41488 – Projeto Industrial

Developer guide de conceitos fundamentais, procedimentos e próximos passos.

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 junho de 2021
Supervisor:	Rui Manuel Escadas

Resumo: Este documento apresenta teoria fundamental necessária para o desenvolvimento de um projeto baseado em tecnologia LoRa. Apresenta também passos dados pelos elementos do grupo durante o desenvolvimento do projeto que envolveram processos/ comandos mais técnicos que serão aqui referenciados.

Todo o documento é da autoria do grupo 4 da unidade curricular de projeto industrial.

1. Teoria Geral do projeto

• LoRa

O projeto Repetidor Lora, baseia-se, como o próprio nome indica na tecnologia LoRa. LoRa, Long Range, é uma técnica de modulação de redes de longa distância e baixo consumo de potência (LPWAN).

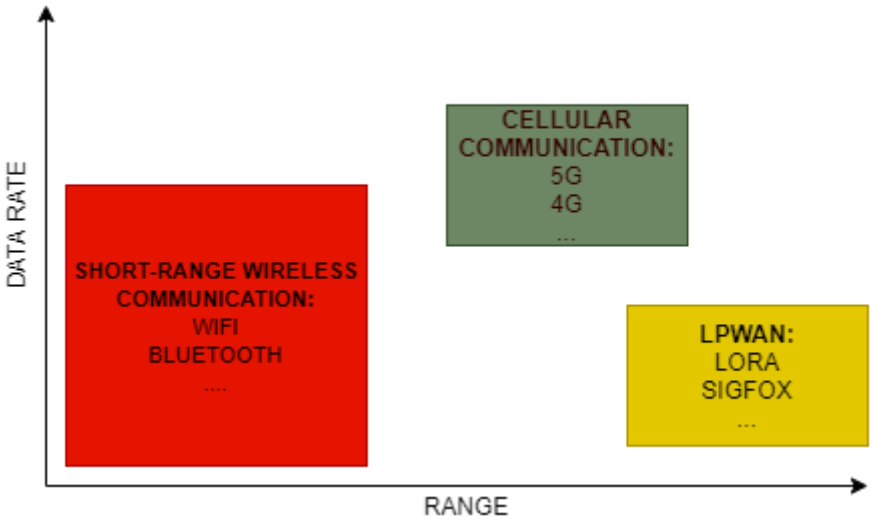


Figure 1 - Data Rate vs. Range

Como é possível verificar na figura 1, as tecnologias LPWAN têm como principal característica o seu longo alcance e baixo consumo de potência. Mas, para tal acontecer, teve que se recorrer a uma diminuição da largura de banda, o que provoca um menor Data Rate. Uma consequência desta característica é a impossibilidade de enviar imagens ou som via LoRa, limitando-se a enviar apenas, dados, como por exemplo, informação lida por um sensor.

Existem ainda algumas restrições ao trabalhar com LoRa, mais concretamente a nível de frequência. As frequências disponíveis para utilização encontram-se na banda ISM (863 a 870MHz) que está disponível para quem a queira utilizar sem necessidade de quaisquer licenças. Daqui vem uma das restrições que, devido a ser uma banda que pode ser utilizada por qualquer pessoa, a data rate diminui.

Com o objetivo de solucionar esta restrição foram especificadas duas regras:

- A potência máxima de transmissão permitida pela tecnologia é de 14dBm.
- O duty-cycle, ou o tempo ativo num certo período, de um dispositivo deve variar entre 0.1% e 1%.

A figura abaixo mostra dois tipos de mensagem que ocorrem entre um dispositivo e um gateway LoRa, denominadas de mensagens de Uplink (dispositivo para gateway) e Downlink (gateway para dispositivo).

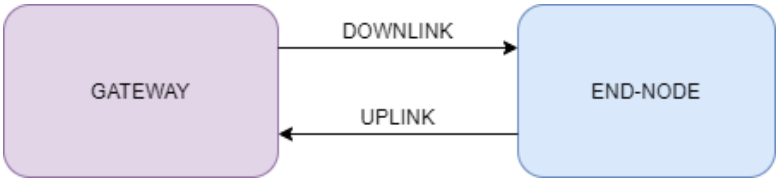


Figure 2 - Dois tipos de mensagem LoRa.

Modulação LoRa

Lora utiliza o protocolo “chirp”, baseado em Chirp Spread Spectrum (CSS) que utiliza modelação em frequência de forma a ser possível aumentar o alcance da onda. A onda enquanto é transmitida é aumentada na freq até chegar à máxima frequência.

