Systèmes Embarqués II

Serveur Web Asynchrone

ISAT – EPHEC 2020-2021

Juan Alvarez et Olivier Grabenweger

2 Table des matières

1 Introduction 2

2 Organigramme – Mindmapping 2

3 Schéma de câblage 3

4 Code source Site Web 3

5 Code source ESP 6

6 Conclusion 11

7 Annexes, bibliographie et illustrations 11

a) Annexes 11

b) Bibliographie 11

...

1 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.

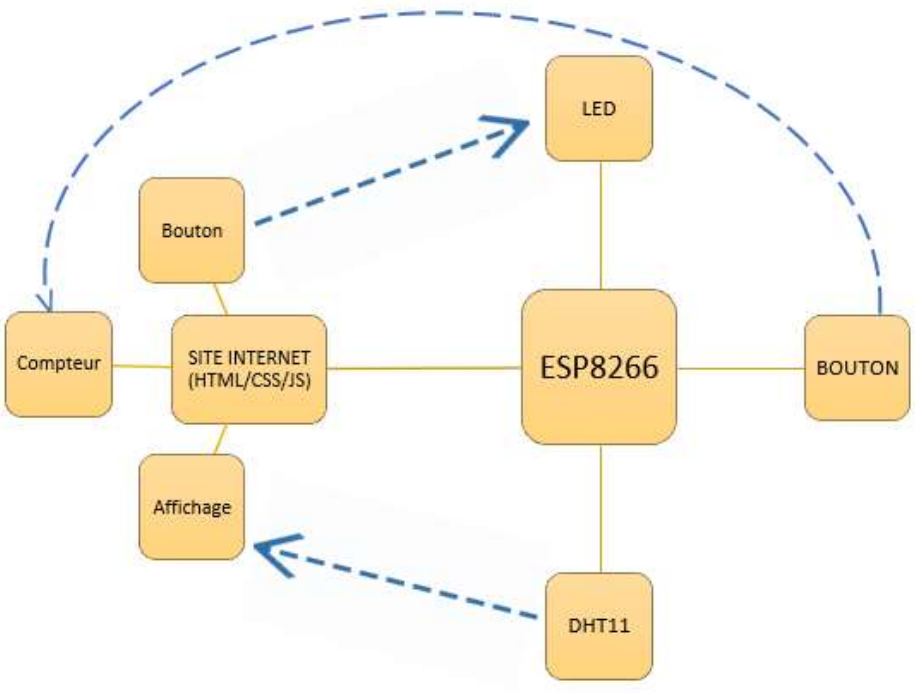
Nous avions également réalisé une première version du site web avec le code HTML intégré au code C mais par la suite nous nous sommes dirigés vers une approche avec plusieurs fichiers.
C'est cette dernière approche que nous allons vous présenter dans ce rapport.

2 Organigramme – Mindmapping

Structure du TP :

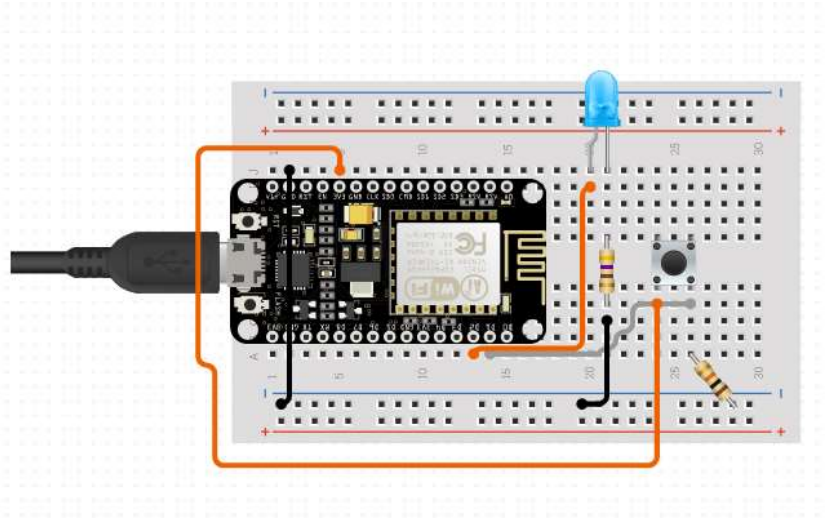
Echange d'informations entre un ESP et un site internet hébergé localement

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.



