

|  |
| --- |
| **Systèmes Embarqués II** |
| Adafruit IO dashboard TP4 |
| ISAT – EPHEC 2020-2021  Juan Alvarez et Olivier Grabenweger |

Table des matières

[1 Introduction 2](#_Toc60835834)

[2 Organigramme 2](#_Toc60835835)

[3 Schéma de câblage 3](#_Toc60835836)

[4 Code source config 4](#_Toc60835837)

[5 Code source ESP32 5](#_Toc60835838)

[6 Conclusion 10](#_Toc60835839)

[7 Annexes, bibliographie et illustrations 10](#_Toc60835840)

[a) Annexes 10](#_Toc60835841)

[b) Bibliographie 10](#_Toc60835842)

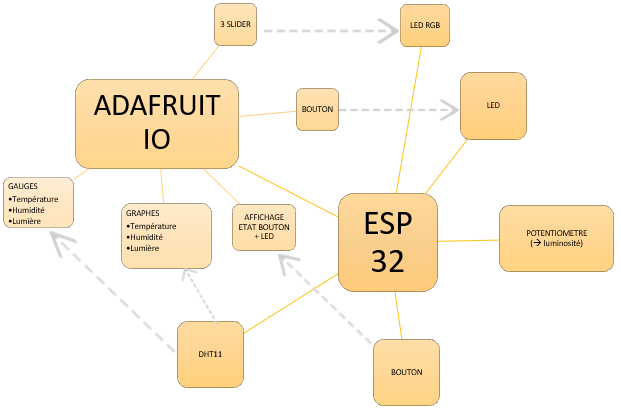
# Introduction

Pour ce TP, il était demandé de réaliser :

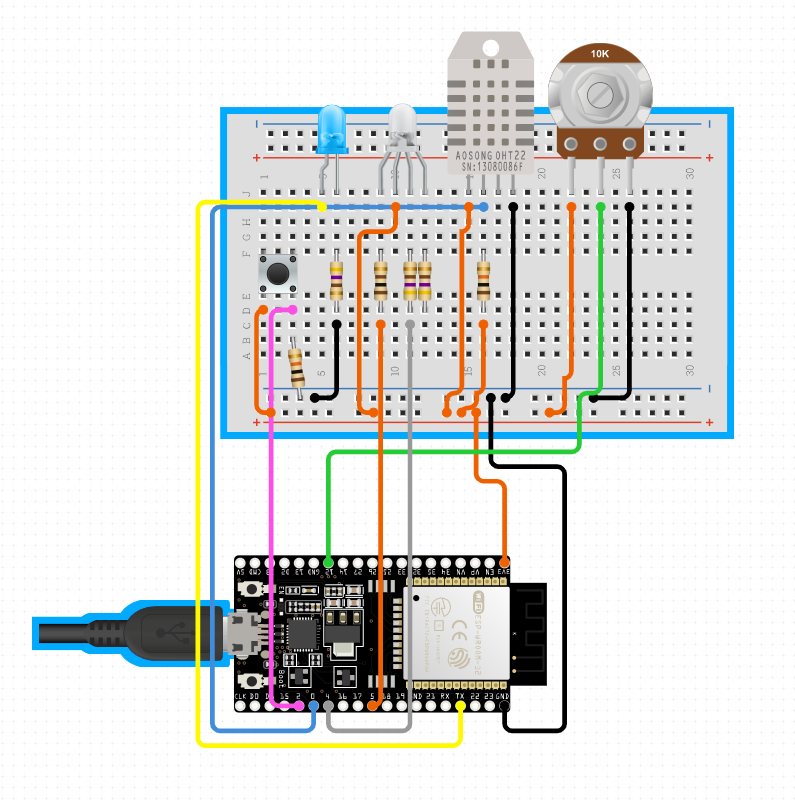
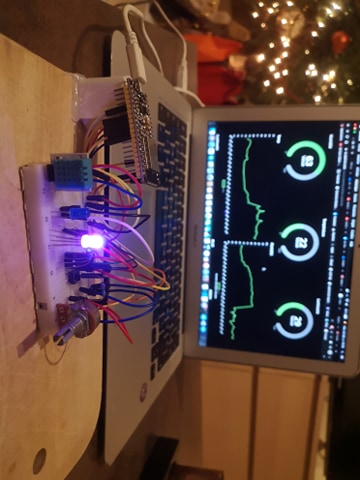
* Un Dashboard contrôlant
* Les données d’un capteur de température et d’humidité
* Les données d’un autre capteur de luminosité
* D’afficher l’état d’un bouton
* D’allumer et de contrôler des appareils tel que la LED et la LED RGB sur un ESP32 par l’appui d’un bouton ou de Slider sur cette même Dashboard.