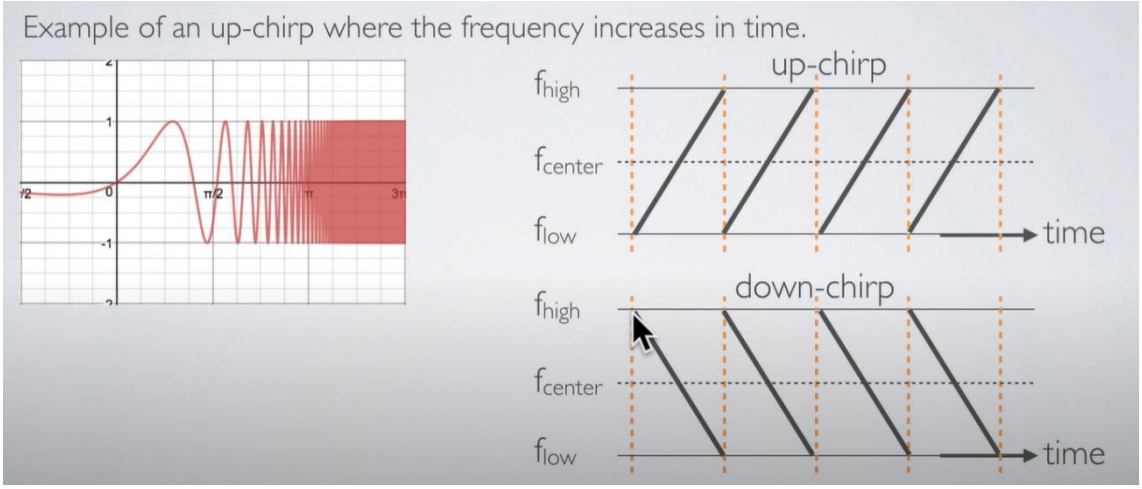


Figure 3 – Exemplo de modulação LoRa.

Para tal existem diferentes *spreading factors*. O SF relaciona o número de chips contidos em cada símbolo pela equação 2^(SF) chips=1 símbolos. Sendo que para cada SF o data rate (inversamente proporcional) muda como apresentado em baixo.

Spreading factor (at 125 kHz)	Bitrate	Range (indicative value, depending on propagation conditions)	Time on Air (ms) For 10 Bytes app payload
SF7	5470 bps	2 km	56 ms
SF8	3125 bps	4 km	100 ms
SF9	1760 bps	6 km	200 ms
SF10	980 bps	8 km	370 ms
SF11	440 bps	11 km	740 ms
SF12	290 bps	14 km	1400 ms
(with coding rate 4/5 ; bandwidth 125Khz ; Packet Error Rate (PER): 1%)			

Figure 4 - Relação data rate com Spreading Factor.

Quanto maior for o SF, mais energia será necessário para transmitir e maior o tempo de propagação, podendo vir a atingir maiores distâncias.

A largura de banda normalmente utilizada é 125KHz (chirps por segundo), podendo se usar 250KHz.

- **LoRaWAN**

A arquitetura de uma rede LoRaWAN baseia-se numa arquitetura em estrela, ou seja, todos os dispositivos estão unidos ponto a ponto, a um nó central. Esta arquitetura tem a vantagem de ser fácil de configurar, não ser necessário parar o funcionamento da rede para inserir ou remover dispositivos e é fácil de detetar dispositivos avariados.

A rede LoRaWAN é dividida em quatro componentes:

- **End Nodes:** estes dispositivos apenas saem sleep mode quando um ou mais sensores notam uma mudança, permitindo assim economizar energia.
- **Gateways:** recebem transmissões de End Nodes e enviam dados para os End Nodes.
- **Servidores de Rede:** é componente central de uma rede LoraWan. É responsável por consolidação da mensagem (analisar a qualidade dos pacotes recebidos), decide a melhor rota para uma mensagem para um End Node (calculado com base na Indicação de Força de Sinal Recebido (RSII) e na relação Sinal-Ruído (SNR)) e controla a rede.
- **Servidores de Aplicação:** é responsável por descodificar os dados recebidos dos sensores e codificar os dados enviados aos End Nodes. Pode estar junto com os Servidores de Rede.

Método do acesso aos canais

O protocolo utilizado por LoRaWAN é o Aloha, que consiste em enviar sempre que tivermos dados e se ocorrer uma colisão, temos de enviar os dados novamente.

A comunicação **classe A** é baseada neste protocolo, por isso temos que a comunicação vai ser sempre iniciada por um end point e vamos ter uma comunicação assíncrona. Ou seja, a qualquer momento pode ocorrer um uplink (end point envia para gateway) e de seguida iremos ter duas janelas de downlink curtas (1s e 2s). Se o servidor não responder em nenhuma das janelas (Figura3 - 1), apenas poderá responder no próximo uplink. O servidor pode responder na primeira (Figura3 - 2) ou na segunda janela (Figura3 – 3) e não deve usar as duas janelas. Concluindo, esta classe é a que consome menos energia, uma vez que o end point pode entrar em modo sleep e não é necessário que este seja despertado pela rede.

