UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

# DISEÑO DE INTERFACES MODULARES PARA FACILITAR PROYECTOS DE IOT

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MIGUEL EDUARDO ESPINOZA PEREIRA

PROFESOR GUÍA:
CLAUDIO ESTÉVEZ MONTERO

MIEMBROS DE LA COMISIÓN:
JOSÉ GONZÁLEZ GARCÍA
FRANCISCO RIVERA SERRANO

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL
ELÉCTRICO
POR: MIGUEL EDUARDO ESPINOZA PEREIRA
FECHA: 2022
PROF. GUÍA: CLAUDIO ESTÉVEZ MONTERO

# DISEÑO DE INTERFACES MODULARES PARA FACILITAR PROYECTOS DE IOT

El *Internet of Things* (IoT) ha experimentado un aumento explosivo en los últimos años, cada vez hay más equipos electrónicos con capacidad de conectarse a la red: relojes inteligentes, sensores inalámbricos, televisores inteligentes, etc. Además, existe un gran interés de las personas por la creación de proyectos amateur de electrónica y robótica como la implementación de luces inteligentes, regadores de pasto automáticos, entre otros. Sin embargo, la creación de un dispositivo de IoT desde cero, requiere la integración de diferentes líneas del conocimiento como electrónica, programación, redes y creación de interfaces gráficas.

En este trabajo se pretende crear un equipo electrónico que busca facilitar el acercamiento de nuevas personas al mundo de la electrónica y al IoT; mediante la creación de interfaces electrónicas para que algunos sensores se puedan conectar de forma *plug and play* a un módulo central y sus mediciones aparezcan en una página web. Este sistema permite que personas ajenas al mundo de la electrónica puedan interactuar con un equipo de IoT de forma sencilla. Además, el código y el hardware del proyecto fue pensado para que sea posible añadir más sensores que los creados por el autor, por lo que un desarrollador puede agregar nuevos sensores al equipo sin la dificultad de tener que crear un proyecto de IoT desde cero.

El módulo central consiste en una Raspberry Pi conectada a una PCB que tiene dos conectores para sensores digitales y dos conectores para sensores analógicos. Por su parte, las interfaces de los sensores cuentan con el sensor, un conector que se inserta en el módulo central y el circuito correspondiente para el funcionamiento del sensor. Las interfaces están diseñadas de tal manera que cada pin de un conector ejecuta una función, ya sea la entrega de voltaje, la conexión a tierra, la identificación de la interfaz, el envío de datos, etc. Gracias a esto, los conectores de las interfaces en la PCB son independientes de los conectores del módulo central, es decir, una interfaz no debe conectarse a un conector en específico, sino que funciona con cualquiera. El módulo central es capaz de detectar la conexión de una interfaz de forma instantánea, identificar el sensor conectado, solicitar la medición del sensor y enviarla a una base de datos. Todo esto en tiempo real. Además, en la Raspberry Pi se creó un servidor web usando Flask y Nginx, por lo que desde Internet es posible ingresar a la página web y visualizar las mediciones del sensor.

El resultado de este proyecto es un equipo electrónico al que se le pueden conectar sensores digitales y analógicos de forma *plug and play*. Las mediciones del sensor son visibles desde Internet, por medio de gráficos y tablas que muestran la información de forma ordenada. Además, existe un panel en la página web qué muestra el estado del módulo central, es decir, permite visualizar que sensores tiene conectados actualmente.

*Para mis seres queridos.*
*Para mis familiares que ya no están en este mundo:*
*mis abuelos que no pude conocer, pero sé que me quisieron mucho,*
*y mis tíos que se han ido y que recuerdo con cariño.*
*Y también para ti que tienes interés en leer mi memoria.*

# Agradecimientos

El llegar al término de la carrera siempre trae recuerdos del pasado, de los momentos agradables, de las experiencias difíciles que sirvieron para crear carácter, y en especial de las personas que acompañaron a lo largo de todo el proceso, el cual comenzó mucho antes de entrar a la universidad, pues es algo que envuelve todos los años de estudio. En esta página quiero agradecer a todos los que me acompañaron y ayudaron a lo largo de estos años.

En primer lugar, agradezco a Dios el permitirme el estudiar la carrera que tanto quería, por darme las capacidades intelectuales y la perseverancia para no dejar de lado los estudios. Agradezco a mis papás Olivia y José por el cariño y por todo el apoyo que me brindaron durante toda mi época de estudio en el colegio, sobre todo por la fe que tuvieron en que todo saldría bien, y en que algún día llegaría a la universidad. Asimismo, agradezco a mis familiares que me apoyaron en diversos momentos, en especial a Mili, Mario y Vivi.

En segundo lugar, me gustaría agradecer a mis grandes amigos: Claudio, Benja, Mapi, Cami y Coty. Gracias por estar siempre ahí acompañándome, por escucharme hablar de mis problemas, por las juntas, las llamadas, los juegos, etc., han sido un apoyo muy grande. Además, agradezco a mis amigos del colegio y de la U: Erick, R, Diego, Jaime, Gato, Nicolini, Peña, Lukas y Guido, gracias por darme tantos momentos inolvidables y chistosos que recuerdo con alegría y nostalgia, y por esos largos tiempos de estudio en grupo. También, quisiera mencionar a los chiquillos de GBU, con los cuales pase grandes momentos en los últimos años, en especial el período de pandemia, con las reuniones online, los juegos, las oraciones en la noche, etc. Junto con los chiquillos de la UJ crearon un ambiente seguro y divertido, en el que se podía escapar un poco de la realidad que se estaba viviendo en ese entonces, dentro de estos grupos destaco a Nicole, a Diego y a los Boanerges.

Asimismo, agradezco al profesor Claudio Estévez por aceptar mi solicitud de hacer una memoria con él como profesor guía, ya que me propuso un tema que me interesó mucho y que disfrute ejecutándolo. Gracias por acompañarme durante el desarrollo de la memoria y por darse el tiempo de enseñarme en las reuniones. Asimismo, le doy las gracias al profesor José González, por sus consejos sobre el diseño y construcción de la PCB, y por ayudarme con el envío de la placa. También, agradezco al profesor Francisco por sus correcciones en mis presentaciones y en el borrador; y a Karina del Laboratorio de Electrónica, ya que me asistió mucho en la construcción de la PCB con el soldado de componentes y me prestó varios equipos electrónicos.

Por último, agradezco a los profesionales del SEMDA Ingeniería, ya que me asistieron en un momento muy difícil de mi carrera y de mi vida.

# Tabla de Contenido

# Índice de Tablas

# Índice de Ilustraciones

# Capítulo 1

# Introducción

## 1.1. Motivación y Antecedentes

En el mundo actual cada vez hay más personas conectadas a Internet [1], ya sea para subir fotos, comunicarse con amigos, ver una transmisión streaming, jugar videojuegos online, etc. Para tal propósito, poseen teléfonos inteligentes, computadores y tablets, equipos que tienen como una de sus funciones principales el conectar a las personas a la red. Pero. últimamente, también las "cosas" están adquiriendo la capacidad de conectarse a Internet: existen Smart TV, Smartwatch, lavadoras inteligentes con conectividad WiFi, e incluso ampolletas inteligentes que pueden encenderse a través de comandos de voz. A la tecnología que da vida a todas esas nuevas capacidades, y a muchas más, se le llama Internet de las Cosas.

El Internet de las Cosas, (IoT por sus siglas en inglés), término acuñado el año 1999 por Kevin Ashton en una presentación ante Procter & Gamble [2], se define como un paradigma en el cual los objetos equipados con sensores, actuadores y procesadores se comunican unos con otros para lograr un propósito [3]. El IoT presenta una gran capacidad de revolucionar la vida cotidiana, pues permite que objetos comunes en un hogar, se comuniquen entre sí y extraigan información de su entorno, con el fin de facilitar la vida a sus dueños. Asimismo, posee la capacidad de producir beneficios a las industrias, automatizando sus procesos productivos. De hecho, el IoT es una de las tecnologías base de la llamada Industria 4.0 [4].

El uso del término *Internet of Things*, ha tenido un gran crecimiento en los últimos años. En [5] se muestra como la cantidad de publicaciones sobre IoT en el Web of Science Core Collection ha crecido de manera exponencial desde el año 2008. Asimismo, la cantidad de dispositivos de IoT ha experimentado un explosivo aumento, llegando a superar la cantidad de personas en el mundo. Se predecía que para el año 2020 iban a existir cerca de 20 miles de millones de dispositivos, de acuerdo a la estimación de Gartner, y 26 miles de millones de dispositivos, de acuerdo a Cisco [6].

Por lo tanto, desde la academia y la industria hay un interés por esta tecnología, pero también el público general ha empezado a interesarse sobre este término. Como se puede observar en [7], la tendencia de búsqueda en Google del término *iot* ha aumentado en los últimos años. De igual manera, plataformas de aprendizaje en línea, como Coursera y Udemy, también han agregado programas de IoT a su lista de cursos online, los cuales han tenido bastante éxito, pues cuentan con más de 250.000 alumnos inscritos [8].

## 1.2. Descripción del problema

A pesar de este gran interés del público, se produce un efecto que ocurre generalmente con las nuevas tecnologías: las barreras de entrada son muy altas. Se requieren múltiples conocimientos, principalmente de electrónica, telecomunicaciones, protocolos de Internet, microcontroladores, etc., para lograr hacer un proyecto de Internet de las Cosas. Las personas aficionadas a las nuevas tecnologías y los estudiantes que recién empiezan con una carrera de Ingeniería Eléctrica o similar, no cuentan con los conocimientos necesarios y requerirían mucho tiempo de estudio para lograr hacer un proyecto funcional.

Para empezar, la conexión de los sensores puede ser complicada, debido a que estos pueden requerir diferentes niveles de voltaje, distintos tipos de señal, o configuraciones muy específicas. La programación de un microcontrolador requiere conocimientos de programación a bajo nivel. Para generar la comunicación inalámbrica, se necesita conocer los distintos protocolos de comunicación y cuales son sus fortalezas y debilidades en varios entornos. Por último, el diseño de un sitio web tampoco es una tarea sencilla.

Para solucionar estos problemas lo que se debe hacer es remover las dificultades del desarrollo de un sistema de IoT, ocultando todo el proceso interno, omitiendo las capas intermedias del proyecto. Las personas deberían ser capaces de conectar un sensor y visualizar automáticamente los datos que está enviando en una página web, del mismo modo como se conectan a Internet a través de sus dispositivos móviles, en donde sólo deben hacer clic y automáticamente están en la red. Así se lograría que puedan ver de forma práctica las aplicaciones que tiene esta nueva tecnología y, así, motivarse a crear sus propios proyectos, haciendo menos empinada la curva de aprendizaje.

## 1.3. Objetivos

### 1.3.1. Objetivos generales

Diseñar e implementar interfaces electrónicas que permitan la conexión *plug and play* de sensores a un módulo central. Se busca que cuando el usuario conecte un sensor al módulo central, sus datos estén disponibles por medio de Internet para que puedan ser visualizados en tiempo real.

### 1.3.2. Objetivos específicos

Los objetivos específicos de este trabajo, que delimitan y señalan los alcances de este son:

- Diseñar y construir interfaces electrónicas para la conexión de sensores, en donde estos se puedan conectar de forma sencilla, con conectores específicamente diseñados para este propósito. También es parte de los objetivos del trabajo, el estudio y la selección de los sensores a utilizar.
- Diseñar e implementar una interfaz gráfica accesible desde Internet, en donde se puedan observar las mediciones de los sensores. Esta interfaz debe ser modificable por el usuario, para que muestre los datos que este requiera.

- Diseñar y construir placas PCB para el módulo central y cada uno de los sensores a utilizar.
- El sistema a construir debe tener un bajo costo económico y debe ser sencillo de utilizar por usuarios sin conocimientos de electrónica.

En la figura 1.1 se puede ver un diagrama en bloques que representa el diseño del trabajo a realizar en la memoria. Como se puede ver, el módulo central tiene cuatro componentes principales: la Raspberry Pi, los conectores de sensores digitales, los conectores de sensores analógicos y un conversor análogico digital (ADC). Se implementaron dos interfaces digitales: la interfaz de temperatura y la interfaz de distancia; estas interfaces se pueden conectar a cualquiera de los dos conectores digitales del módulo central. Además, se implementó una interfaz analógica: la interfaz de de distancia; la cual, de igual manera, se puede conectar a cualquiera de los dos conectores analógicos. Cabe notar que los conectores analógicos no se conectan directamente a la Raspberry Pi, si no que pasan antes por un ADC que transforma sus señales analógicas en digitales.

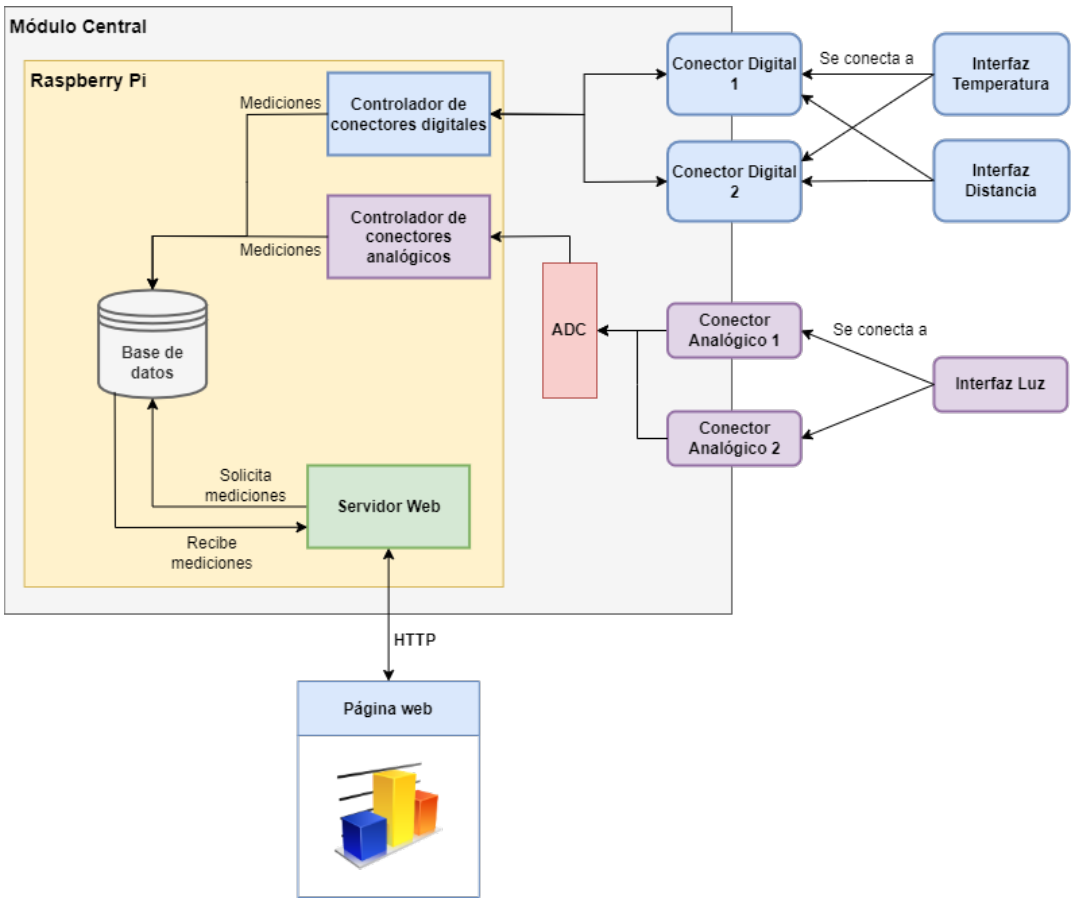


Figura 1.1: Diagrama del proyecto de la memoria.

La Raspberry Pi contiene dos scripts controladores de los conectores analógicos y digitales, que se encargan de detectar cuando se conecta una interfaz a un conector y cuál sensor es el que está conectado. La Raspberry Pi guarda las mediciones de los sensores conectados en una base de datos, la cual es accedida por el servidor web cuando el usuario que está en la página web lo requiere. La página web muestra un gráfico y una tabla con los últimos valores

del sensor seleccionado, y se actualiza automáticamente cuando llegan nuevos datos.

## 1.4. Estructura de la memoria

El capítulo 2 trata acerca del marco teórico y el estado del arte, por lo que se enfoca en explicar las tecnologías que se utilizan en la memoria y en mencionar y analizar proyectos similares que han sido desarrollados. El capítulo 3 presenta los sensores y componentes utilizados, junto al diseño del PCB y el soldado de componentes en la placa; es decir, es un capítulo enfocado exclusivamente al hardware utilizado. El capítulo 4 trata sobre el código realizado para identificar los sensores conectados al módulo central y leer las mediciones de estos para, luego, guardarlas en la base de dato. Por lo tanto, profundiza en el software que hace funcionar al proyecto. El capitulo 5 expone la implementación de la página web, mostrando el diseño realizado en el front end y la lógica que se ejecuta en el back end. Los resultados y análisis del proyecto se muestran en el capítulo 6. Por último, el capítulo 7 consiste en las conclusiones del trabajo realizado y futuras adiciones al proyecto.

# Capítulo 2

# Marco Teórico y Estado del Arte

## 2.1. Internet of Things

Una definición más técnica de IoT es que es: *una red de cosas, con inteligencia, identificación, y capacidades de monitoreo y actuación, que conectan personas y cosas a través de Internet* [6]. Como se indica en [3], es posible definir una arquitectura de IoT dividiendolo en cinco capas, ordenadas jerárquicamente de forma similar a la capas del modelo OSI y TCP-IP. Cada una de las capas se encarga de forma independiente de cierta tarea, utilizando la información que le proporciona la capa anterior. Ordenadas desde la inferior a la superior, estas capas son:

- **Capa de percepción**: esta encargada de la percepción de cantidades físicas a través de sensores.
- **Capa de transporte**: se ocupa de la transmisión de datos desde la capa de percepción a la capa de procesamiento. Generalmente, utiliza redes inalámbricas para cumplir ese propósito.
- **Capa de procesamiento**: se encarga del procesamiento y almacenamiento de la información obtenida de las capas anteriores.
- **Capa de aplicación**: es la que se ocupa de proveer los servicios para el usuario, ya sea entregando la información requerida o actuando de la manera indicada.
- **Capa de negocios**: su función es crear modelos, gráficos y todo tipo de elementos de análisis, a partir de la información de la capa de aplicación. [9]

En el proyecto de memoria se pretende ocultar las capas de transporte y de procesamiento al usuario, de tal manera que sólo puede interactuar con las capas de percepción y de aplicación. La capa de negocios se omite debido a que no se busca estudiar los datos entregados por la capa de aplicación.

## 2.2. Raspberry Pi

La Raspberry Pi es un microcomputador de bajo costo y de tamaño pequeño. Esta diseñado para aficionados que quieren crear proyectos de electrónica, para estudiantes que desean

aprender de programación y para Ingenieros que lo deseen utilizar en sus proyectos. Una Raspberry Pi es un computador de placa única (*single-board computer*), esto quiere decir que tiene las mismas funcionalidades que cualquier otro computador del sistema operativo Linux, pero en un sólo chip [10].

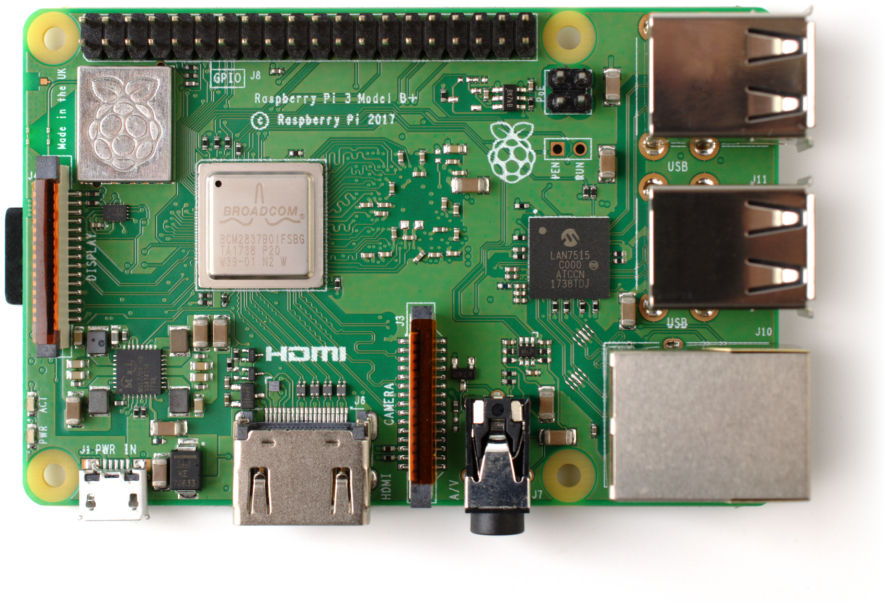


Figura 2.1: Raspberry Pi Modelo 3 B+. [11]

En la figura 2.1, se observa una Raspberry Pi Model 3B+, la cual como se puede ver presenta componentes típicos de un computador y de un microcontrolador:

- Procesador
- Memoria RAM
- 4 Puertos USB
- Puerto HDMI
- Puerto Ethernet
- Tarjeta Inalámbrica WiFi/Bluetooth
- Puerto Jack 3.5mm
- Puerto Micro SD
- 40 Pines GPIO

Gracias a que cuenta con todos estos componentes, es posible hacer un uso corriente de la Raspberry conectando un *mouse* y un teclado a los puertos USB, y un monitor al puerto HDMI, para trabajar la Raspberry Pi como un computador de escritorio y, en consecuencia,

hacer las tareas típicas de ofimática y de programación. Además, como cuenta con comunicación WiFi, es posible acceder a ella a distancia mediante protocolos como SSH. Funciona con 5V de alimentación, los que son proporcionados a través de una conexión micro-USB.

La Raspberry Pi cuenta con un sistema operativo llamado *Raspberry Pi OS*, el cual está basado en Linux. Este sistema operativo se encuentra en una tarjeta Micro SD que se inserta en el puerto correspondiente y que actúa también como un disco duro, en el que se pueden guardar los archivos del microcomputador. Cabe recalcar que es posible usar otros tipos de sistemas operativos como Ubuntu o Debian.

Para este proyecto uno de los componentes de mayor relevancia son los 40 pines, debido a que permiten la conexión de la Raspberry con sensores, actuadores, ADCs, y otros dispositivos electrónicos. Como se puede ver en la figura 2.2, la mayoría de los pines son GPIO (General Purpose Input and Output), lo que indica que pueden ser utilizados según la necesidad del usuario. Sin embargo, también existen pines listados como alimentación (de 3.3V o 5V), y de tierra (GND). Asimismo, algunos pines GPIO presentan funciones alternativas, que permiten la conexión de dispositivos mediante los protocolos de comunicación UART, I2C o SPI, entre otros. Es importante señalar que los pines GPIO sólo soportan un máximo de 3.3V, por lo que en caso de que un sensor entregue señal de 5V, será necesario usar un divisor de voltaje o un conversor de niveles lógicos, para reducir el voltaje de 5V a 3.3V y así evitar dañar la Raspberry [12]. Por último, todos los pines de la Raspberry son digitales, por lo que para leer las mediciones de un sensor analógico se debe usar un ADC.

