

Informe Trabajo Final

Programación Orientada a Objetos

Alumno: Amenta Joaquin F.

Docentes:

Profesor Titular: Dr. Roberto Marcelo Hidalgo

Jefe de Trabajos Prácticos: Ing. Pablo Daniel Spennato.

Ayudante de Primera: Ing. Marco Viola.

Asignatura: Programación Orientada a Objetos

Ciclo lectivo: 2020

Universidad Nacional de Mar del Plata, Facultad de Ingeniería

Índice

Índice	1
Introducción	2
Desarrollo	3
Propuesta de Trabajo	3
Diseño e Implementación	3
Arquitectura	5
Código Fuente	7
Instalación	7
Android Development	8
Workflow Demo	9
Demo de configuración NodeMCU	12
Referencias	14

Introducción

La carrera de Ingeniería en Computación de la Facultad de Ingeniería, Universidad Nacional de Mar del Plata, cuenta con la materia Programación Orientada a Objetos, perteneciente al 3er año de cursada. La misma trata los conceptos básicos del paradigma de programación orientado a objetos.

A modo de trabajo final, la cátedra propone realizar un proyecto de desarrollo de software. El presente trabajo tiene como objetivo mostrar el desarrollo e implementación de dicho trabajo, permitiendo así que este quede como antecedente y otros alumnos o docentes puedan utilizarlo como referencia.

A continuación, se desarrollarán los siguientes temas:

- Propuesta de Trabajo
- Diseño e Implementación
- Arquitectura
- Código Fuente
- Workflow Demo

Desarrollo

Propuesta de Trabajo

La propuesta de trabajo de la cátedra fue extender un proyecto realizado por otro alumno (Ezequiel Salagoity). En base a su trabajo final se propuso desarrollar un sistema que permita interactuar con sensores y componentes conectados a un *NODEMCU* a través del protocolo de comunicación MQTT, el cual debía poder ser utilizado en varias plataformas como por ejemplo android y web.

Como resultado, el sistema de software que se desarrolló fue una aplicación multiplataforma la cual permitía medir la temperatura y humedad en tiempo real, encender y apagar un led y cambiar la intensidad de un dimmer. Además, daba la posibilidad de configurar la conexión de acuerdo a las preferencias del usuario, accediendo desde cualquier dispositivo que lo soporte.

El alcance de este proyecto incluye el desarrollo del código para los dispositivos de hardware, el desarrollo de la aplicación multiplataforma, la documentación y correcta implementación del mismo. No se incluye historial de versiones ni justificación o análisis de los proyectos previos realizados por otros alumnos.

Diseño e Implementación

A continuación se detallan los dispositivos y las tecnologías utilizadas tanto para el software como para el hardware. Así como también el protocolo de comunicación.

Para el hardware se utilizó una placa NodeMCU 8266, el sensor DHT11, un led, un dimmer, y un botón a presión.

NodeMCU es un firmware open source basado en LUA para el chip wifi ESP8266. El firmware NodeMCU viene con la placa y kit de desarrollo ESP8266. El ESP8266 es un chip Wi-Fi de bajo costo desarrollado por Espressif Systems con protocolo TCP / IP.

El software se desarrolló en JavaScript como lenguaje principal con el framework *Quasar*, ya que se especializa en desarrollo multiplataforma. Este framework además utiliza *Vue.js*. La instalación de los paquetes necesarios se realizó por medio de npm (Node Package Manager) perteneciente a *node.js*. La comunicación entre el dispositivo NodeMCU y las aplicaciones se realiza mediante el Protocolo MQTT.

Explicación de las tecnologías utilizadas:

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Vue es un framework de JavaScript progresivo para crear interfaces de usuario. Vue está diseñado desde cero para ser adoptable gradualmente. La biblioteca principal se centra solo en la capa de vista y es fácil de recoger e integrar con otras bibliotecas o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar aplicaciones sofisticadas de una sola página cuando se usa en combinación con herramientas modernas y bibliotecas de soporte.

