

|  |
| --- |
| **Systèmes Embarqués II** |
| NodeRed TP6 |
| ISAT – EPHEC 2020-2021  Juan Alvarez et Olivier Grabenweger |

Table des matières

[1 Introduction 2](#_Toc60854902)

[2 Organigramme – Mindmapping 3](#_Toc60854903)

[3 Schéma de câblage 4](#_Toc60854904)

[4 Code source site Web 4](#_Toc60854905)

[5 NodeRed 8](#_Toc60854906)

[6 Code source ESP32 10](#_Toc60854907)

[7 Conclusion 16](#_Toc60854908)

[8 Annexes, bibliographie et illustrations 16](#_Toc60854909)

[a) Annexes 16](#_Toc60854910)

[b) Bibliographie 19](#_Toc60854911)

# 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.

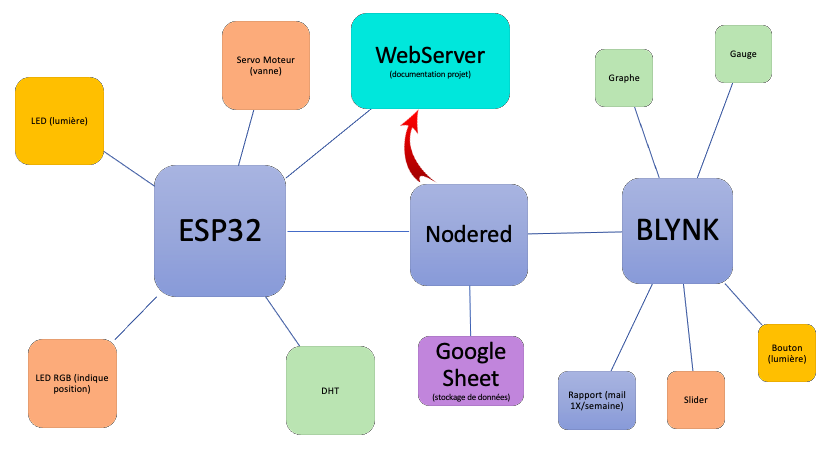
Pour faire le lien entre les différents éléments du projet, nous avons créé un flow sur NodeRed qui reçoit et envoie des données via Mqtt avec l’ESP32, et nous avons par l‘intermédiaire de docker déployé un serveur Mosquitto comme broker.

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.

# 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 à capteur de température et d’humidité
* Une LED RGB qui indique l’état d’ouverture de la vanne avec une échelle de couleurs (vert à vanne 100% ouverte, jauneà60%, orange à30%, et enfin rouge à0% vanne totalement fermée).
* Un servo moteur à 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.

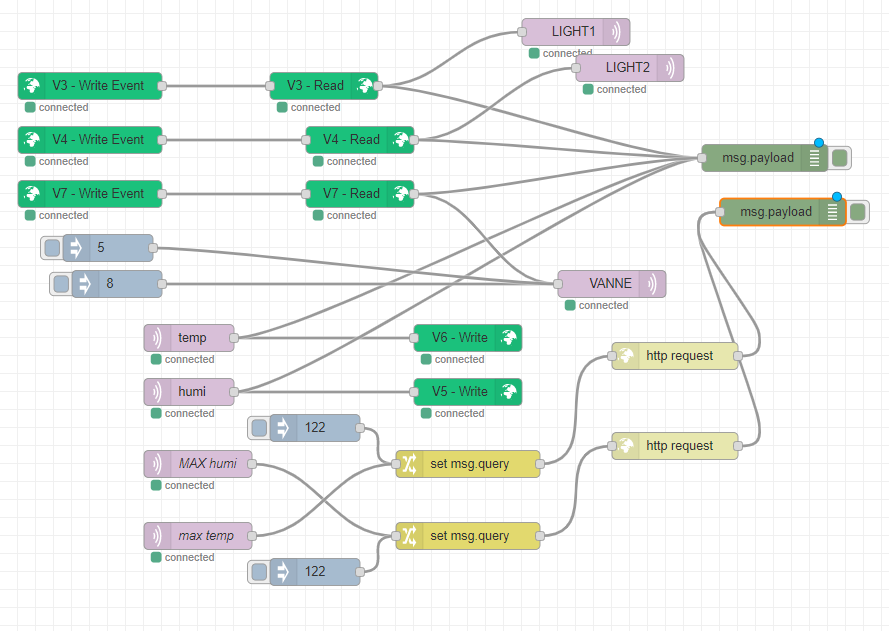
# Schéma de câblage

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

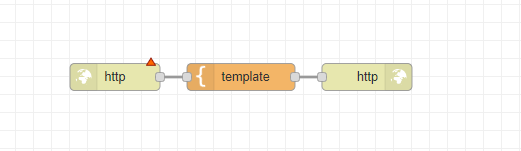
# Code source site Web

|  |  |
| --- | --- |
| 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 | <!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>  <a href="#ill" class="w3-bar-item w3-button w3-hover-grey">Illustrations</a>  </div>  <!-- Page Content -->  <div id="main" style="text-align:left">  <div class="w3-blue-grey" style = "text-align:left">  <button class="w3-button w3-blue-grey w3-xlarge" onclick="w3\_open()">☰</button>  <header class="w3-container w3-blue-grey">  <h1 class="w3-animate-top"> NodeRed Web Server </h1>  </header>  </div>  <script>  **function** w3\_open() {  document.getElementById("main").style.marginLeft = "25%";  document.getElementById("mySidebar").style.width = "25%";  document.getElementById("mySidebar").style.display = "block";  document.getElementById("openNav").style.display = 'none';  }  **function** w3\_close() {  document.getElementById("main").style.marginLeft = "0%";  document.getElementById("mySidebar").style.display = "none";  document.getElementById("openNav").style.display = "inline-block";  }  </script>  <h4 class="w3-animate-fading"> Par Juan Alvarez et Olivier Grabenweger</h4>  <h2 id="infos">Infos pratiques :</h2>  <ul>  <p>Gestion et visualisation des données de température et  d'humidité dans une pièce avec un esp32 en passant par NodeRed<p/>  <p>Pour ce faire nous avons déployé les objets suivants<p/>  <li>Une page web, développée par nous mêmes et basée sur le contenu  d'un précédent tp avec pour but de présenter le contenu du  tp et visualiser librement les données.  <li> Afin de déployer notre page web nous avons pensé le faire depuis  nodeRed et ce pour 2 raisons, la première car se faisant on peut allèger  le code, même en multi fichiers, ensuite si on venait à déconnecter  l'ESP la page web continuerais d'exister ce qui nous semble  être une approche plus pro du serveur web. Mais pour l'instant elle est  toujours déployée depuis l'esp32  <li> Pour faire communiquer notre ESP et notre NodeRed,  nous somme passés par un broker gratuit(mosquitto) que nous  avons déployé depuis docker également.  <li>Enfin afin d'avoir une interface accessible et intuitive  pour visualiser et interragir avec les objets installés dans la pièce  nous avons utilisé Blynk pour réaliser une appication de GSM basée  sur le travail fournit pour le TP3.  <li>Enfin afin d'avoir des données, nous avons conçu un système  qui contrôle la température et l'humidité. Pour interragir avec le  système nous avons installé un servo moteur sur la vanne de chauffage et  avons installé un témoins (led RGB)qui permet de visuellement confirmer  le degré d'ouverture de la vanne par un code couleur(fermé-->rouge,100%  ouvert-->vert, et des nuances allant de jaune vers orange)  </ul>  <br>  <h5 style="text-align:center">QR code pour arriver sur l'appli</h5>  <div class="w3-content w3-display-container">  <img src="\*image en annexe\*"⁩ style="width:40%;height:35%">  </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++;  **if** (myIndex > x.length) {myIndex = **1**}  x[myIndex-**1**].style.display = "block";  setTimeout(carousel, **2000**); // Change image every 2 seconds  }  </script>  </div>  <footer class="w3-container w3-blue-grey w3-margin-top">  <h5> WebServer </h5>  </footer>  </body>  </html> |