A comunicação classe B é baseada na comunicação classe A, mas temos que as janelas de downlink são sincronizadas à rede, usando sinalizadores periódicos. Ou seja, podemos enviar comunicações downlink com uma latência programável. Com isto, vamos aumentar o consumo de energia.

A comunicação classe C é baseada na comunicação classe A, mas reduz ainda mais a latência do downlink, sendo que mantém o recetor do end point sempre aberto, fechando apenas quando está a transmitir. Apesar de ficarmos sem latência (muito reduzida) comprometemos muito os dispositivos autónomos, sendo que os dispositivos classe C são adequados para aplicações onde a energia contínua esteja disponível.

Limitações de LoRaWAN

LoRaWAN não é adequado para todos os casos, este protocolo tem algumas limitações. Não é adequado por exemplo para dados em tempo real, uma vez que apenas podemos enviar pequenos pacotes a cada dois minutos.

Para o envio de dados de um end point envia para gateway (uplink) podemos concluir que as principais limitações são:

- O tamanho do pacote deve ser o menor possível, é necessário codificar os dados em binário;
- O intervalo entre mensagens é de vários minutos;
- Taxa de transmissão de dados deve ser a maior possível, que leva ao aumento da largura de banda.
- Para o envio de dados de um gateway para um end point (downlink) podemos concluir que as principais limitações são:
- Um gateway não tem comunicação full-duplex, não conseguem receber e transmitir dados ao mesmo tempo;

- Taxa de transmissão de downlink é baseada na taxa de transmissão uplink;
- As mensagens downlink devem ser evitadas, se forem enviadas devem ser com um tamanho pequeno;
- Confirmar o uplink muitas vezes não é necessário.

- **LoRa vs LoRaWAN**

Figure 5 - Várias camadas de uma rede.

Para compreender a diferença entra LoRa e LoRaWAN é importante perceber em que camadas se encontram. Necessitamos então de perceber o modelo de referência OSI (Open System Interconnection) da ISO (International Organization for Standardization). Este modelo divide as redes de computadores em 7 camadas, em que cada camada tem uma certa funcionalidade, para isso é necessário que em cada camada funcione um determinado protocolo.

LoRa é uma tecnologia de modulação de rádio que se enquadra na camada física deste modelo. A camada física define as especificações para a transferência de dados, ou seja, é responsável por definir se a transmissão pode ou não pode ser realizada.

LoraWAN é uma rede (protocolo) que usa LoRa, que está na camada de dados. Esta camada é composta por duas subcamadas, o Logical Link Control (LLC) e o Media Access Control (MAC). O LLC é responsável por identificar protocolos de rede, por detetar, ou eventualmente corrigir erros que possam existir da camada física e por sincronização (receção, delimitação e transmissão de frames). A segunda subcamada, usa endereços MAC para conectar dispositivos e definir permissões para transferir e receber dados.

- **Estrutura de uma rede LoRa/LoRaWAN**

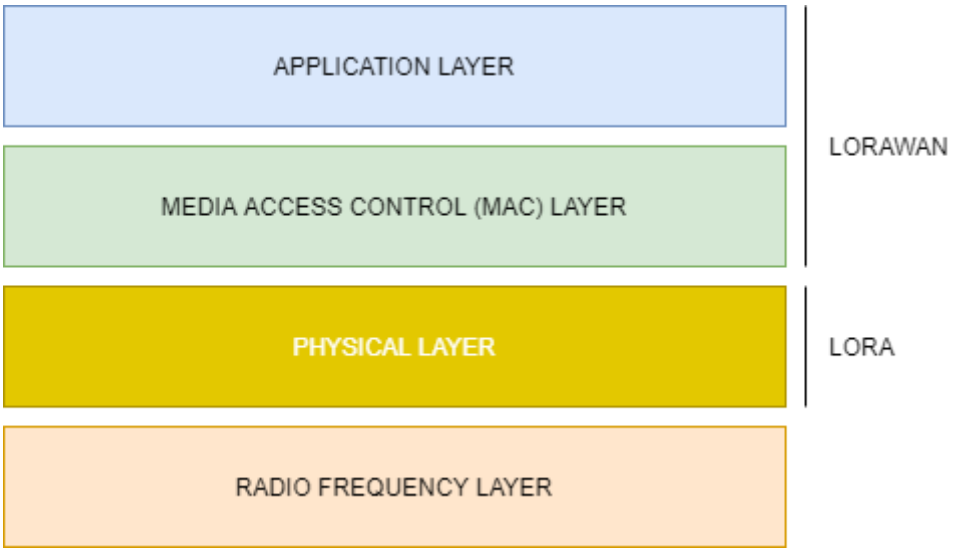


Figure 6 - Várias camadas de uma rede LoRa/LoRaWAN.

