ESP32 WiFi library use explanation

> To run the program please copy all the code in the same file
> Each M5Stack and ESP32 board has a static IP that is defined in the access point previosly

We start including the libraries that the program will required to work, this files are located in a folder called ***lib*** in the player folder.

ESP32 Server instances and variables

For explanation purpouses we will focus in the behaviour of ***WiFi.h***, we will generate two instances of the WiFiClient class called:

```
WiFiClient client_M5Stack;
WiFiClient client_ESP;
```

This two instances will be used to received the information to this ESPserver and the second one will send the information as a client to the Processing application.

ESP32 Server instances and variables

In the next step we will create an instance of the ***WiFiServer*** class called ***server()***, as you can see it had a number inside, this is the number of the port that we will use to create the communication between ESP32 and M5Stack, this number can any, it is recommended to review if the port that you set is not used for the any other protocol.

The next two variables that we will create are two constant char that we will use to set the name of the access point or Wifi that we want to access, the next varible will contain the password of the access point.

```
const char *ssid = "name of the access point";
const char *password = "password";
```

Client instances and variables

To connect the ESP32 with the Raspberry or any other PC is necessary to generate a variable that contains the number of the port that we will use for this purpose, keep in mind that this port must be the same as the port used in the server (Raspberry or PC). We also generate a variable called *host, this variable will set the IP address of the server that we will connect, in other words this IP must be the same of the destination PC.

> const uint16_t port = 3000;
> const char *host = "192.168.0.101"; // ip or dns

```
#define FIRSTLED_ALLOW_INTERRUPTS 0
#include <Arduino.h>
#include <WiFi.h>
#include <FastLED.h>

#define LED_PIN 27
#define NUM_LEDS 16

// Functions
void watch();
void server_raspberry();
void valves_actuation();

WiFiClient client_M5Stack;
WiFiClient client_ESP;

CRGB leds[NUM_LEDS];
//PGM Valve
int PGM_valve1 = 25;
int PGM_valve2 = 26;
int PGM_valve3 = 32;
int PGM_valve4 = 33;

//WiFi Connection
int wifiLED = 4;

//Player life
int lifeCount = 5;

// Server instances for ESP
WiFiServer server(80);
//Server variables
const char *ssid = "Super_Human_HM";
const char *password = "1234superhuman";
char c;

// Client variables to send to raspberry
const uint16_t port = 3000;
const char * host = "192.168.0.101"; // ip or dns
// flag to print just one time the IP of the connected server
char flag_activated = 'N';
```

Watch function

This function will "wait" until one M5stack is connected to this ESP32, if is connected it will print a string "New Client." in the serial monitor, and it will keep this client available while the client is connected to the network.

All the variables received by the client will be keep it in:

> c = client_M5Stack.read();