Quasar es un framework open source basado en Vue.js con licencia del MIT, que le permite a los desarrolladores web crear rápidamente sitios web / aplicaciones responsivos. Utilizar *Quasar* permite escribir código una vez y desplegarlo simultáneamente como un sitio web, una aplicación móvil y / o una aplicación de escritorio.

Node Package Manager (npm), es dos cosas: en primer lugar, es un repositorio en línea para la publicación de proyectos Node.js de código abierto; en segundo lugar, es un comando de línea de comandos para interactuar con dicho repositorio, gestiona por defecto la instalación de paquetes, administración de versiones y administración de dependencias. Una gran cantidad de bibliotecas y aplicaciones de Node.js se publican en npm, y se agregan muchas más todos los días. Estas aplicaciones se pueden buscar en <http://npmjs.org/>.

Cabe destacar que por elección personal se utilizó **Visual Studio Code** como editor de código, sin embargo esto no afecta a la implementación del proyecto, sino que va a la comodidad del desarrollador.

Message Queuing Telemetry Transport (MQTT) es un protocolo de publicación/suscripción que permite que los dispositivos edge-of-network publiquen en un broker. Los clientes se conectan a este intermediario, que luego media la comunicación entre los dos dispositivos. Cada dispositivo puede suscribirse o registrarse a tópicos particulares. Cuando otro cliente publica un mensaje sobre un tópico, el broker reenvía el mensaje a cualquier cliente que se haya suscrito.

MQTT es bidireccional y mantiene el conocimiento de la sesión con estado. Si un dispositivo edge-of-network pierde conectividad, todos los clientes suscritos serán notificados con la función "Última voluntad y testamento" del servidor MQTT para que cualquier cliente autorizado en el sistema pueda publicar un nuevo valor, manteniendo la conectividad bidireccional.

A continuación se puede ver un ejemplo del funcionamiento del protocolo MQTT. Dos dispositivos (un celular y una laptop) se suscriben al tópico "temperature", cuando el sensor publica en el broker bajo este tópico, el broker lo publica a todos los dispositivos suscritos.