...

3 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.

4 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 ».

```
<!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>
```

```
28    <h4> Par Juan Alvarez et Olivier Grabenweger</h2>
29      <img src="Onizuka.jpg" alt="GTO" align="center" border=3>
30    <h2>Ennoncé :</h2>
31    <ul>
32      <li>Commander un appareil à distance via une page web
33      <li> Afficher l'état d'un bouton
34      <li> Afficher en temps réel les mesures d'un capteur
35    </ul>
36    <br>
37    <fieldset>
38      <legend style ="text-align:center;font-size:25px">
39        Data display</legend>
40      <span style="color:blue;size:20;text-align: center;">
41         Temperature: </span><span id="capteur_t";
42         style="color:green;size:20;"></span>
43      <br>
44      <span style="color:blue;size:20;text-align: center;">
45        Humidity: </span>
46        <span id="capteur_h";style="color:green;size:20;">
47        </span>
48      <br>
49      <span style="color:green;size:20;text-align: center;">
50        Button State: </span>
51        <span id="state_b";style="color:green;size:20;">
52        </span>
53      <br>
54      <span style="color:green;size:20;text-align: center;">
55        The button has been pressed : </span>  <span id="count";
56        style="color:green;size:20;"></span>
57        <span style="color:green;size:20;">_times</span>
58      <br>
59      <span style="text-align: center;"> LED status : </span>
60      <span id="state_LED"></span>
61    </fieldset>
62    <br>
63    <p style ="text-align:center"> Switch LED state </p>
64    <hr>
65    <br>
66    <div>
67      <button onclick="ON()">LED ON</button>
68      <p2 style="color:white;"> ceci est du texte </p2>
69      <button onclick="OFF()">LED OFF</button>
70    </div>
71    <br>
72    <hr style ="border: 1px solid green;">
73    <br>
74    <p style ="text-align:center"> Data Requests </p>
75    <hr>
76  </body>
77 </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  /*function callServer(url,cFunction)
2  {
3    var xhttp = new XMLHttpRequest();
4    xhttp.onreadystatechange = function()
5    {
6      if (this.readyState == 4 && this.status == 200)
7      {
8        cFunction(this);
9      }
10   };
11   xhttp.open("GET", url, 1);
12   xhttp.send();
13 }
14 function handleResponse(xhttp)
15   {
16     document.getElementById("state_LED").innerHTML = "LED" + xhttp.responseText;
17   }
18 function handleResponse2(xhttp)
19   {
20     document.getElementById("state_b").innerHTML =  xhttp.responseText;
21   }
22
23 setInterval(function handleResponse3(xhttp)
24 {
25   document.getElementById("capteur_t").innerHTML =  xhttp.responseText;
26 }, 2000);
27 setInterval(function handleResponse4(xhttp)
28 {
29   document.getElementById("capteur_h").innerHTML =  xhttp.responseText;
30 }, 2000);*/
31
32 ////////////////////////////////////////////////////////////////////////
33 function callServer(ID,url)
34 {
35     var xhttp = new XMLHttpRequest();
36
37     xhttp.onreadystatechange = function()
38     {
39         if(this.readyState == 4 && this.status == 200)
40         {
41             document.getElementById(ID).innerHTML = this.responseText;
42         }
43     };
44
45     xhttp.open("GET", url, true);
46     xhttp.send();
47 }
48 function ON() {
49     callServer("state_LED","LEDisON")
50 }
```

...

```
51 function OFF() {
52     callServer("state_LED", "LEDisOFF")
53 }
54
55 setInterval(function getData()
56 {
57     callServer("capteur_h","humidity")
58     callServer("capteur_t","temperature")
59     callServer("state_b","bp")
60 }, 2000);
61 setInterval(function getData()
62 {
63     callServer("count","compteur")
64 }, 500);
```

