Systèmes Embarqués II

Adafruit IO dashboard TP4

ISAT – EPHEC 2020-2021

Juan Alvarez et Olivier Grabenweger

...

2 Table des matières

...

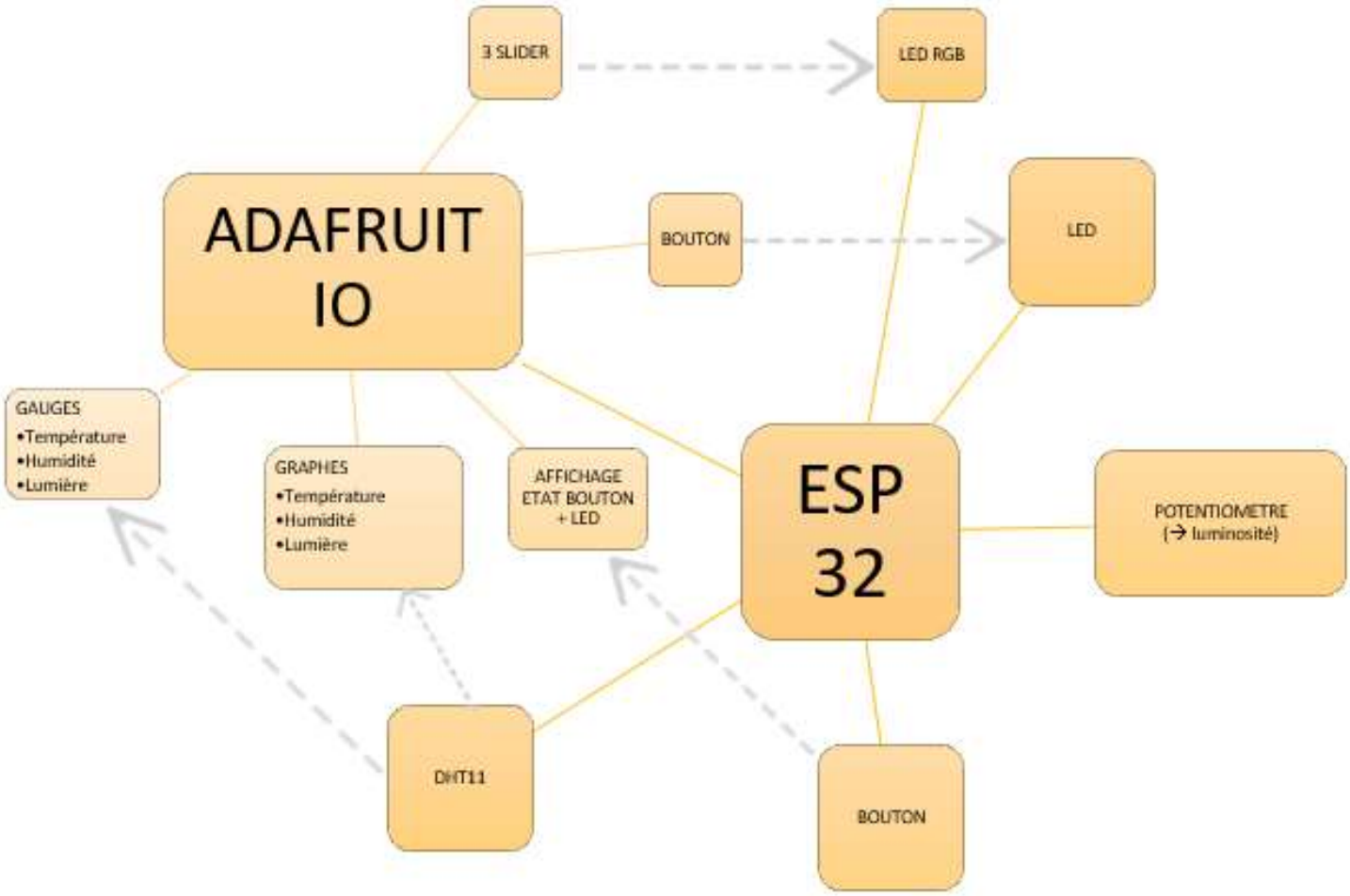
1 Introduction

Pour ce TP, il était demandé de réaliser :

- Un Dashboard contrôlant
  - Les données d'un capteur de température et d'humidité
  - Les données d'un autre capteur de luminosité
- D'afficher l'état d'un bouton
- D'allumer et de contrôler des appareils tel que la LED et la LED RGB sur un ESP32 par l'appui d'un bouton ou de Slider sur cette même Dashboard.

