

INSTITUTO SUPERIOR MANUEL TEIXEIRA GOMES



## **Protótipo LoRa - Geolocalizador de animais**

**Relatório de Trabalho Final de Curso**

**Pedro Miguel Carmona Semedo Pinto**

Trabalho orientado por:

Prof. Doutor Francisco Melo Pereira

2023



INSTITUTO SUPERIOR MANUEL TEIXEIRA GOMES



## **Protótipo LoRa - Geolocalizador de animais**

**Relatório de Trabalho Final de Curso**

**Pedro Miguel Carmona Semedo Pinto**

Trabalho orientado por:

Prof. Doutor Francisco Melo Pereira

2023



# Protótipo LoRa - Geolocalizador de animais Declaração de autoria

Declaro ser o (a) autor(a) do trabalho apresentado neste relatório, sendo original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

---

(Assinatura)

O Instituto Superior Manuel Teixeira Gomes tem o direito, perpétuo e sem limites geográficos, de arquivar e publicitar este trabalho através de quaisquer meios, de o divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos não comerciais, desde que seja dado crédito ao autor.

---

(Assinatura)



# Agradecimentos

Gostaria de expressar meus sinceros agradecimentos às pessoas e instituição que desempenharam um papel crucial na minha jornada académica e na conclusão deste trabalho, primeiramente à minha família e amigos que sempre me apoiaram em todos os momentos e sem eles não seria possível ter chegado até aqui. Agradecer também ao meu orientador Professor Doutor Francisco Melo Pereira e ao Mestre Rui Mesquita, pela orientação valiosa e apoio incansável. Agradeço também ao meu colega Leandro Fonseca pela colaboração neste projeto. Agradecer também ao resto do corpo docente da licenciatura em Engenharia de informática do ISMAT que me acompanhou neste percurso e contribuiu para o meu conhecimento e ainda à instituição do ISMAT por me acolher durante estes 3 anos.



# Resumo

Este relatório apresenta um projeto de pesquisa que se foca na implementação de soluções de geolocalização de animais através de tecnologia de comunicação Long Range (LoRa). O objetivo principal do projeto é desenvolver e comparar protótipos.

O projeto envolve a criação de vários protótipos utilizando diferentes placas de desenvolvimento compatíveis com a tecnologia LoRa. Cada protótipo foi equipado com sensores Global Positioning System (GPS) para a geolocalização de animais perdidos em intervalos de tempo.

Uma parte fundamental do projeto foi a comparação das autonomias dos diferentes protótipos. Cada placa de desenvolvimento possuía características únicas que poderiam afetar o consumo de energia e, consequentemente, a duração da bateria. Foram realizados cálculos teóricos para avaliar a eficiência energética de cada protótipo.

Os resultados da pesquisa revelaram informações importantes sobre as placas de desenvolvimento, as configurações de transmissão, a frequência de recolha de dados e a vida útil da bateria. Essas descobertas contribuem significativamente para o avanço da tecnologia de geolocalização de animais, permitindo escolhas mais informadas ao selecionar a melhor solução para diferentes cenários.

**Palavras-chave:** Geolocalização de animais perdidos, Tecnologia LoRa, Protótipos de dispositivos, Autonomia.



# Abstract

This report presents a research project focusing on the implementation of animal geolocation solutions through LoRa (Long Range) communication technology. The main objective of the project is to develop and compare prototypes for monitoring the location of lost animals.

The project involves creating several prototypes using different development boards compatible with LoRa technology. Each prototype was equipped with GPS sensors to collect location data from lost animals at regular intervals.

A crucial part of the project was comparing the autonomy of the different prototypes. Each development board had unique characteristics that could affect power consumption and, consequently, battery life. Theoretical calculations were performed to assess the energy efficiency of each prototype.

The research results revealed significant insights into the development boards, transmission settings, data collection frequency, and battery life. These findings contribute significantly to advancing animal geolocation technology, enabling more informed choices when selecting the best solution for different scenarios.

**Keywords:** Lost animal geolocation, LoRa technology, Device prototypes, Autonomy and energy efficiency.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	LoRa . . . . .	1
1.2	Motivação . . . . .	2
1.3	Contextualização . . . . .	2
1.4	Problemas Identificados . . . . .	3
1.5	Objetivos . . . . .	4
1.6	Estrutura . . . . .	5
<b>2</b>	<b>Estado da arte</b>	<b>7</b>
<b>3</b>	<b>Análise e Desenho</b>	<b>9</b>
3.1	Requisitos . . . . .	9
3.1.1	Requisitos Funcionais . . . . .	9
3.1.2	Requisitos Não Funcionais . . . . .	10
3.2	Diagramas . . . . .	11
3.3	Lógica . . . . .	12
<b>4</b>	<b>Desenvolvimento</b>	<b>15</b>
4.1	Arquitetura . . . . .	15
4.2	Protocolos de Comunicação . . . . .	16
4.3	Base de Dados . . . . .	17
4.4	MVC . . . . .	18
4.5	Software . . . . .	20
4.5.1	Ambientes de desenvolvimento . . . . .	20
4.5.2	Bibliotecas . . . . .	21
4.6	Equipamento Utilizado . . . . .	23
4.6.1	Placas de desenvolvimento . . . . .	23
4.6.2	Sensores . . . . .	29

<b>5 Implementação</b>	<b>31</b>
5.1 Protótipos . . . . .	31
5.1.1 Cubecell . . . . .	31
5.1.2 ESP32 . . . . .	32
5.1.3 LoRa Radio Node v1 . . . . .	33
5.1.4 Heltec CubeCell GPS-6502 . . . . .	35
5.1.5 Arduino-Pro-Mini . . . . .	36
5.2 Gateway . . . . .	38
5.3 APP . . . . .	38
5.3.1 Dispositivos da aplicação . . . . .	40
5.3.2 Pacotes da aplicação . . . . .	41
5.4 Dificuldades na implementação . . . . .	42
<b>6 Resultados</b>	<b>43</b>
6.1 Fórmulas . . . . .	43
6.2 Cálculos . . . . .	46
6.2.1 CubeCell 1/2AA Node . . . . .	46
6.2.2 CubeCell GPS-6502 . . . . .	47
6.2.3 Heltec WiFi LoRa 32 V2 . . . . .	48
6.2.4 LoRa Radio Node v1.0 . . . . .	49
<b>7 Conclusão</b>	<b>51</b>
7.1 Melhorias Futuras . . . . .	52
<b>Bibliografia</b>	<b>55</b>



# **Lista de Figuras**

3.1	Diagrama de Sequência . . . . .	11
3.2	Diagrama de Casos de Usos da aplicação . . . . .	11
3.3	Diagrama de atividade . . . . .	12
3.4	Máquina de estados . . . . .	13
4.1	Arquitetura da Aplicação . . . . .	15
4.2	Exemplo de pacote LoRa em formato JSON . . . . .	16
4.3	Base de Dados . . . . .	17
4.4	Diagrama MVC aplicado usando uma API . . . . .	19
4.5	Heltec Cubecell 1/2AA Node . . . . .	23
4.6	Heltec CubeCell GPS-6502 . . . . .	24
4.7	Heltec WiFi LoRa 32 V2 . . . . .	25
4.8	TTGO LoRa32 V1 . . . . .	26
4.9	LoRa Radio Node v1.0 . . . . .	27
4.10	Arduino Pro Mini + Reyax RYLR896 . . . . .	28
4.11	Módulo GPS Neo6-M . . . . .	29
4.12	Módulo RTC DS3231 . . . . .	30
5.1	Esquemática - Cubecell 1/2AA . . . . .	31
5.2	Esquemática - ESP32 . . . . .	32
5.3	Esquemática - LoRa Radio Node v1 . . . . .	34
5.4	Esquemática - CubeCell GPS-6502 . . . . .	35
5.5	Esquemática - Arduino-Pro-Mini . . . . .	38
5.6	Dispositivos da aplicação . . . . .	40
5.7	Pacotes da aplicação . . . . .	41



# **Lista de Tabelas**

4.1	Especificações Heltec Cubecell 1/2AA Node[1] . . . . .	23
4.2	Especificações Heltec CubeCell GPS-6502[2] . . . . .	24
4.3	Especificações Heltec WiFi LoRa 32 v2[3] . . . . .	25
4.4	Especificações TTGO LoRa32 V1[4] . . . . .	26
4.5	Especificações LoRa Radio Node v1.0[5] . . . . .	27
4.6	Especificações Arduino Pro Mini[6] + Reyax RYLR896[7] . . . . .	28
4.7	Especificações GPS Neo6-M [8] . . . . .	29
4.8	Módulo RTC DS3231 [9] . . . . .	30



# Siglas

**API** Application Programming Interface. xi, 13, 15, 16, 19, 38

**AVR** Advanced Virtual RISC. 20

**GPS** Global Positioning System. v, 7, 21, 22, 31–33, 35, 36

**HTTP** Hypertext Transfer Protocol. 15, 16, 38

**I2C** Inter-Integrated Circuit. 33, 36

**IoT** Internet of Things. 1, 2, 7, 21, 23, 51

**JSON** Javascript Object Notation. xi, 9, 15, 16, 21

**LoRa** Long Range. v, xi, 1, 2, 4, 9, 10, 15, 16, 21–23, 38, 43, 44, 51

**MVC** Model-View-Controller. xi, 18, 19

**RFID** Radio-frequency identification. 7

**RTC** Real-Time Clock. 22, 33, 35, 36

**SCL** Serial Clock. 33, 36

**SDA** Serial Data. 33, 36

**SPI** Serial Peripheral Interface. 21

**WI-FI** Wireless Fidelity. 1

# Introdução

Este projeto surge a partir de uma colaboração entre a Universidade do ISMAT e a câmara municipal de Lagoa relativo ao contrato para desenvolver o seguinte tema: serviços de prototipagem de sensores para rede LoRa.

## 1.1 LoRa

Nos últimos anos, o avanço tecnológico tem revolucionado a forma como interagimos com o mundo ao nosso redor. Uma parte fundamental dessa revolução é a proliferação da Internet das Coisas (IoT), que tem como objetivo conectar uma vasta gama de dispositivos e sensores à internet, tornando possível a recolha e troca de informações de forma inteligente e eficiente. No entanto, a diversidade de dispositivos IoT, bem como a variedade de cenários de aplicação, têm apresentado desafios únicos para garantir uma conectividade confiável e abrangente.

Aqui entra a tecnologia Long Range (LoRa), uma solução inovadora que busca enfrentar os desafios de conectividade de longo alcance e baixo consumo de energia inerentes à IoT. Desenvolvida para viabilizar a comunicação de dispositivos em áreas extensas e remotas, a tecnologia LoRa oferece uma alternativa altamente eficaz às tecnologias tradicionais de comunicação sem fio, como Wireless Fidelity (WI-FI) e Bluetooth, que muitas vezes são limitadas em termos de alcance e consumo de energia.

O princípio fundamental da tecnologia LoRa reside na sua capacidade de transmitir dados em distâncias consideráveis, mesmo em ambientes onde obstáculos físicos podem interferir com a comunicação. A tecnologia LoRa permite uma propagação robusta de sinal, tornando-a particularmente adequada para aplicações que abrangem grandes áreas geográficas, como monitoramento agrícola, cidades inteligentes, monitoramento ambiental e muito mais.

Além do alcance abrangente, a tecnologia LoRa destaca-se pelo seu baixo consumo de energia, um fator crítico para dispositivos IoT que muitas vezes são alimentados por baterias de longa duração. Ao adotar técnicas eficientes de modulação e transmissão, esta tecnologia consegue manter a conectividade enquanto minimiza o uso de energia, permitindo que os dispositivos IoT permaneçam ativos por períodos significativos sem a necessidade de trocas frequentes de bateria.

## 1.2 Motivação

Num mundo em constante evolução tecnológica, procura-se sempre explorar algo desafiante e inovador. A hipótese de se desenvolver um protótipo eficaz a rastrear e localizar animais domésticos, apresenta um desafio emocionante e motivador. Ao mesmo tempo a possibilidade de aprender mais sobre a tecnologia LoRa permite alargar os horizontes do conhecimento e oferecer soluções mais eficientes e sustentáveis. Além disso, a geolocalização de animais tem aplicações práticas em áreas como agricultura, localizar animais perdidos, entre outros.

## 1.3 Contextualização

A incapacidade de rastrear e localizar animais perdidos pode resultar na angústia e preocupação para seus proprietários. Nesse contexto, surge a necessidade de uma solução eficaz que permita a geolocalização precisa desses animais, aumentando drasticamente as chances de sucesso.

## 1.4 Problemas Identificados

- **Precisão da Localização:** A precisão da geolocalização é crucial para a utilidade dos dados coletados. Problemas de precisão podem resultar em informações incorretas sobre a localização dos animais, comprometendo a eficácia.
- **Duração da Bateria:** Muitos dispositivos de rastreamento possuem restrições de tamanho e peso, o que pode limitar o tamanho da bateria. Isso pode resultar em autonomia limitada, especialmente se os dispositivos consomem energia rapidamente.
- **Conforto e Bem-Estar dos Animais:** A fixação de dispositivos de rastreamento nos animais deve ser realizada de forma a não causar desconforto ou afetar o seu bem-estar. Problemas de segurança e de saúde dos animais podem surgir se os dispositivos não forem projetados e colocados adequadamente.
- **Interferência de Sinal:** Ambientes urbanos e naturais podem ter interferências de sinal que afetam a qualidade das transmissões. Problemas de conectividade e perda de dados podem ocorrer se o sinal for bloqueado ou distorcido.
- **Custos:** A tecnologia de rastreamento de alta qualidade pode ser cara, o que limita sua aplicação em projetos com orçamentos restritos.

## 1.5 Objetivos

- **Desenvolver protótipos de dispositivos de geolocalização:** Criar diferentes protótipos de dispositivos de geolocalização de animais utilizando tecnologia LoRa e explorar diversas configurações e sensores para avaliar a sua viabilidade e eficácia.
- **Comparar autonomias:** Realizar cálculos para comparar a autonomia dos diversos protótipos desenvolvidos. Analisar o consumo de energia e identificar como diferentes parâmetros afetam a vida útil da bateria.
- **Monitorizar a localização de animais:** Implementar sistemas de monitorização para rastrear a localização de animais em tempo real ou em intervalos regulares.
- **Obter conhecimento sobre placas de desenvolvimento:** Investigar e documentar as características únicas de cada placa de desenvolvimento utilizada nos protótipos. Compreender como essas características impactam no desempenho geral e na eficiência energética.
- **Fornecer informações para decisões futuras:** Chegar a conclusões que orientem futuros projetos de monitorização e rastreamento de animais. Selecionar as melhores soluções com base em cenários específicos e características das espécies.

## 1.6 Estrutura

A estrutura deste relatório segue as normas de formatação de um relatório final.

No capítulo 1 apresenta uma introdução ao tema, onde se pode ler sobre o que trata este relatório, o que motivou a ser feito, o seu contexto os objetivos que se propõe a atingir e os problemas que pretende resolver.

No capítulo 2 resume e analisa trabalhos anteriores, pesquisas relevantes e desenvolvimentos recentes relacionados ao tema.

No capítulo 3 mostra a análise e o desenho feitos sobre o projeto passando pelos requesitos, diagramas e lógica do próprio software a ser implementado.

No capítulo 4 fala-se sobre o desenvolvimento do projeto onde explora os fundamentos sobre os quais assenta a base de todo o projeto, e mostra também uma análise aos equipamentos utilizados.

No capítulo 5 comprehende os diversos protótipos desenvolvidos bem como as suas ligações e características e a restante implementação como uma gateway feita a medida e também imagens de um back office de uma possível futura aplicação.

No capítulo 6 apresenta os cálculos de autonomia de cada protótipo para se poder fazer uma comparação das placas de desenvolvimento exploradas.

No capítulo 7 conclui-se o relatório e lista algumas possíveis melhorias.



## Estado da arte

No mundo moderno, os animais de estimação desempenham um papel crucial na vida das pessoas. No entanto, a perda de animais de estimação é uma preocupação comum para muitos proprietários. A geolocalização de animais domésticos perdidos tornou-se uma área de pesquisa e desenvolvimento fundamental, proporcionando maneiras inovadoras de rastrear e localizar animais desaparecidos.

As tecnologias mais usadas para combater este problema são RFID, GPS e IoT. Na tecnologia RFID geralmente é usado dispositivos na coleira ou sob a pele que podem ser lidos por dispositivos de leitura em uma curta distância, no entanto é algo limitado no que toca a localizar animais perdidos devido à sua escala limitada. Já as tecnologias GPS e IoT mostram-se valiosas na localização de animais perdidos, juntas complementam-se para desempenhar um papel fundamental no desenvolvimento de novos dispositivos. Oferecem uma solução precisa e em larga escala enviando dados para a nuvem, onde os proprietários podem acessá-los em tempo real, melhorando a capacidade de resposta.

Apesar dos avanços notáveis, ainda existem desafios a serem superados. Questões de tamanho, custos e autonomia são alguns dos obstáculos a serem enfrentados. No entanto alguns projetos apresentam soluções com propostas muito inovadoras tais como o Pet Tracker[10] que apresenta dimensões muito reduzidas, peso leve, e autonomia de um mês com bateria recarregável, aplicável até em gatos.



# Análise e Desenho

## 3.1 Requisitos

### 3.1.1 Requisitos Funcionais

Código	Nome	Descrição
R.01	Pedir estado do dispositivo	Deve ser capaz de pedir o estado do dispositivo
R.02	Receber estado do dispositivo	Deve ser capaz de receber o estado do dispositivo
R.03	Obter a geolocalização do dispositivo	Deve ser capaz de obter a geolocalização do dispositivo
R.04	Enviar a geolocalização do dispositivo	Deve ser capaz de enviar a geolocalização do dispositivo
R.05	Enviar dados em formato JSON	Deve ser capaz de enviar dados em formato JSON
R.06	Receber dados em formato JSON	Deve ser capaz de receber dados em formato JSON
R.07	Entrar em modo sleep	Deve conseguir entrar em modo sleep
R.08	Conexão LoRa	Deve conseguir conectar-se com outros dispositivos LoRa

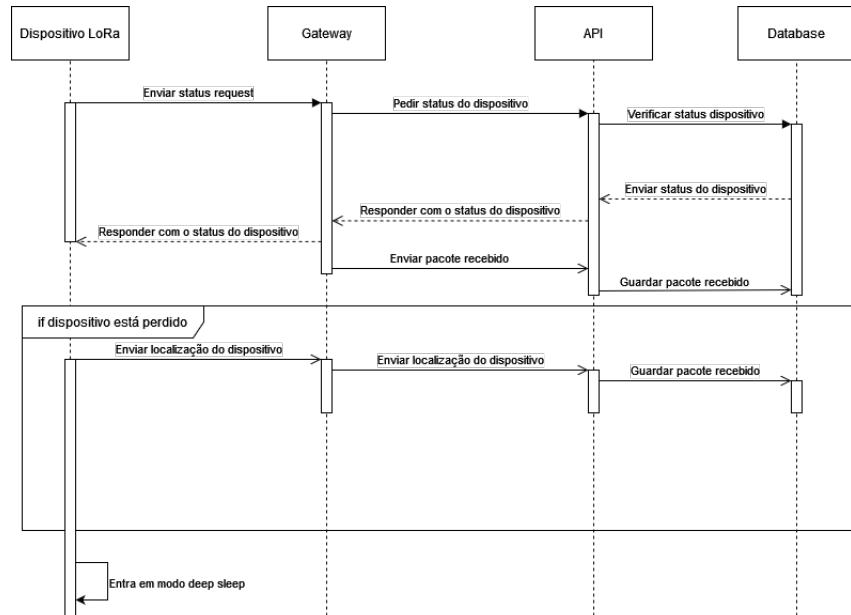
### 3.1.2 Requisitos Não Funcionais

**Eficiência Energética:** Os dispositivos LoRa devem ser eficientes em termos de consumo de energia para garantir uma vida útil prolongada da bateria.

