



<b>TP 2</b>	
<b>Achitecture et plateformes IoT</b>	<b>Année universitaire : 2025-2026</b>
<b>Enseignant : Chiheb Ameur ABID</b>	

Ce laboratoire vise à configurer une mote LoRa basée sur une Arduino Uno équipée d'un Dragino LoRa Shield (SX1276) pour envoyer la chaîne de caractères "Hello!" toutes les 60 secondes à une passerelle (ex. : Dragino LG01-N) connectée à The Things Network (TTN). La mote opère en mode ABP (Activation By Personalization) et en Classe A LoRaWAN (uplinks suivis de deux fenêtres de réception RX1/RX2). Le sketch utilise la bibliothèque MCCI LoRaWAN LMIC pour gérer le protocole LoRaWAN. Après chaque transmission, les trames reçues (downlinks) sont affichées dans le moniteur série de l'IDE Arduino.

### 1) Créer un nouveau projet Arduino

Ouvrez l'IDE Arduino.

- Allez à **File > New** pour créer un nouveau sketch.
- Enregistrez le projet (ex. : **lora\_abp\_hello.ino**) dans un dossier de votre choix (ex. : **C:\Users\).**

### 2) Configurer la carte

- Connectez l'Arduino Uno avec le Dragino LoRa Shield via USB.
- Dans l'IDE Arduino, sélectionnez **Tools > Board > Arduino Uno**.
- Sélectionnez le port USB : **Tools > Port** (ex. : **COM3** sous Windows).

### 3) Installer la bibliothèque MCCI LoRaWAN LMIC library

La bibliothèque MCCI LoRaWAN LMIC implémente le protocole LoRaWAN pour l'Arduino Uno avec le Dragino LoRa Shield (SX1276). Elle supporte la région EU868 et le module radio SX1276.

Installation via l'IDE Arduino :

- Allez à **Sketch > Include Library > Manage Libraries**.
- Recherchez « MCCI LoRaWAN LMIC library » et installez la version 5.0.1 (ou la plus récente).

Vérification :

Sous Windows, le dossier de la bibliothèque est installé dans

**C:\Users\.**

Confirmez la présence de **src/lmic.h** et **project\_config/lmic\_project\_config.h**.

### 4) Configurer la bibliothèque MCCI LoRaWAN LMIC library

Trouvez **lmic\_project\_config.h** dans

**libraries\MCCI\_LoRaWAN\_LMIC\_library\project\_config/**. Le répertoire **libraries/** se trouve normalement au même niveau du répertoire du projet.

Avec un éditeur du texte, éditez le fichier **lmic\_project\_config.h**, et s'assurer que :

- La région choisie est **eu868**
- Le module radio activé est **sx1276\_radio**

Le fichier **lmic\_project\_config.h** doit avoir un contenu similaire à celui-ci :

```
// project-specific definitions
#define CFG_eu868 1
// #define CFG_us915 1
// #define CFG_au915 1
// #define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /* for as923-JP; also define CFG_as923 */
// #define CFG_kr920 1
// #define CFG_in866 1
#define CFG_sx1276_radio 1
// #define CFG_sx1261_radio 1
// #define CFG_sx1262_radio 1
// #define ARDUINO_heltec_wifi_lora_32_V3
// #define LMIC_USE_INTERRUPTS
```

### 5) Configurer l'end-device dans TTN

Connectez-vous à la console TTN.

Créez une application (**Applications > Add Application**).

Ajoutez un end-device en mode ABP (**End Devices > Add End Device > Manually > ABP**).

Notez les clés générées :

- Device Address (DEVADDR) : Ex. **0x260B2C4F**.
- Network Session Key (NWKSKEY) : Ex. **31 BA 24 6B E9 F3 66 D1 1F 04 53 8F 2C 1B E8 B4**.
- App Session Key (APPSKEY) : Ex. **D0 B5 1F 09 F9 48 9C 8A CD F4 B7 87 0F CF D8 63**.

### 6) Sketch d'envoi

Compléter le sketch suivant pour pouvoir envoyer à chaque minute la chaîne de caractère "Hello!". En plus, on vérifie à l'envoi la réception d'une trame LoRa qu'on affichera.

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

static const u4_t DEVADDR = ...;
```



```
static const PROGMEM u1_t NWKSKEY[16] = { ... };
static const u1_t PROGMEM APPSKEY[16] = { .... };

void os_getArtEui(u1_t* buf) {}
void os_getDevEui(u1_t* buf) {}
void os_getDevKey(u1_t* buf) {}

static uint8_t mydata[] = "Hello!";
static osjob_t sendjob;

const unsigned TX_INTERVAL = 60;

const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 9,
  .dio = { 2, 6, 7 },
};

void onEvent(ev_t ev) {
  Serial.print(os_getTime());
  Serial.print(F(": "));
  if (ev == EV_TXCOMPLETE) {
    // Vérifier la réception d'une trame LoRa
    ...
    // Planifier le prochain envoi
    ...
  }
}

void do_send(osjob_t* j) {
  // Vérifier qu'il n'y a pas un envoi en cours
  ...
  // Envoyer le payload
  ...
  // Afficher la payload envoyé
}

void setup() {
  Serial.begin(115200);
  Serial.println(F("Starting"));
}
```



```
os_init();
LMIC_reset();

uint8_t appskey[sizeof(APPSKEY)];
uint8_t nwkskey[sizeof(NWKSKEY)];
memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
LMIC_setSession(0x1, DEVADDR, nwkskey, appskey);
LMIC_setupChannel(0, 868100000, DR_SF7, 1);
LMIC_setLinkCheckMode(0);
LMIC.dn2Dr = DR_SF9;
LMIC_setDrTxpow(DR_SF7, 14);

...
}

void loop() {
  os_runloop_once();
}
```

#### 7) Vérification de l'envoi

Accédez à l'interface web de l'administration de la passerelle avec l'adresse **192.162.168.100:8000** dans votre navigateur et vérifiez dans **LogRead > LoRa Log** pour vérifier le transfert des messages

Vérifiez depuis la console TTN la réception des messages envoyés par le end-device : **Applications > End Devices > [end-device] > Live data**.

Le payload est affiché en hexadécimal. Ajoutez un décodeur dans **Payload Formatters > Uplink**,

```
function Decoder(bytes, port) {
  if (port === 1) {
    return { message: String.fromCharCode.apply(null, bytes) };
  }
  return {};
}
```

Puis revenez dans **Live data** pour voir **"Hello!"** décodé.

À l'aide de la console TTN, envoyez un message, qu'on doit l'afficher sur le moniteur série de la mote LoRa.