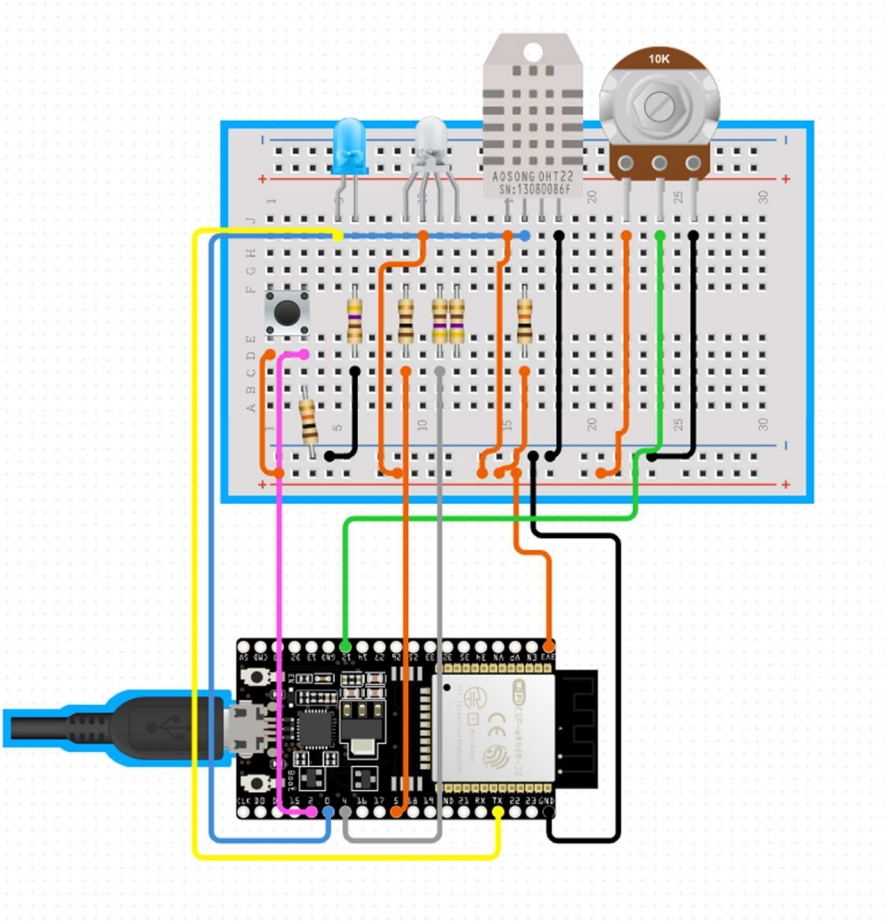
2 Organigramme

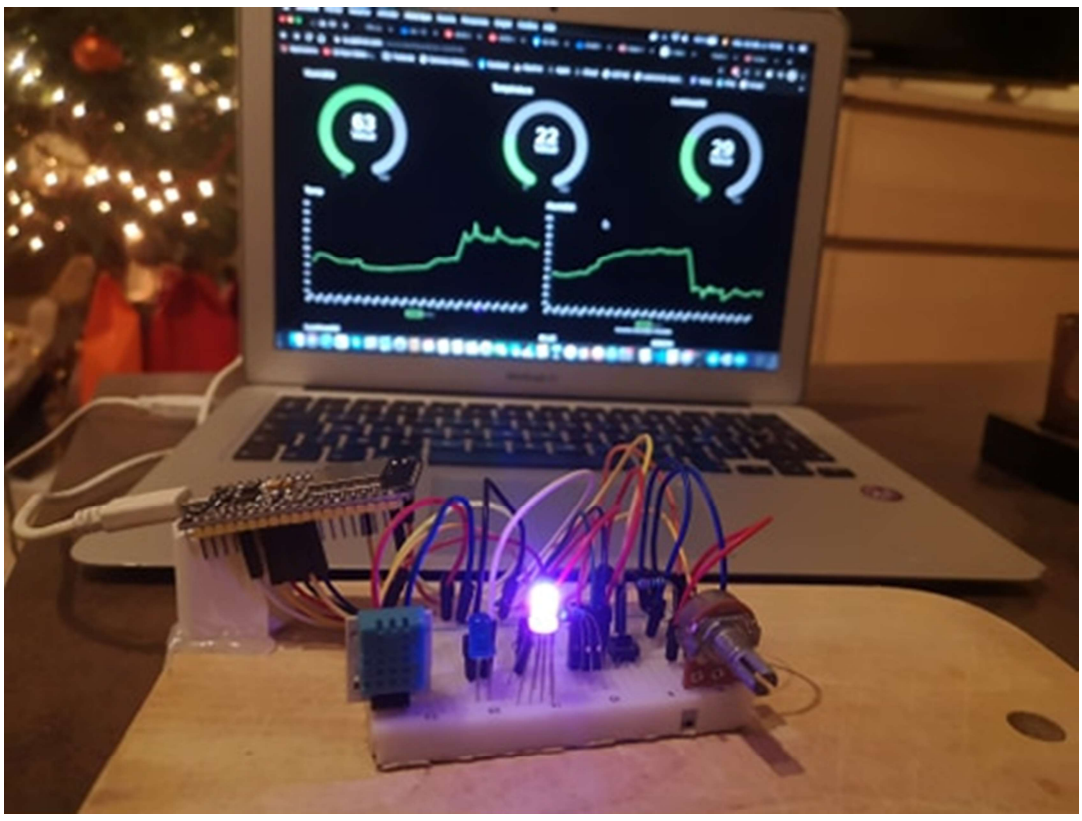
Structure du TP :



...

3 Schéma de câblage





...

4 Code source config

Voici un fichier header que l'on a créé pour se connecter au wifi et afin de déclarer les variables propres à Adafruit.

```
/************************ Adafruit IO Config *******************************/

#define IO_USERNAME "j_factor"
#define IO_KEY "aio_wfbx54qiWpi9TYCEWzISadyuWB3R"

/******************************* WIFI **************************************/

// the AdafruitIO_WiFi client will work with the following boards:
//   - HUZZAH ESP8266 Breakout -> https://www.adafruit.com/products/2471
//   - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
//   - Feather HUZZAH ESP32 -> https://www.adafruit.com/product/3405
//   - Feather M0 WiFi -> https://www.adafruit.com/products/3010
//   - Feather WICED -> https://www.adafruit.com/products/3056
//   - Adafruit PyPortal -> https://www.adafruit.com/product/4116
//   - Adafruit Metro M4 Express AirLift Lite ->
//  https://www.adafruit.com/product/4000
//   - Adafruit AirLift Breakout -> https://www.adafruit.com/product/4201

#define WIFI_SSID "juanitopepito"
#define WIFI_PASS "aaaa1230"

#include "AdafruitIO_WiFi.h"

#if defined(USE_AIRLIFT) || defined(ADAFRUIT_METRO_M4_AIRLIFT_LITE) ||         \
    defined(ADAFRUIT_PYPORTAL)
// Configure the pins used for the ESP32 connection
#if !defined(SPIWIFI_SS) // if the wifi definition isnt in the board variant
// Don't change the names of these #define's! they match the variant ones
#define SPIWIFI SPI
#define SPIWIFI_SS 10  // Chip select pin
#define SPIWIFI_ACK 9  // a.k.a BUSY or READY pin
#define ESP32_RESETN 6 // Reset pin
#define ESP32_GPIO0 -1 // Not connected
#endif
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS, SPIWIFI_SS,
                   SPIWIFI_ACK, ESP32_RESETN, ESP32_GPIO0, &SPIWIFI);
#else
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
#endif
```

...

