

Introducción:

Se desea implementar un montaje de 4 leds sobre la placa ESP8266, asignados correspondientemente a pines de dicha placa.

También se requiere de un formulario web (HTML + PHP) donde se tenga un selector de encendido/apagado para cada led.

Se debe guardar la información del formulario en un archivo **Status.info**, este será leído por la placa Arduino e interpretado para manejar los leds según su contenido indique.

Para la infraestructura web se utilizará una pc conectada a la red local donde se levantará un servidor con Xampp, de modo tal que el Arduino se conectará como cliente para acceder al archivo Status.info.

Seguidamente se muestran los materiales necesarios para realizar la implementación sobre la placa.

Materiales:

Arduino ESP8266

Protoboard 400 puntos

4 leds de colores

Resistencias de 1 K Ω (una por cada led)

5 Cables Dupont macho-hembra.

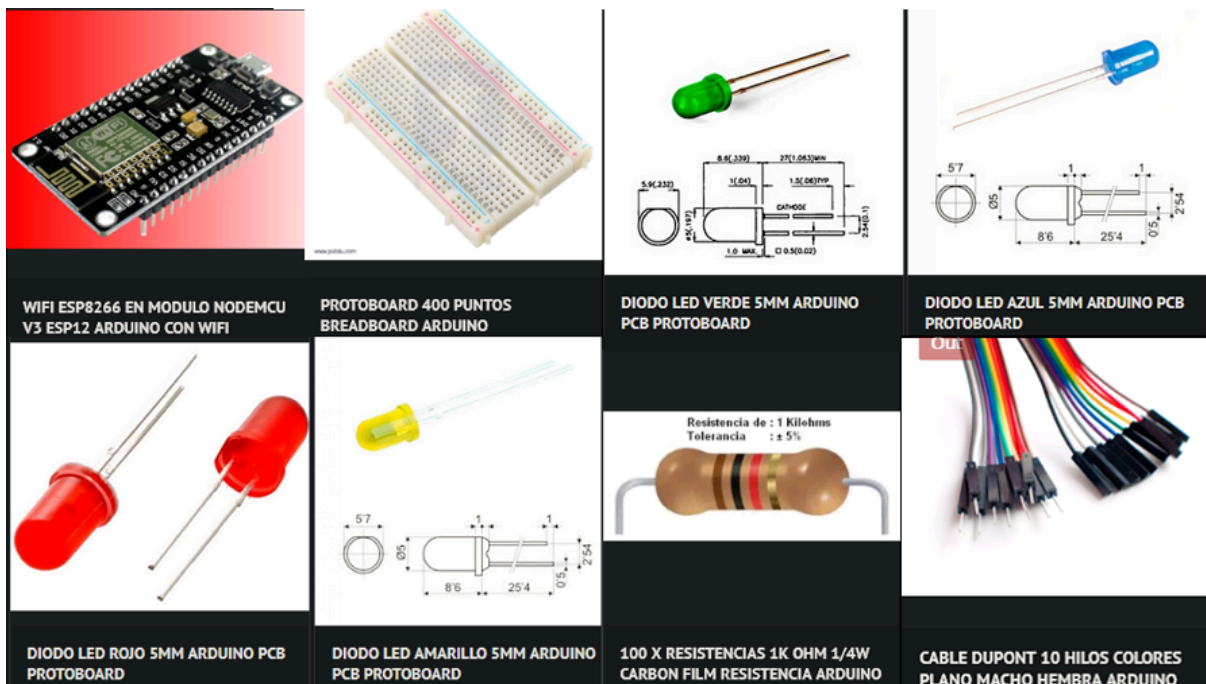
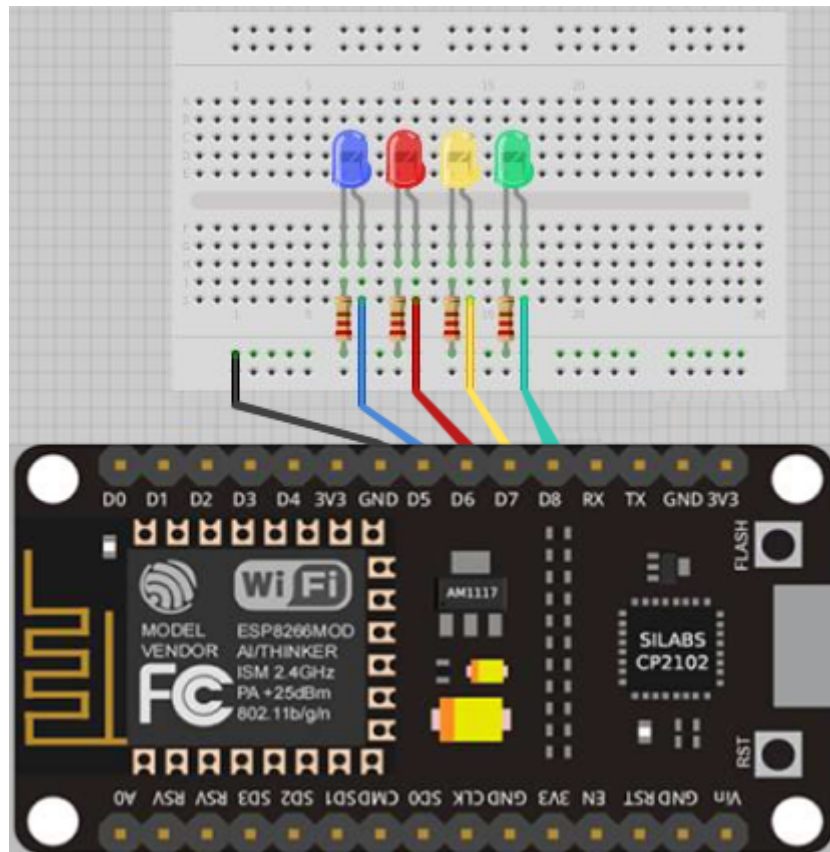
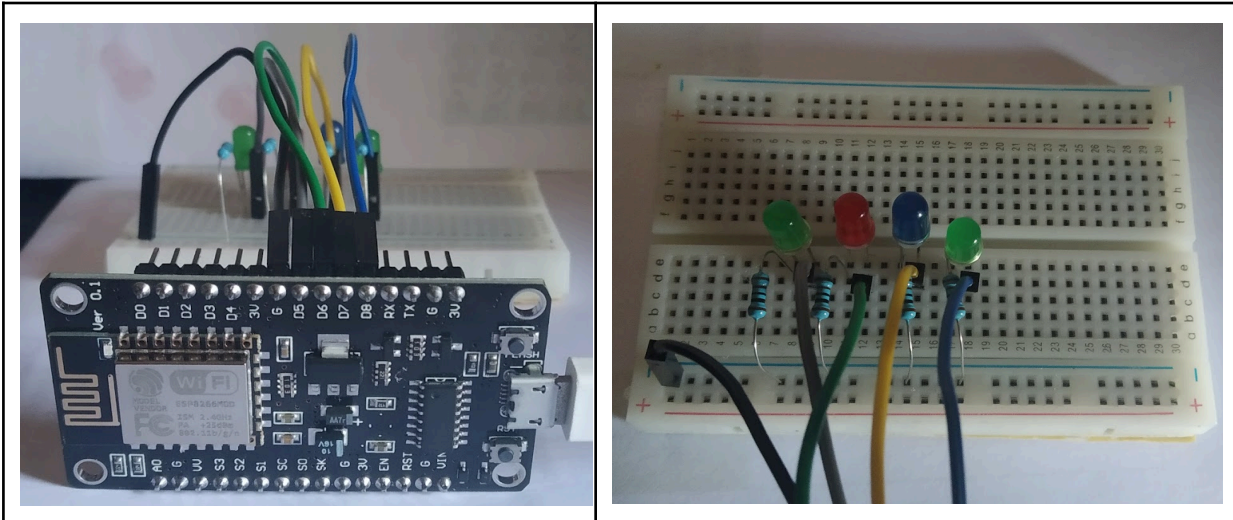


Diagrama de la implementación:**Fotografía del diagrama implementado:**

Led.html:

En la creación del html usamos el elemento de entrada checkbox que permite insertar un vector o array de valores. El atributo checked se usa para indicar que el elemento está en 1.

