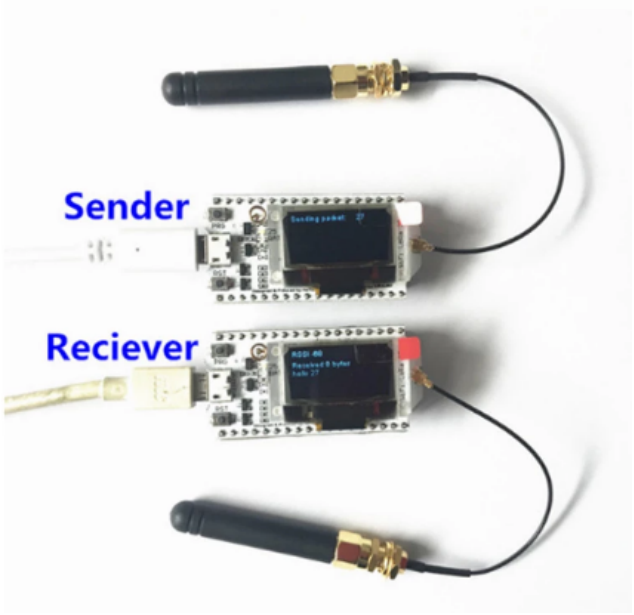
TP 3 CARTE HELTEC V3 LORA
TRANSMISSION LORA PAIR À PAIR.

Objectifs :

- Faire communiquer plusieurs équipements LoRa entre eux
- Ajuster les caractéristiques d'un couple d'émetteur et de récepteur afin que les échanges ne se mélangent pas s'il y a plusieurs émetteurs/récepteurs dans la même zone.
- Afficher le niveau de réception et évaluer son évolution en fonction de la distance.



1. TABLEAU DE REPARTITION DES BINOMES

1.1. Cocher et noter le numéro de binôme que le professeur vous a attribué, relever la fréquence et le facteur d'étalement de votre binôme. Il faudra les remplacer dans les programmes.

binôme	Fréquence	Facteur d'étalement (Spreading Factor)
1 ❑	867100000	7
2 ❑	867100000	8
3 ❑	867500000	7
4 ❑	867500000	8
5 ❑	867900000	7
6 ❑	867900000	8
7 ❑	868300000	7
8 ❑	868300000	8

2. TRAVAIL A EFFECTUER EN BINÔME **CHAQUE ELEVE REALISE SA PARTIE**

Les cartes utilisées sont des ESP32 LoRa Heltec V3.

Élève 1 : carte émettrice :

2.1. Lancer Arduino choisir le type de carte **Heltec Wifi Lora 32 V3.**

2.2. Créer un fichier LoraSender avec à partir du fichier :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Lora2Lora/LoRaSenderOLEDv3.ino