5 Code source ESP32

Etant donné qu'on ne disposait que de 10 feed dans la version gratuite du programme, nous avons, au départ, travaillé avec le RGB utilisant 1 seul feed mais comme on nous demandait de le faire avec 3 feed, nous avons décidé de tout de même laisser la fonction avec1 feed en la commentant tout en faisant celle avec 3 feed.

```
1   /*****  Juan and Oli dev for Embedded Systems *****/
2  // Creating a dashbord on adafruit
3
4  #include <Arduino.h>
5  #include <AdafruitIO.h>
6  #include "config.h"
7  #include <Adafruit_Sensor.h>
8  #include <DHT.h>
9  #include <DHT_U.h>
10
11 // pin connected to DH11 data line
12 #define DHTPIN 4
13
14 // pin connected to POT
15 #define ANALOG_PIN 35
16
17 // pot state var
18 int current = 0;
19 int last = -1;
20
21 // pins to rgb
22 #define RED_PIN   16
23 #define GREEN_PIN 17
24 #define BLUE_PIN  18
25
26
27 // dht type --> dht11
28 #define DHTTYPE    DHT11     // DHT 11
29
30 // Button Pin
31 #define BUTTON_PIN 5
32
33 // LED Pin
34 #define LED_PIN 2
35
36 //creating an object dht
37 DHT dht(DHTPIN, DHTTYPE);
38
39 float h,t,f;      //DHT var
40 long send_time,send_time2; //sending timers
41 // button state
42 bool btn_state = false;
43 bool prv_btn_state = false;
44
45 // set up the 'led' feed
46 AdafruitIO_Feed *led = io.feed("led");
47
48 // set up the 'button' feed
49 AdafruitIO_Feed *button = io.feed("button");
50
```

...

```
// set up the 'température' feed
AdafruitIO_Feed *temp = io.feed("temp");

// set up the 'humidity' feed
AdafruitIO_Feed *humi = io.feed("humi");


// set up the 'humidity' feed
AdafruitIO_Feed *color = io.feed("color");

// set up RGB feed
AdafruitIO_Feed *RGB_B = io.feed("RGB_B");
AdafruitIO_Feed *RGB_G = io.feed("RGB_G");
AdafruitIO_Feed *RGB_R= io.feed("RGB_R");

// analog feed for adc
AdafruitIO_Feed *lumi = io.feed("lumi");

// this function is called whenever a 'led' message
// is received from Adafruit IO. it was attached to
// the counter feed in the setup() function above.
void handleLed(AdafruitIO_Data *data) {
  Serial.print("received <- ");

  if(data->toPinLevel() == HIGH)
    Serial.println("LED_TURNED_ON");

  digitalWrite(LED_PIN, data->toPinLevel());
}


// FUNCTION TO HANDLE RGB COLOR

/*void handlecolor(AdafruitIO_Data *data)
{
  // print RGB values and hex value
  Serial.println("Received:");
  Serial.print("  - R: ");
  Serial.println(data->toRed());
  Serial.print("  - G: ");
  Serial.println(data->toGreen());
  Serial.print("  - B: ");
  Serial.println(data->toBlue());
  Serial.print("  - HEX: ");
  Serial.println(data->value());

  // invert RGB values for common anode LEDs
  #if defined(ARDUINO_ARCH_ESP32) // ESP32 analogWrite
    ledcWrite(1, 255 - data->toRed());
    ledcWrite(2, 255 - data->toGreen());
    ledcWrite(3, 255 - data->toBlue());
  #else
    analogWrite(RED_PIN, 255 - data->toRed());
    analogWrite(GREEN_PIN, 255 - data->toGreen());
    analogWrite(BLUE_PIN, 255 - data->toBlue());
  #endif

 } */
```

...

```
109
110 // RGB FROM 3 SLIDERS
111
112 void handleRGBB(AdafruitIO_Data *data) {
113   #if defined(ARDUINO_ARCH_ESP32) // ESP32 analogWrite
114     ledcWrite(3,255 - data->toInt());
115   #else
116     analogWrite(BLUE_PIN, 255 - data->toBlue());
117   #endif
118 }
119
120 void handleRGBR(AdafruitIO_Data *data) {
121   #if defined(ARDUINO_ARCH_ESP32) // ESP32 analogWrite
122     ledcWrite(1,255 - data->toInt());
123   #else
124     analogWrite(RED_PIN, 255 - data->toRed());
125   #endif
126 }
127
128 void handleRGBG(AdafruitIO_Data *data) {
129   #if defined(ARDUINO_ARCH_ESP32) // ESP32 analogWrite
130     ledcWrite(2,255 - data->toInt());
131   #else
132     analogWrite(GREEN_PIN, 255 - data->toGreen());
133   #endif
134 }
135
136 void setup() {
137
138   // set button pin as an input
139   pinMode(BUTTON_PIN, INPUT);
140
141   // set LED pin as an output
142   pinMode(LED_PIN, OUTPUT);
143
144   #if defined(ARDUINO_ARCH_ESP32) // ESP32 pinMode
145     // assign rgb pins to channels
146     ledcAttachPin(RED_PIN,1);
147     ledcAttachPin(GREEN_PIN,2);
148     ledcAttachPin(BLUE_PIN,3);
149
150     //adcpin
151     ledcAttachPin(ANALOG_PIN,4);
152     // init. channels
153     ledcSetup(1, 1000, 8);
154     ledcSetup(2, 1000, 8);
155     ledcSetup(3, 1000, 8);
156   #else
157     pinMode(RED_PIN, OUTPUT);
158     pinMode(GREEN_PIN, OUTPUT);
159     pinMode(BLUE_PIN, OUTPUT);
160   #endif
161
162   // start the serial connection
163   Serial.begin(115200);
164
165   // wait for serial monitor to open
166   while(! Serial);
```

