



C4.2 Programación Microcontrolador NodeMCU ESP32

Comunicación por medio de la conexión Wi-Fi



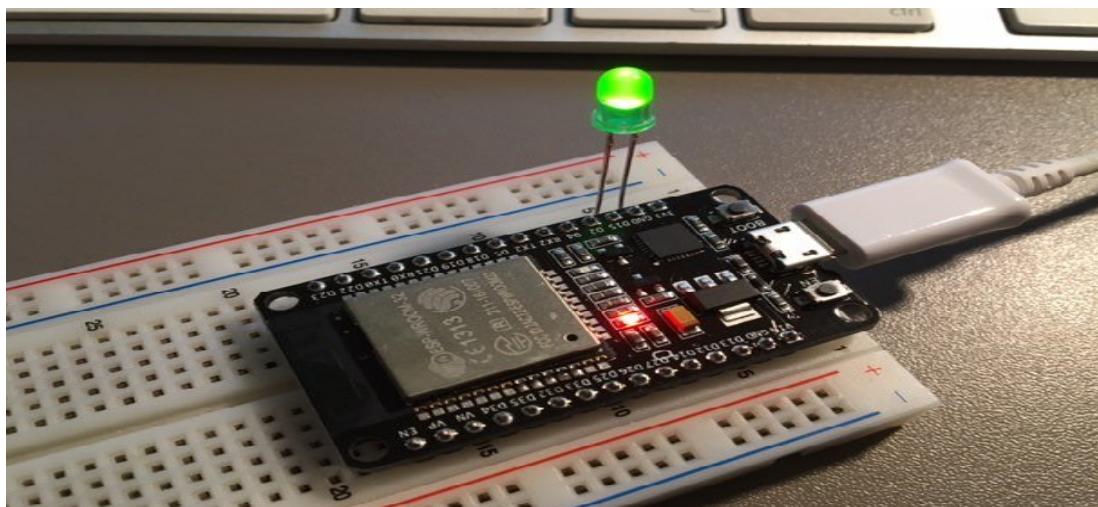
Instrucciones

- 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 extension .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.

Se puede apreciar una página web con el título Mi Primer Servidor Web with ESP32 - Station Mode, la cual se pudo acceder a ella mediante la conexión de un dispositivo con un servidor web apoyandose del ESP32.

En el punto 4, simplemente se agregó un botón para poder manipular un led mediante la conexión establecida anteriormente.

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.

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

const char* ssid = "INFINITUM2732_2.4";
const char* password = "76cf6uNbpu";

int LEDpin = 16;
bool LEDstatus = LOW;
int freq = 5000;
int ledCanal = 0;
int resolucion = 8;

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

```
void setup() {
    Serial.begin(115200);
    Serial.println("Try Connecting to ");
    Serial.println(ssid);
    pinMode(LEDpin, OUTPUT);
    ledcSetup(ledCanal, freq, resolucion);
    ledcAttachPin(LEDpin, ledCanal);

    // 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.on("/ledON", handle_ledon);
    server.on("/ledOFF", handle_leloff);

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

void loop() {
    server.handleClient();
    if(LEDstatus){
        ledcWrite(ledCanal, 255);
    }
    else{
        ledcWrite(ledCanal, 0);
    }
}

// HTML & CSS contents which display on web server
String HTML(uint8_t ledStatus) {
    String texto = "<!DOCTYPE html>\n<html>\n<style> \n.center {text-align:center;} \n.button {display: block;background-color: #48d056;border: none;color:white;padding: 13px 30px;text-decoration: none;font-size: 100px;margin: 1% auto 1%;cursor: pointer;border-radius: 4px;}\n.button-on {background-color: #c23636;}\n.button-on:active {background-color: #c77676;}\n.button-off {background-color: #0e977c;}\n.button-off:active {background-color: #629e92;}\n</style>\n<body>\n<h1>ESP32 Web Server</h1>\n<button class='center button'>LED ON</button>\n<button class='center button'>LED OFF</button>\n</body>\n</html>";
    return texto;
}
```

```
body{margin-top: 5%; }\
a {width: 25%; box-shadow: 10px 5px 5px black; margin: 10% auto;}\
h1 {color: #ffffff; } \
h2 {color: #ffffff; }\
div {box-shadow: 10px 5px 5px black; margin: 10px auto 20px; max-width: 100%; width: 80%; background-color: #5f70cf; padding: 13px 30px; border-radius: 20px;}\
</style>\
<body class="center">\
<div>\
<h1> Gestion del LED del penthouse de CodeDevelopers </h1>;

if(ledStatus){
    texto += "<h2> Estado del LED: Encendido </h2><a href=\"/ledOFF\" class=\"button button-on\"> OFF </a>";
}
else{
    texto += "<h2> Estado del LED: Apagado </h2><a href=\"/ledON\" class=\"button button-off\"> ON </a>";
}

texto += "</div></body></html>";
return texto;
}

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

void handle_ledon() {
    LEDstatus = HIGH;
    Serial.println("Metodo encender");
    server.send(200, "text/html", HTML(LEDstatus));
}

void handle_ledoff() {
    LEDstatus = LOW;
    Serial.println("Metodo apagar");
    server.send(200, "text/html", HTML(LEDstatus));
}
```