# NodeRed

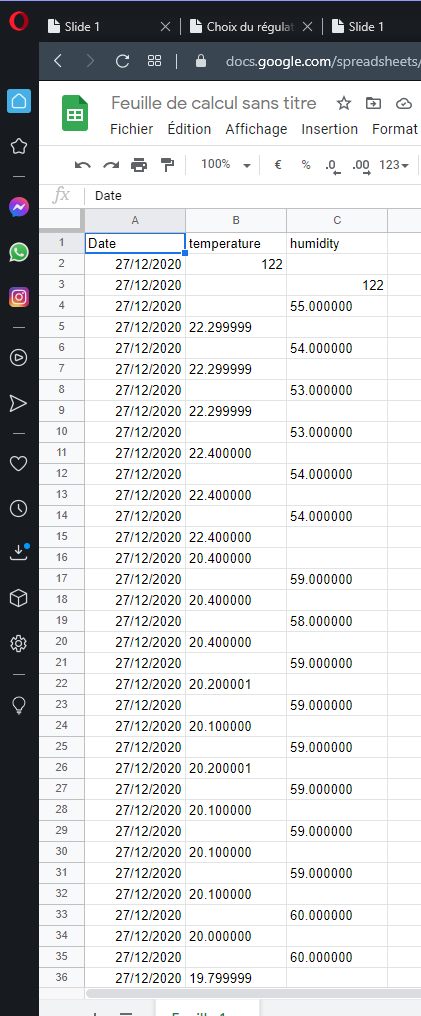


Gestion des données reçues via mqtt et envoyée vers google sheet

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.



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.

