universidade de aveiro deti departamento de electrónica, telecomunicações e informática

41488 – Industrial Project

Project Final Report

Nome do projeto:	Repetidor LoRa
Empresa:	Wavecom - Soluções Rádio, SA
Membros da equipa:	Contacto principal: Carolina Pinho, carolinapinho@ua.pt, 912714948 Outros membros: Francisco Oliveira, franciscojno@ua.pt, 913349971 Miguel Amorim, miguelamorim@ua.pt, 966415488 Ana Vicente, ana.vic@ua.pt, 917716729 Henrique Chaves, henriquechaves09@ua.pt, 967127978 Gonçalo Ferreira, gdferreira99@ua.pt, 968010870
Data:	28 de June de 2021
Supervisor:	Rui Manuel Escadas

Acrónimos

ACK – Acnowledgement

Node-RED - ferramenta de programação construída em Node.js. Fornece um editor, via browser, à base de *flows* e permite a criação de funções *javascript*.

OTAA - Over the Air Authentication

P2P - Point-to-Point

RPi - Raspberry Pi

IP – Endereço de Internet Protocol (IP)

Sumário de Gestão:

Durante o desenvolvimento do projeto Repetidor LoRa no âmbito da unidade curricular de Projeto industrial com o apoio da empresa Wavecom, o grupo definiu desde início que o objetivo principal seria a construção de um protótipo final de um dispositivo que permitisse a repetição de pacotes LoRa chegados de end-devices até ao gateway mais próximo. Para além deste objetivo, a empresa pretendia uma investigação aprofundada sobre a teoria das diferentes possibilidades de implementação deste dispositivo, uma vez que não existe muita informação e documentação disponível tanto na empresa como nos habituais fóruns e meios de acesso a informação deste tema. Pelo que, o objetivo do grupo ao longo do tempo, também devido à dificuldade geral do projeto, fundiu-se com a obtenção de conteúdo relevante que pudesse ser entregue à empresa para que as diversas possibilidades de implementação pudessem ficar mais nítidas e até mesmo excluir determinadas abordagens que se mostrassem ineficazes.

As milestones planeadas foram entregues dentro do prazo estipulado pelo Unidade Curricular sendo que em cada uma destas era feita uma atualização no planeamento ou até mesmo na própria solução do projeto. Estas alterações foram analisadas e reajustadas a fim de minimizar o impacto de tais mudanças no desenvolvimento do projeto.

Devido ao sistema autónomo implementado o custo total do projeto ficou em 86.74 € não constando com as placas dos end devices que foram dadas pela empresa.

Resumo do Projeto

Como já foi descrito acima, pretende-se que o dispositivo desenvolvido neste projeto faça a repetição de pacotes LoRa, de forma que end-devices sem acesso direto à gateway possam comunicar com o servidor através do repetidor desenvolvido com o objetivo de estes cobrirem uma maior área.

Os utilizadores interagem com o sistema através de uma interface que permite mostrar de forma intuitiva, não só a quantidade de pacotes recebidos/enviados, mas também informações relevantes como: o balanço energético do Repetidor LoRa, o *duty-cycle* do canal LoRa em utilização e o tráfego LoRa. Também será possível que o utilizador configure novos endpoints para serem adicionados ao sistema, para isso será desenvolvida a função de acrescentar os Id's desejados à whitelist disponível.

A compilação entre os objetivos do trabalho e os conteúdos enunciados anteriormente permitiu-nos desenvolver um esquema que representa a arquitetura geral do sistema para este projeto que pode ser observado na Figura 1.

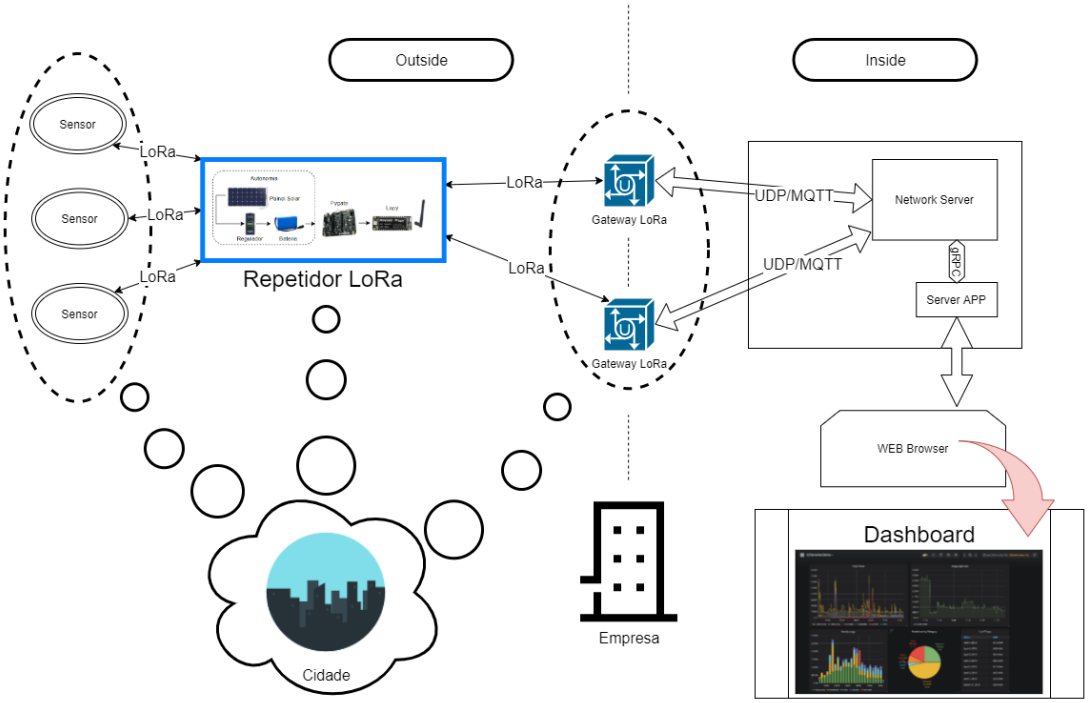

Figura 1 - Esquema da arquitetura do sistema do projeto

Estado do Projeto

Mesmo com o grande nível de dificuldade do projeto o grupo dedicou-se desde início em responder aos objetivos propostos para cada milestone entregue. No entanto, o objetivo principal de conseguir desenvolver um protótipo funcional de um repetidor LoraWan-LoraWan não foi totalmente cumprido, neste sentido foi desenvolvida uma interface de utilizador totalmente funcional, bem como a parte de hardware que implementa um Sistema autónomo com painéis solares e uma Lopy4 com uma placa de expansão Pygate que idealmente estaria a trabalhar como um repetidor.

Tendo em conta as especificações indicadas no ponto anterior, o grupo desenvolveu uma série de soluções que cumprissem com os requisites iniciais.

Inicialmente a solução passava por recorrer à placa Lopy4 junto com a placa de expansão Pygate para a repetição de pacotes Lora, no entanto, à medida que o assunto ia sendo estudado e o grupo ficava mais familiarizado com toda a tecnologia foi se percebendo que para esta a utilização como destas placas como repetidor final, estas (Lopy4+Pygate) teriam de sofrer severas alterações a nível de firmware o que se foi mostrando um trabalho desafiante e trabalhoso. Ao mesmo tempo decorria um estudo das janelas de receção, para que quando a placa estivesse a funcionar como devido termos todo o material para avançar com a repetição de pacotes em si. O desenvolvimento da interface de utilizador decorreu sem grandes problemas.

- **Hipóteses de repetição de pacotes e respetivos testes**

É importante deixar neste relatorio as hipoteses que estiveram em cima da mesa quando no que toca à teoria de repetição de pacote.

O processo de pensamento deste assunto está representado no diagrama da Fig. 10, onde estão enunciadas as possíveis implementações estudadas para repetição de pacotes.

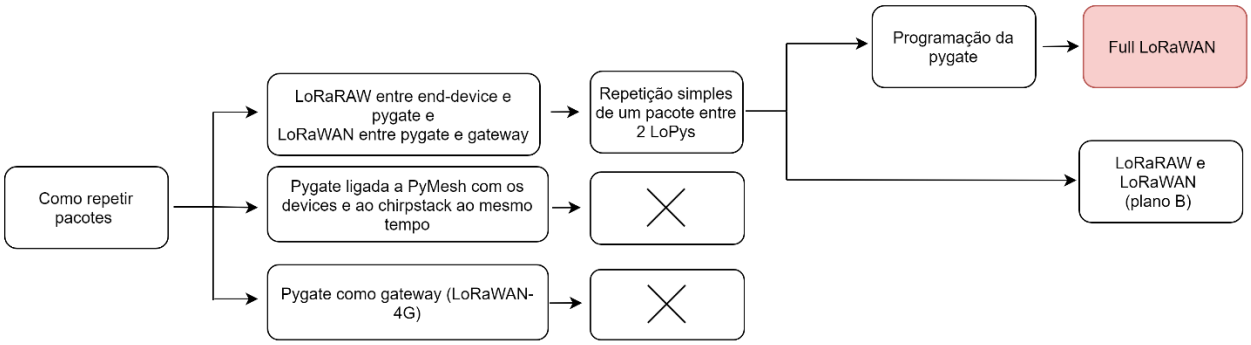

Figura 10 - Diagrama do desenvolvimento das teorias de repetição de pacotes

Consideraram-se, essencialmente, as seguintes três abordagens:

a. LoRaRaw e LoRaWAN

O conceito baseava-se, inicialmente, em dividir o sistema em duas partes:

1. Entre o end-device e o repetidor a comunicação seria toda por Lora P2P, ou seja, sem ir à camada de rede.
2. Entre o repetidor e o gateway, a comunicação seria na camada de rede, sendo tudo por LoRaWAN, mais precisamente por OTAA.