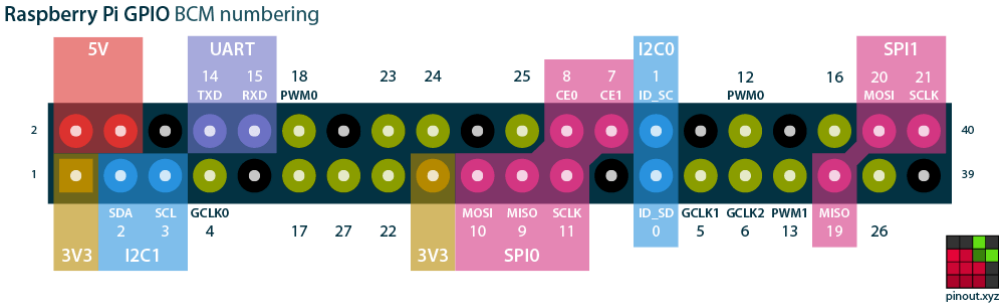


Figura 2.2: Pines GPIO de la Raspberry Pi y su funcionalidad [13].

## 2.3. Resistencias pull-up y pull-down

El estado de un pin de un circuito electrónico puede ser ALTO o BAJO, dependiendo de si está conectado a un voltaje cercano al voltaje de entrada (5V o 3.3V generalmente); o si es cercano a 0V, respectivamente. En todos los proyectos de electrónica es necesario saber de antemano el voltaje que tendrán algunos pines en estado de reposo, dado que de otra manera quedan en estado flotante. Por esta razón, existen las resistencia de pull up y de pull down.

Una resistencia de pull up conectará el pin al voltaje de alimentación positiva, tal como aparece en la figura 2.3, por lo que en estado de reposo al medir el voltaje del pin, se medirá ALTO. En cambio, cuando se cierre el switch, se medirá el valor BAJO en la salida, puesto

que VOUT quedará conectado a tierra. Por otro lado, una resistencia de pull down conecta el pin a tierra, , tal como se ve en la figura 2.4. Por lo que al leer el valor del pin en estado de reposo se leerá BAJO y, cuando el switch esté cerrado, se leerá el valor ALTO.

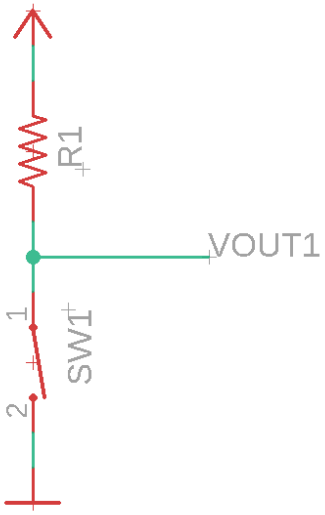


Figura 2.3: Resistencia pull up.

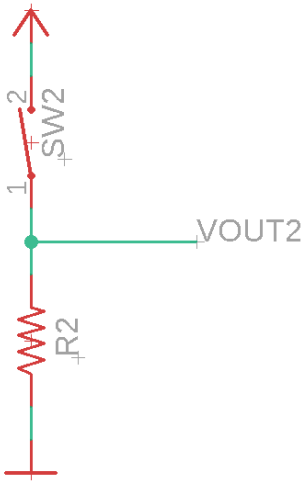


Figura 2.4: Resistencia pull down.

## 2.4. Placa de Circuito Impreso (PCB)

Los PCB son circuitos que en vez de utilizar cables para conectar los distintos dispositivos electrónicos, utilizan pistas de cobre sobre una capa de sustrato rígido, lo que permite que sean permanentes, fáciles de transportar, y que sea posible producirlos en masa [14]. En la figura 2.5, se puede ver que el sustrato es el material de color verde que aporta la base de todo el circuito y las pistas de cobre son las líneas de color oscuro que conectan a los componentes.

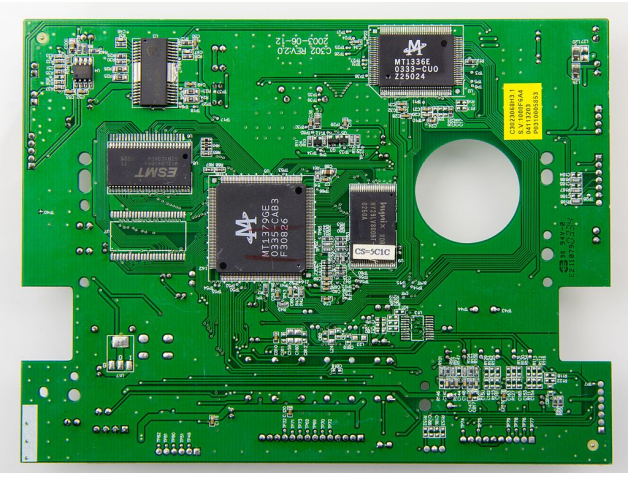

Figura 2.5: Ejemplo de PCB de un reproductor de DVD [15]

Los dispositivos son montados sobre las PCB de dos formas: mediante la tecnología de

agujeros pasantes (THT: *Through-hole technology*) y a través de la tecnología de montaje superficial (SMT: *Surface-mount device*). El método THT consiste en realizar agujeros en donde se insertan los cables de los dispositivos [16]. Por otro lado, en el método SMT, los dispositivos se conectan a pads de metal que se encuentran en la superficie de la placa. En ambas tecnologías, los dispositivos son soldados a la placa para que tengan estabilidad y permanezcan firmes ante movimientos o traslados de la placa. En la figuras 2.6 se puede ver un ejemplos de una PCB con un montado de dispositivos de forma THT.

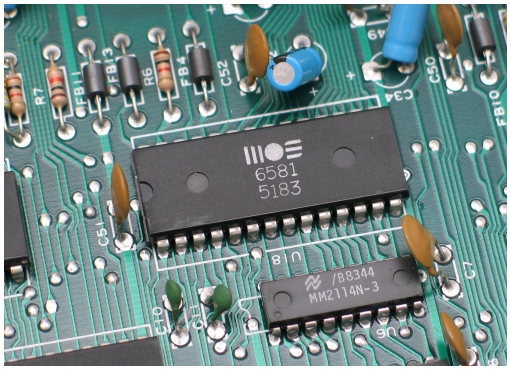


Figura 2.6: THT [17].

Para el diseño de PCB existen programas llamados EDA (*Electronic Design Automation*), entre los que se pueden nombrar Eagle, KiCad y EasyEDA. Dentro de estos programas es común que existan dos espacios de trabajo, el del esquemático y el de la placa. En el área del esquemático se agregan los componentes en forma de esquemático y se realizan las conexiones eléctricas entre ellos, de manera que el programa sepa cuales están conectados entre sí. Mientras que en el área de trabajo de la placa, ya se trabaja con las dimensiones físicas/mecánicas de los componentes y se debe realizar manualmente la interconexión de los dispositivos a través de pistas (o rutas).

Una vez que está creada la placa PCB se puede mandar a manufacturar a una empresa de producción de PCB, como JLCPCB para prototipos. Una vez que la placa está construida, la empresa la envía al cliente.

## 2.5. Desarrollo Web

### 2.5.1. HTTP

Una tecnología esencial para lograr acceder a Internet y a páginas web, es el protocolo HTTP (HyperText Transfer Protocol). HTTP es un protocolo que se encarga de la transmisión de datos (archivos de texto, imágenes, videos, etc.) entre un navegador web y un servidor web. El protocolo HTTP funciona por medio de una lógica cliente-servidor, en el que el navegador hace *requests* (solicitudes) a un servidor web, y el servidor responde a través de *responses* (respuestas) [18]. Se puede encontrar más información del protocolo HTTP en [18]. De igual modo, en [19] se presentan sus métodos y un ejemplo de solicitud HTTP.

### 2.5.2. Front End

El desarrollo de una página web consiste en la implementación del front end y el back end. El front se refiere a la interfaz gráfica que ve el usuario cuando entra al sitio web. Para desarrollar el front end de una página web se utilizan tres tecnologías: HTML, CSS y JavaScript.

#### 2.5.2.1. HTML

HTML (HyperText Markup Language) es un lenguaje de marcado que se encarga del texto que lleva una página web y su estructura. Dado que es lenguaje de marcado, utiliza etiquetas para señalar qué tipo de contenido se encuentra en la página, por lo tanto, HTML permite indicarle al navegador web como se debe mostrar cada sección de texto de una página web [20].

Algunos ejemplos de etiquetas HTML son `<head>`, `<body>`, `<h1>`, `<p>` y `<div>`. Por ejemplo, la etiqueta `<p>` indica que el texto que tiene dentro es un párrafo y, por otro lado, la etiqueta `<h1>` indica que el texto es un título. HTML permite identificar cada uno de los elementos que contiene la página web a través de *clases*, *id* y las propias etiquetas.

#### 2.5.2.2. CSS

CSS (Cascading Style Sheets) es un lenguaje de estilo de hojas, que se encarga de darle el diseño a una página web. CSS dicta el estilo de cada uno de los elementos que tiene una página web. Permite seleccionar el color del texto, el tipo de fuente, el color del fondo de la página, e incluso la organización del texto, mediante la creación de barras de navegación o columnas [20].

Para facilitar el diseño de una página web para no iniciados en CSS, existen librerías con diseños predeterminados, en las que el desarrollador puede implementar una página web con un diseño más profesional, simplemente cargando el estilo correspondiente. Una de esas librerías es Bootstrap [21], la cual se usó para ayudar en el diseño de la página web de este proyecto.

#### 2.5.2.3. JavaScript

JavaScript, a diferencia de las tecnologías previamente mencionadas, sí es un lenguaje de programación. Por lo tanto, cuenta con toda la funcionalidad, como definir variables, funciones, objetos, etc. Gracias a esto, permite interacciones dinámicas entre el usuario y la página web, permite tener mapas interactivos, gráficos seleccionables, jugar videojuegos en el navegador, entre otros [22].

Dentro de un documento HTML, pueden existir varios scripts de JavaScript, que se cargan a través de la etiqueta `<script>`. Al igual que en el resto de los lenguajes de programación, existen librerías que facilitan el desarrollo de diferentes tareas en JavaScript. Para el desarrollo de este proyecto se usaron dos: JQuery y Google Charts.

JQuery es un librería que simplifica el código de JavaScript. Tareas como el acceso a elementos de HTML, el manejo de eventos y el uso de AJAX, se pueden realizar con un código mucho más reducido gracias a JQuery [23]. Por otro lado, Google Charts, es una librería

desarrollada por Google que permite la creación y modificación de gráficos en una página web, de forma sencilla y en tiempo real [24]. Con Google Charts se pueden mostrar gráficos de barras, de líneas o de torta, entre muchos otros.

También es importante mencionar a AJAX. AJAX es un método que permite realizar solicitudes asíncronas a un servidor web, de tal manera que el navegador estará esperando a que llegue la respuesta de parte del servidor sin quedarse detenido. Esto permite que el navegador realice otras tareas mientras tanto. Además, da la posibilidad de que se carguen secciones de una página web de forma independiente, sin necesidad de cargar toda la página por completo, mejorando así la experiencia de usuario. En esta memoria se utiliza AJAX para cargar el gráfico de la página web de forma independiente y para actualizar el panel de información.

### 2.5.3. Back End

El back end de un sitio web se refiere a todas las tecnologías que se encuentran por el lado del servidor, y que el usuario no puede interactuar con ellas. Por ejemplo, la base de datos, el servidor HTTP, o el lenguaje de programación del servidor.

Para el desarrollo de un back end de un sitio web, se pueden usar *stacks* web. Los stacks son *un grupo de tecnologías que trabajan de forma conjunta para lograr un objetivo*. Para el desarrollo de servidores de páginas web, un stack bastante popular es LAMP [25], cuyas siglas consisten en las siguientes tecnologías:

- L: se refiere al sistema operativo Linux.
- A: viene del servidor HTTP Apache.
- M: se refiere al gestor de base de datos MySQL, o a su fork MariaDB.
- P: en la forma original del stack, viene de PHP, aunque versiones posteriores, también utilizan Python o Perl.

En esta memoria, se emplea el stack LEMP [25], el cual es una modificación del stack LAMP, que usa el servidor Nginx en vez de Apache. Por lo tanto en esta memoria, se utiliza el siguiente stack web:

- L: Raspberry Pi OS, una distribución de Linux que viene con la Raspberry Pi.
- E: Servidor HTTP Nginx.
- M: MariaDB.
- P: Python.

#### 2.5.3.1. Linux

Linux es un sistema operativo de código abierto, es decir, su código se encuentra disponible de manera libre y gratis. Consiste, principalmente, en el kernel Linux, el cual es el núcleo del sistema operativo, que lleva a cabo la interacción entre el hardware y el software. Linux se utiliza en sistemas operativos a través de *distribuciones*, las cuales son modificaciones que

se hacen al sistema operativo para adaptarlo a las necesidades del usuario [26]. Algunas de las distribuciones más famosas son: Ubuntu, Debian y Linux Mint.

Raspberry Pi OS es una distribución diseñada específicamente para ser usada en los modelos de Raspberry Pi. Por lo tanto, viene de fábrica con software que se encarga de la configuración del I/O de los pines GPIO, con protocolos de comunicación de electrónica y con otros programas acordes a las funcionalidades que posee el hardware de una Raspberry Pi.

#### 2.5.3.2. Nginx

Nginx es un servidor web de software libre. Nginx se destaca por ser uno de los servidores web más rápidos que existen, de hecho fue diseñado específicamente para ser más rápido que Apache. Asimismo, permite la conexión de muchos usuarios al mismo tiempo, pues nació como solución al problema de tener más de 10000 conexiones a la vez [27]. Por último, utiliza menos memoria RAM que otras alternativas, lo cual es particularmente útil para la Raspberry Pi [28], puesto que el modelo 3 B+, que se utiliza en este proyecto, sólo cuenta con 1 GB de RAM.

#### 2.5.3.3. MariaDB

MariaDB un gestor de base de datos basado en MySQL, pues es una bifurcación de este, que nace como respuesta a la compra de los derechos de MySQL por parte de Oracle, pues se quería mantener el proyecto con una licencia de software libre [29]. MariaDB mantiene el mismo lenguaje SQL que MySQL, ya que se trabaja para que ambos tengan los mismos comandos y sean compatibles entre sí.

MariaDB utiliza el lenguaje SQL para realizar consultas en bases de datos [29]. SQL se emplea para acceder a los datos guardados en bases de datos relacionales. Estas bases de datos funcionan a través de la creación de tablas que guardan distintos campos y que se pueden acceder por medio de consultas. También, es posible filtrar resultados mediante comandos y añadir nuevas filas a las tablas.

#### 2.5.3.4. Python

Python es un lenguaje de programación orientado a objetos. Es ampliamente conocido por lo sencillo de su sintaxis, que permite el desarrollo rápido de scripts por programadores novatos, llegando a ser enseñado a estudiantes de computación de varias universidades del mundo. Además, debido a que es un lenguaje interpretado, no se necesita compilar el código, pues puede ser ejecutado de forma instantánea por el intérprete de Python [30].

Para el desarrollo web en Python, existen librerías como Django y Flask. De estas dos se escoge Flask para llevar a cabo el proyecto, pues Flask es más fácil de usar y requiere menos configuración para implementar una página web, lo cual se puede ver al comparar el código necesario para montar una página web sencilla entre ambas librerías [31, 32]. Por tal razón, es llamado un *microframework* [33]. Flask permite crear de forma dinámica documentos HTML, apoyándose en el lenguaje Jinja 2.

Lamentablemente, Flask por sí solo no permite desplegar un sitio web profesional, dado que no se puede comunicar con un servidor web como Nginx y, en consecuencia, no puede aceptar muchas conexiones al mismo tiempo ni está optimizado para trabajar de forma simultánea en varios núcleos. Debido a esto, es necesario utilizar una interfaz que pueda conectar ambos programas, tal interfaz es llamada WSGI. Un WSGI (Web Server Gateway Interface) es un programa que comunica el servidor web con el framework web [34].

En el desarrollo de esta memoria se utilizará el WSGI Gunicorn como interfaz, el cual permite conectar una aplicación web de Python con servidores HTTP en sistemas operativos Unix [35]. Se eligió este WSGI debido a que en Internet existen tutoriales que explican paso a paso la configuración de un servidor web usando Nginx, Gunicorn y Flask [36]. Cabe mencionar que para la elección del WSGI no se consideraron características como la latencia, el uso de CPU o el número de conexiones simultáneas, pues en esta memoria no se busca precisamente hacer una página web rápida, ni capaz de soportar cientos de conexiones. Aún así, una comparación entre WSGI llevada a cabo en [37] concluyó que: *Gunicorn es bueno y se desempeña de forma consistente para cargas medianas*. Asimismo, en los gráficos de las comparaciones se observa que Gunicorn se encuentra en un nivel intermedio en la mayoría de las pruebas.

## 2.6. Estado del Arte

### 2.6.1. Desarrollos realizados en la academia y breve historia

A lo largo de los años han existido varias propuestas de distintos investigadores para crear sensores con posibilidad de conectarse de forma *plug and play* que sean fáciles de usar. Ya desde inicios del año 2000 han habido desarrollos en el área, gracias a la aparición de la familia de estándares IEEE 1451, los cuales buscan definir un grupo de interfaces comunes y abiertas para conectar transductores, microprocesadores y elementos de red [38]; de tal manera que el desarrollo de un sistema de sensores no se vea dificultado por equipos de medición, de procesamiento o de red incompatibles entre sí. Uno de los primeros artículos que explora de forma práctica este grupo de estándares, se encuentra en [39], en donde mediante el uso del TEDS se logra generar una simple funcionalidad plug and play, aunque, debido a su época de escritura, no presenta las capacidades de conexión inalámbrica ni visualización online.

Más adelante en el tiempo, a inicios de la década del 2010, surgen nuevas investigaciones en el área, ahora facilitado con un mayor crecimiento en el área de comunicaciones inalámbricas y mejores velocidades de conexión. En este período, los trabajos más importantes de mencionar para el desarrollo de la memoria, son [40] y [41]. El trabajo desarrollado en [40] presenta la funcionalidad plug and play de sensores usando el reconocido microcontrolador Arduino UNO, lo cual hace interesante este proyecto, pues este microcontrolador se encuentra en la mayoría de tiendas de electrónica y computación y, por ende, presenta la posibilidad de ser replicado fácilmente si se desea. Sin embargo, en este proyecto solo se presenta la solución propuesta para un sólo sensor y, además, toda la implementación es de forma cableada; por lo que queda ver su extensión a un mayor número de sensores. Por otro lado, en [41] se crea una red inalámbrica de sensores usando el protocolo ZigBee. Este trabajo presenta la funcionalidad *plug and play* para un grupo de sensores y la visualización de datos mediante

una interfaz gráfica, en donde, a través de servicios web, se pueden realizar acciones sobre los sensores. No obstante, en el artículo no se presenta la solución para dar una funcionalidad plug and play, sino que se da mayor énfasis a la construcción de la red y a la identificación de los sensores en esta.

Ya en los últimos 5 años, los trabajos realizados por investigadores alcanzan un gran nivel de complejidad. Empezando por [42], en donde se crea un sistema con arquitectura modular que se basa en una USI (Universal Sensor Interface), el cual permite la conexión de variados tipos de sensores a través de sus múltiples conexiones. Sumado a ello, cuenta con seis módulos *plug and play* que realizan el resto de funciones. Este desarrollo está centrado en el monitoreo de la contaminación del aire, por lo que aún falta testearlo en otros entornos. También, se debe nombrar a [43], en donde se desarrolló una red inalámbrica de sensores y actuadores, la cual tiene funcionalidad *plug and play* tanto por el lado de hardware como por el lado de software de la red. Este artículo, sin embargo, está orientado a redes complejas, en las que la red debe descubrir de forma automática nuevos equipos; en este trabajo en cambio, se prioriza la conexión manual de las interfaces, pero con una detección e identificación automáticas de los sensores.

Por último, un trabajo muy novedoso en esta área es el realizado en [44], en donde se implementa un plataforma de sensores inalámbricos mediante módulos. Estos módulos pueden tener las funciones de medición, comunicación, procesamiento, alimentación, entre otros. Los módulos son fácilmente integrables en una solución de *IoT*, ya que funcionan como distintos bloques que se pueden conectar a otros circuitos, lo que permite que sean reutilizables y flexibles. En cuanto a la funcionalidad *plug and play*, este trabajo es sumamente complejo y es capaz de resolver una gran cantidad de problemas, sin embargo, su solución no presenta el componente de visualización en una página web, que si presenta la propuesta de esta memoria.

### 2.6.2. Productos comerciales

Dentro de los productos comerciales que facilitan el desarrollo de proyectos de *IoT*, se encuentra el producto WisBlock, desarrollado por la empresa RAK Wireless. WisBlock es un sistema modular que permite implementar redes inálambricas de baja potencia en una solución de *IoT* [45]. Presenta muchos módulos que se encargan de diferentes trabajos: WisBlock Base se encarga de energizar y permitir la conexión de otros módulos; WisBlock Core da el procesamiento y la comunicación mediante LoRaWAN; WisBlock Sensor provee muchos tipos de sensores; entre otros. Los módulos se pueden conectar a través de tornillos y se pueden programar mediante el IDE de Arduino. No obstante, no presenta la capacidad de visualización inmediata de los datos en una interfaz gráfica; esto debe ser desarrollado por el mismo usuario. WisBlock sólo provee la funcionalidad *plug and play* por el lado del hardware.

Por otro lado, desde la empresa Upswift, se ofrece el servicio de gestión de dispositivos inteligentes desde un sólo lugar [46]. Permiten el control remoto, el monitoreo y la observación de equipos de *IoT* a través de su plataforma web, facilitando el trabajo de desarrolladores de software e ingenieros informáticos. Sin embargo, la implementación de un proyecto de *IoT* por el lado de hardware no es parte de sus servicios, por lo que el usuario debe encargarse de ello.

# Capítulo 3

# Diseño y Construcción de Interfaces Modulares

## 3.1. Sensores

Para este proyecto se decidió utilizar tres sensores: el sensor de temperatura DS18B20, sensor de distancia HCSR-04 y el sensor de luz TEMT6000. Se eligieron estos sensores por dos razones: en primer lugar, son fáciles de conseguir en el mercado chileno, ya que, muchas tiendas de electrónica cuentan con ellos. Y en segundo lugar, porque existen muchos tutoriales de uso, dado que son sensores ampliamente reconocidos. En esta sección se procederá a detallar las especificaciones de cada sensor utilizado, para luego mostrar las conexiones de estos con la Raspberry Pi.

### 3.1.1. Sensor de Temperatura DS18B20

El DS18B20, que aparece en la figura 3.1, es un sensor de temperatura digital que utiliza el protocolo 1-Wire. El protocolo 1-Wire permite la transmisión de datos digitales a través de un único pin. Requiere una alimentación desde los 3.0V a los 5.5V, y es capaz de medir temperaturas de -10ºC a 85ºC con una precisión de 0.5ºC [47]. El sensor DS18B20 cuenta 3 pines:

- GND: Tierra.
- DQ: Pin que envía los datos a través del protocolo 1-Wire.
- VDD: Pin de alimentación de 3.0V a los 5.5V.

