M5Stack 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.

For explanation purpouses we will focus in the behaviour of ***WiFi.h*** and ***WiFiMulti.h***, we will generate one instance of the WiFiClient class and other one of the WiFiMulti class called:

> WiFiMulti WiFiMulti;
> WiFiClient client_M5Stack;

WiFiMulti will be used to connect to the WiFi network and client_M5Stack will be used has as the client.

This part two lines will set the port and the IP of the device that work as server and we want to connect, in this case the ESP32:

> const uint16_t port = 80;
> const char * host = "192.168.0.111"; // ip or dns

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";

```
#pragma mark - Depend ESP8266Audio and ESP8266_Spiram libraries
/*
cd ~/Arduino/libraries
git clone https://github.com/earlephilhower/ESP8266Audio
git clone https://github.com/Gianbacchio/ESP8266_Spiram
Use the "Tools->ESP32 Sketch Data Upload" menu to write the MP3 to SPIFFS
Then upload the sketch normally.
https://github.com/me-no-dev/arduino-esp32fs-plugin
*/
#include <Arduino.h>
#include <M5Stack.h>
#include <IRremote.h>
```

```
#include <SPIFFS.h>
#include <AudioFileSourceSD.h>
#include <AudioGeneratorWAV.h>
#include <AudioOutputI2S.h>
#include <WiFi.h>
#include <WiFiMulti.h>

//Function
void watch_functions();
void IR_Receptor();
void IR_Transmitter();
void Game_over();
void wifi_connection_esp(); //function for Wi-Fi
void Fsr_charging();        // Function for fsr sensor

/*Pin*/
int switch_PIN = 2;      //SHOT switch
int IR_receptorPin = 17; //Pin used to read IR values
int fsrPin = 36;         //Chetan: Pin used for reading FSR sensor readings.

/*Game status*/
int switchOut = 0;
int last_switchOut = 0;
int shootCount = 0;
int lifeCount = 5;
int recoverFlag = 0; //If recoverFlag = 0, player can recover 1 muscle. After use r

// Following four variables are needed for FSR based device charging mechanism.
int wristforce;
int squezecounter, maxsquezecount = 5;
int fsrthreshold = 4000;

/*Other*/
int i = 0;
int j = 10;

AudioGeneratorWAV *wav;
AudioFileSourceSD *file_sound;
AudioOutputI2S *out;

/*IR*/
IRsend irsend;                 //Pin number 3 is IR
IRrecv irrecv(IR_receptorPin); //Create an object
decode_results results;

// Wi-Fi objects
WiFiMulti WiFiMulti;
WiFiClient client_M5Stack;

// This port and IP are the ones that we will use to connect to ESP32
const uint16_t port = 80;
const char * host = "192.168.0.111"; // ip or dns
int stack_ip;

// Wifi network
const char * ssid = "Super_Human_HM";
```

```
const char * password = "1234superhuman";

void IR_Receptor()
{
  if (irrecv.decode(&results))
  {
    if (results.decode_type == SONY)
    {
      // send one character H (Hit) every time the players was shooted
      /*Make hitting sound*/
      file_sound->close();
      file_sound = new AudioFileSourceSD("/se_maoudamashii_battle12.wav");
      wav->begin(file_sound, out);
      dacWrite(25, 0);

      client_M5Stack.print("H");
      M5.Lcd.fillRect(20, 70 + ((5 - lifeCount) * 25), 60, 20, BLACK); //Remove a l
      M5.Lcd.setCursor(0, 0);
      lifeCount--;
    }
    irrecv.resume(); // Receive the next value
  }
}
```

Data transmision

This command will be used to transmit a char character to the server:

> client_M5Stack.println("S");

As you can see client_M5Stack is the instance that we create before we add ***.println()*** to reference this data member of the class to send the data.

```
void IR_Transmitter()
{
  /*Shoot switch was pressed*/
  /*Read switch*/
  switchOut = digitalRead(switch_PIN);
  if (shootCount > 0)
  {
    if (last_switchOut != switchOut)
    {
      if (switchOut == HIGH)
      {
        irsend.sendSony(0xa90, 12);
        /*Make shooting sound*/
        file_sound->close();
        file_sound = new AudioFileSourceSD("/se_maoudamashii_battle_gun05.wav");
        wav->begin(file_sound, out);
```

```
        dacWrite(25, 0);

        client_M5Stack.println("S");
        M5.Lcd.fillRect(130, 195 - (shootCount * 25), 60, 20, BLACK); //Remove a sh
        shootCount = shootCount - 1;
      }
    }
    last_switchOut = switchOut;
  }
}

// Fsr function
void Fsr_charging()
{
  wristforce = analogRead(fsrPin);
  Serial.println(wristforce);
  if (wristforce > fsrthreshold)
  {
    ++squezecounter;
    M5.Lcd.fillRect(240, 195 - (squezecounter * 25), 60, 20, YELLOW);
    delay(50);
    if (squezecounter == maxsquezecount)
    {
      squezecounter = 0;
      ++shootCount;
      M5.Lcd.fillRect(240, 70, 60, 20, BLACK);
      M5.Lcd.fillRect(240, 95, 60, 20, BLACK);
      M5.Lcd.fillRect(240, 120, 60, 20, BLACK);
      M5.Lcd.fillRect(240, 145, 60, 20, BLACK);
      M5.Lcd.fillRect(240, 170, 60, 20, BLACK);
      M5.Lcd.fillRect(130, 195 - (shootCount * 25), 60, 20, GREEN);
    }
  }
}

void Game_over()
{
  if (lifeCount == 0)
  {
    M5.Lcd.fillScreen(BLUE);
    M5.Lcd.setTextFont(4);
    M5.Lcd.setTextColor(WHITE);
    M5.Lcd.setCursor(70, 120);
    M5.Lcd.print("GAME OVER");
    delay(100000);
    exit(0);
  }
}

void watch_functions()
{
  /*LCD setup*/
  M5.Lcd.setTextFont(4);
  M5.Lcd.setCursor(0, 0);
  M5.Lcd.print("PLAYER1");
  M5.Lcd.setCursor(0, 30);
```

```
    M5.Lcd.print("IP :");
    M5.Lcd.setCursor(40, 30);
    M5.Lcd.print(host);

    M5.Lcd.drawRect(10, 60, 80, 140, WHITE);
    M5.Lcd.setCursor(25, 215);
    M5.Lcd.print("LIFE");

    M5.Lcd.setTextFont(4);
    M5.Lcd.drawRect(120, 60, 80, 140, WHITE);
    M5.Lcd.setCursor(115, 215);
    M5.Lcd.print("BULLET");

    M5.Lcd.setTextFont(4);
    M5.Lcd.drawRect(230, 60, 80, 140, WHITE);
    M5.Lcd.setCursor(215, 215);
    M5.Lcd.print("CHARGE");

    /*IR Receptor*/
    IR_Receptor();

    /*IR Transmitter*/
    IR_Transmitter();

    /*FSR based shooting charging*/
    Fsr_charging();

    /*Game over*/
    Game_over();

    /*??*/
    if (wav->isRunning())
    {
      if (!wav->loop())
      {
        wav->stop();
      }
    }
  }
```