Através desta abordagem foi desenvolvida a primeira repetição de pacotes já enunciada no ponto anterior. No entanto, foi efetuada apenas entre 2 LoPys com o transceiver SX1276, ou seja, utilizando apenas um canal para esta transmissão. Uma vez que, o facto de haver apenas um canal de transmissão implica um aumento da probabilidade de haver pacotes que não serão ouvidos pelo repetidor, esta abordagem não é válida para o objetivo do trabalho.

Para além disso, esta hipótese iria implicar que o repetidor permutasse entre Lora e LoRaWAN, guardando na memória volátil as informações de rede para não ter de se autenticar novamente sempre que pretendesse descer para LoRa. E, também, envolveria um sistema de ACK entre os end-devices e o repetidor. O que, obviamente, provocaria um aumento no consumo energético do sistema.

Esta hipótese, mesmo com os seus problemas, permitiu que se realizasse outros testes devido à sua fácil implementação e que se obtivesse conhecimentos práticos.

b. PyMesh

Outra hipótese estudada foi a utilização da tecnologia PyMesh da PyCom entre o repetidor e os vários end-devices. Contudo, esta implementação implicava a utilização de outra interface de rede, por exemplo BLE ou WiFi, o que implicaria, novamente, um alto consumo energético.

Sendo assim, tomou-se a decisão de descartar esta opção, em conjunto com a empresa.

c. Pygate como gateway

A Pygate é uma placa que vem autoconfigurada para se comportar como um gateway. No entanto, a sua funcionalidade neste projeto seria, juntamente com a LoPy, representar um repetidor.

Para tal, é necessário que esta comunicasse com outros gateways e aplicasse um tratamento de dados aos pacotes, externo ao ChirpStack. Este tratamento passaria então por utilizar uma aplicação externa responsável pela descodificação de pacotes que passam pelo gateway, organização de dados, interligação com a interface gráfica, entre outros.

O problema que se mantém neste caso, é que para ocorrer comunicação direta entre o repetidor e essa aplicação externa, seria necessário algum tipo de módulo WiFi, Ethernet ou LTE para permitir uma comunicação por TCP entre ambos, devido à configuração intrínseca da Pygate.

Esta hipótese foi descartada, pois, como se trata de um produto autónomo, a adição de um módulo desses iria comprometer bastante o consumo energético.

- **Firmware do Repetidor**

De forma mais pormenorizada pode dizer-se que a nível de firmware foi necessário analisar todo o repositório encontrado com o código referente ao firmware da Pygate como um gateway. O foco principal esteve no ficheiro LoRa_pkt_fwd.c[1]. Com esta análise foi possível perceber e construir uma sequência de processos que aconteciam desde que a placa era ligada até ao momento em que estaria a enviar pacotes (uplinks e donwlinks), bem como, que funções são utlizadas para cada uma das situações. Desta forma, foi possível uma compreensão geral do firmware default da placa que foi essencial para avançar no desenvolvimento de um novo código firmware.

Este novo firmware é baseado no repositório em estudo com o objetivo de retirar tudo o que envolvesse o protocolo TCP e substituir por comunicação apenas LoRaWAN. De seguida, o grupo desenvolveu o seu primeiro código de teste de firmware onde o objetivo inicial seria receber um pacote recorrendo apenas à thread_up, responsável pelos uplinks, do repositório. Para isso foi usada uma LoPy4 que enviava pacotes a cada 15s, como se ilustra na Figura 2.

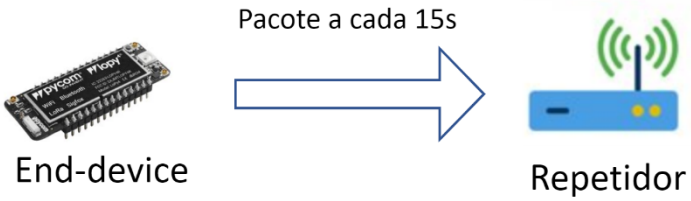

Figura 2 - Esquema do teste realizado

Após se efetuar o flash do novo firmware para a placa e correr um código de receção de pacotes na mesma obteve-se o resultado da Figura 3. A figura mostra um pacote a ser recebido pelo concentrador (placa como encaminhador).

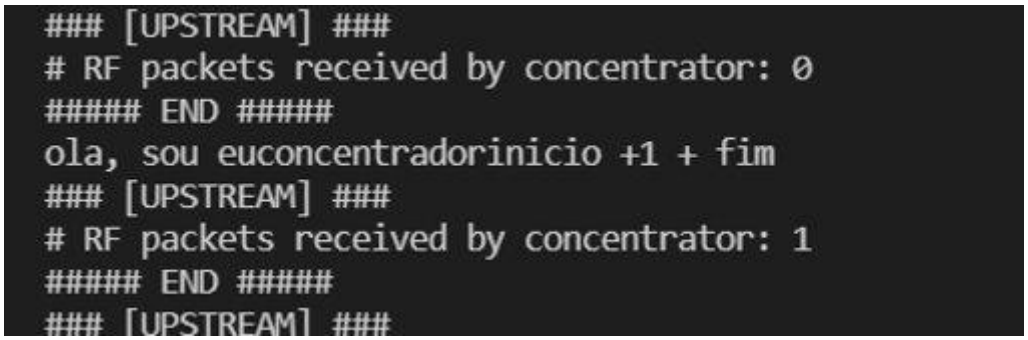

Figura 3 - Output do teste efetuado

No entanto, este pacote só era recebido muito esporadicamente, o que fez com que este teste não tivesse o resultado esperado. De qualquer das formas, este teste mostra que o grupo conseguiu desenvolver um código sem qualquer ligação TCP e mesmo assim a placa ligou e aceitou o novo firmware.

Ao tentar resolver esta entrave o grupo encontrou uma outra possível solução, que passaria pelo uso em paralelo de 2 transceivers[2] SX1257, que a Pygate contém, ou seja, um para efetuar a receção e outro para a transmissão simultânea de pacotes.

Desta forma, a placa iria funcionar como um end-device para o gateway, usando um dos transceivers para enviar informação e uplinks e como gateway para os diversos end-devices, utilizando o outro transceiver para estar à escuta de novos dados. Mais informação sobre este assunto está presenta no developer guide, visto que contém informação bastante relevante para a empresa dar continuidade a este projeto.

- **Interface de utilizador**

A interface de utilizador encontra-se funcional e os seus objetivos passaram por conter:

- Um separador de utilizadores que apresenta as informações dos utilizadores registados no servidor (na versão atual foi removido, pois o LoRaServer já o tem);
- Um separador com uma visão geral dos repetidores do cliente, que apresenta uma tabela com todos os dispositivos conectados, um gráfico com a média de pacotes enviados e recebidos pelos repetidores e também os consumos médios dos mesmos;
- Um separador para monitorização individual de cada repetidor que contém:
    - Whitelist, tabela com os dispositivos que utilizam o repetidor-x para que as suas mensagens cheguem ao servidor e com quais gateways este repetidor comunica;
    - Gráfico de barras, que apresenta duas barras, uma para o número de pacotes enviados e outra para os recebidos pelo repetidor-x;
    - Monitorização de energia, contém um ponteiro com a percentagem atual da bateria do repetidor-x, mensagem com o tempo restante de bateria do repetidor-x e um gráfico com a evolução do consumo energético do repetidor-x nos últimos 15 dias.

De forma a responder a estes requisitos, a solução encontrada foi utilizar a ferramenta Grafana, uma vez que é mais flexível e interativa e também porque a interface gráfica implementada no dashboard do Node-red tinha dois grandes problemas:

- O utilizador tinha de mudar de separador para poder ver as informações de repetidores diferentes, pois não tinha a possibilidade de filtrar as tabelas da whitelist e as informações dos separadores, de modo a no mesmo separador poder escolher informações relativas a um determinado repetidor;
- Os gráficos apresentados não permitiam alterar a escala temporal.