5 Code source ESP

```
1   #include <arduino.h>
2   #include <ESP8266WiFi.h>
3   #include <WiFiClient.h>
4   #include <Adafruit_Sensor.h>
5   #include <DHT.h>
6   #include <ESPAsyncWebServer.h>
7   #include "LittleFS.h"
8
9   #ifndef STASSID
10  #define STASSID "bbox-Sophie1"
11  #define STAPSK  "20150509Sophi"
12  #endif
13
14  #define DHTPIN 12     // Digital pin connected to the DHT sensor
15  #define DHTTYPE    DHT11     // DHT 11
16
17  DHT dht(DHTPIN, DHTTYPE);
18  // current temperature & humidity, updated in loop()
19  float t = 0.0;
20  float h = 0.0;
21
22  unsigned long previousMillis = 0;    // will store last time DHT was updated
23
24  // Updates DHT readings every 10 seconds
25  const long interval = 10000;
26
27  const int BUTTON = D0;
28  const int LEDred = D1;
29  const int LEDblue = D2;
30
31
32  int state_button = 0;
33  int flag_button = 0;
34  int count = 0;
35  const char* ssid = STASSID;
```

...

```
36 const char* password = STAPSK;
37 bool flag;
38 char str_temp[16];
39 char str_hum[16];
40 char count_txt[10];
41 String mybutton;
42
43 AsyncWebServer server(80);
44
45
46 void temp() {
47     sprintf(str_temp, "%f", t);
48 }
49 void humi() {
50     sprintf(str_hum, "%f", h);
51 }
52
53
54 void bp() {
55     if (digitalRead(BUTTON) == LOW)
56     {
57         mybutton = "bouton_actif";
58     }
59     else
60     {
61         mybutton = "bouton_non_actif";
62     }
63 }
64
65 void Compteur() {
66     sprintf(count_txt, "%d", count);
67 }
68
69 void button_Read()
70 {
71     state_button = digitalRead(BUTTON);
72     if (state_button == LOW)
73     {
74         flag_button = 1;
75         digitalWrite(LEDblue, HIGH);
76     }
77     else
78     {
79         digitalWrite(LEDblue, LOW);
80     }
81     if (flag_button == 1 && state_button == HIGH)
82     {
83         count += 1;
84         flag_button = 0;
85     }
86 }
```

...

```
87
88 void LEDisON() {
89     digitalWrite(LEDred, 1);
90 }
91 void LEDisOFF() {
92     digitalWrite(LEDred, 0);
93 }
94
95 void handleNotFound(AsyncWebServerRequest *request) {
96     request->send(404, "text/plain", "Not found");
97 }
98 // Replaces placeholder with DHT values
99 String processor(const String& var) {
100     //Serial.println(var);
101     if (var == "TEMPERATURE") {
102         return String(t);
103     }
104     else if (var == "HUMIDITY") {
105         return String(h);
106     }
107     return String();
108 }
109 void setup() {
110     pinMode(LEDred, OUTPUT);     // Initialize the LED pin as an output
111     digitalWrite(LEDred, LOW);
112     pinMode(LEDblue, OUTPUT);     // Initialize the LED pin as an output
113     digitalWrite(LEDblue, LOW);
114
115     pinMode(BUTTON, INPUT);
116
117     Serial.begin(115200);
118     dht.begin();
119     WiFi.mode(WIFI_STA);
120     WiFi.begin(ssid, password);
121     Serial.println("");
122
123   if(!LittleFS.begin())
124   {
125     Serial.println("Erreur LittleFS...");
126     return;
127   }
128
129   File root = LittleFS.open("/", "r");
130   File file = root.openNextFile();
131
132   while(file)
133   {
134     Serial.print("File: ");
135     Serial.println(file.name());
136     file.close();
137     file = root.openNextFile();
```

...

```
138 }
139
140   // Wait for connection
141   while (WiFi.status() != WL_CONNECTED) {
142     delay(500);
143     Serial.print(".");
144   }
145   Serial.println("");
146   Serial.print("Connected to ");
147   Serial.println(ssid);
148   Serial.print("IP address: ");
149   Serial.println(WiFi.localIP());
150
151   server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
152   {
153    request->send(LittleFS, "/index.html", "text/html");
154   });
155
156   server.on("/w3.css", HTTP_GET, [](AsyncWebServerRequest *request)
157   {
158    request->send(LittleFS, "/w3.css", "text/css");
159   });
160   server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest *request)
161   {
162      request->send(LittleFS, "/script.js", "text/javascript");
163   });
164   server.on("/Onizuka.jpg", HTTP_GET, [](AsyncWebServerRequest *request)
165   {
166    request->send(LittleFS, "/Onizuka.jpg", "image/jpg");
167   });
168   server.on("/humidity", HTTP_GET, [](AsyncWebServerRequest *request)
169   {
170    humi();
171    request->send(200, "text/plain", str_hum);
172   });
173
174   server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest *request)
175   {
176    temp();
177    request->send(200, "text/plain", str_temp);
178   });
179   server.on("/LEDisON", HTTP_GET, [](AsyncWebServerRequest *request)
180   {
181    LEDisON();
182    request->send(200,"text/plain", " is on");
183   });
184   server.on("/LEDisOFF", HTTP_GET, [](AsyncWebServerRequest *request)
185   {
186    LEDisOFF();
187    request->send(200,"text/plain", " is off");
188   });
```

