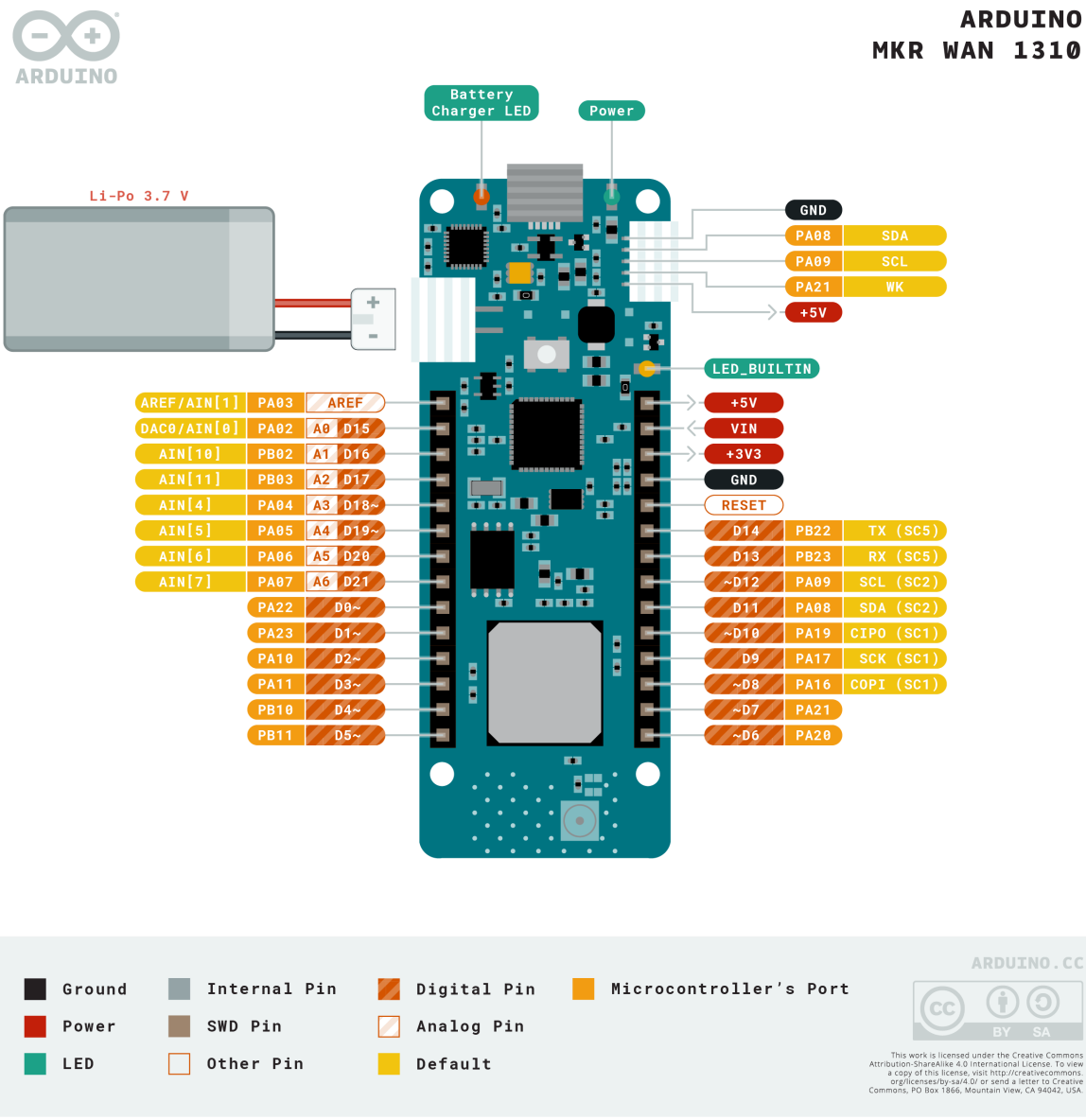
Prise en main de la carte MKRWAN1310



Technical specifications :

Board	**Name**	Arduino® MKR WAN 1310
	SKU	ABX00029
	Compatibility	MKR
Microcontroller	SAMD21 Cortex®-M0+ 32bit low power ARM MCU	