Com o grafana o problema da tabela foi resolvido com uma caixa que permite filtrar a whitelist geral para uma tabela em que se observa só os dispositivos conectados a um determinado repetidor, neste caso repetidor 1 escolhido através da caixa 1. Relativamente à escala temporal, o Grafana permite rapidamente ajustá-la na caixa 2. Estas caixas podem ser vistas a vermelho na Figura 4.

Figura 4 - Interface monitorização individual implementada no Grafana

Na figura 5, podem-se observar os dois gráficos temporais implementadas no Grafana com valores fictícios, onde o da esquerda corresponde ao número de pacotes recebidos e o da direita ao consumo do repetidor em questão.

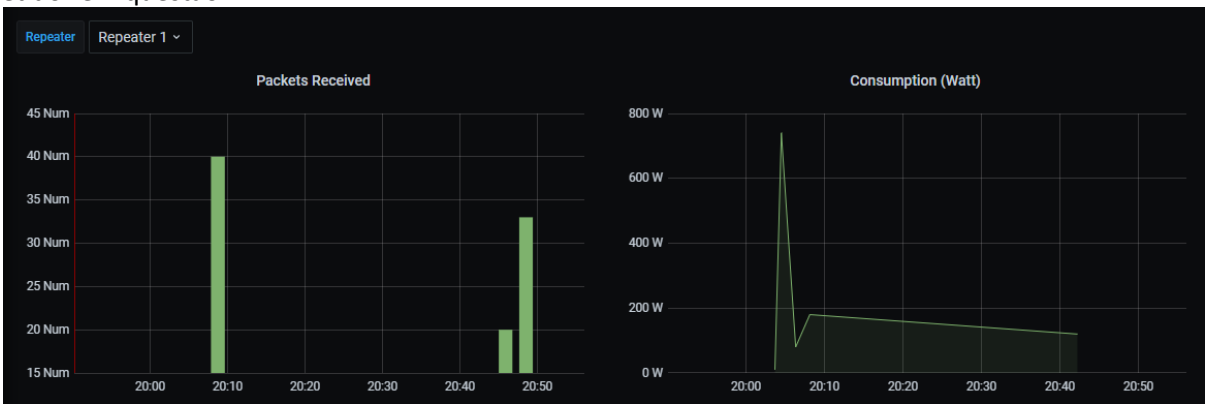

Figura 5 - Gráficos simulados em Grafana

Esta interface é desenvolvida através das interligações entre a base de dados InfluxDB e as ferramentas Node-Red e Grafana. No NR é feito o tratamento das informações provenientes do servidor e é enviada a informação para a base de dados que posteriormente é acedida no Grafana. Tendo em conta isto, os detalhes relativamente ao desenvolvimento e interligação entre as várias ferramentas está presente no Developer_guide_server_external_app.

- **Autonomia**

O sistema de autonomia foi implementado com o material descrito no documento **Sistema Autonomo.docx**, como se pode observar na figura 11, e foram efetuados alguns testes de consumo energético de modo a obter uma ideia do consumo geral do Repetidor.

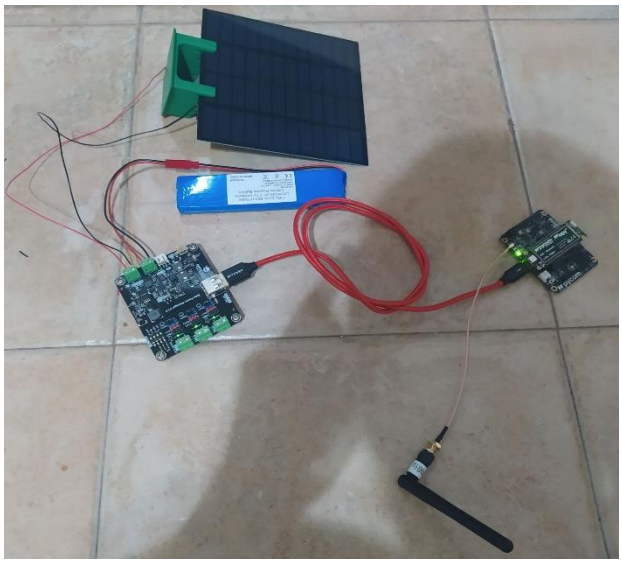

Figura 6 - Repetidor com sistema autónomo acoplado

Os cálculos, o material utilizado e os testes realizados/por realizar deste sistema encontram-se no documento referido acima.

O dimensionamento do sistema autónomo proporcionou um conhecimento aprofundado sobre os sistemas de baixo consumo alimentados a energia solar. Foi possível também obter conhecimento sobre o modulo de gestão energética que apresenta diversas funcionalidades e pode ser aplicado em vários contextos.

Definição do Projeto

No que toca a redes de acesso IOT, as tecnologias LPWAN, Low-Power Wide-Area network, são bastante utilizadas devido a apresentarem baixa potência e longo alcance. Como é o caso da principal tecnologia envolvida neste projeto, LoRa.

No entanto, estas tecnologias apresentam limitações ao nível de cobertura rádio. De certa forma, o longo alcance acaba por ser apenas teórico, uma vez que o alcance pode ser encurtado pela presença de obstáculos tão simples como uma parede, impedindo assim que um endpoint, cuja localização seja pouco favorável, consiga estabelecer ligação com um gateway, como se ilustra na Figura 7.

Este problema ocorre, por exemplo, com contadores de água ou caixotes do lixo subterrâneos. Deve-se ao facto de os contadores estarem normalmente dentro de estruturas metálicas e os caixotes do lixo debaixo da terra, pelo que constituem um obstáculo à transmissão de informação.

Este problema poderia ser resolvido com a instalação de mais gateways LoRa pela cidade, mas esta instalação é dispendiosa, difícil e pouco prática.

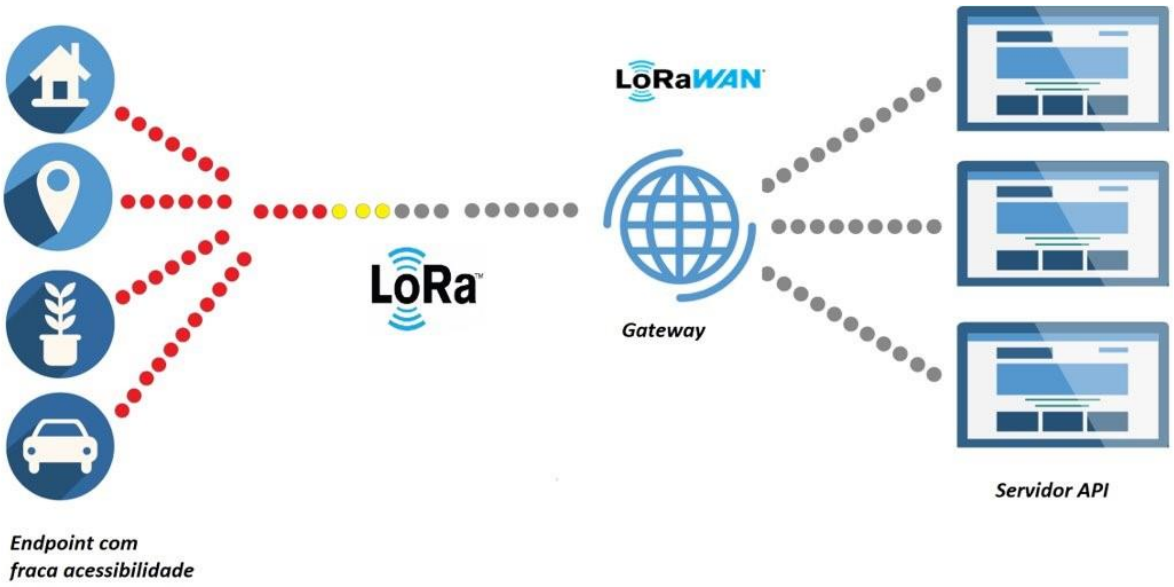

Figura 7 - Ilustração do problema

Daqui surge a necessidade de implementar um sistema que garanta a conectividade entre gateways e endpoints inacessíveis e que ao mesmo tempo seja de fácil instalação, mais em conta e autónomo. A solução que se pretende implementar é um Repetidor LoRa que terá de cumprir todos estes objetivos principais. (ver figura 8).

