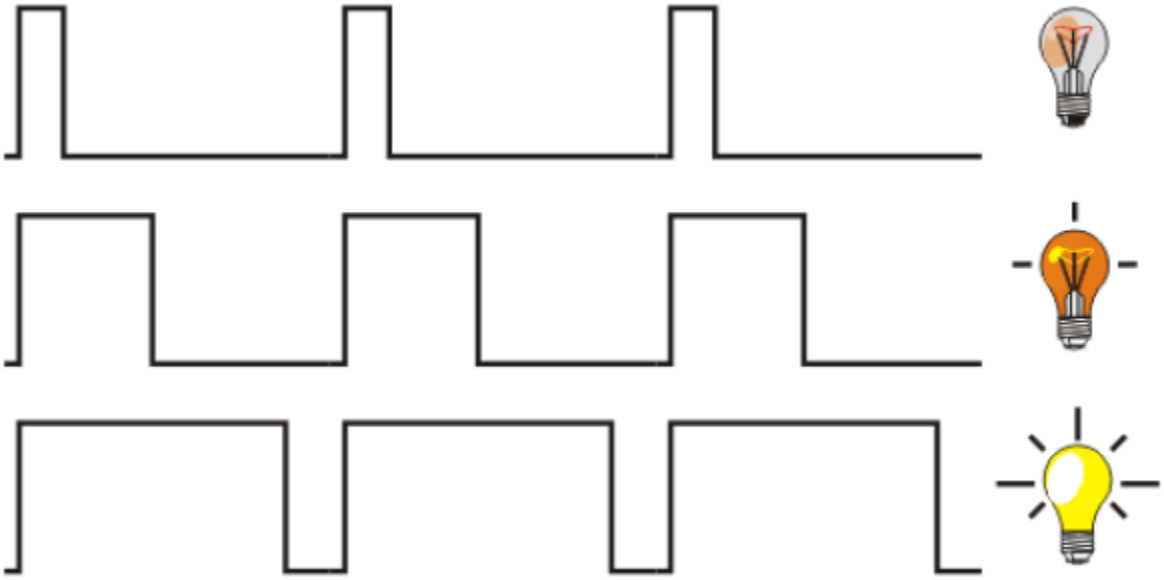
SERVIDOR WEB PARA CONTROL DE INTENSIDAD LUMINOSA MEDIANTE PWM

Diseño de interface gráfica para control con deslizador

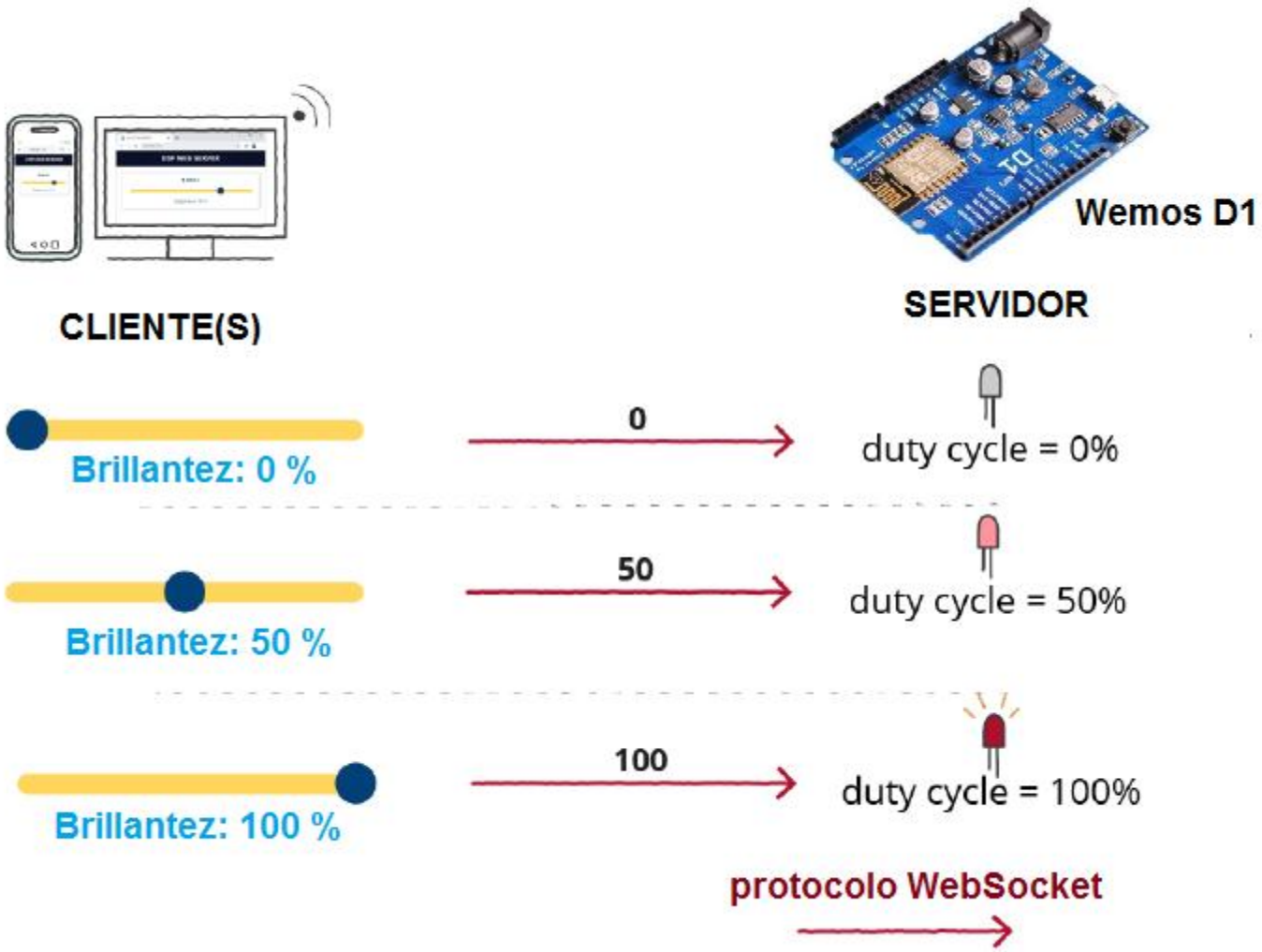
La modulación por ancho o de pulso (en inglés Pulse Width Modulation PWM) es un tipo de señal de voltaje utilizada para enviar información con la finalidad de modificar la cantidad de energía que se envía a una carga.

Esta acción tiene en cuenta la modificación del proceso de trabajo de una señal de tipo periódico.



Desarrollaremos una interface con control de intensidad luminosa en un LED mediante un deslizador.

Mediante el protocolo WebSocket al arrastrar el círculo de color azul del deslizador se obtendrá distinta intensidad luminosa en un LED conectado al GPIO /D4 de la placa Wemos D1:



LISTA DE MATERIALES:

1 Wemos D1 R1.
1 cable USB a USB micro.
1 resistor de 220 Ohms
1 LED de cualquier color
1 protoboard
1 alambre dupont macho-macho color NEGRO.
1 alambre dupont macho-macho color NARANJA.