...

```
167   // Initialize device.
168   dht.begin();
169
170   // connect to io.adafruit.com
171   Serial.print("Connecting to Adafruit IO");
172   io.connect();
173
174   // set up a message handler for the count feed.
175   // the handleMessage function (defined below)
176   // will be called whenever a message is
177   // received from adafruit io.
178   led->onMessage(handleLed);
179
180   //color->onMessage(handlecolor);
181   RGB_R->onMessage(handleRGBR);
182   RGB_B->onMessage(handleRGBB);
183   RGB_G->onMessage(handleRGBG);
184
185   // wait for a connection
186   while(io.status() < AIO_CONNECTED) {
187     Serial.print(".");
188     delay(500);
189   }
190
191   // we are connected
192   Serial.println();
193   Serial.println(io.statusText());
194   led->get();
195
196 }
197
198 void loop() {
199
200   io.run();
201
202   if(send_time <= millis()){
203     // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
204     h = dht.readHumidity();
205     // Read temperature as Celsius (the default)
206     t = dht.readTemperature();
207     // Read temperature as Fahrenheit (isFahrenheit = true)
208     f = dht.readTemperature(true);
209
210     // Check if any reads failed and exit early (to try again).
211     if (isnan(h) || isnan(t) || isnan(f)) {
212       Serial.println(F("Failed to read from DHT sensor!"));
213       //return;
214     }
215     // Compute heat index in Fahrenheit (the default)
216     float hif = dht.computeHeatIndex(f, h);
217     // Compute heat index in Celsius (isFahreheit = false)
218     float hic = dht.computeHeatIndex(t, h, false);
219
220     //SERIAL PRINTING
221     Serial.print(F("Humidity: "));
222     Serial.print(h);
223     Serial.print(F("%  Temperature: "));
224     Serial.print(t);
```

...

```
225     Serial.print(F("°C "));
226     Serial.print(f);
227     Serial.print(F("°F  Heat index: "));
228     Serial.print(hic);
229     Serial.print(F("°C "));
230     Serial.print(hif);
231     Serial.println(F("°F"));
232 
233       // save the btn_state state to the 'temp' feed on adafruit io
234     Serial.print("sending temperature -> ");
235     Serial.println(t);
236     temp->save(t);
237 
238     // save the btn_state state to the 'humi' feed on adafruit io
239     Serial.print("sending humidity-> ");
240     Serial.println(h);
241     humi->save(h);
242 
243     // set next time you want to do read from dht
244     send_time = millis() + 10000;
245   }
246 // grab the current state of the photocell
247   current = analogRead(ANALOG_PIN);
248   current = map(current,0,4095,0,100);
249 
250   // return if the value hasn't changed
251   if(current != last){
252     // save the current state to the 'lumi' feed
253     Serial.print("sending -> ");
254     Serial.println(current);
255     // store last photocell state
256     last = current;
257   }
258   if(send_time2 <= millis()){
259     lumi->save(current);
260     send_time2 = millis() + 60000;
261   }
262       // grab the btn_state state of the button.
263   if(digitalRead(BUTTON_PIN) == LOW)
264     btn_state = false;
265   else
266     btn_state = true;
267   // return if the btn state hasn't changed
268   if(btn_state == prv_btn_state)
269     return;
270 
271   // save the btn_state state to the 'button' feed on adafruit io
272   Serial.print("sending button -> ");
273   Serial.println(btn_state);
274   button->save(btn_state);
275 
276   // store last button state
277   prv_btn_state = btn_state;
278 }
```

...