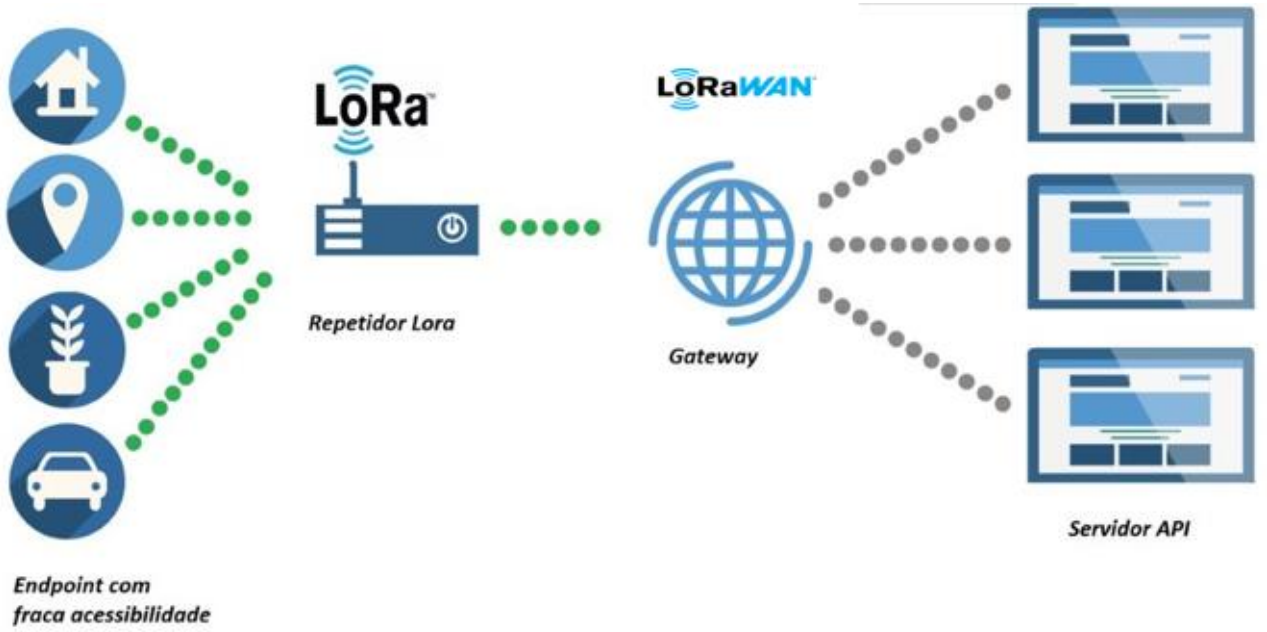

Figura 8 - Ilustração da solução proposta

Portanto os requisitos principais do projeto podem resumir-se em 3 pilares que se encontram na Figura 9, onde as funcionalidades do dispositivo desenvolvido são:

- Fácil instalação;
- Autónomo e low power;
- Evitar colisões e descartar pacotes;
- Gestão de dados;
- Comunicação com o gateway.

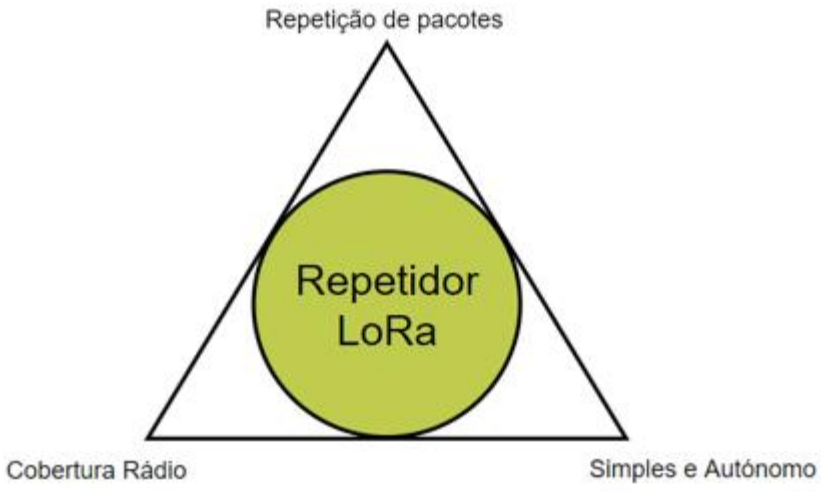

Figura 9 - Triangulação dos objetivos principais

O produto final é idealizado para ser usado por empresas que enfrentem o problema de cobertura rádio, querendo garantir a circulação de informação entre si e os endpoints com sucesso. Normalmente, estas são empresas que têm como objetivo a monitorização de determinado ambiente ou espaço.

No ponto de vista do utilizador a nossa oferta ao mercado será o acesso a dados que se encontram com fraca acessibilidade por algum obstáculo imposto ou a própria distância entre o endpoint e o gateway.

De forma a responder a estes objetivos o grupo passou pela utilização dos seguintes componentes tecnológicos:

- Utilização de Node Red e Grafana junto com interligações entre a base de dados InfluxDB para a implementação da interface de utilizador;
- Simuladores como GN3 e Matlab para a compreensão numa fase inicial da tecnologia em estudo;
- Sistema autónomo composto por painéis solares;
- Utilização de ESP-IDF para configuração do firmware e Visual Studio, com a integração da livraria Pymakr, para debug e tratamento de código de alto nível;

Arquitetura do Projeto

- **Arquitetura geral**

No que toca à arquitetura do projeto, tal como era de esperar sofreu atualizações à medida que o grupo foi entendendo melhor os conceitos do projeto. Inicialmente a arquitetura que definia o projeto estava limitada a algo tão simples como o apresentado na Figura 8. No entanto, após a pesquisa sobre como efetivamente proceder à repetição de pacotes, que está presente no ponto em cima neste mesmo relatório,

a nível de camadas de rede surgiu uma atualização na arquitetura que se apresenta na Figura 10 que já se apresenta mais completa que a anterior.

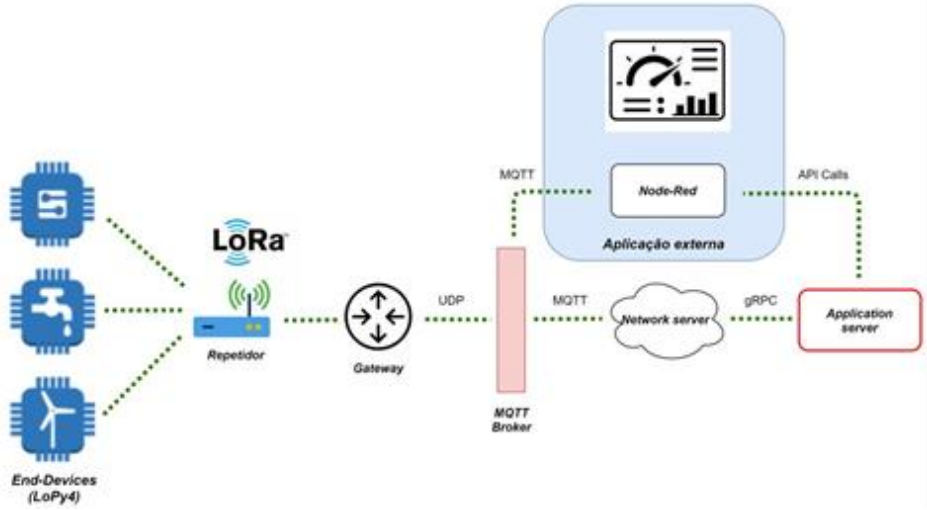

Figura 10 - Arquitetura geral do projeto atualizada

A Pygate é uma placa que vem autoconfigurada para se comportar como um gateway que necessita de uma ligação TCP. No entanto, a sua funcionalidade neste projeto seria, juntamente com a LoPy, ser um repetidor que comunique com outros gateways e aplique o devido tratamento de dados aos pacotes.

Este tratamento passaria então por utilizar uma aplicação externa responsável por fazer a descodificação de pacotes que passam pelo gateway, organizar os dados recebidos dos vários dispositivos (gateways,end-devices e repetidores), interligação com a interface gráfica, entre outros.

Desta forma, como a comunicação entre o gateway e o network server ocorre por MQTT através de um broker, foi possível subscrever o tópico em questão permitindo a observação e manipulação dos pacotes externamente ao network server. Para este efeito, utilizou-se a plataforma node-RED que passou a ser a nossa aplicação externa.

- **Desenho de Hardware**

Considerando uma abordagem inicial de alto nível é possível descrever a arquitetura do projeto (Figura 11). Esta arquitetura pode ser dividida por 2 módulos principais: o modulo de alimentação e o modulo do repetidor.

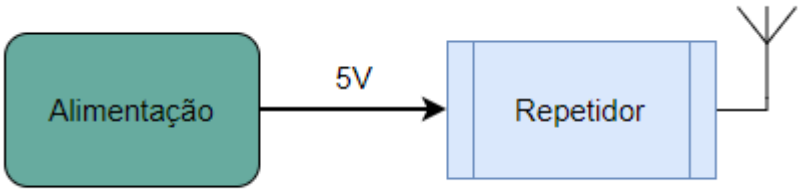

Figura 11 - Módulos da arquitetura do sistema

1. Módulo de alimentação

É responsável por fornecer energia ao sistema. Tem por base um sistema autónomo composto por um painel solar, uma bateria e um módulo responsável pela gestão energética, ou seja, responsável pela

gestão de carregamento das baterias e regulação da tensão recebida para a tensão de alimentação do módulo do repetidor.