ARCHIVO main.cpp

```
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include "LittleFS.h"

const char* ssid = "MEGACABLE-rt342de";
const char* password = "S376krlS";
// Crea el objeto AsyncWebServer en el puerto 80
AsyncWebServer server(80);
/* La biblioteca ESPAsyncWebServer incluye un complemento WebSocket que facilita el
manejar las conexiones WebSocket. Crear un objeto AsyncWebSocket llamado ws para
manejar las conexiones en la ruta / ws. */
AsyncWebSocket ws("/ws");

const int ledPin = 4;  // Conexión para el LED a controlar PWM
String ValorDeslizador = "0";
int dutyCycle; // variable para contolar la brillantez del LED
// Inicializa LittleFS
void initFS() {
 if (!LittleFS.begin()) {
 Serial.println("Un error ha ocurrido mientras se montaba LittleFS");
 }
 Serial.println("LittleFS montado exitosamente");
}
// Inicializa WiFi
void initWiFi() {
 WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
 Serial.print("Conectando a la red  WiFi ..");
 while (WiFi.status() != WL_CONNECTED) {
 Serial.print('.');
 delay(1000);
 }
 Serial.println(WiFi.localIP());
}
//**************************************************************************
void handleWebSocketMessage(void *arg, uint8_t *data, size_t len)
/* La función handleWebSocketMessage () es una función de devolución de llamada que
se ejecutará siempre que recibamos nuevos datos de los clientes a través del
protocolo WebSocket. */
 {
 AwsFrameInfo *info = (AwsFrameInfo*)arg;
 if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT) {
 data[len] = 0;
 ValorDeslizador = (char*)data;  // El cliente envía el valor actual del control
 // deslizante al servidor. Entonces, guardamos ese valor en la variable ValorDeslizador.
 dutyCycle = map(ValorDeslizador.toInt(), 0, 100, 0, 1023);
```

```
 /* ValorDeslizador puede ser un valor entre 0 y 100, pero el ciclo de trabajo es un valor
entre 0 y 1023. Entonces, usamos la función map () para convertir desde el rango 0-100
al rango de 0 a 1023.
 */
 Serial.println(dutyCycle); // Envía el valor del ciclo útil al monitor
 /* NOTA:En el monitor se actualiza al valor deslizado por el último cliente
    porque está programado en modo Web socket. Para actualizar en la gráfica
    se debe actualizar la página Web  */
 }
}
void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType  type, void
*arg, uint8_t *data, size_t len) {
 /*Ahora necesitamos configurar un detector de eventos para manejar los diferentes pasos
 asincrónicos del protocolo WebSocket. Este controlador de eventos se puede implementar
 definiendo onEvent () de esta manera: */
 // El argumento de tipo representa el evento que ocurre. Puede tomar los siguientes valores:
 switch (type) {
 case WS_EVT_CONNECT: // cuando un cliente ha iniciado sesión
 Serial.printf("WebSocket cliente #%u conectado de %s\n", client->id(), client-
>remoteIP().toString().c_str());
 break;
 case WS_EVT_DISCONNECT: // cuando un cliente se desconecta
 Serial.printf("WebSocket cliente #%u desconectado\n", client->id());
 break;
 case WS_EVT_DATA: // cuando se recibe un paquete de datos del cliente
 handleWebSocketMessage(arg, data, len);
 break;
 case WS_EVT_PONG: // En respuesta a una solicitud de ping
 /* Ping es un comando o una herramienta de diagnóstico que permite hacer una verificación
  del estado de una determinada conexión o host local.
Es un acrónimo para Packet Internet Groper, lo que literalmente significa
 “buscador de paquetes en redes”. Se trata de un comando que permite verificar el estado de
 una conexión para determinar si una dirección IP específica o host es accesible desde la red o no. */
 case WS_EVT_ERROR: // cuando se recibe un error del cliente.
 break;
 }
}
void initWebSocket() { // Inicializa el protocolo WebSocket
 ws.onEvent(onEvent);
 server.addHandler(&ws);
}
void setup() {
 Serial.begin(115200);
 pinMode(ledPin, OUTPUT);
 initFS(); // se inicializa Littlefs
 initWiFi(); // inicializa Wi Fi
 initWebSocket(); // inicializa el protocolo Websocket
/* Las siguientes líneas manejan lo que sucede cuando recibe una
 solicitud en la raíz (/) URL (dirección IP de ESP8266).
```

```
 Cuando recibe esa solicitud, envía el texto HTML guardado en el archivo
 index.html a construir la página web.  */
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
 request->send(LittleFS, "/index.html", "text/html");
 });
  /* Cuando el archivo HTML se cargue en su navegador, solicitará el CSS,
Archivos JavaScript y favicon. Estos son archivos estáticos guardados en
el mismo directorio (SPIFFS o LittleFS). Entonces, podemos simplemente
agregar la siguiente línea para servir archivos en un directorio cuando
sea solicitado por la URL raíz. Servirá los archivos CSS y favicon
automáticamente (también se agrega una imagen en este ejemplo) */
 server.serveStatic("/", LittleFS, "/");
 server.on("/ValorActual", HTTP_GET, [](AsyncWebServerRequest *request){
 /* Cuando se abre la página web por primera vez, realiza una solicitud en
 la ruta /ValorActual para solicitar el ValorDeslizador actual.
 ValorActual vincula aquí con script.js */
 request->send(200, "/text/plain", String(ValorDeslizador).c_str());
 /* Simplemente puede enviar una cadena con esa información
(tercer argumento de la función send ()). */
 });
 // Inicia el servidor
 server.begin();
}
void loop() {
 analogWrite(ledPin, dutyCycle);
 //El LED se controlará físicamente en el loop()
 ws.cleanupClients();
 /* Los navegadores a veces no cierran correctamente
 la conexión WebSocket, incluso cuando se llama a la función close () en
 JavaScript. Esto eventualmente agotará los recursos del servidor web y hará
 que el servidor se bloquee. Llamar periódicamente a la función
 cleanupClients () desde el bucle principal () limita el número de clientes
  cerrando el cliente más antiguo cuando se ha superado el número máximo
  de clientes. Esto se puede llamar en cada ciclo, sin embargo, si desea
  usar menos energía, entonces es suficiente llamar con tan poca frecuencia
  como una vez por segundo. */
}
```

