RAPPORT PROJET LORA

Table des matières

RGUIBI Taha

BENZAOUIA Oussama

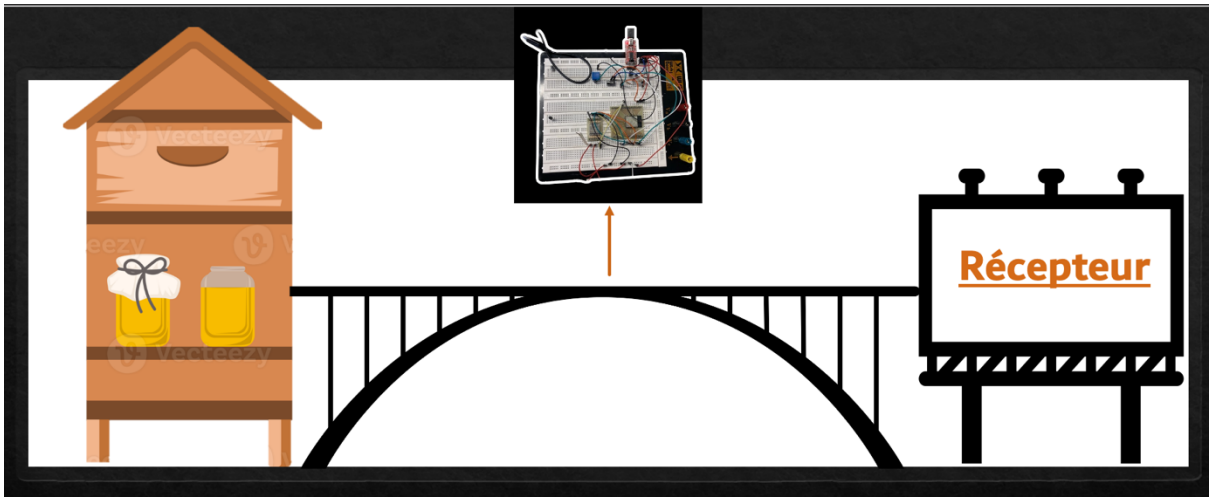
1) Contexte Globale de la carte :

a) Objectifs :

Le système final contient un modules LoRa et un uC ESP32

- Émission trames LoRa
- Réception trames LoRa
- Réduire la consommation du uC
- Réduire la consommation du module LoRa

Relais qui va nous servir à rallonger la portée de la transmission des données de capteurs de la ruche et qui va les transmettre.



2) Problématique objectif :

Il existe aujourd'hui un certain nombre de technologies de communication disponibles pour l'interaction entre les dispositifs IoT, et les plus populaires sont le WI-FI et le Bluetooth. Le problème de ces deux dispositifs est leur forte consommation d'énergie.

Ils ont également des limites comme une portée limitée, des points d'accès limités.

Les réseaux cellulaires ont également les mêmes problèmes de consommation d'énergie élevée et les réseaux LAN et cellulaires sont assez coûteux pour couvrir une large zone.

Les industries de l'IOT ont introduit de nombreuses technologies, mais aucune d'entre elles n'était idéale pour les appareils IOT, jusqu'à ce que la technologie LoRa soit introduite.

La technologie LoRa peut effectuer des transmissions à très longue distance tout en consommant peu d'énergie.

LoRa (Long Range) est une technologie sans fil qui offre une transmission de données à longue portée, à faible consommation d'énergie et sécurisée pour les applications machine à machine et IOT. Dans la région européenne, elle fonctionne dans la bande **868 MHZ** et, selon les informations officielles, LoRa peut atteindre une distance de **715 km** lorsqu'il n'y a pas d'obstacle entre le nœud et la passerelle.

Ici, l'objectif de ce projet est d'avoir un module qui fera le relais entre la ruche et le **module GSM**, donc la ruche va envoyer des données (température, humidité etc) qui vont être transmis avec un protocole d'envoi LoRa à notre carte qui elle va le transmettre vers le GSM.

Vous allez vous dire que le module va être allumer **24/24, 7/7** et donc l'énergie de la batterie va être consommer en peu de temps c'est vrai c'est donc pour ça qu'on avait comme objectif d'avoir un protocole d'envoi LoRa comme expliqué ci-dessus LoRa est un protocole de communication qui ne demande pas beaucoup de puissance pour un envoi de grande portée et en plus de ça l'**ESP32** et le module **Rf SX1262** vont rentrer en sleep mode au moment où il n'y a pas d'envoi de valeur des capteurs et rentrer en active mode le moment où il reçoit une trame et donc envoyer au GSM.

Comme le relais doit fonctionner 24h/24

Le relais doit fonctionner **24h/24**, on aura besoin d'un système qui fonctionne avec une consommation basse d'énergie, et donc le protocole de communication **LoRaWAN** est le plus adapter à ça.

Après il va falloir mettre l'**ESP32** et le **SX1262** en **Deep sleep** mode en cas de non envoie de trame comme ça on va avoir une consommation de 2uA au côté du **SX** et **40uA** au côté du **uC**.