A diferença entre Lora e Lorawan é que o primeiro está presente na camada física, onde se encontra todo o hardware. O Lorawan está mais presente nas camadas acima onde ocorre o endereçamento e tratamento necessário para a comunicação.

- ## Janelas de Receção e Tempo no Ar

A tecnologia LoRa está muito dependente da configuração da janela de receção e a sua respetiva sincronização. A janela de receção é definida como o tempo que um dispositivo estará à espera de receber informação após terminar a sua transmissão.

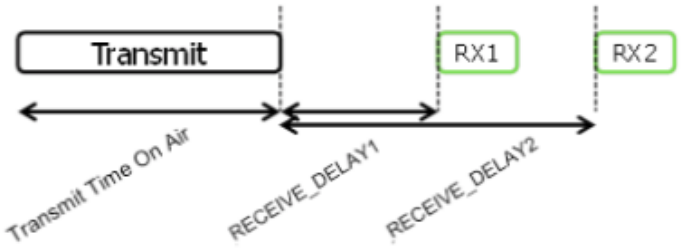


Figure 7 - Diagrama de funcionamento.

Como é possível ver através da imagem se um pacote chegar antes do tempo que a janela de receção está aberta, este pacote irá ser perdido e irá ser retransmitido no próximo downlink. Quando existem múltiplas mensagens a ser transmitidas e recebidas, uma errada configuração do tempo que esta janela demora a abrir para receber informação, pode levar a uma elevada taxa de perda de pacotes.

Fizeram-se os cálculos relativos ao tempo no ar para a seguinte comunicação:

Figura 8 - Comunicação base

Os cálculos foram efetuados com base no seguinte script:

```
BW=125000; % Largura de Banda Hz
n_preamble=8; %escolhido para banda Europeia 868MHz
% parâmetros
SF=9; %Spreading factor
PL=[20, 40, 51,70,90,150,200];  %payload em bytes
Ts=2^(SF)/BW;  %Periodo de transmissão de um simbolo = 2^SF
CRC=1; %cyclic redundancy check 1-> enabled 2-> disabled
H=0;  %Header 1->implicit mode 0->explicit mode
CR=1; %Coding Rate
if(SF>=11 && SF<=12)
    DE=1;  %LowDataRateOptimizer 1-> enabled 0->disabled
elseif (SF<11 && SF>=7)
    DE=0;  %LowDataRateOptimizer 1-> enabled 0->disabled
end
T_preamble=(n_preamble+4.25)*Ts;  %tempo envio da preamble
T_payload=Ts*(8+(max(ceil((8.*PL-4.*SF+28+16.*CRC-20.*H)./(4.*(SF-
2*DE))).*(CR+4),0))); %tempo envio da payload
T_processamento=0.5; %a alterar aquando da implementação final
TonA=T_payload+T_preamble+T_processamento; % tempo no ar (tempo de envio de
pacote) em segundos
```

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.

Obteve-se os seguintes resultados para uma comunicação ponto a ponto.

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

Logo, para a comunicação da figura 8 tem-se:

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

Quanto à cobertura 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

2. Descodificação de um pacote LoRaWAN:

Após a comunicação entre o *end-device* e o gateway estar estabelecida, foi necessário analisar os pacotes que seriam enviados entre estes dois pontos. Para isso começou-se por estudar que tópico MQTT é que seria necessário subscrever para ver a comunicação entre o gateway e o LoRaServer. Descobriu-se então que de forma a ver os eventos da gateway é necessário subscrever o tópico default "gateway/#" onde o "#" sinaliza todos os subtópicos que serão os ids das gateways ligados ao servidor.

A partir daqui já é possível ver mensagens que vão passando entre estes dois pontos como é possível ver na figura 9.

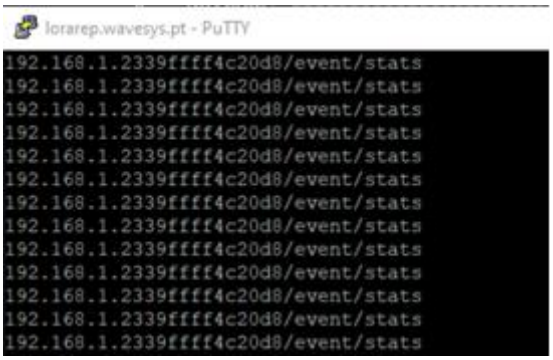


Figura 9 - Print screen terminal MQTT

De forma a facilitar o trabalho com os pacotes recebidos utilizou-se a plataforma Node-Red (NR)[3] que permitia também que se efetue a subscrição de tópicos MQTT usando um nó apropriado que se escuta o que passa-se pelo gateway.
Intitulou-se este nó de "nó MQTT", como é possível ver na figura 10, onde o bloco a verde serve simplesmente para visualizar o que passa no nó.