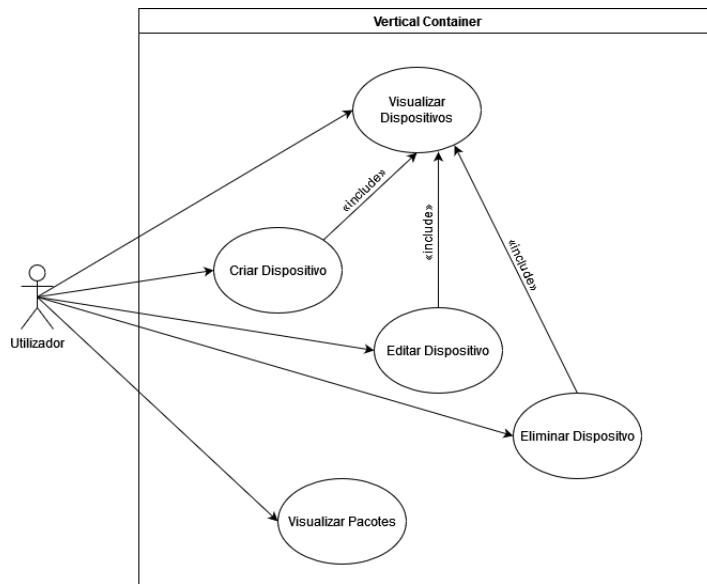
**Confiabilidade:** O sistema deve ser altamente confiável para garantir que as solicitações de verificação de perda sejam tratadas com sucesso.

**Escalabilidade:** O sistema deve ser escalável para acomodar um grande número de dispositivos LoRa, à medida que mais animais são registrados.

## 3.2 Diagramas

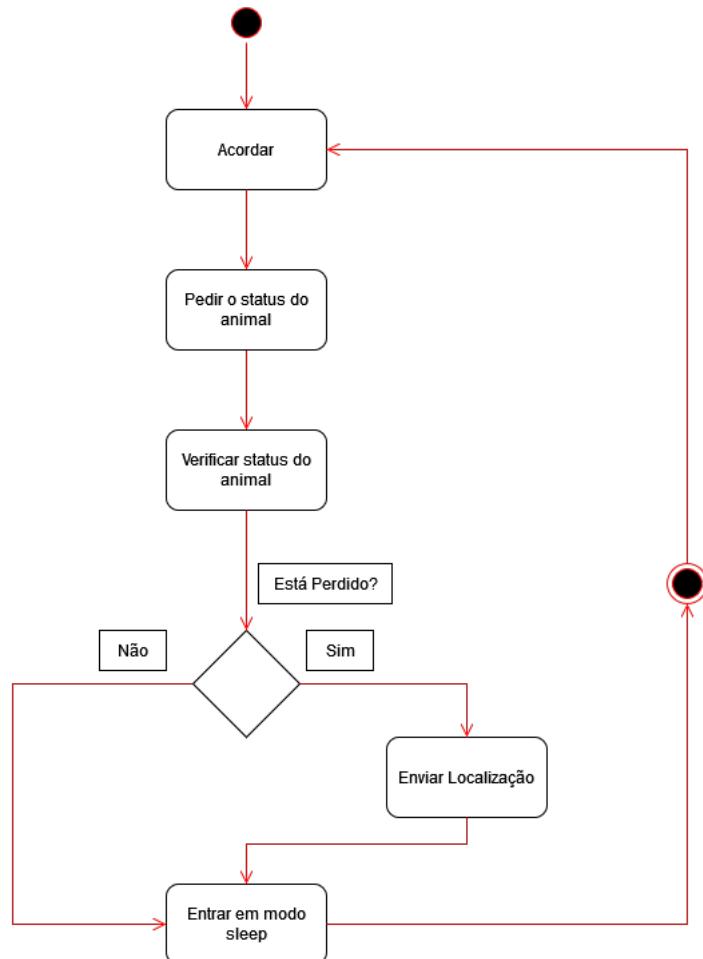


**Figura 3.1 – Diagrama de Sequência**

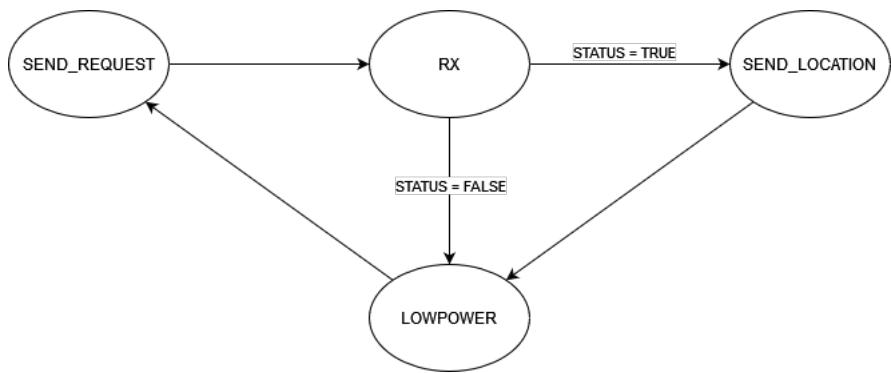


**Figura 3.2 – Diagrama de Casos de Usos da aplicação**

### 3.3 Lógica



**Figura 3.3 – Diagrama de atividade**



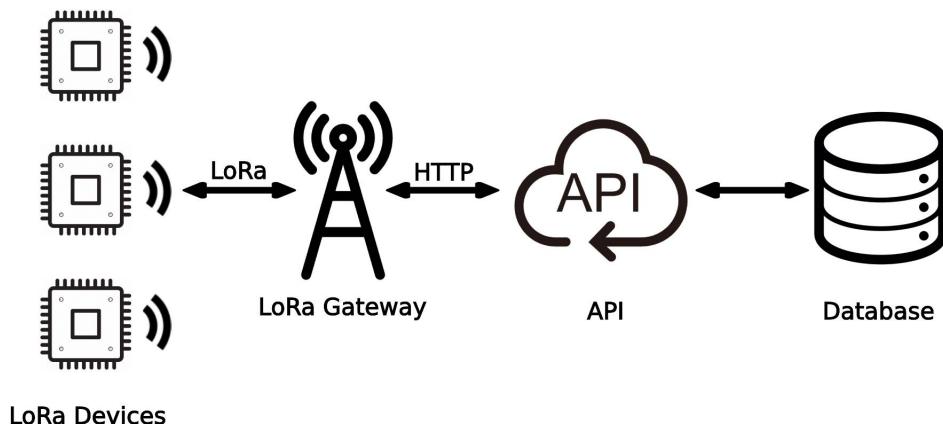
**Figura 3.4** – Máquina de estados

Para explicar a lógica do software é preciso perceber que este foi feito a pensar numa máquina de estados. Existem vários estados nomeadamente um estado para receber informação(RX), um estado para enviar um request (SEND\_REQUEST), um estado para enviar a localização do animal perdido (SEND\_LOCATION) e um estado para dormir (LOWPOWER). No início de uma novo ciclo o dispositivo acorda e altera o seu estado para SEND\_REQUEST onde envia um request à API para saber o status do animal. O estado volta a ser alterado desta vez para RX, onde o dispositivo fica a esperar de receber a informação sobre o status e dependendo da resposta que obtém o dispositivo pode ter um de dois comportamentos. No caso de o animal não estar perdido o dispositivo altera o seu estado para LOWPOWER onde entra em modo sleep, caso o animal esteja perdido o dispositivo altera o seu estado para SEND\_LOCATION onde envia a sua localização e posteriormente entra em modo sleep.



# Desenvolvimento

## 4.1 Arquitetura



**Figura 4.1 – Arquitetura da Aplicação**

1. Os dispositivos LoRa recolhem dados periodicamente que são posteriormente transmitidos
2. As gateways LoRa encaminham os dados rececionados na sua área de cobertura para o servidor utilizando o protocolo Hypertext Transfer Protocol (HTTP).
3. Para facilitar a manipulação dos dados tanto a informação presente nos pacotes LoRa e nas respostas da API têm o formato Javascript Object Notation (JSON).
4. A informação fica acessível através de uma API para interação com utilizadores ou sistemas armazenando-a numa base de dados.

## 4.2 Protocolos de Comunicação

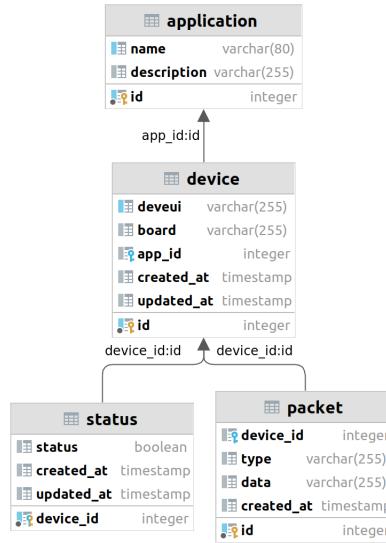
- A API web usa por padrão o protocolo HTTP para a comunicação de mensagens
- Estas mensagens de resposta têm o formato JSON.
- Os dispositivos LoRa vão enviar pacotes em formato JSON.

```
{  
    "devEUI": "00:00:00:00:00:00:00:01",  
    "application": "Water Tank",  
    "board": "Cubecell 1/2AA Node",  
    "data":  
        {  
            "distance": 378  
        }  
}
```

**Figura 4.2** – Exemplo de pacote LoRa em formato JSON

- "**devEUI**" Identificação do dispositivo através de um endereço de 64 bits.
- "**application**" Nome da aplicação ao qual está associado.
- "**board**" Nome da placa de desenvolvimento.
- "**data**" informação contida no pacote.

## 4.3 Base de Dados



**Figura 4.3 – Base de Dados**

A base de dados apresenta um modelo relacional PostgreSQL[11] contida em Docker[12].

