

|  |
| --- |
| **Systèmes Embarqués II** |
| Serveur Web Asynchrone |
| ISAT – EPHEC 2020-2021  Juan Alvarez et Olivier Grabenweger |

Table des matières

[1 Introduction 2](#_Toc60833454)

[2 Organigramme – Mindmapping 2](#_Toc60833455)

[3 Schéma de câblage 3](#_Toc60833456)

[4 Code source Site Web 3](#_Toc60833457)

[5 Code source ESP 6](#_Toc60833458)

[6 Conclusion 11](#_Toc60833459)

[7 Annexes, bibliographie et illustrations 11](#_Toc60833460)

[a) Annexes 11](#_Toc60833461)

[b) Bibliographie 11](#_Toc60833462)

# Introduction

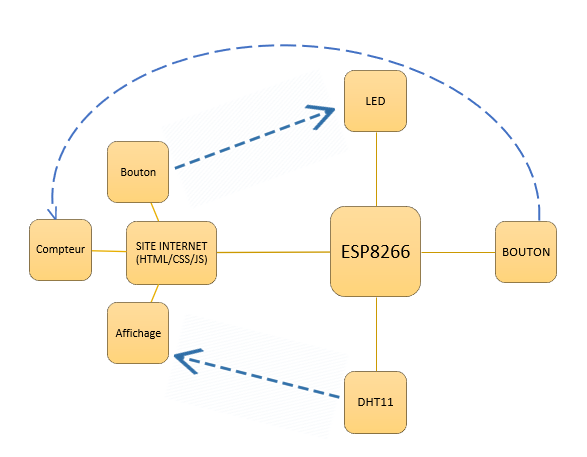
Pour ce TP, il était demandé de réaliser un serveur Web asynchrone (qui s’actualise en continu sans devoir recharger la page), afin de visualiser les données d’un capteur de température et d’interagir avec le NodMCU (ESP8266).

Dans un premier temps, nous avons réalisé un serveur Web dans lequel une recharge de la page était indispensable pour actualiser les données. Mais cela nous a semblé peu pratique et peu esthétique.

Nous avions également réalisé une première version du site web avec le code HTML intégré au code C mais par la suite nous nous sommes dirigés vers une approche avec plusieurs fichiers.   
C’est cette dernière approche que nous allons vous présenter dans ce rapport.

# Organigramme – Mindmapping

Structure du TP :

Echange d’informations entre un ESP et un site internet hébergé localement

|  |  |
| --- | --- |
| **Site** | **ESP** |
| On visualise les données du capteur et par un appui on allume une LED sur l’ESP.  Les données s’actualisent de manière automatique. | On déploie le serveur Web et par l’appui d’un bouton physique on incrémente un compteur qui s’affiche sur le site. |

# Schéma de câblage

Le logiciel utilisé (Circuito) ne nous a pas permis de personnaliser les pins, cependant nous avons tout de même illustré le câblage avec ce dernier.

# Code source Site Web

Ce TP était notre première approche du HTML/CSS et nous savions qu’il existait des exemples déjà tout fait mais nous voulions éditer nous même du code et avons donc pris l’initiative de créer le site entièrement.  
Nous avons utilisé la bibliothèque CSS : « W3.css ».

