



C4.2 Programación Microcontrolador NodeMCU ESP32

Comunicación por medio de la conexión Wi-Fi



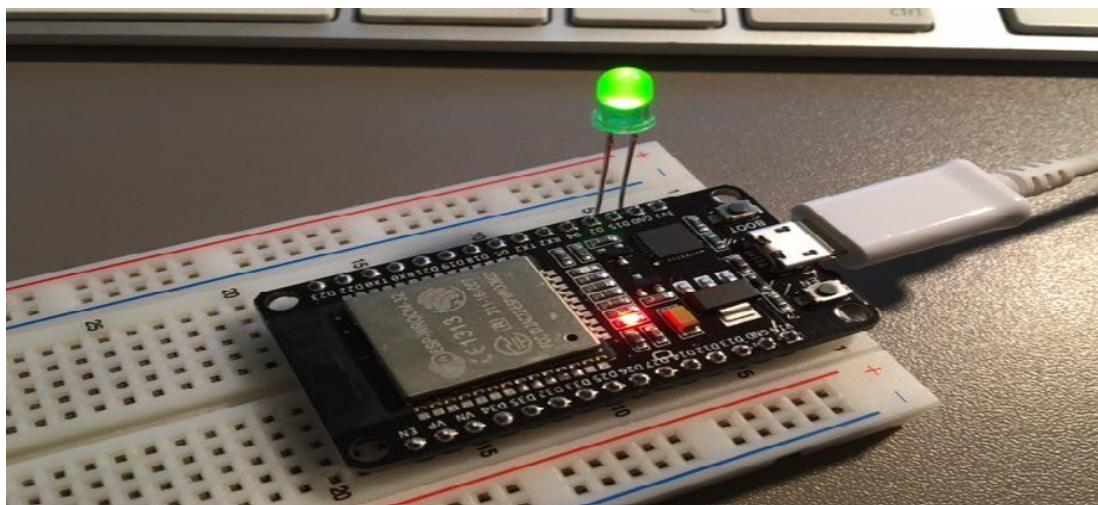
Instrucciones

- De acuerdo con la información presentada por el asesor referente al tema, desarrollar lo que se indica dentro del apartado siguiente.
- Toda actividad o reto se deberá realizar utilizando el estilo **MarkDown con extensión .md** y el entorno de desarrollo VSCode, debiendo ser elaborado como un documento **single page**, es decir si el documento cuenta con imágenes, enlaces o cualquier documento externo debe ser accedido desde etiquetas y enlaces.
- Es requisito que el archivo .md contenga una etiqueta del enlace al repositorio de su documento en Github, por ejemplo **Enlace a mi GitHub**
- Al concluir el reto el reto se deberá subir a github el archivo .md creado.
- Desde el archivo **.md** se debe exportar un archivo **.pdf** con la nomenclatura **C4.2_NombreAlumno_Equipo.pdf**, el cual deberá subirse a classroom dentro de su apartado correspondiente, para que sirva como evidencia de su entrega; siendo esta plataforma **oficial** aquí se recibirá la calificación de su actividad por individual.
- Considerando que el archivo .pdf, fue obtenido desde archivo .md, ambos deben ser idénticos y mostrar el mismo contenido.
- Su repositorio ademas de que debe contar con un archivo **readme.md** dentro de su directorio raíz, con la información como datos del estudiante, equipo de trabajo, materia, carrera, datos del asesor, e incluso logotipo o imágenes, debe tener un apartado de contenidos o índice, los cuales realmente son ligas o **enlaces a sus documentos .md**, evite utilizar texto para indicar enlaces internos o externo.
- Se propone una estructura tal como esta indicada abajo, sin embargo puede utilizarse cualquier otra que le apoye para organizar su repositorio.

```
| readme.md
|   | blog
|   |   | C4.1_TituloActividad.md
|   |   | C4.2_TituloActividad.md
|   |   | C4.3_TituloActividad.md
|   |   | C4.4_TituloActividad.md
|   |   | C4.5_TituloActividad.md
|   | img
|   | docs
|   |   | A4.1_TituloActividad.md
|   |   | A4.2_TituloActividad.md
```

Desarrollo

1. Basado en el siguiente circuito, ensamblarlo, utilizando los elementos electrónicos observados.



Fuente de consulta: [Random Nerd Tutorials](#)

2. Analice y apóyese del programa que se muestra a continuación para elaborar el reto.

```
/*
WiFi Web Server Simple
*/
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "<identificador>";
const char* password = " <password>";

WebServer server(80); // Object of WebServer(HTTP port, 80 is defult)

void setup() {
  Serial.begin(115200);
  Serial.println("Try Connecting to ");
  Serial.println(ssid);

  // Connect to your wi-fi modem
  WiFi.begin(ssid, password);

  // Check wi-fi is connected to wi-fi network
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected successfully");
  Serial.print("Got IP: ");
  Serial.println(WiFi.localIP()); //Show ESP32 IP on serial
```

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

server.begin();
Serial.println("HTTP server started");
delay(100);
}

void loop() {
    server.handleClient();
}

// HTML & CSS contents which display on web server
String HTML = "<!DOCTYPE html>\n<html>\n<body>\n<h1>Mi Primer Servidor Web with ESP32 - Station Mode &#128522;</h1>\n</body>\n</html>";

// Handle root url (/)
void handle_root() {
    server.send(200, "text/html", HTML);
}
```

3. Pruebe y observe los resultados obtenidos explicándolos en esta sección.

Lo que hace el código anterior es conectar el ESP32 a la red de la casa y hacerse pasar como un servidor el cual puedes acceder con el numero de IP que te arroja el mismo para que puedas ver la pagina.

4. Al programa anterior agregue las instrucciones necesarias para que se despliegue en la interface un botón que permita encender y apagar un Led tal como se muestra en la figura 1.

5. Inserte aquí las imágenes que considere como evidencias para demostrar el resultado obtenido.


```
#include <WiFi.h>

const char* ssid = "100 Euros";
const char* password = "Enrollados En Tu Ano";

WiFiServer server(80);

String header;
String output25State = "off";
String output26State = "off";
String output27State = "off";

unsigned long currenTime = millis();
unsigned long previousTime = 0;
const long timeoutTime = 2000;

#define LEDR 25
#define LEDG 26
#define LEDB 27

#define R_channel 0
#define G_channel 1
#define B_channel 2

#define pwm_Frequency 5000
#define pwm_resolution 8

void setup() {
    ledcAttachPin(LEDR, R_channel);
    ledcAttachPin(LEDG, G_channel);
    ledcAttachPin(LEDB, B_channel);

    ledcSetup(R_channel, pwm_Frequency, pwm_resolution);
    ledcSetup(G_channel, pwm_Frequency, pwm_resolution);
    ledcSetup(B_channel, pwm_Frequency, pwm_resolution);

    pinMode(R_channel, OUTPUT);
    pinMode(B_channel, OUTPUT);
    pinMode(G_channel, OUTPUT);

    Serial.begin(115200);
    Serial.println("Trv Connecting to ");
```

```
Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}

Serial.println("");
Serial.println("WiFi connected successfully");
Serial.print("Got IP: ");
Serial.println(WiFi.localIP());

server.begin();
}

void loop()
{
WiFiClient client = server.available();

if(client)
{
currentTime = millis();
previousTime = currentTime;
Serial.println("New Client");
String currentLine = "";

while(client.connected() && currentTime - previousTime <= timeoutTime)
{
currentTime = millis();
if (client.available())
{
char c = client.read();
Serial.write(c);
header += c;
if(c == '\n')
{
if(currentLine.length() == 0)
{
client.println("HTTP/1.1 200 OK");
client.println("Content-type:text/html");
client.println("Connection: close");
client.println();

if(header.indexOf("GET /25/on") >= 0)
{
Serial.println("GPIO 25 on");
output25State = "on";
ledcWrite(R_channel, 255);
}
else if(header.indexOf("GET /25/off") >= 0)
{
Serial.println("GPIO 25 off");
output25State = "off";
ledcWrite(R_channel, 0);
}
}
}
}
}
```

