Programmation

On va donc construire un mini serveur, qui quand il recevra une requête exécutera différentes actions. On va faire en sorte de lui donner les fonctions suivantes :

- répondre par une page un bouton « ouvrir »
- procéder à l'ouverture en elle même
- quand l'ordre d'ouverture est exécuté envoyer une page de confirmation

On va donc lancer son IDE Arduino et commencer par inclure les librairies qu'il faut :

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
```

Ensuite on va créer notre serveur sur un port normalement libre (grosso modo éviter les ports « classiques » des différents protocoles internet, une recherche google vous les donnera).

```
long port = 504;
ESP8266WebServer server(port); // serveur HTTP
```

On va ajouter une petite fonctionnalité gadget qui est le mDNS : vous vous souvenez du serveur DNS ? Ca marche sur internet, mais chez vous ? Et oui y'en a pas … Donc si vous voulez accéder à votre module sans connaitre son IP par coeur, il faut utiliser du mDNS. L'idée c'est d'envoyer un message à tout le monde le réseau et répond qui se sent concerné. On lance donc un « serveur » mDNS qui permettra de répondre à un domaine précis. Ici on va rentrer ça :

```
MDNSResponder mdns; // serveur mDNS

...

if (mdns.begin("esp8266", WiFi.localIP())) {
Serial.println("MDNS responder started");
}
```

Et on pourra appeler notre module dans le navigateur, connecté sur son réseau local (ça ne marchera pas depuis l'extérieur) en tapant « esp8266.local ».

Ensuite on va définir les réponses de notre serveur :

```
server.on("/", handle_root);
server.on("/open", handle_open);
```

Avec ces commandes, si je tape monip:monport/open j'exécute directement la routine d'ouverture, et il me répondra avec « opening ». Si je tape monip:monport il me répondra avec une page internet que vous visualisez ici en HTML. Lorsque j'appuierais sur le bouton « Ouvrir » de la page, ça appellera la page /open et donc l'ouverture.

Evidemment il faut se raccorder au wifi, on spécifie donc ces paramètres de connexion au départ :

```
char* ssid = "xxxxx"; // votre SSID
char* password = "yyyyyyy"; // votre mot de passe wifi
```

Le reste du code est au final assez logique : on pilote le relais, on lance la connexion, on lance les serveurs et on debug sur le port série. Ce qui donne au final :

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

char* ssid = "xxxxx"; // votre SSID
char* password = "yyyyyyy"; // votre mot de passe wifi
MDNSResponder mdns; // serveur mDNS
long port = 504;
ESP8266WebServer server(port); // serveur HTTP

const int led = 16; // led integree au NodeMCU, attention logique inverse
const int relay = 5; // relais connecte au GPIO5

void setup(void) {

/* Configuration des entree/sortie */
pinMode(relay, OUTPUT);
pinMode(led, OUTPUT);

digitalWrite(led, 1);
digitalWrite(relay, 0);

Serial.begin(115200); // initialisation du port serie
```

```
connect(ssid, password); // connexion au reseau Wifi

Serial.println("");

/* demarrage du serveur mDNS sur esp8266.local */
if (mdns.begin("esp8266", WiFi.localIP())) {
Serial.println("MDNS responder started");
}

/* ajout des actions du serveur */
server.on("/", handle_root);
server.on("/open", handle_open);

server.begin(); // demarrage du serveur

Serial.println("HTTP server started");
}

void loop(void) {

server.handleClient(); // gestion du serveur

/* Si connecté au wifi alors la LED s'allume */
if (WiFi.status() == WL_CONNECTED) digitalWrite(led, 0);
else digitalWrite(led, 1);

}

/* Routine pour se connecter à un reseau wifi */
void connect(char *_SSID, char* _PWD) {

Serial.println("");
Serial.print("Connecting ");
Serial.print(_SSID);
Serial.print(" password = ");
Serial.print( _PWD);

WiFi.begin(_SSID, _PWD);
Serial.println("");
```

```
int h = 0;

// Wait for connection
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");

if (h++ > 40) { // si trop long on abandonne
Serial.println();
Serial.println("Failed to connect");
return;
}

}

Serial.println("");
Serial.print("Connected to ");
Serial.println(ssid);
Serial.print("IP address: ");
Serial.println(WiFi.localIP());

}

/* Action serveur sur /open */
void handle_open() {
server.send(200, "text/plain", "opening"); // repond "opening"
open(); // actionne le relais
}

void handle_root() {

/* page[] contient notre page web et renvois vers le domaine /open si on appuie sur le bouton
*/
char page[] = "<h1>Ouverture portail</h1><p><a href=\"open\"><button>Ouvrir</button></a></p>";

server.send(200, "text/html", page); // repond avec la page web codee en HTML

}

void open() {
```

```
digitalWrite(relay, 1);
Serial.println("Opening");
delay(1000);
digitalWrite(relay, 0);

}
```

Sur Github

Revision #2
Created 2020-03-04 14:27:22 UTC
Updated 2020-03-04 16:12:03 UTC