...

```
189    server.on("/bp", HTTP_GET, [](AsyncWebServerRequest *request)
190    {
191      bp();
192      request->send(200, "text/plain", mybutton);
193    });
194    server.on("/compteur", HTTP_GET, [](AsyncWebServerRequest *request)
195    {
196      Compteur();
197      request->send(200, "text/plain", count_txt);
198    });
199
200    server.onNotFound(handleNotFound);
201    server.begin();
202    Serial.println("HTTP server started");
203 }
204
205 void loop() {
206    unsigned long currentMillis = millis();
207    if (currentMillis - previousMillis >= interval) {
208       // save the last time you updated the DHT values
209       previousMillis = currentMillis;
210       // Read temperature as Celsius (the default)
211       float newH = dht.readTemperature();
212       // Read temperature as Fahrenheit (isFahrenheit = true)
213       //float newT = dht.readTemperature(true);
214       // if temperature read failed, don't change t value
215       if (isnan(newH)) {
216          Serial.println("Failed to read from DHT sensor!");
217       }
218       else {
219          h = newH;
220          Serial.println(t);
221       }
222       // Read Humidity
223       float newT = dht.readHumidity();
224       // if humidity read failed, don't change h value
225       if (isnan(newT)) {
226          Serial.println("Failed to read from DHT sensor!");
227       }
228       else {
229          t = newT;
230          Serial.println(h);
231       }
232    }
233    button_Read();
234 }
```

...

6 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.

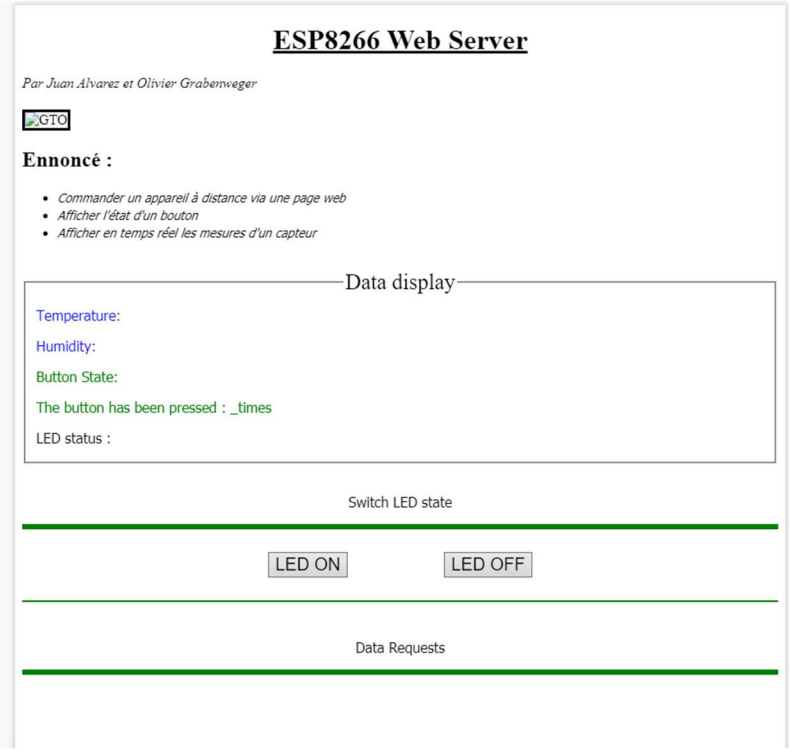
C'était un TP dont la difficulté ne résidait pas dans la complexité de la programmation mais dans la manipulation de nouveaux outils, c'est pourquoi les programmes abordés reste relativement simples (allumage d'une LED) mais les différents outils utilisés nous ont permis d'étoffer notre gamme de connaissances.

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.

7 Annexes, bibliographie et illustrations

a) 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.



b) 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.