2.3. Modifier le fichier de la manière suivante :

- En jaune les lignes à rajouter/ modifier
- En vert la valeur du facteur d'étalement de votre binôme.

```
#include "LoRaWan_APP.h"
#include "Arduino.h"
#include <Wire.h>
#include "HT_SSD1306Wire.h"

SSD1306Wire  oledDisplay(0x3c, 500000, SDA_OLED, SCL_OLED, GEOMETRY_128_64, RST_OLED); // addr , freq ,
i2c group , resolution , rst

#define BINOME  VOTRE_NUMERO_DE_BINOME
#define RF_FREQUENCY      VOTRE_FREQUENCE                // Hz, fréquence LoRa

#define TX_OUTPUT_POWER                              5    // dBm,
#define LORA_BANDWIDTH                               0    // [0: 125 kHz, 1: 250 kHz, 2: 500 kHz]

#define LORA_SPREADING_FACTOR   VOTRE_SPREADING_FACTOR   // [SF7..SF12]

#define LORA_CODINGRATE                              1    // [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]

#define LORA_PREAMBLE_LENGTH                         8
#define LORA_SYMBOL_TIMEOUT                          0
#define LORA_FIX_LENGTH_PAYLOAD_ON                   false
#define LORA_IQ_INVERSION_ON                         false

#define BUFFER_SIZE                                  30 // Taille du payload

unsigned int packetCounter = 0; // Compteur de paquet
bool lora_idle=true;
static RadioEvents_t RadioEvents;
void OnTxDone( void );
void OnTxTimeout( void );

void setup() {
    Serial.begin(115200);
    Mcu.begin();
    Serial.println();
    VextON();
    delay(100);

    // Initialising the UI will init the display too.
    oledDisplay.init();
    oledDisplay.setFont(ArialMT_Plain_10);

    RadioEvents.TxDone = OnTxDone;
    RadioEvents.TxTimeout = OnTxTimeout;

    Radio.Init( &RadioEvents );
    Radio.SetChannel( RF_FREQUENCY );
    Radio.SetTxConfig( MODEM_LORA, TX_OUTPUT_POWER, 0, LORA_BANDWIDTH,
                       LORA_SPREADING_FACTOR, LORA_CODINGRATE,
                       LORA_PREAMBLE_LENGTH, LORA_FIX_LENGTH_PAYLOAD_ON,
                       true, 0, 0, LORA_IQ_INVERSION_ON, 3000 );
}
```

```
void VextON(void) {
    pinMode(Vext,OUTPUT);
    digitalWrite(Vext, LOW);
}

void loop() {
    char txpacket[BUFFER_SIZE];

    if(lora_idle == true) {
        packetCounter++; // Incrémentation du compteur de paquets
        sprintf(txpacket,"Bnm:%d Packet #%u", BINOME, packetCounter); // Préparation du message à
envoyer

        // Affichage du message sur l'écran OLED
        oledDisplay.clear();
        oledDisplay.drawStringMaxWidth(0, 0, 128, txpacket);
        oledDisplay.display();

        // Envoi du paquet
        Serial.printf("\r\nSending packet: \"%s\"\r\n", txpacket);
        Radio.Send((uint8_t *)txpacket, strlen(txpacket));
        lora_idle = false;
    }
    Radio.IrqProcess();
    delay(2000); // Petite pause pour la stabilité
}

void OnTxDone(void) {
    Serial.println("TX done......");
    lora_idle = true;
}

void OnTxTimeout(void) {
    Radio.Sleep();
    Serial.println("TX Timeout......");
    lora_idle = true;
}
```

Élève 2 : carte réceptrice :

2.4. Lancer Arduino choisir le type de carte **Heltec Wifi Lora 32 V3.**

2.5. Créer un fichier LoraReceiver avec à partir du fichier :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Lora2Lora/LoRaReceiverOLEDv3.ino