# Code source ESP32

|  |  |
| --- | --- |
| 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  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338 | /\*\*\*\*\* 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;  DHT **dht**(DHTPIN, DHTTYPE);  **float** h,t,f,maxtemp,maxhumi; // var dht  **char** send\_buffer\_temp [**16**];  **char** send\_buffer\_humi [**16**];  **long** send\_time,send\_time2;  // button state  **bool** btn\_state = false;  **bool** prv\_btn\_state = false;  Servo myservo; // create servo object to control a servo  //NODERED que faire quand un msg arrive  **void** **callback**(**char**\* topic, byte\* payload, **unsigned** **int** length) {  Serial.print("Message arrived [");  Serial.print(topic);  Serial.print("] ");  **char** msg[**32**];  **for** (**int** i = **0**; i < length; i++) {  Serial.print((**char**)payload[i]);  msg[i]=payload[i];  }  Serial.println();  Serial.print(msg);  Serial.println();  memset(msg,**0**,**sizeof**(msg));  /\*\*\*\*\* Action en fonction du payload \*\*\*\*\*\*/  **if** ((**char**)payload[**0**] == '0') {  digitalWrite(LED\_PIN, LOW); // Turn the LED off  }  **if** ((**char**)payload[**0**] == '1')  {  digitalWrite(LED\_PIN, HIGH); // Turn the LED on  }  **if** ((**char**)payload[**0**] == '3')  {  digitalWrite(LED\_PIN, LOW); // Turn the LED off  }  **if** ((**char**)payload[**0**] == '4')  {  digitalWrite(LED\_PIN, HIGH); // Turn the LED on  }  **if** ((**char**)payload[**0**] == '5')  {  //ferme vanne et couleur rouge  myservo.write(**0**);  ledcWrite(**1**,**0**);  ledcWrite(**2**,**255**);  ledcWrite(**3**,**255**);  }  **if** ((**char**)payload[**0**] == '6')  {  //couleur orange  myservo.write(**60**);  ledcWrite(**1**,**20**);  ledcWrite(**2**,**220**);  ledcWrite(**3**,**250**);  }  **if** ((**char**)payload[**0**] == '7')  {  //jaune  myservo.write(**120**);  ledcWrite(**1**,**0**);  ledcWrite(**2**,**128**);  ledcWrite(**3**,**255**);  }  **if** ((**char**)payload[**0**] == '8')  {  //vert  myservo.write(**180**);  ledcWrite(**1**,**255**);  ledcWrite(**2**,**0**);  ledcWrite(**3**,**255**);  }  **else**  {  Serial.print("Unknown Payload");  }  }  **void** **reconnect**() { //NODERED mqqt (re)connexion  // Loop until we're reconnected  **while** (!client.connected()) {  Serial.print("Attempting MQTT connection...");  // Create a random client ID  String clientId = "ESP32Client-";  clientId += String(random(**0xffff**), HEX);  // Attempt to connect  **if** (client.connect(clientId.c\_str())) {  Serial.println("connected");  // ... and resubscribe  client.subscribe("LIGHT1");  client.subscribe("LIGHT2");  client.subscribe("VANNE");  } **else** {  Serial.print("failed, rc=");  Serial.print(client.state());  Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying  delay(**5000**);  }  }  }  **void** **setup**(){  send\_time=millis();  send\_time2=millis();  pinMode(BUTTON\_PIN, INPUT);  // set LED pin as an output  pinMode(LED\_PIN, OUTPUT);  //------------------------------- ---Serial  Serial.begin(**115200**);  // Connect to Wi-Fi  WiFi.begin(WIFI\_SSID, WIFI\_PASS);  **while** (WiFi.status() != WL\_CONNECTED) {  delay(**1000**);  Serial.println("Connecting to WiFi..");  }  //-------------------------------SPIFFS  // check for files and begin SPIFFS  **if**(!SPIFFS.begin())  {  Serial.println("Erreur SPIFFS...");  **return**;  }  File root = SPIFFS.open("/");  File file = root.openNextFile();  **while**(file)  {  Serial.print("File: ");  Serial.println(file.name());  file.close();  file = root.openNextFile();  }  Serial.println("Successfully Ended");  // RGB esp32  #**if** defined(ARDUINO\_ARCH\_ESP32) // ESP32 pinMode  // assign rgb pins to channels  ledcAttachPin(RED\_PIN,**1**);  ledcAttachPin(GREEN\_PIN,**2**);  ledcAttachPin(BLUE\_PIN,**3**);  // 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  // Allow allocation of all timers  ESP32PWM::allocateTimer(**0**);  ESP32PWM::allocateTimer(**1**);  ESP32PWM::allocateTimer(**2**);  ESP32PWM::allocateTimer(**3**);    //pwm.attachPin(APin, freq, 10); // 1KHz 8 bit  myservo.setPeriodHertz(**50**); // standard 50 hz servo  myservo.attach(servoPin); // attaches the servo on pin 18 to the object  Serial.println(WiFi.localIP());  client.setServer(mqtt\_server,**1883**); //set server on 1883  client.setCallback(callback); // set call for callback  //----------------------------SERVER  server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  // Route for root / web page  request->send(SPIFFS, "/index.html","text/html");  //request->send(200, "text/html",index\_html);  });  server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){  //t = dht.readTemperature();  //Blynk.virtualWrite(V6, t);  memset(send\_buffer\_temp,'0',**sizeof**(send\_buffer\_temp));  sprintf(send\_buffer\_temp,"%f",t);  request->send(**200**, "text/plain",send\_buffer\_temp);  });  server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){  //h = dht.readHumidity();  //Blynk.virtualWrite(V5, h);  memset(send\_buffer\_humi,'0',**sizeof**(send\_buffer\_humi));  sprintf(send\_buffer\_humi,"%f",h);  request->send(**200**, "text/plain",send\_buffer\_humi);  });  dht.begin();  //Blynk.begin(auth, WIFI\_SSID, WIFI\_PASS, IPAddress(193,190,65,122), 8080);  // Start server  server.begin();  }    **void** **loop**(){  //Blynk.run();  **if** (!client.connected()) { //NODERED  reconnect();  }  client.loop(); //NODERED  **if**(send\_time <= millis()){  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.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 'button' feed on adafruit io  snprintf (msg, MSG\_BUFFER\_SIZE, "%f", t);  Serial.print("Publish message: ");  Serial.println(msg);  client.publish("temp", msg);    // save the btn\_state state to the 'button' feed on adafruit io  snprintf (msg, MSG\_BUFFER\_SIZE, "%f", h);  Serial.print("Publish message: ");  Serial.println(msg);  client.publish("humi", msg);    //memorise max temp and humi to be send to db  **if**(maxtemp<t){  maxtemp = t;  }  **if**(maxhumi<h){  maxhumi = h;  }  send\_time = millis() + **10000**; // set next time you want to do anything  }  **if**(send\_time2 <= millis()){  **if**(count==**0**){  Serial.println("print to dataBase");  snprintf (msg, MSG\_BUFFER\_SIZE, "%f", maxtemp);  client.publish("DBT",msg);  maxtemp=**0**;  }  **else** **if** (count==**1**)  {  Serial.println("print to dataBase");  snprintf (msg, MSG\_BUFFER\_SIZE, "%f", maxhumi);  client.publish("DBH",msg);  maxhumi=**0**;  }  send\_time2 = millis() + **60000**;  count +=**1**;  **if** (count ==**2**) count=**0**;  }  } |