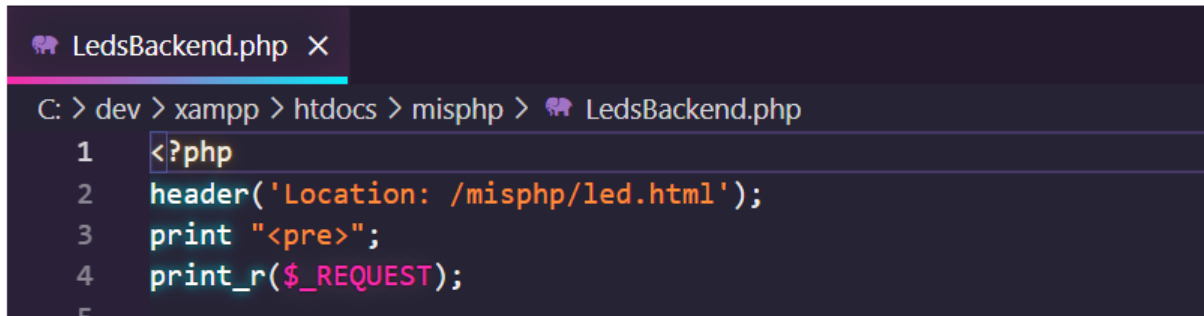
Para enviar el valor 0 se utilizó el atributo hidden para esconder y enviar el valor cuando este no selecciona el checkbox del led correspondiente.

Cuando se pulsa el botón enviar esos valores se almacenan y se carga la pagina php.

```
<> led.html X
C: > dev > xampp > htdocs > misphp > <> led.html > html > head > meta
1  <!DOCTYPE html><html Lang='en'><head><meta charset='UTF-8'>
2  <meta name='viewport' content='width=device-width, initial-scale=1.0'>
3  <title>ControlLeds</title></head>
4  <body style='font-family: Century gothic; width: 800;'><center>
5  <div style='box-shadow: 0px 0px 20px 8px rgba(0,0,0,0.22);
6  padding: 20px; width: 300px; display: inline-block; margin: 30px;'>
7  <h1>Control Leds</h1>
8
9  <form action="LedsBackend.php">
10
11  <p>Prender LEDS con los botones: </p>
12
13  <input type="hidden" name=1 value= 0> <br/>
14  <input type="checkbox" name=1 value= 1> LED 1 <br/>
15
16  <input type="hidden" name=2 value= 0> <br/>
17  <input type="checkbox" name=2 value= 1> LED 2 <br/>
18
19  <input type="hidden" name=3 value= 0> <br/>
20  <input type="checkbox" name=3 value= 1> LED 3 <br/>
21
22  <input type="hidden" name=4 value= 0> <br/>
23  <input type="checkbox" name=4 value= 1> LED 4 <br/>
24
25  <p><input type="submit" value="Enviar" </p>
26  </form>
27  </html>
28
```

LedsBackend.php:

En php se utiliza una matriz `$_REQUEST` que recibe los los valores del html en este caso.



```
LedsBackend.php X
C: > dev > xampp > htdocs > misphp > LedsBackend.php
1  <?php
2  header('Location: /misphp/led.html');
3  print "<pre>";
4  print_r($_REQUEST);
5
```