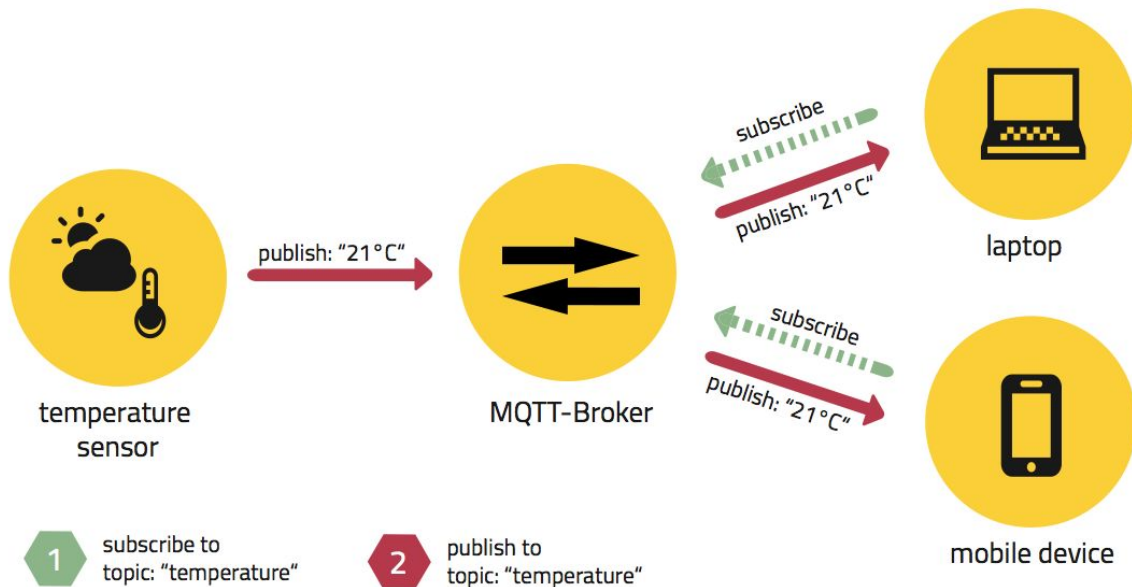


Imagen 1: ejemplo de funcionamiento del protocolo MQTT

Arquitectura

La conexión se realizó utilizando un broker público y gratuito de la empresa *Eclipse Foundation Inc.* hospedado en *test.mosquitto.org*.

Se eligió este broker ya que permite una conexión de MQTT over Websockets, que es requerida para poder desarrollar la versión web de la aplicación.

El broker cuenta con los siguientes puertos:

- 1883 : MQTT, unencrypted
- 8883 : MQTT, encrypted
- 8884 : MQTT, encrypted, client certificate required
- 8080 : MQTT over WebSockets, unencrypted
- 8081 : MQTT over WebSockets, encrypted

Como ya se mencionó anteriormente la placa NodeMCU utiliza el protocolo TCP/IP por lo que se conecta al puerto 1883 de este servidor. Y la aplicación se conecta al puerto 8080. Ambos sin encriptar para facilitar el desarrollo. De igual forma, la configuración de la aplicación incluye la opción de *usuario* y *contraseña* en caso de que sea requerido.

El proyecto cuenta con un diseño arquitectónico de los componentes de hardware el cual se muestra a continuación en los siguientes esquemas. Los diagramas de conexiones (Imagen 2, Imagen 4) y una foto real del proyecto (Imagen 3).