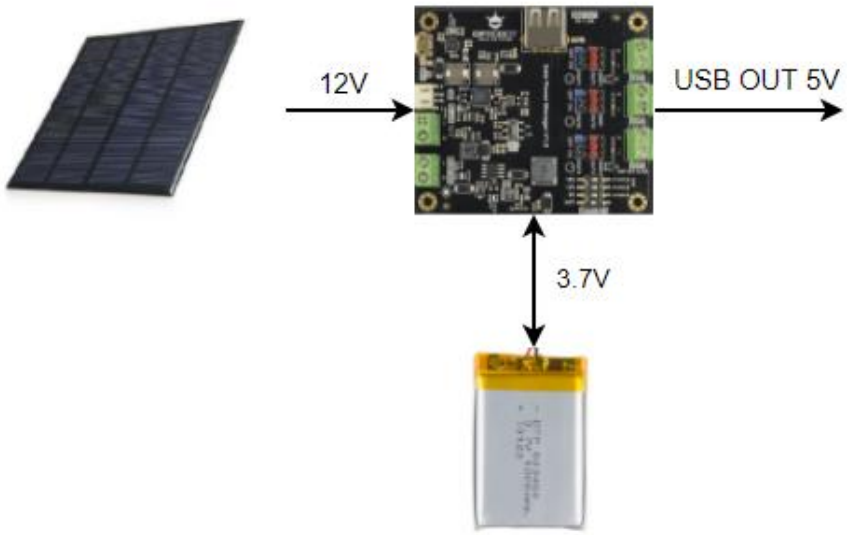

Figura 12 - Arquitetura do módulo de alimentação

Como se pode observar na Figura 12, o sistema é alimentado por uma bateria de 3.7V, mas a sua escolha deve ser cuidadosa, pois existem dois aspetos importantes a ter em conta:

- A placa responsável pelo carregamento e gestão energética apenas é compatível com baterias de 3.7V Por isso, serão ligadas caso necessário várias baterias em paralelo de modo a obter-se a capacidade necessária. Esta dependerá da corrente que o Pygate necessita para ser alimentado (aprox. 360mA).
- Temperatura máxima de operação, pois é provável que o repetidor seja exposto à luz solar fazendo com que as baterias atinjam temperaturas elevadas.

Apesar de não fazer parte do sistema em si, será utilizado ainda um outro módulo Lopy4 de modo a simular um endpoint. A sua utilização permitirá a ligação entre o gateway e este último, passando ainda pelo repetidor.

2. Módulo do repetidor

É composto por uma placa de desenvolvimento Pygate em conjunto com LoPy4, para comunicação LoRa (figura 13). As tensões consideradas relevantes para o sistema estão representadas nas figuras. Quanto à ligação entre os dois módulos abordados, esta será feita através de um cabo USB – MICRO USB.

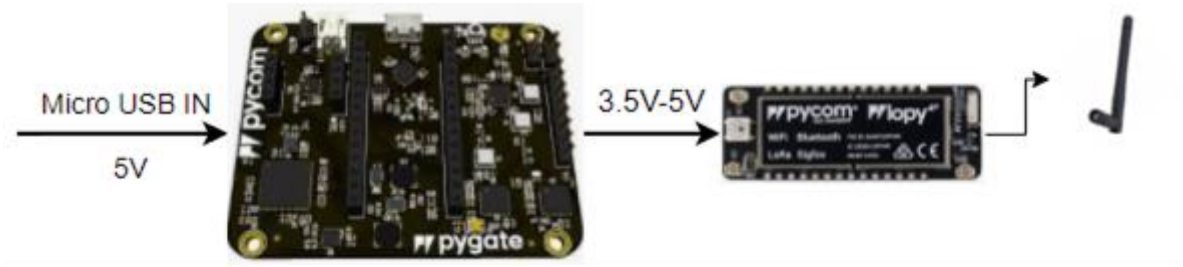

Figura 13 - Arquitetura do módulo do repetidor

- **Desenho de Software**

Nas etapas iniciais deste projeto projetou-se um esquema que demonstrava a ideia do grupo para poder aceder e tratar as informações do sistema de modo a cumprir os seguintes requisitos: gerir a whitelist de endpoints/gateways, gerir os mecanismos de agregação de pacotes, monitorizar os pacotes recebidos/enviados e controlar o consumo energético e respetiva autonomia. Consoante esta descrição o esquema que permite apresentar essa ideia é apresentado na Figura 14.

enunciada

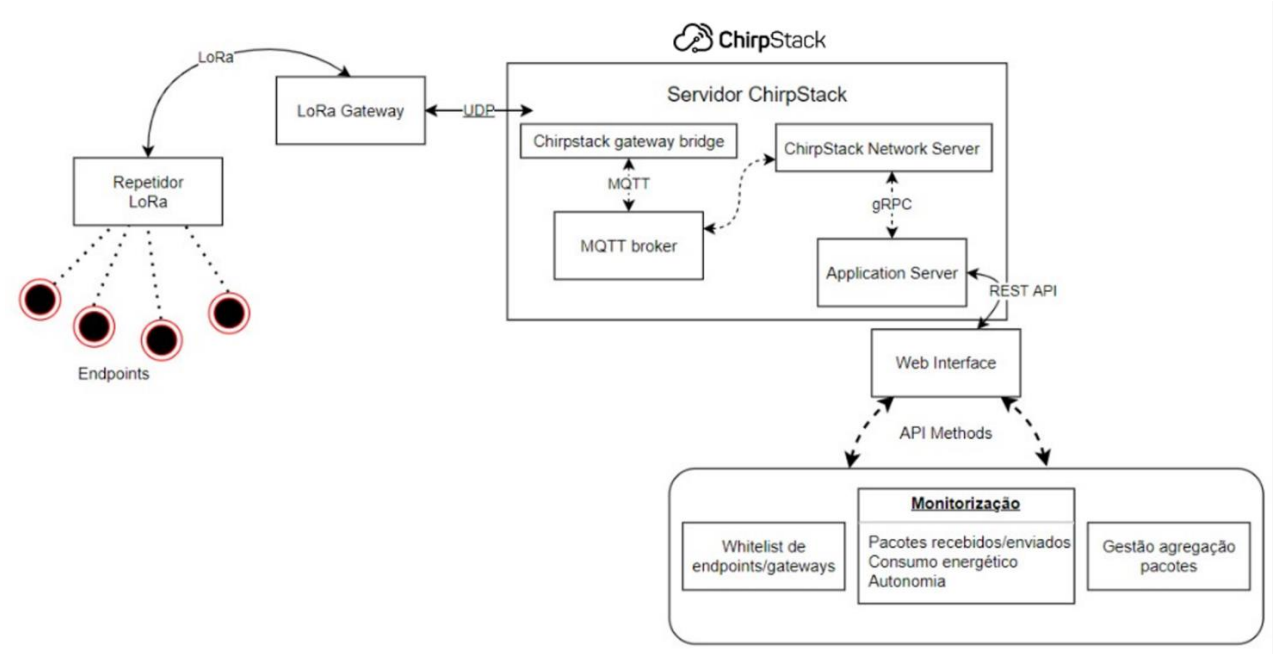

Figura 14 - Diagrama inicial da arquitetura de software

Mediante as condições de avanço do projeto, não foi possível existir informação sobre o número de pacotes trocados pelo repetidor nem fazer a gestão da agregação de pacotes, os valores para consumo energético e autonomia foram todos simulados.

Assim sendo, atualizou-se o desenho de software do projeto para uma arquitetura que recorre às ferramentas enunciadas na interface de utilizador e permite desenvolver a aplicação externa alcançando os objetivos mencionados nessa mesma secção.

Documentação a Entregar

Nesta parte é importante deixar os repositórios utilizados, tanto como referência para desenvolvimento do nosso próprio código como o nosso código em si.

- O repositório onde se encontra o firmware default da pygate como repetidor é: https://github.com/pycom/pycom-micropython-sigfox
- O nosso código é: carolinapinho/RepetidorLoRa (github.com)
- Developer_guide_server_external_app
- Developer Guide de conceitos fundamentais, procedimentos e próximos passos.

- User_guide_Interface
- Sistema Autónomo
- Lessons Learned

Testes

- **Testes Teóricos:**

Inicialmente o grupo investiu numa pesquisa detalha dos diversos aspetos da tecnologia LoRa através do estudo da mesma com recurso a simuladores.
Através da ferramenta Matlab, foram feitas várias simulações e testes de forma a ser possível uma melhor compreensão do comportamento de dispositivos LoRa e respetiva comunicação. As simulações feitas foram as seguintes:

Dependência da largura de banda (LB) utilizada perante mudança de parâmetros:

Para a comunicação ocorrer é necessário definir alguns parâmetros como o *Spreading Factor*, a frequência de trabalho e a potência da antena de transmissão. De notar que o Spreading Factor (SF) relaciona o número de bits de chirps contidos por símbolo, ou seja, a quantidade de informação por símbolo.