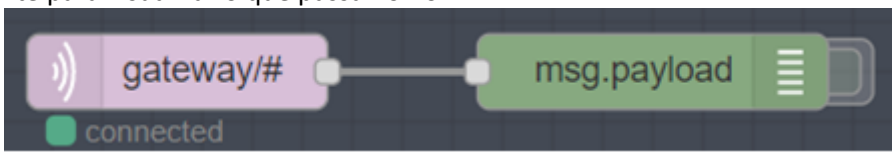


Figura 10 - Subscrição do subtópico MQTT de eventos "up" através do NR

Com isto, o próximo objetivo seria retirar o devEUI de modo a identificar o dispositivo que está a tentar comunicar com o gateway. Este vem num pacote do tipo *join request* efetuado pelo respetivo dispositivo, portanto o tópico da subscrição MQTT foi limitado a pacotes "up**"** alterando o Nó MQTT como se fez na figura 11.

Figura 11 - Subscrição do subtópico MQTT de eventos "up" através do Node Red

Quando se liga a LoPy4, que se comporta como um *device,* e se envia um *join request* , obtém-se no Nó MQTT as mensagens ilegível que se encontram na figura 12.

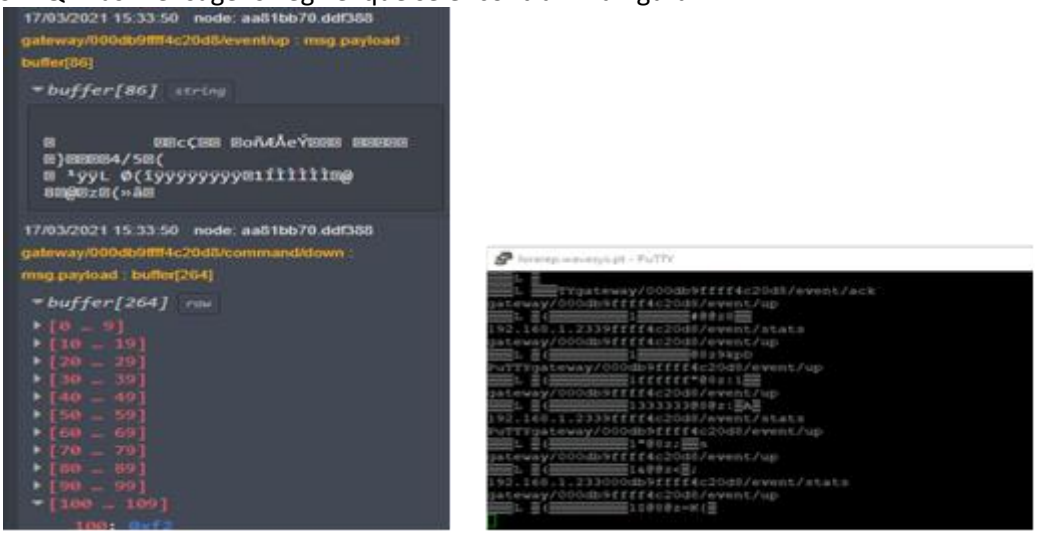


Figura 12 - Pacotes join request recebidos no canal MQTT subscrito

Rapidamente foi percebido que era necessário aplicar algum algoritmo de descodificação para ler a informação do pacote LoRaWAN recebido pela gateway. Na documentação

do ChirpStack[4] apareceu a primeira pista para a descodificação de pacotes LoRaWAN. Estes são codificados em *strings* de base64, pelo que ter-se-ia de configurar o nó MQTT para que este reconhecesse que as suas mensagens estavam em base64. Posteriormente, foi feita uma função "toHex" no **NR** de forma a transformar a saída do Nó MQTT em hexadecimal, verificamos através de um conversor online[5] que a função estava bem implementada.
Após esta primeira fase, faltava a parte principal da descodificação em si. Durante a pesquisa sobre este assunto encontrou-se uma ferramenta[6] que fazia a descodificação de pacotes LoRaWAN em hexadecimal. O principal desafio seria, portanto, desenvolver um módulo que fizesse esta descodificação em NR.
Chegou-se então à arquitetura apresentada na figura 6 que descodifica o pacote recebido.

