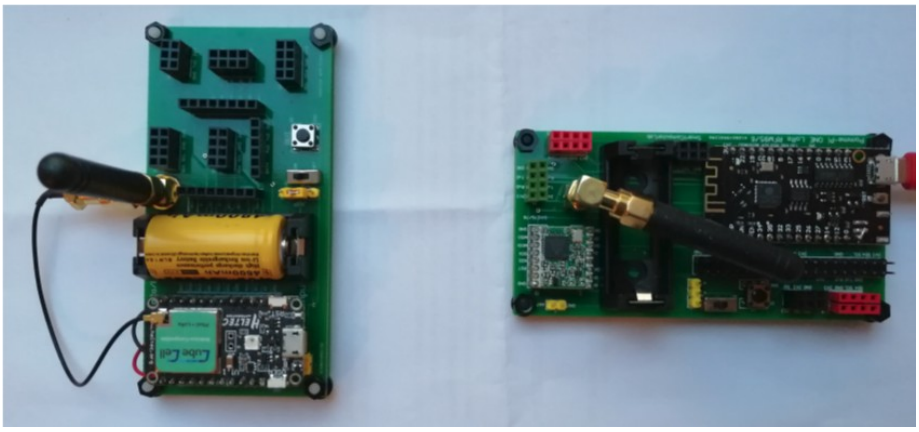
Exercise 1

Sending and receiving LoRa packets

In this exercise we are going to use two different IoT DevKits

- Pomme-Pi ONE LoRa DevKit and
- CubleCell: LoRa/LoRaWAN Devkit

as shown on the following figure:



Both boards are programmed with **Arduino IDE** or **PlatformIO**.

Attention:
This exercise is the **starting point** for our 4 hours lab including the development of complete IoT architecture with low power terminal nodes and IoT gateway with Lora to WiFi relay. The gateway sends the data to MQTT and TS IoT servers.

1.1 Receiver side on Lolin D32

The following is the code for receiver on Pomme-Pi ONE LoRa board. Note the use of **`union`** construct to send/receive LoRa packets in a simple but formatted way.

```
union pack
{
 uint8_t frame[16]; // trames avec octets
  float  data[4];   // 4 valeurs en virgule flottante
} rdp ;  // paquet d'émission
```

The same **`union`** should be used at the sender side.

```
#include <SPI.h>
#include <LoRa.h>
#define SCK     18   // GPIO18 -- SX127x's SCK
#define MISO    19   // GPIO19 -- SX127x's MISO
#define MOSI    23   // GPIO23 -- SX127x's MOSI
#define SS       5   // GPIO05 -- SX127x's CS
#define RST     15   // GPIO15 -- SX127x's RESET
#define DI0     26   // GPIO26 -- SX127x's IRQ(Interrupt Request)
#define freq    8685E5
#define sf 9
#define sb 125E3

union pack
{
 uint8_t frame[16]; // trames avec octets
  float  data[4];   // 4 valeurs en virgule flottante
} rdp ;  // paquet d'émission
```

```
void setup() {
  Serial.begin(9600);
  delay(1000);
  SPI.begin(SCK,MISO,MOSI,SS);
  LoRa.setPins(SS,RST,DI0);
  Serial.println();delay(100);Serial.println();
  if (!LoRa.begin(freq)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }

Serial.println("Starting LoRa OK!");
delay(1000);
LoRa.setSpreadingFactor(sf);
LoRa.setSignalBandwidth(sb);
LoRa.setCodingRate4(5);
}

float d1=0.0, d2=0.0 ;
int rssi;

void loop()
{
int packetLen;
packetLen=LoRa.parsePacket();
if(packetLen==16)
  {
  int i=0;
  while (LoRa.available()) {
    rdp.frame[i]=LoRa.read();i++;
    }
  d1=rdp.data[0];d2=rdp.data[1];
  rssi=LoRa.packetRssi();  // force du signal en réception en dB
  Serial.printf("Received packet:%2.2f,%2.2f\n",d1,d2);
  Serial.printf("RSSI=%d\n",rssi);
  }
}
```