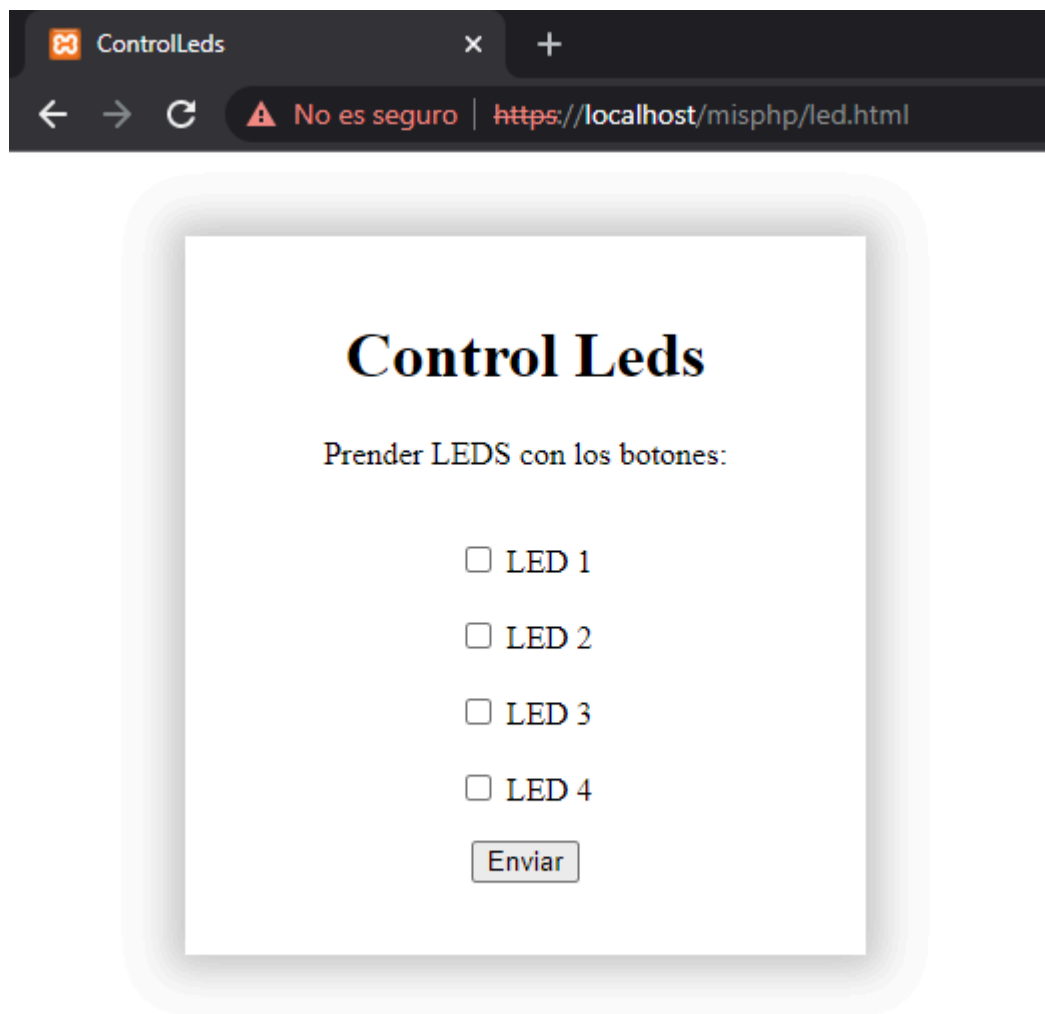
Se crean las variables \$status para almacenar cada array del \$_REQUEST y si ese valor es igual a 1, se almacena el valor 1 o 0 si no lo es.

```
5
6  print "</pre>\n";
7
8  $status1 = $_REQUEST['1'];
9  if ($status1 == 1)
10 {
11     $status1 = 1;
12 }
13
14 else
15 {
16     $status1 = 0;
17 }
18
19 $status2 = $_REQUEST['2'];
20 if ($status2 == 1)
21 {
22     $status2 = 1;
23 }
24
25 else
26 {
27     $status2 = 0;
28 }
29
30 $status3 = $_REQUEST['3'];
31 if ($status3 == 1)
32 {
33     $status3 = 1;
34 }
35
36 else
37 {
38     $status3 = 0;
39 }
40
41 $status4 = $_REQUEST['4'];
42 if ($status4 == 1)
43 {
44     $status4 = 1;
45 }
46
47 else
48 {
49     $status4 = 0;
50 }
```

Cuando termina de almacenar los status, se utiliza la función fopen() para crear, abrir y editar archivos.