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | <!DOCTYPE HTML>  <html>  <head>  <title>ESP8266 websever</title>  <meta name="viewport" content="width=device-width,  initial-scale=1"charset="UTF-8" />  <link rel="stylesheet" type="text/css" href="w3.css">  <script type="text/javascript" src="script.js"></script>  <style>  a**:link**, a**:visited** { **background-color**: **#f44336**;  **color**: white;**padding**: **15px** **25px**; **text-align**: **center**;  **text-decoration**: **none**; **display**: **inline**-**block**;}  a**:hover**, a**:active** { **background-color**:red; }  h1 {**text-align**:**center**;**text-decoration**:**underline**}  h4 {**font-weight**:**normal**;**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**}  img {**width**: **20**%; **height**: **20**%;}  </style>  </head>  <body>  <h1> ESP8266 Web Server </h1>  <h4> Par Juan Alvarez et Olivier Grabenweger</h2>  <img src="Onizuka.jpg" alt="GTO" align="center" border=3>  <h2>Ennoncé :</h2>  <ul>  <li>Commander un appareil à distance via une page web  <li> Afficher l'état d'un bouton  <li> Afficher en temps réel les mesures d'un capteur  </ul>  <br>  <fieldset>  <legend style = "text-align:center;font-size:25px">  Data display</legend>  <span style="color:blue;size:20;text-align: center;">  Temperature: </span><span id="capteur\_t";  style="color:green;size:20;"></span>  <br>  <span style="color:blue;size:20;text-align: center;">  Humidity: </span>  <span id="capteur\_h";style="color:green;size:20;">  </span>  <br>  <span style="color:green;size:20;text-align: center;">  Button State: </span>  <span id="state\_b";style="color:green;size:20;">  </span>  <br>  <span style="color:green;size:20;text-align: center;">  The button has been pressed : </span> <span id="count";  style="color:green;size:20;"></span>  <span style="color:green;size:20;">\_times</span>  <br>  <span style="text-align: center;"> LED status : </span>  <span id="state\_LED"></span>  </fieldset>  <br>  <p style = "text-align:center"> Switch LED state </p>  <hr>  <br>  <div>  <button onclick="ON()">LED ON</button>  <p2 style="color:white;"> ceci est du texte </p2>  <button onclick="OFF()">LED OFF</button>  </div>  <br>  <hr style = "border: 1px solid green;">  <br>  <p style = "text-align:center"> Data Requests </p>  <hr>  </body>  </html>  Nous avons rédigé quelques fonctions en JAVAscript afin d’actualiser les données de la page.  La première partie est commentée car elle était utilisée lorsque nous avions fait le site non asynchrone mais par la suite, nous avons changé d’approche.  Il nous semblait néanmoins pertinent de laisser le code étant donné qu’il permet de visualiser le travail qui a été effectué.   |  |  | | --- | --- | | 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 | /\*function callServer(url,cFunction)  {  var xhttp = new XMLHttpRequest();  xhttp.onreadystatechange = function()  {  if (this.readyState == 4 && this.status == 200)  {  cFunction(this);  }  };  xhttp.open("GET", url, 1);  xhttp.send();  }  function handleResponse(xhttp)  {  document.getElementById("state\_LED").innerHTML = "LED" + xhttp.responseText;  }  function handleResponse2(xhttp)  {  document.getElementById("state\_b").innerHTML = xhttp.responseText;  }  setInterval(function handleResponse3(xhttp)  {  document.getElementById("capteur\_t").innerHTML = xhttp.responseText;  }, 2000);  setInterval(function handleResponse4(xhttp)  {  document.getElementById("capteur\_h").innerHTML = xhttp.responseText;  }, 2000);\*/  //////////////////////////////////////////////////////////////////////  **function** callServer(ID,url)  {  **var** xhttp = **new** XMLHttpRequest();  xhttp.onreadystatechange = **function**()  {  **if**(**this**.readyState == **4** && **this**.status == **200**)  {  document.getElementById(ID).innerHTML = **this**.responseText;  }  };  xhttp.open("GET", url, **true**);  xhttp.send();  }  **function** ON() {  callServer("state\_LED","LEDisON")  }  **function** OFF() {  callServer("state\_LED", "LEDisOFF")  }  setInterval(**function** getData()  {  callServer("capteur\_h","humidity")  callServer("capteur\_t","temperature")  callServer("state\_b","bp")  }, **2000**);  setInterval(**function** getData()  {  callServer("count","compteur")  }, **500**); | |