1.2 Sender side on CubeCell board

```
#include "LoRaWan_APP.h"
#include "Arduino.h"
#ifndef LoraWan_RGB
#define LoraWan_RGB 0
#endif
#define RF_FREQUENCY                                868500000 // Hz
#define TX_OUTPUT_POWER                             14        // dBm
#define LORA_BANDWIDTH                              0         // [0: 125 kHz,
                                                              //  1: 250 kHz,
                                                              //  2: 500 kHz,
                                                              //  3: Reserved]
#define LORA_SPREADING_FACTOR                       9         // [SF7..SF12]
#define LORA_CODINGRATE                             1         // [1: 4/5,
                                                              //  2: 4/6,
                                                              //  3: 4/7,
                                                              //  4: 4/8]
#define LORA_PREAMBLE_LENGTH                        8         // Same for Tx and Rx
#define LORA_SYMBOL_TIMEOUT                         0         // Symbols
#define LORA_FIX_LENGTH_PAYLOAD_ON                  false
#define LORA_IQ_INVERSION_ON                        false
#define RX_TIMEOUT_VALUE                            1000
#define BUFFER_SIZE                                 128 // Define the payload size here

char txPacket[BUFFER_SIZE];
static RadioEvents_t RadioEvents;
void OnTxDone( void );
void OnTxTimeout( void );

typedef enum
{
    LOWPOWER,  ReadVoltage,  TX   // 3 states (1,2,3)
} States_t;

States_t state;
bool sleepMode = false;
int16_t rssi,rxSize;
uint16_t voltage;

union pack
{
 uint8_t frame[16]; // trames avec octets
  float  data[4];   // 4 valeurs en virgule flottante
} sdp ;  // paquet d'émission

void setup()
{
    Serial.begin(9600);
    voltage = 0;
    rssi=0;
    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 );
    state=ReadVoltage;
}

void loop()
{
  switch(state)
  {
    case TX:
    {
      memset(txPacket,0x00,BUFFER_SIZE);
      sprintf(txPacket,"%s","ADC_battery (mV): ");
      int plen= strlen(txPacket);
      sprintf(txPacket+plen,"%d",voltage);
      sdp.data[0] = (float)voltage;
      if(voltage<(uint16_t)3680)turnOnRGB(COLOR_SEND,0);
      else turnOnRGB(COLOR_RECEIVED,200);
      Serial.printf("\r\nsending packet \"%s\"\r\n",txPacket);
```

```
     // Radio.Send( (uint8_t *)txPacket, strlen(txPacket) );
      Radio.Send(sdp.frame,16);
      Serial.println(strlen(txPacket));delay(100);
      state=LOWPOWER;
      break;
    }
    case LOWPOWER:
    {
      lowPowerHandler();delay(100);
      turnOffRGB();
      delay(2000);  //LowPower time
      state = ReadVoltage;
      break;
    }
    case ReadVoltage:
    {
      pinMode(VBAT_ADC_CTL,OUTPUT);
      digitalWrite(VBAT_ADC_CTL,LOW);
      voltage=analogRead(ADC)+550; //*2;
      pinMode(VBAT_ADC_CTL, INPUT);
      state = TX;
      break;
    }
     default:
          break;
  }
  Radio.IrqProcess();
}

void OnTxDone( void )
{
  Serial.print("TX done!");
  turnOnRGB(0,0);
}

void OnTxTimeout( void )
{
    Radio.Sleep( );
    Serial.print("TX Timeout......");
    state=ReadVoltage;
    Serial.print(state);
}
```

To do:

Analyze both codes:

- LoRa modem initialization
- LoRa radio parameters
- Lora packets structure and content

Modify the Lora radio parameters

Add a sensor to send second value (the first is battery voltage)

Then you can start the main lab – Building complete IoT architecture with LoRa and WiFi links

Documents to be used:

IoT.Labs.1.and2.Low.Power.IoT.Architectures.2023

and

IoT.Labs.ESP32.D32.arduino.2023