Systèmes Embarqués II

NodeRed TP6

ISAT – EPHEC 2020-2021

Juan Alvarez et Olivier Grabenweger

2 Table des matières

1 Introduction

Pour ce TP, il était demandé de réaliser un travail en rapport avec le NodeMCU, dans notre cas nous avons utilisé un ESP32.

Tout d'abord, il est important pour nous de travailler sur des sujets que nous sentons proche de la réalité, c'est pourquoi nous avons décidé de prendre un scénario « réaliste » comme base pour ce tp.

Dont voici le scénario : Dans une maison un thermostat défaillant peine à chauffer les chambres correctement (soit trop chaud soit trop froid), les propriétaires décident donc d'ouvrir eux-mêmes les vannes des chauffages, un nouveau thermostat étant trop chère.

Pour ce faire, nous avons réalisé un petit montage avec un servo moteur et un dht11, et pour que les habitants puissent chauffer la maison si besoin avant d'arriver, nous avons connecté les ESP au wifi et avons réalisé une petite application sur blynk.

Pour évaluer le fonctionnement de l'installation et pour pouvoir garder les données indéfiniment nous avons enregistré les données d'humidité et température sur une google Sheet.

Blynk n'étant accessible que sur téléphone, nous avons réalisé un serveur web pour pouvoir également interagir avec les vannes et visualiser les données depuis un PC. Dans le travail effectué le serveur web est hébergé localement mais il est envisageable d'autoriser les connexions distantes sur l'installation domestique. Nous avons également profité du web serveur pour présenter notre travail, ainsi quelqu'un qui désire utiliser notre projet peut connecter l'ESP et à travers le serveur web, comprendre le fonctionnement.

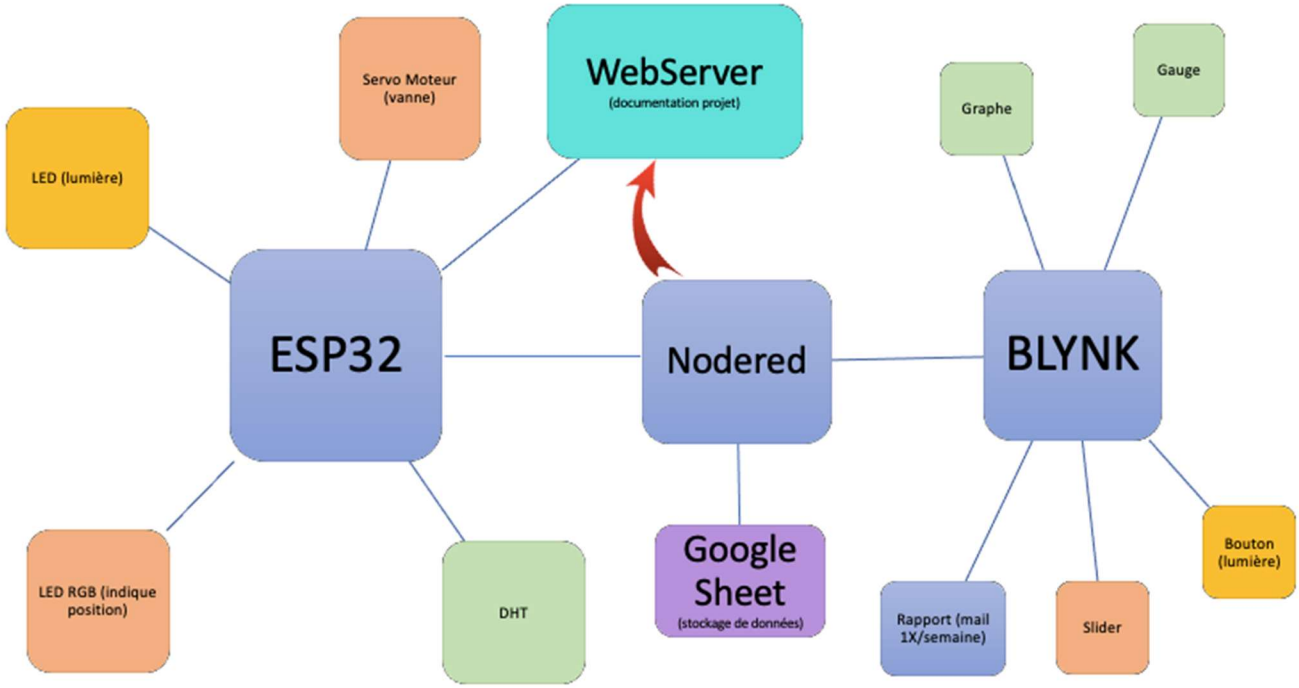
Pour faire le lien entre les différents éléments du projet, nous avons créé un flow sur NodeRed qui reçoit et envoie des données via Mqtt avec l'ESP32, et nous avons par l'intermédiaire de docker déployé un serveur Mosquitto comme broker.

Notons premièrement que nous avons déployé Nodered et Mosquitto depuis un ordinateur (c'est peu écologique de laisser un ordinateur allumé en continu pour réaliser des tâches aussi simplistes) mais il est tout à fait envisageable de déployer le tout depuis un Raspberry Pi, c'est d'ailleurs l'objectif à terme.

Enfin, il est utile de préciser également qu'en ouvrant les ports de notre réseaux domestique pour y accéder depuis l'extérieur, nous rendons ce dernier vulnérable, c'est un aspect à prendre en compte et c'est pourquoi nous n'avons pas poussé l'idée du web serveur plus loin, nous pensons que l'application de téléphone suffit et avons développé le serveur web comme un complément afin de présenter l'installation, son fonctionnement et les données.

2 Organigramme – Mindmapping

Structure du TP :



Légende couleur : Les cases portant la même couleur ont toutes un lien entre elles.

Détaillons un peu le fonctionnement, sur l'ESP se trouve :

- Une LED (placée comme « bonus » au cas où nous voudrions commander autre chose)
- Un DHT11 → capteur de température et d'humidité
- Une LED RGB qui indique l'état d'ouverture de la vanne avec une échelle de couleurs (vert → vanne 100% ouverte, jaune→60%, orange →30%, et enfin rouge →0% vanne totalement fermée).
- Un servo moteur → il permet l'ouverture de la vanne (ici nous utilisons un servo mais la course d'un servo étant en général limitée à 180° il est évident qu'il ne permet pas réellement l'ouverture d'une vanne, un Steppeur était plus indiqué mais n'en n'ayant pas, nous avons réalisé notre tp avec un servo comme exemple).