Debido a que 1-Wire es un protocolo especial de comunicación, en los kernel del sistema operativo Raspberry Pi OS más antiguos (previos a la versión 4.9.28 del kernel [48]), sólo se podía usar el pin 7 de la Raspberry para la comunicación 1-Wire. Esto hubiese impedido la conexión de este sensor de forma PnP en cualquier conector, de tal manera que no se hubiese cumplido uno de los objetivos de la memoria. Afortunadamente, los nuevos kernel permiten el uso del protocolo 1-Wire en varios pines al mismo tiempo, e incluso, agregar nuevos buses 1-Wire de forma dinámica [48]. Por lo que para hacer funcionar el sensor, fue necesario actualizar el kernel de la Raspberry Pi.

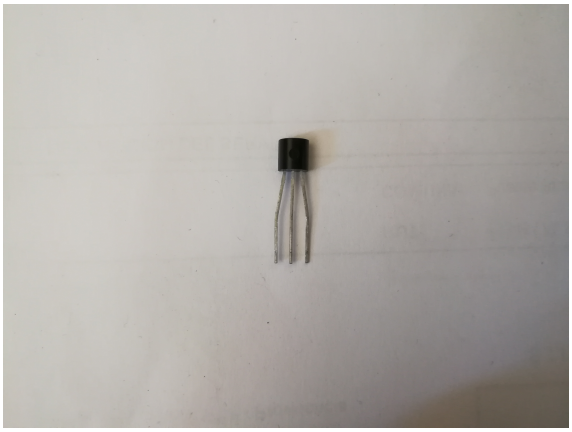

Figura 3.1: Sensor de temperatura DS18B20. Dimensiones: 5mm x 4mm x 12 mm.

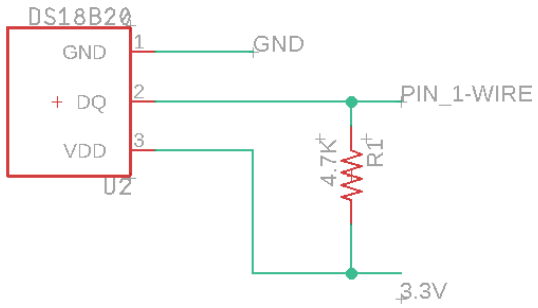


Figura 3.2: Esquemático de sensor de temperatura DS18B20.

Las conexiones del sensor se presentan en la figura 3.2. El pin VDD se conecta a 3.3V, el pin GND a tierra y el pin DQ a un pin Raspberry Pi para la comunicación 1-Wire. Cabe la pena destacar que el protocolo 1-Wire requiere una resistencia de pull up de 4.7 kΩ, que se utiliza para poner el pin DQ en voltaje alto.

### 3.1.2. Sensor de Distancia HC-SR04

El HC-SR04 (figura 3.3) es un sensor de distancia digital con un rango de medición de 2 cm a 400 cm, y una precisión de hasta 3mm [49]. Este sensor funciona en base al envío de una señal ultrasónica a través del módulo transmisor y la recepción de esta misma en el módulo receptor. Cuenta con 4 pines:

- VCC: Voltaje de alimentación de 5V.
- Trig: Pin de entrada. Envía una señal de ultrasonido al recibir un pulso de $10\mu s$ [49] desde el microcontrolador.
- Echo: Pin de salida. Envía una señal de nivel alto (5V) cuando detecta el retorno de la señal ultrasónica.
- GND: Tierra.

Para calcular la distancia, basta con saber la duración del viaje de ida y regreso de la señal, y conocer la velocidad del sonido en el aire. Sea $t$ el tiempo de ida y vuelta, $d$ la distancia del sensor al obstáculo y $v$ la velocidad del sonido en el aire ($343m/s$ o $34300cm/s$), entonces:

$$d = \frac{v \cdot t}{2} \tag{3.1}$$

$$d = \frac{34300 \cdot t}{2} \quad [cm] \tag{3.2}$$

$$d = 17150 \cdot t \quad [cm] \tag{3.3}$$

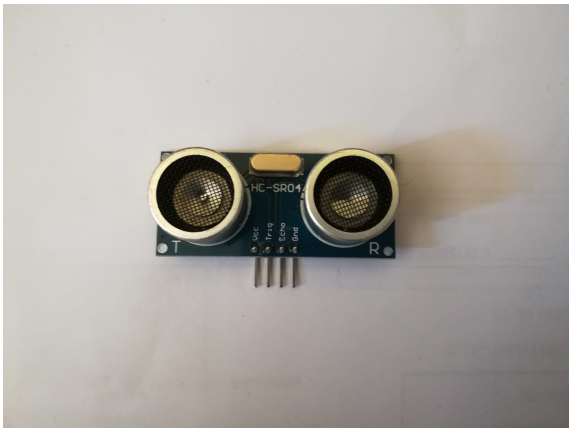


Figura 3.3: Sensor de distancia HC-SR04. Tamaño: 43 x 20 x 17 mm.

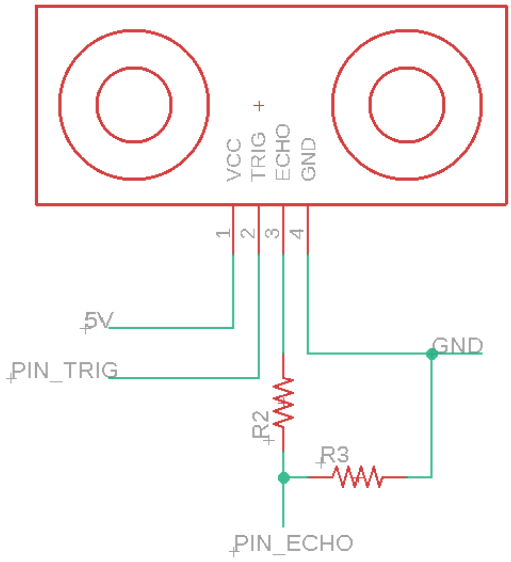


Figura 3.4: Esquemático de sensor de distancia HC-SR04.

Un cálculo similar a este se realiza en la librería `gpiozero` de Python, que calcula la distancia del obstáculo al sensor de forma automática.. Además, se envía el pulso de $10\mu s$ y se recibe la señal del pin ECHO del sensor.

El esquemático de este sensor se muestra en la figura 3.4. El pin VCC se conecta a 5V, el pin GND se conecta a tierra y el pin TRIG se conecta a un pin de la Raspberry. En cuanto al pin ECHO, debido a que este entrega una señal de 5V de salida, es necesario ocupar un divisor de voltaje por medio de resistencias para disminuir su valor a 3.3V, dado que los pines de la Raspberry sólo soportan 3.3V como máximo. Posteriormente, este pin se conecta a otro pin de la Raspberry Pi.

### 3.1.3. Sensor de Luz TEMT6000

El sensor de luz TEMT6000 (figura 3.5) es un sensor analógico que consiste en un fototransistor que mide el nivel de luz del ambiente [50]. Este sensor mide la iluminancia, que consiste en el flujo luminoso (lumen) incidente en una superficie por unidad de área [51]. La iluminancia se mide en lux y se denota como: $lux = lm/m^2$; es decir, un lux es un lumen por metro cuadrado. Debido a que el sensor sólo mide el flujo de luz por área, para él es equivalente, una fuente de luz brillante situada lejos, que una fuente de luz débil que se encuentre más cerca.

Como se ve en la figura 3.5, se cuenta con un circuito dentro del cual está el sensor en sí. Este circuito entrega directamente el valor del voltaje de salida, gracias a que una resistencia de 10kΩ actúa como un divisor de voltaje, por lo que facilita las lecturas de luz. Este sensor cuenta con 3 pines:

- SIG: Pin que entrega la señal de salida.

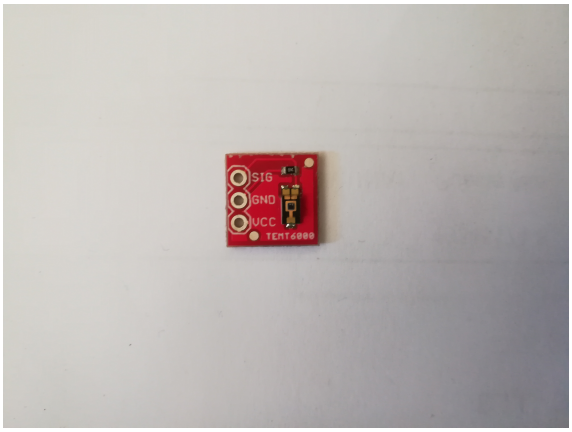


Figura 3.5: Sensor de luz TEMT6000. Tamaño: 9 x 9 x 2 mm.

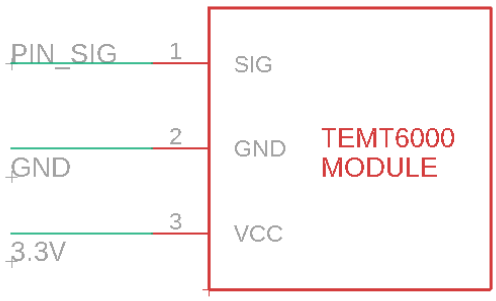


Figura 3.6: Esquemático de sensor de luz TEMT6000.

- GND: Tierra.
- VCC: Voltaje de alimentación que funciona a 5V y a 3.3V.

La figura 3.6 muestra el esquemático de este circuito, en donde se puede ver que la conexiones son bastante sencillas: el pin GND se conecta a tierra, el pin VCC a un voltaje de 3.3V y el pin SIG se debe conectar a un ADC para transformar la señal analógica en una digital.

Para leer la iluminancia medida por este sensor hay que basarse en la figura 3.7, en donde aparece la corriente en función de la iluminancia. En primer lugar, se debe medir el voltaje en SIG, mediante el uso de un ADC en el caso de la Raspberry Pi. Luego, se utiliza la ley de Ohm para obtener la corriente a partir del voltaje y la resistencia de $10k\Omega$ del sensor:

$$I = \frac{SIG}{10000} \quad [A] \tag{3.4}$$

Luego, se pasa la corriente de amperios a microamperios, para que sea consistente con el gráfico:

$$I = 100 * SIG \quad [\mu A] \tag{3.5}$$

Finalmente, se multiplica la corriente obtenida por dos, de acuerdo a la relación que aparece en el gráfico:

$$\text{Iluminancia} = 2 * I \quad [lux] \tag{3.6}$$

## 3.2. Diseño de Interfaces PnP

Como fue posible ver en la sección anterior, cada sensor tiene distinto número de pines, voltaje de funcionamiento, entrega distinto tipo de señal, etc., e incluso, cada sensor tiene distinto comportamiento por el lado del software. Por lo tanto, se debe diseñar una interfaz plug and play en la que sea posible conectar un amplio número de sensores similares entre sí. Esta interfaz debe cumplir varias características:

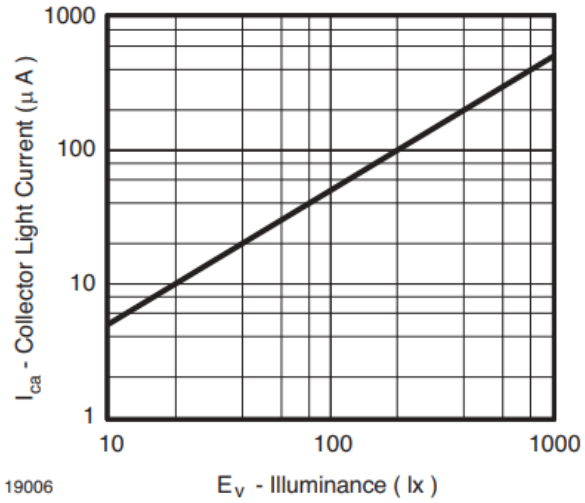


Figura 3.7: Gráfico de la Corriente medida en el sensor TEMT6000 vs la Iluminancia. Obtenido de [50]

1. Debe ser capaz de entregar distinto voltaje de alimentación dependiendo de las especificaciones de cada sensor.
2. Debe contar con la capacidad de aceptar sensores con distinto número de pines.
3. La interfaz debe ser de fácil conexión, estilo *plug and play*.
4. Debe funcionar para sensores digitales.
5. Debe funcionar para sensores analógicos.

Dadas estas cuatro restricciones, la solución ideada consiste en utilizar conectores eléctricos que sean posibles de conectar de forma *plug and play* y que tengan el número de pines suficientes para la conexión de diferentes tipos de sensores. Con el fin de facilitar el desarrollo de la memoria, se decidió usar conectores eléctricos distintos para los sensores digitales y los analógicos.

### 3.2.1. Sensores Digitales

Los sensores digitales utilizarán para la interfaz el conector DB9, el cual pertenece a la familia de conectores SUB-D por su forma similar a la letra D mayúscula. Como indica su nombre, el conector DB9 cuenta con nueve pines y se utiliza generalmente para conectar periféricos al computador y en el protocolo de comunicaciones RS232. En el módulo central se colocará un conector DB9 hembra, mientras que en las interfaces de los sensores se colocará un conector DB9 macho, ambos aparecen en la figura 3.8.

Para lograr una interfaz general para múltiples sensores, es necesario asignar a cada pin del conector una función específica, ya sea de alimentación, tierra, envío de datos, etc. Para este proyecto se decidió utilizar la asignación de pines que aparece en la tabla 3.1.

Se pasará a explicar el funcionamiento de cada pin:

Figura 3.8: Conectores DB9 macho y hembra.

| Pin | Función |
|---|---|
| 1 | GPIO 1. |
| 2 | Alimentación de 5V. |
| 3 | GPIO 2. |
| 4 | Alimentación de 3.3V |
| 5 | GPIO 3. |
| 6 | GND. |
| 7 | Identificación del sensor (ID). |
| 8 | GND. |
| 9 | Estado de conexión (STATE). |

Tabla 3.1: Pines del conector DB9 y su funcionalidad.

- Los pines de alimentación entregan el voltaje de 5V o 3.3V dependiendo del requerimiento del conector.
- Los pines GND conectan a tierra.
- El pin ID se utiliza para identificar al sensor. Cada sensor utilizado debe tener un número de identificación único, para que así la Raspberry Pi pueda saber cuál sensor esta siendo conectado y reconocerlo.
- El pin de estado de conexión STATE, indica si el sensor está conectado o no.
- Los pines GPIO son de propósito general y se pueden ocupar según lo requiera cada sensor.

Es necesario explicar en mayor detalle el funcionamiento del pin STATE. Este pin se conecta a una resistencia de pull down, de manera que en estado de reposo su voltaje es BAJO. Cuando se conecta una interfaz al conector se cierra el circuito y el pin queda conectado a un voltaje de 3.3V, por lo que su voltaje pasa a ser ALTO. De esta manera, cuando el voltaje del pin pasa a ser ALTO, se sabe que se ha conectado una interfaz al conector.

### 3.2.2. Sensores Analógicos

Para los sensores analógicos se utilizarán conectores Molex de 5 pines, su versión macho aparece en la figura 3.9 y su versión hembra aparece en la figura 3.10. Estos son conectores

son plug and play, ya que simplemente el conector macho se debe insertar en el conector hembra para funcionar. Se decidió utilizar conectores distintos para los sensores digitales y los analógicos debido a que la lectura de ambos tipos de sensores es muy distinta. Como la Raspberry Pi no tiene conversor analógico digital, se requiere utilizar un ADC externo para leer los datos de los sensores analógicos. Diseñar la electrónica necesaria para crear una interfaz capaz de leer los 2 tipos de sensores a la vez, hubiese complicado innecesariamente el trabajo. Por eso se prefiere separarlos utilizando conectores distintos.

Figura 3.9: Conector macho de cinco pines.

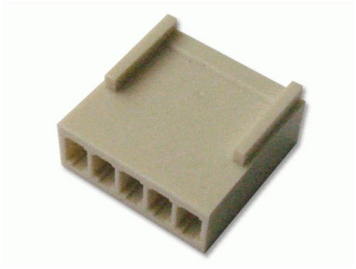

Figura 3.10: Conector hembra de cinco pines.

Los sensores analógicos generalmente tienen una electrónica más sencilla que los sensores digitales, puesto que envían sus mediciones a través de un sólo pin y no necesitan otros pines GPIO como si lo requieren los sensores digitales. Por lo que el conector necesita cinco pines solamente. En la tabla 3.2, aparece la funcionalidad de los cinco pines del sensor. Como se puede ver, las funciones de los pines son muy similares a las que tiene la interfaz de los sensores digitales. La principal diferencia es que no hay pines GPIO, ya que como los sensores analógicos generalmente sólo tienen un pin de salida que entrega los datos, no necesitan tener tantos pines de entrada y de salida. La salida del sensor está denotada por el pin OUT. Este pin luego se conecta a un ADC para entregar a la Raspberry el valor digital de la salida. Además, no se necesita el pin ID, dado que por ahora sólo se cuenta con un sensor analógico.

| **Pin** | **Función** |
|---|---|
| 1 | Alimentación de 5V. |
| 2 | Alimentación de 3.3V. |
| 3 | Tierra. |
| 4 | Salida del sensor (OUT). |
| 5 | Estado del sensor (STATE). |

Tabla 3.2: Pines del conector de cinco pines y su funcionalidad.

### 3.2.3. ADS1015

Para la lectura de los sensores analógicos se utilizará el conversor analógico digital ADS1015 (figura 3.11). Este ADC transmite los datos a través del protocolo I2C y es capaz de conectarse hasta con cuatro sensores analógicos al mismo tiempo, dado que posee cuatro pines de

entrada [52]. En el PCB, el ADC se encuentra en el módulo central, conectado directamente a la Raspberry Pi.

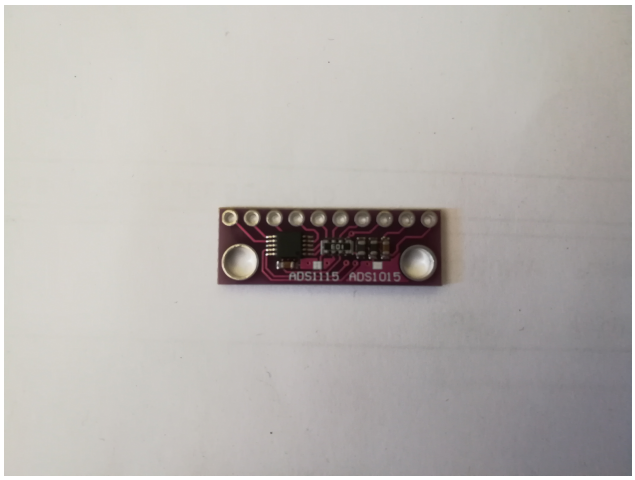


Figura 3.11: Conversor analógico digital ADS1015.

Los pines que tiene el ADC son los siguientes:

- VDD: Voltaje de alimentación, funciona de 3V a 5V.
- GND: Tierra.
- SCL: Señal de reloj para protocolo I2C.
- SDA: Envío y transmisión de datos a través de protocolo I2C.
- ADDR: Se utiliza para cambiar la dirección de I2C del equipo.
- ALRT: Salida del comparador.
- A0, A1, A2, A3: Entradas analógicas.

Mediante software es posible seleccionar que entrada leer, y el ADC la entregará a través del pin SDA. El pin ADDR, se utiliza para cambiar la dirección del ADC en el protocolo I2C por si se tiene otros equipos con la misma dirección. Es posible cambiar la dirección del equipo entregando al pin ADDR una de las cuatro siguientes señales: GND, VDD, SCL y SDA.

## 3.3. Diseño de PCB

Para diseñar la placa se utiliza el programa de diseño electrónico Eagle. Eagle permite diseñar esquemáticos de circuitos y luego diseñar el PCB que contendrá el circuito. Primeramente, se diseñó el esquemático del circuito, creando las interfaces de los sensores y el módulo central.

### 3.3.1. Esquemáticos

Para el módulo central se diseñó el circuito que aparece en la figura 3.12. La Raspberry Pi aparece en el centro de la figura y, como se puede ver, se utilizan gran parte de sus pines, de los 40 que tiene se utilizan 27. Esto es debido a que utiliza los pines de alimentación y, además, está conectada a dos conectores DB9 y al ADC. Se puede ver que los pines de cada conector están organizados tal como se mencionó en las tablas 3.1 y 3.2. De igual manera, aparece el conversor análogo digital ADS1015, que funciona de interfaz entre las salidas analógicas de los conectores Molex y la Raspberry Pi, para que esta puede leer los datos de conectores analógicos. Asimismo, la hoja muestra los agujeros de montado que van en el módulo central, para después atornillar la Raspberry Pi a la placa.

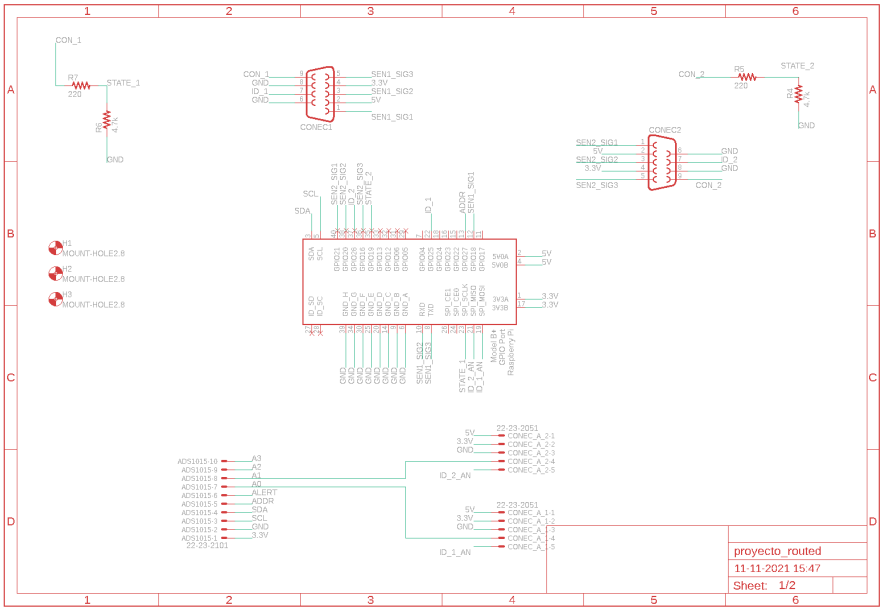


Figura 3.12: Esquemático del módulo central.

Por otra parte, las interfaces de los sensores se muestran en la figura 3.13. Se puede ver que hay 3 bloques separados, cada uno corresponde a un sensor. El número 1 corresponde a la interfaz del sensor de temperatura DS18B20. El número 2 corresponde a la interfaz del sensor de distancia HC-SR04. Por último, el número 3 es la interfaz del sensor de luz TEMT6000. Se puede ver en la figura cómo las interfaces se adaptan para cada sensor. Por ejemplo, para el sensor de temperatura la conexión a 5V está al aire y el sensor sólo recibe la alimentación de 3.3V; mientras que para el sensor de distancia es al revés. De esta manera, gracias a la organización de los pines, se logra una interfaz plug and play adaptable a los requerimientos de cada sensor.