# 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.

# Annexes, bibliographie et illustrations

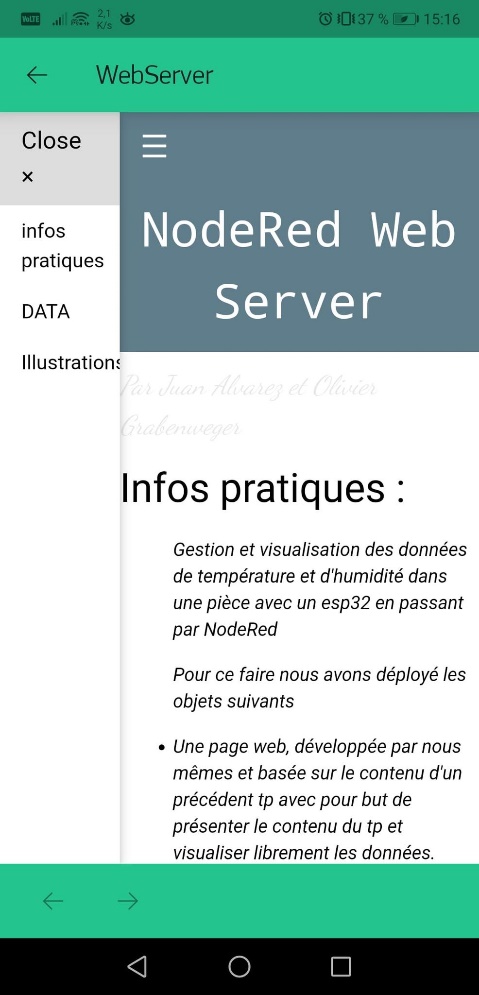
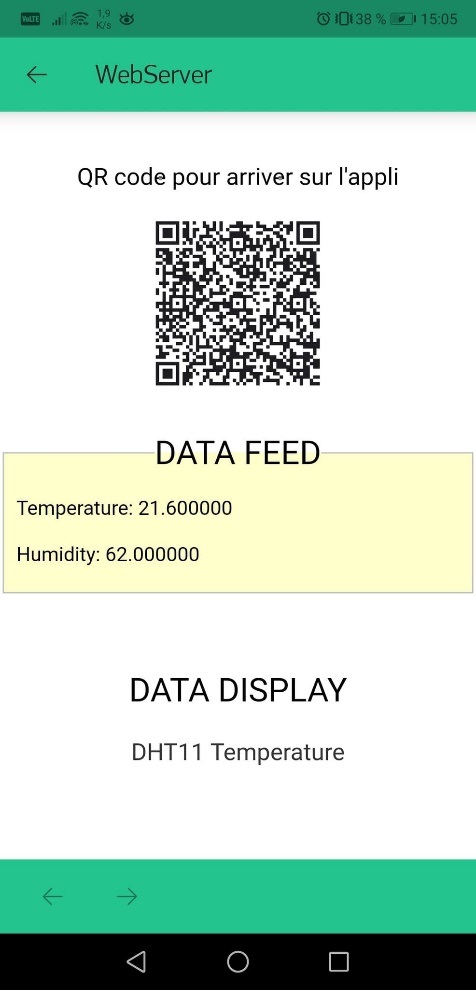
## Annexes

