ephec -isat

tAQUET tIMON Michael Nwogburu

Portfolio

SYSTEMES EMBARQUES INDUSTRIE 4.0



Table des matières

[1) TP1\_BUSCAN 2](#_Toc118488051)

[Objectif du TP 2](#_Toc118488052)

[Introduction 2](#_Toc118488053)

[Code source avec commentaires 2](#_Toc118488054)

[Analyse du code source 2](#_Toc118488055)

[Lien software hardware 2](#_Toc118488056)

[Conclusion 2](#_Toc118488057)

[2) TP2\_Webserver 3](#_Toc118488058)

[Objectif du TP 3](#_Toc118488059)

[Introduction 3](#_Toc118488060)

[Postman 3](#_Toc118488061)

[Langage de programmation utilisée 3](#_Toc118488062)

[HTML 3](#_Toc118488063)

[JAVASCRIPT 3](#_Toc118488064)

[CSS 3](#_Toc118488065)

[C++ 3](#_Toc118488066)

[Code source avec commentaires 3](#_Toc118488067)

[Analyse du code source 3](#_Toc118488068)

[Lien software hardware 3](#_Toc118488069)

[Réalisation 3](#_Toc118488070)

[Conclusion 3](#_Toc118488071)

[3) TP3\_Google\_Sheet 3](#_Toc118488072)

[Objectif du TP 3](#_Toc118488073)

[Introduction 3](#_Toc118488074)

[Google Sheet 3](#_Toc118488075)

[Postman 3](#_Toc118488076)

[Code source avec commentaires 3](#_Toc118488077)

[Analyse du code source 3](#_Toc118488078)

[Lien software hardware 3](#_Toc118488079)

[Réalisation 3](#_Toc118488080)

[Conclusion 3](#_Toc118488081)

[4) TP4\_MQTT 3](#_Toc118488082)

[Objectif du TP 3](#_Toc118488083)

[Introduction 4](#_Toc118488084)

[PROTOCOLE MQTT 4](#_Toc118488085)

[ADAFRUIT\_IO 4](#_Toc118488086)

[FEEDS 4](#_Toc118488087)

[MQTT-Broker 4](#_Toc118488088)

[Réalisation 4](#_Toc118488089)

[Code source avec commentaires 4](#_Toc118488090)

[Analyse du code source 4](#_Toc118488091)

[Lien software hardware 4](#_Toc118488092)

[Conclusion 4](#_Toc118488093)

[5) TP5\_Blynk 4](#_Toc118488094)

[Objectif du TP 4](#_Toc118488095)

[Introduction 4](#_Toc118488096)

[Blynk 4](#_Toc118488097)

[Serveur 4](#_Toc118488098)

[Réalisation 4](#_Toc118488099)

[Code source avec commentaires 4](#_Toc118488100)

[Analyse du code source 4](#_Toc118488101)

[Lien software hardware 4](#_Toc118488102)

[Conclusion 4](#_Toc118488103)

[6) TP6\_Nodered 4](#_Toc118488104)

# 1) TP1\_BUSCAN

## Objectif du TP

## introduction

## Code source avec commentaires

## Analyse du code source

## Lien software hardware

## Conclusion

# 2) TP2\_Webserver

## Objectif du TP

## Introduction

## Postman

## **Websocket**

## Langage de programmation utilisée

### HTML

### JAVASCRIPT

### CSS

### C++

## Code source avec commentaires

## Analyse du code source

## Lien software hardware

## Réalisation

## Conclusion

# 3) TP3\_Google\_Sheet

## Objectif du TP

## Introduction

## Google Sheet

## PowerBI

## Postman

## Réalisation

## Code source avec commentaires

## Analyse du code source

## Lien software hardware

## Conclusion

# 4) TP4\_MQTT

# Objectif du TP

L’objectif du TP4 est de nous familiariser avec un Dashboard de l’environnement ADAFRUIT\_IO.

Avec le protocole MQTT comme nouvelle méthode de communication.

Nous devrons commencer par afficher 3 jauges et 3 graphes pour visualiser les variations des grandeurs suivantes :

* Température
* Humidité
* Luminosité

Un bouton on/off pour commander une LED suivie d’une zone de texte qui affichera bien l’état actuel de la LED.

Et pour finir 3 glissières pour faire varier les couleurs d’une LED RGB néo pixel.

## Introduction

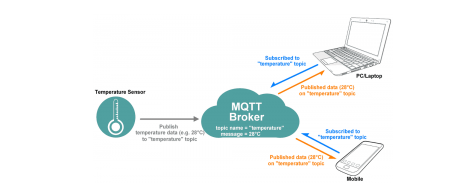
## PROTOCOLE MQTT

Le protocole MQTT qui signifie Message Queuing Telemetry Transport est un protocole qui a été fait pour l’Internet des objets.  
La télémétrie est extrêmement légère et a été conçue pour les appareils contraints et avec une faible bande passante.

