Laboratoires IoT avec PlatformIO

SmartComputerLab

Contenu

Laboratoire 1

1.1 Premier exemple – l'affichage des données

Dans cet exercice nous allons simplement afficher un titre et 2 valeurs numériques sur l'écran OLED ajouté à notre ESP32.
Vous pouvez créer un nouveau projet **`PlatformIO`** type **`Arduino`**, par exemple, **`Lab1.1.oled`**
Remarque :
Ce projet peut exploiter le fichier de configuration élaboré dans **Lab0**.

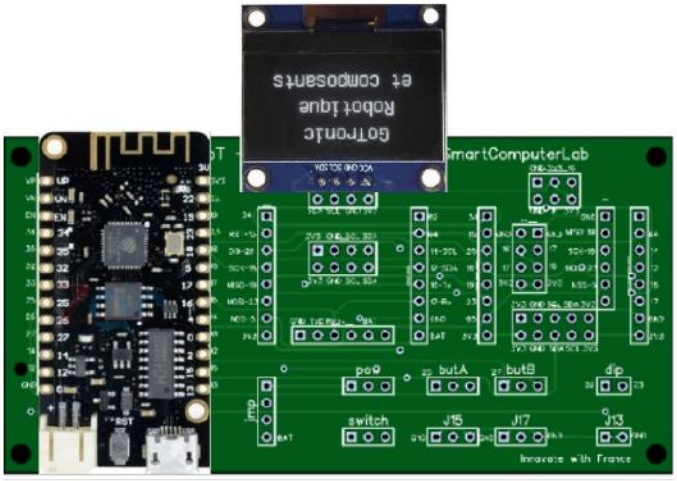

Figure 1.1 Ecran **OLED** ajouté à la carte ESP32 (**`LOLIN D32`**)

Vous devez installer la bibliothèque **`OLED`** compatible avec ESP32 (recherchez – **`SSD1306.h`** dans le **`libraries`**)

Le code

```
#include <Wire.h>
#include "SSD1306Wire.h"
SSD1306Wire display(0x3c, 12, 14);

void display_SSD1306(float d1,float d2,float d3,float d4)
{
  char buff[64];
  display.init();
  //display.flipScreenVertically();
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.setFont(ArialMT_Plain_16);
  display.drawString(0, 0, "ETN – IoT DevKit");
  display.setFont(ArialMT_Plain_10);
  sprintf(buff,"d1: %2.2f, d2: %2.2f\nd3: %2.2f, d4: %2.2f",d1,d2,d3,d4);
  display.drawString(0, 22, buff);
  display.drawString(20, 52, "SmartComputerLab");
  display.display();
}

void setup() {
  Serial.begin(9600);
  Wire.begin(12,14);   // SDA – 12,  SCL – 14
  Serial.println("Wire started");
}
```

```
float v1=0.0,v2=0.0,v3=0.0,v4=0.0;

void loop()
{
  Serial.println("in the loop");
  display_SSD1306(v1,v2,v3,v4);
  v1=v1+0.1;v2=v2+0.2;v3=v3+0.3;v4=v4+0.4;
  delay(2000);
}
```

A faire:

Compléter, compiler, et charger ce programme.

1.2 Deuxième exemple – capture et affichage des valeurs

1.2.1 Capture de la température/humidité par SHT21

Dans cet exercice nous allons lire les valeurs fournies par le capteur Température/Humidité **SHT21** et afficher les 2 valeurs sur l'écran OLED ajouté sur la carte ESP32. Le capteur **SHT21** doit également être connecté sur le bus **I2C**.

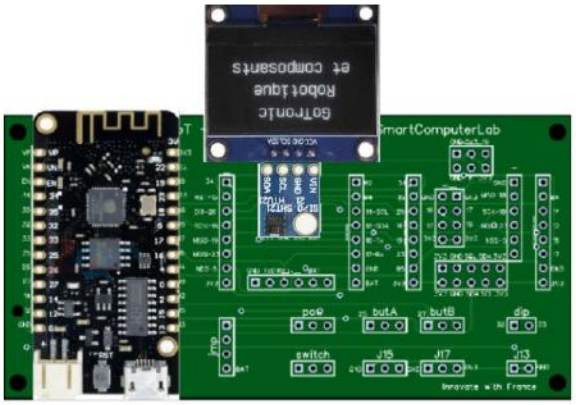

Figure 1.5. **IoT DevKit** avec un capteur de température/humidité **SHT21** et l'écran **OLED.**

Code :

