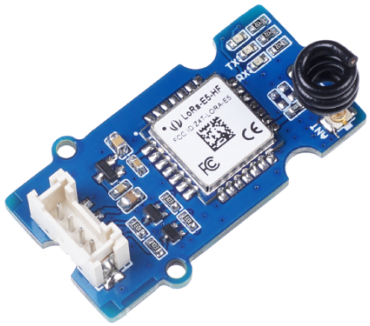
Prise en main du module Grove LoRa E5 avec le réseau TTN (The Things Network)



Grove LoRa-E5 embedded with LoRa-E5 STM32WLE5JC, powered by ARM Cortex M4 ultra-low-power MCU core and LoRa SX126x, is a wireless radio module supporting LoRa and LoRaWAN protocol on the EU868 & US915 frequency and (G)FSK, BPSK, (G)MSK, LoRa modulations. Grove - LoRa-E5 can endow your development boards' strong features of ultra-long transmitting range by easily plug and play with Grove connector on board.

By connecting Grove - LoRa-E5 to your development boards, your devices are able to communicate with and control LoRa-E5 conveniently by AT command through UART connection. Grove LoRa-E5 will be a superior choice for IoT device development, testing, and long-distance, ultra-low power consumption IoT scenarios like smart agriculture, smart office, and smart industry. It is designed with industrial standards with a wide working temperature at -40℃ ~ 85℃, high sensitivity between -116.5 dBm and -136 dBm, and power output between 10 dBm and 22 dBm.

Features¶

- LoRa-E5 (STM32WLE5JC) embedded
- Support LoRaWAN protocol on EU868/US915 frequency band
- Ultra-long transmitting range up to 10km (Ideal value in open space)
- Easy control by AT command via UART connection
- Rapid prototyping with plug-and-play Grove interfaces
- Ultra-low power consumption and high performance

Etape 1 : créer un compte sur The Things Network (TTN) et ajouter un « end device »

Créer un compte sur TTN : https://account.thethingsnetwork.org
Sur la console, sélectionnez « Go to application »
Créer une nouvelle application → «+Add application »
Donnez lui un nom et cliquez sur create application.
Allez sur « End devices », créez un nouveau device en allant sur « +Add end device »
Pour la sélection du end device, dans le champ Brand, sélectionnez Seeed Technology, puis le modèle « Seeed Studio lorawan Dev kit » et enfin le Profil « EU_863_870 »
Pour la 2ème partie « Enter registration data » :
Frequency plan : Europe 863-870 MHz (SF9 for RX2 – recommended)
AppEUI : Fill with zeros pour l'instant
DevEUI : Generate
AppKey : Generate
End device ID : ce que vous voulez !
Et cliquez du Register end device

Etape 2 : commandes AT depuis un PC

Dans cette première partie, vous allez commander le module avec des commandes AT depuis un PC à travers un convertisseur série/USB.

Pour cela vous allez utiliser la carte Arduino Uno comme convertisseur USB/Série en y mettant le code suivant :

```
#include<SoftwareSerial.h>

SoftwareSerial e5(6, 7); // (RX, TX)

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  e5.begin(9600);
}

void loop() {
     while (Serial.available() > 0) {
       e5.write(Serial.read());
    }

    while (e5.available() > 0) {
       Serial.write(e5.read());
    }
}
```

Connectez-le module Grove LoRa E5 avec la carte Arduino en reliant les broches suivantes :

Grove loRa E5	Arduino Uno
TX	D6(Rx)
RX	D7(Tx)
+5V	Vcc
GND	GND

Si vous utilisez le Grove Base Shield, il suffit de connecter le module Grove LoRa E5 sur le connecteur D6.

Communiquer avec le module :
Maintenant à l'aide du Moniteur série (Outils -> Moniteur série), vous pouvez commencer à communiquer avec le module LoRa E5.

Format de la liaison série : 8 bit, pas de parité, 1 bit de stop et 9600 bauds

Pour tester que la communication fonctionne bien, commencez par envoyer la commande **AT** sur laquelle vous devriez recevoir la réponse **OK**.

Ensuite vous pouvez tester les commandes AT suivantes :

```
AT                  Renvoie invariablement la chaîne "OK"
AT+ID               Renvoie les identifiants du module (DevAddr, AppEUI, DevEUI)

AT+DR=EU868              # set the zone, can be U915...
+DR: EU868
```

Pour changer les identifiants du module :

```
AT+ID=DevEUI,0B68BADE43C2FADE      # set the DevEUI
```

```
AT+ID=AppEUI,81263C32387B3144      # set the AppEUI
AT+KEY=APPKEY,216BBEF3ADBBE2DXXXXXXXXXXXXXXXX
```

Pour sélectionner le mode OTAA afin de rejoindre le réseau

```
AT+MODE=LWOTAA                # set OTAA join mode

AT+DR=DR0                     # change speed for SF7 BW125
AT+DR=DR1                     # change speed for SF8 BW125
AT+DR=DR2                     # change speed for SF9 BW125
AT+DR=DR3                     # change speed for SF10 BW125
AT+DR=DR4                     # change speed for SF11 BW125
AT+DR=DR5                     # change speed for SF12 BW125
```