Se puede ver también en la figura 3.12 que los pines del ADS1015 también están conectados a la Raspberry Pi. En particular el pin ADDR del ADS1015 está conectado de forma directa a un pin de la Raspberry Pi. El valor de voltaje de este pin se debe poder fijar en las 4 señales aptas para ADDR, las cuales son: GND, VDD, SCL y SDA.

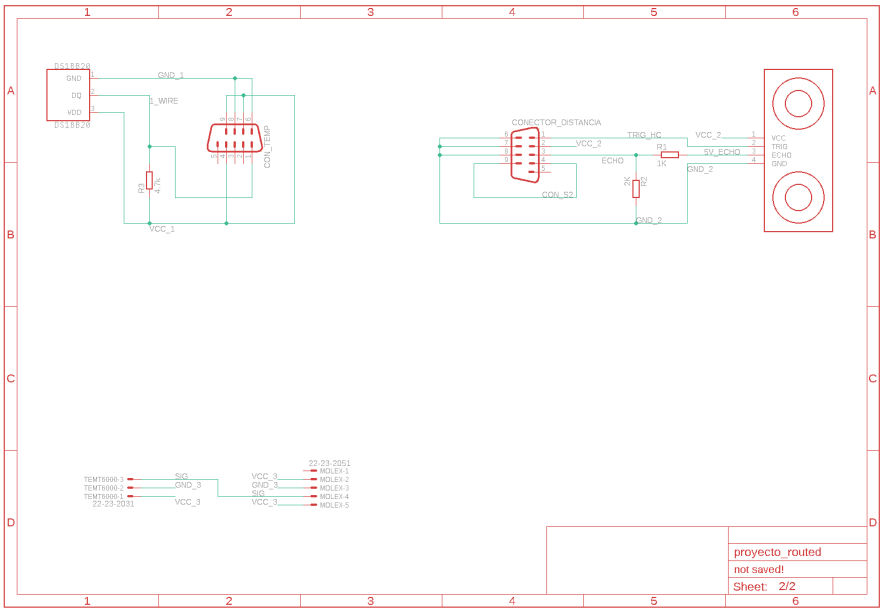


Figura 3.13: Esquemático de los 3 sensores.

### 3.3.2. Placa PCB

Luego de diseñar el circuito, se procedió a diseñar la placa que llevaría el circuito. En la figura 3.14 se muestra la placa que contiene el circuito del módulo central y de los 3 sensores. Los 4 circuitos se encuentran dentro de la misma placa, pero las líneas de fresado que se encuentran en el diseño, permitirán que cuando se construya el PCB cada una de las partes sea fácilmente separable. Por esta misma razón pareciera que algunos componentes están superpuestos unos sobre otros, pero cuando se corte la placa quedaran separados. En el módulo central también aparecen los agujeros de montado que luego servirán para colocar el sistema en una base de plástico u otro material.

En la figura 3.14 se dibujaron 4 recuadros para indicar cada una de las placas independientes: el número 1 designa la placa del módulo central, el segundo recuadro corresponde a la placa del sensor de temperatura, el tercer recuadro es del sensor de distancia y el recuadro 4 corresponde al sensor de luz. Se decidió utilizar una sola placa para los 4 circuitos ya que uno de los objetivos de la memoria es que el proyecto sea de bajo costo económico y el mandar a construir muchas placas de forma independiente cuesta más dinero que mandar a hacer solo una. Por la misma razón, la placa sólo mide 85mm x 105mm, con el tal de que tenga el menor tamaño posible.

En la placa de PCB aparecen las rutas de cobre. Las líneas de color rojo designan las pistas de cobre que se encuentran en la parte superior de la placa, mientras que las líneas de color celeste corresponden a pistas de cobre en la parte inferior de la placa. Se dibujaron pistas de cobre más anchas para las líneas de alimentación, es decir, las pistas que conectan a 3.3V, 5V o GND. El ancho de estas pistas es de 0.508 mm, mientras que el ancho del resto de pistas es de 0.254, es decir, la mitad de ancho. Los pequeños agujeros de color verde representan a las *vias*, que son agujeros que permiten el paso de la pista de cobre de un lado de la placa al otro. Se utilizan cuando ya no queda espacio por donde pasar un cable. Por último, cabe

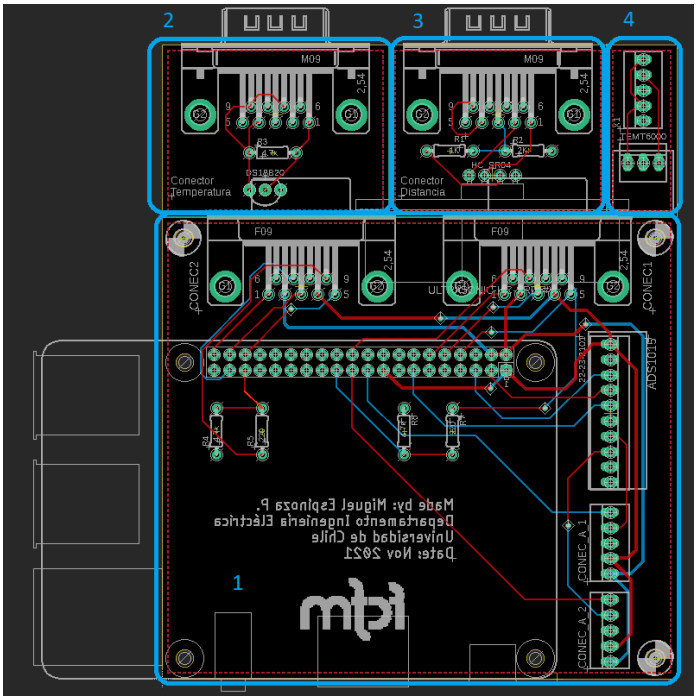


Figura 3.14: Placas PCB del módulo central y los sensores.

mencionar que en el diseño del PCB se dibujó un polígono alrededor del borde de las 4 placas, de forma similar a los recuadros de color celeste de la figura 3.14, para colocar la señal de tierra GND.

De esta manera, el resultado de la placa impresa debe quedar como aparece en la figura 3.15 para la parte superior y como la figura 3.16 para la parte inferior. Se puede ver que en la parte inferior de la placa aparece el nombre del estudiante, del Departamento, de la Universidad y la fecha en que fue diseñada la placa, junto con el logo de la Facultad. Este texto es con el fin de identificar la placa y destacar que es un proyecto del alumno del Departamento.

La construcción de la PCB y el soldado de los equipos se presenta en el anexo A.

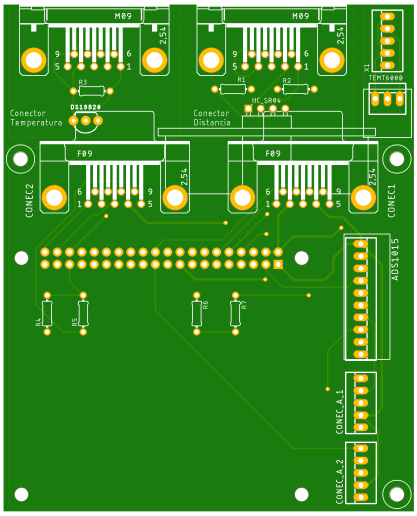


Figura 3.15: Lado superior de la placa PCB.

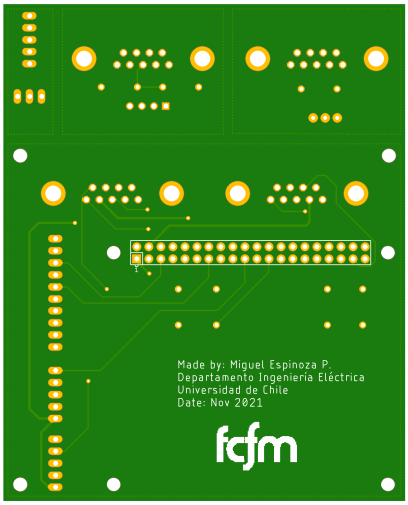


Figura 3.16: Lado inferior de la placa PCB.

# Capítulo 4

# Software

Esta sección describe el código que hace funcionar el proyecto. Dentro de él, se describen los archivos de Python encargados de la programación del proyecto, la base de datos y los archivos de configuración de redes e interfaces, entre otros.

Para poder leer los datos de los sensores conectados al módulo central y guardarlos en la base de datos, se escribieron dos archivos de Python, uno para cada tipo de conector, que realizaban las siguientes tareas:

1. Detectar cuando una interfaz se conectaba al módulo central.
2. Identificar la interfaz conectada, es decir, saber cuál es el sensor que se ha conectado.
3. Leer la medición del sensor.
4. Guardar la medición en la base de datos.

Con el fin de ser ordenado, se explicará cada paso por separado. Pero antes, se comenzará describiendo la estructura del código y los paquetes creados.

## 4.1. Estructura del código

Existen 3 paquetes principales en el código: `conectores`, `database` y `webpage`. El paquete `conectores` se encarga del hardware del proyecto: la detección de la interfaces, la identificación de los sensores y la lectura de datos. El paquete `database` se ocupa de acceder a la base de datos y añadir nuevas mediciones. Por último, el paquete `webpage` está encargado de la implementación de la página web y se describirá en el siguiente capítulo.

Adicionalmente, existe una carpeta llamada `config` que contiene los archivos de configuración que se usaron para lograr el funcionamiento del proyecto. Dentro de ellos se encuentran archivos del sistema operativo de la Raspberry Pi que fueron modificados y archivos de configuración de servicios de Linux, los cuales se describirán posteriormente.

## 4.2. Implementación de conectores en el código

Esta sección es simplemente un resumen de la implementación. Si se desea conocer una explicación más detallada del código, dirigirse al anexo B.

Un diagrama UML de clases simplificado del paquete `conectores` aparece en la figura 4.1. La clase `Conector` representa de manera virtual a los conectores digitales del módulo central y la clase `ConectorAnalogico` representa a los conectores analógicos. Ambos tienen métodos que se encargan de comprobar si hay algún sensor conectado a la interfaz y, si es así, identificarlo. Se puede observar que los conectores digitales y los conectores analógicos están separados en el diagrama debido a su distinto funcionamiento. La clase `Pin` se encarga de guardar los números de los pines y su función. Como se puede ver, cada objeto de la clase `Conector` contiene 5 instancias de la clase `Pin`, los cuales representan los pines de un conector que pueden recibir o enviar señales digitales. Mientras que la clase `ConectorAnalogico` cuenta con solo un objeto de la clase `Pin`.

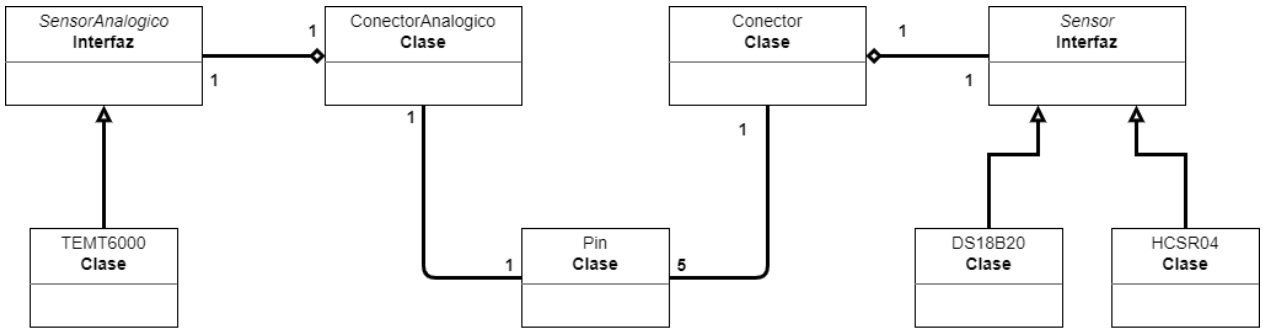


Figura 4.1: Diagrama de clases UML simplificada del proyecto.

Por otro lado, la clase `Conector` también tiene relación con la clase `Sensor`, puesto que cada conector puede tener un sensor conectado. La interfaz `Sensor` se encarga de proporcionar campos y métodos comunes a los sensores digitales que se utilizan en el proyecto y a futuros sensores digitales que se deseen añadir. La primera clase derivada de la interfaz `Sensor` es la clase `DS18B2O`, que representa al sensor de temperatura DS18B20. La segunda es la clase `HCSR04`, que simboliza al sensor de distancia HC-SR04. De igual manera, la clase `ConectorAnalogico` se relaciona con la interfaz `SensorAnalogico`, la cual cuenta con una sola clase derivada: la clase TEMT6000 que representa al sensor de luz TEMT6000. Cada clase que simboliza un sensor cuenta con los campos y métodos necesarios para llevar a cabo las funciones específicas del sensor y entregar el valor de sus mediciones.

### 4.2.1. Detección de interfaces

La detección de interfaces en ambos tipos de conectores se realiza detectando desde el código un cambio del voltaje del pin de estado de los conectores. Cuando este pin pase de BAJO a ALTO se sabrá que se ha conectado una interfaz de un sensor. Es importante mencionar que para la manipulación y lectura de los pines de la Raspberry Pi, se utiliza la librería *gpiozero* [53], la cual facilita tales tareas.

Debido a que una interfaz puede ser conectada o desconectada en cualquier momento por

el usuario y sustituida por otra, el código debe ser robusto ante estos cambios. Para ello, se definen 4 estados en los que puede estar el programa:

1. **Ningún conector tiene conectada una interfaz**: en este caso, ambos conectores estarán esperando que se conecte una interfaz para detectarla. Por lo tanto, los conectores deberán calcular de forma periódica si el pin de estado pasó de BAJO a ALTO.

2. **Sólo el conector 1 tiene conectada una interfaz**: aquí, el programa debe estar pendiente de cuando se desconecta la interfaz del conector 1 y cuando se conecta una interfaz al conector 2. Para detectar cuando se desconecta una interfaz, se debe detectar el cambio de voltaje del pín de ALTO a BAJO.

3. **Sólo el conector 2 tiene conectada una interfaz**: es equivalente al caso 2, sólo que al revés.

4. **Ambos conectores tienen conectada una interfaz**: en este caso, ambos conectores deben estar detectando cuando se desconecte una interfaz.

Es importante mencionar que se chequea el estado de los pines de estado cada 1 s y luego se pone el código en estado de suspensión hasta que pase otro segundo.

### 4.2.2. Identificación del sensor

La identificación de los sensores digitales se realiza leyendo el valor del pin de identificación. El sensor de temperatura DS18B20 tiene el ID 1, lo que quiere decir que cuando está conectado, el pin de identificación debe tener un voltaje ALTO. Por otro lado, el sensor de distancia HC-SR04 tiene el ID 0, por lo que cuando está conectado, el valor del pin de identificación debe ser BAJO. Es importante leer el valor del pin de identificación sólo cuando se verifique hay un sensor conectado, puesto que el pin de identificación no posee una resistencia de *pull up* o *pull down*, por lo que su voltaje cuando no hay ningún sensor conectado es ambiguo.

Por el lado del conector analógico, ya que sólo hay un sensor funcionando, no es necesario hacer la identificación, sino que cuando se conecta un sensor, se presume automáticamente que es el sensor de luz TEMT6000.

### 4.2.3. Lectura de mediciones

La lectura de mediciones las realiza cada subclase de `Sensor` y `SensorAnalogico` internamente, ya que debido a que los sensores tienen funcionamiento distinto, cada uno tiene su implementación única para la medición de datos.

### 4.2.4. Acceso a la base de datos

La base de datos se realizó con MariaDB. Este programa no tiene una interfaz gráfica, por lo que se maneja solamente a través de comandos, los que pueden ser escritos por medio de la Terminal de Linux o la PowerShell de Windows. Dentro de MariaDB se crea una base de datos llamada `sensores` y, en ella, una tabla para cada sensor. Cada una de las tablas cuenta

con 3 columnas: el id de la medición, el tiempo, y el valor de la medición.

En el código el acceso a la base de datos se realiza en el paquete `database`. Este paquete ejecuta las tareas de conectarse a la base de datos, extraer datos ya almacenados y añadir nuevos datos. Afortunadamente, es posible realizar estas tareas en Python gracias a la librería MariaDB Connector.

Por último, es importante mencionar que se envían mediciones a la base de datos cada 10 segundos, con el fin de no sobrecargar la base de datos con nuevos datos a cada instante.

### 4.2.5. Registro de ejecución

Se realiza un registro del programa usando la librería `logging` de Python. Esto permite que se pueda verificar el correcto funcionamiento del proyecto y que se puedan revisar posteriormente los errores ocurridos, pues el registro se almacena en un archivo de texto.

En la figura 4.2 se muestra un ejemplo del registro. Como se puede apreciar, el registro muestra la fecha y la hora, el archivo de Python que generó el mensaje, el nivel de severidad y, finalmente, el mensaje en sí. Todas estas capacidades de la librería `logging` dan la posibilidad de saber cuando y donde se produjo un problema en el programa y, en consecuencia, tener todas las herramientas para poder replicarlo y arreglarlo.

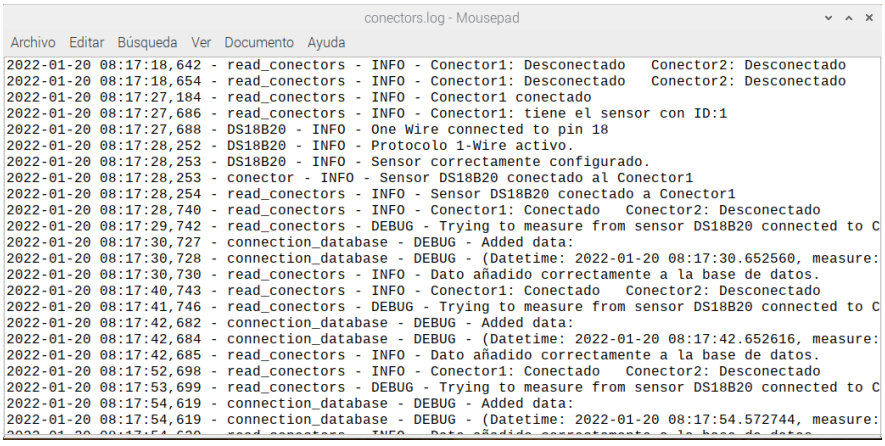


Figura 4.2: Captura de pantalla del archivo de registro que se encuentra en la Raspberry Pi.

### 4.2.6. Comunicación con el servidor web

El código del software no se comunica directamente con el servidor web, sino que se comunica a través de archivos de apoyo JSON que guardan el estado de los conectores. El mismo script que identifica los sensores y lee sus datos, se encarga de actualizar el estado de los conectores cada un segundo en un archivo de texto. El servidor, por su parte, también accede a este archivo para conocer el estado de los conectores y mostrarlo en la página web. Existe un archivo de apoyo JSON distinto para los conectores analógicos y para los conectores digitales.

## 4.3. Archivos de configuración

Con el fin de tener un acceso sencillo a la configuración de la Raspberry Pi que se usó para implementar el proyecto, todos los archivos de configuración del sistema operativo se guardaron en una carpeta llamada `config`. Los archivos que se configuraron fueron los referentes a:

- El arranque de la Raspberry Pi, para habilitar la interfaz I2C y la resolución HDMI.
- La configuración de red, para darle una IP estática a la Raspberry.
- El protocolo 1-Wire, con el fin de modificar los atributos del protocolo.

Además, se crearon tres servicios de `systemd` para llevar a cabo tareas del proyecto. Los servicios son programas que se ejecutan en segundo plano mientras está encendido el computador, realizando tareas periódicas, escuchando puertos, esperando la conexión de dispositivos a entradas, etc. Además, el programa administrador `systemd` permite que los servicios se inicien automáticamente al arrancar el sistema operativo. A los servicios también se les llamada *daemon*. Los tres servicios que se crearon durante esta memoria fueron: `conectors.service`, `analog_conectors.service` y `gunicorn_project.service`. Sólo se hablará de los dos primeros, ya que el último se explicará en el siguiente capítulo, pues esta relacionado con el funcionamiento de la página web.

El servicio `conectors.service` realiza todas las tareas relacionadas a la detección de la conexión de sensores a las interfaces, la identificación del sensor, la lectura de la medición y el envío de datos a la base de datos. Los pasos realizados para la creación de este servicio aparecen en el anexo B, al igual que el resto archivos de configuración. El servicio `analog_conectors.service` trabaja de igual manera, pero para los conectores analógicos.

# Capítulo 5

# Diseño e Implementación de la Interfaz Gráfica

## 5.1. Implementación de la página web

En la figura 5.1 aparece la estructura de la página web. En la figura se nombran las tecnologías que se encargan de cada una de las tareas que realiza el back end y el front end del sitio web. En las siguientes secciones se explicará en detalle la configuración y el funcionamiento de cada una de estas tecnologías.

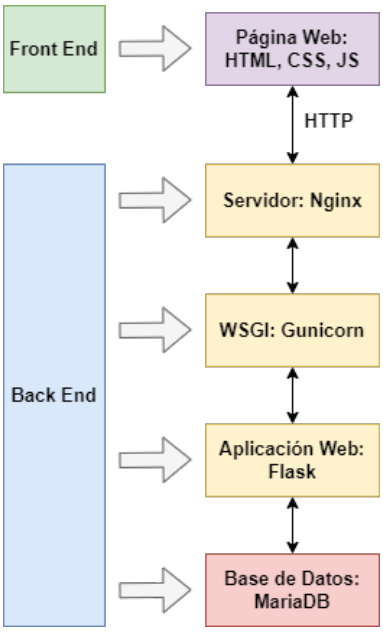


Figura 5.1: Estructura de la página web creada. Notar que el front end y el back end se comunican a través de HTTP.

### 5.1.1. Back End

El back end consiste en la aplicación de Flask, el WSGI Gunicorn, el servidor Nginx y la base de datos MariaDB. El funcionamiento de MariaDB ya fue descrito en el Capítulo 4, por

lo que no se explicará nuevamente.

#### 5.1.1.1. Flask

La aplicación de Flask contiene todo el código de la página web y se encarga de manipular y transmitir los datos requeridos al front end. Se ocupa de recibir las peticiones del navegador, de adquirir las últimas mediciones de la base de datos y de enviarlas de forma estándar, por medio de archivos JSON. Puede funcionar sin configuración previa, ya que se puede crear una página web de prueba simplemente corriendo la aplicación de Flask en modo `debug` en la consola. Sin embargo, debido a que no viene con un servidor especializado, no sirve para una página web real.

El manejo de la aplicación web por parte de Flask, funciona mediante la vinculación de rutas URLs a funciones, esto se le llama *enrutamiento* [54]. Por lo tanto, cuando el usuario ingresa una URL en la barra de direcciones, la aplicación web de Flask ejecuta una función asociada a tal URL. Por ejemplo, la URL `/` se vincula a la función que entrega la página web principal del sitio web. La implementación de las rutas se encuentra en el anexo C.

Cabe destacar que en el sitio web que diseñó, el usuario nunca ingresa a mano estas URL mencionadas, sino que al presionar ciertos botones o menús que se encuentran en la página web, el código de JavaScript se comunica con el back end por medio de AJAX y llama a estas URL. En consecuencia, Flask ejecuta las funciones vinculadas y entrega el resultado a JavaScript, el cual modifica la página web acorde al resultado de la función. El usuario desconoce totalmente lo que pasa internamente, él solo ve el cambio en la página web al pulsar un botón.

#### 5.1.1.2. Gunicorn

En esta sección se describe la configuración del WSGI Gunicorn, el intermediario entre Flask y Nginx. Para que funcione se debe crear en primer lugar un punto de entrada desde donde acceder a la aplicación de Flask.

Luego de esto, se crea un servicio `systemd` de Unix que cargue automáticamente Gunicorn al encender la Raspberry Pi y transmita las solicitudes HTTP al puerto de entrada ya creado. De manera tal que cuando esté funcionando el servidor Nginx, Gunicorn envíe automáticamente las peticiones a Flask y este entregue la página web. Los servicios de `systemd` se inician al prender la Raspberry, por lo tanto, ya no será necesario correr el script desde la consola cada vez que se quiera mostrar la página web, sino que basta con prender la Raspberry Pi para que la página web ya esté funcionando. Al archivo de `systemd` que se crea se le llama `gunicorn_project.service`. Dentro de este servicio se crea un socket que trabaja en segundo plano y se encarga de comunicar a Flask con Nginx.

La implementación del punto de entrada y del servicio se encuentra en el anexo C.

#### 5.1.1.3. Nginx

La base del back end es el servidor HTTP Nginx. Este servidor recibe todas las peticiones HTTP del navegador en su versión más cruda y luego las transmite hacia arriba a la aplicación de Flask. Nginx se debe configurar de tal manera que transmita las peticiones al socket de

Gunicorn que fue creado. Su implementación se detalla en el anexo C.

### 5.1.2. Front End

El front end de la página web se desarrolló creando una archivo HTML que contenía la estructura de la página y desde el que se cargaban los archivos de CSS y los scripts de JavaScript. El CSS de la página se realizó prácticamente en su totalidad con la librería Bootstrap. Gracias a ella se diseñaron botones para actualizar el gráfico y menús *dropdown* para cambiar el número de datos mostrados en el gráfico y el tipo de sensor que se muestra en la página. El diseño de la tabla también fue hecho por medio de Bootstrap. El gráfico se desarrolló gracias a la librería Google Charts, usando un gráfico de línea (*linechart*). El panel que aparece en la parte superior de la página se actualiza de forma periódica para mostrar el estado de los conectores. En la figura 5.2 se muestra la página web con estos elementos.

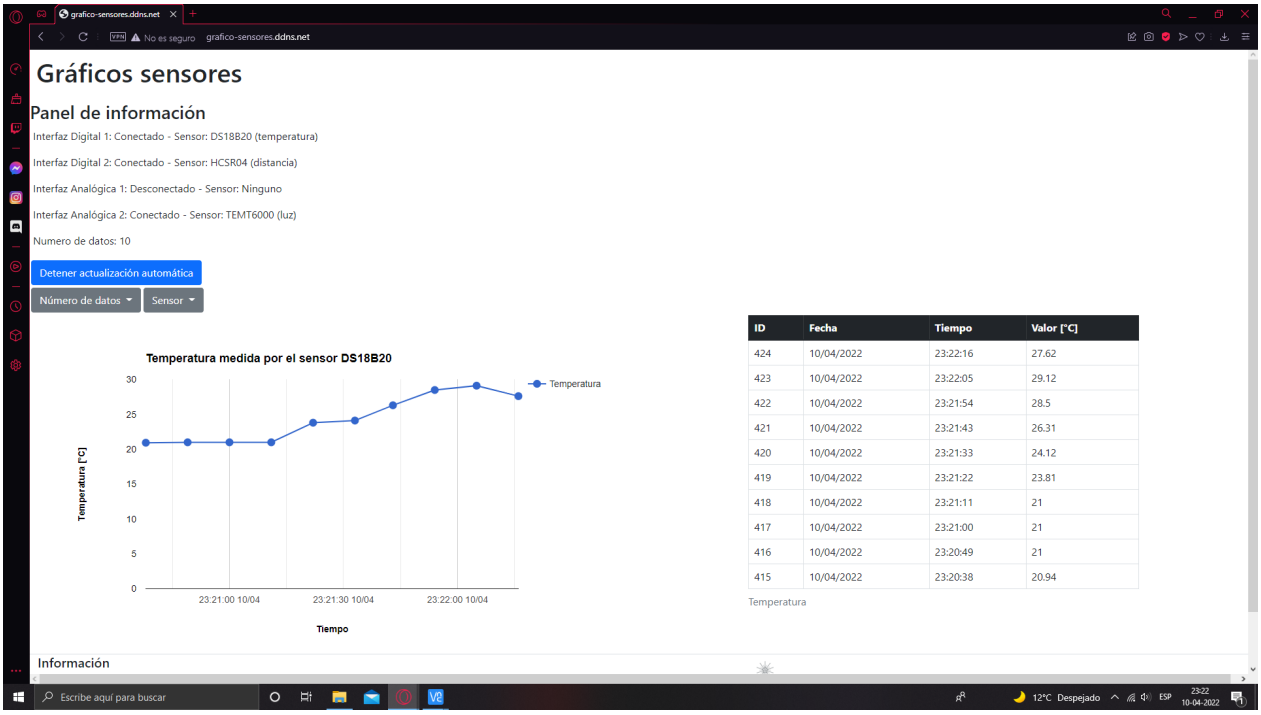


Figura 5.2: Página web construida.

Para dibujar el gráfico se creó una función que le da el estilo, el título, los nombres de los ejes, el color de la línea, etc. Además, se implementó una función auxiliar que actualiza el ancho del gráfico al cambiar la resolución de la página web. De tal manera que su tamaño sea acorde al espacio disponible. Para el resto de elementos, Bootstrap permite fijar su tamaño en función de la resolución de la pantalla.

Además, se crean 2 funciones que se ejecutan periódicamente cada 5 segundos, para actualizar los elementos dinámicos de la página web. La primera función se encarga de actualizar el panel de información de los conectores. La segunda se ocupa de actualizar el gráfico y la tabla si existen nuevos datos; para ello, ingresa a la base de datos para revisar si hay nuevas mediciones en el sensor seleccionado.

Por otro lado, los menús también están asociados a funciones. Cuando el usuario selecciona el sensor en el menú, se llama un evento que ejecuta una función que cambia el sensor mostrado en el gráfico y en la tabla. También, cuando se selecciona un número de datos distinto al actual, se llama a una función que accede a la base de datos para extraer ese número de datos y mostrarlos en la página web.

La comunicación entre el front end y el back end se realiza a través de AJAX. Como se mencionó previamente, Flask funciona a través de la vinculación de URL a funciones. Por lo que el código de JavaScript debe acceder a esas URL para que el back end ejecute esas funciones. AJAX es el que se encarga de ese trabajo. Además, permite la actualización de elementos de la página web de forma asíncrona cuando el navegador hace una solicitud. Es por esto que el gráfico y la tabla se actualizan de manera independiente a la página. No es necesario recargar toda la página para agregar un nuevo dato al gráfico.

### 5.1.3. Acceso desde Internet

Para acceder a la página web desde cualquier lugar del mundo, es necesario hacer visible la Raspberry Pi a Internet. Esto se logra conociendo la dirección pública del router de la red local y dejando a la Raspberry Pi expuesta a Internet, mediante la configuración de una zona desmilitarizada para ella. Así, se puede acceder a la página web ingresando solamente la dirección IP del router en la barra de direcciones del navegador web. Para mayor facilidad de uso, se consigue un servicio de DDNS que asigna un dominio al sitio web, por lo que el usuario puede acceder al sitio escribiendo el dominio y no la dirección IP.

Para configurar la Raspberry Pi en la zona desmilitarizada, en primer lugar se fija una IP estática para la Raspberry, de tal manera que no se vea afectada por el DHCP (Dynamic Host Configuration Protocol) del router. El DCHP modifica las direcciones IP de todos los dispositivos de la red local cada cierto tiempo, o cuando se añaden nuevos equipos a la red. Esto provoca que periódicamente se deba verificar si la dirección IP de un equipo sigue siendo la misma, perdiendo tiempo y recursos. Al fijar una IP estática en la red local, se evitan esos problemas, dado que esta dirección no cambiará nunca. La fijación de la IP estática se implementa de 2 maneras distintas, con el fin de que sea más robusta, mediante el archivo de configuración de red interno de la Raspberry y mediante la configuración del router. La implementación de ambos aparece en el anexo C.

Finalmente, el último paso para lograr que la Raspberry Pi sea accesible desde fuera de la red, es configurar la DMZ, lo cual es posible hacerlo desde el router, siguiendo los pasos que se muestran en el anexo C.

Con los pasos realizados previamente, es posible acceder a la página web ingresando la dirección IP pública del router en un navegador web. Al momento de la escritura de esta sección, esta dirección es `190.161.13.15`. Esta forma de ingresar a la página web, escribiendo la dirección IP, funciona correctamente, sin embargo, es muy tediosa, pues hay que recordar un número muy largo. Además, un usuario promedio que quiera ingresar a la página web no entenderá nada de que significan esos números, e incluso puede pensar que está ingresando a un sitio peligroso. Por otro lado, la dirección IP del router cambia de forma períodica cada cierto tiempo debido al ISP, el cual modifica sin previo aviso la dirección IP de los clientes,

por lo que en unos días más, la dirección IP del router de la casa del alumno será totalmente distinta y ya no se podrá acceder a la página web desde Internet.

Para solventar este problema, se debe utilizar un servicio de DNS dinámico (DDNS). Este servicio se encarga de mantener el contacto con el router aunque cambie la dirección IP del router cuando el ISP la modifique. Asimismo, el DDNS es capaz asignar un dominio web a una IP dinámica, de tal manera que ya no sea necesario ingresar el número de la IP, sino que ahora la página web tendrá un dominio fijo que no cambiará, aunque se modifique la dirección IP del router. Esto facilita enormemente la visita de usuarios a la página web, pues nadie quiere estar recordando números para ingresar a un sitio web.

El servicio de DDNS que se utiliza en esta memoria, corresponde a NoIP, el cual ofrece DDNS gratuito para un sitio web. Para usar este servicio, se debe crear una cuenta en NoIP y configurar la URL del sitio web, el cual se eligió como `grafico-sensores.ddns.net`. Nótese que el dominio `.ddns.net` indica que el sitio web utiliza DDNS para ser accesible desde Internet. Lamentablemente, este dominio no puede ser cambiado por otro en el servicio gratuito de NoIP.

Por el lado del router es necesario hacer ciertas configuraciones para que el DDNS funcione, las cuales se detallan en el anexo C.

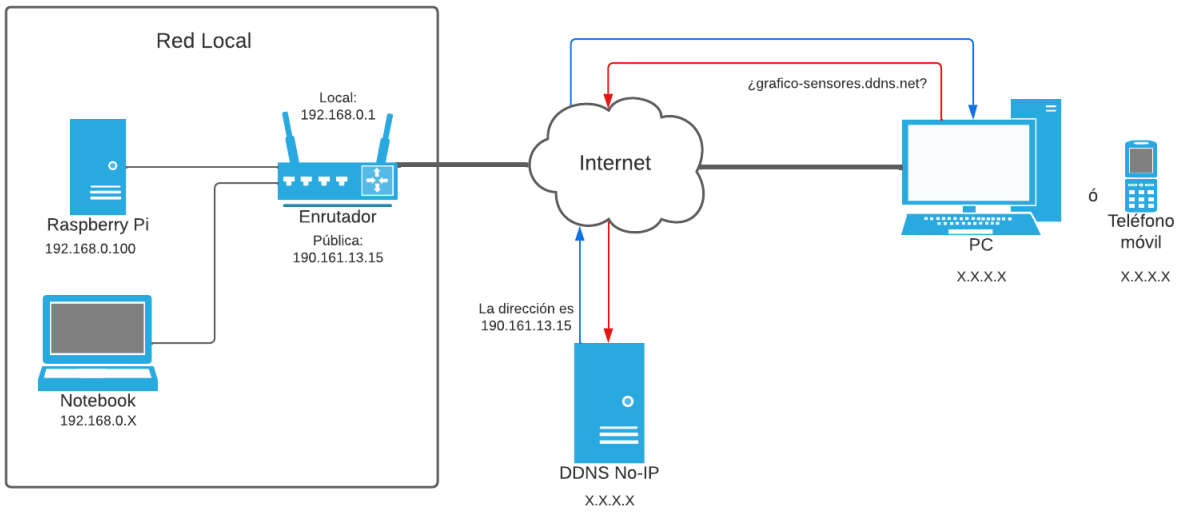


Figura 5.3: Diagrama de red del proyecto.

La figura 5.3 muestra el diagrama de red del proyecto. En primer lugar, se puede ver que en la red local 192.168.0.0 /24 aparecen el router, un notebook y la Raspberry Pi. La Raspberry y el router tienen direcciones fijas, pero la dirección del notebook cambia dependiendo del protocolo DHCP. Luego, al salir de la red local, se entra a Internet donde se ve que está un PC cualquiera por un lado y, por otro, el servidor DDNS de No-IP. Cuando el PC quiere acceder al sitio web `grafico-sensores.ddns.net`, le solicita la dirección del sitio al servidor de No-IP, y este responde con la dirección pública del router, la cual es actualmente `190.161.13.15`. Con esta dirección el PC puede conectarse con el router y, dado que todas las peticiones HTTP se

entregan a la Raspberry Pi, el PC se comunica directamente con la dirección 192.168.0.100 de la red local y, así, accede al sitio web.

# Capítulo 6

# Resultados y Análisis

## 6.1. PCB

En primer lugar, se presenta la placa construida. La figura 6.1 muestra el módulo central visto desde la parte superior ya con todos los componentes soldados: los dos conectores DB9, el ADS1015, los dos header de 5 pines y las resistencias. Asimismo, se ve que la Raspberry Pi ya está insertada en el socket de 20x2 y que los separadores de color oro que la ensamblan a la placa ya están atornillados.

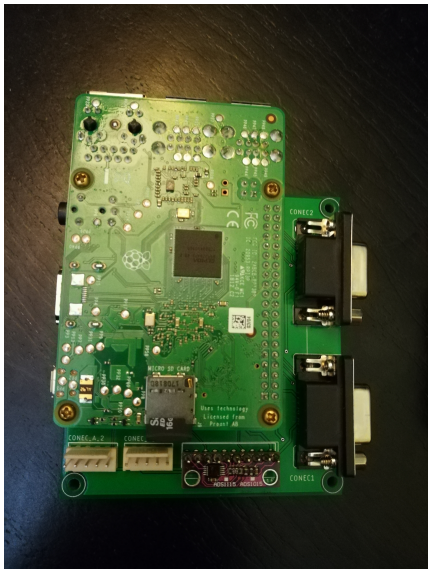

Figura 6.1: Lado superior de la placa PCB del módulo central.

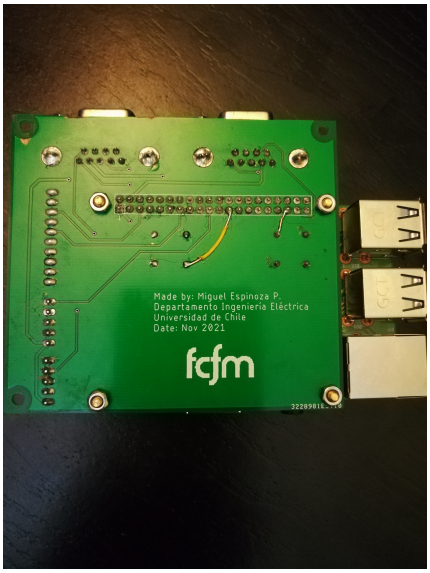


Figura 6.2: Lado inferior de la placa PCB del módulo central.

En la figura 6.2 se muestra la placa del módulo central en su sección inferior, en donde se pueden ver las pistas que van por debajo de la placa y la etiqueta que se escribió junto al logo de la Facultad. También se puede ver que hay dos cables soldados a mano, que conectan pines de la Raspberry Pi con elementos de la placa. La razón de esto, es que cuando se mando a construir la placa, faltó rutear esas dos pistas a los pines indicados. Se había realizado una modificación respecto al diseño original, y se olvidó rehacer esas dos conexiones eléctricas en el esquemático. Lamentablemente, después al rehacer las conexiones, se olvidó conectar

esos dos cables a los pines correspondientes, por lo que el programa no los consideró al hacer las rutas en el PCB. La funcionalidad de esos pines era de indicar el estado de conexión de las interfaces, es decir, señalaban si había una interfaz conectada a los conectores DB9, por lo que era absolutamente necesario el unirlos mediante un cable, puesto que gran parte del código dependía de eso.

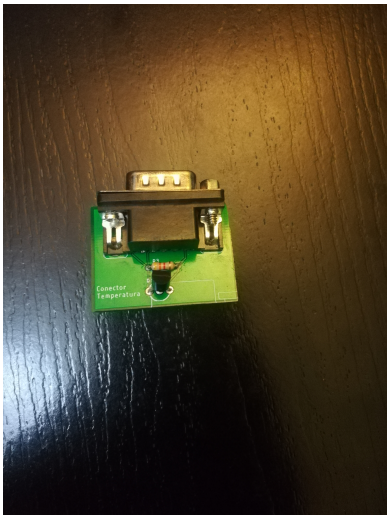


Figura 6.3: PCB de interfaz del sensor de temperatura.


Figura 6.4: PCB de interfaz del sensor de distancia.

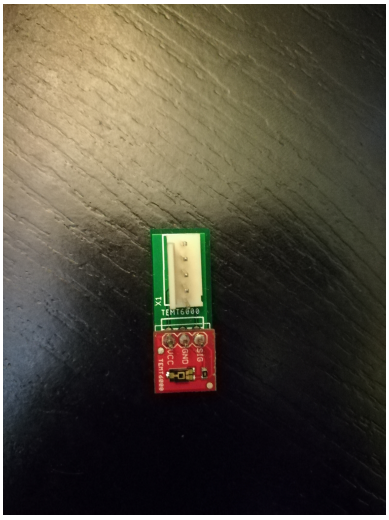


Figura 6.5: PCB de interfaz del sensor de luz.

En la figura 6.3 se muestra la interfaz del sensor de temperatura. Como se observa, cuenta con el sensor de temperatura DS18B20, una resistencia de pull-up para el protocolo 1-Wire, y un conector DB9 macho. La figura 6.4 muestra la interfaz del sensor de distancia, la cual cuenta con el sensor de distancia HC-SR04, dos resistencias que conforman un divisor de voltaje de 5V a 3.3V y un conector DB9 macho. Finalmente, la figura 6.5 presenta la interfaz del sensor de luz. La placa sólo tiene como componentes al sensor de luz TEMT6000 y al conector de cinco pines.

Por último, la figura 6.6, presenta al módulo central con todas las interfaces acopladas mediante los conectores. Los sensores digitales se conectan mediante los conectores DB9 y los sensores analógicos mediante un cable gris que une ambos conectores Molex. El resultado final en un sistema bastante firme, en el que las interfaces se pueden conectar de forma sencilla a los conectores, pero que no se salen fácilmente de ellos. Por lo tanto, se puede mover el sistema mientras está funcionando sin miedo a que se caigan las interfaces. Por otro lado, como se explicó en el Capítulo 4, el código está hecho para soportar el desacople de las interfaces sin sufrir ningún error. Esto otorga una estabilidad tanto por el lado de software como por el lado del hardware.

### 6.1.1. Problema con sensor de luz y ADC

Es importante mencionar que hubo un problema con el sensor de luz TEMT6000 y el conversor análogo digital ADS1015 en la placa, que ya fue solucionado, pero que en la versión preliminar de este informe no había sido resuelto. El problema que ocurría era que la

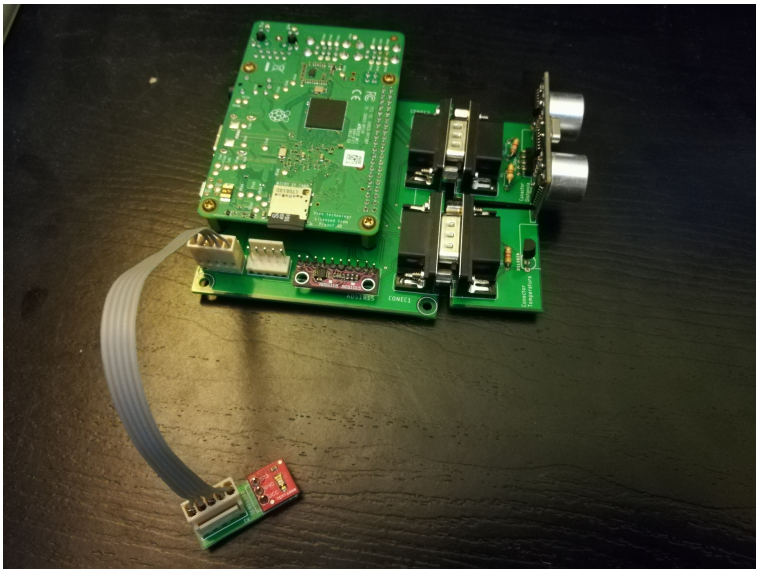

Figura 6.6: Placa PCB del sistema con todas las interfaces acopladas.

Raspberry Pi no reconocía el ADC a través de I2C, por lo que no podía comunicarse con este equipo.

Para intentar resolver este problema primero se probaron todas las conexiones de elementos, para buscar si había una mala conexión que impidiera la lectura del ADC. Se encontró que, efectivamente, había una mala conexión, pues había un cortocircuito entre algunos pines del ADS1015, por lo que el equipo se había quemado. Por lo tanto, la única solución era quitar el ADC malo y cambiarlo por uno nuevo.

Así que se extrajo el ADC defectuoso de la placa y, en su lugar, se soldó un socket de 10x1 a la placa; y por otro lado, el ADS1015 se soldó a un pin header de 10x1. De tal manera que para conectar el nuevo ADC a la placa simplemente era necesario insertar el pin header en el socket. Al probar la lectura de I2C con el nuevo ADC, se encontró que nuevamente la Raspberry Pi no podía comunicarse con este. Por lo tanto, ya no era un problema del ADC, si no que era algo más profundo.

Posteriormente, se intentó ver si la Raspbery Pi podía detectar el ADC al estar conectados ambos en una protoboard, sin la placa de por medio. Así que se extrajeron ambos componentes de la placa y se insertaron en una protoboard, manteniendo las mismas conexiones de antes, pero ahora con cables en vez de rutas de cobre. El resultado de esta prueba fue que la Raspberry Pi si pudo comunicarse con el ADC a través de I2C, e incluso leer datos del sensor de luz. Este resultado indicó que no era un problema de software, ya que se pensó que el código para conectarse con el ADC estaba malo.

Por lo que, se volvió a conectar los equipos a la placa y se midió la conectividad eléctrica. Se encontró que entre los conectores analógicos y el ADC, la conectividad no era consistente, sino que a ratos funcionaba y en otros momentos no, por lo que se volvieron a soldar estos componentes. Luego, al probar la comunicación I2C, se encontró que finalmente si era posible detectar el ADC y obtener las mediciones de luz, por lo que simplemente era un problema

de soldadura.