USB connector	Micro USB (USB-B)	
Pins	**Built-in LED Pin**	6
	Digital I/O Pins	8
	Analog Input Pins	7 (ADC 8/10/12 bit)
	Analog Output Pins	1 (DAC 10 bit)
	PMW Pins	13 (0 - 8, 10, 12, A3, A4)
	External interrupts	10 (0, 1, 4, 5, 6, 7, 8 ,9, A1, A2)
Connectivity	**LoRa**	Murata CMWX1ZZABZ
	Carrier frequency	433/868/915 MHz
	Region	EU/US
	Secure element	ATECC508A
Communication	**UART**	Yes
	I2C	Yes
	SPI	Yes
Power	**I/O Voltage**	3.3V
	Input Voltage (nominal)	5-7V
	DC Current per I/O pin	7 mA
	Supported battery	Li-Po Single Cell, 3.7V, 1024mAh Minimum
	Battery connector	JST PH

	Lowest power consumption	104uA
Clock speed	**Processor**	48 MHz
	RTC	32.768 kHz
Memory	**SAMD21G18A**	256KB Flash, 32KB SRAM
	Murata CMWX1ZZABZ	192KB Flash, 20KB RAM

Pour installer l'environnement **Arduino** et les dépendances pour les cartes **MKR** qui sont muni de microcontrôleur **SAMD21** (initialement développé par ATMEL, racheté par Microchip en 2016), vous pouvez suivre le guide qui se trouve sur le lien suivant : https://docs.arduino.cc/software/ide-v1/tutorials/getting-started/cores/arduino-samd

Ou visiter le lien suivant : https://www.arduino.cc/en/Guide/MKRWAN1310

Première partie : Envoyer des données sur le réseau The Things Network (TTN)

Pour envoyer des données sur le réseau TTN (Ther Things Network), vous allez suivre un tutoriel fournit par Arduino. Mais avant cela, lisez bien les quelques consignes ci-dessous.

Connecting MKR WAN 1310 to The Things Network (TTN) :
https://docs.arduino.cc/tutorials/mkr-wan-1310/the-things-network

Pour ce TP, vous vous connecterez au compte suivant afin d'envoyer des informations via le réseau LoRaWAN appelé The Things Network : https://console.cloud.thethings.network

Login : TTC2023
Password : Lorawan23#

Lorsque vous devrez créer une **application** dans TTN pour votre device, il faudra choisir un nom tel que **mkrwan-xxx** (avec xxx correspondant à un numéro entre 0 et 999). Il faudra ensuite coller une étiquette sur la carte MKRWAN en inscrivant le numéro choisi.

Pour le nom de **device**, il faudra choisir un nom tel que **eui-mkrwan-xxx** (avec xxx correspondant au même numéro que précédemment utilisé).

Attention : lors de l'enregistrement de la carte, le **JoinEUI** correspond au **AppEUI**. Le code renseigné par l'exécutable **Firstconfiguration**, vous donne le **DevEUI**.

Deuxième partie : Envoyer les données de température et humidité sur Ubidots

Ensuite, vous pouvez tester l'exemple **Test_MKRWAN.ino** ci-dessous pour envoyer des données sur TTN. Il faudra modifier cet exemple avec vos codes **appEUI** et **appKEY**.

```
#include <MKRWAN.h>
LoRaModem modem;
String appEui = "Votre_code_appEUI";
String appKey = "Votre_code_appKEY";
bool connected;
int err_count;
short con;
void setup() {
   Serial.begin(115200);
  while (!Serial);
  Serial.println("Welcome to MKR WAN 1300/1310 ");
   modem.begin(EU868);
   delay(1000);      // apparently the murata dislike if this tempo is removed...
   connected=false;
   err_count=0;
   con =0;
}
void loop() {
   char msg[2] = {3,4};
   short test = 27;
   if ( !connected ) {
    Serial.print("Join test : ");
    Serial.println(++con);
    int ret=modem.joinOTAA(appEui, appKey);
    if (ret ) {
      connected=true;
      modem.minPollInterval(60);
      Serial.println("Connected");
      modem.dataRate(5);   // switch to SF7
      delay(100);          // because ... more stable
      err_count=0;
    }
  }
  if ( connected ) {
    int err=0;
    modem.beginPacket();
    modem.write(msg,2);
    modem.write(test);
    modem.print(test);
    err = modem.endPacket();
    if ( err <= 0 ) {
      Serial.print("Error : ");
      Serial.println(err);
```

```
      // Confirmation not received – jam or coverage fault
      err_count++;
      if ( err_count > 50 ) {
        connected = false;
      }
      // wait for 2min for duty cycle with SF12 – 1.5s frame
      for ( int i = 0 ; i < 120 ; i++ ) {
        delay(1000);
      }
    } else {
      err_count = 0;
      // wait for 20s for duty cycle with SF7 – 55ms frame
      delay(20000);
      Serial.println("Message envoyé");
    }
  }
}
```