Al final se cierra y guarda el archivo con la función fclose.

```
51
52  $Fichero = fopen("Status.info","w");
53
54  $arrayLeds = array ($status1, $status2,$status3, $status4);
55
56  $texto = <<<_END
57  $arrayLeds[0]$arrayLeds[1]$arrayLeds[2]$arrayLeds[3]
58  _END;
59
60  fputs ($Fichero, $texto);
61  fclose ($Fichero);
62
63  ?>
```

Interfaz web:

ledsRemotosWeb.ino:

```
ledsRemotosWeb
// Esta es la librería para utilizar las funciones de red del ESP8266
#include <ESP8266WiFi.h>
#include <stdio.h>
#include <iostream>
#include <string>

#define ledPin1 D5
#define ledPin2 D6
#define ledPin3 D7
#define ledPin4 D8

const char* ssid = "alfacharly"; // Rellena con el nombre de tu red WiFi
const char* password = "H90zGM364Md"; // Rellena con la contraseña de la red WiFi

const char* host = "192.168.1.6";
const char* url_test = "https://192.168.1.6/misphp/Status.info";

String line;
String Led1;
String Led2;
String Led3;
String Led4;
String uno = "1";
```

- Librería para utilizar las funciones de red del ESP 8266.
- Librerías para el manejo de cadenas de caracteres.

- Asignamos a variables los pines que utilizaremos.

URL del archivo Status.info

- Variables globales que se utilizarán para manejar el contenido del archivo Status.info
Y traducir su información a acciones sobre los leds.

Void Setup():

```
void setup()
{
    Serial.begin(115200);
    pinMode(ledPin1, OUTPUT); // Se declara este LED y los siguientes
    pinMode(ledPin2, OUTPUT);
    pinMode(ledPin3, OUTPUT);
    pinMode(ledPin4, OUTPUT);
    digitalWrite(ledPin1, LOW); // Apaga los LEDs
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW);
    digitalWrite(ledPin4, LOW);
    delay(10);
    // Conectamos a la red WiFi
    Serial.println(url_test);
    Serial.println();
    Serial.println();
    Serial.print("Conectandose a: ");
    Serial.println(ssid);

    /* Configuramos el ESP8266 como cliente WiFi. Si no lo hacemos
       se configurará como cliente y punto de acceso al mismo tiempo */
    WiFi.mode(WIFI_STA); // Modo cliente WiFi
    WiFi.begin(ssid, password);

    // Esperamos a que estemos conectados a la red WiFi
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi conectado");
    Serial.println("Direccion IP: ");
    Serial.println(WiFi.localIP()); // Mostramos la IP
    Serial.println("//-----");
}
```

Void Loop():

```
void loop() {  
    Serial.print("conectando a ");  
    Serial.println(host);  
    // Creamos el cliente  
    WiFiClient client;  
    const int httpPort = 80; // Puerto HTTP  
  
    if (!client.connect(host, httpPort)) {  
        // ¿hay algún error al conectar?  
        Serial.println("Ha fallado la conexión, todo mal che !");  
        return;  
    }  
  
    Serial.print("URL de la petición: ");  
    Serial.print(host);  
    Serial.print(":");  
    Serial.print(httpPort);  
    Serial.println(url_test);  
  
    Serial.println("//-----");  
    // Enviamos la petición  
    String peticionHTTP= "GET /misphp/Status.info";  
    client.println(peticionHTTP);  
    unsigned long timeout = millis();  
    while (client.available() == 0) {  
        if (millis() - timeout > 5000) {  
            Serial.println(">>> Superado el tiempo de espera !");  
            client.stop();  
            return;  
        }  
    }  
    Serial.println("Mostrando el contenido de Status.info: ");  
  
    // Leemos la respuesta y la enviamos al monitor serie  
    while(client.available()){  
        line = client.readStringUntil('\r');  
        Serial.print(line);  
    }  
}
```

- Con el comando GET
traeremos el contenido del archivo
- Para eso el string del comando debe ser
enviado al cliente como petición.