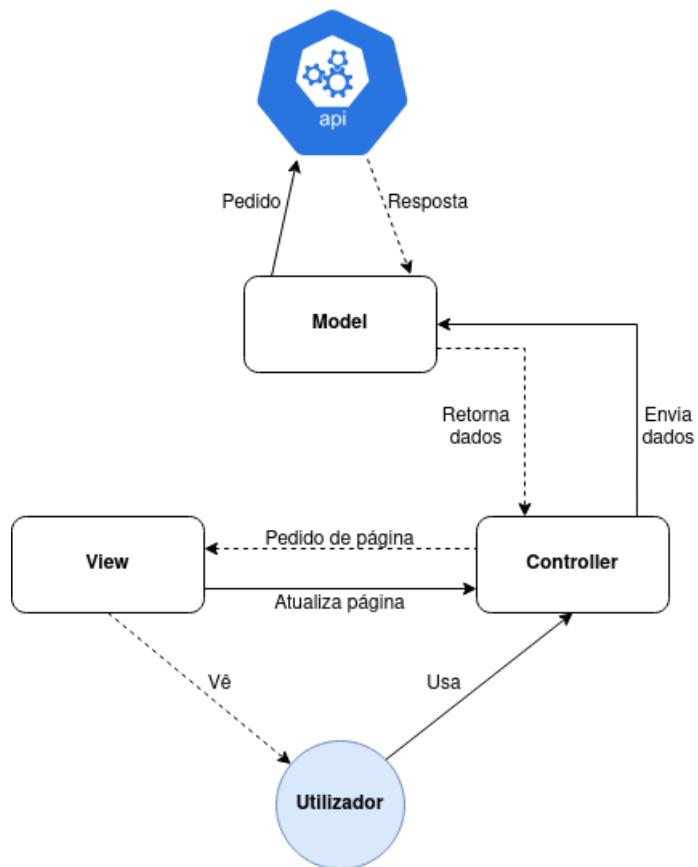
- **application** Cada aplicação tem um id como primary key, um nome e uma descrição.
- **device** Cada dispositivo tem um id como primary key, um endereço, um nome, um id da aplicação e informação sobre quando foi criado e atualizado.
- **status** Cada status tem um id de dispositivo como primary key nome e informação sobre quando foi criado e atualizado.
- **packet** Cada status tem um id de dispositivo como primary key, um tipo de pacote, o conteúdo do pacote e informação sobre quando foi criado.

## 4.4 MVC

A arquitetura MVC [13] é um padrão de design amplamente utilizado na engenharia de software para organizar e estruturar aplicações de forma modular e eficiente. Separa a aplicação em três componentes principais:

- **Model (Modelo):** O Modelo representa a lógica de negócios da aplicação. Lida com a gestão de dados, regras de negócios e a interação com a base de dados. É responsável por armazenar e recuperar informação, bem como por processar os dados para fornecer à aplicação as informações necessárias.
- **View (Visão):** A Visão é a camada responsável pela apresentação dos dados aos utilizadores. Ela lida com a interface, exibindo informações de forma comprehensível e interativa.
- **Controller (Controlador):** O Controlador atua como intermediário entre o Modelo e a Visão. Recebe as entradas do utilizador, e direciona essas ações para o Modelo apropriado. O Controlador também atualiza a Visão quando o Modelo é modificado.

A principal vantagem da arquitetura MVC é a separação clara de responsabilidades, o que torna o código mais organizado, fácil de manter e escalar. Isso também permite que diferentes desenvolvedores trabalhem em partes diferentes da aplicação simultaneamente, desde que sigam as convenções estabelecidas pelo padrão.



**Figura 4.4** – Diagrama MVC aplicado usando uma API

## 4.5 Software

### 4.5.1 Ambientes de desenvolvimento

- **Arduino AVR Boards [14]**

É um ambiente de desenvolvimento necessário para um conjunto de placas e configurações dentro da plataforma Arduino que são baseadas em microcontroladores Advanced Virtual RISC (AVR).

- **ESP32 [15]**

É um ambiente de desenvolvimento necessário para um conjunto de placas e configurações dentro da plataforma Arduino que são baseadas em microcontroladores ESP32.

- **Heltec ESP32 Series Dev-boards [16]**

É um ambiente de desenvolvimento necessário para um conjunto de placas de desenvolvimento da Heltec e configurações dentro da plataforma Arduino que são baseadas em microcontroladores ESP32.

- **CubeCell Development Framework [17]**

É um ambiente de desenvolvimento necessário para um conjunto de placas de desenvolvimento da Heltec e configurações dentro da plataforma Arduino que são baseadas em microcontroladores ASR650x.

## 4.5.2 Bibliotecas

- **LowPower.h [18]**

É uma biblioteca útil para otimizar o consumo de energia em projetos eletrônicos e de IoT, ajudando a prolongar a vida útil da bateria e a tornar os dispositivos mais eficientes em termos energéticos

- **Arduino\_JSON.h [19] e ArduinoJson.h[20]**

É uma biblioteca poderosa para trabalhar com dados JSON em projetos Arduino, tornando mais fácil a integração com fontes externas de dados e serviços da web, bem como a manipulação e formatação de dados JSON em seus próprios dispositivos e aplicações

- **TinyGPSPlus.h[21]**

É uma biblioteca para dispositivos Arduino e outras plataformas de microcontroladores que permite a fácil manipulação de informações de GPS. Ela é usada para processar e interpretar dados provenientes de um receptor GPS

- **SPI.h[22]**

É uma biblioteca para programação de microcontroladores e placas Arduino que serve para controlar dispositivos que utilizam a comunicação Serial Peripheral Interface (SPI). O SPI é um protocolo de comunicação serial síncrona que permite a transferência rápida de dados entre um microcontrolador (mestre) e dispositivos periféricos (escravos) como sensores, displays, módulos de memória, e muitos outros dispositivos.

- **LoRa.h[23]**

É uma biblioteca comumente usada em projetos que envolvem comunicação LoRa. É particularmente associada a dispositivos baseados em placas Arduino e outros microcontroladores que usam módulos de rádio LoRa para estabelecer comunicações de longa distância

- **SoftwareSerial.h[24] e ESPSoftwareSerial[25]**

É uma biblioteca padrão no ambiente de desenvolvimento Arduino e é utilizada para criar portas seriais virtuais em microcontroladores baseados na plataforma Arduino. Esta é especialmente útil quando se deseja estabelecer comunicação serial com dispositivos externos (como sensores, módulos GPS, módulos Bluetooth, outros microcontroladores, etc.) que exigem uma porta serial adicional, mas o Arduino tem um número limitado de portas seriais hardware.

- **RTClib.h[26]**

É uma biblioteca normalmente usada em programação para microcontroladores, como o Arduino, para lidar com dispositivos de relógio em tempo real Real-Time Clock (RTC). Ela permite interagir com RTCs para obter e definir datas e horários com precisão, independentemente da alimentação principal do dispositivo.

- **LoRa\_APP.h[27]**

É uma biblioteca para enviar e receber dados através de rádios LoRa nas placas de desenvolvimento CubeCell.

## 4.6 Equipamento Utilizado

Necessitou-se de placas de desenvolvimento de baixo consumo e conectividade LoRa, para esse fim escolheram-se placas que funcionassem a nível lógico de 3.3 Volts e desenhadas para aplicações IoT e outras de uso geral com um módulo LoRa.

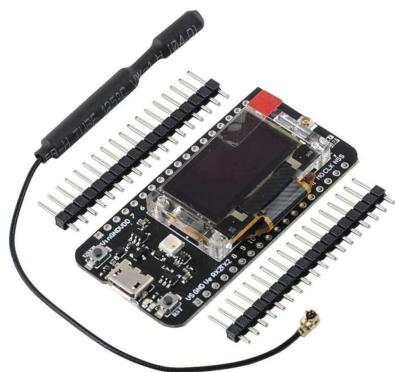
### 4.6.1 Placas de desenvolvimento



**Figura 4.5** – Heltec Cubecell 1/2AA Node

Heltec Cubecell 1/2AA Node	
<b>Microprocessador</b>	ASR605x (ARM Cortex M0+ Core e SX1262)
<b>Nível lógico</b>	3.3V
<b>Módulo Rádio</b>	SX1262
<b>Compatibilidade</b>	LoRaWan (1.0.2) e Arduíno Framework
<b>Consumo deepSleep</b>	3.5 uA (3.3V header) 11 uA (VBAT)
<b>Consumo Rx</b>	10 mA
<b>Consumo Tx</b>	70 mA (10dB Tx Power) 90 mA (14dB Tx Power) 100 mA (17dB Tx Power) 105 mA (20dB Tx Power)
<b>Características</b>	Encaixe para bateria 1/2 AA com mudança de alimentação automática quando conectado no USB Interface USB-Serial Interface Micro USB com proteção ESD, curto circuito e RF shielding 56.6 x 24 x 21.5 mm

**Tabela 4.1** – Especificações Heltec Cubecell 1/2AA Node[1]



**Figura 4.6** – Heltec CubeCell GPS-6502

<b>Heltec CubeCell GPS-6502</b>	
<b>Microprocessador</b>	ASR605x (ARM Cortex M0+ Core e SX1262)
<b>Nível lógico</b>	3.3V
<b>Módulo Rádio</b>	SX1262
<b>Compatibilidade</b>	LoRaWan (1.0.2) e Arduíno Framework
<b>Consumo deepSleep</b>	21 uA (3.3V header) 27 uA (VBAT)
<b>Consumo Rx</b>	10 mA
<b>Consumo Tx</b>	70 mA (10dB Tx Power) 90 mA (14dB Tx Power) 100 mA (17dB Tx Power) 105 mA (20dB Tx Power)
<b>Características</b>	Interface SH1.25-2 com sistema integrado de para gerir baterias de litíio (Gestão de carga e descarga, proteção de sobrecarga , deteção de bateria de mudança automática de alimentação entre bateria e USB) Sistema integrado de gestão de painel solar podendo ser conectado diretamente num painel solar de 5.5~7V Ecrã integrado OLED 0.96 polegadas 128*64 Modulo GPS Air530 integrado Interface USB-Serial Interface Micro USB com proteção ESD, curto circuito e RF shielding Consumo GPS = 36.7 mA 55.9 x 27.9 x 9.5 mm

**Tabela 4.2** – Especificações Heltec CubeCell GPS-6502[2]



**Figura 4.7** – Heltec WiFi LoRa 32 V2

<b>Heltec WiFi LoRa 32 v2</b>	
<b>Microprocessador</b>	ESP32 (240MHz Tensilica LX6 dual-core + ULP core e SX1276)
<b>Nível lógico</b>	3.3V
<b>Módulo Rádio</b>	SX1276
<b>Compatibilidade</b>	LoRaWan (1.0.2) e Arduíno Framework
<b>Consumo deepSleep</b>	800 uA
<b>Consumo Rx</b>	30 mA
<b>Consumo Tx</b>	50 mA (10dB Tx Power) 60 mA (12dB Tx Power) 110 mA (15dB Tx Power) 130 mA (20dB Tx Power)
<b>Características</b>	Interface SH1.25-2 com sistema integrado de para gerir baterias de litio (Gestão de carga e descarga, proteção de sobrecarga, deteção de bateria de mudança automática de alimentação entre bateria e USB) Ecrã integrado OLED 0.96 polegadas 128*64 WiFi e Bluetooth com antenas integradas Interface USB-Serial Interface Micro USB com proteção ESD, curto circuito e RF shielding Interface para antena LoRa (IPEX) 51 x 55.5 x 10.6 mm