## 6.2. Página web

En la figura 6.7 se muestra la página web que se implementó. En la parte superior aparece un panel que indica los sensores conectados actualmente a las interfaces y el número de datos mostrados en el gráfico. Se señala si hay algún sensor conectado a la interfaz y, si es así, se especifica cúal es. Debido a la conexión plug and play de los sensores, estos pueden ser conectados y desconectados en cualquier momento, por lo que este panel se actualiza de forma periódica al visitar la página web.

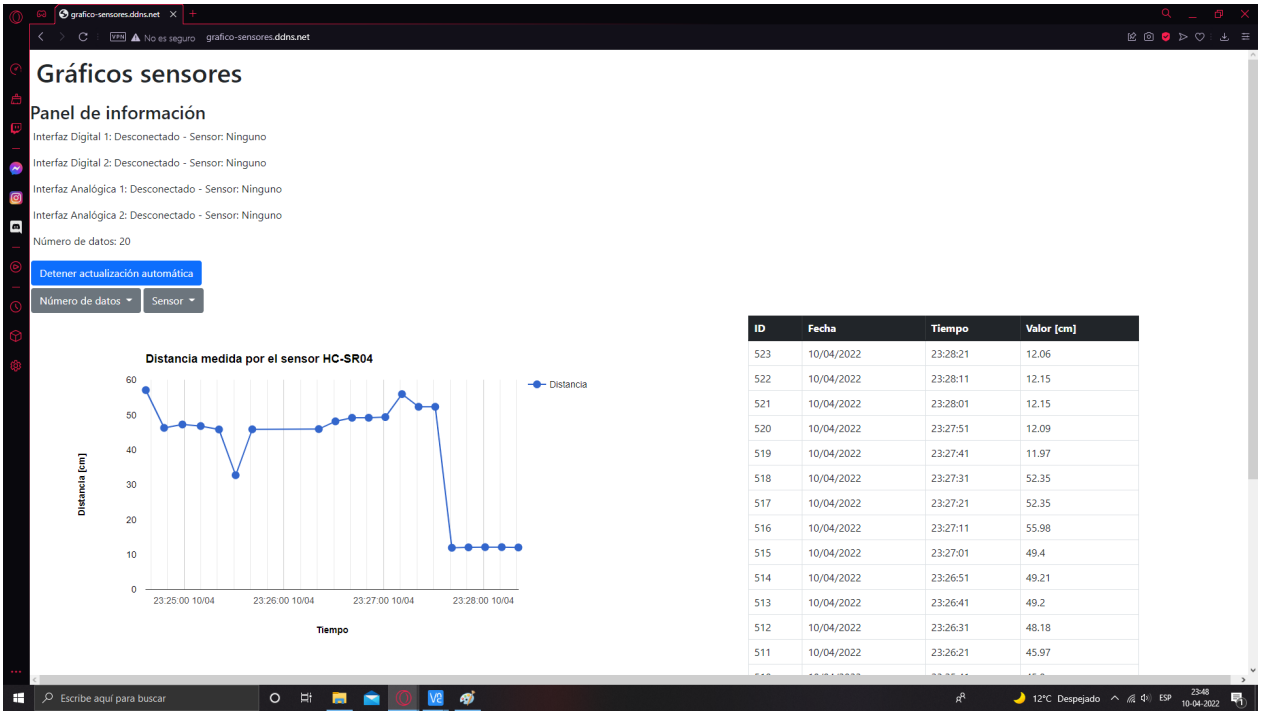


Figura 6.7: Se muestra la página web construida en el navegador web Opera de PC.

La página muestra un gráfico con las últimas mediciones del sensor seleccionado. El eje X del gráfico corresponde al tiempo y el eje Y al valor de la medición. Una tabla presenta los datos de manera ordenada del más nuevo al más antiguo. La página se actualiza de manera automática cuando hay nuevos datos, pero el botón de color azul permite detener esta actualización automática. Existen dos menús: el primero, para seleccionar el sensor que se quiere visualizar y, el segundo, el número de datos a ser mostrados por el gráfico y la tabla. En el anexo se muestran capturas de pantalla de la página web cuando se selecciona el otro sensor y se cambia el número de datos a visualizar.

Como se puede ver en la figura 6.8, la parte inferior de la página web aparece la información de la página web para dar contexto del trabajo. Junto a la información se agrega una imagen del logo del Departamento de Ingeniería Eléctrica para enfatizar que esta página web es parte del trabajo de un alumno del Departamento.

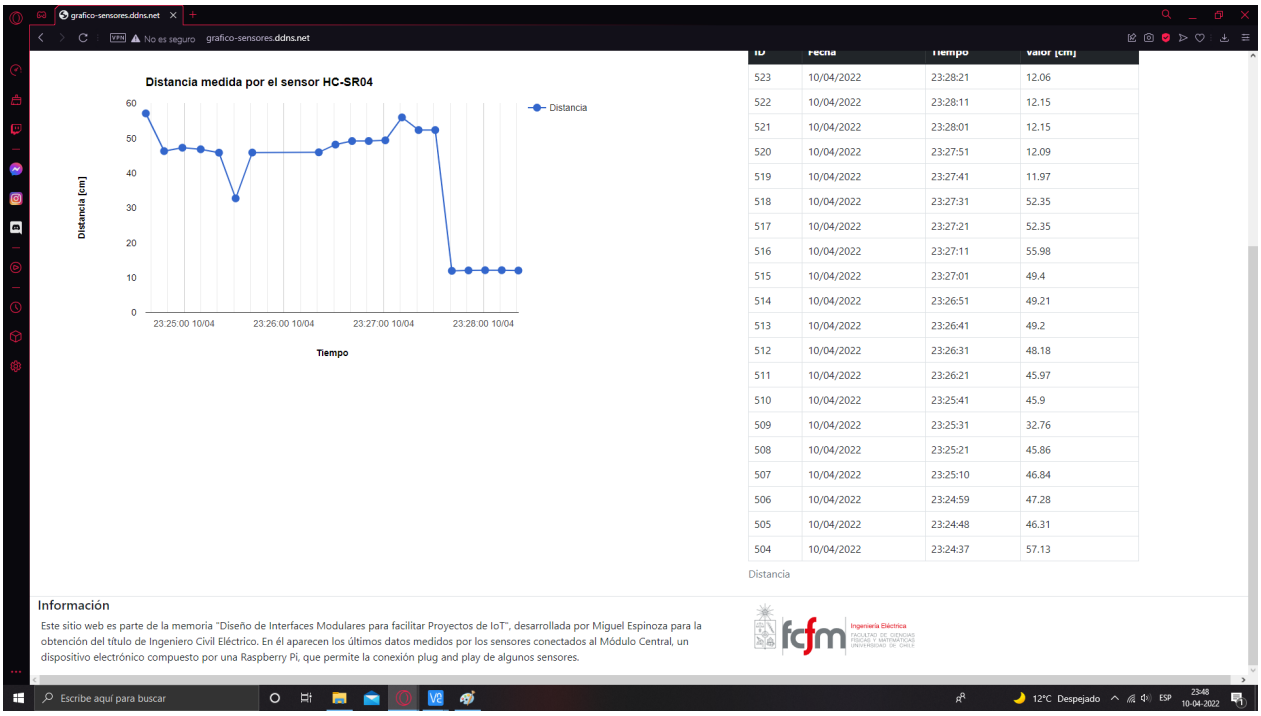


Figura 6.8: Sección inferior de la página web construida. Se observa el resto de los datos de la tabla y un *footer* con la información de la página web.

Además, se presenta la página web vista desde un celular, para mostrar que todos sus elementos son adaptables a tamaños de pantalla más pequeños. Debido a que el celular cuenta con un ancho menor, la página queda más larga, por lo que fue necesario tomar tres capturas de pantalla para lograr mostrar las secciones más importantes de la página: el panel de información en la figura 6.9; el gráfico y la tabla en la figura 6.10; y el texto de información junto al logo de la Facultad en la figura 6.11.

## 6.3. Costo económico del proyecto

Dado que uno de los objetivos específicos de este trabajo de título es realizar el proyecto con el menor costo económico posible, se detalla en la tabla A.1 cada uno de los componentes utilizados en el desarrollo del proyecto y su costo. Los componentes fueron comprados en las tiendas de electrónica MCIElectronics, AFEL y Victronics.

De la tabla se puede ver que el costo total de los componentes, es de $72.800, siendo el mayor costo debido a la Raspberry Pi, la cual tiene un valor de $49.990. Es decir, la Raspberry Pi representa cerca de un 70 % del costo del proyecto. Afortunadamente, este componente fue prestado por el Departamento de Ingeniería Eléctrica, por lo que no tuvo costo para el alumno. Considerando el costo total, se ve que el proyecto no es barato, pensando en que para una persona común y corriente de este país que gana $500.000 mensuales o menos, el equipo constituye al menos un 14 % de su sueldo mensual. Esto se debe a que la Raspberry Pi es demasiado cara, lo cual es normal considerando las capacidades que tiene, puesto que


Figura 6.9: Vista del panel en un celular.

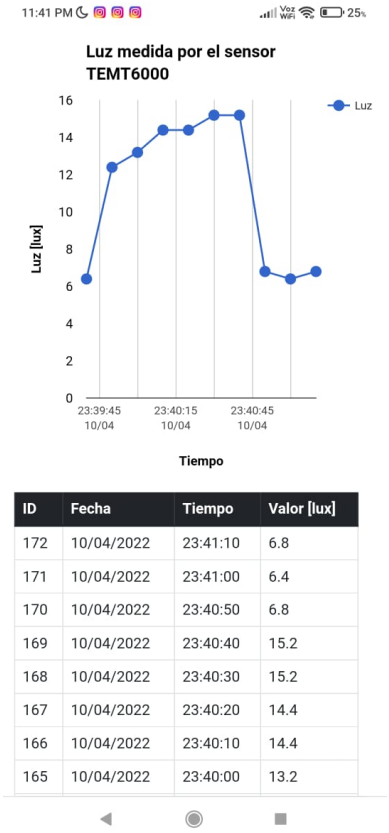


Figura 6.10: Gráfico y tabla vistos desde un celular.

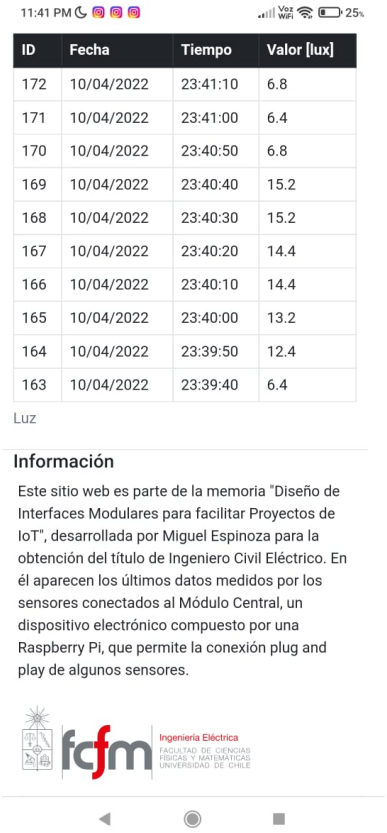


Figura 6.11: Vista del footer desde un celular.

es un minicomputador. Sin contar la Raspberry Pi, el costo de los demás materiales es de $22.810; este valor es mucho más asequible para desarrollar el proyecto a gran escala.

Cabe considerar que a pesar de el proyecto sea un poco caro, la funcionalidad que ofrece es difícil de lograr con otro dispositivo, puesto que la Raspberry Pi maneja el software, el hardware y la página web sin necesidad de ningún otro dispositivo ni servidor web externo. Si se hubiera realizado el proyecto con un Arduino o con un microcontrolador, hubiese sido más barato, pero probablemente hubiese sido necesario considerar el costo de utilizar un servidor externo para la página web. Esto sin contar si es que realmente hubiese sido posible implementar las interfaces *plug and play* en otro microcontrolador, tanto por el lado de software como el hardware. Por otro lado, los pines de la Raspberry Pi se utilizan en casi su totalidad, de los 40 pines que tiene, 27 están siendo usados. Por lo tanto, no es como que su capacidad de hardware esté siendo desaprovechada. Considerando lo anterior, aunque el costo sea un poco alto, se ve difícil implementar el proyecto en una plataforma más barata.

### 6.3.1. Comparación con alternativas

Con el fin de saber si el proyecto es caro o barato, se compará su costo con un sistema similar hecho con los equipos WisBlock de RAK Wireless [45]. Para desarrollar un proyecto similar con WisBlock, principalmente se deberían comprar los siguientes componentes:

| Componente | Cantidad | Precio Unitario | Valor |
|---|---|---|---|
| Raspberry Pi 3 Modelo B+ | 1 | $49.990 | $49.990 |
| Sensor de temperatura One Wire DS18B20 | 1 | $4.990 | $4.990 |
| Sensor de proximidad de ultrasonido HC-SR04 | 1 | $1.290 | $1.290 |
| Sensor de luz ambiental TEMT6000 | 1 | $3.290 | $3.290 |
| Convertidor Análogo Digital ADS1015 | 1 | $3.500 | $3.500 |
| Separador Niquelado M3x10MM (4u) | 1 | $890 | $890 |
| Conector DB9P Macho 90º para PCB | 2 | $750 | $1.500 |
| Conector DB9S Hembra 90º para PCB | 2 | $750 | $1.500 |
| Pin Header 40X1 Fila 2.54mm | 1 | $250 | $250 |
| Socket PCB 20x2 Paso 2.54mm Alto 9mm TH | 1 | $380 | $380 |
| Terminal Crimp AWG28-22 para Header 0.1" (50u) | 1 | $1.090 | $1.090 |
| Header PCB 5P 0.1" Recto (10u) | 1 | $610 | $610 |
| Housing 5T 0.1" Cable (10u) | 1 | $370 | $370 |
| PCB | 1 | $3.150 | $3.150 |
| **Total** | | | $72.800 |

Tabla 6.1: Lista de componentes y equipos comprados.

- WisBlock RAK19007 Base Board 2nd Gen: es un equipo que forma la placa base del sistema IoT, energiza los demás componentes y permite programar el software. Precio: $11.90 USD.
- WisBlock Core RAK11200: es el centro de procesamiento. Cuenta con un microprocesador ESP32 que permite la conexión WiFi. Precio: $13.90 USD.
- WisBlock Sensor RAK12007: es un sensor ultrasónico de distancia. Precio: $4.00 USD.
- WisBlock Sensor RAK12010: sensor de luz ambiental. Precio: $5.00 USD.
- WisBlock Sensor RAK1901: sensor temperatura y humedad. Precio: $4.60 USD.

El costo total de todos estos equipos es de $39.40 USD, lo cual es equivalente a $32.155 pesos chilenos. Ahora, hay que tomar en cuenta que RAKWireless es una empresa de EEUU, por lo que se debe calcular el costo del envío a Santiago. Según la página de RAKWireless, el costo del envío a Santiago es de $36.99 USD, que equivale a $30.193 pesos chilenos. Así, el costo total del producto es de $62.348.

Comparado con el costo del proyecto de la memoria, que fue de $72.800, se observa que el equipo de RAKWireless es levemente más barato, cuesta cerca de $10.000 menos. E incluso,

cuesta aún menos el equipo en sí, ya que prácticamente la mitad del costo se debe al envío a Santiago.

Sin embargo, el sistema construido en la memoria tiene puntos fuertes con los que no cuenta el equipo de WisBlock: en primer lugar, cuenta con página web accesible desde Internet, con WisBlock, el usuario deberá crear un sitio web y pagar por un servidor. En segundo lugar, el proyecto construido es realmente *plug and play*, ya que el usuario no debe saber nada de electrónica, en cambio, en WisBlock el usuario de igual manera debe saber programar y compilar código [55]. Como último punto, la Raspberry Pi cuenta con una memoria mucho más amplia y un procesador mucho más rápido, que el microcontrolador ESP32, lo cual le da un mayor almacenamiento de datos y más velocidad.

## 6.4. Funcionamiento del equipo

Para hacer funcionar el equipo se debe energizar la Raspberry Pi, y luego acceder al sitio web `grafico-sensores.ddns.net`. Si el servidor está activo, se muestran los últimos datos del sensor de temperatura, de forma similar a como aparece en la figura 6.7. Si la página web no es accesible, es porque el servidor aún no se inicia.

Una vez que la página web está activa, es posible conectar las interfaces de los sensores a los conectores. Para ello simplemente se debe insertar una interfaz con un conector macho en alguno de los conectores hembra que posee el módulo central. Cabe mencionar que los conectores DB9 son para los conectores digitales, y los conectores de cinco pines son para los sensores analógicos. Alrededor de cinco segundos después, como máximo, se verá en el panel de la página web que hay una interfaz conectada y, también, aparecerá a cuál sensor pertenece. Cerca de 10 segundos después de la conexión el sensor empezará a efectuar las mediciones y las enviará a la base de datos, por lo que aparecerán en la página web y se actualizará el gráfico y la tabla.

Para cambiar el sensor que se quiere visualizar, se debe seleccionar en el menú `Sensor`. Si se desea cambiar el número de datos mostrados, se debe seleccionar en el menú `Número de datos`.

El usuario puede sacar o insertar una interfaz en cualquier momento, el sistema es robusto ante esto. También puede conectar cualquiera de las dos interfaces digitales a cualquiera de los dos conectores digitales; no son específicas para cada sensor. Y, se puede conectar la interfaz analógica a cualquiera de los dos conectores analógicos.

Lamentablemente, la página web sólo es accesible desde Internet cuando la Raspberry Pi está dentro de la red local de la casa del alumno, ya que sólo se configuró el router de su casa. El resto del sistema funciona de forma correcta en otros lugares pero no es visible desde Internet, solo se puede ver el funcionamiento del sistema conectando la Raspberry Pi a un monitor a través de un puerto HDMI y abriendo la terminal, pero en ese caso sólo se verá el registro de ejecución, no el gráfico ni la tabla. Sin embargo, si se conecta la Raspberry Pi a una red WiFi o Ethernet, otros computadores o equipos que estén dentro de la misma red, podrán acceder a la página web ingresando la dirección IP de la Raspberry Pi en la red local. Esto permitiría probar el sistema en la Facultad, ya que los usuarios que estén dentro de la red WiFi de la Facultad podrán acceder al sitio web y, así, visualizar los datos en el gráfico

y en la tabla.

# Capítulo 7

# Conclusiones

En esta memoria se desarrollaron interfaces modulares capaces de conectarse de forma *plug and play* a un módulo central. El módulo central está compuesto por una placa en la que hay una Raspberry Pi, dos conectores DB9 hembra, dos conectores Molex de cinco pines macho y un ADC. Los conectores DB9 se utilizan para las sensores digitales y los conectores de cinco pines para los sensores analógicos. Por otro lado, las interfaces se componen de un sensor, resistencias y conectores DB9 macho, en el caso de los sensores digitales, y conectores de cinco pines hembra, en el caso de los sensores analógicos. De tal manera, que las interfaces se acoplan a los conectores del módulo central de forma sencilla. Los sensores que se seleccionaron para el proyecto fueron los sensores digitales: DS18B20 y HC-SR04, de temperatura y distancia, respectivamente; y el sensor de luz analógico TEMT6000. Estos sensores se eligieron debido a que eran fáciles de conseguir en el mercado de electrónica chileno y eran sencillos de utilizar. Las interfaces se diseñaron de tal manera que fueran independientes del tipo de sensor, por lo que un sensor digital se puede conectar a cualquiera de los dos conectores digitales que tiene el módulo central, y una interfaz analógica se puede conectar a cualquiera de los dos conectores analógicos.

Se logró implementar un software capaz de detectar el momento en que se conecta una interfaz a los conectores, e identificar cual sensor es el que está conectado, por esta razón las interfaces son independientes del tipo de sensor. Junto a ello, el programa puede leer las mediciones de los sensores y enviarlas a una base de datos, en donde son almacenadas.

Se implementó una página web que muestra un panel de información en donde aparecen los sensores digitales y analógicos conectados al módulo central. Junto a ello, se muestra un gráfico y una tabla con los últimos datos del sensor seleccionado. El usuario puede elegir que sensor visualizar a través de menús y botones de la página web. La página web actualiza de forma periódica el panel de información y el gráfico para mostrar los últimos cambios. Es capaz de mostrar si se conecta o desconecta una interfaz en tiempo real, y también puede actualizar el gráfico si detecta nuevos datos.

El sistema en su totalidad es fácil de usar, puesto que el usuario simplemente debe conectar la Raspberry Pi a la fuente de poder, y luego acceder a la página web. Dentro de la página web aparecerá el gráfico con los últimos datos medidos. Ahora el usuario puede conectar cualquier interfaz a un conector, y podrá ver como se actualiza el panel, informando que hay un sensor conectado. Posteriormente, la tabla y el gráfico se actualizaran con un nuevo da-

to. Se resalta que el usuario puede conectar o desconectar en cualquier momento las interfaces.

El costo económico del proyecto fue cerca de $70.000, siendo elevado principalmente por la Raspberry Pi, dado que su costo es de $49.990. Al compararlo con la alternativa dada por WisBlock que se encuentra en el mercado [45], se encuentra que la otra solución tiene un valor cercano a los $62.000, considerando el envío a Santiago. Por lo que el sistema construido es más caro. Sin embargo, la solución de WisBlock no cuenta con la página web, ni tampoco es totalmente *plug and play*, pues el usuario debe saber de programación en Arduino. Por lo tanto, se puede decir que a pesar que el sistema construido sea más caro que las alternativas, cuenta con más capacidades y facilita la vida para el usuario promedio.

Como trabajo a futuro, se pueden implementar interfaces para más sensores. Sin embargo, para ello se debe rediseñar el método de identificación que tienen los sensores, pues actualmente, el sistema sólo puede identificar dos sensores diferentes, puesto que sólo se usa un bit para guardar las id de los sensores, ya que este trabajo era una prueba de concepto. Se debe crear un sistema que permita identificar muchos más sensores, un número bastante bueno sería tener 4 bits para la identificación, lo que permitiría distinguir entre 16 interfaces distintas. Cabe notar que si se cambia el método de identificación sería recomendable diseñar una nueva PCB, aunque es posible usar la misma añadiendo cables.

Una vez que se haya diseñado un nuevo método de identificación, se debe diseñar una interfaz para el nuevo sensor y construirla en una placa PCB. Luego, se debe implementar un script para leer las mediciones del nuevo sensor, mediante la creación de una nueva clase en el código. Posteriormente debe añadir una nueva tabla a la base de datos, para almacenar las mediciones del sensor. Finalmente, se debe cambiar ligeramente el diseño de la página web, para que el nuevo sensor aparezca en la selección del menú de sensores.

También, se puede quitar el servidor web y la base de datos de la Raspberry Pi, y traspasarlo a un servicio de servidores web profesionales. Para así lograr que la página web sea accesible desde Internet cuando la Raspberry Pi este en cualquier lugar, y no sólo cuando esta se encuentre en el hogar del alumno.

Por último, se pueden cambiar los conectores utilizados, en vez de usar el conector de cinco pines, se puede utilizar un conector de seis o siete pines, para tener mayor cantidad de señales; o un conector RJ-11, para así tener mayor rango de distancia. E incluso, se puede modificar el diseño para hacerlo inalámbrico, usando Bluetooth para comunicarse con las interfaces, y desarrollando las placas de las interfaces con baterías, para que no necesiten estar energizadas por la Raspberry Pi.