NodeRed fait le lien entre ESP et Blynk et il envoie également les données vers notre google sheet. Cette dernière n'a pas fait l'objet d'un travail poussé, elle est donc très simple c'est-à-dire en remplissant le rôle minimum à savoir écrire les données dans la bonne colonne. Néanmoins, nous pensons améliorer cet aspect puisqu'il s'agit de notre outil de diagnostic et qu'il permet de manipuler les données sur de longues périodes.

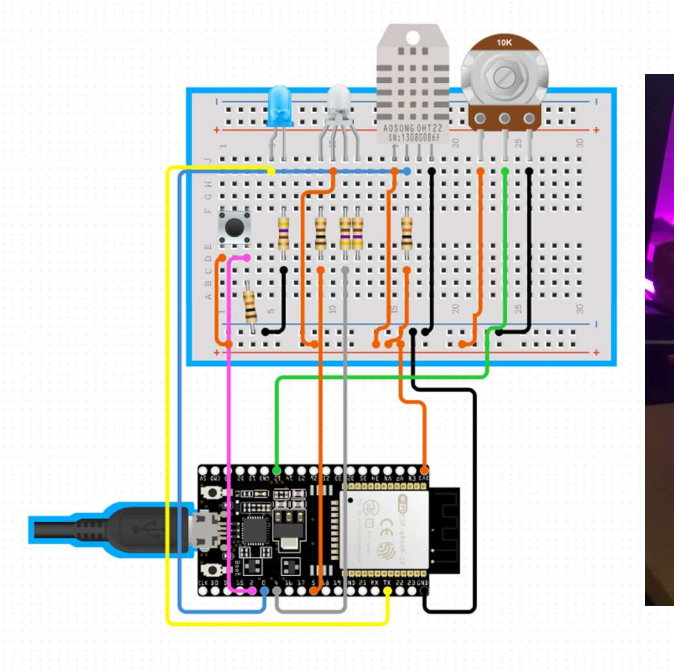
Enfin côté Blynk se trouve :

- 2 gauges et 2 graphes pour visualiser la température et l'humidité
- Deux boutons pour allumer et éteindre une LED (comme précisé plus haut, le but ici est de prévoir des sorties utilisables au cas où l'utilisateur souhaiterait ajouter des fonctionnalités)
- Un Slider pour contrôler l'ouverture de la vanne.
- Un générateur de rapport qui envoie un mail chaque semaine avec les chiffres de l'installation.

Finalement ajoutons simplement que le web serveur est déployé depuis l'ESP32 malgré qu'il était possible de le faire selon nous depuis Nodered. Cette dernière solution a été explorée mais nous sommes restés bloqués longtemps sur l'actualisation de cette dernière et faute de temps avons préféré le faire depuis l'ESP c'est malgré tout une piste qui reste à explorer.

3 Schéma de câblage

Nous utilisons le même montage que pour le tp4 par facilité en ajoutant le servo. (Le pot et le bouton ne sont pas utilisés).



4 Code source site Web

```
<!DOCTYPE html>
<html>
  <head>
    <title>ESP websever</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <meta name="viewport" content="width=device-width,initial-scale=1">
    <script src="https://code.highcharts.com/highcharts.js"></script>
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
    <style>
      h1 {font-family:monospace;text-align:center}
      h4 {font-family:cursive;font-style:italic}
      ul {font-family:tahoma;font-weight:normal;font-style:italic;
font-size:14px;line-height:20px}
      hr {border: 3px solid green;}
      p {font-family:tahoma}
      span {font-family:tahoma;line-height:35px}
      div {text-align:center}
      button {font-size:20px}
      .mySlides {display:none;}
    </style>
  </head>
  <body class="w3-animate-opacity">
    <!-- Sidebar -->
<div class="w3-sidebar w3-bar-block w3-card w3-animate-left"
style="display:none" id="mySidebar">
  <button onclick="w3_close()" class="w3-bar-item w3-large w3-hover-grey">
Close &times;</button>
  <a href="#infos" class="w3-bar-item w3-button w3-hover-grey">infos pratiques</a>
  <a href="#data" class="w3-bar-item w3-button w3-hover-grey">DATA</a>
```

```
30   <a href="#ill" class="w3-bar-item w3-button w3-hover-grey">Illustrations</a>
31 </div>
32 <!-- Page Content -->
33  <div id="main" style="text-align:left">
34 <div class="w3-blue-grey" style = "text-align:left">
35   <button class="w3-button w3-blue-grey w3-xlarge" onclick="w3_open()">☰</button>
36     <header class="w3-container w3-blue-grey">
37        <h1 class="w3-animate-top"> NodeRed Web Server </h1>
38     </header>
39   </div>
40 <script>
41 function w3_open() {
42   document.getElementById("main").style.marginLeft = "25%";
43   document.getElementById("mySidebar").style.width = "25%";
44   document.getElementById("mySidebar").style.display = "block";
45   document.getElementById("openNav").style.display = 'none';
46 }
47 function w3_close() {
48   document.getElementById("main").style.marginLeft = "0%";
49   document.getElementById("mySidebar").style.display = "none";
50   document.getElementById("openNav").style.display = "inline-block";
51 }
52 </script>
53
54     <h4 class="w3-animate-fading"> Par Juan Alvarez et Olivier Grabenweger</h4>
55     <h2 id="infos">Infos pratiques :</h2>
56     <ul>
57         <p>Gestion et visualisation des données de température et
58         d'humidité dans une pièce avec un esp32 en passant par NodeRed<p/>
59         <p>Pour ce faire nous avons déployé les objets suivants<p/>
60       <li>Une page web, développée par nous mêmes et basée sur le contenu
61         d'un précédent tp avec pour but de présenter le contenu du
62         tp et visualiser librement les données.
63       <li> Afin de déployer notre page web nous avons pensé le faire depuis
64         nodeRed et ce pour 2 raisons, la première car se faisant on peut allèger
65         le code, même en multi fichiers, ensuite si on venait à déconnecter
66         l'ESP la page web continuerais d'exister ce qui nous semble
67         être une approche plus pro du serveur web. Mais pour l'instant elle est
68         toujours déployée depuis l'esp32
69       <li> Pour faire communiquer notre ESP et notre NodeRed,
70         nous somme passés par un broker gratuit(mosquitto) que nous
71          avons déployé depuis docker également.
72       <li>Enfin afin d'avoir une interface accessible et intuitive
73         pour visualiser et interragir avec les objets installés dans la pièce
74         nous avons utilisé Blynk pour réaliser une appication de GSM basée
75         sur le travail fournit pour le TP3.
76       <li>Enfin afin d'avoir des données, nous avons conçu un système
77         qui contrôle la température et l'humidité. Pour interragir avec le
78         système nous avons installé un servo moteur sur la vanne de chauffage et
79         avons installé un témoins (led RGB)qui permet de visuellement confirmer
80         le degré d'ouverture de la vanne par un code couleur(fermé-->rouge,100%
81         ouvert-->vert, et des nuances allant de jaune vers orange)
82     </ul>
83     <br>
84     <h5 style="text-align:center">QR code pour arriver sur l'appli</h5>
85     <div class="w3-content w3-display-container">
86      <img src="*image en annexe*" style="width:40%;height:35%">
87    </div>
```

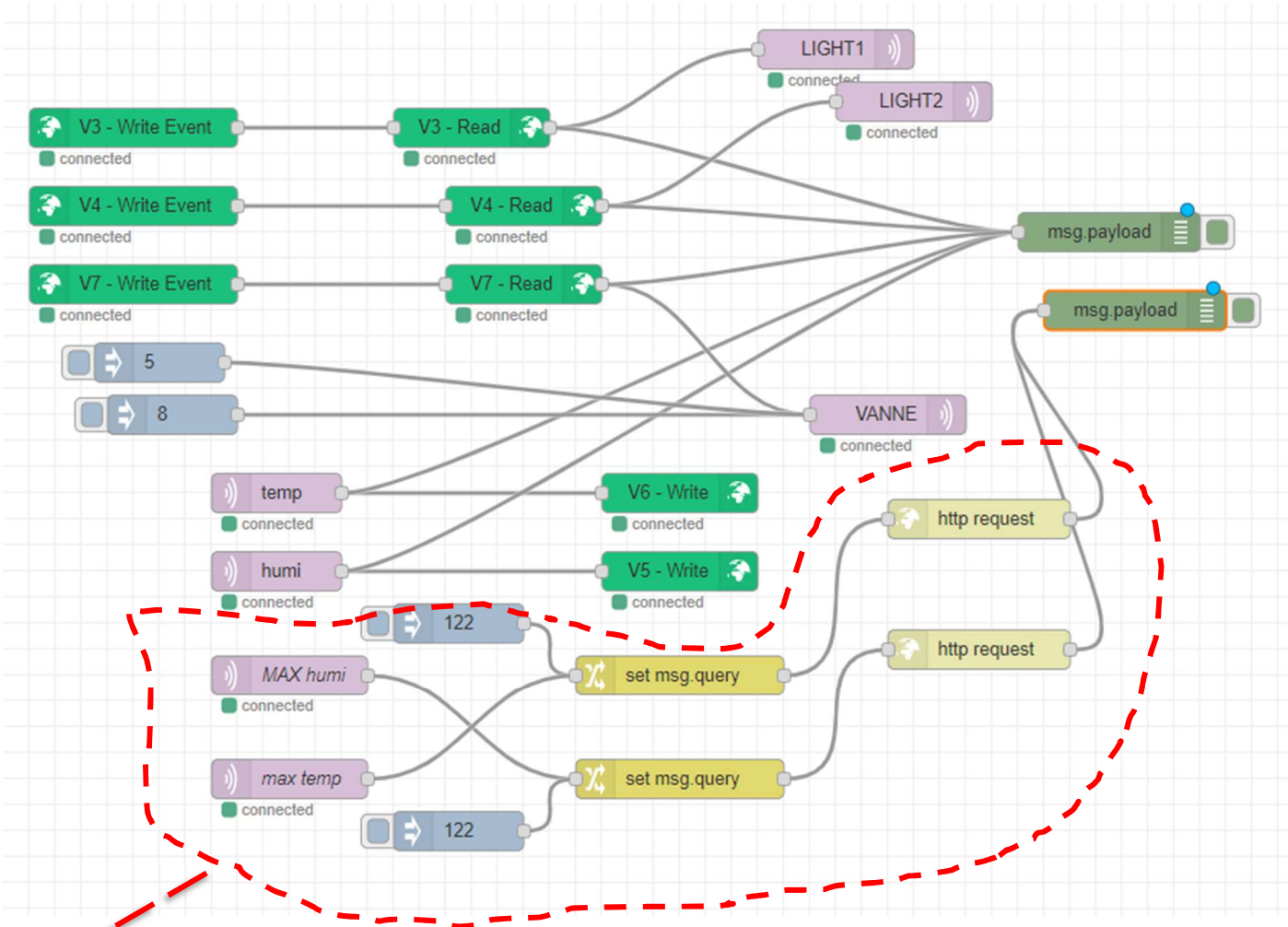
```
    <br>
    <fieldset class="w3-pale-yellow">
      <legend id="data" style = "text-align:center;font-size:25px">
DATA FEED</legend>
      <span style="size:20"> Temperature: </span>
<span id="capteur_t";style="color:green;size:20;"></span>
      <br>
      <span style="size:20;"> Humidity: </span>
<span id="capteur_h";style="color:green;size:20;"></span>
    </fieldset>
    <br>
    <br>
        <h2 style = "text-align:center;font-size:25px">
DATA DISPLAY </h2>
  <div id="chart-temperature" class="container"></div>
  <div id="chart-humidity" class="container"></div>
    <script>
var chartT = new Highcharts.Chart({
  chart:{ renderTo : 'chart-temperature' },
  title: { text: 'DHT11 Temperature' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    },
    series: { color: '#059e8a' }
  },
  xAxis: { type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Temperature (Celsius)' }
    //title: { text: 'Temperature (Fahrenheit)' }
  },
  credits: { enabled: false }
});
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("capteur_t").innerHTML = this.responseText;
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartT.series[0].data.length > 40) {
        chartT.series[0].addPoint([x, y], true, true, true);
      } else {
        chartT.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/temperature", true);
  xhttp.send();
}, 1000 ) ;
      var chartH = new Highcharts.Chart({
```

```
  chart:{ renderTo:'chart-humidity' },
  title: { text: 'DHT11 Humidity' },
  series: [{
    showInLegend: false,
    data: []
  }],
  plotOptions: {
    line: { animation: false,
      dataLabels: { enabled: true }
    }
  },
  xAxis: {
    type: 'datetime',
    dateTimeLabelFormats: { second: '%H:%M:%S' }
  },
  yAxis: {
    title: { text: 'Humidity (%)' }
  },
  credits: { enabled: false }
});
setInterval(function ( ) {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("capteur_h").innerHTML = this.responseText;
      var x = (new Date()).getTime(),
          y = parseFloat(this.responseText);
      //console.log(this.responseText);
      if(chartH.series[0].data.length > 40) {
        chartH.series[0].addPoint([x, y], true, true, true);
      }
      else {
        chartH.series[0].addPoint([x, y], true, false, true);
      }
    }
  };
  xhttp.open("GET", "/humidity", true);
  xhttp.send();
}, 1000 ) ;
    </script>
    <br>
    <br>
        <h2 id="ill"class="w3-center">Illustrations</h2>
    <div class="w3-content w3-section">
     <img class="mySlides" src="*image en annexe*" style="width:50%;height:60%">
     <img class="mySlides" src="*image en annexe*" style="width:20%;height:25%">
    </div>
    <script>
var myIndex = 0;
carousel();

function carousel() {
  var i;
  var x = document.getElementsByClassName("mySlides");
  for (i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
  myIndex++;
```

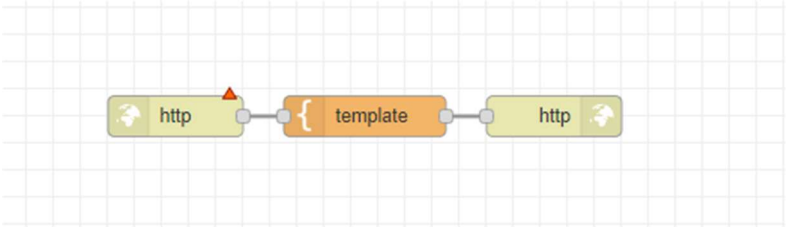
```
204   if (myIndex > x.length) {myIndex = 1}
205   x[myIndex-1].style.display = "block";
206   setTimeout(carousel, 2000); // Change image every 2 seconds
207 }
208 </script>
209     </div>
210     <footer class="w3-container w3-blue-grey w3-margin-top">
211         <h5> WebServer </h5>
212     </footer>
213   </body>
214 </html>
```

5 NodeRed



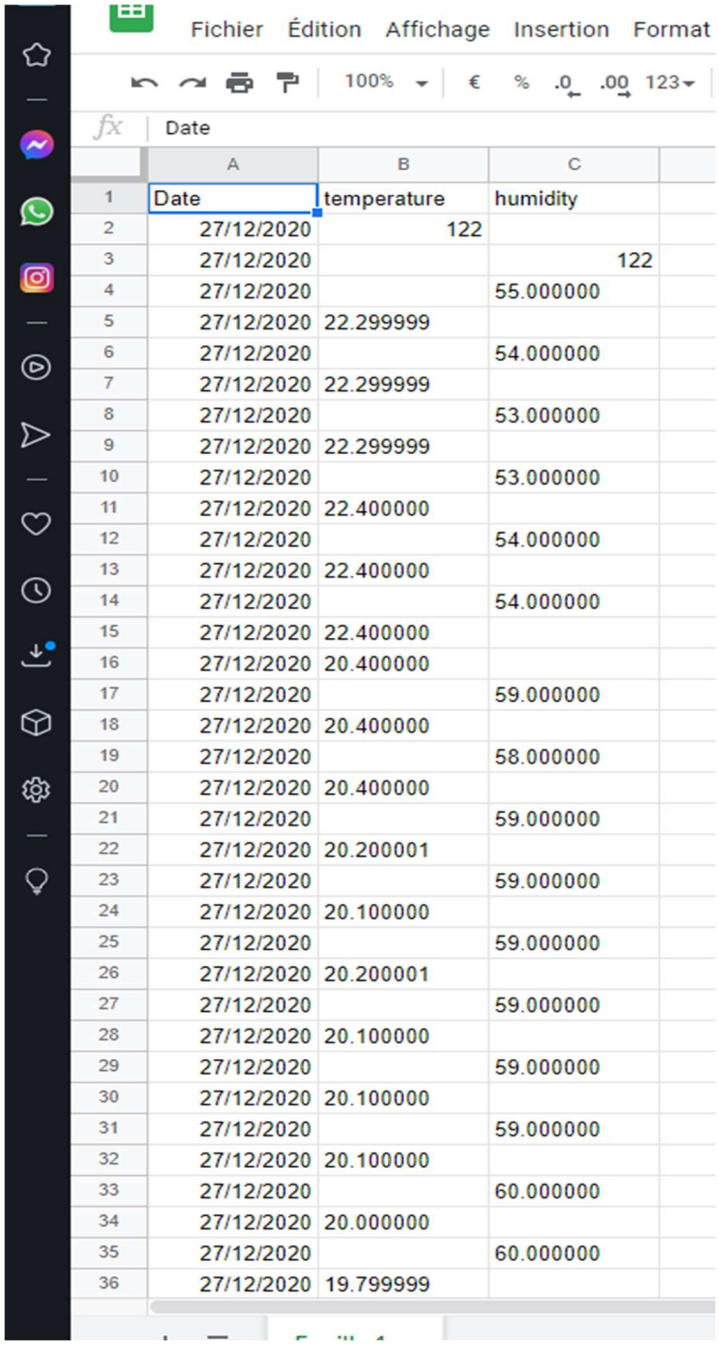
Gestion des données reçues via mqtt et envoyée vers google sheet

Si le site devait être déployé depuis Nodered nous ajouterions le nœud suivant :



Cependant étant donné que ne sommes pas parvenu à actualiser le site correctement, nous avons mis de côté cette idée.

Enfin, voici rapidement la google sheet, comme déjà dit, elle est vraiment très basique son seul but étant d'être présente. En effet, nous n'avons pas réellement approfondis son rôle dans notre projet, mais à terme l'objectif était de mettre en place un outil d'analyse des données sur le long terme.



	A	B	C
1	Date	temperature	humidity
2	27/12/2020	122	
3	27/12/2020		122
4	27/12/2020		55.000000
5	27/12/2020	22.299999	
6	27/12/2020		54.000000
7	27/12/2020	22.299999	
8	27/12/2020		53.000000
9	27/12/2020	22.299999	
10	27/12/2020		53.000000
11	27/12/2020	22.400000	
12	27/12/2020		54.000000
13	27/12/2020	22.400000	
14	27/12/2020		54.000000
15	27/12/2020	22.400000	
16	27/12/2020	20.400000	
17	27/12/2020		59.000000
18	27/12/2020	20.400000	
19	27/12/2020		58.000000
20	27/12/2020	20.400000	
21	27/12/2020		59.000000
22	27/12/2020	20.200001	
23	27/12/2020		59.000000
24	27/12/2020	20.100000	
25	27/12/2020		59.000000
26	27/12/2020	20.200001	
27	27/12/2020		59.000000
28	27/12/2020	20.100000	
29	27/12/2020		59.000000
30	27/12/2020	20.100000	
31	27/12/2020		59.000000
32	27/12/2020	20.100000	
33	27/12/2020		60.000000
34	27/12/2020	20.000000	
35	27/12/2020		60.000000
36	27/12/2020	19.799999	

L'affichage des données une fois sur deux provient du script de la feuille qui actualise les deux données à chaque requête.

Le but était de présenter une feuille fonctionnelle actualisée depuis nodered, mais il est évident que la forme de cette dernière reste à peaufiner. Mais comme évoqué précédemment, cela passera également par l'apprentissage du javascript.

