Individual Project

IT essentials 1ITF

Pierina Lopez 1ACS-1
r0913865

CAMPUS

Geel

Technology
Elektronics-ICT / Applied informatics

IT essentials

Course unit: IT essentials

Educational activity: IT essentials

First tier

Academiejaar 2022-2023

Table of contents

Content

Weather Station

1.1 Description

With this Weather Station, it is possible to messure: Temperature, Pressure, and the light in the place the devices are, for the display of this information we use a web site, to show current data and the Ubidots platform to show historical data and send email notifications.

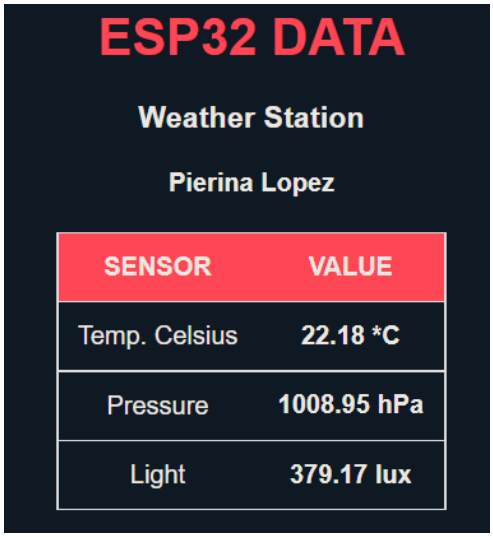

Figure 1 Screenshot of web page

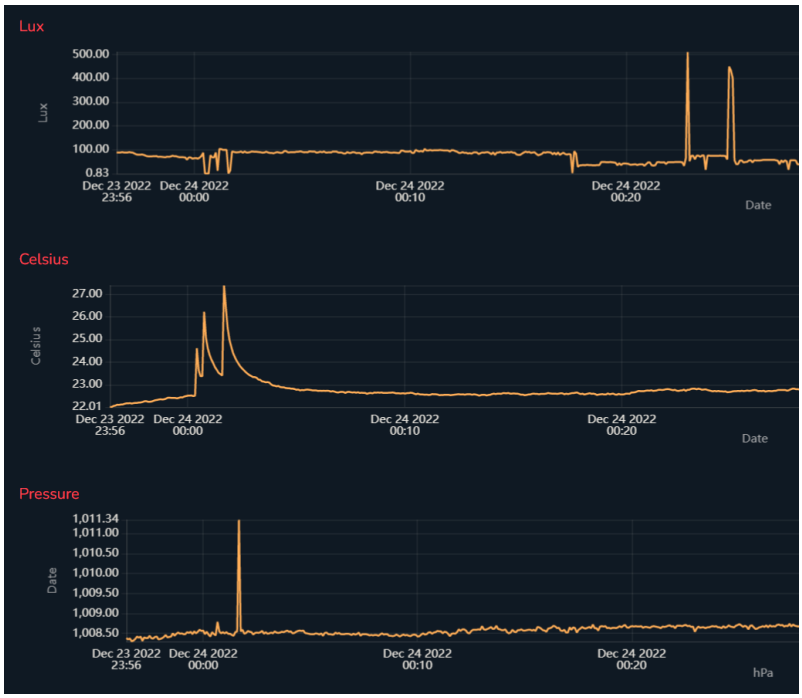

Figure 2 Screenshot of Ubidots

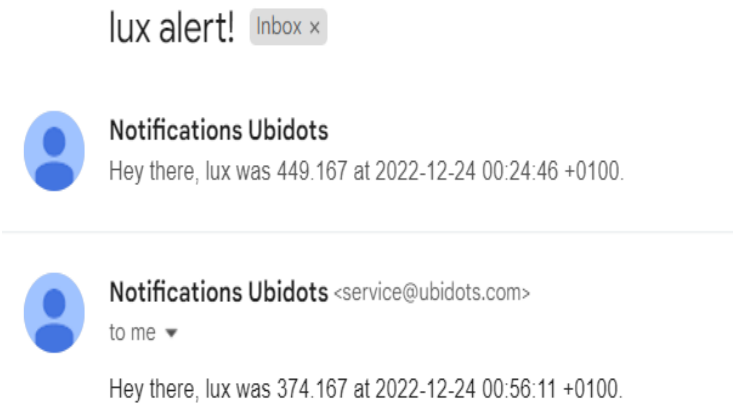

Figure 3 Screenshot of email notifications

1.2 Hardware

To build the Weather Station we need the following:

- ESP32

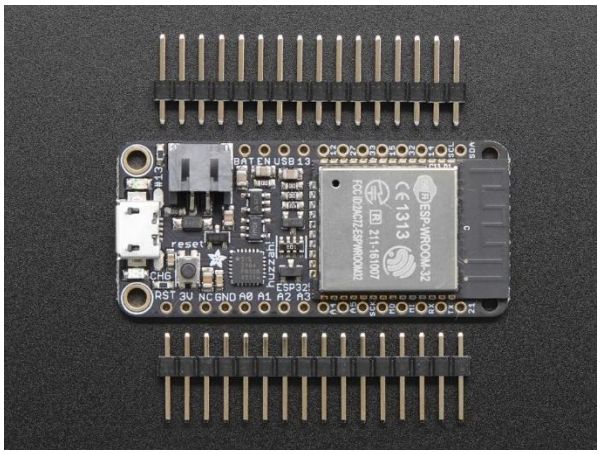

Figure 4 Obtain from https://www.rpibolt.hu/Adafruit-Feather-HUZZAH-ESP32-WiFi-BLE

- Micro USB cable

Figure 5 Obtain from https://iotessentials.be/product/componenten-pakket/

- Jumper cables M-M

Figure 6 Obtain from https://articulo.mercadolibre.com.mx/MLM-594531233-cable-jumpers-dupont-m-m-65-pzas-protoboard-arduino-_JM

- Breadboard

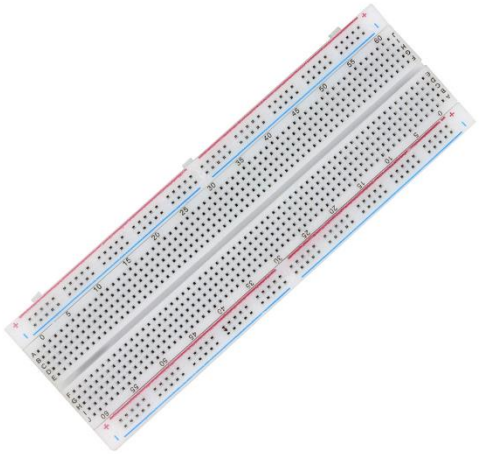

Figure 7 Obtain from https://iotessentials.be/product/componenten-pakket/

- BH1750 // Light sensor

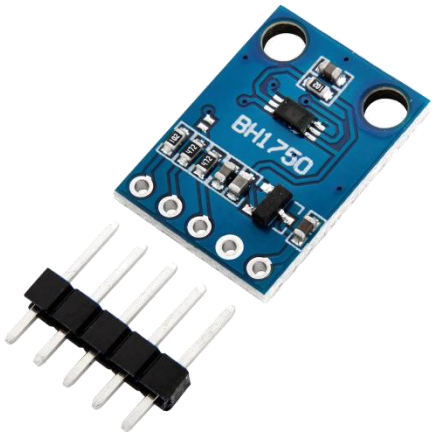

Figure 8 Obtain from https://iotessentials.be/product/componenten-pakket/

- BMP280 // Temperature and pressure sensor

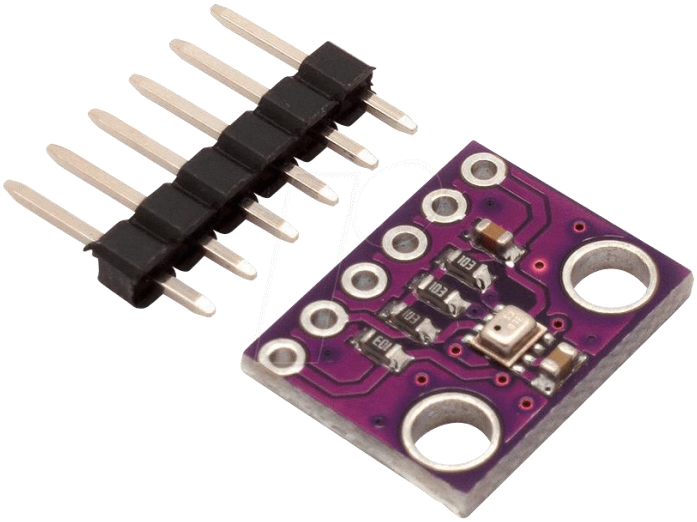

Figure 9 Obtain from https://iotessentials.be/product/componenten-pakket/

1.3 Software and Platforms

For the project we use the tools provide in class, such as:

- Visual Studio Code
- PlatformIO

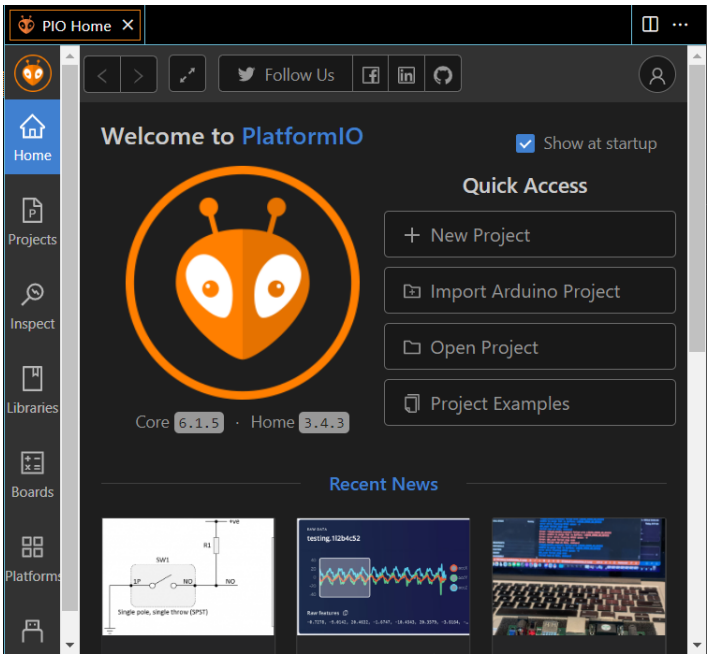

Figure 10 Screenshot of PlatformIO in VS Code

The primary language we use was C++, also for the website we use HTML and CSS.

Additional to this we use Ubidots platform as mention before.

2. Setup Procedure

After getting the materials, the next step is to get ready to code, but before, we need the sofware, after installing "VS CODE" and "PlatformIO", we will need the following libraries:

- Adafruit BMP280 Library
- BH1750
- PubSubClient
- Ubidots ESP MQTT Library

For a fast and easy install of this libraries, after creating your project, go to platformio.ini

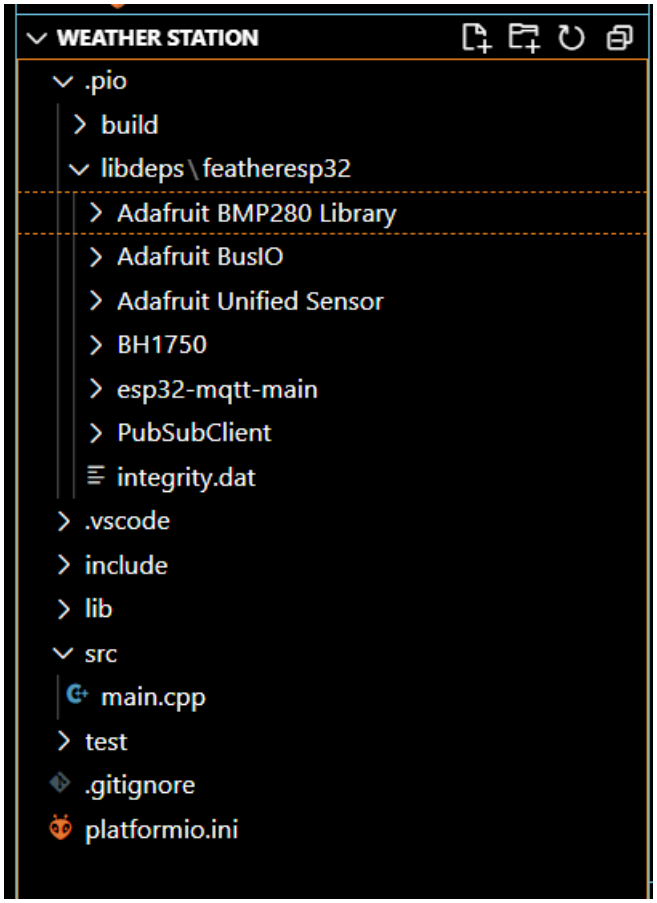

Figure 11 Screenshot of VS Code

Copy and Paste this lines:

```
lib_deps =
    adafruit/Adafruit BMP280 Library @ ^2.6.6
    adafruit/Adafruit BMP280 Library @ ~2.6.6
    adafruit/Adafruit BMP280 Library @ 2.6.6
    claws/BH1750 @ ^1.3.0
    claws/BH1750 @ ~1.3.0
    claws/BH1750 @ 1.3.0
    knolleary/PubSubClient @ ^2.8
    knolleary/PubSubClient @ ~2.8
    knolleary/PubSubClient @ 2.8
```