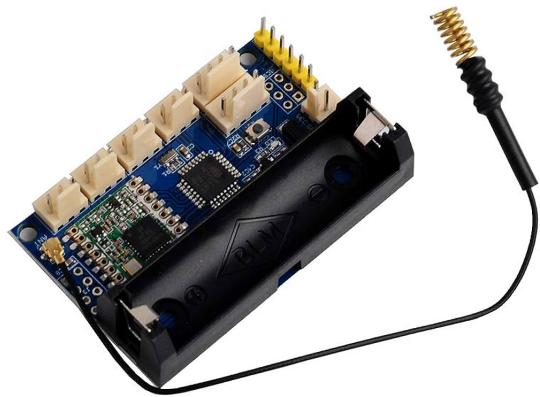
**Tabela 4.3** – Especificações Heltec WiFi LoRa 32 v2[3]



**Figura 4.8** – TTGO LoRa32 V1

TTGO LoRa32 V1	
<b>Microprocessador</b>	ESP32 (240MHz Tensilica LX6 dual-core + ULP core e SX1276)
<b>Nível lógico</b>	3.3V
<b>Módulo Rádio</b>	SX1276
<b>Compatibilidade</b>	LoRaWan (1.0.2) e Arduíno Framework
<b>Consumo deepSleep</b>	~600uA
<b>Consumo Rx</b>	Não descrito
<b>Consumo Tx</b>	Não descrito
<b>Características</b>	Interface SH1.25-2 com sistema integrado de gerir baterias de litíio (Gestão de carga e descarga, proteção de sobrecarga, deteção de bateria de mudança automática de alimentação entre bateria e USB) Ecrã integrado OLED 0.96 polegadas 128*64 WiFi e Bluetooth com antenas integradas Interface USB-Serial Interface para antena LoRa (IPEX) 51.4 x 25.2 mm

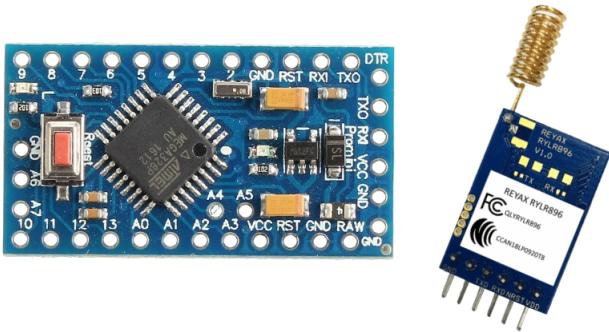
**Tabela 4.4** – Especificações TTGO LoRa32 V1[4]



**Figura 4.9** – LoRa Radio Node v1.0

LoRa Radio Node v1.0	
<b>Microprocessador</b>	ATmega328P (8MHz)
<b>Nível lógico</b>	3.3V
<b>Módulo Rádio</b>	RFM95 (SX1276)
<b>Compatibilidade</b>	Arduíno Framework
<b>Consumo deepSleep</b>	~300uA
<b>Consumo Rx</b>	~40 mA
<b>Consumo Tx</b>	~120 mA (20dB Tx Power)
<b>Características</b>	Interface para bateria 14500 LiPo 3.7V
40 x 60 mm	

**Tabela 4.5** – Especificações LoRa Radio Node v1.0[5]

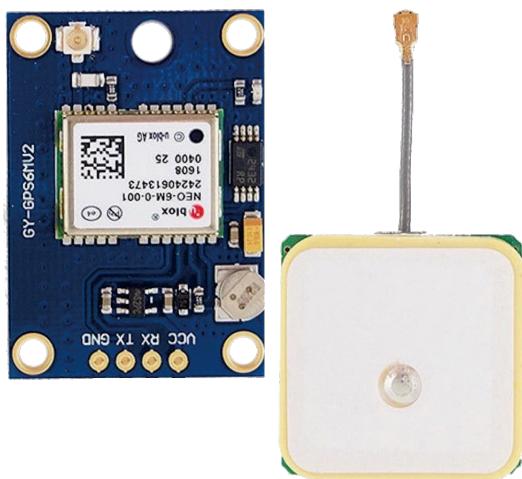


**Figura 4.10 – Arduino Pro Mini + Reyax RYLR896**

<b>Arduino Pro Mini + Reyax RYLR896</b>	
<b>Arduino Pro Mini</b>	
<b>Microprocessador</b>	ATmega328P (8MHz)
<b>Nível lógico</b>	3.3 V
<b>Consumo acordado</b>	~6 mA
<b>Consumo deepSleep</b>	~300 uA
<b>Modulo LoRa Reyax RYLR896</b>	
<b>Nível lógico</b>	3.3 V
<b>Módulo rádio</b>	SX1276
<b>Consumo Sleep</b>	0.5 uA
<b>Consumo Rx</b>	16.5mA
<b>Consumo Tx</b>	43 mA (15dB Tx Power)
<b>Características</b>	Controlável através de comandos AT Comunicação Serial UART 42.5 x 18.36 x 5.5 mm 33 x 18 x 2 mm

**Tabela 4.6 – Especificações Arduino Pro Mini[6] + Reyax RYLR896[7]**

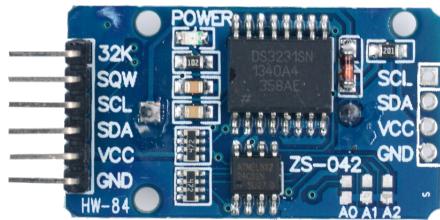
## 4.6.2 Sensores



**Figura 4.11** – Módulo GPS Neo6-M

Modulo GPS NEO-6M	
<b>Nível de operação</b>	3.3V
<b>Baudrate padrão</b>	9600bps
<b>Consumo médio</b>	45 mA
<b>Tempo de captação</b>	1~27s
<b>Protocolos de comunicação</b>	NMEA, UBX Binary, RTCM
<b>Características</b>	Comunicação Serial UART Interface para antena GPS (IPEX) 25 x 35mm

**Tabela 4.7** – Especificações GPS Neo6-M [8]



**Figura 4.12** – Módulo RTC DS3231

<b>Modulo RTC DS3231</b>	
<b>Nível de operação</b>	3.3V
<b>Consumo médio</b>	200 uA (3.3V) 300 uA (5V)
<b>Características</b>	Interface de comunicação I2C Modulo capaz de contar Segundos, Minutos, Horas, Dia do Mês, Mês , Dia da Semana e Ano, com validação de ano bissexto até 2100 2 alarmes programáveis Sensor de temperatura integrado Memória EEPROM integrada 25.5 x 16.4 x 12.5mm

**Tabela 4.8** – Módulo RTC DS3231 [9]

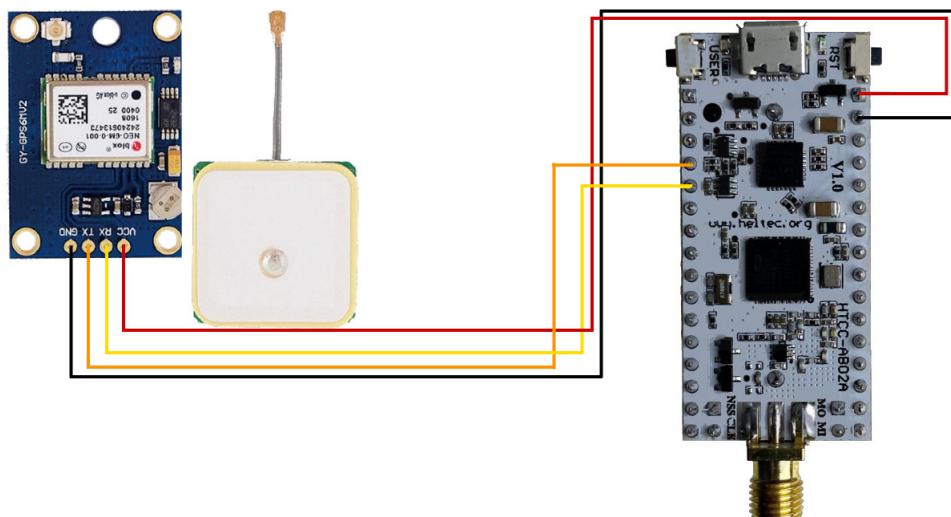
# Implementação

## 5.1 Protótipos

### 5.1.1 Cubecell

#### Módulo GPS

- O módulo possuí ligação do pino VCC e do pino GND ao pino VDD (3.3V) e ao pino GND da placa respetivamente.
- Os pinos Tx e Rx do módulo, ligam nos pinos Rx2 e Tx2 da placa respetivamente.
- O pino VCC do módulo também pode ser conectado num dos pinos VEXT da placa, pino esse que pode ser ligado ou desligado por software.

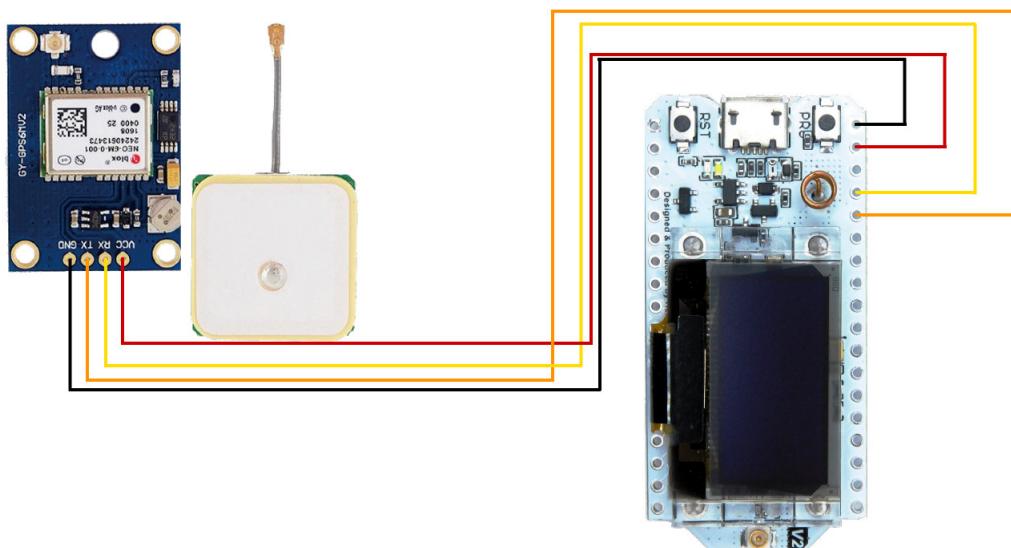


**Figura 5.1 – Esquemática - Cubecell 1/2AA**

## 5.1.2 ESP32

### Módulo GPS

- O módulo possuí ligação do pino VCC e do pino GND ao pino VDD (3.3V) e ao pino GND da placa respetivamente.
- Os pinos Tx e Rx do módulo, ligam nos pinos GPIO37 e GPIO36 da placa respetivamente.
- O pino VCC do módulo também pode ser conectado num dos pinos VEXT da placa, pino esse que pode ser ligado ou desligado por software.



