TP n°1 : Programmation d'une carte arduino MKR-1010 sur kit Opla IoT

L'objectif de ce TP va consister à programmer une carte Arduino MKR WiFi 1010 du kit Opla IoT et à la faire communiquer vers l'extérieure.

1. – Installation logiciel et prise en main du matériel :

Vous allez utiliser un kit Opla IoT avec carte MKR WiFi 1010 avec un boitier multifonction (Carrier). La carte et le boitier sont déjà assemblés. Vous trouverez les détails de l'utilisation et le montage du kit sur : https://opla.arduino.cc/opla/module/carrier/lesson/discovering-the-mkr-iot-carrier

En vous aidant d'Internet :

Quel est le processeur principal de cette carte ? : _____

Quel est le module Wifi de cette carte ? : _____

Quel système permet d'assurer la communication autour de cette carte ? : _____

Quels sont les capteurs disponibles sur ce kit ? :

_____

_____

Avant de la connecter, vous aller installer les logiciels et librairies nécessaires.
Connectez vous avec un compte d'administrateur (consignes au tableau).
Allez sur https://www.arduino.cc/ , menu Software et téléchargez « Arduino IDE » pour Windows.
Lancez l'installation (précisez Student et Education aux questions).
Pendant l'installation, il va être proposé d'installer divers autres logiciels et drivers de périphériques (Adafruit LLC Ports, Drivers USB, …), installez les.
Quand l'installation est terminée, branchez votre kit en USB.
L'IDE doit vos proposer d'installer le paquet pour utiliser la carte MKR WiFi 1010, installez le.
Il est possible que l'IDE vous propose d'autres librairies, installez les.

Dans l'IDE, faites « Nouveau Programme » et enregistrez le sous un nom adéquat (TP…).
Un programme Arduino est toujours composé de 2 parties : une méthode « setup » et une méthode « loop »
A quoi sert la méthode « setup » ? :

_____

A quoi sert la méthode « loop » ? :

_____

Compilez ce premier programme en cliquant sur la première icône : 

La sortie de la compilation se situe sur la partie basse de l'IDE.

Lancez le programme en cliquant sur la deuxième icône: 
L'IDE doit vous proposer de sélectionner le Port Série approprié dans une boîte de dialogue, faites « OK ».

La compilation va sans doute échouer car vous n'avez pas précisé le type de carte utilisée.
Vous devez installer la librairie « Arduino_MKRIoTCarrier » ainsi que les librairies nécessaires à celle-ci : menu « Outils/Gérer les bibliothèques » et taper « mkriot + Entrée » dans la partie recherche.
Installez cette librairie et toutes ses dépendances.
Vous devez aussi sélectionner la bonne carte mère : menu « Outils/Type de carte/Arduino SAMD (32 bits ARM …) Board » et sélectionnez « Adruino MKR WiFi 1010 ».

Votre IDE est opérationnel. Relancer votre premier programme. Vous devez entendre 2 clics suivis d'un autre sur le Kit.
La partie du bas ne vous montre que le résultat de la compilation et du téléversement.
Ouvrez le moniteur du port : menu « Outils/moniteur série »
Ajoutez le code suivant à « setup » :

```
Serial.begin(9600);
while (!Serial) {   }
Serial.println("Fct setup terminée");
```

Et ce code à « loop » :

```
delay(2000);
Serial.println("Fct loop");
```

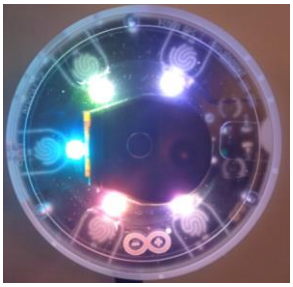
Lancez votre programme et observez votre moniteur.