```
        else if(header.indexOf("GET /26/on") >= 0)
    {
        Serial.println("GPIO 26 on");
        output26State = "on";
        ledcWrite(G_channel, 255);
    }
    else if(header.indexOf("GET /26/off") >= 0)
    {
        Serial.println("GPIO 26 off");
        output26State = "off";
        ledcWrite(G_channel, 0);
    }

    else if(header.indexOf("GET /27/on") >= 0)
    {
        Serial.println("GPIO 27 on");
        output27State = "on";
        ledcWrite(B_channel, 255);
    }
    else if(header.indexOf("GET /27/off") >= 0)
    {
        Serial.println("GPIO 27 off");
        output27State = "off";
        ledcWrite(B_channel, 0);
    }
    client.println("<!DOCTYPE html><html>");
    client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
    client.println("<link rel=\"icon\" href=\"data:,\">");
    client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center; }");
    client.println(".button { background-color: #4CAF50; border: none; color: white; padding: 16px 40px; }");
    client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer; }");
    client.println(".button2 {background-color: #555555;}</style></head>");

    client.println("<body><h1>ESP32 Web Server</h1>");

    client.println("<p>RED COLOR - State " + output25State + "</p>");
    if (output25state=="off") {
        client.println("<p><a href=\"/25/on\"><button class=\"button\">ON</button></a></p>");

    } else {
        client.println("<p><a href=\"/25/off\"><button class=\"button button2\">OFF</button></a></p>");

    }

    client.println("<p>GREEN COLOR - State " + output26State + "</p>");
    if (output26State=="off") {
        client.println("<p><a href=\"/26/on\"><button class=\"button\">ON</button></a></p>");

    } else {
        client.println("<p><a href=\"/26/off\"><button class=\"button button2\">OFF</button></a></p>");

    }

    client.println("<p>BLUE COLOR - State " + output27State + "</p>");
    if (output27State=="off") {
        client.println("<p><a href=\"/27/on\"><button class=\"button\">ON</button></a></p>");