For the Ubidots ESP MQTT Library, you will need to do other procedure, download the library with this link:

https://github.com/ubidots/esp32-mqtt/archive/refs/heads/main.zip

Go to the project Directory, the open ".pio" directory, "libdeps", "featheresp32" and paste the directory "esp32-mqtt-main".

Now for the connection requiered this is what we have done:

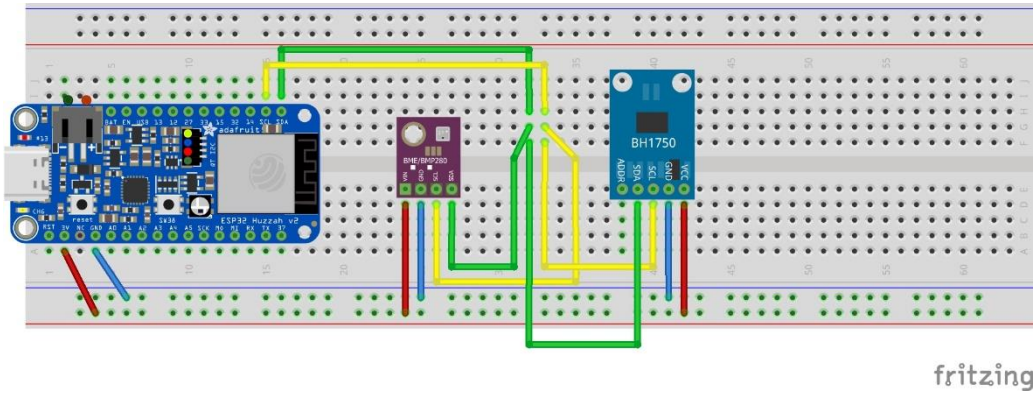

Figure 12 Obtain with Fritzing

Ubidots:

You need to create an account and copy the the token id, that you will need for the code.