2. - Affichage des Leds et utilisation du buzzer et de l'écran :

Ajoutez la librairie du kit :	`#include <Arduino_MKRIoTCarrier.h>` `MKRIoTCarrier carrier;`
Initialisez ensuite votre plateforme (setup) :	`carrier.begin();`
Allumez une première Led avec (setup) :	`carrier.leds.setPixelColor(3, 255, 0 , 255);` `carrier.leds.show();`
Et éteignez là après 2 secondes (fct delay) en lui passant la couleur RGB (0,0,0)	

Que devez-vous faire pour laisser cette led clignoter régulièrement ? : _____

Modifiez votre programme pour que les 5 Leds s'allument avec des couleurs différentes pendant 2 secondes au démarrage.



Le kit possède un buzzer que vous allez tester au démarrage (et pas plus d'1 seconde, merci).

Appelez la méthode « sound » de la propriété « Buzzer » du kit « carrier ». La valeur à passer en paramètre correspond à la fréquence du buzzer, vous pouvez utiliser 500. Ajoutez ensuite une durée de 1 seconde et appelez ensuite la méthode « nosound ».
L'écran situé au centre se gère avec l'objet « display » du « carrier ». Pour afficher un écran de bienvenue :

- appeler la méthode « fillScreen » avec le paramètre « ST77XX_RED » (fond rouge)
- appeler la méthode « setTextColor » avec le paramètre « ST77XX_WHITE » (texte blanc)
- appeler la méthode « setTextSize » avec le paramètre « 2 » (taille moyenne)
- appeler la méthode « setCursor » avec les valeurs 40 et 110 (position x et y)
- appeler la méthode « print » avec le texte de bienvenue

3. – Utilisation des autres capteurs :

Déclarez une variable « fTemp » de type « float » qui va représenter la température.
Appelez la méthode « readTemperature» de la propriété « Env » du kit « carrier ».
Affichez là avec l'unité sur l'écran.

Faites ensuite la même chose pour l'humidité relative et la pression atmosphérique.
Les 3 valeurs doivent s'afficher sur l'écran.
Attention la pression doit être en hPa et corrigée de l'altitude (+1 hPa tous les 8 m)

4. – Connexion au WiFi :

La carte MKR1010 intègre un module WiFi. Vous allez le connecter au réseau WiFi disponible (instruction au tableau).
Il faut en premier ajouter la bibliothèque « WiFiNINA ». Faites menu « Croquis/Inclure une Bibliothèque/Gérer les Bibliothèques, tapez « Wifinina » et installez la librairie associée avec ses dépendances si il y en a.
Ajoutez ensuite cette librairie : menu « Croquis/Inclure une Bibliothèque/ WiFiNINA » ce qui va rajouter la ligne : #include <WiFiNINA.h>)
Ajoutez 2 chaînes de caractères comme variables globales avec les valeurs du SSID et de la clé du réseau WiFi.

```
char ssid[] = "valeurduSSID";
char pass[] = "cléWifi";
```

Ajoutez enfin la boucle suivante qui boucle tant que la connexion au réseau WiFi n'est pas effective :

```
int status = WL_IDLE_STATUS;
while (status != WL_CONNECTED) {
  Serial.print("Tentative de connexion au réseau ");
  Serial.println(ssid);
  status = WiFi.begin(ssid, pass);
  delay(10000);
}
```

Lancez votre programme, si celui-ci fonctionne, la connexion WiFi est bien correcte. Sinon vérifiez dans le moniteur de contrôle.

Vous allez ajouter les 2 méthodes suivantes et appeler la première après la connexion WiFi.

```
void printCurrentNet() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  byte bssid[6];
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  printMacAddress(bssid);
  byte encryption = WiFi.encryptionType();
  Serial.print("Encryption Type:");
  Serial.println(encryption, HEX);
  Serial.println();
}
```

```
void printMacAddress(byte mac[]) {
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      Serial.print("0");
    }
    Serial.print(mac[i], HEX);
    if (i > 0) {
      Serial.print(":");
    }
  }
  Serial.println();
}
```

Appelez la méthode « printCurrentNet » et vérifiez que les informations s'affichent bien sur le moniteur.