ARCHIVO index.html

```
<!DOCTYPE html>
<html>
<head>
<title>PWM</title>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="icon" type="image/png" href="favicon.png">
 <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/al
l.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="an
onymous">
 <link rel="stylesheet" type="text/css" href="style.css">
 <!-- debe mantenerse el nombre stylesheet-->
</head>
<body>
   <div class="marquesina">
      <h1>SERVIDOR WEB CONTROL DE INTENSIDAD POR PWM</h1>
      </div>
      <div class="contenido">
      <div class="cuadricula-tarjeta">
      <div class="tarjeta">
       <!--muestra un título para la tarjeta como en los proyectos anteriores. Contiene
       el texto "GPIO 4". -->
      <p class="titulo-tarjeta"><i class="fas fa-lightbulb"></i> GPIO 4</p>
      <p class="switch">
<!--Para crear un control deslizante en HTML, usa la etiqueta <input>.
La etiqueta <input> especifica un campo donde el usuario puede ingresar datos.
Existe una amplia variedad de tipos de entrada. Para definir un control deslizante,
use el atributo de tipo con el valor del rango.
En un control deslizante, también debe definir el mínimo y el máximo
rango usando los atributos min y max (en este caso, 0 y 100, respectivamente).
También necesita definir otros atributos como:
• el atributo de paso especifica el intervalo entre números válidos. En nuestro caso,
se establece en 1;
• la clase para diseñar el deslizador (class = "deslizador");
• la identificación para que podamos manipular el valor del control deslizante usando JavaScript
(id = "DeslizadorPWM");
• el atributo oninput para llamar a una función (updateSliderPWM (this)) cuando
mueva el control deslizante. Esta función envía el valor actual del control deslizante a
través del protocolo  WebSocket protocolo al cliente. La palabra clave this se refiere
al elemento deslizante HTML. El control deslizante está dentro de un párrafo con el
nombre de la clase del interruptor. Entonces, aquí están las etiquetas para
crear el control deslizante

Eventos oninput vs onchange:
El evento oninput ocurre cuando un elemento recibe una entrada del usuario. Este evento ocurre
cuando
se cambia el valor de un elemento <input> (arrastre el control deslizante).
```

```
Este evento es similar al evento onchange. La diferencia es que el evento oninput
ocurre inmediatamente después de que el valor de un elemento ha cambiado (el evento ocurre
mientras arrastra el control deslizante), mientras que onchange se produce cuando el elemento
pierde el foco, después de que el contenido ha sido cambiado (después de soltar el control
deslizante).
Para comprender mejor el diferencia, puedes probar ambos sentidos y ver cuál te gusta más.
        -->
        <input type="range" oninput="updateSliderPWM(this)" id="DeslizadorPWM" min=
       "0" max="100" step="1" value="0" class="deslizador">
        </p>
<!--Por último, hay un párrafo con una etiqueta de extensión <span id = "textoValorDeslizador">
</span>,
para que podamos insertar el valor actual del control deslizante en ese párrafo. -->
        <p class="estado">Brillantez: <span id="textoValorDeslizador"></span> &percnt;</p>
        </div>
        </div>
        </div>
        <p></p>
        <script src="script.js"></script>
      <IMG SRC="PWM.png" ALIGN = MIDDLE>
</body>
</html>
```