6 Conclusion

L'utilisation d'un outil externe pour réaliser des dashboard est particulièrement intéressant. A travers ce TP, nous avons donc pu découvrir Adafruit IO.

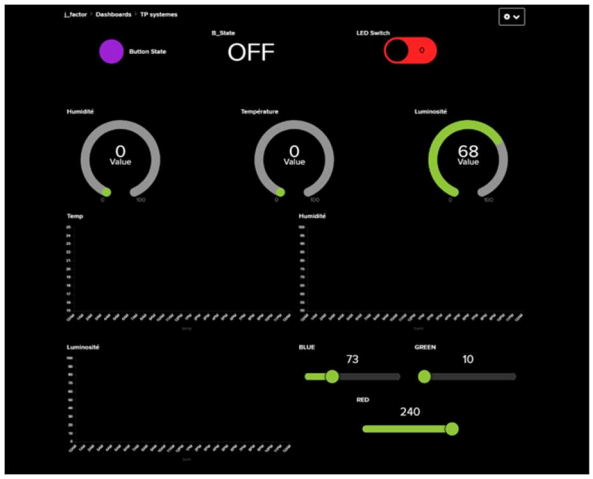
Seul quelques défauts liés à la version gratuite rendent l'utilisation d'un tel logiciel un peu moins accessible. Néanmoins, en comparaison de ce qui avait été réalisé lors du tp avec le serveur Web, il est facile de se rendre compte de la puissance d'un tel programme. En effet, l'esthétique que nous offre Adafruit IO est loin devant celle offerte par notre Webserver, cela permet d'économiser énormément de temps de développement et de l'investir pour perfectionner notre travail.

Outre l'aspect esthétique qu'offre Adafruit IO, il permet également la sécurisation des données. On ne doit donc pas penser à sécuriser nous-même nos données, ce qui permet encore une fois d'axer notre travail sur du concret.

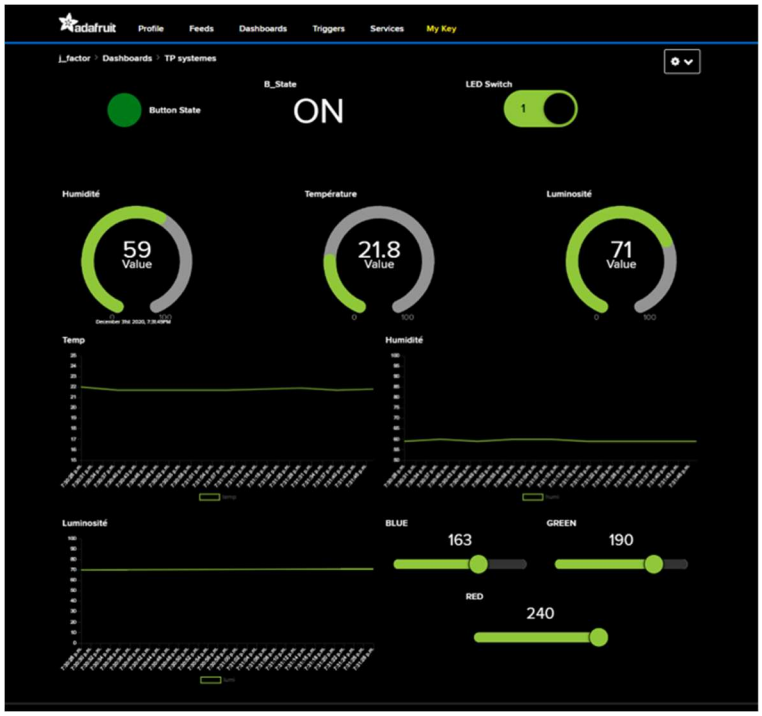
7 Annexes, bibliographie et illustrations

a) Annexes

Voici les photos de l'aspect de notre Dashboard sur Adafruit IO.



Hors fonctionnement



En fonctionnement (bouton appuyé sur l'ESP et envoie des données de température et d'humidité.

b) Bibliographie

- Ressources Moodle
- https://github.com/adafruit/Adafruit_IO_Arduino