Avant de commencer le codage il faut télécharger et installer la bibliothèque **`SHT21.h`** compatible avec notre carte. Elle se trouve dans le **github**: [e-radionicacom/SHT21-Arduino-Library]

```
#include <Wire.h>
#include "SSD1306Wire.h"
SSD1306Wire display(0x3c, 12, 14);
#include "SHT21.h"
SHT21 SHT21;
float stab[4]= {0.0,0.0,0.0,0.0};

void get_SHT21()
{
    SHT21.begin();
    delay(1000);
    stab[0]=SHT21.getTemperature();
    delay(100);
    stab[1]=SHT21.getHumidity();
    Serial.printf("T:%2.2f, H:%2.2f\n",stab[0],stab[1]);
}

void display_SSD1306(float d1,float d2,float d3,float d4)
{
  char buff[64];
  display.init();  //display.flipScreenVertically();
  display.setTextAlignment(TEXT_ALIGN_LEFT);
  display.setFont(ArialMT_Plain_16);
  display.drawString(0, 0, "ETN - IoT DevKit");
  display.setFont(ArialMT_Plain_10);
  sprintf(buff,"T: %2.2f, H: %2.2f\nd3: %2.2f, d4: %2.2f",d1,d2,d3,d4);
  display.drawString(0, 22, buff);
  display.drawString(20, 52, "SmartComputerLab");
  display.display();
}
```

```
void setup() {
  Serial.begin(9600);
  Wire.begin(12,14);
  Serial.println();
}

void loop()
{
  Serial.println("in the loop");
// à completer
  delay(2000);
}
```

A faire:

1. Créer un projet **PlatformIO**, compléter, compiler, et charger ce programme.
2. Tester le programme suivant (**I2CScan**) pour vérifier que les deux dispositifs (l'écran **SSD1306** et le capteur **SHT21**) sont visibles avec leurs adresses correspondantes (**0x3C** et **0x40**).

```
#include <Wire.h>

void I2Cscan()
{
  byte address;  int nDevices;  delay(200);
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ )
    {
    Wire.beginTransmission(address);
    if(! Wire.endTransmission())  // active address found
      {
      Serial.print("I2C device found at address 0x");
      if (address<16) Serial.print("0");
      Serial.print(address,HEX);  Serial.println("  !");
      nDevices++;
      }
    }
  if (nDevices == 0)
    Serial.println("No I2C devices found\n");
  else
    Serial.println("done\n");
}

void setup()
{
  Serial.begin(9600);
  Wire.begin(12,14,400000); // SDA, SCL on ESP32, 400 kHz rate
  delay(1000); Serial.println();Serial.println();
  I2Cscan(); delay(1000);
}

void loop(){  }
```

1.2.2 Capture de la luminosité par BH1750

Dans cet exercice nous utilisons le capteur **de la luminosité BH1750**

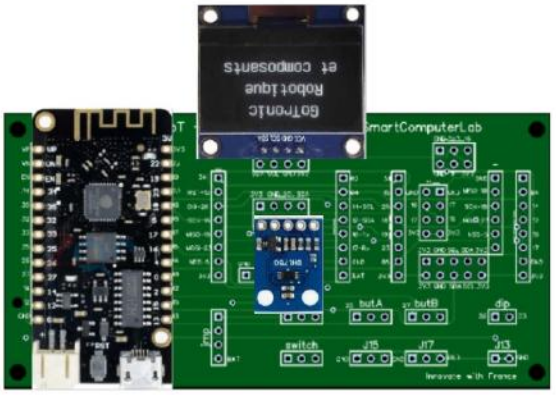

Figure 1.3. **IoT DevKit** avec le capteur Luminosité **BH1750**

Attention : Avant la compilation il faut installer la bibliothèque **`BH1750.h`**

```
#include <Wire.h>
#include <BH1750.h>
BH1750 lightMeter;

void setup(){
  Wire.begin(12,14);
  Serial.begin(9600);
  lightMeter.begin();
  Serial.println("Running...");
  delay(1000);
}

void loop() {
  uint16_t lux = lightMeter.readLightLevel();
  delay(1000);
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(1000);
}
```

A faire :

1. Tester le programme
2. Ajouter l'affichage sur l'écran OLED.

1.2.3 Capture de la luminosité par MAX44009

Dans cet exemple nous utilisons un capteur de luminosité type **MAX44009** (GY-49). Ce capteur est connectée comme le capteur BH1750 sur le bus I2C.
Nous communiquons avec ce capteurs **directement** par les trames **I2C** ce qui permet de mieux comprendre le fonctionnement de ce bus.