Esta é a variável que irá ser parametrizada na primeira simulação. Fixando-se para a frequência de trabalho e potência de transmissão os valores estabelecidos a nível Europeu, ou seja, frequência igual a 870MHz e potência igual a 14dBm.

Esta simulação consiste na comunicação entre um emissor e um recetor com o tamanho de mensagem fixo. Recorreu-se à biblioteca LoRaMatlab e variou-se o Spreading Factor. Desta forma foi possível entender o impacto deste parâmetro na LB ocupada na transmissão.

Abaixo encontram-se dois casos, um bem-sucedido de LB<125KHz na Figura 15 e um malsucedido d LB>125KHz na Figura 16:

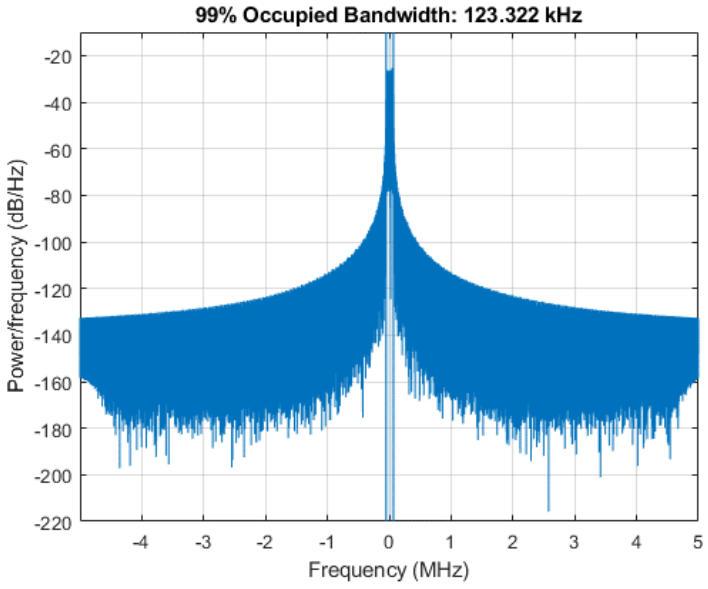

Figura 15 - Densidade espetral de potência para SF = 7

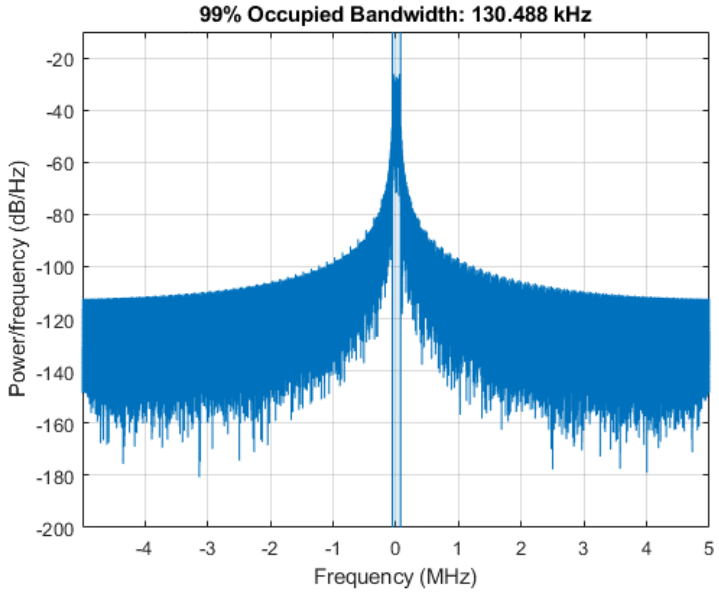

Figura 16 - Densidade espetral de potência para SF=12

Estudo de TimeOnAir (*ToA*) de um pacote Lora perante mudança de parâmetros:

Para posterior estudo e projeção adequada do sistema, foi implementado um script que variando os parâmetros de comunicação e tamanho de mensagem a transmitir, devolve o tempo de envio do *preamble* (cabeçalho) e do *payload* (data), que somados devolvem o tempo total que o pacote se encontrou no ar.

Para confirmar este estudo, uma vez que estes valores serão fundamentais no estabelecimento de comunicação LoRa entre dispositivos, comparou-se os valores obtidos para os diversos cenários com os valores calculados pelo site LoraTools[5], para as mesmas condições.

Concluiu-se que o tempo no ar é proporcional ao Spreading Factor, no entanto para SFs maiores é possível enviar-se mensagens igualmente maiores.

O tempo no ar de uma comunicação unilateral ponto a ponto, em função do SF estão apresentados na seguinte tabela.

Payload	**SF**					
	7	**8**	**9**	**10**	**11**	**12**
20	0.556576	0.602912	0.685344	0.870688	1.241376	1.818912
40	0.582176	0.654112	0.787744	1.034528	1.569056	2.474272
51	0.602656	0.684832	0.828704	1.116448	1.814816	2.965792
70	0.628256	0.725792	0.910624			
90	0.658976	0.776992	1.013024			
150	0.746016	0.930592	1.279264			
200	0.817696	1.063712	1.504544			
222	0.848416	1.114912				
Máx bytes	222	222	115	51	51	51

Tabela 1 - Estudo do timeonAir para vários SF

Fez-se variar o payload possível para cada SF para perceber a tendência do tempo no ar. O máximo payload para o respetivo SF está na última linha da tabela. É de notar, ainda, que se considerou um tempo de processamento igual a 0.5ms, no entanto este valor poderá estar sujeito a alterações de acordo com a implementação final do produto.

No entanto, a grande maioria das comunicações realizadas serão multi-hop, assim assumiu-se como exemplo a comunicação representada na Figura 17.

Figura 17 - Comunicação base

Num uplink, considera-se apenas o segundo segmento do caminho, uma vez que a janela de receção do primeiro segmento só abrirá após a transmissão estar concluída. A somar a este tempo vai estar também o tempo de processamento dos dispositivos. Aquando da resposta, ou seja, downlink, os 2 troços da transmissão têm que ser incluídos e são baseados nos mesmos cálculos utilizados para o uplink.

Tendo-se obtido os valores de tempo no ar em função do SF da tabela seguinte:

SF	ToA (s)
7	2.54525
8	3.34474
9	4.51363
10	3.34934
11	5.44445
12	8.89737

Tabela 2 - Valores de ToA(s) para SF de 7 a 12

Estudo da cobertura de sinal:

Foram também efetuados cálculos com base no modelo de WINNER. Considerou-se o pior caso, ou seja, o tamanho máximo de payload para cada SF. Para estes cálculos foi necessário assumir o seguinte: f=868MHz, T=20ºC, LB=125kHz, h_{tx}=6m, h_{rx}=1m, G_{antena}=10dB e P_{tx}=14dBm. Obteve-se os seguintes resultados em função do SF utilizado.

SF	Sensibilidade (dBm)	Alcance c/ LOS (m)	Alcance s/ LOS (m)	ToA (s)
7	-123	5904 m	1849	2.54525
8	-126	7017	2199	3.34474
9	-129	8340	2616	4.51363
10	-132	9912	3111	3.34934
11	-133	10499	3297	5.44445
12	-136	12478	3921	8.89737

Tabela 3 - Estudo das caracteristicas para cada SF

Sendo assim, o alcance máximo teórico desta implementação deverá rondar os 12.5km.