6 Code source ESP32

```
/*****  Juan and Oli dev for Embedded Systems *****/
// Creating an MQTT application

#include <Arduino.h>

//#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <PubSubClient.h>   //NODERED
// #include <BlynkSimpleEsp32.h>
#include <SPIFFS.h>
#include <ESP32Servo.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

// Button Pin
#define BUTTON_PIN 5

// LED Pin
#define LED_PIN 13

// pin connected to DH11 data line
#define DHTPIN 4

#define servoPin 18
ESP32PWM pwm;
int freq = 1000;

// pins to rgb
#define RED_PIN   16
#define GREEN_PIN 17
#define BLUE_PIN  26

// dht type --> dht11
#define DHTTYPE    DHT11     // DHT 11

const char* mqtt_server ="192.168.0.10";    // mosquitto adress

#define WIFI_SSID "DESKTOP-JJORSV4 5823"
#define WIFI_PASS "salut123"

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE        (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
int count;
```

```
57
58 DHT dht(DHTPIN, DHTTYPE);
59
60 float h,t,f,maxtemp,maxhumi; // var dht
61 char send_buffer_temp [16];
62 char send_buffer_humi [16];
63 long send_time,send_time2;
64 // button state
65 bool btn_state = false;
66 bool prv_btn_state = false;
67
68 Servo myservo;  // create servo object to control a servo
69
70 //NODERED que faire quand un msg arrive
71 void callback(char* topic, byte* payload, unsigned int length) {
72   Serial.print("Message arrived [");
73   Serial.print(topic);
74   Serial.print("] ");
75   char msg[32];
76   for (int i = 0; i < length; i++) {
77     Serial.print((char)payload[i]);
78     msg[i]=payload[i];
79   }
80   Serial.println();
81   Serial.print(msg);
82   Serial.println();
83   memset(msg,0,sizeof(msg));
84   /***** Action en fonction du payload ******/
85
86   if ((char)payload[0] == '0') {
87     digitalWrite(LED_PIN, LOW);   // Turn the LED off
88   }
89   if ((char)payload[0] == '1')
90   {
91    digitalWrite(LED_PIN, HIGH);  // Turn the LED on
92   }
93   if ((char)payload[0] == '3')
94   {
95     digitalWrite(LED_PIN, LOW);   // Turn the LED off
96   }
97   if ((char)payload[0] == '4')
98   {
99     digitalWrite(LED_PIN, HIGH);   // Turn the LED on
100  }
101  if ((char)payload[0] == '5')
102  {
103    //ferme vanne et couleur rouge
104   myservo.write(0);
105    ledcWrite(1,0);
106    ledcWrite(2,255);
107    ledcWrite(3,255);
108  }
109  if ((char)payload[0] == '6')
110  {
111    //couleur orange
112    myservo.write(60);
113    ledcWrite(1,20);
114    ledcWrite(2,220);
```

```
115     ledcWrite(3,250);
116   }
117   if ((char)payload[0] == '7')
118   {
119     //jaune
120    myservo.write(120);
121     ledcWrite(1,0);
122     ledcWrite(2,128);
123     ledcWrite(3,255);
124   }
125   if ((char)payload[0] == '8')
126   {
127     //vert
128     myservo.write(180);
129     ledcWrite(1,255);
130     ledcWrite(2,0);
131     ledcWrite(3,255);
132   }
133   else
134   {
135     Serial.print("Unknown Payload");
136   }
137 }
138
139 void reconnect() {     //NODERED mqqt (re)connexion
140   // Loop until we're reconnected
141   while (!client.connected()) {
142     Serial.print("Attempting MQTT connection...");
143     // Create a random client ID
144     String clientId = "ESP32Client-";
145     clientId += String(random(0xffff), HEX);
146     // Attempt to connect
147     if (client.connect(clientId.c_str())) {
148       Serial.println("connected");
149       // ... and resubscribe
150       client.subscribe("LIGHT1");
151       client.subscribe("LIGHT2");
152       client.subscribe("VANNE");
153     } else {
154       Serial.print("failed, rc=");
155       Serial.print(client.state());
156       Serial.println(" try again in 5 seconds");
157       // Wait 5 seconds before retrying
158       delay(5000);
159     }
160   }
161 }
162
163 void setup(){
164   send_time=millis();
165   send_time2=millis();
166   pinMode(BUTTON_PIN, INPUT);
167
168   // set LED pin as an output
169   pinMode(LED_PIN, OUTPUT);
170  //------------------------------ ---Serial
171   Serial.begin(115200);
172
```

```
173   // Connect to Wi-Fi
174   WiFi.begin(WIFI_SSID, WIFI_PASS);
175   while (WiFi.status() != WL_CONNECTED) {
176     delay(1000);
177     Serial.println("Connecting to WiFi..");
178   }
179
180    //--------------------------------SPIFFS
181    // check for files and begin SPIFFS
182   if(!SPIFFS.begin())
183   {
184     Serial.println("Erreur SPIFFS...");
185     return;
186   }
187
188   File root = SPIFFS.open("/");
189   File file = root.openNextFile();
190
191   while(file)
192   {
193     Serial.print("File: ");
194     Serial.println(file.name());
195     file.close();
196     file = root.openNextFile();
197   }
198
199   Serial.println("Successfully Ended");
200
201
202 // RGB esp32
203   #if defined(ARDUINO_ARCH_ESP32) // ESP32 pinMode
204     // assign rgb pins to channels
205     ledcAttachPin(RED_PIN,1);
206     ledcAttachPin(GREEN_PIN,2);
207     ledcAttachPin(BLUE_PIN,3);
208
209     // init. channels
210     ledcSetup(1, 1000, 8);
211     ledcSetup(2, 1000, 8);
212     ledcSetup(3, 1000, 8);
213   #else
214     pinMode(RED_PIN, OUTPUT);
215     pinMode(GREEN_PIN, OUTPUT);
216     pinMode(BLUE_PIN, OUTPUT);
217   #endif
218
219
220 //  Allow allocation of all timers
221         ESP32PWM::allocateTimer(0);
222         ESP32PWM::allocateTimer(1);
223         ESP32PWM::allocateTimer(2);
224         ESP32PWM::allocateTimer(3);
225
226   //pwm.attachPin(APin, freq, 10); // 1KHz 8 bit
227
228   myservo.setPeriodHertz(50);    // standard 50 hz servo
229         myservo.attach(servoPin); // attaches the servo on pin 18 to the object
230   Serial.println(WiFi.localIP());
```

```
231
232   client.setServer(mqtt_server,1883);    //set server on 1883
233   client.setCallback(callback);         // set call  for callback
234
235   //----------------------------SERVER
236
237   server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
238   {
239       // Route for root / web page
240     request->send(SPIFFS, "/index.html","text/html");
241     //request->send(200, "text/html",index_html);
242   });
243   server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request){
244     //t = dht.readTemperature();
245     //Blynk.virtualWrite(V6, t);
246     memset(send_buffer_temp,'0',sizeof(send_buffer_temp));
247     sprintf(send_buffer_temp,"%f",t);
248     request->send(200, "text/plain",send_buffer_temp);
249   });
250   server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request){
251     //h = dht.readHumidity();
252     //Blynk.virtualWrite(V5, h);
253     memset(send_buffer_humi,'0',sizeof(send_buffer_humi));
254     sprintf(send_buffer_humi,"%f",h);
255     request->send(200, "text/plain",send_buffer_humi);
256   });
257
258   dht.begin();
259   //Blynk.begin(auth, WIFI_SSID, WIFI_PASS, IPAddress(193,190,65,122), 8080);
260   // Start server
261   server.begin();
262 }
263
264 void loop(){
265   //Blynk.run();
266   if (!client.connected()) {   //NODERED
267     reconnect();
268   }
269   client.loop();      //NODERED
270   if(send_time <= millis()){
271     h = dht.readHumidity();
272     // Read temperature as Celsius (the default)
273     t = dht.readTemperature();
274     // Read temperature as Fahrenheit (isFahrenheit = true)
275     f = dht.readTemperature(true);
276     // Check if any reads failed and exit early (to try again).
277     if (isnan(h) || isnan(t) || isnan(f)) {
278       Serial.println(F("Failed to read from DHT sensor!"));
279       //return;
280     }
281     // Compute heat index in Fahrenheit (the default)
282     float hif = dht.computeHeatIndex(f, h);
283     // Compute heat index in Celsius (isFahreheit = false)
284     float hic = dht.computeHeatIndex(t, h, false);
285
286     Serial.print(F("Humidity: "));
287     Serial.print(h);
288     Serial.print(F("%  Temperature: "));
```

```
289     Serial.print(t);
290     Serial.print(F("°C "));
291     Serial.print(f);
292     Serial.print(F("°F  Heat index: "));
293     Serial.print(hic);
294     Serial.print(F("°C "));
295     Serial.print(hif);
296     Serial.println(F("°F"));
297 
298       // save the btn_state state to the 'button' feed on adafruit io
299 
300     snprintf (msg, MSG_BUFFER_SIZE, "%f", t);
301     Serial.print("Publish message: ");
302     Serial.println(msg);
303     client.publish("temp", msg);
304 
305     // save the btn_state state to the 'button' feed on adafruit io
306     snprintf (msg, MSG_BUFFER_SIZE, "%f", h);
307     Serial.print("Publish message: ");
308     Serial.println(msg);
309     client.publish("humi", msg);
310 
311     //memorise max temp and humi to be send to db
312   if(maxtemp<t){
313       maxtemp = t;
314     }
315   if(maxhumi<h){
316       maxhumi = h;
317     }
318     send_time = millis() + 10000;  // set next time you want to do anything
319   }
320   if(send_time2 <= millis()){
321     if(count==0){
322       Serial.println("print to dataBase");
323       snprintf (msg, MSG_BUFFER_SIZE, "%f", maxtemp);
324       client.publish("DBT",msg);
325       maxtemp=0;
326       }
327     else if (count==1)
328     {
329       Serial.println("print to dataBase");
330       snprintf (msg, MSG_BUFFER_SIZE, "%f", maxhumi);
331       client.publish("DBH",msg);
332       maxhumi=0;
333     }
334     send_time2 = millis() + 60000;
335     count +=1;
336     if (count ==2) count=0;
337     }
338 }
```

7 Conclusion

A travers ce TP, nous avons pu explorer Nodered, et il en ressort que malgré le fait de n'avoir pu consacrer que quelques heures à son apprentissage, il apparait comme un outil d'une efficacité évidente.
En effet, ce qui demande un programme assez long en temps normal ne demande que quelques blocs sur Nodered à l'image du Google Sheet.