- Se muestra el
contenido del archivo
en el monitor serial

```

Serial.println();
Serial.println("//-----");
Serial.print("Este será el estado del led 1: ");
Led1=line[0];
Serial.println(Led1);
Serial.print("Este será el estado del led 2: ");
Led2=line[1];
Serial.println(Led2);
Serial.print("Este será el estado del led 3: ");
Led3=line[2];
Serial.println(Led3);
Serial.print("Este será el estado del led 4: ");
Led4=line[3];
Serial.println(Led4);

if (Led1.compareTo(unos) == 0) {
  Serial.println("Se prende LED 1");
  digitalWrite(ledPin1, HIGH);
} else {
  Serial.println("Se apaga LED 1");
  digitalWrite(ledPin1, LOW);
}

if (Led2.compareTo(unos) == 0) {
  Serial.println("Se prende LED 2");
  digitalWrite(ledPin2, HIGH);
} else {
  Serial.println("Se apaga LED 2");
  digitalWrite(ledPin2, LOW);
}

if (Led3.compareTo(unos) == 0) {
  Serial.println("Se prende LED 3");
  digitalWrite(ledPin3, HIGH);
} else {
  Serial.println("Se apaga LED 3");
  digitalWrite(ledPin3, LOW);
}

if (Led4.compareTo(unos) == 0) {
  Serial.println("Se prende LED 4");
  digitalWrite(ledPin4, HIGH);
} else {
  Serial.println("Se apaga LED 4");
  digitalWrite(ledPin4, LOW);
}

Serial.println("Próxima revisión en 60s.");
Serial.println("//-----");
delay(60000);
}

```

- Se muestran los valores que tendrán los leds.

- Se asigna el valor de cada espacio del String a una variable para cada led.

Ejemplo:

led1=line[0]; // Siendo [0] la posición 1 del string que correspondería al led 1.

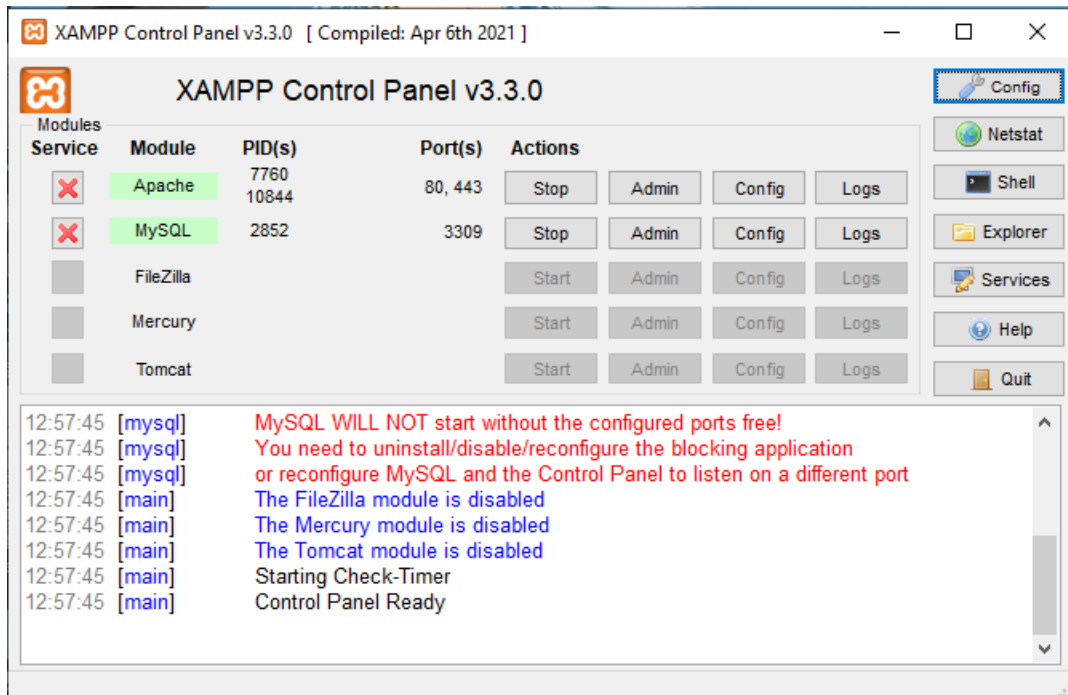
- Aquí finalmente se hace el control necesario para apagar o prender los leds:

- Teniendo una variable para cada led (paso anterior); Se compara cada variable correspondiente a un led con la variable "unos" para prenderlo (en caso de ser iguales) o apagarlo (en caso de no ser iguales).

Con esto finaliza el código del archivo **ledsRemotosWeb.ino**

Prueba:

Iniciamos el servidor Xampp:



Datos enviados:

Control Leds

Prender LEDS con los botones:

☒ LED 1

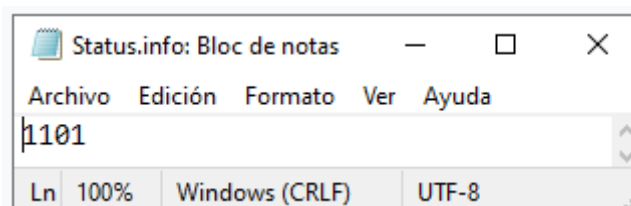
☒ LED 2

☐ LED 3

☒ LED 4

Enviar

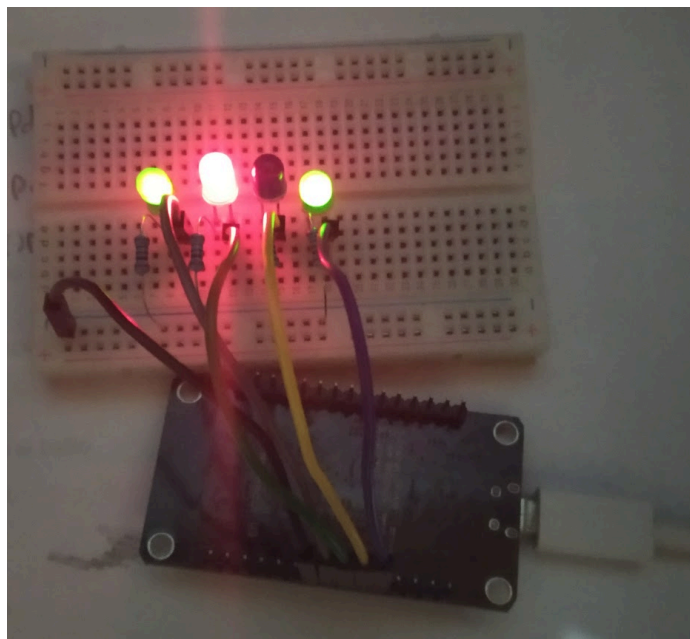
Datos de **status.info** enviados desde el formulario anterior:



Monitor Serie:

```
COM4
.....
WiFi conectado
Direccion IP:
192.168.1.11
//-----
conectando a 192.168.1.6
URL de la petición: 192.168.1.6:80https://192.168.1.6/misphp/Status.info
//-----
mostrando el contenido de Status.info:
1101
//-----
Este será el estado del led 1: 1
Este será el estado del led 2: 1
Este será el estado del led 3: 0
Este será el estado del led 4: 1
Se prende LED 1
Se prende LED 2
Se apaga LED 3
Se prende LED 4
Próxima revisión en 60s.
//-----
```

☒ Autoscroll ☐ Mostrar marca temporal Nueva línea 115200 baudio Limpiar salida

Foto demostrativa:

Se comprueba el funcionamiento óptimo de la implementación.

Video demostrativo:

<https://youtu.be/F6Oxc3In0Ww>