ARCHIVO style.css

```
/* Utilizamos los mismos estilos que se utilizaron en proyectos anteriores.
Se han agreado nuevas instrucciones para formatear el rango del control
deslizante */
html {
    font-family: Arial, Helvetica, sans-serif;
    display: inline-block;
    text-align: center;
   }
   h1 {
    font-size: 1.8rem;
    color: white;
   }
   p {
    font-size: 1.4rem;
   }
   .marquesina {
    overflow: hidden;
    background-color: #0A1128;
   }
   body {
    margin: 0;
   }
```

```
   .contenido {
    padding: 30px;
}
.cuadricula-tarjeta {
 max-width: 700px;
 margin: 0 auto;
 display: grid;
 grid-gap: 2rem;
 grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
}
.tarjeta {
 background-color: white;
 box-shadow: 2px 2px 12px 1px rgba(140,140,140,.5);
}
.titulo-tarjeta {
 font-size: 1.2rem;
 font-weight: bold;
 color: #034078
}
.estado {
 font-size: 1.2rem;
 color: #1282A2;
}
.deslizador {
 -webkit-appearance: none;
 /* Establecer -webkit-apariencia en ninguno (none) anula los estilos CSS
 predeterminados aplicados al deslizador */
 margin: 0 auto;
 /*
Establecer el margen en 0 alinea automáticamente el control deslizante
dentro de su contenedor principal */
 width: 100%;
 /*El ancho del control deslizante se establece en 100% y la altura
  en 15px. */
 height: 15px;
 border-radius: 10px;
 /* El radio del borde es establecido en 10px. */
 background: #FFD65C;
 /* Establece el color de fondo para el control deslizante */
 outline: none;
 /* establece el contorno en ninguno. */
}
/*
    Establece las propiedades de apariencia de -webkit en none para anular
    las propiedades predeterminadas
*/
.deslizador::-webkit-slider-thumb {
 -webkit-appearance: none;
 appearance: none;
```

```
 width: 30px;
 /*
Establece un ancho, alto y radio de borde específicos para el controlador.
Establece lo mismo ancho y alto con un radio de borde del 50% crea un círculo */
 height: 30px;
 border-radius: 50%;
 background: #034078;
 cursor: pointer;
}

/*Luego, formatea el control deslizante. Utiliza -webkit- para Chrome, Opera, Safari y Edge
navegadores web y -moz- para Firefox */
.deslizador::-moz-range-thumb
{
 width: 30px;
 height: 30px;
 border-radius: 50% ;
 background: #034078;
 cursor: pointer;
}
.switch {
 padding-left: 5%;
 padding-right: 5%;
}
```

ARCHIVO script.js

```
/* Este es responsable de inicializar una conexión WebSocket con el servidor,
solicita al servidor el valor actual del control deslizante tan pronto como
la interfaz web está completamente cargada en el navegador y maneja el intercambio
de datos a través del protocolo WebSocket. */

/*La puerta de enlace es el punto de entrada a la interfaz de WebSocket. */
var PuertaDeEnlaceAlSocket = `ws://${window.location.hostname}/ws`;
/* window.location.hostname obtiene la dirección de la página actual
(la IP del servidor web)
 Crear una nueva variable global llamada miWebSocket. */
var miWebSocket;
/*  Se agrega un detector de eventos que llamará a la función onload
cuando se cargue la página web. */
window.addEventListener('load', onload);
function onload(event) {
    /* La función onload () llama a la función initWebSocket () para
 inicializar una conexión de  WebSocket con el servidor y la función
 initButton () para agregar detectores de eventos a los botones. */
 initWebSocket();
 obtenerValorActual();
```

```
}
function initWebSocket() {
    /* La función initWebSocket () inicializa una conexión WebSocket en
PuertaDeEnlaceAlSocket definida anteriormente. También asignamos varias funciones
de devolución de llamada para cuando la conexión del WebSocket se abre,
se cierra o cuando se recibe un mensaje. */
 console.log('Intentando abrir una conexión WebSocket…');
 miWebSocket = new WebSocket(PuertaDeEnlaceAlSocket);
 miWebSocket.onopen = onOpen;
 miWebSocket.onclose = onClose;
 miWebSocket.onmessage = onMessage;
}
function onOpen(event) {  // función  de devolución de llamada para
    //cuando la conexión del WebSocket se abre
 console.log('Conexión abierta');
}
function onClose(event) {   // función  de devolución de llamada para
    //cuando la conexión del WebSocket se cierra
 console.log('Conexión cerrada');
 setTimeout(initWebSocket, 2000);
}
function onMessage(event) {   // función  de devolución de llamada para
    //cuando se recibe un mensaje
 console.log(event.data);
 }
function obtenerValorActual() {
/* La función obtenerValorActual () realiza una solicitud en la URL /ValorActual y
maneja la respuesta. Se vincula con main.cpp*/
 var xhr = new XMLHttpRequest();
 xhr.onreadystatechange = function() {
 if (this.readyState == 4 && this.status == 200) {
 document.getElementById("DeslizadorPWM").value = this.responseText;
 document.getElementById("textoValorDeslizador").innerHTML = this.responseText;
 }
 };
 xhr.open("GET", "/ValorActual", true);
 xhr.send();
 /* Cuando el servidor recibe esa solicitud, responde con el valor del control deslizante.
Establecemos la posición actual del control deslizante en ese valor. El ID del control
deslizante es DeslizadorPWM  y es posible establecer su valor usando .value como
se realizó previamente con:
document.getElementById("pwmSlider").value = this.responseText;

Finalmente, se agrega el valor al párrafo con el id textoValorDeslizador
(vinculado en index.html)
document.getElementById("textoValorDeslizador").innerHTML = this.responseText;
*/
}
function updateSliderPWM(element) {
```

```
// esta función es llamada cuando se arrastra el deslizador
 var ValorDeslizador = document.getElementById("DeslizadorPWM").value;
 /*La función obtiene el valor actual del control deslizante y lo guarda en
 la variable ValorDeslizador. */
 document.getElementById("textoValorDeslizador").innerHTML = ValorDeslizador;
 /* Luego, actualiza el valor del control deslizante  ValorDeslizador en el párrafo con
  el id de textoValorDeslizador */
 console.log(ValorDeslizador);
 /* presenta el valor del control deslizante en la consola para fines de depuración */
 miWebSocket.send(ValorDeslizador);
 /* Finalmente, envía el valor del control deslizante ValorDeslizador al cliente mediante
 el protocolo WebSocket */
}
```