# Organigramme

Structure du TP :



# Schéma de câblage



# 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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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 |

# 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  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278 | /\*\*\*\*\* Juan and Oli dev for Embedded Systems \*\*\*\*\*/  // Creating a dashbord on adafruit  #include <Arduino.h>  #include <AdafruitIO.h>  #include "config.h"  #include <Adafruit\_Sensor.h>  #include <DHT.h>  #include <DHT\_U.h>  // pin connected to DH11 data line  #define DHTPIN 4  // pin connected to POT  #define ANALOG\_PIN 35  // pot state var  **int** current = **0**;  **int** last = -**1**;  // pins to rgb  #define RED\_PIN 16  #define GREEN\_PIN 17  #define BLUE\_PIN 18  // dht type --> dht11  #define DHTTYPE DHT11 // DHT 11  // Button Pin  #define BUTTON\_PIN 5  // LED Pin  #define LED\_PIN 2  //creating an object dht  DHT **dht**(DHTPIN, DHTTYPE);  **float** h,t,f; //DHT var  **long** send\_time,send\_time2; //sending timers  // button state  **bool** btn\_state = false;  **bool** prv\_btn\_state = false;  // set up the 'led' feed  AdafruitIO\_Feed \*led = io.feed("led");  // set up the 'button' feed  AdafruitIO\_Feed \*button = io.feed("button");  // 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  } \*/  // RGB FROM 3 SLIDERS  **void** **handleRGBB**(AdafruitIO\_Data \*data) {  #**if** defined(ARDUINO\_ARCH\_ESP32) // ESP32 analogWrite  ledcWrite(**3**,**255** - data->toInt());  #**else**  analogWrite(BLUE\_PIN, **255** - data->toBlue());  #endif  }  **void** **handleRGBR**(AdafruitIO\_Data \*data) {  #**if** defined(ARDUINO\_ARCH\_ESP32) // ESP32 analogWrite  ledcWrite(**1**,**255** - data->toInt());  #**else**  analogWrite(RED\_PIN, **255** - data->toRed());  #endif  }  **void** **handleRGBG**(AdafruitIO\_Data \*data) {  #**if** defined(ARDUINO\_ARCH\_ESP32) // ESP32 analogWrite  ledcWrite(**2**,**255** - data->toInt());  #**else**  analogWrite(GREEN\_PIN, **255** - data->toGreen());  #endif  }  **void** **setup**() {  // set button pin as an input  pinMode(BUTTON\_PIN, INPUT);  // set LED pin as an output  pinMode(LED\_PIN, OUTPUT);  #**if** defined(ARDUINO\_ARCH\_ESP32) // ESP32 pinMode  // assign rgb pins to channels  ledcAttachPin(RED\_PIN,**1**);  ledcAttachPin(GREEN\_PIN,**2**);  ledcAttachPin(BLUE\_PIN,**3**);  //adcpin  ledcAttachPin(ANALOG\_PIN,**4**);  // init. channels  ledcSetup(**1**, **1000**, **8**);  ledcSetup(**2**, **1000**, **8**);  ledcSetup(**3**, **1000**, **8**);  #**else**  pinMode(RED\_PIN, OUTPUT);  pinMode(GREEN\_PIN, OUTPUT);  pinMode(BLUE\_PIN, OUTPUT);  #endif  // start the serial connection  Serial.begin(**115200**);  // wait for serial monitor to open  **while**(! Serial);  // Initialize device.  dht.begin();  // connect to io.adafruit.com  Serial.print("Connecting to Adafruit IO");  io.connect();  // set up a message handler for the count feed.  // the handleMessage function (defined below)  // will be called whenever a message is  // received from adafruit io.  led->onMessage(handleLed);  //color->onMessage(handlecolor);  RGB\_R->onMessage(handleRGBR);  RGB\_B->onMessage(handleRGBB);  RGB\_G->onMessage(handleRGBG);  // wait for a connection  **while**(io.status() < AIO\_CONNECTED) {  Serial.print(".");  delay(**500**);  }  // we are connected  Serial.println();  Serial.println(io.statusText());  led->get();  }  **void** **loop**() {  io.run();  **if**(send\_time <= millis()){  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)  h = dht.readHumidity();  // Read temperature as Celsius (the default)  t = dht.readTemperature();  // Read temperature as Fahrenheit (isFahrenheit = true)  f = dht.readTemperature(true);  // Check if any reads failed and exit early (to try again).  **if** (isnan(h) || isnan(t) || isnan(f)) {  Serial.println(F("Failed to read from DHT sensor!"));  //return;  }  // Compute heat index in Fahrenheit (the default)  **float** hif = dht.computeHeatIndex(f, h);  // Compute heat index in Celsius (isFahreheit = false)  **float** hic = dht.computeHeatIndex(t, h, false);  //SERIAL PRINTING  Serial.print(F("Humidity: "));  Serial.print(h);  Serial.print(F("% Temperature: "));  Serial.print(t);  Serial.print(F("°C "));  Serial.print(f);  Serial.print(F("°F Heat index: "));  Serial.print(hic);  Serial.print(F("°C "));  Serial.print(hif);  Serial.println(F("°F"));  // save the btn\_state state to the 'temp' feed on adafruit io  Serial.print("sending temperature -> ");  Serial.println(t);  temp->save(t);  // save the btn\_state state to the 'humi' feed on adafruit io  Serial.print("sending humidity-> ");  Serial.println(h);  humi->save(h);  // set next time you want to do read from dht  send\_time = millis() + **10000**;  }  // grab the current state of the photocell  current = analogRead(ANALOG\_PIN);  current = map(current,**0**,**4095**,**0**,**100**);  // return if the value hasn't changed  **if**(current != last){  // save the current state to the 'lumi' feed  Serial.print("sending -> ");  Serial.println(current);  // store last photocell state  last = current;  }  **if**(send\_time2 <= millis()){  lumi->save(current);  send\_time2 = millis() + **60000**;  }  // grab the btn\_state state of the button.  **if**(digitalRead(BUTTON\_PIN) == LOW)  btn\_state = false;  **else**  btn\_state = true;  // return if the btn state hasn't changed  **if**(btn\_state == prv\_btn\_state)  **return**;  // save the btn\_state state to the 'button' feed on adafruit io  Serial.print("sending button -> ");  Serial.println(btn\_state);  button->save(btn\_state);  // store last button state  prv\_btn\_state = btn\_state;  } |

# 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.

# Annexes, bibliographie et illustrations

## Annexes

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

|  |  |
| --- | --- |
| Hors fonctionnement Aucune description disponible. | Aucune description disponible.En fonctionnement (bouton appuyé sur l’ESP et envoie des données de température et d’humidité. |

## Bibliographie

* Ressources Moodle
* h[ttps://github.com/adafruit/Adafruit\_IO\_Arduino](https://github.com/adafruit/Adafruit_IO_Arduino?fbclid=IwAR2vdvYrLHmWr-z0KUKUJXUNSBf4E0ZppaDktIohq-b2ch4QKGTVtkIQn3U)