2.6. Modifier Le fichier de la manière suivante :

```
#include "LoRaWan_APP.h"
#include "Arduino.h"
#include <Wire.h>
#include "HT_SSD1306Wire.h" // Assurez-vous d'utiliser la bibliothèque correcte pour l'OLED

// Configuration de l'affichage OLED
SSD1306Wire oleddisplay(0x3c, 500000, SDA_OLED, SCL_OLED, GEOMETRY_128_64, RST_OLED);

#define BUFFER_SIZE 64 // Taille de buffer augmentée pour plus de flexibilité

// Paramètres LoRa
#define RF_FREQUENCY VOTRE_FREQUENCE // Hz, fréquence LoRa
#define LORA_BANDWIDTH 0 // [0: 125 kHz, 1: 250 kHz, 2: 500 kHz]
#define LORA_SPREADING_FACTOR VOTRE_SPREADING_FACTOR // [SF7..SF12]
#define LORA_CODINGRATE 1 // [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]
#define LORA_PREAMBLE_LENGTH 8 // Longueur du préambule
#define LORA_SYMBOL_TIMEOUT 0 // Timeout des symboles
#define LORA_FIX_LENGTH_PAYLOAD_ON false // Longueur fixe du payload désactivée
#define LORA_IQ_INVERSION_ON false // Inversion IQ désactivée

char rxBuffer[BUFFER_SIZE]; // Buffer pour les données reçues

static RadioEvents_t RadioEvents; // Évènements LoRa
```

```
void setup() {
Serial.begin(115200);
while (!Serial);

// Initialisation de l'écran OLED
oleddisplay.init();
oleddisplay.setFont(ArialMT_Plain_10);
delay(2000); // Délai pour la lecture
oleddisplay.clear();
oleddisplay.drawString(0, 0, "Init. OK!");
oleddisplay.drawString(0, 15, "Wait for data...");
oleddisplay.display();
delay(1000) ;

// Initialisation LoRa
Mcu.begin();
RadioEvents.RxDone = OnRxDone;
Radio.Init(&RadioEvents);
Radio.SetChannel(RF_FREQUENCY);
Radio.SetRxConfig(MODEM_LORA, LORA_BANDWIDTH, LORA_SPREADING_FACTOR,LORA_CODINGRATE, 0,
LORA_PREAMBLE_LENGTH, LORA_SYMBOL_TIMEOUT, LORA_FIX_LENGTH_PAYLOAD_ON,0, true, 0, 0,
LORA_IQ_INVERSION_ON, true);
Radio.Rx(0); // Commencez à recevoir
}

void loop() {
// La logique de traitement des interruptions LoRa est exécutée dans le background
Radio.IrqProcess();
}

// Callback appelé lorsqu'un paquet est reçu
void OnRxDone(uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr) {
memcpy(rxBuffer, payload, size);
rxBuffer[size] = '\0'; // Assurez-vous que la chaîne est correctement terminée

// Affichage des données reçues et du RSSI sur l'écran OLED
oleddisplay.clear();
oleddisplay.drawString(0, 0, "Paquet recu:");
oleddisplay.drawString(0, 15, String(rxBuffer));
oleddisplay.drawString(0, 30, "RSSI: " + String(rssi) + " dBm");
oleddisplay.display();

Serial.print("Paquet recu: ");
Serial.println(rxBuffer);
Serial.print("RSSI: ");
Serial.println(rssi);

Radio.Rx(0); // Remet le module LoRa en mode de réception
```

2.7 Compiler et téléverser le programme, indiquer ce qu'affiche le petit écran :

- Écran de l'émetteur :
- Écran du récepteur :

2.8 Changer le Spreading factor de L'émetteur en lui ajoutant 2 (par exemple 7+2=9), puis retéléverser le programme. Indiquer comment se comporte l'afficheur du récepteur.

2.9 Modifier le Spreading factor du récepteur en accord avec celui de l'émetteur, re téléverser alors le programme. Indiquer comment se comporte l'afficheur du récepteur.
Conclure sur la configuration du spreading factor entre l'émetteur et le récepteur.

2.10 Modifier la fréquence de l'émetteur en lui ajoutant 200kHz (par exemple 867100000 + 200000 = 867300000). Indiquer comment réagit le récepteur avec la nouvelle fréquence de l'émetteur, accorder alors le récepteur à l'émetteur.

2.11 Conclure quant à la configuration de la fréquence entre l'émetteur et le récepteur.

3 GENERALISATION

3.1 Sachant qu'on peut utiliser les fréquences entre 867.1MHz et 868.5MHz par palier de 200kHz, indiquer combien de canaux on peut utiliser.

3.2 Sachant qu'on peut utiliser les spreading factor SF7 à SF12, sur tous les canaux de fréquence, indiquer combien d'émetteurs/récepteurs peuvent communiquer simultanément sans se brouiller les uns les autres.

3.3 Sachant que sur le réseau LoRaWan les émetteurs ne doivent pas émettre plus de 1% du temps, calculer le nombre théorique d'émetteurs que peut supporter le réseau sur une même zone géographique si on occupe tous les canaux et tous les SF.