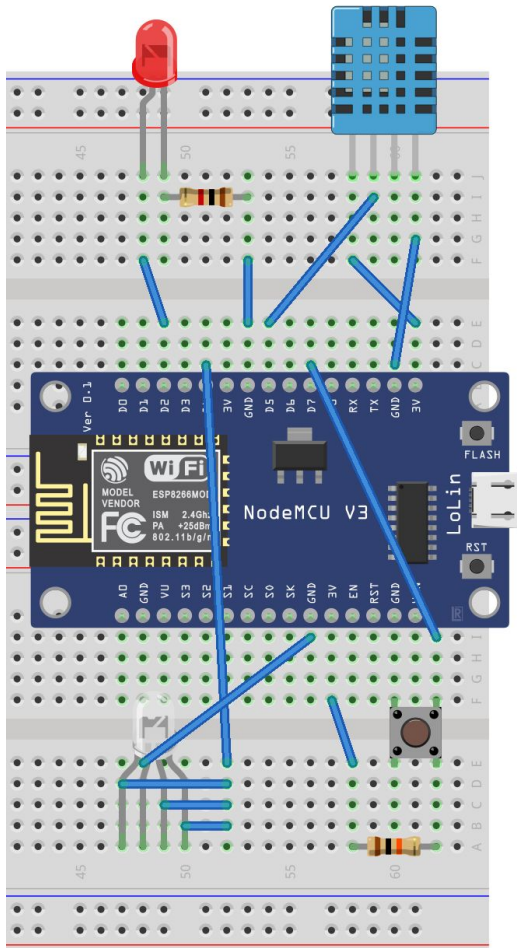


Imagen 2: diagrama sobre protoboard

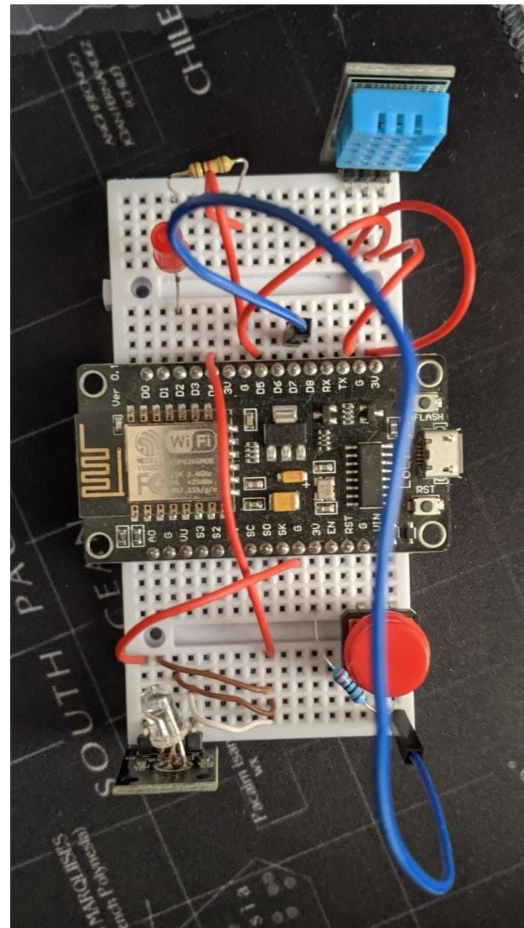


Imagen 3: foto real del proyecto

Los diagramas son representativos de la imagen real ya que no son exactamente los mismos componentes.

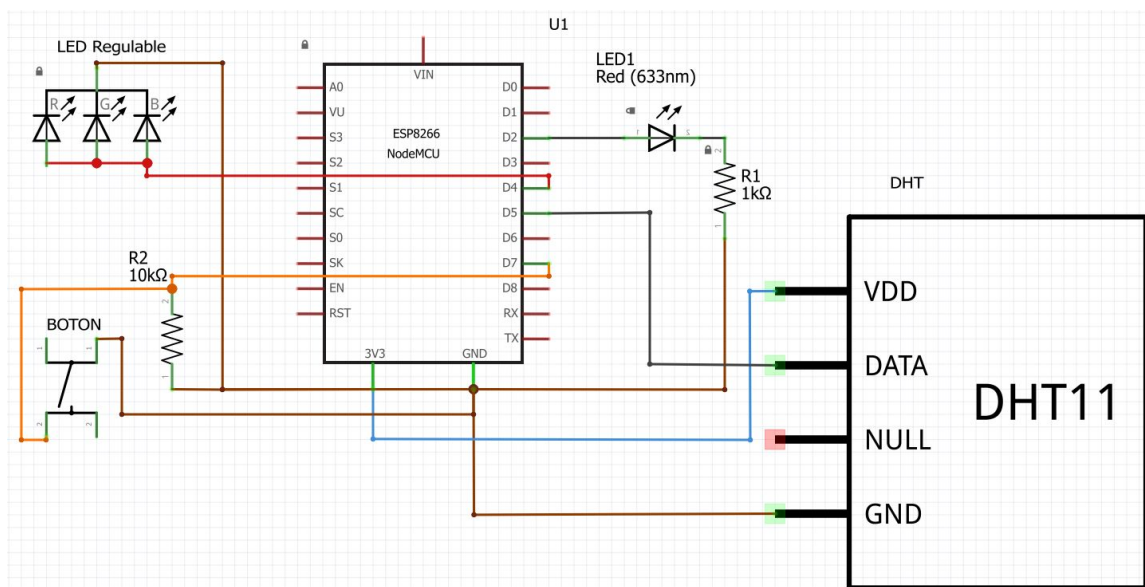


Imagen 4: diagrama de conexión de los componentes

Código Fuente

El código fuente del proyecto se encuentra alojado en la plataforma GitHub, documentado y con libre acceso. El mismo cuenta también con las instrucciones necesarias para su correcta ejecución. Se adjunta a continuación el link de acceso:

Código fuente del proyecto base realizado por Ezequiel Salagoity, el cual contiene una aplicación nativa de android y el código del NodeMCU 8266.

https://github.com/EzequielSalagoity/Trabajo_Final_POO-Smart-Home

Código fuente del proyecto al que hace referencia este informe.