**Figura 5.2 – Esquemática - ESP32**

### 5.1.3 LoRa Radio Node v1

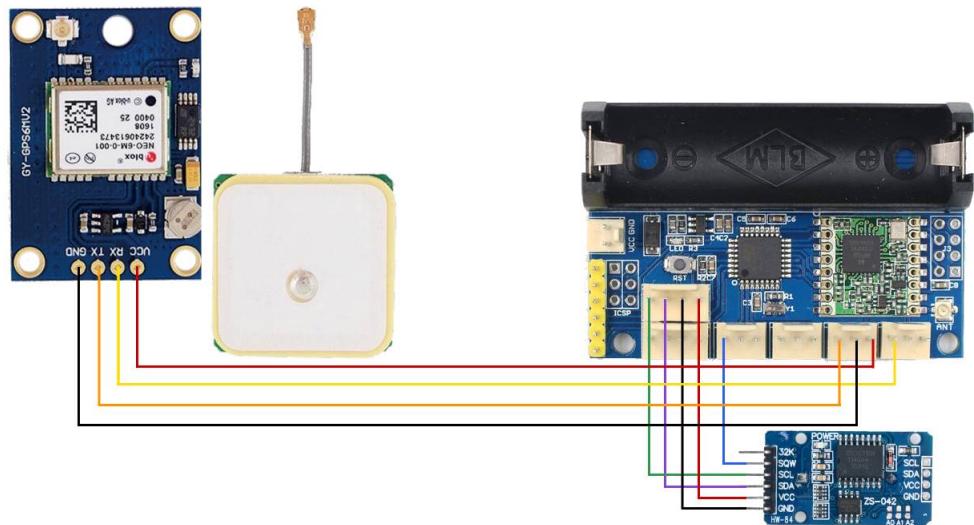
Neste caso existe uma necessidade de interromper o modo deep sleep do dispositivo. Como este não apresenta uma função de deep sleep utilizou-se uma biblioteca que permite desligar alguns componentes da placa para apresentar consumos mais baixos sendo interrompido utilizando um módulo RTC externo.

#### Módulo RTC

- O módulo RTC necessita de uma comunicação I2C para isso conecta-se os pinos SDA e SCL do relógio nos respetivos pinos da placa de desenvolvimento.
- O pino VCC e o pino GND é conectado no pino 3V3 e no GND da placa respetivamente.
- O pino SQW no pino digital 3. Este é responsável por interromper o modo deep sleep da placa.

#### Módulo GPS

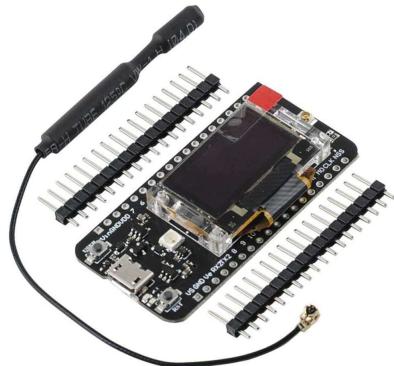
- O módulo GPS possuí ligação do pino VCC e do pino GND ao pino VDD (3.3V) e ao pino GND da placa respetivamente.
- Os pinos Tx e Rx do módulo, ligam nos pinos A0 e A1 da placa respetivamente.
- O pino VCC do módulo também pode ser conectado num dos pinos VEXT da placa, pino esse que pode ser ligado ou desligado por software.



**Figura 5.3 – Esquemática - LoRa Radio Node v1**

### 5.1.4 Heltec CubeCell GPS-6502

Este protótipo possui um módulo GPS e um módulo RTC portanto não há necessidade de fazer qualquer ligação. É uma vantagem em relação ás outras placas de senvolvimento no que toca ao tamanho de todo o sistema e também o seu GPS tem um consumo menor que um módulo à parte.



**Figura 5.4** – Esquemática - CubeCell GPS-6502

## 5.1.5 Arduino-Pro-Mini

Neste caso existe uma necessidade de interromper o modo deep sleep do dispositivo. Como este não apresenta uma função de deep sleep utilizou-se uma biblioteca que permite desligar alguns componentes da placa para apresentar consumos mais baixos sendo interrompido utilizando um módulo RTC externo.

### Módulo RTC

- O módulo necessita de uma comunicação I2C para isso conecta-se os pinos SDA e SCL do relógio pinos A4 e A5 da placa de desenvolvimento respetivamente.
- O pino VCC e o pino GND é conectado no pino 3V3 e no GND da placa respetivamente.
- O pino SQW no pino digital 2. Este é responsável por interromper o modo deep sleep da placa.

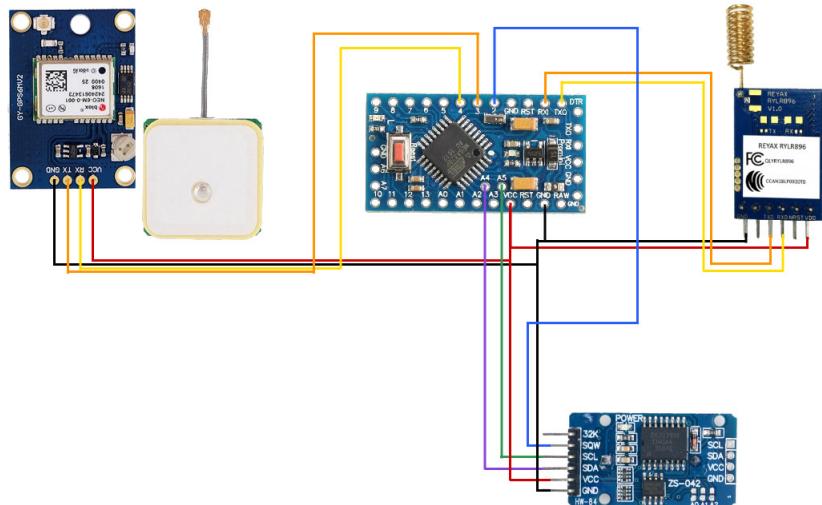
### Módulo GPS

- O módulo possuí ligação do pino VCC e do pino GND ao pino VDD (3.3V) e ao pino GND da placa respetivamente.
- Os pinos Tx e Rx do módulo, ligam nos pinos 3 e 4 da placa respetivamente.
- O pino VCC do módulo também pode ser conectado num dos pinos VEXT da placa, pino esse que pode ser ligado ou desligado por software.

### Módulo rádio REYAX RYLR896

- Conectam-se os pinos Tx e Rx nos pinos Rx e Tx da dispositivo respetivamente.
- O pino VCC e o pino GND do rádio é conectado no VCC e no GND do Arduino Pro Mini.

- Devido a utilizar-se o mesmo canal de comunicação para o módulo rádio e para o adaptador USB to Serial, sempre que se programar o dispositivo é necessário desconectar os pinos Tx e Rx do módulo rádio.



**Figura 5.5 – Esquemática - Arduino-Pro-Mini**

## 5.2 Gateway

É indispensável a presença de uma gateway para que os dispositivos LoRa possam comunicar com uma API. Essa gateway age como um intermediário entre os dispositivos. Para a criação dos protótipos, escolhemos a placa TTGO Lora32 V1 para desempenhar o papel de gateway. A escolha se deu devido à sua capacidade de se conectar ao WiFi, o que possibilita a comunicação por meio do protocolo HTTP com a API.

Quando a gateway recebe um pacote, ela envia uma solicitação HTTP POST para a API, com o objetivo de que o pacote seja armazenado na base de dados. No caso de o pacote ser uma solicitação de status do dispositivo, a gateway envia uma solicitação GET para a API a fim de obter o status do dispositivo e, posteriormente, envia um pacote contendo esse status.

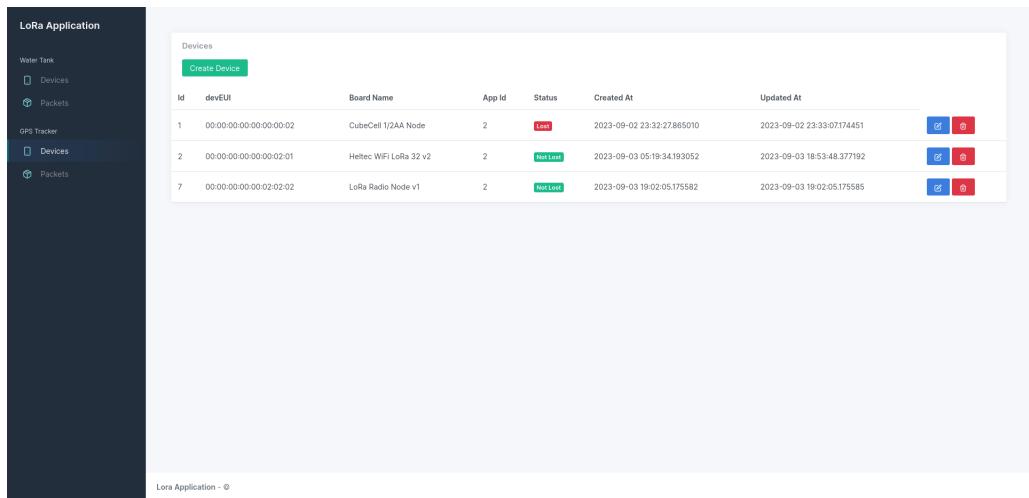
## 5.3 APP

No processo de construção da parte central da aplicação, empregamos a framework Flask[28], reconhecida por sua robustez e eficiência. Com o intuito de assegurar que a aplicação fosse de fácil uso e atraente visualmente, optamos por combinar duas tecnologias

complementares, o Admin Kit[29] e o Bootstrap[30]. No que diz respeito ao armazenamento e gestão de dados da aplicação, selecionamos o PostgreSQL[11] hospedado em um ambiente Docker[12], o que assegura uma gestão de dados eficaz.