# Anexo A

# PCB

Una vez que la PCB estaba diseñada y revisada, se mandó a construir al fabricante JLCPCB y el envío tomó alrededor de 10 días. Mientras tanto se compraron los componentes electrónicos faltantes en la tienda de electrónica Victronics. En la tabla A.1 se detallan los componentes comprados.

El separador de 10mm se utiliza para separar la Raspberry Pi y la placa del módulo central. Se compraron conectores DB9 de 90º para soldarlos a la PCB y que queden con la entrada de forma horizontal. El pin header de 40x1 se compra pues el sensor de luz requiere un pin header de 3x1 para soldarse a la placa y en el de 40x1 se pueden extraer los que necesite el diseñador. El socket PCB de 20x2 se emplea para acoplar la Raspberry Pi a la placa. El terminal crimp se utiliza para encajar los cables grises a los conectores Molex hembra. Los Header 5P y los Housing 5T, son los conectores Molex macho y hembra, respectivamente.

| **Componente** | **Cantidad** | **Precio Unitario** | **Valor** |
|---|---|---|---|
| Separador Niquelado M3x10MM (4u) | 1 | $890 | $890 |
| Conector DB9P Macho 90º para PCB | 2 | $750 | $1.500 |
| Conector DB9S Hembra 90º para PCB | 2 | $750 | $1.500 |
| Pin Header 40X1 Fila 2.54mm | 1 | $250 | $250 |
| Socket PCB 20x2 Paso 2.54mm Alto 9mm | 1 | $380 | $380 |
| Terminal Crimp para Header 0.1” (50u) | 1 | $1.090 | $1.090 |
| Header PCB 5P 0.1” Recto (10u) | 1 | $610 | $610 |
| Housing 5T 0.1” Cable (10u) | 1 | $370 | $370 |
| Construcción y envío de PCB | 5 | $3.150 | $15.750 |
| **Total** | | | $22.340 |

Tabla A.1: Lista de componentes comprados.

Cuando la PCB llegó a Santiago, se visitó el Laboratorio de Electrónica para soldar los componentes, ya que ahí se encontraban los implementos necesarios para realizar la solda-

dura, como el cautín y el estaño. El trabajo de soldadura tomó alrededor de 3 visitas al Laboratorio, ya que en ocasiones los componentes no quedaban bien soldados y había que hacerlos de nuevo. De hecho, fue necesario extraer el conversor análogo digital original, y cambiarlo por uno nuevo, dado que debido a una mala conexión, se había quemado. Afortunadamente, este problema no afectó los demás componentes de la placa.

# Anexo B

# Software

## B.1. Implementación de conectores en el código

### B.1.1. Conectores Digitales

El diagrama de clases UML de los conectores digitales del paquete `conectores` aparece en la figura B.1. Como se puede ver, la clase `Conector` es la más compleja del paquete. Cada objeto del tipo `Conector` cuenta con los siguientes campos:

- `Name`: representa el nombre del conector. Los valores que puede tomar en el código actual son `Conector1` y `Conector2`, haciendo referencia a los 2 conectores del módulo central.
- `Signals`: son objetos de la clase Pin, que indican los pines en las que se encuentran las señales GPIO del conector.
- `Id`: indica el pin por donde se recibe la ID.
- `State`: guarda el pin desde el que se detecta el estado del conector.
- `Isconnected`: es un valor *booleano* que indica si hay una interfaz acoplada al conector.
- `Sensor_connected`: es un objeto de tipo Sensor que registra el sensor conectado actualmente al conector.

Los métodos de la clase `Conector` permiten acceder a cada uno de los campos de la clase, y cambiar el estado de conexión del conector, conectándolo o desconectándolo . Además, gracias a que el conector guarda el sensor que tiene conectado en el campo `sensor_connected`, es posible acceder a los métodos del sensor de forma indirecta.

La clase `Pin` se creó para poder identificar un pin en los 2 modos de numeración posibles que tiene la Raspberry Pi:

1. Numeración física: el número del pin viene dado por la posición que tiene este en la placa, de izquierda a derecha y de arriba a abajo.
2. Numeración Broadcom: esta numeración de los pines GPIO está basada en el chip Broadcom de la Raspberry Pi, por lo que es una numeración interna que no tiene relación con la posición física de los pines. También se le llama numeración GPIO.

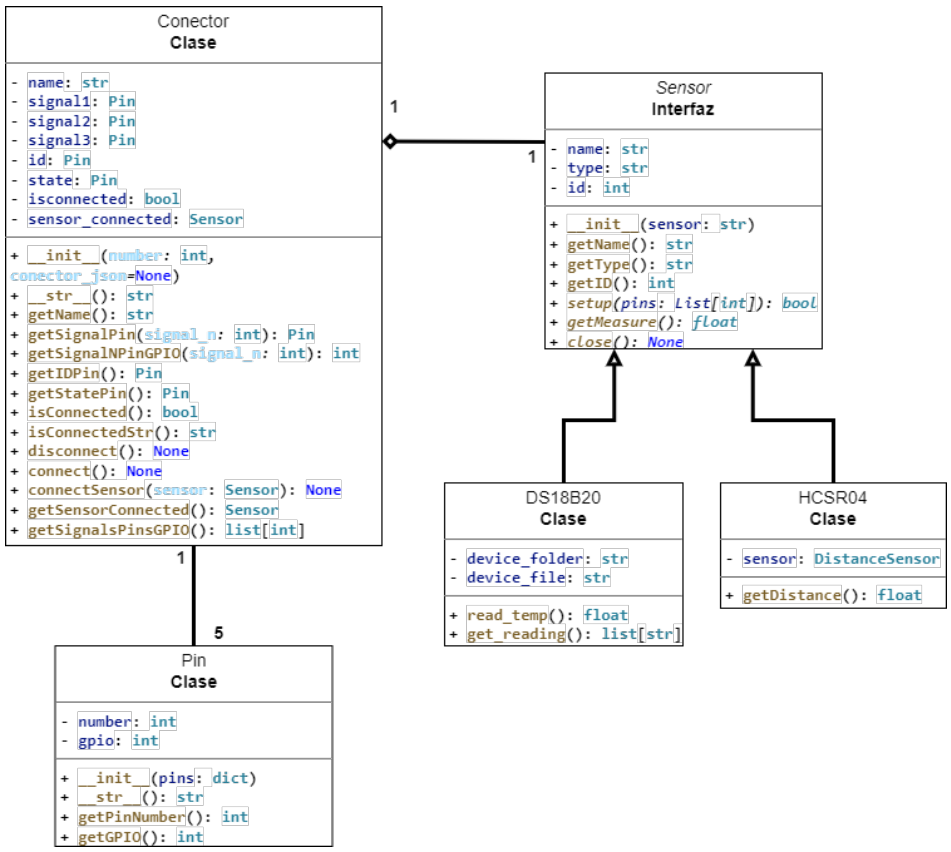


Figura B.1: Diagrama de clases UML de la sección digital del proyecto.

La clase `Pin` guarda la numeración física en el campo `number` y la numeración Broadcom en el campo `gpio`. Además, esta clase contiene métodos que permiten acceder a estos valores de forma sencilla.

La interfaz `Sensor` se encarga de proporcionar campos y métodos comunes a los sensores digitales que se utilizan en el proyecto y a futuros sensores digitales que se deseen añadir al proyecto. Esta interfaz define el comportamiento general de todos los sensores. Sus campos son `name`, `type` y `id`, que indican, respectivamente, el nombre del sensor, su tipo y su id para identificarlo. Al igual que el resto de clases, cuenta con métodos para acceder a sus campos, y también con métodos abstractos que deben implementar sus clases derivadas. Los métodos abstractos son:

- `setup`: se encarga de configurar el sensor para dejarlo listo para funcionar. Inicializa todos los campos que sean necesarios de los sensores y fija los pines a los que está conectado el sensor.
- `getMeasure`: el propósito de este método es obtener el valor de la medición del sensor. Cada sensor puede obtener la medición de la forma más apropiada para sí, pero todos deben entregar el valor a través de este método.
- `close`: este método se utiliza cuando el sensor se desconecta del conector y se encarga de cerrar todas las conexiones que se hayan hecho, limpiar los valores de los pines y eliminar los objetos que se hayan creado.

La primera clase derivada de la interfaz `Sensor` es la clase `DS18B2O`, que representa al sensor de temperatura DS18B20. La segunda es la clase `HCSR04`, que simboliza al sensor de distancia HC-SR04. Estas 2 clases cuentan con los campos y métodos necesarios para llevar a cabo las funciones específicas de cada sensor y entregar el valor de sus mediciones.

### B.1.2. Conectores Analógicos

En la figura B.2 se presenta el diagrama de clases UML de los conectores analógicos. Como se puede ver es bastante similar al diagrama UML de los conectores digitales, por lo que sólo se explicarán los nuevos métodos y campos.

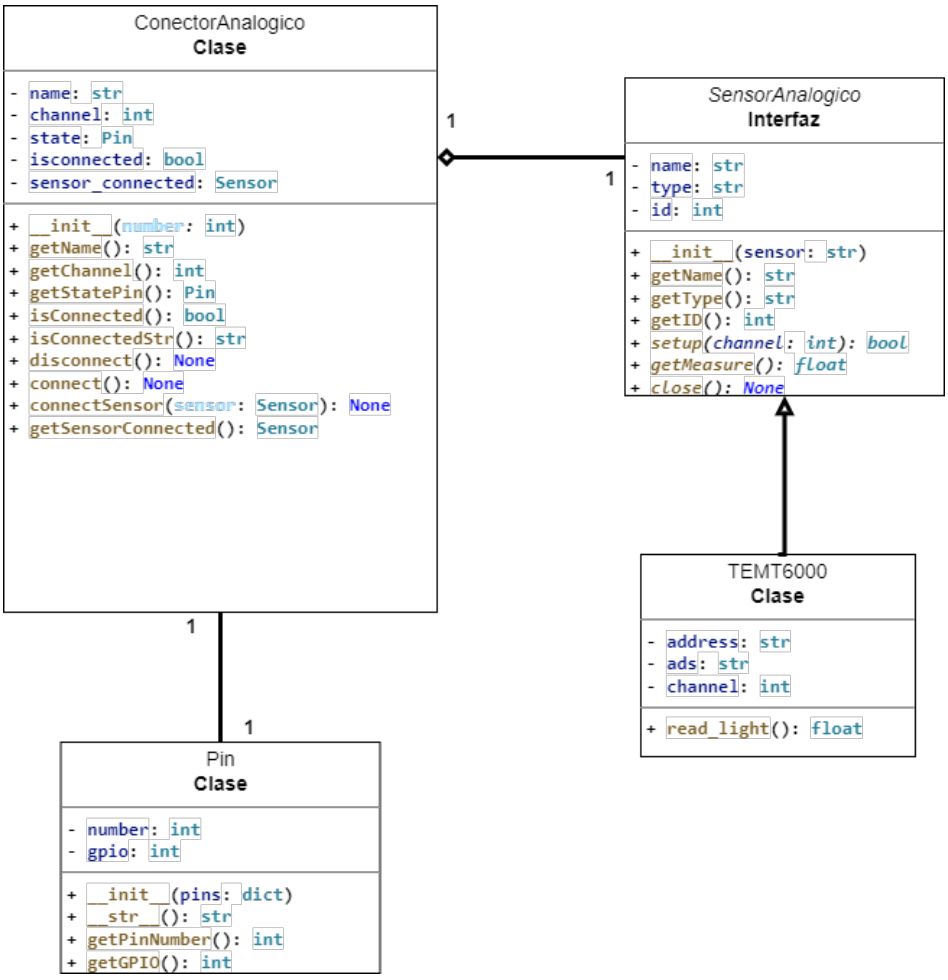


Figura B.2: Diagrama de clases UML de la sección analógica del proyecto.

La clase `ConectorAnalogico` añade el campo `channel`, que indica el canal del ADC al cuál está conectado este conector. Por otro lado, la interfaz `SensorAnalogico` es prácticamente igual a la interfaz `Sensor`, sólo que el método `setup` ahora tiene otro tipo de argumento. La única clase derivada de `SensorAnalogico`, es `TEMT6000`, la cual representa al sensor de luz y cuenta con los métodos y campos necesarios para la lectura de la iluminancia.

### B.1.3. Detección de interfaces

La detección de interfaces se lleva a cabo a través del pin `state` que poseen ambos conectores. Como se explicó en el capítulo anterior, estos pines están conectados a resistencias *pull down* y, cuando se cierra el circuito, al conectar una interfaz, su voltaje pasa a ser ALTO. Por lo que detectando desde el código un cambio del valor del pin STATE, de BAJO a ALTO, se sabrá cuando se conecte una interfaz.

Para la manipulación y lectura de los pines de la Raspberry Pi, es esencial utilizar la librería *gpiozero* [53]. Esta librería permite asociar a un pines un objeto llamado `button` (botón). Un `button` representa un dispositivo de entrada que puede tomar dos valores de voltaje: ALTO y BAJO; de forma equivalente a un botón que está conectado a una resistencia pull up o pull down. La librería gpiozero puede detectar si el pin tiene un voltaje ALTO, mediante la propiedad `is_pressed` del *botón* asociado al pin.

Debido a que una interfaz puede ser conectada o desconectada en cualquier momento por el usuario y sustituida por otra, el código debe ser robusto ante estos cambios. Para ello, se definen 4 estados en los que puede estar el programa:

1. **Ningún conector tiene conectada una interfaz**: en este caso, ambos conectores estarán esperando que se conecte una interfaz para detectarla. Por lo tanto, los conectores deberán calcular de forma periódica el valor de la propiedad `is_pressed` de los `buttons`, para saber si el pin pasó de BAJO a ALTO. Si se detecta que alguno pasó a ALTO, se llama a la función `sensor_connected_to_conector`, que identifica el sensor conectado al conector. Esta función, además, pone el conector en estado `conectado` y añade el sensor al conector.

2. **Sólo el conector 1 tiene conectada una interfaz**: aquí, el programa debe estar pendiente de cuando se desconecta la interfaz del conector 1 y cuando se conecta una interfaz al conector 2. Para detectar cuando se desconecta una interfaz, se debe detectar el cambio de voltaje del pín `state` de ALTO a BAJO. Cuando ocurre esto, se llama a una función que se encarga de desconectar la interfaz del conector y de aplicar el método `close` en el sensor conectado y quitarlo del conector.

3. **Sólo el conector 2 tiene conectada una interfaz**: es equivalente al caso 2, sólo que al revés.

4. **Ambos conectores tienen conectada una interfaz**: en este caso, ambos conectores deben estar detectando cuando se desconecte una interfaz.

Es importante mencionar que se chequea el estado de los pines `state` cada 1 s y luego se pone el código en estado de suspensión hasta que pase otro segundo.

### B.1.4. Identificación del sensor

La función `sensor_connected_to_conector` identifica el sensor leyendo el valor del pin `id`. El sensor de temperatura DS18B20 tiene el ID 1, lo que quiere decir que cuando está conectado, el pin `id` debe tener un voltaje ALTO. Por otro lado, el sensor de distancia HC-SR04 tiene el ID 0, por lo que cuando está conectado, el valor del pin `id` debe ser BAJO.

Además, la función ejecuta otras tareas como, conectar una interfaz al conector y configurar el sensor que se ha conectado. Es importante leer el valor del pin `id` sólo cuando el pin `state` ha cambiado a ALTO, puesto que el pin `id` no posee una resistencia de pull up o pull down, por lo que su voltaje cuando no hay ningún sensor conectado es ambiguo.

En cuanto a los conectores digitales, sólo está el sensor TEMT6000, por lo que no es necesario identificarlo. Basta con configurarlo y empezar a leer las mediciones.

### B.1.5. Lectura de mediciones

La lectura de mediciones se realiza llamando al método `getMeasure` de cada sensor. Debido a que los sensores tienen funcionamiento distinto, cada uno implementa el método `getMeasure` de forma diferente, por lo que se explicará su implementación para cada sensor por separado.

#### B.1.5.1. Sensor DS18B20

El sensor de temperatura se comunica a la Raspberry Pi a través del protocolo 1-Wire. Este protocolo lo que hace es almacenar las mediciones del sensor en las subcarpetas de un directorio llamado `/sys/bus/w1/devices/`. La subcarpeta en particular en el que se guardan los datos depende del modelo y la id del sensor de temperatura. Por lo tanto, en la configuración del sensor, lo primero que se hace es ver si la subcarpeta está creada; puesto que así se sabe que el sensor está correctamente conectado y que se está comunicando con la Raspberry. Si no está conectado, no se podrá encontrar la subcarpeta.

Cuando ya se tiene el archivo con las mediciones de temperatura, se lee la última medición y se entrega cuando se llama al método `getMeasure`.

#### B.1.5.2. Sensor HC-SR04

En el caso del sensor HC-SR04, la medición de distancia es más sencilla gracias a la librería *gpiozero*, pues esta misma cuenta con una clase llamada `DistanceSensor`. Esta clase ya tiene toda los cálculos para entregar la distancia, por lo que simplemente creando un objeto de esta clase y llamando a su campo `distance` se puede obtener el valor de la distancia. Además, en la misma construcción del objeto `DistanceSensor` se pueden fijar los pines que utilizará el sensor.

#### B.1.5.3. Sensor TEMT6000

En cuanto al sensor de luz TEMT6000, es necesario usar la librería `adafruit-circuitpython-ads1x15` de Adafruit, que facilita la conexión y transmisión de datos con el conversor analógico digital ADS1015 [56]. Gracias a esta librería se puede crear un bus I2C con el que la Raspberry Pi se comunique con el ADC. Luego, seleccionando el canal del ADC se puede acceder al voltaje de salida del sensor luz, y realizando las operaciones indicadas en el Capítulo 3, se obtiene el valor de la iluminancia.

### B.1.6. Acceso a la base de datos

Como se mencionó en el Marco Teórico, la base de datos se realizó con MariaDB. Este programa no tiene una interfaz gráfica, por lo que se maneja solamente a través de comandos, los que pueden ser escritos por medio de la Terminal de Linux o la PowerShell de Windows. Por lo tanto, mediante comandos se crea una base de datos llamada `sensores` y, en ella, 3 tablas llamadas: `temperatura`, `distancia` y `luz`. Cada una de las tablas cuenta con 3 columnas: el id de la medición, el tiempo, y el valor de la medición.

Algunos de los comandos más utilizados son `SELECT` e `INSERT`. El comando `SELECT` permite extraer elementos de una tabla de la base de datos y el comando `INSERT` permite insertar datos en una tabla.

En la figura B.3 se puede ver la base de datos MariaDB. La interfaz es simplemente la terminal de Linux, ya que como se mencionó previamente, no cuenta con una interfaz gráfica como tal. En la figura se puede ver el acceso a MariaDB mediante el comando `mysql`, junto al usuario `root` y la contraseña. Posteriormente, se ingresa a la base de datos `sensores` y, finalmente, se obtienen los últimos 10 datos de la tabla `temperatura`, por medio del comando `SELECT`.

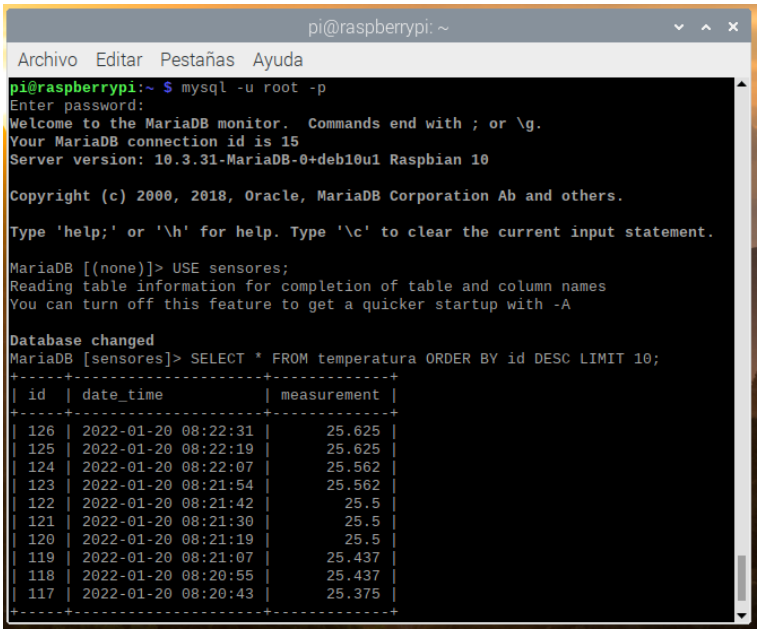


Figura B.3: Interfaz de MariaDB en la consola. Se observan los últimos 10 datos de la tabla *temperatura*.

En el código el acceso a la base de datos se realiza en el paquete `database`, el cual tiene sólo un archivo, `connection_database`. Este archivo realiza las tareas de conectarse a la base de datos, extraer datos ya almacenados y añadir nuevos datos. Es posible realizar estas tareas en Python gracias a la librería MariaDB Connector, la cual permite conectarse a la base de datos ingresando un usuario y su contraseña. Para obtener mediciones de la base de datos se utiliza un comando similar al que aparece en la figura B.3.

Es importante mencionar que se envían mediciones a la base de datos cada 10 segundos, con el fin de no sobrecargar la base de datos con nuevos datos a cada instante y, además, es poco probable que ocurra un cambio brusco en las mediciones de un segundo a otro.

## B.2. Archivos de configuración

En esta sección se presentan, en primer lugar, los archivos de configuración del sistema operativo que fueron reescritos durante la ejecución del proyecto. Posteriormente, se detalla la creación del servicio de `systemd` llamado `conectors.service`, que se mencionó en el Capítulo 4.

Los archivos de configuración que se modificaron son los siguientes:

- `config.txt`: este archivo guarda la configuración para el arranque de la Raspberry Pi. En él se activó la interfaz I2C y se cambiaron algunos parámetros de HDMI para dar una mejor resolución en el monitor.
- `dhcpd.conf`: guarda la configuración del protocolo DHCP de la Raspberry Pi. Se modificó para darle a la Raspberry Pi una dirección IP estática. En el Capítulo 5 aparece con mayor detalle la modificación realizada.
- `w1.conf`: contiene opciones para la captura de datos a través del protocolo 1-Wire. En particular, se cambiaron los valores de los parámetros `timeout` y `slave_ttl`. Estos parámetros indican el tiempo que debe esperar la Raspberry cuando pierde contacto con el protocolo 1-Wire antes de cerrar la comunicación. Previamente, el valor de `timeout` era de 10 s, por lo que la Raspberry seguía intentando conectar con el sensor durante mucho tiempo. Ahora ese valor se cambió a sólo 1 s, lo cual reduce en gran manera el tiempo de espera.

Ahora, respecto a los servicios a los servicios de Unix, el primer paso en la creación de un servicio `systemd`, es crear de un archivo tipo `service` en la carpeta `/etc/systemd/system`. Un servicio `systemd` contiene 3 secciones principales:

- La sección `[Unit]` especifica las características y descripción del servicio [57].
- La sección `[Service]` indica el usuario y el grupo que debe correr el servicio. Y también el directorio donde se encuentra la aplicación y el comando para ejecutar el servicio [58].
- La sección `[Install]` se utiliza para indicar a systemd para cuales usuarios se debe instalar este servicio.