ARCHIVO platformio.ini

Después de agregar las libraries necesarias para este proyecto, el archivo de configuración platformio.ini queda de la siguiente manera:

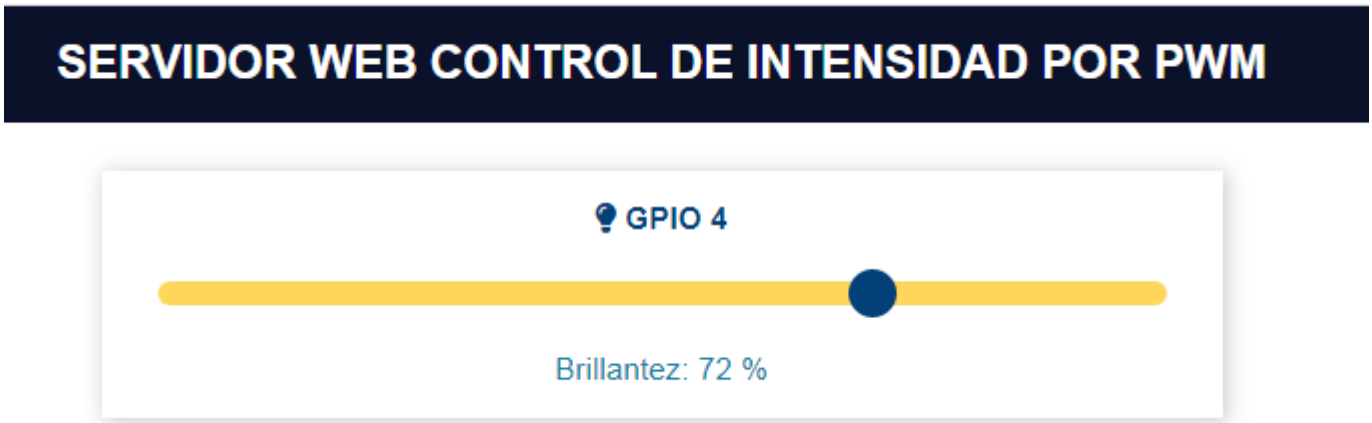
```
[env:d1]
platform = espressif8266
board = d1
framework = arduino
monitor_speed = 115200
lib_deps =
    ESP Async WebServer
        ottowinter/ESPAsyncTCP-esphome@^1.2.3
        ottowinter/ESPAsyncWebServer-esphome@^1.2.7
board_build.filesystem = littlefs
```

CONEXIONES:

Conectar un LED de cualquier color, con un resistor en serie de 220 Ohms hacia tierra y la otra terminal del LED (positiva) conectarla al GPIO /D4 de Wemos D1.

PRUEBAS:

Teclear en un navegador la IP que se proporciona en el programa monitor, y deslizar el control para observar cómo cambia la intensidad de luz en el LED.



Invitar a otro cliente para que desde su equipo mueva el círculo de color azul, ante esta acción verificar en el monitor de Platformio que se actualiza el valor ajustado por el último cliente.