### 5.3.1 Dispositivos da aplicação

Criou-se uma página onde é permitido criar editar e apagar dispositivos(algumas funções não estão implementadas) permitindo a sua visualização bem como os seus atributos.



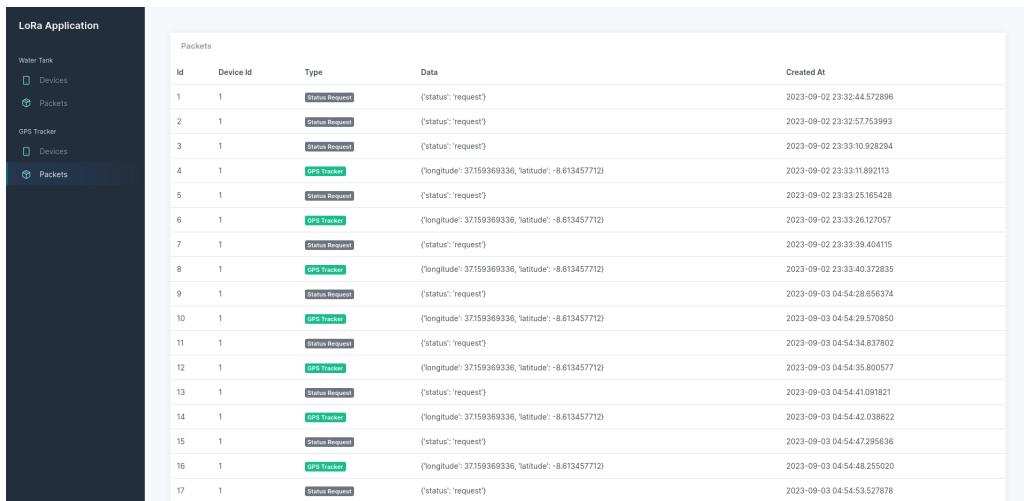
The screenshot shows the 'LoRa Application' interface with a dark sidebar on the left containing navigation links: Water Tank, Devices (selected), Packets, GPS Tracker, Devices (selected again), and Packets. The main content area is titled 'Devices' and features a 'Create Device' button. Below it is a table with columns: Id, devEUI, Board Name, App Id, Status, Created At, and Updated At. The table contains three rows of data:

Id	devEUI	Board Name	App Id	Status	Created At	Updated At	Action Buttons
1	00:00:00:00:00:00:02	CubeCell 1/2AA Node	2	Lost	2023-09-02 23:32:27.885010	2023-09-02 23:33:07.174451	 
2	00:00:00:00:00:02:01	Heltec WiFi LoRa 32 v2	2	Not Lost	2023-09-03 05:19:34.193052	2023-09-03 18:53:48.377192	 
7	00:00:00:00:02:02:02	LoRa Radio Node v1	2	Not Lost	2023-09-03 19:02:05.175582	2023-09-03 19:02:05.175585	 

**Figura 5.6 – Dispositivos da aplicação**

### 5.3.2 Pacotes da aplicação

Criou-se uma página que permite a visualização de pacotes e todos os seus atributos. Consegue-se perceber que tipo de pacote é e que informação está contida no mesmo bem como as datas da sua criação.



The screenshot shows a web application titled "LoRa Application". On the left, there is a sidebar with navigation links: "Water Tank" (Devices, Packets), "GPS Tracker" (Devices, Packets), and "Packets". The "Packets" link under "GPS Tracker" is highlighted. The main content area is titled "Packets" and contains a table with 17 rows of data. The columns are: Id, Device Id, Type, Data, and Created At. The "Type" column uses color-coded buttons to indicate the type of packet: "Status Request" (grey) and "GPS Tracker" (green). The "Data" column contains JSON-like strings representing the packet content. The "Created At" column shows dates and times from September 2023. The table has a light grey background with alternating row colors.

Packets				
Id	Device Id	Type	Data	Created At
1	1	Status Request	{"status": "request"}	2023-09-02 23:32:44.572896
2	1	Status Request	{"status": "request"}	2023-09-02 23:32:57.753993
3	1	Status Request	{"status": "request"}	2023-09-02 23:33:10.928294
4	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-02 23:33:11.892113
5	1	Status Request	{"status": "request"}	2023-09-02 23:32:25.165428
6	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-02 23:33:26.127057
7	1	Status Request	{"status": "request"}	2023-09-02 23:33:39.404115
8	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-02 23:33:40.372835
9	1	Status Request	{"status": "request"}	2023-09-03 04:54:28.656374
10	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-03 04:54:29.570850
11	1	Status Request	{"status": "request"}	2023-09-03 04:54:34.837802
12	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-03 04:54:35.800577
13	1	Status Request	{"status": "request"}	2023-09-03 04:54:41.091821
14	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-03 04:54:42.038622
15	1	Status Request	{"status": "request"}	2023-09-03 04:54:47.295636
16	1	GPS Tracker	{"longitude": 37159369336, "latitude": -8.613457712}	2023-09-03 04:54:48.255020
17	1	Status Request	{"status": "request"}	2023-09-03 04:54:53.527878

**Figura 5.7 – Pacotes da aplicação**

## 5.4 Dificuldades na implementação

Durante a implementação deste projeto, alguns desafios surgiram, mas apenas alguns foram ultrapassados com sucesso. Um dos maiores problemas acabou por ser a receção de dados no protótipo Arduino-Pro-Mini com o REYAX RYLR896, onde não se conseguiu chegar à conclusão se o problema consistia no módulo REYAX RYLR896 ser proprietário ou não. Problema esse que acabou por ficar para trás sendo substituído por outro protótipo. Também se registou problemas de estabilidade na placa LoRa Radio Node v1. Durante o período de testes registou-se perdas de pacotes frequentes cuja causa também não se conseguiu concluir.

# Resultados

## 6.1 Fórmulas

Para saber o tempo de transmissão de um pacote LoRa, foram feitos os cálculos abaixo, como descritos na datasheet dos módulos rádio[31].

$$ToA = \frac{2^SF}{BW} * N_{symbol}$$

$$N_{symbol} = N_{symbolpreamble} + 4.25 + 8 + ceil\left(\frac{\max(8+N_{bytepayload}+N_{bitCRC}-4*SF+N_{symbolheader})^0}{4*SF}\right) * (CR + 4)$$

$N_{symbolpreamble}$  = é o comprimento do preâmbulo.

$N_{bytepayload}$  = é o número de bytes do payload do pacote.

$N_{bitCRC}$  = se o CRC estiver ativo são 16 bits, se não 0 bits.

$SF$  = é o fator de expansão, representa o número de bits da modulação.

$N_{symbolheader}$  = 20 se o header for explícito se não 0.

$CR$  = 1, 2, 3, 4 para os respectivos coding rate 4/5, 4/6, 4/7 ou 4/8.

$BW$  = é a largura de banda em kHz

Para a realização dos protótipos estes foram os parâmetros do rádio LoRa escolhidos:

$$N_{symbol\ preamble} = 8$$

$$N_{byte\ payload} = 255$$

$$N_{bit_{CRC}} = 0$$

$$SF = 7$$

$$N_{symbol_{header}} = 0$$

$$CR = 1$$

$$BW = 125$$

Para um pacote de 255 bytes nestes parâmetros, o tempo de transmissão está demonstrado abaixo:

$$ToA = 394.496ms$$

$$Bt = \frac{B_c}{\frac{I_{sts}}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}} + \frac{I_{tr}t_r}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}} + \frac{I_{ctc}}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}} + \frac{I_{ptp}}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}} + \frac{I_{rtr}}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}} + \frac{I_{gpstgps}}{t_s + t_{tr} + t_c + t_p + t_r + t_{gps}}}$$

- $B_l$  = Tempo de vida da bateria
- $B_c$  = Capacidade da bateria
- $I_s$  = Consumo de corrente do dispositivo em Modo de Repouso
- $I_{tr}$  = Consumo de corrente do dispositivo ao transmitir status request
- $I_c$  = Consumo de corrente do dispositivo ao coletar dados
- $I_p$  = Consumo de corrente do dispositivo ao processar dados
- $I_r$  = Consumo de corrente do dispositivo ao receber dados
- $I_{gps}$  = Consumo de corrente do dispositivo ao transmitir a geolocalização
- $t_s$  = Tempo gasto em Modo de Repouso (por ciclo)
- $t_{tr}$  = Tempo gasto na transmissão status request (por ciclo)
- $t_c$  = Tempo gasto na coleta de dados (por ciclo)
- $t_p$  = Tempo gasto no processamento de dados (por ciclo)
- $t_r$  = Tempo gasto a receber dados (por ciclo)
- $t_{gps}$  = Tempo gasto a transmitir a geolocalização (por ciclo)

## 6.2 Cálculos

### 6.2.1 CubeCell 1/2AA Node

Variável	Valores
<b>Bc</b>	1200 mAh
<b>Is</b>	11 uA
<b>Itr</b>	105 mA
<b>Ic</b>	12 mA(Placa) + 45 mA(GPS) = 57 mA
<b>Ip</b>	12 mA
<b>Ir</b>	10 mA
<b>Igps</b>	105 mA
<b>ts</b>	1 hora
<b>ttr</b>	394.496 ms
<b>tc</b>	1s
<b>tp</b>	1s
<b>tr</b>	3s
<b>tgps</b>	394.496 ms

Se o dispositivo nunca estiver perdido com ciclos de 1 hora

**Bt** ~ 35159 horas ~ 1465 dias ~ 4 anos

Se o dispositivo sempre estiver perdido com ciclos de 10 minutos

**Bt** ~ 3858 horas ~ 161 dias

Se o dispositivo sempre estiver perdido com ciclos de 30 minutos

**Bt** ~ 10746 horas ~ 447 dias ~ 1.2 anos

## 6.2.2 CubeCell GPS-6502

Variável	Valores
<b>Bc</b>	1200 mAh
<b>Is</b>	27 uA
<b>Itr</b>	105 mA
<b>Ic</b>	$12 \text{ mA(Placa)} + 36.7 \text{ mA(GPS)} = 48.7 \text{ mA}$
<b>Ip</b>	12 mA
<b>Ir</b>	10 mA
<b>Igps</b>	105 mA
<b>ts</b>	1 hora
<b>ttr</b>	394.496 ms
<b>tc</b>	1s
<b>tp</b>	1s
<b>tr</b>	3s
<b>tgps</b>	394.496 ms