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

Screenshot of a Slack workspace showing a conversation in the channel `code_developers`. The message from **EDUARDO MORGADO JACOME** at 20:51 includes a link to a Visual Studio Live Share session. The message from **ELDEN HUMBERTO CRUZ VERA** at 21:05 says "me parece perfecto". The message from **ABNER JESUS PERALES NIEBLA** at 21:05 says "A darle 7u7". The message from **EDUARDO MORGADO JACOME** at 21:24 includes another link to a Visual Studio Live Share session.

Screenshot of a Discord server interface. In the left sidebar, under the **Projects** category, there is a channel named `sistemas-programables`. A message from **Abner** at 09/12/2020 shows a snippet of code:

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

A floating window titled "Edit The Code" displays the CSS for a button:

```
.button1 {font-size: 10px;}
```

The right side of the screen shows a message from **Elden Cruz** at 09/12/2020 with a link to a tutorial: <https://lastminuteengineers.com/creating-esp8266-web-server-arduino-ide/>.

Last Minute Engineers

In-depth: Create A Simple ESP8266 NodeMCU Web Server In Arduino IDE

Learn to create simple ESP8266 NodeMCU web server in Arduino IDE as a Access Point (AP mode) & Staion (STA mode) with Detailed Code Explanation.



Voz conectada
General / Projects

Código.html y 10 páginas...

BlitzZerk Crazy mode

Enviar mensaje a #sistemas-programables

Gestión del LED del penthouse de CodeDevelopers

Estado del LED: Apagado

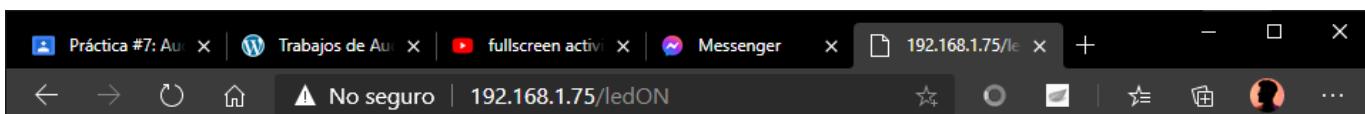
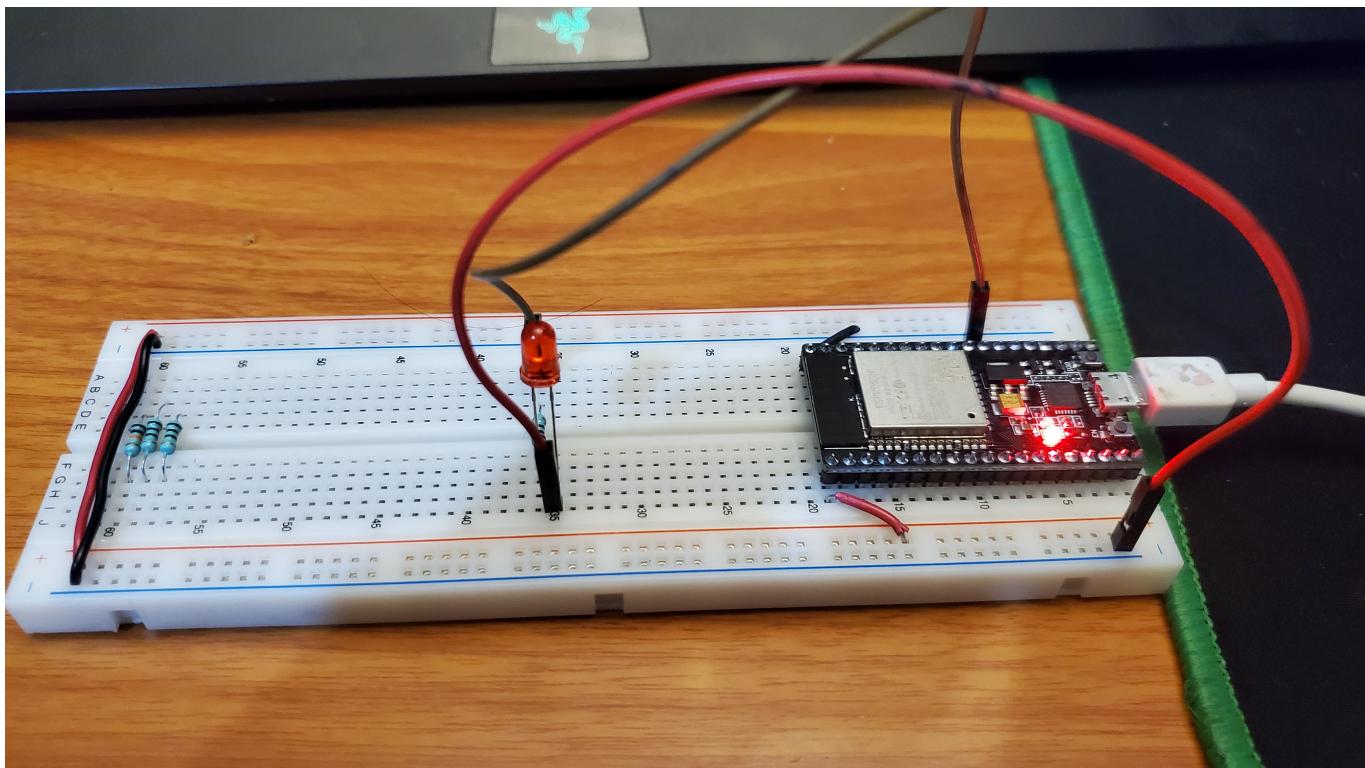
ON