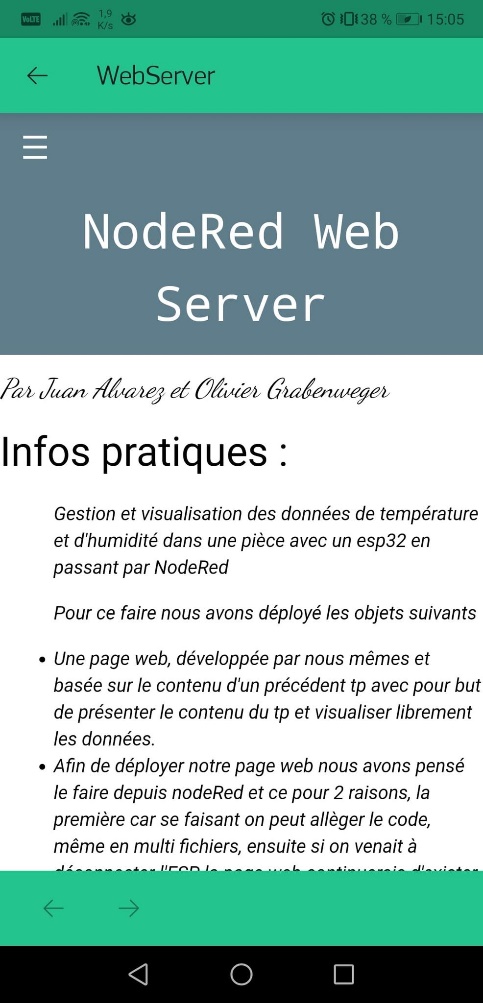
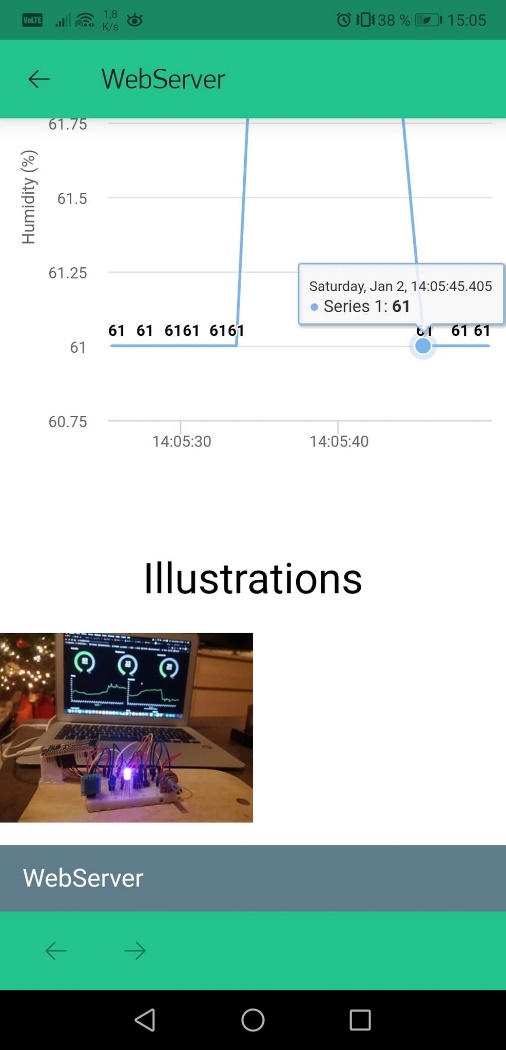
Les liens des 3 images sortis du code html car la mise en page Word était affectée à cause de leur longueur.

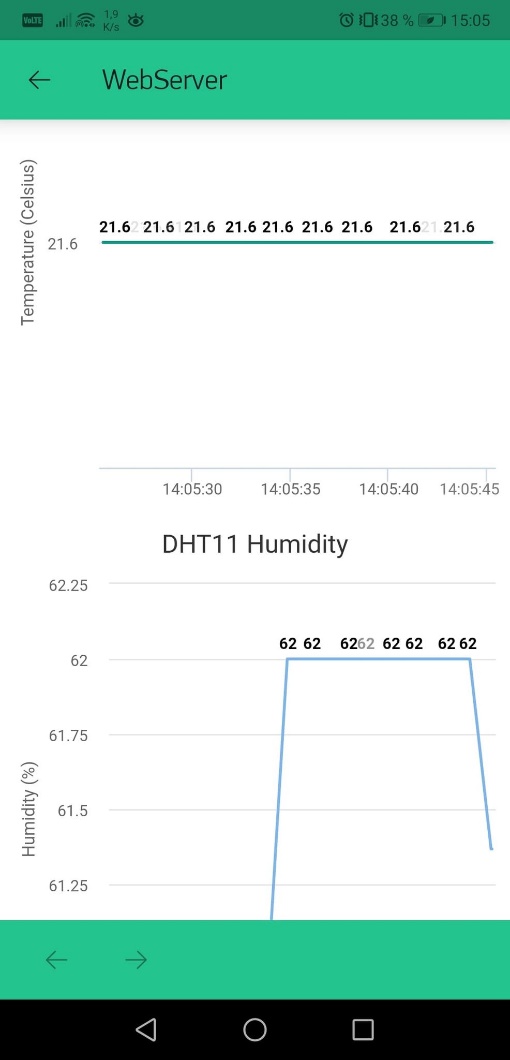
<https://scontent.fbru5-1.fna.fbcdn.net/v/t1.15752-9/132680314_430948934773714_6821396418431504269_n.jpg?_nc_cat=100&ccb=2&_nc_sid=ae9488&_nc_ohc=zYDrnN_bHZcAX9DQaff&_nc_ht=scontent.fbru5-1.fna&oh=49ad5c06212f195e823c9ce77805ca69&oe=60098CE1>