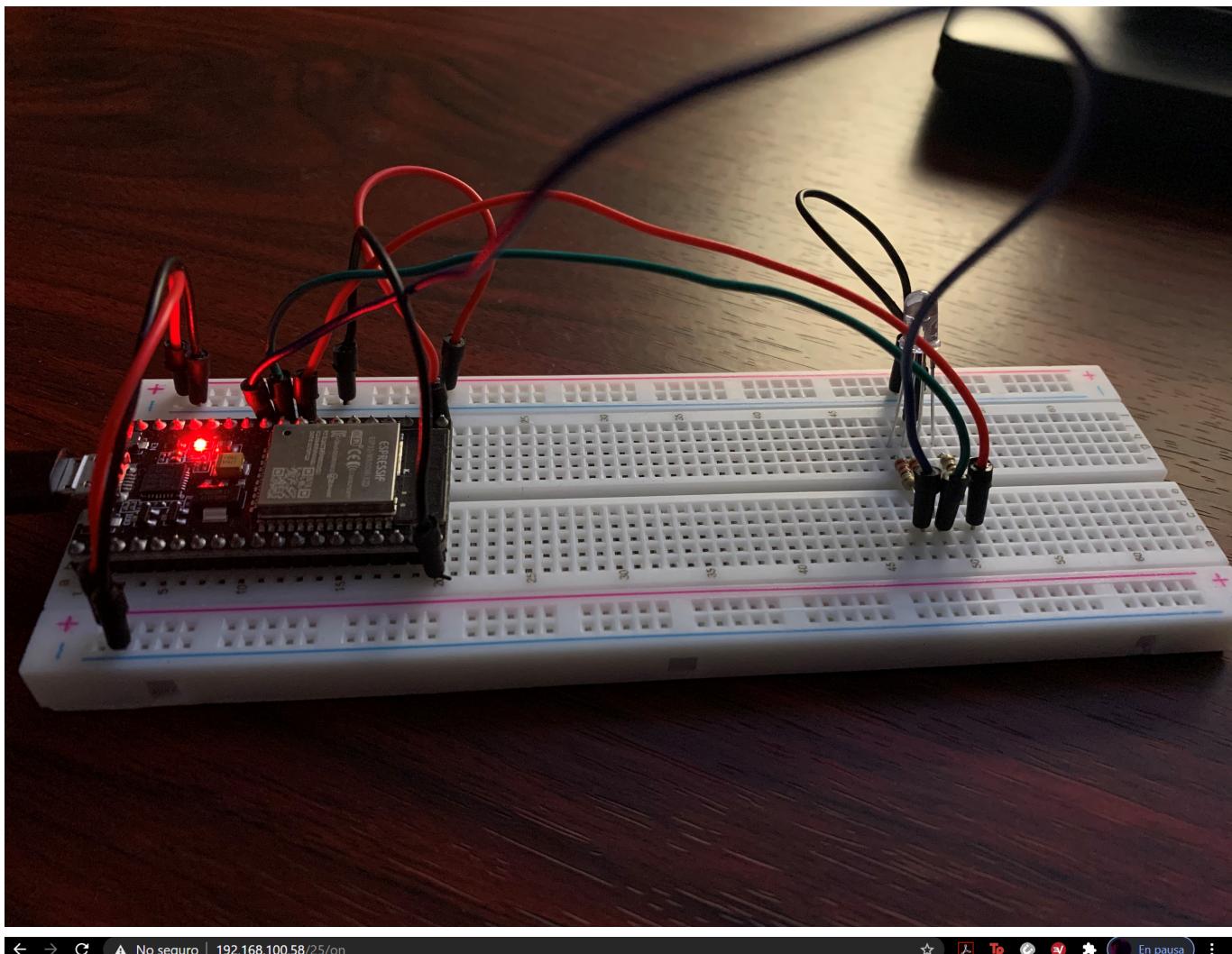
    } else {
        client.println("<p><a href=\"/27/off\"><button class=\"button button2\">OFF</button></a></p>");

    }

    client.println("</body></html>");

    client.println();
    break;
}
else
{
    currentLine = "";
}
}
else if (c != '\r')
{
    currentLine += c;
}
}
}