<https://github.com/joaquin-amenta/smart-home>

Instalación

Para poder utilizar el código es necesario tener instalado node.js.

A través de consola luego se instala vue.js y *Quasar* con los siguientes comandos

```
npm install vue
npm install -g @quasar/cli
```

A continuación se debe clonar el repositorio. Abrir la consola en la carpeta del proyecto (se puede abrir la carpeta del proyecto en *visual studio code* y utilizar la consola integrada) y ejecutar el siguiente comando, el cual se encarga de instalar/actualizar todas las dependencias listadas en el archivo package.json:

```
npm install
```

Luego, deberá ejecutarse:

```
quasar dev
```

Con este último comando se abrirá una pestaña en el navegador por defecto, con la dirección localhost:8000

Esto sirve para desarrollar la app web y ver en tiempo real los cambios en ella.

Para hacer el build final se utiliza simplemente el comando:

```
quasar build
```


Android Development

En caso de querer desarrollar en *Android*, se detallan a continuación los pasos a seguir (cabe destacar que estas instrucciones son válidas sólo para el sistema operativo *Windows*).

Primero se debe instalar *cordova* (el framework de desarrollo mobile que utiliza *Quasar*) con el siguiente comando en la carpeta principal del proyecto

```
npm install -g cordova
```

Para continuar se debe tener instalado [Android Studio](#), junto con el [JDK de Oracle](#).

Luego se ejecutan los comandos:

```
quasar mode add cordova  
cd src-cordova  
cordova platform add android  
cordova requirements
```

Este último comando verifica que todo lo necesario esté instalado.

A continuación para desarrollar en *Android*, ejecutamos el comando que abrirá el *Android Studio* con el proyecto.

```
quasar dev -m cordova -T android --ide
```

Una vez abierto, primero se debe configurar la versión de *gradle* y no actualizarla nunca para este proyecto. Para realizar esto se debe hacer click, dentro de la barra de herramientas, en:

File → *Project Structure*

Dentro de la pestaña *project* se debe seleccionar la *Gradle Version* 4.10.3

En caso de que no aparezca en el menú desplegable se debe [descargar](#) y agregar a la carpeta "C:\Users\username\.gradle\wrapper\dists" (dirección por defecto).

Una vez seteada la versión de *gradle*, en la configuración del SDK se agrega la versión de *android-28*, que es requerida por *Quasar*. Luego puede utilizar el AVD (*Android Virtual Device*) para probar la app.

Para generar la APK, en la barra de herramientas de *Android Studio*, ir a:

Build → *Generate Signed Bundle / APK...*

Y seguir los pasos indicados por el IDE.

Este proyecto se basó en la [documentación de Quasar](#) donde se pueden encontrar todos estos pasos explicados y más información al respecto. Si es usuario de linux/mac, o si quiere desarrollar en iOS los pasos no difieren en gran medida, estos se encuentran también detallados en la documentación del link provisto.

Aunque no se abordó en este proyecto, *Quasar* también permite el desarrollo de [aplicaciones de escritorio](#) por medio de la utilización de *Electron*.

Workflow Demo

En esta sección, se realiza una demostración completa del *workflow* de la aplicación, mostrando su completo funcionamiento por medio de la captura de pantallas de la misma.

Cuando se inicia la aplicación, lo primero que se observa es la **Ventana de Inicio**[\(1\)](#). La aplicación se conecta por defecto al broker `ws://test.mosquitto.org:8080`

Y se suscribe a los tópicos:

“casa/luz/estado”

“casa/luz_reg/intensidad”

“casa/sensor/humedad”

“casa/sensor/temperatura”

Desde la Ventana de Inicio, se puede seleccionar la opción Luz y se navega a la **Ventana de Luz**[\(2\)](#), del mismo modo, se puede navegar a las ventanas de **Luz Regulable**[\(3\)](#) y **Temperatura y Humedad**[\(4\)](#).