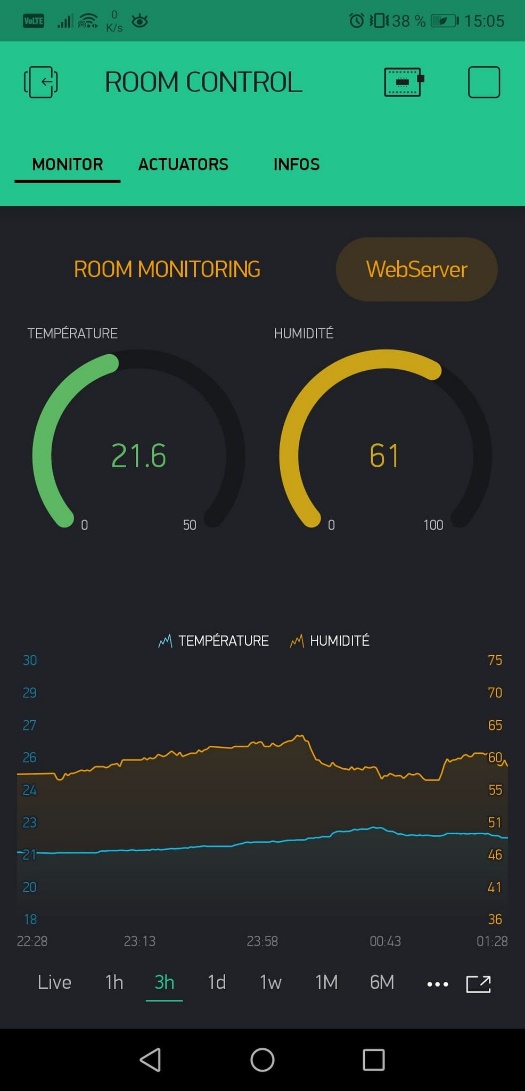
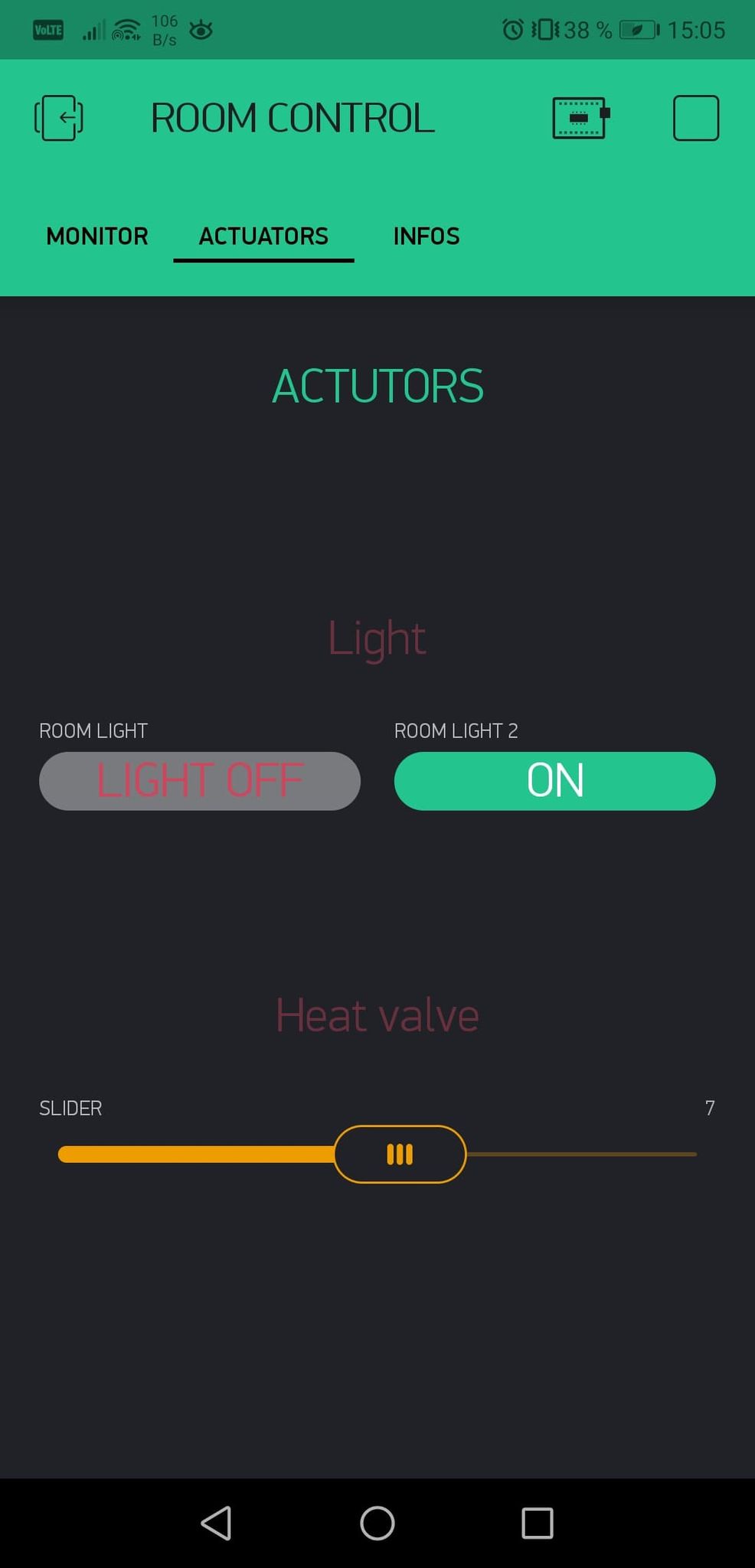
<https://scontent.fbru5-1.fna.fbcdn.net/v/t1.15752-9/132290689_390433512246324_3831939363267429479_n.jpg?_nc_cat=109&ccb=2&_nc_sid=ae9488&_nc_ohc=KUEIGZuw2y4AX_GZZSB&_nc_ht=scontent.fbru5-1.fna&oh=30be909b4e908f4daa53aee3263e3444&oe=600784DD>