Estudo de possíveis colisões no ar perante várias transmissões:

O protocolo LoRa permite que um dispositivo aproveite a ortogonalidade dos Spreading Factors e receba mensagens com diferentes valores do mesmo, ao mesmo tempo. Para ver o comportamento de transmissões simultâneas, o estudo dividiu-se em duas partes:

1. **Visualização de colisões num espaço com 500 dispositivos a transmitir durante 60 segundos com Spreading Factor aleatório:**

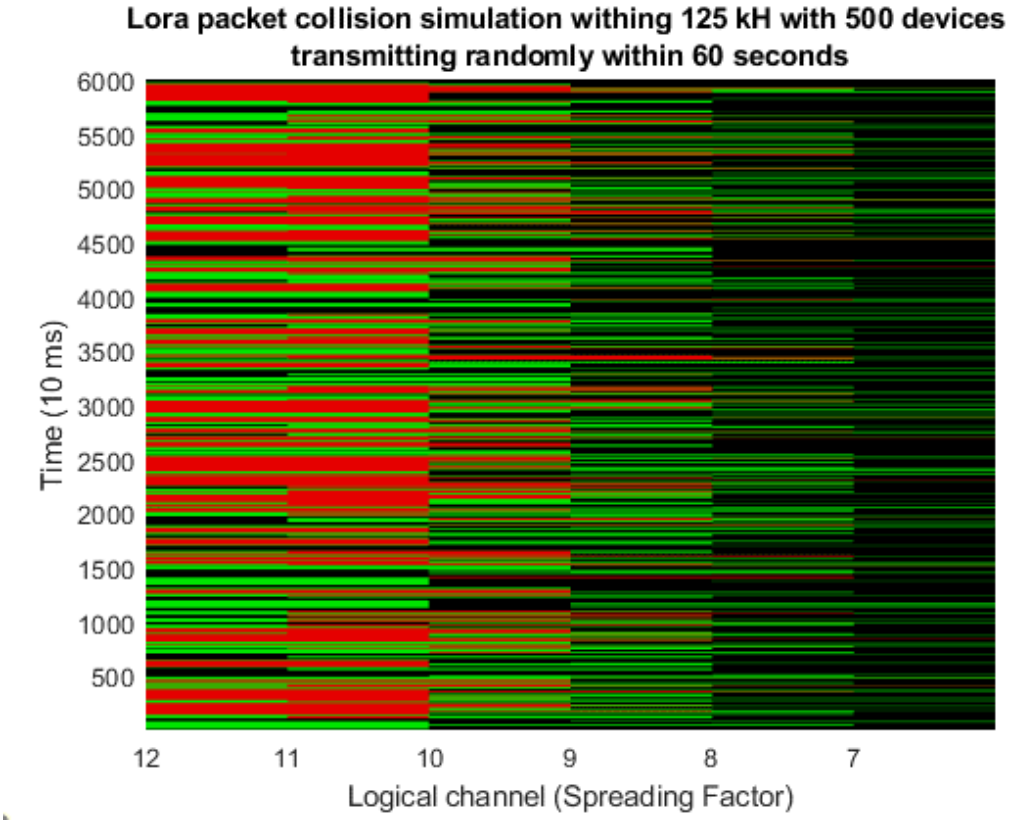

Figura 18 - Relação colisões/SF ao longo de 60 segundos com 500 dispositivos

Na Figura 18 a cor verde representa transmissões bem-sucedidas, enquanto a cor vermelha representa as colisões. Como é possível observar, as mensagens com um Spreading Factor maior (maior *ToA*) transmitidas simultaneamente têm mais possibilidade de provocar colisões.

2. **Visualização de Packet Error Rate e nº de colisões, variando o tráfego de mensagens durante 60 segundos:**

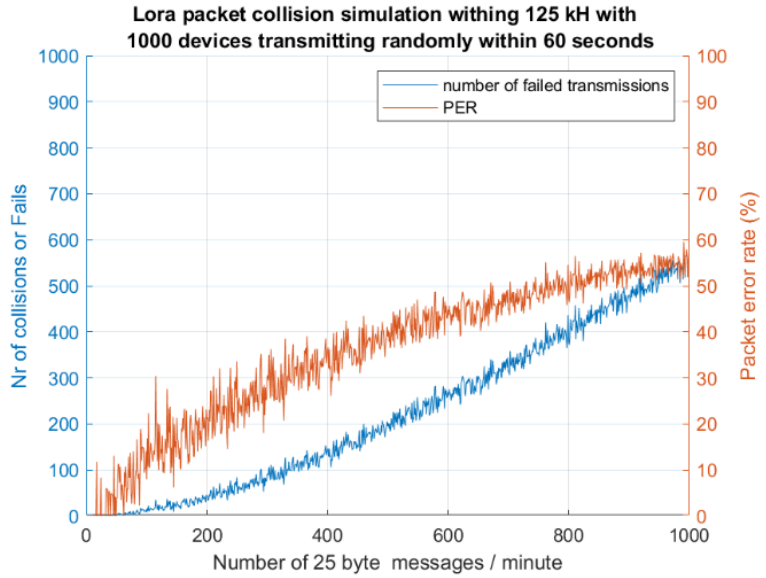

Figura 19 - Relação de colisões vs probabilidade de erro/tráfego da rede para SF=12

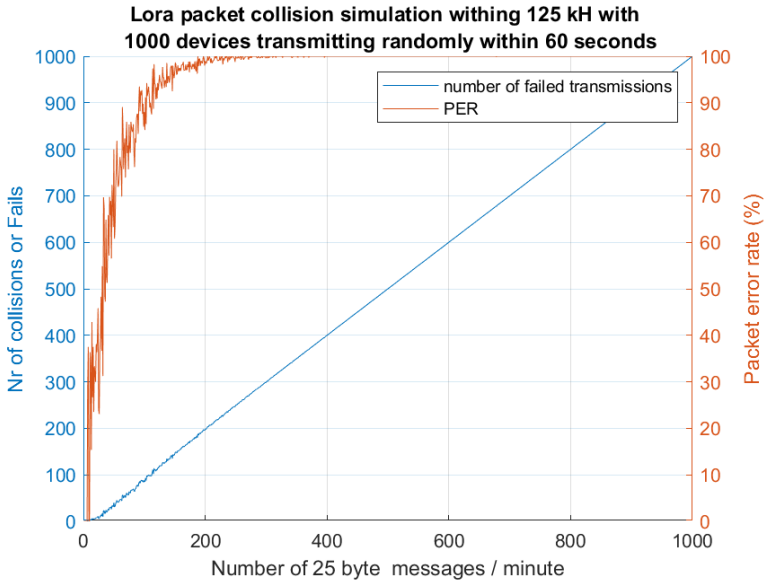

Figura 20 - Relação de colisões vs probabilidade de erro/tráfego da rede para SF=7

Observando estas duas últimas figuras (Figura 19 e 20), conclui-se que para Spreading Factors maiores, a probabilidade de erro e o número de colisões é muito maior. Nota-se que há um aumento linear para SF7. Enquanto para SF12 à volta das 300 mensagens/minuto a probabilidade aproxima-se dos 100%, ou seja, cresce exponencialmente.

Este estudo permitiu projetar duas formas de comunicação possíveis:

3. Join Accept e Join Request[3]

Quando um dispositivo é desconhecido pela rede e é adicionado, terá de passar por um processo de ativação. Para tal existem dois métodos, o *OTTA* (Over-The-Air Activation) e o *ABP* (Activation By Personalization). Neste caso apenas iremos considerar o método *OTTA*.

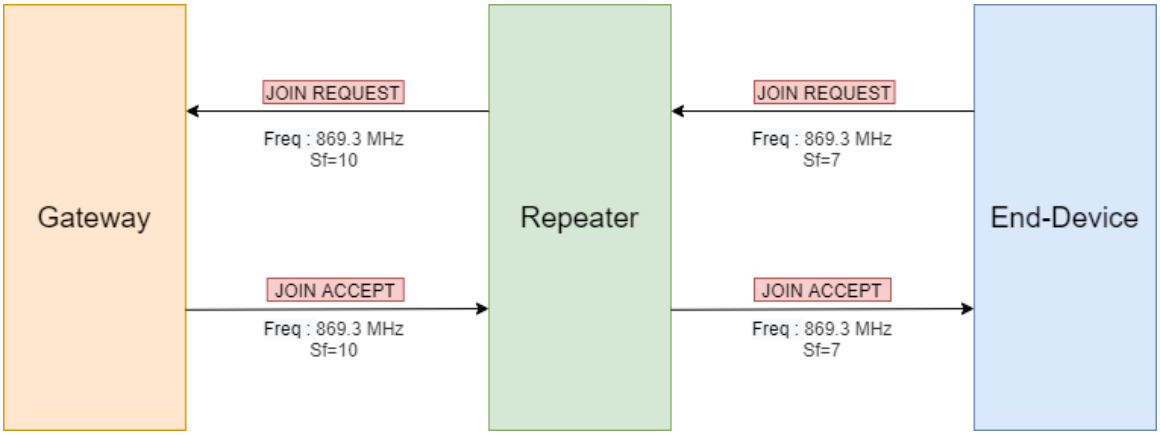

Figura 21 - Representação de Join Request

Este método consiste no envio de uma mensagem Join Request por parte do end-device. O repetidor, ao receber o pacote, repete o mesmo, potenciando o seu alcance de forma a ser recebido por um Gateway. O pacote será tratado posteriormente pelo Application Server. Finalizado o processo, o Gateway emite um pacote Join Accept que fará o caminho inverso do anterior. Recebendo-o, o End-Device gera uma nova sessão.

4. Comunicação e envio de dados

Depois de um dispositivo estar na rede poderá haver troca de informação.