header = "";
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}
}
```



← → C ▲ No seguro | 192.168.100.58/25/on ☆ 🔍 ⏪ 🔍 🎮 🎵 En pausa ⋮

ESP32 Web Server

RED COLOR - State on

OFF

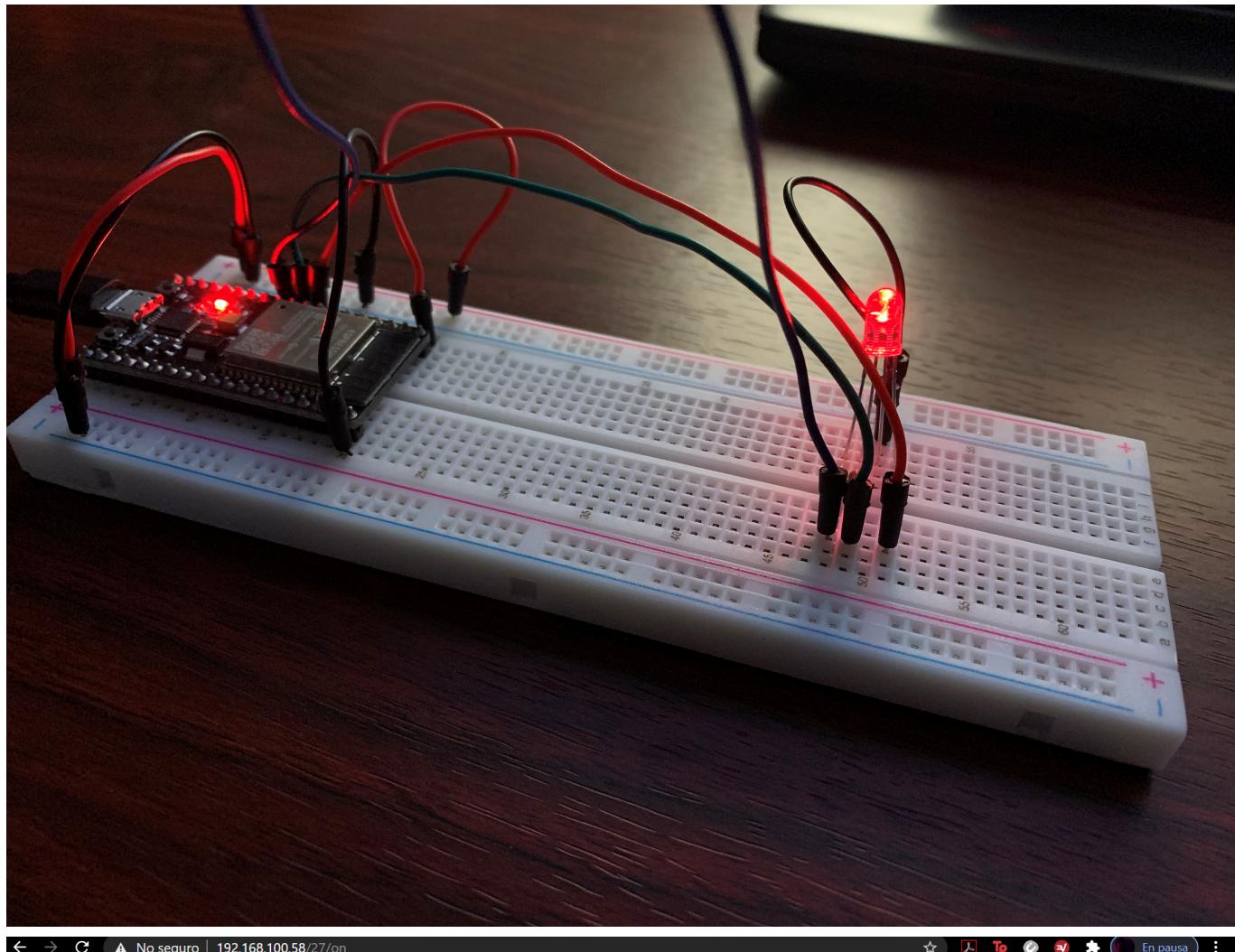
GREEN COLOR - State off

ON

BLUE COLOR - State off

ON

192.168.100.58/25/off



← → C ▲ No seguro | 192.168.100.58/27/on ⋮

ESP32 Web Server

RED COLOR - State off

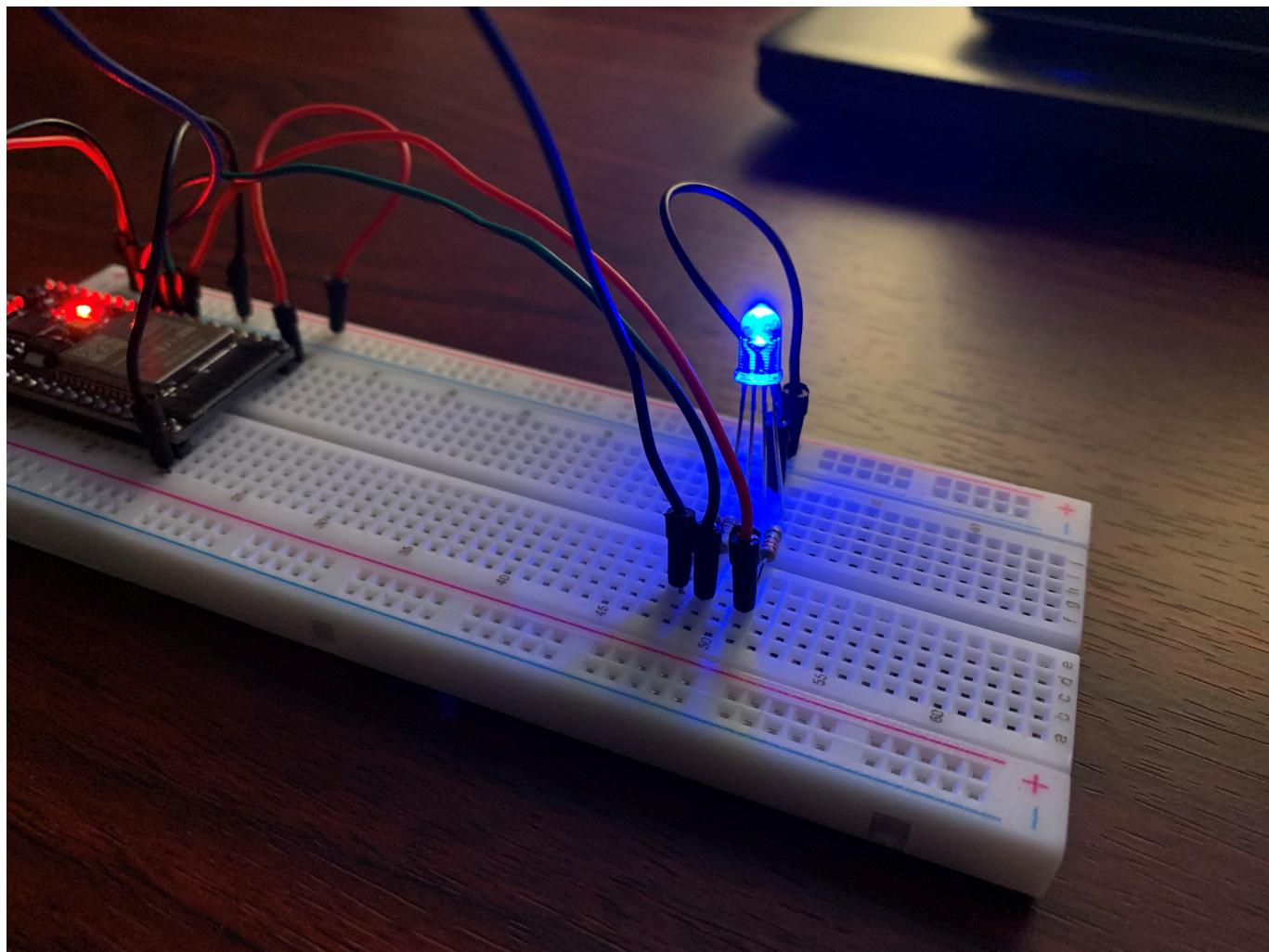
ON

GREEN COLOR - State off

ON

BLUE COLOR - State on

OFF



6. Conclusion

El código para el desarrollo de esta actividad fue más largo de lo que pensé. Por un momento pense que era agregar un par de líneas al código que se encuentra en la parte de arriba, pero no, se tiene que cambiar un poco y eliminar unas de las librerías que se usa. Además de imprimir todo el código HTML para que sea visible la página en el ciclo Loop del código.

Rubrica

Criterios	Descripción	Puntaje
Instrucciones	Se cumple con cada uno de los puntos indicados dentro del apartado Instrucciones?	20
Desarrollo	Se respondió a cada uno de los puntos solicitados dentro del desarrollo de la actividad?	80

 [Ir a Github](#)