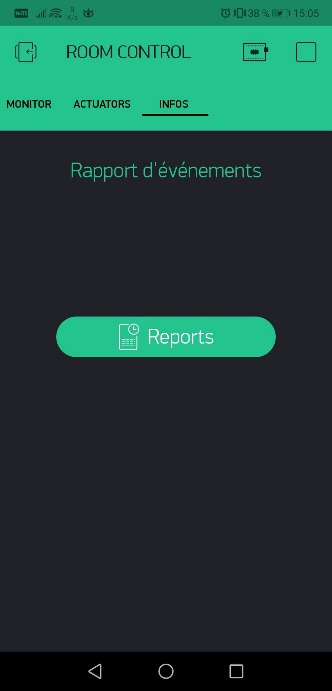
<https://scontent-bru2-1.xx.fbcdn.net/v/t1.15752-9/133031592_900521113821783_6859392920115411230_n.png?_nc_cat=104&ccb=2&_nc_sid=ae9488&_nc_ohc=a8qe5MRndEcAX-_HleL&_nc_ht=scontent-bru2-1.xx&oh=bef962a216ec3a808d60440b25197867&oe=6016DA77>

Illustrations du site





Illustrations appli

* Génération de rapports hebdo

## 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=EIxdz-2rhLs>
* https://mosquitto.org
* Et divers autres liens et vidéos entre aurtre parmi les ressources sur moodle