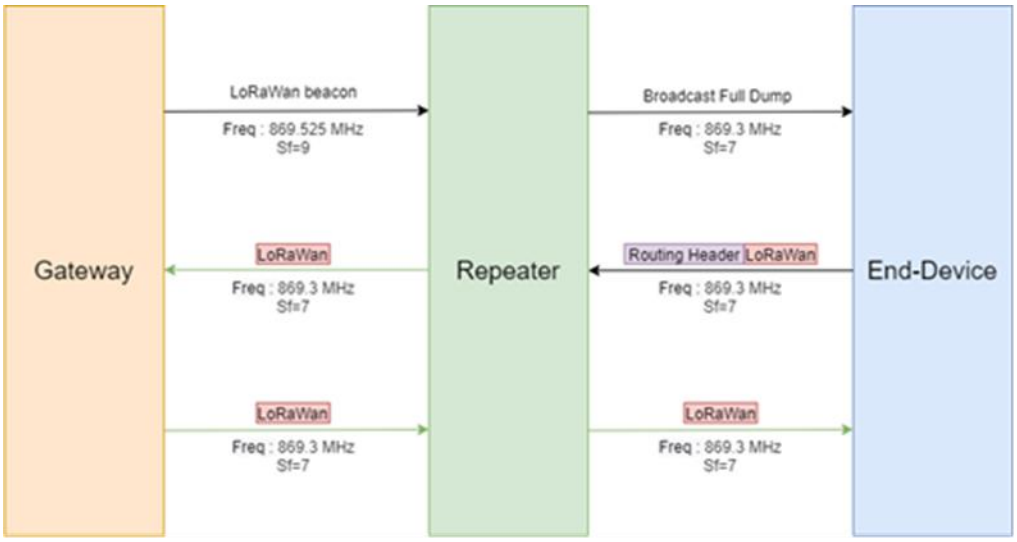

Figura 22 - Transmissão de pacotes de dados com Uplink e Downlink

A rede apresentada trata-se de uma rede multi-hop, ou seja, permite saltos entre dispositivos para comunicação através de um protocolo de Routing. O repetidor vai comunicar com vários dispositivos ao mesmo tempo e terá necessidade de ter uma Routing Table atualizada.

Para tal, a cada intervalo de tempo definido, o gateway irá emitir um beacon com informações para que caso seja necessário, o repetidor possa atualizar a sua Routing Table. Após a receção do beacon, o repetidor emite um pacote com as informações novas de ligações para todos os dispositivos ao seu alcance. Quando um end-device deseja comunicar com um gateway através do repetidor, terá de incluir um Routing Header no pacote, de forma que seja tratado e enviado corretamente pelo repetidor.

- **Autonomia:**

Devido ao estado do projeto não foi possível testar o Repetidor em fase de carga (com os painéis solares a carregar a bateria), mas já se conseguiu ligar a bateria e módulo de gestão energética à placa de expansão Pygate com a LoPy4 acoplada. Os testes de consumo efetuados foram feitos com um end-device (LoPy4) nos seguintes modos de funcionamento:

- LoPy4 em sleep-mode;
- Lopy a conectar-se à rede.

Os valores obtidos para os dois casos acima foram de potência instantânea, em mili-watt, e foram os seguintes (Tabela 4).

Casos de estudo	Potência Instantânea (mW)
LoPy4 em sleep-mode	136
LoPy4 a conectar-se à rede	185

Tabela 4 - Testes de consumo já efetuados

Com os testes efetuados já se pode ter uma ideia de que o sistema é de baixo consumo, e que caso a questão do firmware tivesse sido resolvida este sistema estaria pronto a ser diretamente integrado no projeto.

Testes relativos à interface de utilizador podem ser encontrados no documento Developer_guide_server_external_app que começa por apresentar a ideia inicial para o desenvolvimento da interface gráfica, a evolução da mesma e os procedimentos efetuados para chegar à versão final da mesma.

Com os equipamentos disponibilizados pela empresa também possível correr um simples código em 4 placas diferentes a simular valores de bateria que aparecem no servidor Chirpstack e de seguida foram recebidos na aplicação externa com o objetivo de, por fim, essa informação detalhada fosse apresentada na interface gráfica.

Milestones Importantes

ID	Descrição	Data atual de conclusão planeada	Data de conclusão
1	Pesquisa	23/01/2021	23/01/2021
2	Comunicação simples entre os dispositivos	14/04/2021	14/04/2021
3	Descodificação de um pacote lorawan	16/04/2021	16/04/2021
4	Implementação LoRaRaw/LoRaWAN para testes	29-03-2021	01/04/2021
5	Arquitetura do sistema revista	-	03/05/2021
6	Alteração de firmware	10/06/2021	-
7	Projeto do sistema de autonomia painel solar/bateria	22/05/2021	05/05/2021
8	Interligação do sistema desenvolvido com a API	13/06/2021	-
9	Versão final da API	10/06/2021	-

Tabela 5 - Tabela que apresenta as características das milestones importantes

Progresso e Desvios do Plano

O plano inicialmente proposto pelo grupo sofreu diversas revisões ao longo do decorrer do projeto. A principal foi quando se encontrou o problema de alteração de firmware a equipa teve de se ajustar rapidamente para procurar novas soluções e analisar qual a sua viabilidade sempre com o objetivo de reduzir o impacto no desenvolvimento no projeto. A solução passou por passar 3 membros do grupo para a investigação sobre o firmware deixando a questão de organização da Whitelist para segundo plano.

A interface de utilizador passou por várias fases e todos os obstáculos encontrados durante o seu desenvolvimento foram ultrapassados pelo que o grupo conclui que este requisito do projeto está dentro do pretendido.

As janelas de receção sempre foi um tópico com bastante interesse para a empresa uma fez que era pertinente a continuidade do projeto e conhecimento geral. Posto isto o grupo dedicou-se a esta parte e desenvolver uma pesquisa com 2 membros neste assunto.

Riscos

Ao longo desta fase não apareceram novos riscos sendo que a maior parte já foram ultrapassados. No entanto, em seguida são apresentados alguns que ainda se mantêm e são considerados de elevada importância.

1. **Falha na adaptação do firmware na placa de desenvolvimento**

Uma vez que a adaptação de firmware se está a revelar muito trabalhosa, apesar de cumprir todos os requisitos, é um risco bastante provável que não se consiga concluir esta adaptação. Sendo assim, tenciona-se começar agora a estudar uma segunda abordagem que poderá ser mais simples, no entanto comprometerá um dos requisitos do projeto relativos à sua implementação low power.

2. **Falha de segurança 1**

Alguém que encontre a APP-key que estamos a utilizar e a coloque num dos seus dispositivos, passando estes a comunicar com o nosso servidor.

3. **Falha de segurança 2**

Alguém que aceda ao IP do servidor e encontre o jwt *Token* que se está a utilizar, podendo utilizar as API *calls* em que pode obter informações privadas, ou até mesmo eliminá-las.

Estado Financeiro

#	Categoria	Descrição	estado	Valor
1	Componentes	2x Painel solar 12V 3W	Recebido	27,00 €
2	Componentes	Bateria Li-Po 3.7V 7Ah	Recebido	33,21 €
3	Componentes	Cabo USB à Micro-USB	Recebido	1,20 €
4	Componentes	Solar Power Manager	Recebido	25,33 €
			Total:	86,74 €

Tabela 6 - Tabela do estado financeiro

Contribuição do Grupo

Membro do grupo	Contribuições	Work share (%)
Carolina Pinho	**Firmware**	**16**
Ana Vicente	**Janelas de receção**	**13**
Gonçalo Ferreira	**Firmware**	**22**
Francisco Oliveira	**Autonomia e firmware**	**16**
Henrique Chaves	**Interface**	**20**
Miguel Amorim	**Janelas de receção**	**13**

Tabela 7 - Tabela da contribuição do grupo

Referências

[1] Repositorio pycom-micropython-sigfox/LoRa_pkt_fwd.catDev · pycom/pycom-micropython-sigfox (github.com)

[2]Pygate Datasheet, Pycom
https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_Pygate_v1.pdf?fbclid=IwAR3swCdLBrQXKuThX4-8-7S4d9FbbfNfhXGrOhggnVkTjOjk3cMAupIOrtE
[3]Techplayon, "LoRa – Device activation call flow (Join procedure) using OTAA and ABP". Disponível em: http://www.techplayon.com/lora-device-activation-call-flow-join-procedure-using-otaa-and-abp/ [Acedido em 28/12/2020]
[4] LoRaTools, "Calculate the air time of your LoRa frame". Disponível em: https://www.loratools.nl/#/airtime
Pinho C., Oliveira F., Amorim M., Vicente A., Chaves H. e Ferreira G. "*Inception*".
Pinho C., Oliveira F., Amorim M., Vicente A., Chaves H. e Ferreira G. "*Elaboration*".
Pinho C., Oliveira F., Amorim M., Vicente A., Chaves H. e Ferreira G. "*Construction1*".
Pinho C., Oliveira F., Amorim M., Vicente A., Chaves H. e Ferreira G. "*Construction2*".