5.7 *Case study 1* et le travail à faire - Un terminal IoT avec trois tâches

L'exemple suivant est un terminal IoT avec deux ou plus capteurs communicant avec un point d'accès WiFi et le serveur **ThingSpeak.fr/.com**.
Les fonctionnalités du terminal sont décomposées en tâches de telle façon que chaque dispositif est indépendant et asynchrone par rapport aux autres.

Les tâches sont suivantes :

- tâche d'afficheur OLED
- tâche de communication WiFi – **ThingSpeak**
- tâche de fond - tâche de capteurs (à compléter/modifier selon la disponibilité de capteurs)

Voici le code :

```
#include <WiFi.h>
#include "ThingSpeak.h"
#include <Adafruit_MLX90614.h>
#include "SHT21.h"
SHT21 SHT21;
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
WiFiClient  client;
#include <U8x8lib.h>
U8X8_SSD1306_128X64_NONAME_SW_I2C u8x8(15, 4, 16);
char ssid[] = "...";         //  your network SSID (name)
char pass[] = "...";   // your network passw
uint32_t delayMS;
QueueHandle_t oledqueue,TSqueue;
int queueSize=4;
float sensors[4];

void taskTS( void * parameter)
{
  const TickType_t xDelay = 20000 / portTICK_PERIOD_MS;
  float sensors[4];
  while(true)
  {
  xQueueReceive(TSqueue,(float *)sensors,10000);
  ThingSpeak.setField(1, sensors[0]);
  ThingSpeak.setField(2, sensors[1]);
  ThingSpeak.setField(3, sensors[2]);
  ThingSpeak.setField(4, sensors[3]);
  while (WiFi.status() != WL_CONNECTED) {
      delay(500);Serial.print(".");
      }
    ThingSpeak.writeFields(4, "4M9QG56R7VGG8ONT");
    vTaskDelay(xDelay);
  }
}
```

```
void taskOLED( void * parameter)
{
  const TickType_t xDelay = 10000 / portTICK_PERIOD_MS; // portTICK_PERIOD is 1 ms
  float sensors[4];
  char dbuff[32]; int count=0;
  while(true)
  {
  xQueueReceive(oledqueue,(float *)sensors,10000);
  u8x8.clear();
  sprintf(dbuff,"ATemp:%2.2f*C",sensors[0]);
  u8x8.drawString(0,0,dbuff);
  sprintf(dbuff,"OTemp:%2.2f",sensors[1]);
  u8x8.drawString(0,1,dbuff);
  sprintf(dbuff,"Temp:%2.2f*C",sensors[2]);
  u8x8.drawString(0,2,dbuff);
  sprintf(dbuff,"Humi:%2.2f",sensors[3]);
  u8x8.drawString(0,3,dbuff);
  sprintf(dbuff,"Count:%4.4d",count); count++;
  u8x8.drawString(0,5,dbuff);
  vTaskDelay(xDelay);
  }
}

void setup() {
  char dbuff[32];
  Serial.begin(9600);
  Wire.begin(21,22);   // initialize I2C bus - SDA, SCL
  delay(100);
  mlx.begin();
  delay(100);
  SHT21.begin();
   WiFi.disconnect(true);
   delay(1000);
   WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
    }
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);
  Serial.println("WiFi setup ok");
    ThingSpeak.begin(client);
    delay(1000);
  Serial.println("ThingSpeak begin");
```

```
  u8x8.begin();  // initialize OLED
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  u8x8.clear();
  u8x8.drawString(0,0,"IP address");
  sprintf(dbuff,"%d.%d.%d.%d",ip[0],ip[1],ip[2],ip[3]);
  u8x8.drawString(0,1,dbuff);
  delay(100);
  u8x8.drawString(0,3,"Start TS");
  delay(3000);
oledqueue = xQueueCreate(queueSize,4*sizeof(float));
TSqueue = xQueueCreate(queueSize,4*sizeof(float));

    xTaskCreate(
                    taskOLED,          /* Task function. */
                    "taskOLED",        /* String with name of task. */
                    10000,            /* Stack size in words. */
                    NULL,             /* Parameter passed as input of the task */
                    1,                /* Priority of the task. */
                    NULL);

    xTaskCreate(
                    taskTS,          /* Task function. */
                    "taskTS",        /* String with name of task. */
                    10000,            /* Stack size in words. */
                    NULL,             /* Parameter passed as input of the task */
                    2,                /* Priority of the task. */
                    NULL);
}

int count=0;

void loop() {      // task IDLE - priority 0
  float sensors[4];
  char dbuff[32];
  Serial.println("in the loop");
  xQueueReset(oledqueue);
  xQueueReset(TSqueue);
    sensors[0]=(float) mlx.readAmbientTempC();
    Serial.print("Ambient temp ");
    Serial.print(sensors[0]);
    Serial.println(" *C");
    delay(300);
    sensors[1]=(float)mlx.readObjectTempC();
    Serial.print("Object temp ");
    Serial.print(sensors[1]);
```

```
    Serial.println("%");
    delay(300);
    sensors[2]=(float)SHT21.getTemperature();
    Serial.print("Temp SHT21 ");
    Serial.print(sensors[2]);
    Serial.println("*C");
    delay(300);
    sensors[3]=(float)SHT21.getHumidity();
    Serial.print("Humi SHT21 ");
    Serial.print(sensors[3]);
    Serial.println("%");
    Serial.print("Count:");Serial.println(count); count++;
  xQueueSend(oledqueue,(float *)sensors,0);
  xQueueSend(TSqueue,(float *)sensors,0);
delay(5000);
}
```

A faire :

1. Choisir les capteurs selon leur disponibilité : température, humidité, luminosité, qualité-d'air, température_d'objet, pression atmosphérique, UV, geste, distance, etc (max 4 capteurs) et modifier le code pour les intégrer dans l'application.

2. Remplacer les capteurs par une fonction de réception sur le lien LoRa. Cette fonction doit être implémentée par le biais d'une **ISR** (**Interrupt Service Routine**) `onReceive()` activée par l'arrivée d'une trame .
La ISR doit communiquer avec la tâche dans `loop()` par l'intermédiaire d'une fille de messages.