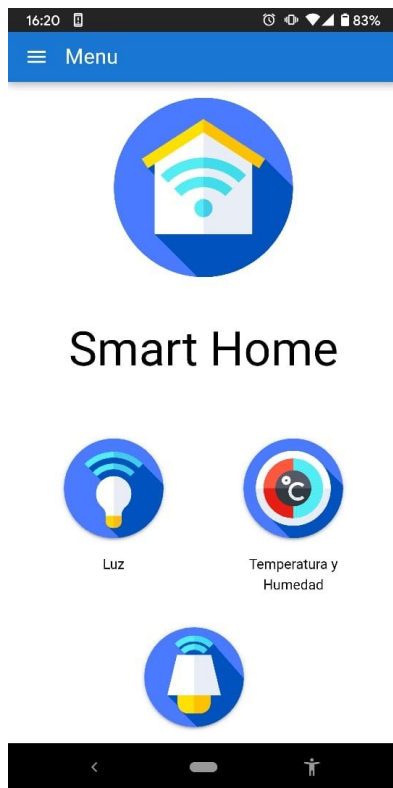
Al utilizar el botón on/off de la Ventana de Luz, o la barra desplazable de la Ventana de Luz Regulable, se envían mensajes al broker con el estado seleccionado, publicando bajo el tópico correspondiente.

Los mensajes a *luz* y *luz_reg* se envían con la opción *retain: true*, esto permite que cuando un usuario se conecta por primera vez, recibe el último estado en el que se encuentra cada componente.

Por otro lado, los datos de temperatura y humedad los envía el sensor cada dos segundos al broker, publicando bajo el tópico correspondiente y permitiendo así que los datos se actualicen constantemente.

Al intentar acceder a una de las páginas de la aplicación, aparecerá una **Pantalla de Carga**[\(5\)](#), esperando a recibir el último estado de los componentes o algún mensaje del broker. Si no se recibe ningún mensaje del broker, ya sea porque la configuración no es la correcta o el dispositivo no está conectado a internet, aparecerá un **Mensaje de Error**[\(6\)](#) sugiriendo que se revise la **Configuración** [\(7\)](#).

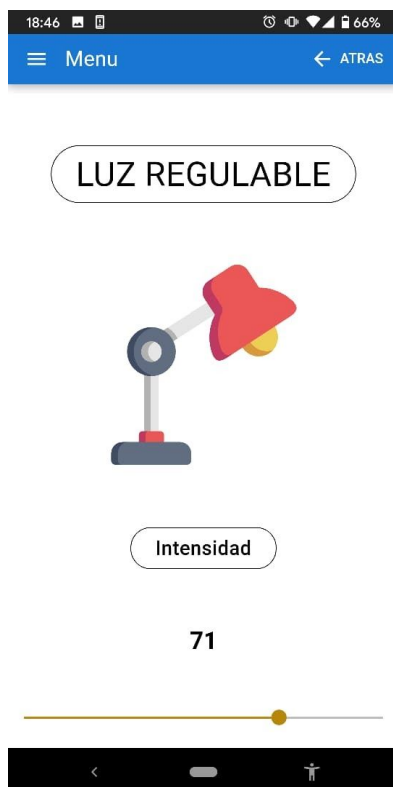
La aplicación también cuenta con un **Menú desplegable**[\(8\)](#) para acceder a todas las páginas, y con una **Ventana de Ayuda**[\(9\)](#) para intentar asistir en caso de algún error inesperado.