Setup

We start by connecting to a WiFi network, in the WiFiMulti instance that we create before we call the constructor ***.addAP*** with the variables that we create before for the name of the network that we are using and the currently password of this network:

> WiFiMulti.addAP(ssid, password);

The next lines of the code will call the constructor ***.run*** to start the connection of the client to the WiFi access and wait until is connected.

```
while (WiFiMulti.run() != WL_CONNECTED)
{
M5.Lcd.setCursor(i, j);
M5.Lcd.print(".");
delay(10);
i = i + 5;
}
```

Finally we call the constructor ***.connect*** of the instance ***client_M5Stack*** to determinate if the client is connected to the server, as we can see this constructor use 2 variables, one for the host, IP of the server, and other for the port that we will use to send the information, note that this variables were declared in the first part of the code, so is not necessary to write them again here.

```
if (!client_M5Stack.connect(host, port))
{
M5.Lcd.setCursor(0, 100);
M5.Lcd.print("connection failed");
M5.Lcd.setCursor(0, 115);
M5.Lcd.print("wait...");
delay(50);
return;
}
```

```
void setup()
{
  Serial.begin(115200);
  M5.begin();

  // /*LCD setup*/
  M5.Lcd.setTextFont(2);
  M5.Lcd.setTextColor(WHITE);
  M5.Lcd.setCursor(0, 0);
  M5.Lcd.print("Wait for WiFi... ");

  // WiFi setup
  // We start by connecting to a WiFi network
  WiFiMulti.addAP(ssid, password);
  // Start the connection of the client and wait until connect to the lan
  while (WiFiMulti.run() != WL_CONNECTED)
  {
    M5.Lcd.setCursor(i, j);
    M5.Lcd.print(".");
    delay(10);
    i = i + 5;
  }
  M5.Lcd.fillScreen(BLACK);
  M5.Lcd.setCursor(0, 25);
```

```
  M5.Lcd.print("WiFi connected");
  // //print the IP assigned to the device
  M5.Lcd.setCursor(0, 40);
  M5.Lcd.print("IP address: ");
  M5.Lcd.setCursor(0, 55);
  M5.Lcd.print(WiFi.localIP());
  //delay(500);
  M5.Lcd.setCursor(0, 70);
  M5.Lcd.print("connecting to ");
  M5.Lcd.setCursor(0, 85);
  M5.Lcd.print(host);
  // This will comprobe if Stack is connected to the ESP-Server
  if (!client_M5Stack.connect(host, port))
  {
    M5.Lcd.setCursor(0, 100);
    M5.Lcd.print("connection failed");
    M5.Lcd.setCursor(0, 115);
    M5.Lcd.print("wait...");
    delay(50);
    return;
  }
  /*Audio setup*/
  /*Please move music file(se_maoudamashii_battle_gun05.wav) into SD.
  This file put on the music folder*/
  file_sound = new AudioFileSourceSD("/se_maoudamashii_battle_gun05.wav");
  out = new AudioOutputI2S(0, 1); // Output to builtInDAC
  out->SetOutputModeMono(true);
  wav = new AudioGeneratorWAV();
  //wav->begin(file, out);

  /*Game setup*/
  pinMode(switch_PIN, INPUT);
  pinMode(fsrPin, INPUT); // Reading FSR sensor readings.
  M5.Lcd.fillScreen(BLACK);

  /*IRrecev setup*/
  irrecv.enableIRIn(); // Start the receiver

  /*Show your live*/
  M5.Lcd.fillRect(20, 70, 60, 20, RED);
  M5.Lcd.fillRect(20, 95, 60, 20, RED);
  M5.Lcd.fillRect(20, 120, 60, 20, RED);
  M5.Lcd.fillRect(20, 145, 60, 20, RED);
  M5.Lcd.fillRect(20, 170, 60, 20, RED);

  irsend.sendSony(0xa90, 12);
}

void loop()
{
  watch_functions();
  M5.update();
}
```