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

```
#include <Arduino_MKRIoTCarrier.h>
MKRIoTCarrier carrier;
// Data pour connexion Wifi
#include <WiFiNINA.h>
char ssid[] = "TP-Link_F99E";
char pass[] = "05186150";
IPAddress ip;

// Données pour Thingspeak
WiFiClient clientThingSpeak;
char server[] = "api.thingspeak.com"; // "52.204.7.22";
String writeAPIKey = "3FKRFV5RSEQ5VNE1";

// Variables globales
float temperature;
float pressure;
float humidity;

// Fonction d'initialisation
void setup() {
    Serial.begin(9600);
    while (!Serial)
    {
        Serial.println("");
    }
    // Connexion au Wifi
    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(1000);
    }
    Ping();

    // Ecran d'accueil
    carrier.begin();
    delay(1000);
    carrier.Buzzer.sound(500);  // Buzzer à la fréquence 500Hz
    //delay(100);
    carrier.Buzzer.noSound();
    carrier.display.fillScreen(ST77XX_RED);         //Fond d'écran rouge
    carrier.display.setTextColor(ST77XX_WHITE);     //Texte blanc
    carrier.display.setTextSize(2);                 //Taille du texte moyenne
    carrier.display.setCursor(40, 80);              //Position du texte
```

```
    carrier.display.print("Bienvenue !");           //Texte affiché

    delay(2000);
    Serial.println("FIN FONCTION SETUP");
}

void httpRequestThingSpeaks() {
  if (clientThingSpeak.connect(server, 80))
  {
      String data = "field1=" + String(temperature) + "&field2=" + String(pressure) +
"&field3=" + String(humidity);
                //Il faut envoyer "field1=29.44&field2=1025.89&field3=66.31"
      clientThingSpeak.println("POST /update HTTP/1.1");
      clientThingSpeak.println("Host: api.thingspeak.com");
      clientThingSpeak.println("Connection: close");
      clientThingSpeak.println("User-Agent: ArduinoWiFi/1.1");
      clientThingSpeak.println("X-THINGSPEAKAPIKEY: " + writeAPIKey); // Variable définie en
haut
      clientThingSpeak.println("Content-Type: application/x-www-form-urlencoded");
      clientThingSpeak.print("Content-Length: ");
      clientThingSpeak.print(data.length());
      clientThingSpeak.print("\n\n");
      clientThingSpeak.print(data);
      Serial.print("Données envoyées : "); Serial.println(data);
  }
  else
  {
    Serial.println("Connexion TS en échec");
  }
}

void Ping()
{
    int pingResult = WiFi.ping("google.com");
    if (pingResult >= 0)
    {
      Serial.println("Ping OK");
    }
    else
    {
      Serial.print("Ping en échec : "); Serial.println(pingResult);
    }
}

// Fonction permanente qui tourne en boucle
void loop() {
  delay(2000);
  fct_Led_GO();
  delay(4000);
  fct_Led_Stop();
  readMeteo();
  Serial.println("FCT LOOP");
```

```
  httpRequestThingSpeaks();
  delay(4000);
}

void readMeteo()
{
  carrier.display.fillScreen(ST77XX_RED);
  humidity = carrier.Env.readHumidity();          //reads humidity
  carrier.display.setCursor(65, 60);
  carrier.display.print("humidite: ");
  carrier.display.setCursor(75, 80);
  carrier.display.print(humidity);
  temperature = carrier.Env.readTemperature();    //reads temperature
  carrier.display.setCursor(50, 110);
  carrier.display.print("Temperature: ");
  carrier.display.setCursor(70, 130);
  carrier.display.print(temperature);
  carrier.display.print(" C");
  pressure = carrier.Pressure.readPressure();     //reads pressure
  pressure = pressure*10; // conversion en hPa
  pressure = pressure + 270/8; // Correction Pression 1hPa tous les 8 mètres, 3iL est à 270m
  carrier.display.setCursor(60, 160);
  carrier.display.print("Pression: ");
  carrier.display.setCursor(60, 180);
  carrier.display.print(pressure);
  carrier.display.print(" hPa");
}

void fct_Led_GO() // Affichage des Leds
{
  carrier.leds.setPixelColor(0, 200, 200 , 255);
  carrier.leds.setPixelColor(1, 100, 255 , 100);
  carrier.leds.setPixelColor(2, 255, 0 , 255);
  carrier.leds.setPixelColor(3, 255, 100 , 100);
  carrier.leds.setPixelColor(4, 100, 100 , 255);
  carrier.leds.show();
}

void fct_Led_Stop() // Extinction des Leds
{
  carrier.leds.setPixelColor(0, 0, 0 , 0);
  carrier.leds.setPixelColor(1, 0, 0 , 0);
  carrier.leds.setPixelColor(2, 0, 0 , 0);
  carrier.leds.setPixelColor(3, 0, 0 , 0);
  carrier.leds.setPixelColor(4, 0, 0 , 0);
  carrier.leds.show();
}

void printCurrentNet() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  byte bssid[6];
```

```
  WiFi.BSSID(bssid);
  Serial.print("BSSID: ");
  printMacAddress(bssid);
  ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  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();
}
```