Pour rejoinder le reseau

```
AT+JOIN
+JOIN: Start
+JOIN: NORMAL
+JOIN: Network joined
+JOIN: NetID 326548 DevAddr 48:00:00:42
+JOIN: Done

# in case of problem during join you get
+JOIN: Start
+JOIN: NORMAL
+JOIN: Join failed
+JOIN: Done

Pour envoyer un message sur le réseau :
AT+MSGHEX=01020304        # send payload 0x01,0x02,0x03,0x04 / 4B
+MSGHEX: Start
+MSGHEX: Done

or to send plain text

AT+MSG=01020304           # send payload "01020304" / 8 chars
+MSG: Start
+MSG: Done
```

La liste complète des commandes AT est disponible à partir de ce lien :
https://files.seeedstudio.com/products/317990687/res/LoRa-E5%20AT%20Command%20Specification_V1.0%20.pdf

Tester ces différentes commandes et mettez à jour les informations sur l'ID du end Device (DevEUI, AppEUI et AppKEY) dans la console de TTN. Il faut que les 3 valeurs (DEVEUI, APPEUI et APPKEY) soient identique sur le module et dans la console de TTN.

Testez l'envoi de plusieurs messages et vérifiez que ceux-ci arrivent bien sur la console de votre end device dans le champ Live Data.

Etape 3 : envoi de message depuis la carte Arduino

Maintenant vous allez envoyer des données depuis la carte Arduino en utilisant le code suivant : (mettre à jour APPKEY, DEVEUI et APPEUI avec les valeurs générées sur la console de TTN)

```
#include<SoftwareSerial.h>

SoftwareSerial e5(6, 7); // (RX, TX)
static char recv_buf[512];
static bool is_exist = false;
static bool is_join = false;
static int led = 0;
int ret=0;
short tmp =20;
short hum=50;

static int at_send_check_response(char *p_ack, int timeout_ms, char *p_cmd, ...)
{
    int ch;
    int num = 0;
    int index = 0;
    int startMillis = 0;
    memset(recv_buf, 0, sizeof(recv_buf));
    e5.write(p_cmd);
    Serial.write(p_cmd);
    delay(200);
    startMillis = millis();
    do
    {
        while (e5.available() > 0)
        {
            ch = e5.read();
            recv_buf[index++] = ch;
            Serial.write(ch);
            delay(2);
        }
    } while (millis() - startMillis < timeout_ms);

    if (strstr(recv_buf, p_ack) != NULL)
        {
            return 1;
        }
    else return 0;
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  e5.begin(9600);

  Serial.print("E5 LORAWAN TEST\r\n");

  if(at_send_check_response("+AT: OK", 100, "AT\r\n"))
  {
       is_exist = true;
       at_send_check_response("+ID: AppEui", 1000, "AT+ID\r\n");
       at_send_check_response("+MODE: LWOTAA", 1000, "AT+MODE=LWOTAA\r\n");
       at_send_check_response("+DR: EU868", 1000, "AT+DR=EU868\r\n");
       at_send_check_response("+CH: NUM", 1000, "AT+CH=NUM,0-2\r\n");
       at_send_check_response("+KEY: APPKEY", 1000,
"AT+KEY=APPKEY,\"35DE73F781531184CEB7EECE0DA9FB0F\"\r\n");
        at_send_check_response("+KEY: DEVEUI", 1000, "AT+KEY=DEVEUI,\"2CF7F12032302911\"\r\n");
       at_send_check_response("+KEY: APPEUI", 1000, "AT+KEY=APPEUI,\"8000000000000016\"\r\n");
        at_send_check_response("+CLASS: C", 1000, "AT+CLASS=A\r\n");
        ret=at_send_check_response("+PORT: 8", 1000, "AT+PORT=8\r\n");
        delay(200);
        is_join = true;
   }
   else
    {
        is_exist = false;
        Serial.print("No E5 module found.\r\n");
```

```
    }
}

void loop() {

  if (is_exist)
    {
        int ret = 0;
        if (is_join)
        {

            ret = at_send_check_response("+JOIN: Network joined", 12000, "AT+JOIN\r\n");
            if (ret)
            {
                is_join = false;
                Serial.println();
                Serial.print("Network JOIN !\r\n\r\n");
            }
            else
            {
                at_send_check_response("+ID: AppEui", 1000, "AT+ID\r\n");
                Serial.println();
                Serial.print("JOIN failed!\r\n\r\n");
                delay(5000);
            }
        }
        else
        {
            char cmd[128];
            sprintf(cmd, "AT+CMSGHEX=%04X%04X\r\n", tmp++, hum++);
            at_send_check_response("ACK Received", 5000, cmd);
            delay(20000);
        }
    }
    else
    {
        delay(1000);
    }
}
```