(1) *Ventana de inicio*



(2) *Ventana de Luz*



(3) *Ventana de Luz Regulable*



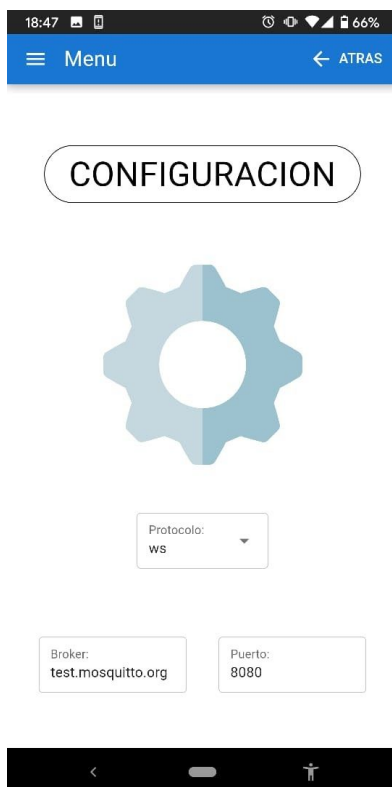
(4) *Ventana de Temperatura y Humedad*



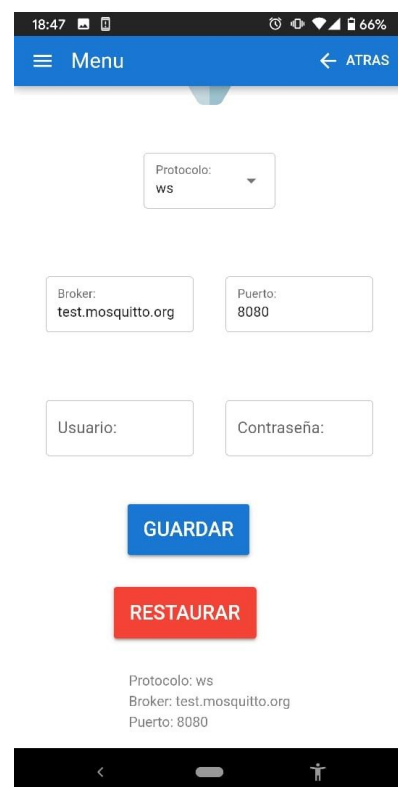
(5) Pantalla de carga

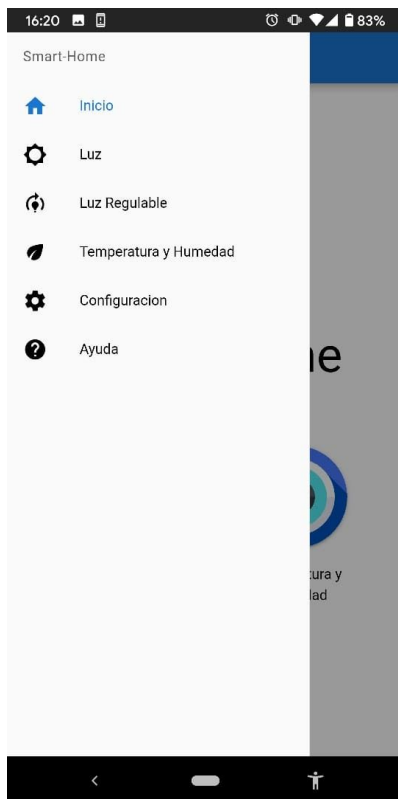


(6) Mensaje de error

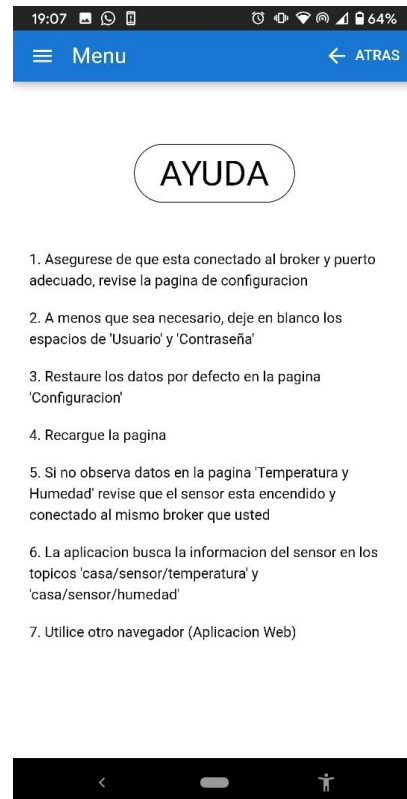


(7) Configuración





(8) Menú desplegable



(9) Ventana de ayuda

Demo de configuración NodeMCU

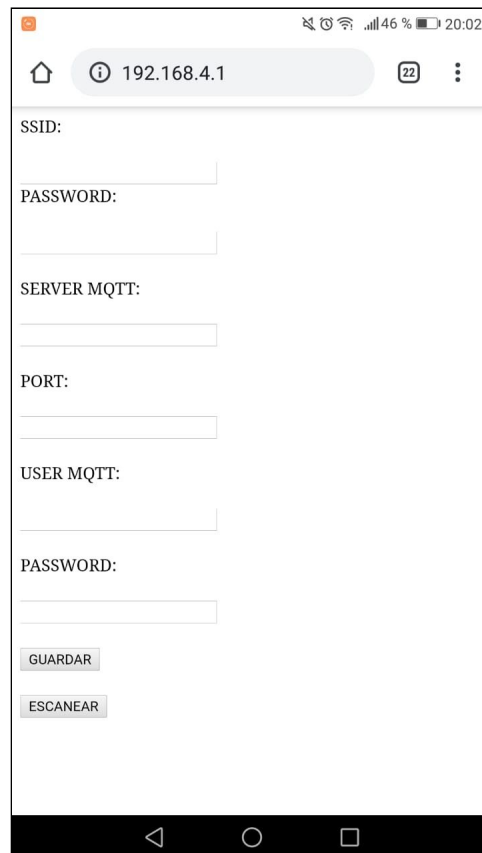
La placa NodeMCU se programó para poder configurar la red y el broker de la conexión. Para esto primero se conecta a una PC con un cable USB, luego se mantiene presionado el botón a presión y se aprieta el botón de reset integrado en la placa. De esta forma entra en modo *access point*. Los datos se pueden leer en el puerto COM al que esté conectado:

```
-> -----> MODO ACCESS POINT <-----
->
-> ----> DATOS DEL ACCESS POINT <---
->     -> SSID: NodeMCU_AP
->     -> Contraseña: 12345678
->
-> IP del access point: 192.168.4.1
-> WebServer iniciado...
```

Luego de conectarse a la red de la placa con algún dispositivo, se debe abrir una ventana de un Browser e ingresar a la dirección IP del access point, allí se mostrará la pantalla de **Configuración**⁽¹⁰⁾.

Al guardar la nueva configuración y volver a apretar el botón de reset, la placa intentará conectarse a los nuevos datos ingresados.

Es necesario que la app y el NodeMCU se conecten al mismo broker.

A screenshot of a mobile browser displaying the NodeMCU configuration page. The address bar shows the IP address 192.168.4.1. The page contains several input fields for configuration: SSID, PASSWORD, SERVER MQTT, PORT, USER MQTT, and another PASSWORD field. At the bottom, there are two buttons labeled 'GUARDAR' (Save) and 'ESCANEAR' (Scan). The interface is simple and functional, with a white background and black text.

(10) *Pantalla de configuración NodeMCU*

Referencias

Se adjuntan varios links de consulta y referencia respecto a las tecnologías utilizadas en el desarrollo del proyecto.

<https://vuejs.org/v2/guide/>

https://www.nodemcu.com/index_en.html

<https://quasar.dev/>

<https://nodejs.org/es/>

<https://www.npmjs.com/>

<https://www.npmjs.com/package/mqtt>