Se o dispositivo nunca estiver perdido com ciclos de 1 hora

**Bt**  $\approx$  23947 horas  $\approx$  998 dias  $\approx$  2.7 anos

Se o dispositivo sempre estiver perdido com ciclos de 10 minutos

**Bt**  $\approx$  3831 horas  $\approx$  160 dias

Se o dispositivo sempre estiver perdido com ciclos de 30 minutos

**Bt**  $\approx$  9755 horas  $\approx$  406 dias  $\approx$  1.1 anos

### 6.2.3 Heltec WiFi LoRa 32 V2

Variável	Valores
Bc	1200 mAh
Is	800 uA
Itr	130 mA
Ic	40 mA(Placa) + 45 mA(GPS) = 85 mA
Ip	40 mA
Ir	30 mA
Igps	130 mA
ts	1 hora
ttr	394.496 ms
tc	1s
tp	1s
tr	3s
tgps	394.496 ms

Se o dispositivo nunca estiver perdido com ciclos de 1 hora

**Bt** ~ 1413 horas ~= 59 dias

Se o dispositivo sempre estiver perdido com ciclos de 10 minutos

**Bt** ~ 911 horas ~= 38 dias

Se o dispositivo sempre estiver perdido com ciclos de 30 minutos

**Bt** ~ 1233 horas ~= 51 dias

## 6.2.4 LoRa Radio Node v1.0

Variável	Valores
<b>Bc</b>	1200 mAh
<b>Is</b>	$300 \mu\text{A}(\text{Placa}) + 200\mu\text{A}(\text{RTC}) = 500 \mu\text{A}$
<b>Itr</b>	$120 \text{ mA} + 200 \mu\text{A}(\text{RTC}) = 120.2 \text{ mA}$
<b>Ic</b>	$30 \text{ mA}(\text{Placa}) + 45 \text{ mA}(\text{GPS}) + 200 \mu\text{A}(\text{RTC}) = 75.2 \text{ mA}$
<b>Ip</b>	$30 \text{ mA} + 200 \mu\text{A}(\text{RTC}) = 30.2 \text{ mA}$
<b>Ir</b>	$40 \text{ mA} + 200 \mu\text{A}(\text{RTC}) = 40.2 \text{ mA}$
<b>Igps</b>	$20 \text{ mA} + 200 \mu\text{A}(\text{RTC}) = 120.2 \text{ mA}$
<b>ts</b>	1 hora
<b>ttr</b>	394.496 ms
<b>tc</b>	1s
<b>tp</b>	1s
<b>tr</b>	3s
<b>tgps</b>	394.496 ms

Se o dispositivo nunca estiver perdido com ciclos de 1 hora

**Bt**  $\approx$  2165 horas  $\approx$  90 dias

Se o dispositivo sempre estiver perdido com ciclos de 10 minutos

**Bt**  $\approx$  1171 horas  $\approx$  49 dias

Se o dispositivo sempre estiver perdido com ciclos de 30 minutos

**Bt**  $\approx$  1775 horas  $\approx$  74 dias



# Conclusão

Concluíndo é possível afirmar-se que a tecnologia LoRa destacou-se como a escolha ideal para ir de encontro às necessidades específicas deste projeto de geolocalização de animais domésticos perdidos. A capacidade de longo alcance do LoRa permitiu que os geolocalizadores transmitissem dados de forma confiável, mesmo em áreas rurais ou com pouca rede. A eficiência energética foi fundamental para a maximização da autonomia dos dispositivos garantindo que os animais de estimação pudessem ser rastreados continuamente sem a necessidade de trocas frequentes de bateria. Oferece também uma solução de baixo custo, o que era fundamental para tornar a geolocalização de animais de estimação acessível.

Explorar as várias placas de desenvolvimento trouxe informações valiosas sobre a importância do conhecimento aprofundado da tecnologia em qualquer projeto de IoT. Cada placa apresenta as suas próprias características e recursos, o que faz com que cada uma tenha uma personalidade diferente com desafios diferentes a serem superados, alguns dos quais não foram ultrapassados, mas permitiu obter conhecimento significativo sobre a implementação de dispositivos de geolocalização.

Os resultados obtidos demonstraram que a tecnologia LoRa se revelou altamente eficaz para responder as questões impostas por este projeto. Comparando as autonomias de cada uma das placas de desenvolvimento conseguimos chegar ás mais variadas conclusões, como por exemplo o que mais importa nas características da placa mais adequada a serem escolhidas e os parâmetros adequados para cenários e necessidades diferentes.

## 7.1 Melhorias Futuras

### **Testes em Ambientes Reais:**

Verificar a eficácia e a confiabilidade dos dispositivos em diversas condições e cenários.

### **Consumo de Energia Eficiente:**

Desenvolver soluções de baixo consumo de energia para aumentar a vida útil das baterias dos dispositivos que os animais carregam.

### **Tamanho e peso reduzidos:**

Diminuir o tamanho e o peso do dispositivo para garantir que seja confortável para animais de diferentes tamanhos usarem.

### **Segurança de Dados:**

Garantir que todos os dados de localização sejam protegidos e sigam as regulamentações de privacidade de dados.

### **Rastreamento de Saúde e Atividade:**

Integrar sensores de saúde e atividade aos dispositivos para que os proprietários possam monitorar a saúde de seus animais em tempo real.

### **Apliação de Acompanhamento para Proprietários:**

Criar uma aplicação móvel para os proprietários possam geolocalizar os seus animais em tempo real e receber notificações.

### **Customização e Escalabilidade:**

Projetar dispositivos personalizados para diferentes tipos de animais e escaláveis para trazer um grande número de utilizadores.



# Bibliografia

- [1] “Cubecell 1/2aa node,” <https://heltec.org/project/htcc-ab02a/>, accessed: 11-06-2023.
- [2] “Cubecell gps-6502,” <https://heltec.org/project/htcc-ab02s/>, accessed: 11-06-2023.
- [3] “Heltec wifi lora 32 v2,” <https://heltec.org/project/wifi-lora-32/>, accessed: 11-06-2023.
- [4] “Ttgo lora32 v1,” <https://www.lilygo.cc/products/lora32-v1-0-lora-868mhz-915mhz>, accessed: 11-06-2023.
- [5] “Lora radio node v1.0,” <https://www.tindie.com/products/IOTMCU/lora-radio-node-v10/>, accessed: 14-06-2023.
- [6] “Arduino pro mini,” <https://docs.arduino.cc/retired/boards/arduino-pro-mini>, accessed: 15-06-2023.
- [7] “Reyax rylr896,” <https://reyax.com/products/rylr896/>, accessed: 15-06-2023.
- [8] “Neo-6 gps,” [https://content.u-blox.com/sites/default/files/products/documents/NEO-6\\_DataSheet\\_%28GPS.G6-HW-09005%29.pdf](https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf), accessed: 09-06-2023.
- [9] “Ds3231,” <https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf>, accessed: 14-06-2023.

- [10] “PET TRACKER invoxia,” <https://www.invoxia.com/pt/product/pet-tracker>, accessed: 16-06-2023.
- [11] “Postgresql,” <https://www.postgresql.org/>, accessed: 13-07-2023.
- [12] “Docker,” <https://www.docker.com/>, accessed: 13-07-2023.
- [13] J. Deacon, “Model-view-controller (mvc) architecture,” *Online/[Citado em: 10 de março de 2006.]* <http://www.jdl.co.uk/briefings/MVC.pdf>, vol. 28, 2009.
- [14] “Arduino avr boards,” <https://github.com/arduino/ArduinoCore-avr>, accessed: 08-07-2023.
- [15] “Esp32,” <https://github.com/espressif/arduino-esp32>, accessed: 08-07-2023.
- [16] “Heltec esp32 series dev-boards,” [https://github.com/Heltec-Aaron-Lee/WiFi\\_Kit\\_series](https://github.com/Heltec-Aaron-Lee/WiFi_Kit_series), accessed: 08-07-2023.
- [17] “Cubecell development framework,” <https://github.com/HeiTecAutomation/CubeCell-Arduino>, accessed: 08-07-2023.
- [18] “Lowpower library,” <https://github.com/LowPowerLab/LowPower>, accessed: 10-07-2023.
- [19] “Arduino\_json,” [https://github.com/arduino-libraries/Arduino\\_JSON](https://github.com/arduino-libraries/Arduino_JSON), accessed: 10-07-2023.
- [20] “Arduinojson,” <https://github.com/bblanchon/ArduinoJson>, accessed: 10-07-2023.
- [21] “Tinygps library,” <https://github.com/mikalhart/TinyGPSPlus>, accessed: 10-07-2023.
- [22] “Spi library,” <https://docs.arduino.cc/learn/communication/spi>, accessed: 10-07-2023.
- [23] “Arduino lora library,” <https://github.com/sandeepmistry/arduino-LoRa>, accessed: 10-07-2023.

- [24] “Softwareserial library,” <https://docs.arduino.cc/learn/built-in-libraries/software-serial>, accessed: 10-07-2023.
- [25] “Espsoftwareserial library,” <https://github.com/plerup/espsoftwareserial/>, accessed: 10-07-2023.
- [26] “Rtc library,” <https://github.com/adafruit/RTClib>, accessed: 10-07-2023.
- [27] “Cubecell lora library,” <https://github.com/HeiTecAutomation/CubeCell-Arduino/tree/master/libraries/LoRa>, accessed: 10-07-2023.
- [28] K. Relan and K. Relan, “Beginning with flask,” *Building REST APIs with Flask: Create Python Web Services with MySQL*, pp. 1–26, 2019.
- [29] “Admin kit,” <https://adminkit.io/>, accessed: 22-07-2023.
- [30] “Bootstrap,” <https://getbootstrap.com/>, accessed: 22-07-2023.
- [31] “Datasheet do sx1262,” <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000Un7F/yT.fKdAr9ZAo3cJLc4F2cBdUsMftpT2vsOICP7NmMo>, accessed: 19-08-2023.