2.1. The code

```
// Include Libraries
#include <WiFi.h>
#include <Wire.h>
#include <Adafruit_BMP280.h>
#include <BH1750.h>
#include "UbidotsEsp32Mqtt.h"


#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BMP280 bmp; // I2C
BH1750 lightMeter; // I2C
// Network credentials
const char* ssid     = "SSID";
const char* password = "PASS";
// Set web server port number to 80
WiFiServer server(80);
// Variable to store the HTTP request
String header;
// Current time
unsigned long currentTime = millis();
// Previous time
```

```
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;
// Ubidots Variables - Define Constants
const char *UBIDOTS_TOKEN = "TOKEN";  // Put here your Ubidots
TOKEN
const char *DEVICE_LABEL = "ESP32";   // Put here your Device
label to which data  will be published
const char *VARIABLE_LABEL1 = "Lux"; // Put here your Variable
label to which data  will be published
const char *VARIABLE_LABEL2 = "Celsius"; // Put here your Variable
label to which data  will be published
const char *VARIABLE_LABEL3 = "hPa"; // Put here your Variable
label to which data  will be published
const int PUBLISH_FREQUENCY = 5000; // Update rate in milliseconds

unsigned long timer; //set variable for publishing time

Ubidots ubidots(UBIDOTS_TOKEN);

// Auxiliar Functions

void callback(char *topic, byte *payload, unsigned int length)
{
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
// setup function, where we initialize variables
void setup() {
  Serial.begin(9600);
  Wire.begin();
  lightMeter.begin();

  // check status of BMP280 SENSOR
  bool status;
  if (!bmp.begin(0x76)) {
    Serial.println("Could not find a valid BMP280 sensor");
    while (1);
  }

  //CONNECTING TO WIFI
  Serial.print("Connecting to...");
```

```
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // print IP adress from where we'll see webpage
  Serial.println("");
  Serial.println("IP address: ");
  Serial.println("**copy/paste* ");
  Serial.println(WiFi.localIP());
  server.begin();

  // initialize ubidots
  ubidots.connectToWifi(ssid, password);
  ubidots.setCallback(callback);
  ubidots.setup();
  ubidots.reconnect();

  timer = millis();
}



// MAIN FUNCTION
void loop(){

  // Connection to Ubidots
  if (!ubidots.connected())
  {
    ubidots.reconnect();
  }
  // setting variables to be print in ubidots
  if ((millis() - timer) > PUBLISH_FREQUENCY) // triggers the
routine every 5 seconds
  {
    float lux = lightMeter.readLightLevel();
    ubidots.add(VARIABLE_LABEL1, lux); // Insert your variable
Labels and the value to be sent
    ubidots.add(VARIABLE_LABEL2, bmp.readTemperature());
    ubidots.add(VARIABLE_LABEL3, bmp.readPressure() / 100.0F);
    ubidots.publish(DEVICE_LABEL);
    timer = millis();
  }
  ubidots.loop();


// connecting to wifi and creating the webpage
```

```
  WiFiClient client = server.available();   // reading the new
clients
  // codition when user "client" open the webpage
if (client) {
    currentTime = millis();
    previousTime = currentTime;
    // show that a new user is connected
    Serial.println("New User.");
    String currentLine = "";   // collects data from user
    // while loop for when user is connected
    while (client.connected() && currentTime - previousTime <=
timeoutTime) {
      currentTime = millis();
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        header += c;
        if (c == '\n') {

          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code
(e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's
coming, then a blank line:
            // calling client to send http info
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();


            // show HTML page
            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<style>body { text-align: center;
font-family: Arial; background-color:#0F1923;color: #ECE8E1;}");
            client.println("h1 { color:#FF4655; }");
            client.println("table { border-collapse: collapse;
margin-left:auto; margin-right:auto; }");
            client.println("th { padding: 12px; background-color:
#FF4655; color: #ECE8E1; }");
            client.println("tr { border: 1px solid #ddd; padding:
12px; }");
            client.println("tr:hover { background-color: #0F1923;
}");
            client.println("td { border: none; padding: 12px; }");
```

```
            client.println(".sensor { color:#ECE8E1; font-weight:
bold; padding: 1px; }");
            client.println(".sensor:hover { color:#ECE8E1;
background-color: #FF4655; font-weight: bold; padding: 1px; }");

            // Styling with CSS
            // creating table with information
            client.println("</style></head><body><h1>ESP32
DATA</h1>");
            client.println("</style></head><body><h3>Weather
Station</h3>");
            client.println("</style></head><body><h4>Pierina
Lopez</h4>");
            client.println("<table><tr><th>SENSOR</th><th>VALUE</t
h></tr>");
            client.println("<tr><td>Temp. Celsius</td><td><span
class=\"sensor\">");
            client.println(bmp.readTemperature());
            client.println(" *C</span></td></tr>");
            client.println("<tr><td>Pressure</td><td><span
class=\"sensor\">");
            client.println(bmp.readPressure() / 100.0F);
            client.println(" hPa</span></td></tr>");
            client.println("<tr><td>Light</td><td><span
class=\"sensor\">");
            float lux = lightMeter.readLightLevel();
            client.println(lux);
            client.println(" lux");
            client.println("</body></html>");

            client.println();
            break;
          } else {
            currentLine = "";
          }
        } else if (c != '\r') {
          currentLine += c;
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client out.");
    Serial.println("");
  }
}
```

REFERENCES

- https://randomnerdtutorials.com/esp32-web-server-arduino-ide/
- https://help.ubidots.com/en/articles/748067-connect-an-esp32-devkitc-to-ubidots-over-mqtt

3. Youtube URL

https://youtu.be/CUPVl6ACsKQ

4. Self evaluation

What	Max	Score	Comment
Two I2C sensors on Serial Monitor	11	11	Both sensors working correctly
Current data on esp32 web page	3	3	Sensor Data shows in web page
Historical data on web page (choose only 1 approach)	3	-	
On ESP32	(2)	-	
In Cloud + deep sleep	(3)	-	
With MQTT + deep sleep	(3)	2	I have use Ubidots with MQTT, but not deep sleep
Extra (cumulative)	3		
Style	(1)	1	
Sensor	(1)		
API	(1)	1	From Ubidots, I can send email, sms, telegram, etc. notification about sensor data
Total	**20**	**18**	