COM7

```

.
WiFi connected successfully
Got IP: 192.168.1.75
HTTP server started

```

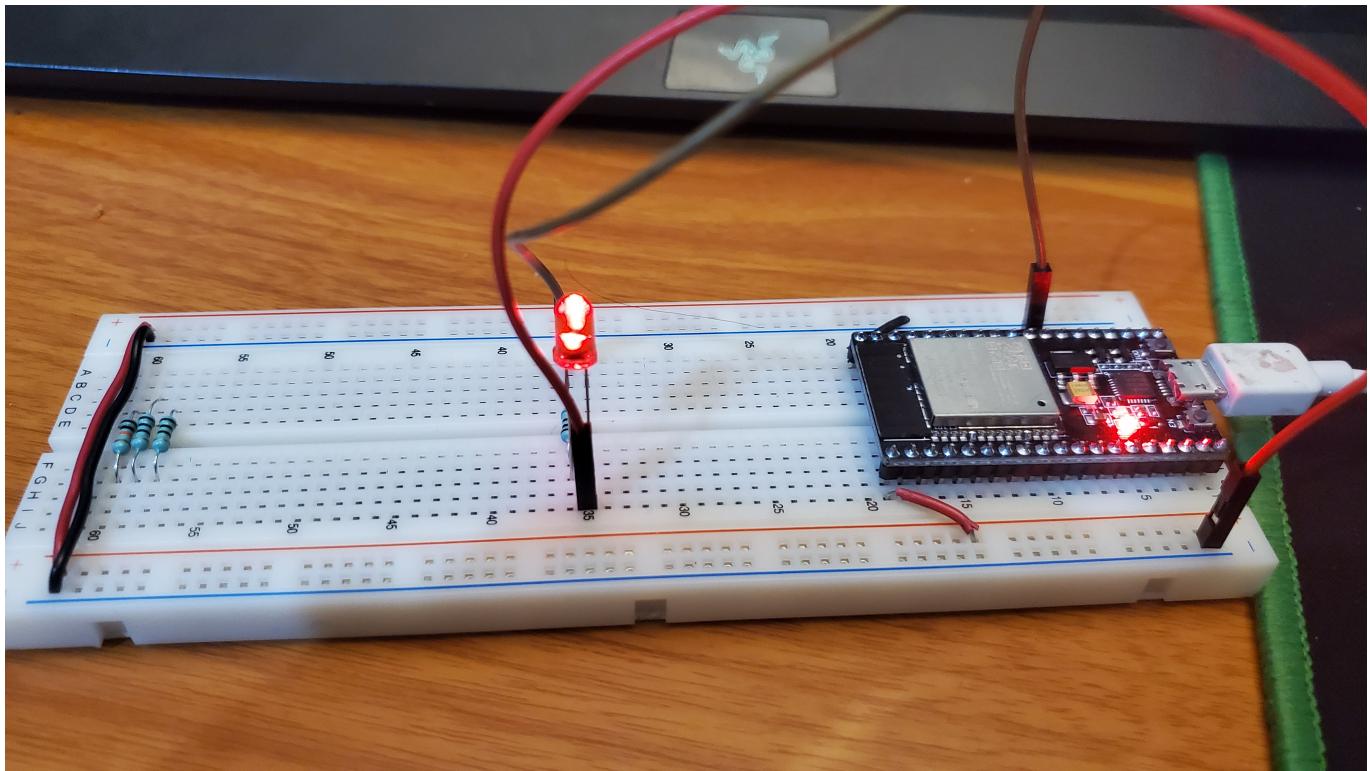


Gestion del LED del penthouse de CodeDevelopers

Estado del LED: Encendido

OFF

```
COM7
.
WiFi connected successfully
Got IP: 192.168.1.75
HTTP server started
Metodo encender
```



Desde el telefono



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 inicio](#)

[Repositorio de Github de Morgado Jacome Eduardo](#)

[Repositorio de Github de Cruz Vera Elden Humberto](#)

 [Repositorio de Github de Perales Niebla Abner Jesús](#) 