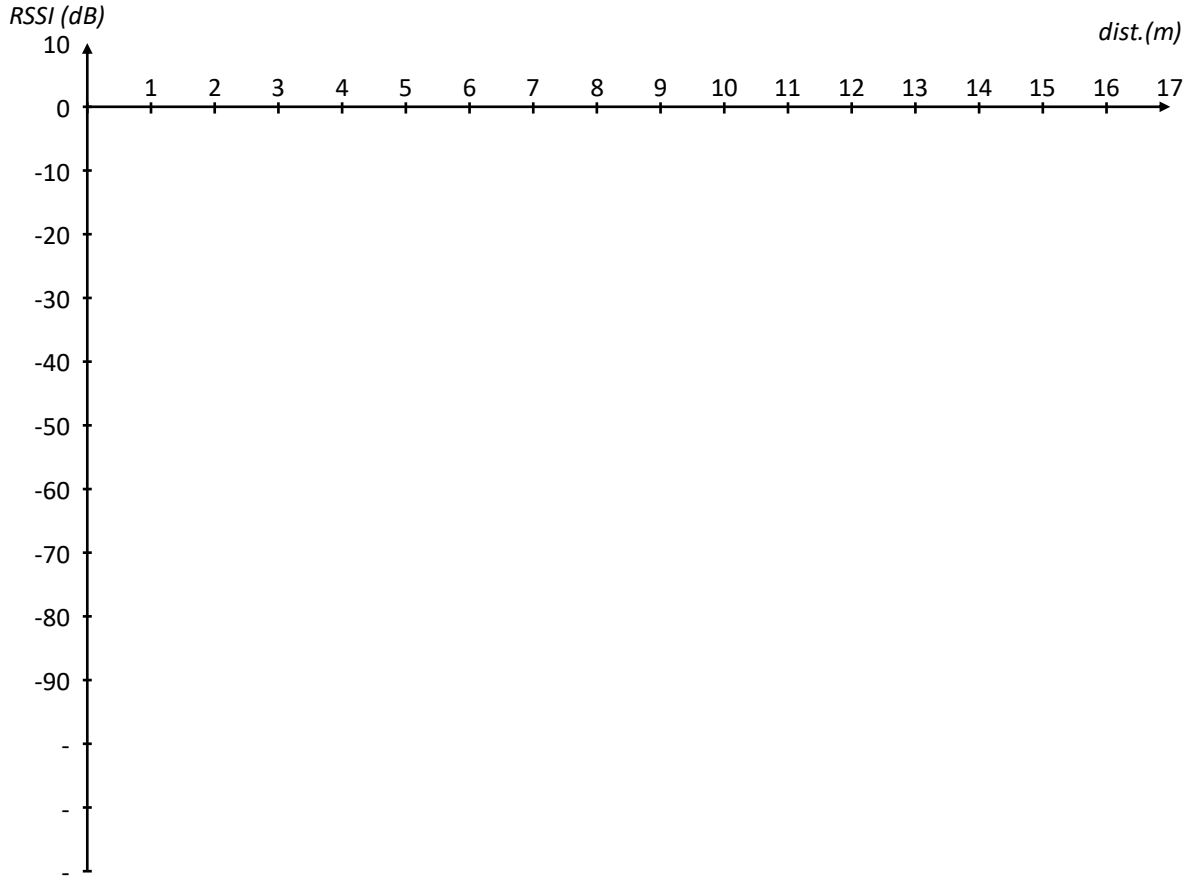
4 RAPPORT ENTRE RSSI ET DISTANCE

Le RSSI (**Received Signal Strength** Indication) ou indication du niveau de réception s'exprime en dBm. Quand la valeur en dBm est positive, on parle d'amplification, quand elle est négative, on parle d'atténuation.
L'idéal pour cette expérience est d'avoir un power-pack afin d'alimenter de manière autonome le récepteur.

4.1 Relever le niveau RSSI lorsque les antennes sont :

- L'une contre l'autre
- Environ 10cm
- Environ 1m, 2m, 3m, …
- L'endroit le plus éloigné de la salle de classe.

Tracer la courbe montrant l'évolution du RSSI en fonction de la distance.



5 EXPLOITATION

5.1 Plus on s'éloigne, plus le signal est :

- ❑ Amplifié
- ❑ Atténué

0dBm, correspond à une puissance de 1mW. Chaque fois qu'on perd 10dBm, la puissance reçue est divisée par 10.

5.2 Calculer la puissance reçue si le récepteur indique RSSI = -60dBm puis -90dBm.

Source :
Formation IOT (François Riotte)
Université savoie Mont-blanc (Sylvain Montagny)