Pour pouvoir communiquer avec un protocole de type MQTT il faut d’abord avoir les éléments suivants :

* **Broker :** c’est un serveur qui distribue et qui reçois les informations au clients intéressés.
* **Client :** Le clients sont les appareils qui se connecte au broker par exemple pc, esp32, smartphone.
* **Topic :** Le topic est le sujet sur lequel les clients pourront souscrire des informations ou bien en publier.

Après avoir implémenté le broker, client et le topic le client peut alors envoyer des informations à un certain **Topic** avec la fonction **Publish** et peut aussi en souscrire avec la fonction **Subscribe.**



## ADAFRUIT\_IO

Adafruit est une entreprise fondée en 2005 par [Limor Fried](https://fr.wikipedia.org/wiki/Limor_Fried" \o "Limor Fried), une [ingénieure](https://fr.wikipedia.org/wiki/Ing%C3%A9nieure) américaine, qui se spécialise dans la vente, la production de composants électroniques et mets à disposition des code sources pour diverse utilisation.

Adafruit\_IO est un environnement WEB permettant de collecter les données des objets connectés dans des feeds et de d'organiser leur représentation dans des Dashboard.

### **Feeds**

Les Feeds qui veut dire fils de données est en quelque sortes des variables contenant des informations modifiables, ces informations la pourront être retransmis ou modifier selon son utilisation.

Après avoir créé un Feed il reste plus qu’à le lier à un widget de notre Dashboard.

### **Réalisation**

**FEED**

**Dashboard**

## Code source avec commentaires

|  |  |
| --- | --- |
| 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 | #include <AdafruitIO\_WiFi.h>  #include <Adafruit\_Sensor.h>  #include <Adafruit\_NeoPixel.h>  #include <DHT.h>  #include <DHT\_U.h>  #include <SPI.h>  #define IO\_USERNAME "Michael\_Nwg"  #define IO\_KEY "aio\_MiTf37vC7EtaAPboALYoJyncahkg"  #define WIFI\_SSID "Wifi-Abosi"  #define WIFI\_PASS "Nwogburu234"  #define RED\_PIN 4  #define GREEN\_PIN 5  #define BLUE\_PIN 2  #define LDR 39  #define RGB\_PIN 0  #define LED\_PIN 2  #define DATA\_PIN 26  #define Push\_Button1 12  *// RGB Var*  int couleurRed = 0;  int couleurBlue = 0;  int couleurGreen = 0;  *// LDR*  int LDR\_Value= 0;  AdafruitIO\_WiFi io (IO\_USERNAME,IO\_KEY,WIFI\_SSID, WIFI\_PASS);  DHT\_Unified dht(DATA\_PIN, DHT11);  Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(30, RGB\_PIN, NEO\_GRB + NEO\_KHZ800);  *// set up the feed*  AdafruitIO\_Feed \*RGB\_BLUE = io.feed("RGB\_BLUE");  AdafruitIO\_Feed \*RGB\_RED = io.feed("RGB\_RED");  AdafruitIO\_Feed \*RGB\_GREEN = io.feed("RGB\_GREEN");  AdafruitIO\_Feed \*LED = io.feed("LED");  AdafruitIO\_Feed \*Temperature = io.feed("temperature");  AdafruitIO\_Feed \*Humidite = io.feed("humidite");  AdafruitIO\_Feed \*luminosite = io.feed("luminosité");  AdafruitIO\_Feed \*TXT\_BOX = io.feed("TXT\_BOX");  *// Fonction de reception des données //*  void handle\_LED(AdafruitIO\_Data \*data) {  Serial.print("received <- ");  **if**(data->toPinLevel() == HIGH){  Serial.println("HIGH");  TXT\_BOX->save("Led: ON");  }    **else**{  Serial.println("LOW");  TXT\_BOX->save("LED: OFF");  }  *// write the current state to the led*  digitalWrite(LED\_PIN, data->toPinLevel());  }  void handleRGB\_BLUE(AdafruitIO\_Data \*data){  couleurBlue = data->toInt();  *// print RGB value*  Serial.print("B: <-" );  Serial.println(couleurBlue);  *// RGB Neopixel*  **for** (int i = 0; i < strip.numPixels(); i++)  {  strip.setPixelColor(i, strip.Color(couleurRed, couleurGreen, couleurBlue));  }  strip.show();  }  void handleRGB\_GREEN(AdafruitIO\_Data \*data){  couleurGreen = data->toInt();  *// print RGB value*  Serial.print("G: <-");  Serial.println(couleurGreen);  *// RGB Neopixel*  **for** (int i = 0; i < strip.numPixels(); i++)  {  strip.setPixelColor(i, strip.Color(couleurRed, couleurGreen, couleurBlue));  }  strip.show();  }  void handleRGB\_RED(AdafruitIO\_Data \*data){  couleurRed = data->toInt();  *// print RGB value*  Serial.print("R: <-");  Serial.println(couleurRed);  *// RGB Neopixel*  **for** (int i = 0; i < strip.numPixels(); i++)  {  strip.setPixelColor(i, strip.Color(couleurRed, couleurGreen, couleurBlue));  }  strip.show();  }  *// Fonction d'actualisation des données*  void Actualisation(){  sensors\_event\_t event;  dht.temperature().getEvent(&event);  float celsius = event.temperature; *// ver*  Serial.print("celsius: ");  Serial.print(celsius);  Serial.println("C");  Temperature->save(celsius);  dht.humidity().getEvent(&event);  Serial.print("humidité: ");  Serial.print(event.relative\_humidity);  Serial.println("%");  Humidite->save(event.relative\_humidity);  LDR\_Value = analogRead(LDR);  Serial.print("Luminosité -> ");  Serial.println(LDR\_Value);  luminosite->save(LDR\_Value);  }  void setup() {  pinMode(LED\_PIN, OUTPUT);  pinMode(Push\_Button1, INPUT\_PULLDOWN);  Serial.begin(115200);  **while**(!Serial);  dht.begin();  Serial.print("Connecting to Adafruit IO");  io.connect();  *// Création de message CALLBACK*  LED->onMessage(handle\_LED);  RGB\_BLUE->onMessage(handleRGB\_BLUE);  RGB\_RED->onMessage(handleRGB\_RED);  RGB\_GREEN->onMessage(handleRGB\_GREEN);  **while**(io.status() < AIO\_CONNECTED) {  Serial.print(".");  delay(500);  }  Serial.println();  Serial.println(io.statusText());    LED->get();  RGB\_BLUE->get();  RGB\_RED->get();  RGB\_GREEN->get();  }  void loop() {  io.run();  **if** (digitalRead(Push\_Button1) == HIGH){  Actualisation();  }  } |