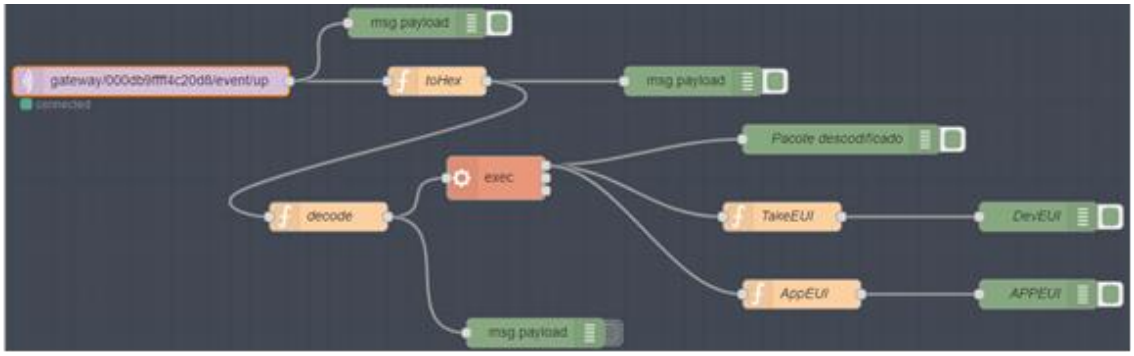


Figura 6 - Arquitetura final do flow MQTT desenvolvido em Node Red

Como é possível observar na Figura 6, recorrendo ao nó executável (*exec*) do **NR** foi possível correr o comando bash "lora-pACKet-decode –hex" com o *input msg.payload*, ficando com a descodificação desejada do pacote recebido (Fig. 7).
Assim, através do código desenvolvido em javascript criou-se a função "TakeEUI" que retirará o valor do *DeviceEUI* e a função "AppEUI" que devolve o valor *AppEUI* desejado, Fig. 8.
Com o identificador do *end-device* numa mensagem do tipo *msg.payload* do **NR,** cria-se esse dispositivo através de dois métodos da API:

- Um para criar o dipositivo na aplicação desejada do servidor com o nome "Device_LoRA_ (nº aleatório)", uma descrição coerente com o tipo de dispositivo;
- Outra para atribuir as *device-keys* OTAA em que a *key* da aplicação é igual ao longo do desenvolvimento do projeto e a *Gen Application key* como sendo zeros.

Figura 7 - Pacote após o nó exec

Figura 8 - devEUI retirado com o auxílio dos vários nós

De forma semelhante conseguiu-se descodificar uma mensagem recebida pelo repetidor que contém o devEUI do end-device que envia o join request para o servidor através do mesmo repetidor.

3. Interface Node-RED:

Recorrendo à ferramenta Node-RED desenvolveu-se uma interface que consoante o devEUi devolvido da função anterior, permite ao utilizador criar e ativar esse dispositivo no servidor através de um único botão de OK.

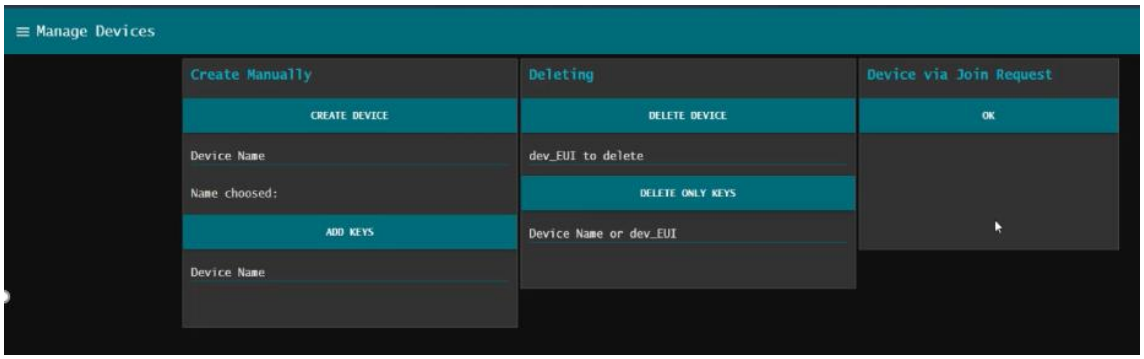


Figura 9 - Interface Node-RED

Para aceitar o request enviado pelo end-device basta clicar no botão OK à direita e esta aplicação externa através da API cria o dispositivo no servidor Chirpstack, adiciona as keys de OTAA e, consequentemente, ativa-o:

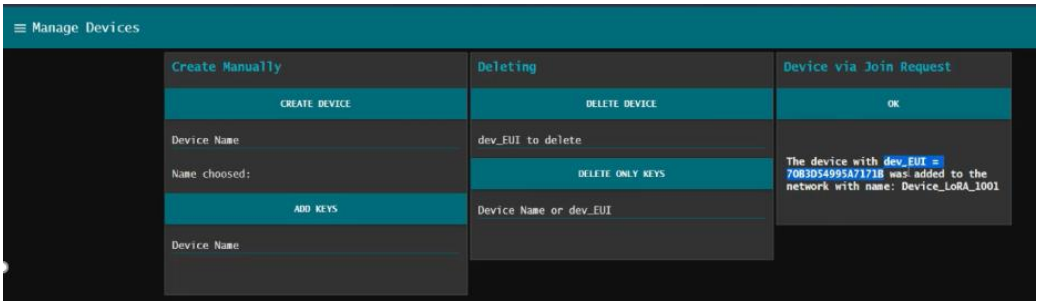


Figura 10 - Procedimento através da interface do Node-RED

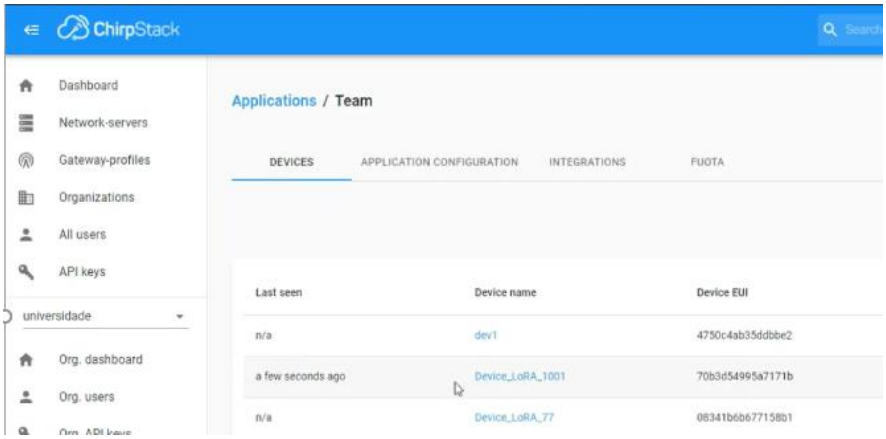


Figura 11 -Confirmação da ativação do respetivo dispositivo

Como próximos passos desta interface seria vantajoso adicionar uma forma do nome escrito na caixa de texto ser utilizado como nome do dispositivo a adicionar e também permitir escolher a APP key através da interface.

4. Firmware developer guide[7]

De forma a ser possível fazer o varrimento de firmware da placa, é necessário utilizar o esp-idf para tratamento e envio do código para a placa. É utilizado o esp-idf pois a arquitetura das placas da pycom é semelhante ao de um ESP32. Para este propósito, é necessário então instalar o esp-idf e declarar o seu caminho via IDF_PATH.

Finalizada a instalação é necessário utilizar também um compilador, neste caso o xtensa gcc compiler, que requer um update no PATH também de forma a estar disponível na sessão de terminal a utilizar. De seguida faz-se "clone" do repositório GitHUB com as pastas de firmware que, após análise se se encontram sem erros, poderemos utilizar para alteração e posterior envio para a placa.

Finalizadas todas as configurações, é possível começar a alterar código e a fazer upload para a placa. Para isso podemos entrar no diretório EPS32 onde se encontram as pastas relativas ao firmware das placas da PYCOM, e onde se encontra a pasta pygate que tem os ficheiros a ser alterados (possivelmente será necessário alterar outros após implementação aprofundada). Finalizadas as alterações introduz-se o código make ESPPORT=/dev/ttyACM0 BOARD=LOPY4 VARIANT=PYGATE flash, onde é especifica a porta onde se encontrada ligada a placa, o tipo de microcontrolador e de placa de expansão.

Para uma explicação e guia mais detalhado podemos ir ao github que tem um guia da instalação passo por passo.

5. Próximos passos firmware

No final do projeto apareceu uma opeção que poderá valer a pena explora. Foi discutido o uso dos dois transceivers sx1257, o nosso primeiro passo foi contactar os responsáveis do fórum GitHUB relativo ao firmware da placa de forma a tentar perceber se o uso de um como recetor e outro como transmissor seria possível.
A resposta dos responsáveis foi que de tal, em princípio seria possível, no entanto seria necessária uma alteração dos registos no processador SX1308 que está conectado aos dois transceivers. Como é possível ver na imagem, apenas o rádio A permite uma comunicação bilateral, enquanto o rádio B apenas serve para receção. Seria então necessário alterar, através dos registos, os caminhos para o os pacotes TX via radio A e os pacotes RX via radio B.
A comunicação entre rádios e processador ocorre segundo o protocolo SPI, onde o processador é considerado o Master e os rádios slaves. As funções relativamente a essas comunicações encontram-se na pasta drivers/sx1308 no repositório GitHub.