Cependant, on active mode (quand le relais doit faire soit l'envoie ou la réception de trame de données) on a une consommation de **200mW.**

Nous avons installé les librairies **LoRa avec Sandeep Mistry sur Arduino Ide** pour envoyer et recevoir facilement des paquets **LoRa avec l'ESP32**.

```
14:26:08.178 -> Received packet 'Hello World!' with RSSI -53
14:26:08.178 -> waiting for packet
14:26:09.206 -> waiting for packet
14:26:10.186 -> Received packet 'Hello World!' with RSSI -53
14:26:10.186 -> waiting for packet
14:26:11.213 -> waiting for packet
14:26:12.190 -> Received packet 'Hello World!' with RSSI -53
14:26:12.190 -> waiting for packet
14:26:13.176 -> waiting for packet
14:26:14.200 -> Received packet 'Hello World!' with RSSI -52
14:26:14.200 -> waiting for packet
14:26:15.207 -> waiting for packet
14:26:16.189 -> Received packet 'Hello World!' with RSSI -52
14:26:16.189 -> waiting for packet
14:26:17.221 -> waiting for packet
14:26:18.202 -> waiting for packet
14:26:19.180 -> Received packet 'Hello World!' with RSSI -60
14:26:19.226 -> waiting for packet
```

Les tests :

L'objectif est de mettre l'ESP32 et le SX1262 en sleep mode, pour le SX1262 on a utilisé radiolib

3) Architecture Hardware :

Le système de test contient principalement deux blocs :

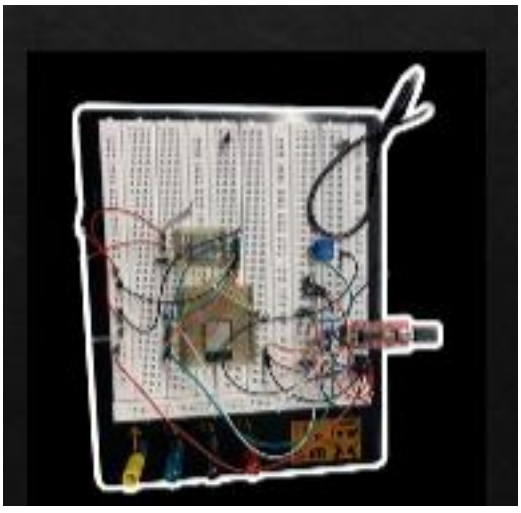
- Émetteur LoRa + uC.
- Récepteur LoRa + uC

Heltec WiFi LoRa 32 cette carte de type ESP32 embarque des composants WiFi, Bluetooth à basse consommation. Il y a aussi une antenne Intégrée assez performante, un système de gestion de batterie LiPo ainsi qu'un écran OLED de 0,96 pouces.

Le module Heltec wifi LoRa simule l'envoi/la réception des trames de la ruche.

On trouve le code directement dans la librairie Heltec wifi LoRa 32 sur Arduino pour l'envoie et la réception.

Et le SX1262 comme point de relais



1) Architecture HW :

Notre système contient les éléments suivants :

- Un **FT232** pour programmer le uC et le circuit qui va avec (BP : rst/Boot)
- Un module LoRa communiquant en SPI avec le uC
- **Un microcontrôleur EPS32**

2) Choix des composants :

- FT232 : disponibilité
- ESP32 : disponibilité + Utilisé dans la globalité du projet (les autres équipes) + Lib/Forum disponible
- RFM95 : Module utilisé anciennement
- SX1262 : Lib dispo + nouvelle génération de module **RF** + sleep mode similaire au **RFM95** mais se programme avec la librairie **RadioLib.**

3) Déroulement du projet :

- Familiarisation avec les composants + Tests Unitaires de l'**ESP32** et **module RF**
- Brainstorming sur les **solutions HW et Soft**, en utilisant le **Deep sleep** du uC et du **module LoRa.**

Discussion de Solutions :

1ére solution : Gestion du mode sleep des 2 modules
Réception de trames RF -> Réveil uC -> Action

2éme solution : uC en Sleep/ shutdown RF (switch transistor) wakeup périodique

On a décidé de commencer par la **1ére solution** étant la plus complexe, mais ça permettait l'optimisation de la consommation car le uC ne se réveillerai que quand le RF reçoit une trame. Et la **2éme solution** est relativement plus simple à mettre en place.

1) Tests de la 1ere solution :

- **ESP32** en sleep mode (Réveil Extern/periodique)
- **RF** en sleep mode **the tricky part.**

a. Problèmes rencontrés :

Sur le RF :

On travaille un module où la **puce SX1262** est déjà intégré, donc ça limité l'utilisation par rapport à la datasheet de la puce

Sur la carte il manquait des connexions entre **le module RF et l'ESP32 .**

Pas assez d'info sur le doc tech du module complet.

Des fuites de courant en mA même en absence du **3V3.**

Ne fallait pas utiliser des pins de l'**ESP32** avec le **RF** car l'esp ne marchait plus.

b. Conclusion et pistes :

- Essayer des modules de switch d'alim contrôlé par un u**C** (coupé le gnd et le Vcc si c'est possible).
- Si le **deep sleep du SX1262** est infaisable en **Soft** => le forcer en **HW** avec des switchs au **niveau du Vcc,** si ce dernier ne suffit pas rajouter un pour le **GND**.
- Essayé des modules dans des **eval boards,** pour se franchir relativement des **bugs HW**.
- Etudier en Details les pins du **RF (Gpio)** pour réveiller le **uC** directement.