Déclarez une variable globale de type « IPAddress » et affichez là dans le moniteur et sur l'écran de bienvenue en dessous du texte. Vous utiliserez la méthode « localIP ».

Maintenant, vous pouvez tester votre accés à Internet. Pour cela vous allez utiliser la méthode « Ping » de votre objet « WiFi ». Dans « setup » après la connexion Wifi, créez une chaîne de caractère valant « www.google.com » et appelez la méthode « Ping » en lui passant cette valeur. Elle retourne un entier qui est supérieur à 0 si le ping a fonctionné. La valeur retournée représente précisement la durée aller-retour du ping (round-trip time ou RTT) que vous pouvez afficher. En cas d'échec la valeur renvoyée est négative.

5. – Export des données mesurées :

Vous allez connecter votre kit au serveur « ThingSpeak ». Allez sur la plateforme https://thingspeak.com/ et connectez-vous (ou recréez un compte).
Créez un nouveau « Channels », donnez-lui un nom et définissez les 3 champs « Field1 », « Field2 » et « Field3 » comme Température, Humidité et Pression. Sauvegardez.

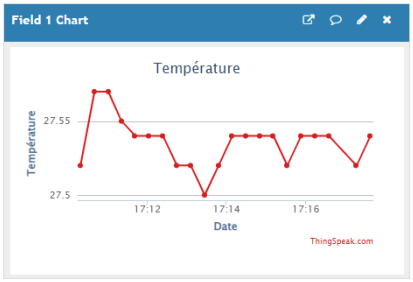
Sur votre programme, ajoutez ensuite une variable globale de type « WiFiClient » et 2 chaînes de caractères, la première contenant « api.thingspeak.com », et la deuxième votre clé API (Write API Key).

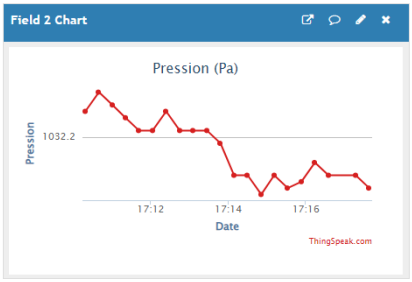
Créez une fonction « httpRequest ». Dans celle-ci :

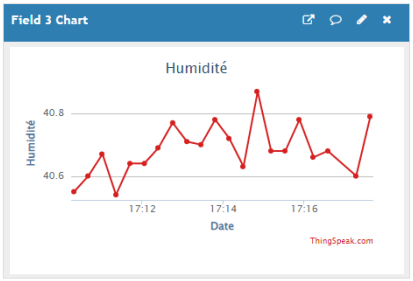
- déclarez une variable « String » qui sera la concaténation de « field1= », la valeur de la température, « &field2= », la valeur de la pression, « &field3= » et la valeur de la variable d'humidité, le tout sans espace (la fonction « String() » permet de convertir un float en String, une concaténation de String se fait avec +)
- appelez la méthode « connect » de votre objet « WiFiClient » en lui passant comme paramètres l'Url du serveur et la valeur « 80 » (port). Cette fonction renvoie vrai si la connexion est correcte

- appelez la méthode « println » de votre objet « WiFiClient » en lui passant dans l'ordre :
  "POST /update HTTP/1.1"
  "Host: api.thingspeak.com"
  "Connection: close"
  "User-Agent: ArduinoWiFi/1.1"
  "X-THINGSPEAKAPIKEY: " suivi de la valeur de votre APIKey
  "Content-Type: application/x-www-form-urlencoded"
- appelez la méthode « print » de votre objet « WiFiClient » en lui passant dans l'ordre :
  "Content-Length: "
  la longueur de votre variable « String » contenant les champs « field »
  "\n\n"
  votre variable « String » contenant les champs « field »

Appelez votre méthode « httpRequest » dans la boucle « loop » et testez le programme.
Au bout de quelques minutes, vous aurez des graphes complets :







Après vos tests, mettez la fonction « httpRequest » en commentaire pour ne pas saturer votre compte ThingSpeak.