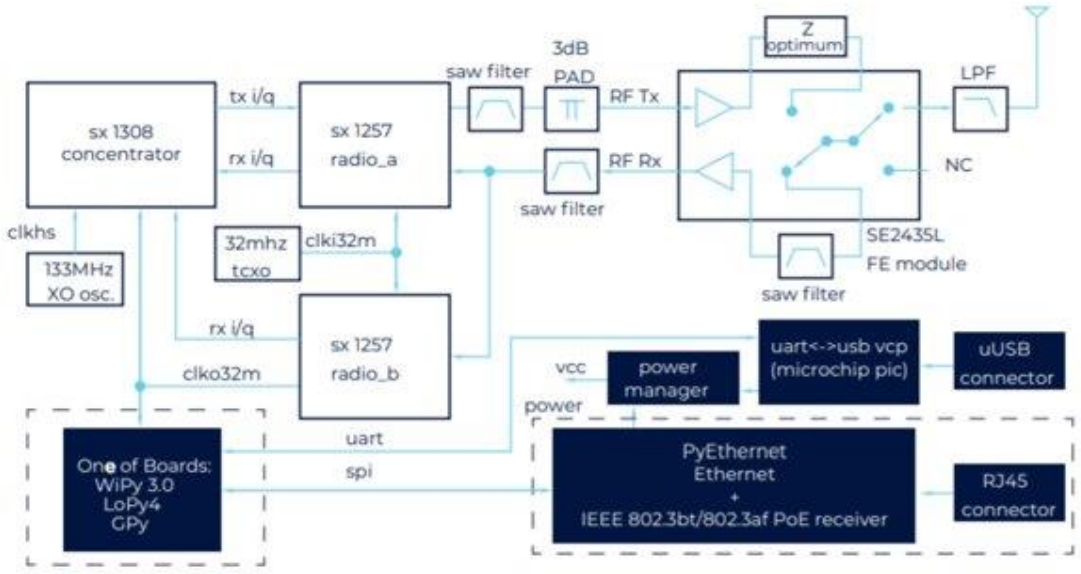
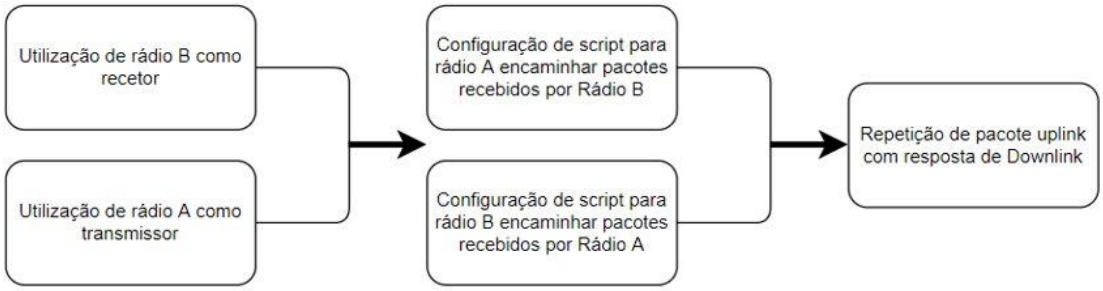


Figura 11 -Arquitetura interna pygate [8]

Os próximos passos a tomar neste projeto são os apresentados no seguinte diagrama:



Inicialmente, utilizando os registos no ficheiro SX1308.c, será necessário trabalhar em cada rádio isoladamente, podendo o trabalho ser feito em paralelo. Após implementada a receção e transmissão nos rádios específicos, é necessário fazer a ligação entre eles, tanto para de B para A (uplinks) como de A para B (downlinks). Essa ligação nunca será direta, será necessário sempre usar o processador como meio, podendo para tal usar os registos igualmente. Finalizados esses passos podemos começar a integrar esses dois módulos de forma a chegar ao modelo final do sistema, um repetidor. Mais tarde poderiam sem implementados métodos de agregação de pacotes de forma aquando for recebido um pacote pequeno, o repetidor esperar pela receção de outro para poder mandar os dois juntos, no entanto isto são questões que só poderão ser abordadas após a implementação do repetidor.

Referências:

[1] *https://www.youtube.com/playlist?list=PLmL13yqb6OxdeOi97EvI8QeO8o-PqeQ0q - Lora/LoraWAN tutorial playlist*
[2] *https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country.html - LoRaWAN Frequency Plans and Regulations by Country*

[3] Node-RED, “Documentation”. Disponível em: Documentation : Node-RED

[4] ChirpStack, “Application Server - Integrations”. Disponível em: Event types - ChirpStack open-source LoRaWAN® Network Server
[5] Cryptii, “Base64 to hex:Enconde and decode bytes online”. Disponível em: Base64 to hex: Encode and decode bytes online — Cryptii
[6] LoRaWAN packet decoder. Disponível em: LoRaWAN 1.0.x packet decoder (runkit.sh)
[7] GitHub, “Pycom-micropython-sigfox”. Disponível em: GitHub - pycom/pycom-micropython-sigfox: A fork of MicroPython with the ESP32 port customized to run on Pycom's IoT multi-network modules.
[8] Pygate datasheet- pycom
https://docs.pycom.io/gitbook/assets/specsheets/Pycom_002_Specsheets_Pygate_v1.pdf?fbclid=IwAR3swCdLBrQXKuThX4-8-7S4d9FbbfNfhXGrOhggnVkTjOjk3cMAupIOrtE