**Nodetest4**

// Programa para controlar un LED y leer un botón mediante Node.js y GPIO en una Raspberry Pi.

//

// Este script establece un servidor HTTP y un WebSocket para permitir el control de un LED y la monitorización

// de un botón a través de una página web. El servidor envía y recibe comandos para controlar el LED y leer el estado del botón.

// Importar el módulo http para crear el servidor HTTP.

const http = require('http');

// Importar el módulo fs (File System) para acceder al sistema de archivos.

const fs = require('fs');

// Importar socket.io para manejar la comunicación WebSocket.

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

// Importar la biblioteca onoff para interactuar con los GPIO de la Raspberry Pi.

const Gpio = require('onoff').Gpio;

// Crear una instancia de Gpio para controlar el GPIO pin 4 como salida.

const LED = new Gpio(4, 'out');

// Crear una instancia de Gpio para usar el GPIO pin 17 como entrada.

const pushButton = new Gpio(17, 'in', 'both');

// Establecer el servidor para escuchar en el puerto 8080.

http.listen(8080);

// Función que maneja las solicitudes HTTP.

// Sirve el archivo index.html o responde con un error 404 si no se encuentra.

function handler (req, res) {

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

if (err) {

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

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

}

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

res.write(data);

return res.end();

});

}

// Configurar la conexión WebSocket para manejar eventos de conexión.

// Inicializa la conexión WebSocket y maneja los eventos de conexión y desconexión.

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

var lightvalue = 0; // Almacena el estado actual del LED.

// Vigila los cambios de estado del botón.

// Emite el estado actual del botón a todos los clientes conectados a través de WebSocket.

pushButton.watch(function (err, value) {

if (err) {

console.error('There was an error', err);

return;

}

lightvalue = value;

socket.emit('light', lightvalue);

});

// Recibe el estado del LED desde el cliente y actualiza el GPIO pin 4 si es necesario.

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

lightvalue = data;

if (lightvalue != LED.readSync()) {

LED.writeSync(lightvalue);

}

});

});

// Manejar la señal SIGINT para limpiar los recursos GPIO al cerrar el programa.

// Asegura que el LED se apague y que los recursos GPIO se liberen correctamente.

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

LED.writeSync(0);

LED.unexport();

pushButton.unexport();

process.exit();

});

Diagrama, Esquemático

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente