
    -1.1856451505307860e+016,
    1.4006946266044288e+017,
    -7.8102632313790259e+017,
    1.7393110833417582e+018
```

```
  };

  double t = 1;
  double r = 0;
  for (double c : terms) {
    r += c * t;
    t *= x;
  }
  return r;
}
```