# Code source ESP

|  |  |
| --- | --- |
| 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 | #include <arduino.h>  #include <ESP8266WiFi.h>  #include <WiFiClient.h>  #include <Adafruit\_Sensor.h>  #include <DHT.h>  #include <ESPAsyncWebServer.h>  #include "LittleFS.h"  #ifndef STASSID  #define STASSID "bbox-Sophie1"  #define STAPSK "20150509Sophi"  #endif  #define DHTPIN 12 // Digital pin connected to the DHT sensor  #define DHTTYPE DHT11 // DHT 11  DHT **dht**(DHTPIN, DHTTYPE);  // current temperature & humidity, updated in loop()  **float** t = **0.0**;  **float** h = **0.0**;  **unsigned** **long** previousMillis = **0**; // will store last time DHT was updated  // Updates DHT readings every 10 seconds  **const** **long** interval = **10000**;  **const** **int** BUTTON = D0;  **const** **int** LEDred = D1;  **const** **int** LEDblue = D2;  **int** state\_button = **0**;  **int** flag\_button = **0**;  **int** count = **0**;  **const** **char**\* ssid = STASSID;  **const** **char**\* password = STAPSK;  **bool** flag;  **char** str\_temp[**16**];  **char** str\_hum[**16**];  **char** count\_txt[**10**];  String mybutton;  AsyncWebServer **server**(**80**);  **void** **temp**() {  sprintf(str\_temp, "%f", t);  }  **void** **humi**() {  sprintf(str\_hum, "%f", h);  }  **void** **bp**() {  **if** (digitalRead(BUTTON) == LOW)  {  mybutton = "bouton\_actif";  }  **else**  {  mybutton = "bouton\_non\_actif";  }  }  **void** **Compteur**() {  sprintf(count\_txt, "%d", count);  }  **void** **button\_Read**()  {  state\_button = digitalRead(BUTTON);  **if** (state\_button == LOW)  {  flag\_button = **1**;  digitalWrite(LEDblue, HIGH);  }  **else**  {  digitalWrite(LEDblue, LOW);  }  **if** (flag\_button == **1** && state\_button == HIGH)  {  count += **1**;  flag\_button = **0**;  }  }  **void** **LEDisON**() {  digitalWrite(LEDred, **1**);  }  **void** **LEDisOFF**() {  digitalWrite(LEDred, **0**);  }  **void** **handleNotFound**(AsyncWebServerRequest \*request) {  request->send(**404**, "text/plain", "Not found");  }  // Replaces placeholder with DHT values  String **processor**(**const** String& var) {  //Serial.println(var);  **if** (var == "TEMPERATURE") {  **return** String(t);  }  **else** **if** (var == "HUMIDITY") {  **return** String(h);  }  **return** String();  }  **void** **setup**() {  pinMode(LEDred, OUTPUT); // Initialize the LED pin as an output  digitalWrite(LEDred, LOW);  pinMode(LEDblue, OUTPUT); // Initialize the LED pin as an output  digitalWrite(LEDblue, LOW);  pinMode(BUTTON, INPUT);  Serial.begin(**115200**);  dht.begin();  WiFi.mode(WIFI\_STA);  WiFi.begin(ssid, password);  Serial.println("");  **if**(!LittleFS.begin())  {  Serial.println("Erreur LittleFS...");  **return**;  }  File root = LittleFS.open("/", "r");  File file = root.openNextFile();  **while**(file)  {  Serial.print("File: ");  Serial.println(file.name());  file.close();  file = root.openNextFile();  }  // Wait for connection  **while** (WiFi.status() != WL\_CONNECTED) {  delay(**500**);  Serial.print(".");  }  Serial.println("");  Serial.print("Connected to ");  Serial.println(ssid);  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  request->send(LittleFS, "/index.html", "text/html");  });  server.on("/w3.css", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  request->send(LittleFS, "/w3.css", "text/css");  });  server.on("/script.js", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  request->send(LittleFS, "/script.js", "text/javascript");  });  server.on("/Onizuka.jpg", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  request->send(LittleFS, "/Onizuka.jpg", "image/jpg");  });  server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  humi();  request->send(**200**, "text/plain", str\_hum);  });  server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  temp();  request->send(**200**, "text/plain", str\_temp);  });  server.on("/LEDisON", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  LEDisON();  request->send(**200**,"text/plain", " is on");  });  server.on("/LEDisOFF", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  LEDisOFF();  request->send(**200**,"text/plain", " is off");  });  server.on("/bp", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  bp();  request->send(**200**, "text/plain", mybutton);  });  server.on("/compteur", HTTP\_GET, [](AsyncWebServerRequest \*request)  {  Compteur();  request->send(**200**, "text/plain", count\_txt);  });  server.onNotFound(handleNotFound);  server.begin();  Serial.println("HTTP server started");  }  **void** **loop**() {  **unsigned** **long** currentMillis = millis();  **if** (currentMillis - previousMillis >= interval) {  // save the last time you updated the DHT values  previousMillis = currentMillis;  // Read temperature as Celsius (the default)  **float** newH = dht.readTemperature();  // Read temperature as Fahrenheit (isFahrenheit = true)  //float newT = dht.readTemperature(true);  // if temperature read failed, don't change t value  **if** (isnan(newH)) {  Serial.println("Failed to read from DHT sensor!");  }  **else** {  h = newH;  Serial.println(t);  }  // Read Humidity  **float** newT = dht.readHumidity();  // if humidity read failed, don't change h value  **if** (isnan(newT)) {  Serial.println("Failed to read from DHT sensor!");  }  **else** {  t = newT;  Serial.println(h);  }  }  button\_Read();  } |

# Conclusion

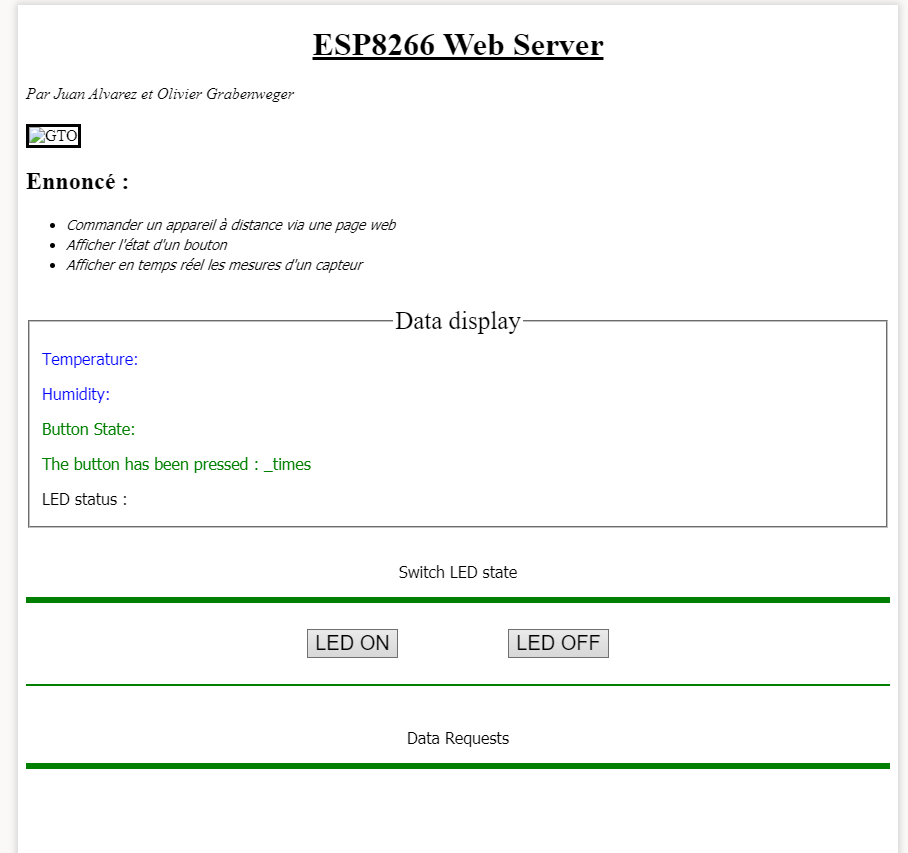
Ce travail nous a permis une première approche de la programmation web. C’était l’occasion de découvrir et de travailler avec des langages de programmation que nous ne sommes pas amenés à apprendre durant notre cursus.

C’était un TP dont la difficulté ne résidait pas dans la complexité de la programmation mais dans la manipulation de nouveaux outils, c’est pourquoi les programmes abordés reste relativement simples (allumage d’une LED) mais les différents outils utilisés nous ont permis d’étoffer notre gamme de connaissances.

Nous n’avons pas rencontré de difficultés particulières dans la réalisation du TP et avons pris plaisir à travailler avec de nouveaux outils mais regrettons tout de même de ne pas avoir pu aller plus loin dans la complexité du travail que nous avons fourni.

# Annexes, bibliographie et illustrations

## Annexes

Le site réalisé ici est assez basique. Mais il a le mérite d’avoir été construit de A à Z, une version améliorée de ce site sera proposée dans un prochain TP.

## Bibliographie

* <https://www.circuito.io/> : schéma de câblage
* <https://www.youtube.com/channel/UCe3v5cVACw-5BKQOcwUaM8w>
* <https://randomnerdtutorials.com>
* <https://www.w3schools.com/w3css/>
* <https://www.w3schools.com/html/default.asp>
* Documentation fournie sur Moodle et autres vidéos.