Figure 1.4. Capteur de luminosité **MAX44009**

Code Arduino :

```
#include <Wire.h>
#define Addr 0x4A

void setup()
{
Wire.begin(12,14);
Serial.begin(9600);
Wire.beginTransmission(Addr);
Wire.write(0x02);Wire.write(0x40);
Wire.endTransmission();
delay(300);
}

void loop()
{
unsigned int data[2];
Wire.beginTransmission(Addr);
Wire.write(0x03);
Wire.endTransmission();
// Request 2 bytes of data
Wire.requestFrom(Addr, 2);
// Read 2 bytes of data luminance msb, luminance lsb
if (Wire.available() == 2)
{
data[0] = Wire.read();data[1] = Wire.read();
}
// Convert the data to lux
int exponent = (data[0] & 0xF0)>>4;
int mantissa = ((data[0] & 0x0F) << 4) | (data[1]& 0x0F);
float luminance = pow(2, exponent) * mantissa * 0.045;
Serial.print("Ambient Light luminance :");
Serial.print(luminance);
Serial.println(" lux");
delay(500);
}
```

A faire :

3. Tester le programme
4. Ajouter l'affichage sur l'écran OLED.

1.2.5 Capture de la pression/température avec capteur BMP180

Le capteur BMP180 permet de capter la pression atmosphérique et la température. Sa précision est seulement de **+/- 100 Pa** et **+/-1.0C**.

Figure 1.6. **C**apteur de pression/température **BMP180**

La valeur standard de la pression atmosphérique est :

101 325 Pa = 1,013 25 bar = 1 atm

Code `platformio.ini` :

```
[env:lolin_d32]
platform = espressif32
board = lolin_d32
framework = arduino
lib_deps =
      https://github.com/adafruit/Adafruit-BMP085-Library.git
```

Code `main.cpp` :

```
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;

void setup() {
  Serial.begin(9600);
  Wire.begin(12,14);
  bmp.begin();
}

void loop() {
    Serial.print("Temperature = ");
    Serial.print(bmp.readTemperature()); Serial.println(" *C");
    Serial.print("Pressure = "); Serial.print(bmp.readPressure());
    Serial.println(" Pa");
    // Calculate altitude assuming 'standard' barometric
    // pressure of 1013.25 millibar = 101325 Pascal
    Serial.print("Altitude = ");
    Serial.print(bmp.readAltitude());
    Serial.println(" meters");
    // you can get a more precise measurement of altitude
    // if you know the current sea level pressure which will
    // vary with weather and such. If it is 1015 millibars
    // that is equal to 101500 Pascals.
    Serial.print("Real altitude = ");
    Serial.print(bmp.readAltitude(101500));  Serial.println(" meters");
    Serial.println();
    delay(500);
}
```

A faire :

1. Complétez le programme ci-dessus afin d'afficher les données de la température et pression atmosphérique sur l'écran OLED

1.2.6 Capture de présence avec un capteur PIR SR602

Dans l'exemple suivant nous utilisons un capteur de type PIR - **SR602** qui permet de détecter la présence d'une personne en mouvement. Il s'agit d'un capteur simple qui signale l'événement par le basculement de son signal de sortie.

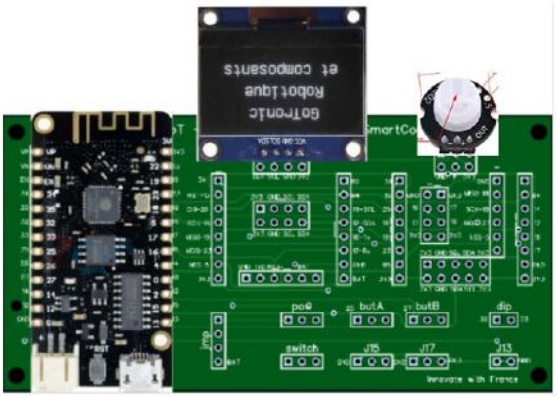

Figure 1.7. **C**apteur de mouvement – **SR602**

Dans le code ci-dessous le changement de la valeur du signal de sortie provoque une interruption captée sur la broche associée à la fonction **`pinChanged()`** qui s'exécute de la façon asynchrone par rapport au déroulement du programme.

```
#define PIR 0  // SIG
bool MOTION_DETECTED = false;

void pinChanged()
{
  MOTION_DETECTED = true;
}
void setup()
{
  Serial.begin(9600);
  pinMode(PIR,INPUT_PULLDOWN);
  attachInterrupt(PIR, pinChanged, RISING);
}

int counter=0;

void loop()
{
  int i=0;
  if(MOTION_DETECTED)
  {
    Serial.println("Motion detected.");
    delay(1000);counter++;
    MOTION_DETECTED = false;
    Serial.println(counter);
  }
}
```

Remarque : La carte doit être réactivée par le bouton **RST**

A faire :

1. Complétez le programme ci-dessus afin d'afficher la valeur du compteur de détection des mouvements sur l'écran OLED