**Nodetest5**

// Crea un servidor HTTP que escucha en el puerto 8080

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

// Utiliza el módulo del sistema de archivos para leer y servir archivos HTML

var fs = require('fs');

// Establece una conexión WebSocket sobre el servidor HTTP

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

// Configura los pines GPIO para control de entrada y salida

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

// Función para manejar solicitudes HTTP y servir la página web

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();

});

}

// Gestiona eventos de WebSocket para controlar el LED y leer el estado del botón

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

// Monitorea el estado del botón y emite cambios a los clientes

console.log("conectado");

var state = gpio.watch\_gpio(INPUT\_PIN, gpio.GPIO\_MODE\_INPUT\_PULLDOWN, DEBOUNCE\_US, gpio.GPIO\_EDGE\_BOTH,

(state, edge) => {

console.log("port", INPUT\_PIN, "state", state, "edge", edge);

socket.emit('light', state);

});

// Responde a mensajes del cliente para alternar el LED

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

gpio.toggle\_gpio(OUTPUT\_PIN);

});

});

// Limpieza de recursos GPIO cuando se interrumpe el proceso

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

gpio.set\_gpio(25, 0);

gpio.deinit\_gpio(25);

gpio.deinit\_gpio(INPUT\_PIN);

process.exit();

});

Diagrama

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media