Le ressentit est proche de celui après la découverte de Labview, dans le sens où il rend accessible des projets qui sont dans l'absolu bien plus difficiles à réaliser. Mais il y a ici un outil qui est selon nous un cran plus haut de par sa flexibilité, en effet nous pouvons, pour peu qu'on maitrise Nodejs, écrire nos propres fonctions/blocs. Ce à quoi s'ajoute le fait de pouvoir interconnecter tout un tas de services, qui pour nous représente une force du programme.

Nous regrettons d'une part notre faible maitrise du js, qui nous a limité dans nos créations, et qui représente clairement un sujet que nous voulons approfondir par la suite. Ensuite, une mauvaise organisation de notre temps nous a poussé à clôturer le projet avant d'avoir pu intégrer toutes nos idées.

Néanmoins l'expérience reste plus que positive et nous espérons être amenés vers ce genre d'applications dans le futur que ce soit au cours de nos stages ou dans notre vie professionnelle.

8 Annexes, bibliographie et illustrations

a) Annexes

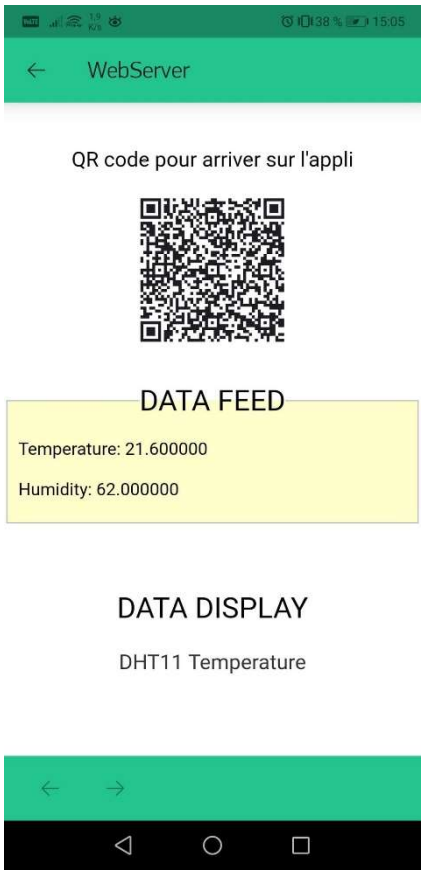
Les liens des 3 images sortis du code html car la mise en page Word était affectée à cause de leur longueur.

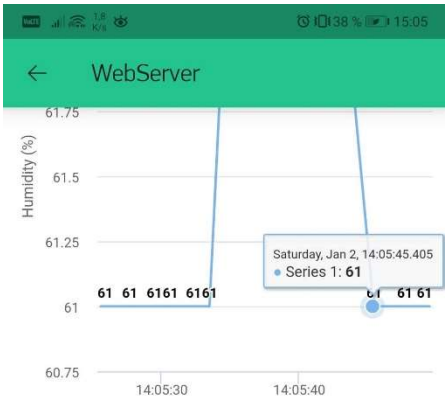
https://scontent.fbru5-1.fna.fbcdn.net/v/t1.15752-9/132680314_430948934773714_6821396418431504269_n.jpg?_nc_cat=100&ccb=2&_nc_sid=ae9488&_nc_ohc=zYDrnN_bHZcAX9DQaff&_nc_ht=scontent.fbru5-1.fna&oh=49ad5c06212f195e823c9ce77805ca69&oe=60098CE1

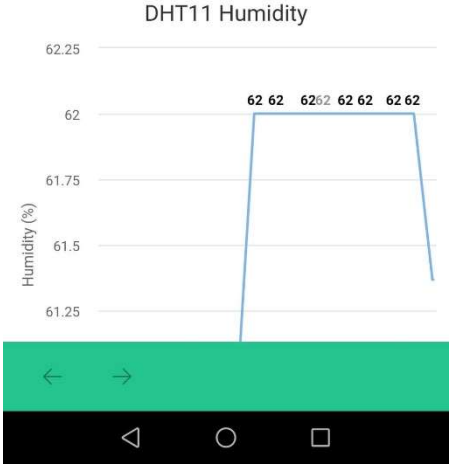
https://scontent.fbru5-1.fna.fbcdn.net/v/t1.15752-9/132290689_390433512246324_3831939363267429479_n.jpg?_nc_cat=109&ccb=2&_nc_sid=ae9488&_nc_ohc=KUElGZuw2y4AX_GZZSB&_nc_ht=scontent.fbru5-1.fna&oh=30be909b4e908f4daa53aee3263e3444&oe=600784DD

https://scontent-bru2-1.xx.fbcdn.net/v/t1.15752-9/133031592_900521113821783_6859392920115411230_n.png?_nc_cat=104&ccb=2&_nc_sid=ae9488&_nc_ohc=a8qe5MRndEcAX-_HleL&_nc_ht=scontent-bru2-1.xx&oh=bef962a216ec3a808d60440b25197867&oe=6016DA77

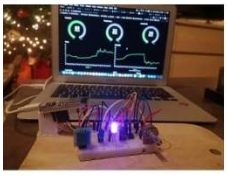
Illustrations du site

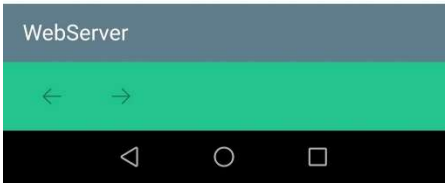




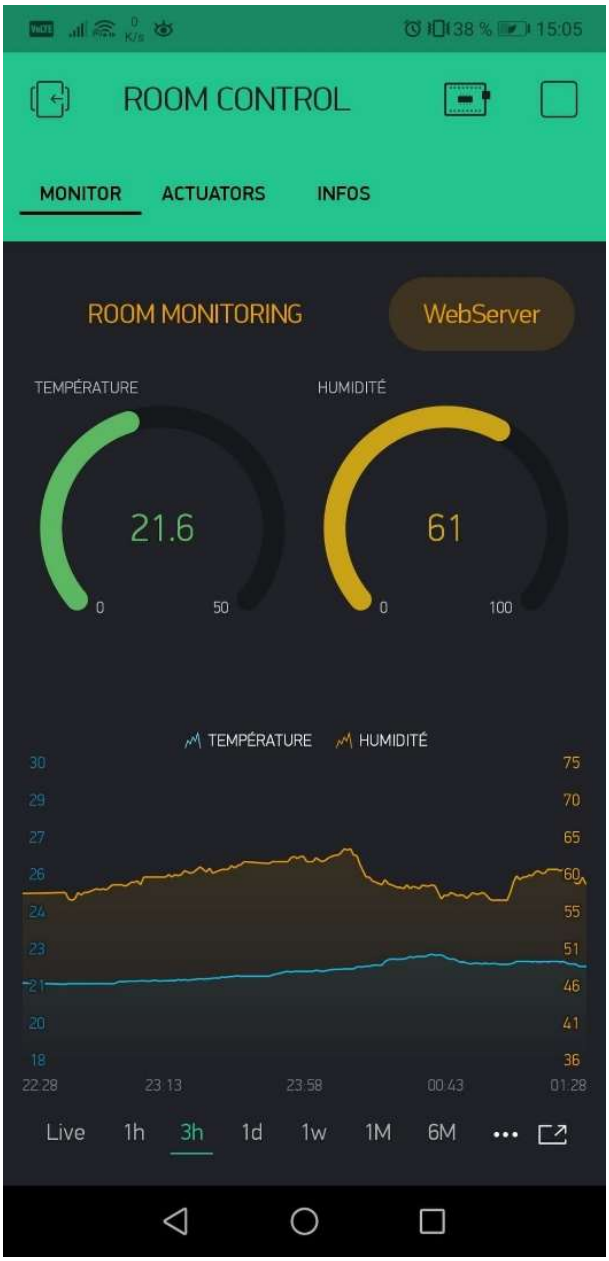


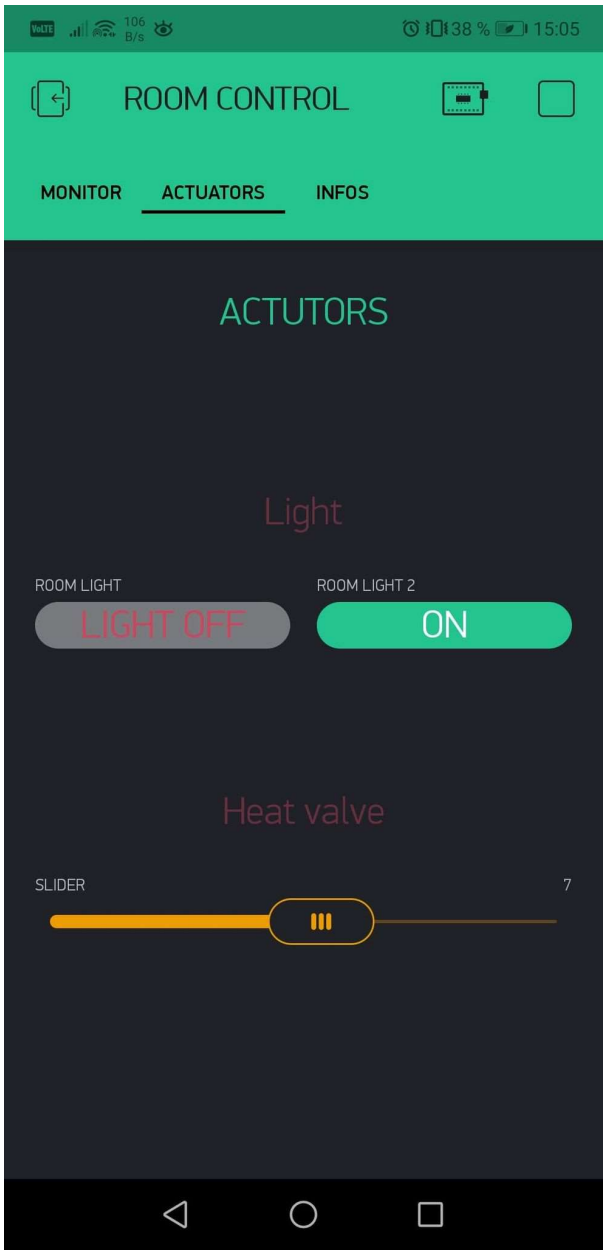
Illustrations

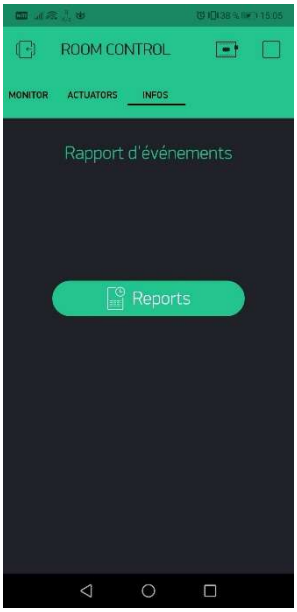




Illustrations appli







⇨ Génération de rapports hebdo

b) **Bibliographie**

- https://www.w3schools.com/w3css/
- https://hub.docker.com
- https://nodered.org/docs/getting-started/docker
- https://blynk.io/en/getting-started
- https://nodemcu.readthedocs.io/en/dev-esp32/modules/ledc/
- https://www.arduinolibraries.info/libraries/esp32-servo
- https://en.wikipedia.org/wiki/MQTT
- https://www.youtube.com/watch?v=Elxdz-2rhLs
- https://mosquitto.org
- Et divers autres liens et vidéos entre aurtre parmi les ressources sur moodle