After this regarding the configuration of each ESP32 will detect different char values

```
void watch()
{
  // listen for incoming clients
  client_M5Stack = server.available();
  if (client_M5Stack)
  {
    // if client connected
    Serial.println("New Client.");
    while (client_M5Stack.connected())
    {
      // loop while the client's connected
      if (client_M5Stack.available())
      {
        // if there's bytes to read from the client
        c = client_M5Stack.read();
        Serial.println(c);
        if (c == 'H')
        {
          lifeCount--;
        }
        // read a byte
        valves_actuation();
        // This function connect to the Raspberry
        server_raspberry();
        c = '0';
      }
    }
  }
}

void valves_actuation()
{
  if (lifeCount == 4)
  {
    digitalWrite(PGM_valve1, HIGH);
    digitalWrite(PGM_valve2, LOW);
    digitalWrite(PGM_valve3, LOW);
    digitalWrite(PGM_valve4, LOW);

    //start this part for first hit
    leds[0] = CRGB(255, 0, 0);
    FastLED.show();
    leds[1] = CRGB(255, 0, 0);
    FastLED.show();
    leds[2] = CRGB(255, 0, 0);
    FastLED.show();
    leds[3] = CRGB(255, 0, 0);
```

```
    FastLED.show();
  }
  else if (lifeCount == 3)
  {
    digitalWrite(PGM_valve1, HIGH);
    digitalWrite(PGM_valve2, HIGH);
    digitalWrite(PGM_valve3, LOW);
    digitalWrite(PGM_valve4, LOW);

    //start this part for second hit
    leds[4] = CRGB(0, 255, 0);
    FastLED.show();
    leds[5] = CRGB(0, 255, 0);
    FastLED.show();
    leds[6] = CRGB(0, 255, 0);
    FastLED.show();
    leds[7] = CRGB(0, 255, 0);
    FastLED.show();
  }
  else if (lifeCount == 2)
  {
    digitalWrite(PGM_valve1, HIGH);
    digitalWrite(PGM_valve2, HIGH);
    digitalWrite(PGM_valve3, HIGH);
    digitalWrite(PGM_valve4, LOW);

    //start this part for third hit
    leds[8] = CRGB(0, 0, 255);
    FastLED.show();
    leds[9] = CRGB(0, 0, 255);
    FastLED.show();
    leds[10] = CRGB(0, 0, 255);
    FastLED.show();
    leds[11] = CRGB(0, 0, 255);
    FastLED.show();
  }
  else if (lifeCount == 1)
  {
    digitalWrite(PGM_valve1, HIGH);
    digitalWrite(PGM_valve2, HIGH);
    digitalWrite(PGM_valve3, HIGH);
    digitalWrite(PGM_valve4, HIGH);

    //start this part for forth hit
    leds[12] = CRGB(255, 200, 20);
    FastLED.show();
    leds[13] = CRGB(255, 200, 20);
    FastLED.show();
    leds[14] = CRGB(255, 200, 20);
    FastLED.show();
    leds[15] = CRGB(255, 200, 20);
    FastLED.show();
  }
}
```

server_raspberry function

This function has the role of send data has a client to the Raspberry or PC once the ***client_M5Stack*** detect something.

> This line will print the IP of the host, this must be the same as the one that was setup in the access point:
> Serial.println(host);

```
void server_raspberry()
{
  // Client connecting to raspberry
  // Print just one time the IP of the server
  if (flag_activated == 'N')
  {
    Serial.print("connecting to ");
    Serial.println(host);
    flag_activated = 'Y';
  }
  // This will send the request to the server
  if (c == 'H')
  {
    client_ESP.print("H");
  }
}
```

setup

Here we will start the connection of the ESP32 card to the WiFi with the next lines:

> WiFi.begin(ssid, password);
> // Connecting to the network
> //assigned when connected to a WiFi network
> Serial.print("Wait for WiFi... ");
> while (WiFi.status() != WL_CONNECTED)
> {
> delay(100);
> Serial.print(".");
> }

This will keep running until the board can connect to the WiFi connection that was programed

The next line will print the IP assigned to the ESP32, we can only see this IP in the serial terminal:

> Serial.println(WiFi.localIP());

Initialize the server that we will use for the M5Stack, for this example was set up in the port number 80:

> server.begin();

This line will comprobe if we are connected to the server that we want to share the information:

> if (!client_ESP.connect(host, port))
> {
> Serial.println("connection failed");
> Serial.println("wait a little...");
> delay(10);
> return;
> }

```
void setup()
{
  Serial.begin(115200);
  //FastLED setup
  FastLED.addLeds<WS2812, LED_PIN, GRB>(leds, NUM_LEDS);
  FastLED.setBrightness(40);
  //delay(10);
  // PIN setup
  pinMode(PGM_valve1, OUTPUT);
  pinMode(PGM_valve2, OUTPUT);
  pinMode(PGM_valve3, OUTPUT);
  pinMode(PGM_valve4, OUTPUT);
  pinMode(wifiLED, OUTPUT);
  // ESP Server
  WiFi.begin(ssid, password);
  // Connecting to the network
  //assigned when connected to a WiFi network
  Serial.print("Wait for WiFi... ");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(100);
    Serial.print(".");
  }
  digitalWrite(wifiLED, HIGH);
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  //delay(500);
  // This is to start the Server (for the M5Stack)
  server.begin();
  // This will comprobe if ESP32 is connected to Raspberry
  if (!client_ESP.connect(host, port))
  {
    Serial.println("connection failed");
```

```
    Serial.println("wait a little...");
    delay(10);
    return;
  }
}

void loop()
{
  // This function connect the watch
  watch();
  //delay(10);
}
```