**Rgbsocketio**

**serverAnode**

// Crea un servidor HTTP que escucha en el puerto 8080. 'http' es una instancia de la clase HTTPServer.

// Esta instancia gestiona todas las solicitudes HTTP dirigidas al servidor en el puerto 8080.

const http = require('http').createServer(handler);

// 'fs' es una instancia de la clase FileSystem, usada para operaciones de lectura de archivos.

// Usado para acceder y leer archivos desde el disco, específicamente para servir páginas web a los clientes.

var fs = require('fs');

// 'io' es una instancia de la clase WebSocketServer, configurada para manejar comunicación en tiempo real sobre el servidor HTTP.

// Permite la comunicación en tiempo real con los clientes conectados, usada para recibir y enviar comandos de control del LED RGB.

var io = require('socket.io')(http);

// 'gpio' es una instancia de la clase GPIO, usada para controlar el hardware de los pines GPIO mediante PWM.

// Inicializa los pines GPIO designados para controlar un LED RGB mediante señales PWM.

const gpio = require("@iiot2k/gpiox");

// Configura los pines GPIO para cada color del LED RGB.

// Inicializa los pines como salidas PWM y configura la frecuencia y el ciclo de trabajo inicial.

const ledRed = 22; // Atributo del objeto GPIO para el pin del LED rojo

const ledGreen = 17; // Atributo del objeto GPIO para el pin del LED verde

const ledBlue = 27; // Atributo del objeto GPIO para el pin del LED azul

gpio.init\_pwm(ledRed, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

gpio.init\_pwm(ledGreen, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

gpio.init\_pwm(ledBlue, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

// Activa el servidor para aceptar conexiones HTTP en el puerto especificado.

// Comienza a escuchar las solicitudes entrantes en el puerto 8080.

http.listen(8080);

// Función para manejar las solicitudes HTTP.

// Lee el archivo 'rgb.html' y lo envía al cliente o devuelve un error 404 si el archivo no se encuentra.

function handler(req, res) {

fs.readFile(\_\_dirname + '/public/rgb.html', function(err, data) {

if (err) {

res.writeHead(404, {'Content-Type': 'text/html'}); // Respuesta con estado 404

return res.end("404 Not Found");

}

res.writeHead(200, {'Content-Type': 'text/html'}); // Respuesta con estado 200

res.write(data); // Envío del contenido del archivo

return res.end();

});

}

// Establece la gestión de eventos para la conexión WebSocket.

// Escucha los eventos de 'rgbLed' para recibir datos del cliente y ajustar el PWM de los LEDs según esos datos.

io.sockets.on('connection', function (socket) {

socket.on('rgbLed', function(data) {

console.log(data); // Muestra los datos recibidos de la interfaz web

// Ajusta el color del LED RGB según los datos recibidos

redRGB = 255 - parseInt(data.red);

greenRGB = 255 - parseInt(data.green);

blueRGB = 255 - parseInt(data.blue);

let redScaled = scaleTo100(redRGB);

let greenScaled = scaleTo100(greenRGB);

let blueScaled = scaleTo100(blueRGB);

gpio.set\_pwm(ledRed, PWM\_FREQ, redScaled);

gpio.set\_pwm(ledGreen, PWM\_FREQ, greenScaled);

gpio.set\_pwm(ledBlue, PWM\_FREQ, blueScaled);

});

});

// Manejo de la señal SIGINT para una terminación segura y limpieza de los recursos GPIO.

// Desinicializa los pines GPIO y detiene el servidor antes de salir del proceso.

process.on('SIGINT', function () {

gpio.deinit\_gpio(ledRed);

gpio.deinit\_gpio(ledGreen);

gpio.deinit\_gpio(ledBlue);

process.exit();

});

Diagrama

Descripción generada automáticamente

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**serverCathode**

// Crear un servidor HTTP que escucha en el puerto 8080. 'http' es una instancia de la clase HTTPServer.

// Esta instancia gestiona todas las solicitudes HTTP dirigidas al servidor en el puerto 8080.

const http = require('http').createServer(handler);

// 'fs' es una instancia de la clase FileSystem, usada para operaciones de lectura de archivos.

// Usado para acceder y leer archivos desde el disco, específicamente para servir páginas web a los clientes.

var fs = require('fs');

// 'io' es una instancia de la clase WebSocketServer, configurada para manejar comunicación en tiempo real sobre el servidor HTTP.

// Permite la comunicación en tiempo real con los clientes conectados, usada para recibir y enviar comandos de control del LED RGB.

var io = require('socket.io')(http);

// 'gpio' es una instancia de la clase GPIO, usada para controlar el hardware de los pines GPIO mediante PWM.

// Inicializa los pines GPIO designados para controlar un LED RGB mediante señales PWM.

const gpio = require("@iiot2k/gpiox");

// Configura los pines GPIO para cada color del LED RGB.

// Inicializa los pines como salidas PWM y configura la frecuencia y el ciclo de trabajo inicial.

const ledRed = 22; // Atributo del objeto GPIO para el pin del LED rojo

const ledGreen = 17; // Atributo del objeto GPIO para el pin del LED verde

const ledBlue = 27; // Atributo del objeto GPIO para el pin del LED azul

gpio.init\_pwm(ledRed, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

gpio.init\_pwm(ledGreen, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

gpio.init\_pwm(ledBlue, gpio.GPIO\_MODE\_PWM, PWM\_FREQ, PWM\_DUTY\_INIT);

// Activa el servidor para aceptar conexiones HTTP en el puerto especificado.

// Comienza a escuchar las solicitudes entrantes en el puerto 8080.

http.listen(8080);

// Función que maneja las solicitudes HTTP, enviando la página 'rgb.html' o devolviendo un error 404 si no se encuentra.

function handler(req, res) {

fs.readFile(\_\_dirname + '/public/rgb.html', function(err, data) {

if (err) {

res.writeHead(404, {'Content-Type': 'text/html'}); // Respuesta con estado 404

return res.end("404 Not Found");

}

res.writeHead(200, {'Content-Type': 'text/html'}); // Respuesta con estado 200

res.write(data); // Envío del contenido del archivo

return res.end();

});

}

// Establece la gestión de eventos para la conexión WebSocket.

// Escucha los eventos de 'rgbLed' para recibir datos del cliente y ajustar el PWM de los LEDs según esos datos.

io.sockets.on('connection', function (socket) {

socket.on('rgbLed', function(data) {

console.log(data); // Muestra los datos recibidos de la interfaz web

// Ajusta el color del LED RGB según los datos recibidos

redRGB = parseInt(data.red);

greenRGB = parseInt(data.green);

blueRGB = parseInt(data.blue);

let redScaled = scaleTo100(redRGB);

let greenScaled = scaleTo100(greenRGB);

let blueScaled = scaleTo100(blueRGB);

gpio.set\_pwm(ledRed, PWM\_FREQ, redScaled);

gpio.set\_pwm(ledGreen, PWM\_FREQ, greenScaled);

gpio.set\_pwm(ledBlue, PWM\_FREQ, blueScaled);

});

});

// Manejo de la señal SIGINT para una terminación segura y limpieza de los recursos GPIO.

// Desinicializa los pines GPIO y detiene el servidor antes de salir del proceso.

process.on('SIGINT', function () {

gpio.deinit\_gpio(ledRed);

gpio.deinit\_gpio(ledGreen);

gpio.deinit\_gpio(ledBlue);

process.exit();

});

Tabla

Descripción generada automáticamente con confianza media

**Imagen que contiene Diagrama

Descripción generada automáticamente**