A partir de cet exemple, analyser le code afin de comprendre la différence entre la méthode modem.write() et modem.print() ;

Remarque : la librairie MKRWAN propose plusieurs méthodes qui permettent de configurer le modules LoRa de MURATA. Pour avoir plus d'informations sur cette librairie, vous pouvez consulter les 2 liens suivants :
Github : https://github.com/arduino-libraries/MKRWAN
Arduino : https://www.arduino.cc/reference/en/libraries/mkrwan/

Maintenant, il faut modifier le code afin d'envoyer la température et l'humidité d'un capteur DHT11 sur TTN.

Vous trouverez plein d'exemple sur internet qui expliquent comment connecter et utiliser le capteur DHT11 avec Arduino. Par exemple :
https://create.arduino.cc/projecthub/techno_z/dht11-temperature-humidity-sensor-98b03b

Enfin il faut envoyer les données de température et humidité vers la plateforme Ubidots STEM. Pour cela il faut créer un compte sur Ubidots STEM en utilisant des identifiants que vous pouvez créer aisément avec une adresse Yopmail tel que **votre_nom@yopmail.com** et le mot de passe **Lorawan23#)** et suivre le tutoriel ci-dessous :
https://help.ubidots.com/en/articles/5096476-plugins-connect-the-things-stack-to-ubidots

Pour décoder le message binaire, vous pouvez utiliser le « **Paylod Formater** » dans la console de TTN ou alors utiliser la fonction **decodeUplink(bytes)** qui se trouve dans « decoder » sur le plugin de Ubidots.

Voici 2 liens vers quelques explications sur le Payload Formater de TTN :
https://www.thethingsindustries.com/docs/integrations/payload-formatters/create/
https://www.thethingsindustries.com/docs/integrations/payload-formatters/javascript/uplink/

Voici un exemple simple de Payload Formater pour TTN en JavaScript lorsque la température et l'humidité sont envoyés sur un octet (char).

```
function decodeUplink(input) {
  var data = {};
  data.temperature = input.bytes[0];
  data.humidity = input.bytes[1];
  return {
    data: data,
    warnings: [],
    errors: []
  };
}
```

Voici un autre exemple lorsque la température et l'humidité sont envoyés sur 2 octets (short) et multiplié par 100 au préalable afin d'obtenir un nombre flottant avec 2 chiffres après la virgule.

```
function decodeUplink(input) {
  var data = {};
  data.temperature = (input.bytes[1] << 8 | (input.bytes[0])) / 100;
  data.humidity = (input.bytes[3] << 8 | (input.bytes[2])) / 100;
  return {
    data: data,
    warnings: [],
    errors: []
  };
}
```

Pour le décodeur du plugin de Ubidots, si vous n'utilisez pas le Payload Formater de TTN, il faudra faire le décodage dans la fonction `decodeUplink(bytes)` comme sur l'exemple ci-dessous :

```
function decodeUplink(bytes) {
  var decoded = {};
      decoded.temperature = (bytes[1] << 8 | (bytes[0])) / 100;
      decoded.humidity = (bytes[3] << 8 | (bytes[2])) / 100;
  return {"data": decoded};
}
```

Si vous utilisez le Payload Formater sur TTN, il faudra dans ce cas-là décommenter la ligne suivante :

```
var decoded_payload = args['uplink_message']['decoded_payload'];
```

Et mettre en commentaire les 2 lignes qui font appel à la fonction de décodage du plugin :

```
//let bytes =  Buffer.from(args['uplink_message']['frm_payload'], 'base64');
//var decoded_payload = decodeUplink(bytes)['data'];
```

Troisième partie : Utiliser le downlink pour changer un paramètre

Tester l'exemple **LoraSendAndReceive** qui se trouve dans la librairie MKRWAN.
A partir de cet exemple, réaliser un programme qui permet d'envoyer la température et l'humidité avec un capteur DHT11 mais qui peut également recevoir un paramètre qui permet de modifier le temps entre 2 envois.
Par exemple, si le downlink vaut 20, le temps d'attente entre 2 envois sera de 20 secondes.