## Analyse du code source

Le code est divisé en 4 parties :

1. Déclaration des données de connexion au wifi et à la plateforme adafruit IO
2. Déclaration des variables globales et des objets de classes.
3. Initialisation des feeds
4. Fonctions des réceptions et actualisation des données

### Déclaration des données de connexion au wifi et à la plateforme adafruit IO

La première partie du code consiste à d’abord inclure toutes les librairies nécessaires pour le bon fonctionnement du TP. Ceci se fait de **la ligne 2 à 7.**

Nous avons ensuite les variables qui seront utilisé pour la connexion au wifi et là connexion à la plateforme adafruit IO.

La connexion à la plateforme adafruit IO se fait grâce à un nom d’utilisateur « IO\_USERNAME » et un code « IO\_KEY ». Ceci se fait de **la ligne 9 à 12.**

### Déclaration des variables globales et des objets de classes

La deuxième partie du code consiste à déclarer les variables globales et à créer des instances de classes pour le DHT, ADAFRUIT\_IO, Neopixel.

Ceci se fait de **la ligne 16 à 36.**

### Initialisation des feeds

L’initialisation des feeds permet de faire un lien entre les feeds de la plateforme ADAFRUIT\_IO et l’esp32 après cela nous pourront demander ou envoyer des informations aux feeds qui seront par la suite affiché sur le Dashboard sous forme de widget. Ceci se fait de **la ligne 39 à 46.**

### Fonctions des réceptions et actualisation des données

La dernière partie du code contient les différentes fonctions qui seront appelé pour la réception des données et l’actualisation des données du Dashboard.

#### **Handle\_LED()**

La fonction « Handle\_LED()» sera appelé lorsque une donnée provenant du feed LED sera disponible la liaison entre le feed et la fonction « Handle\_LED()» se fait à la ligne 140.

Le feed «LED» contiendra l’état du bouton on/off du Dashboard une led sera ensuite allumé ou éteint en fonction de l’état du feed et pour finir l’état de la led sera affiché sur le Dashboard dans un TEXT BOX.

#### **HandleRGB\_BLUE() , HandleRGB\_GREEN() , HandleRGB\_RED()**

Les fonctions HandleRGB… aura pour but de mettre à jour l’état de la led RGB neopixel elle seront appelé lorsqu’il y’aura une variation des 3 sliders sur le Dashboard.

Chaque slider correspond à une couleur bleue, vert et rouge.

Le lien entre les feeds des sliders et des fonctions se fait de **la ligne 141 à 143.**

L’intensité de chaque couleur est modulable d’une valeur de 0 à 255.

La couleurs de la led neopixel sera mis a jour avec la méthode «SetPixelColor » en fonction de la données reçue.

#### **Actualisation ()**

## Lien software hardware

## Conclusion

# 5) TP5\_Blynk

## Objectif du TP

## Introduction

## Blynk

### Serveur

### Réalisation

## Code source avec commentaires

## Analyse du code source

## Lien software hardware

## Conclusion

# 6) TP6\_Nodered