El archivo `conectors.service` aparece en el código B.1. La mayoría de la configuración de este es fácilmente deducible por los nombres en inglés. Sin embargo, conviene profundizar en las opciones `Restart`, `RestartSec` y `After`. La primera indica que este servicio siempre debe reiniciarse después de que se haya caído por algún problema del código o del sistema operativo y, la segunda, especifica que el tiempo de reinicio debe ser a los 2 segundos después de fallar. Se colocó este número para evitar que el servicio se caiga muchas veces seguidas por un problema externo. Por otro lado, el significado de `After` indica el momento del arranque en el que se debe iniciar el servicio, que en este caso, `multi-user.target` significa que se inicie el servicio cuando el sistema ya este listo para aceptar el inicio de sesión de un usuario [59].

```
[Unit]
Description=Conectors service
After=multi-user.target

[Service]
Type=simple
User=pi

Restart=always
RestartSec=2
WorkingDirectory=/home/pi/Documents/Proyecto/
ExecStart=/usr/bin/python3
    ↪ /home/pi/Documents/Proyecto/conectores/read_conectors.py

[Install]
WantedBy=multi-user.target
```

Código B.1: Archivo conectors.service.

Una vez implementado el archivo del servicio, se debe iniciar escribiendo el comando `start` y y luego habilitar mediante el comando `enable` de `systemctl`:

```
sudo systemctl start conectors
sudo systemctl enable conectors
```

Código B.2: Inicialización del servicio creado.

Se realiza un proceso similar para escribir y activar el servicio `analog_conectors.service`, que se encarga de los conectores analógicos. Su implementación aparece en el código B.3.

```
[Unit]
Description=Analog conectors service
After=multi-user.target

[Service]
Type=simple
User=pi

Restart=always
RestartSec=2
WorkingDirectory=/home/pi/Documents/Proyecto/
ExecStart=/usr/bin/python3
    ↪ /home/pi/Documents/Proyecto/conectores/read_analog_conectors.py

[Install]
WantedBy=multi-user.target
```

Código B.3: Archivo analog_conectors.service.

# Anexo C

# Página web

## C.1. Back End

### C.1.1. Flask

En esta sección se mencionan y explican las rutas que se crearon en Flask para el funcionamiento de la página web:

- `/`: es la ruta principal, entrega la página HTML de entrada del sitio web.
- `/get_data/<string:sensor>`: esta ruta está vinculada a una función que obtiene el último dato del sensor seleccionado. El cual luego entrega al front end por medio de un diccionario con formato JSON.
- `/get_last_data/<string:sensor>/<int:last_n>`: esta ruta es similar a la anterior, pues está vinculada a una función que obtiene los últimos n datos del sensor seleccionado y, luego, los envía al front end.
- `/check_interfaces`: esta ruta está asociada a una función que se encarga de revisar el estado de los conectores digitales y las interfaces conectadas a ellos. Para tal fin, revisa el archivo de apoyo JSON que se construyó en el capítulo anterior. El cual contiene el estado actual del conector y el sensor que tiene conectado.
- `/check_interfaces_analog`: esta ruta realiza el mismo trabajo que la anterior, pero para los conectores analógicos.

### C.1.2. Gunicorn

El punto de entrada desde donde acceder a la aplicación de Flask se hace creando un archivo de Python llamado `wsgi.py` y escribiendo lo siguiente dentro de él:

```
1 from application import app
2
3 if __name__ == "__main__":
4     app.run()
```

Este archivo lo que hace es importar la aplicación creada desde Flask y correrla. Así, se convierte en un punto de entrada para toda la aplicación web.

El servicio `gunicorn_project.service` se muestra en el código C.1, y dado que la configuración de este servicio es distinto al de `conectors.service` detallado en el anexo B, se explicarán las opciones escritas.

- En este caso, el momento de arranque del servicio, dado por la opción `After`, es después de que estén iniciados los servicios de red.
- El grupo de trabajo es `www-data`, para que Nginx también se pueda comunicar con el servicio de Gunicorn.
- El comando de ejecución indica la dirección del ejecutable de Gunicorn. Luego, fija el número de procesos. Después, crea un socket de Unix para que se vincule con el archivo de servicio. Y, finalmente, señala el punto de entrada WSGI de la aplicación web.

```
1  [Unit]
2  Description=Gunicorn instance to serve memoria
3  After=network.target
4
5  [Service]
6  User=pi
7  Group=www-data
8  WorkingDirectory=/home/pi/Documents/Proyecto/webpage
9  ExecStart=/home/pi/.local/bin/gunicorn --workers 3 --bind
       ↪ unix:gunicorn_project.sock wsgi:app
10
11 [Install]
12 WantedBy=multi-user.target
```

Código C.1: Servicio de systemd que corre Gunicorn.

Para iniciar el servicio y activarlo, se escriben las siguientes líneas en la consola:

```
1 $ sudo systemctl start gunicorn_project
2 $ sudo systemctl enable gunicorn_project
```

Con esto queda configurado el servicio de Gunicorn que conecta a Flask y a Nginx.

### C.1.3. Nginx

Para la implementación del servidor Nginx, en primer lugar, se crea el archivo de configuración llamado `proyecto` en la carpeta `sites-available` de Nginx y se edita:

```
1 $ sudo nano /etc/nginx/sites-available/proyecto
```

Dentro de este archivo se debe configurar el servidor; indicando el puerto que debe escuchar, el nombre del servidor y la dirección del socket al que se conecta. El texto del archivo de configuración se expone en el código C.2. En este caso se escucha el puerto 80, que es el puerto de HTTP, y el socket es el creado previamente en la sección de Gunicorn. El nombre del servidor no es importante en este proyecto.

```
server {
  listen 80;
  server_name your_domain www.your_domain;

  location / {
    include proxy_params;
    proxy_pass http://unix:
      /home/pi/Documents/Proyecto/webpage/gunicorn_project.sock;
  }
}
```

Código C.2: Archivo de configuración del servidor de Nginx.

Luego, se enlaza este archivo a la carpeta `sites-enabled` de Nginx, para que quede dentro de los sitios web disponibles:

```
$ sudo ln -s /etc/nginx/sites-available/proyecto /etc/nginx/sites-enabled
```

Finalmente, para actualizar el sitio web, se debe reiniciar el servicio de Nginx para que lea la nueva configuración:

```
$ sudo systemctl restart nginx
```

## C.2. Front End

Las figuras C.1 y C.2 muestran la página al cambiar el sensor seleccionado y el número de datos del gráfico. Es posible ver que en ambos casos, al aumentar el número de datos ya no es posible ver toda la página web en la pantalla, pues la tabla crece de tamaño y desplaza al resto de elementos hacia abajo. Además, se observa que en la figura C.2 el panel se actualiza porque no hay ningún sensor conectado a las interfaces.

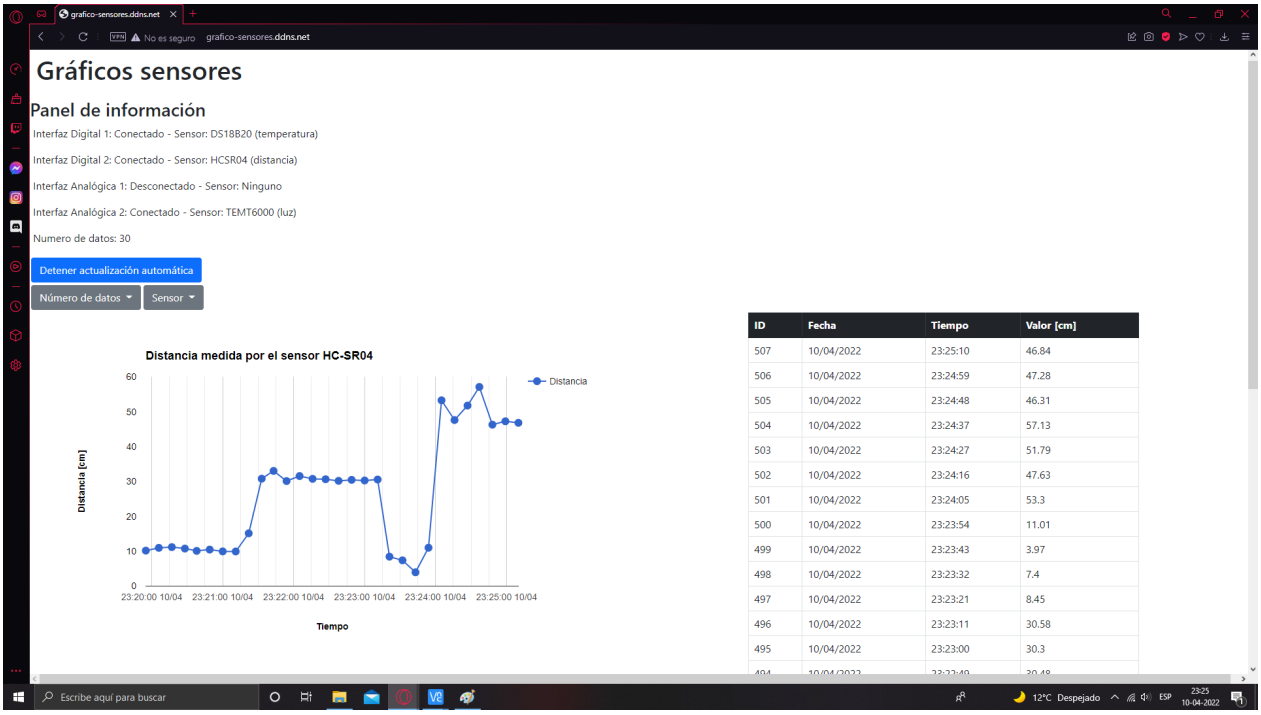


Figura C.1: Página web al mostrar 30 datos del sensor de distancia.

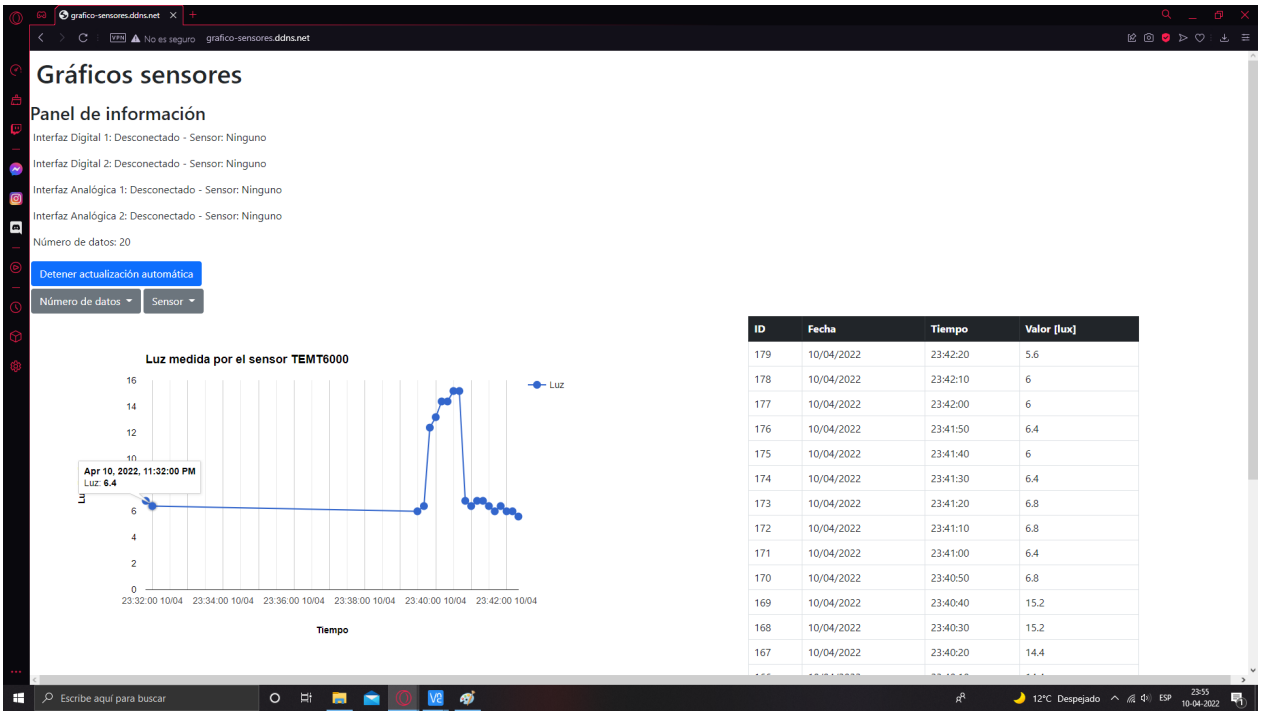


Figura C.2: Página web al mostrar 20 datos del sensor de luz. Además, se puede ver que el panel se actualiza cuando no hay ningún sensor conectado a las interfaces, sin embargo aún se pueden ver los datos medidos previamente.

## C.3. Acceso desde Internet

Para la fijación de la IP estática, primeramente, desde la Raspberry Pi se modifica el archivo de configuración de red `dhcpcd.conf`, que se encuentra en la carpeta `/etc/` del sistema operativo, las líneas que se agregaron se muestran en el código C.3.

```
# Example static IP configuration
interface wlan0
static ip_address=192.168.0.100/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

Código C.3: Nueva configuración del archivo dhcpcd.conf para fijar una IP estática.

La primera línea indica que se está trabajando con la interfaz `wlan`, que quiere decir que es la red local inalámbrica (Wireless LAN). Luego, con el segundo comando se fijó la dirección IP como 192.168.0.100. Se eligió esta IP porque era fácil de recordar y porque está fuera del rango de direcciones que se asignan a los otros equipos del hogar, las cuales se encuentran generalmente en el rango 192.168.0.3 y 192.168.0.10. El tercer comando fija la dirección del router y, el último, indica los DNS a los que debe acceder para encontrar una página web.


Figura C.3: Configuración del router para que la Raspberry Pi tenga una dirección IP fija.

En segundo lugar, es necesario configurar el router para que le dé siempre la misma dirección a la Raspberry Pi. Para realizarlo se ingresa al router desde el navegador web, colocando la dirección IP de este, es decir, 192.168.0.1, en la barra de direcciones. Una vez en la página es posible acceder al router ingresando el usuario y la contraseña. Posteriormente, se selecciona la pestaña `LAN`, y dentro de ella, el menú `Lista de clientes`. Al hacer esto, aparece la página de la figura C.3. Dentro de está página es posible agregar equipos a la lista de clientes con IP reservada, como ya está hecho con la Raspberry Pi.


Figura C.4: Configuración del router para que la Raspberry Pi se ubique en la DMZ.

Para crear la zona desmilitarizada, se debe acceder al router nuevamente y seleccionar la pestaña `Firewall` y luego el menú `DMZ`. Al hacerlo, la página se verá como la figura C.4. En esta página se puede escribir la IP privada que se va encontrar en la DMZ que, en este caso, sería la misma IP estática que se fijó para la Raspberry Pi en los pasos anteriores. Como detalle adicional, es necesario destacar que en la misma página se pueden ver los peligros de aplicar la DMZ a un equipo, por lo que sólo hay que hacerlo si se cuenta con toda la seguridad necesaria.

Como se explicó en el Capítulo 5, se debe configurar el router para activar el DDNS. Por lo que dentro del router se debe habilitar el DDNS y escribir la empresa que presta el servicio. Además, se debe escribir el usuario y contraseña de la cuenta de NoIP y, por último, el nombre del sitio web. En la figura C.5 aparece la configuración que se realizó en el router. Con los pasos realizados, ahora se puede acceder a la Raspberry Pi, simplemente escribiendo `grafico-sensores.ddns.net` en el buscador del navegador web. Este nombre, claramente, es mucho más fácil de recordar que el número de la dirección IP.

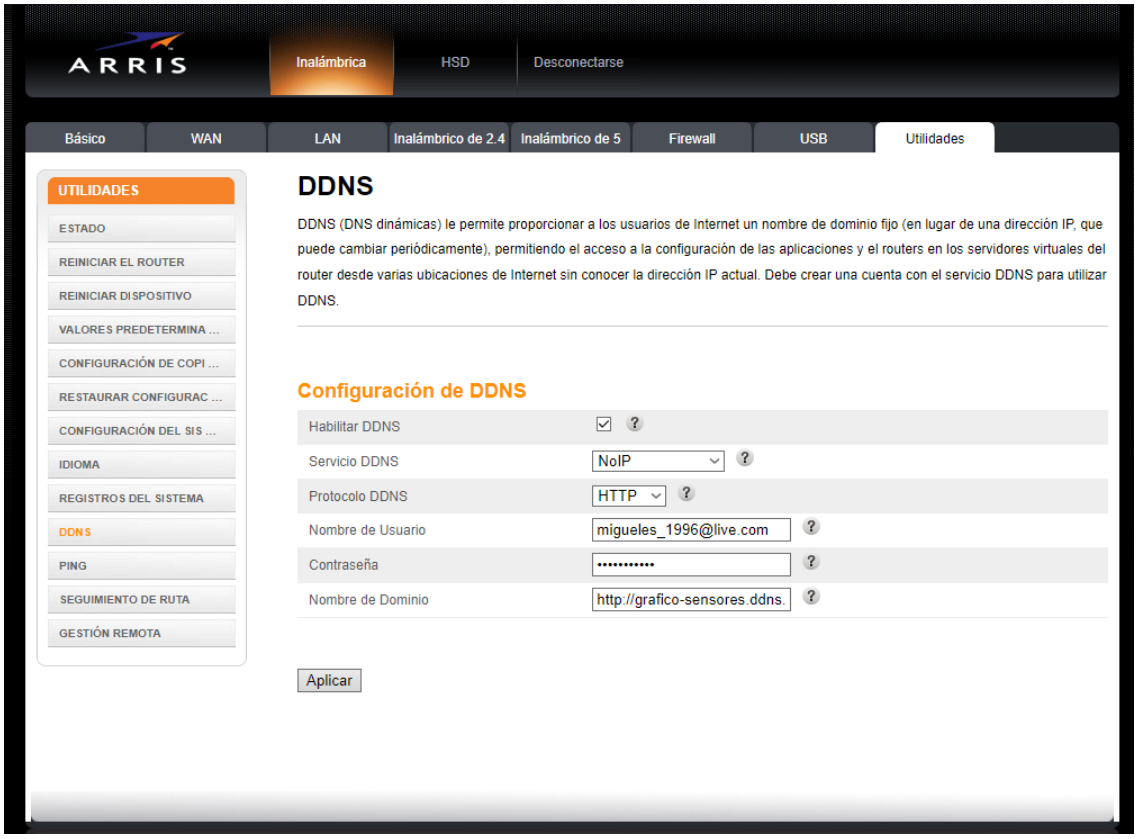


Figura C.5: Configuración del DDNS en el router.

# Bibliografía


[1] Max Roser, H. R. y Ortiz-Ospina, E., "Internet," Our World in Data, 2015. https://ourworldindata.org/internet.

[2] Fortino, G. y Trunfio, P., Internet of things based on smart objects: Technology, middleware and applications. 2014, doi:10.1007/978-3-319-00491-4.

[3] Sethi, P. y Sarangi, S. R., "Internet of Things: Architectures, Protocols, and Applications," Journal of Electrical and Computer Engineering, vol. 2017, 2017, doi:10.1155/2017/9324035.

[4] Milenkovic, M., Internet of Things: Concepts and System Design. Springer International Publishing, 2020, doi:10.1007/978-3-030-41346-0_1.

[5] Börner, K., Scrivner, O., Cross, L. E., Gallant, M., Ma, S., Martin, A. S., Record, L., Yang, H., y Dilger, J. M., "Mapping the co-evolution of artificial intelligence, robotics, and the internet of things over 20 years (1998-2017)," PLOS ONE, vol. 15, pp. 1–21, 2020, doi:10.1371/journal.pone.0242984.

[6] Rayes, A. y Salam, S., Internet of Things From Hype to Reality: The Road to Digitization. Springer Publishing Company, Incorporated, 2nd ed., 2018.

[7] "Tendencias de búsqueda en Google del término IoT." https://trends.google.es/trends/explore?date=all&q=iot, 2021. [Online] Accedido: 15-Mayo-2021.

[8] Coursera, "Introduction to the Internet of Things and Embedded Systems." https://es.coursera.org/learn/iot?specialization=iot, 2022. [Online] Accedido: 03-Abril-2022.

[9] Marques, G., Garcia, N., y Pombo, N., A Survey on IoT: Architectures, Elements, Applications, QoS, Platforms and Security Concepts, vol. 22. 2017, doi:10.1007/978-3-319-45145-9_5.

[10] Halfacree, G., The Official Raspberry Pi Beginner's Guide: How to use your new computer. Raspberry Pi Press, 2020.

[11] The Wikimedia Foundation, "Raspberry Pi." https://es.wikipedia.org/wiki/Raspberry_Pi. [Online] Accedido: 25-Octubre-2021.

[12] Watkiss, S., Learn Electronics with Raspberry Pi. APress, 2020.

[13] Pinout.xyz, "Raspberry Pi Pinout." https://pinout.xyz. [Online] Accedido: 25-Octubre-2021.

[14] Harrold, C., Practical Smart Device Design and Construction. APress, 2020.

[15] Raimond Spekking, "Printed Circuit Board." https://commons.wikimedia.org/wiki/File:SEG_DVD_430_-_Printed_circuit_board-4276.jpg. [Online] Accedido: 25-Octubre-2021.

[16] Jens Lienig, J. S., Fundamentals of Layout Design for Electronic Circuits. Springer, 2020.

[17] Christian Taube, "THT." https://commons.wikimedia.org/wiki/File:MOS6581_chtaube061229.jpg. [Online] Accedido: 25-Octubre-2021.

[18] MDN contributors, "An overview of HTTP." https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview, 2021. [Online] Accedido: 23-Diciembre-2021.

[19] MDN contributors, "HTTP request methods." https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods, 2021. [Online] Accedido: 23-Diciembre-2021.

[20] Collins, M., Pro HTML5 with CSS, JavaScript, and multimedia: complete website development and best practices. Apress, 2017.

[21] Bootstrap team, "Build fast, responsive sites with Bootstrap." https://getbootstrap.com, 2021. [Online] Accedido: 24-Diciembre-2021.

[22] MDN contributors, "What is JavaScript?." https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript, 2021. [Online] Accedido: 23-Diciembre-2021.

[23] OpenJS Foundation, "What is jQuery?." https://jquery.com, 2021. [Online] Accedido: 24-Diciembre-2021.

[24] Google, "Google Charts: Display live data on your site." https://developers.google.com/chart, 2021. [Online] Accedido: 24-Diciembre-2021.

[25] S. Simic, "What is LAMP Stack?." https://phoenixnap.com/kb/what-is-a-lamp-stack, 2019. [Online] Accedido: 26-Diciembre-2021.

[26] Oracle, "What is Linux?." https://www.oracle.com/linux/what-is-linux/, 2021. [Online] Accedido: 26-Diciembre-2021.

[27] NGINX, "What is NGINX?." https://www.nginx.com/resources/glossary/nginx/, 2021. [Online] Accedido: 26-Diciembre-2021.

[28] Raspberry Pi FR, "Install Nginx Raspbian, and accelerate your Raspberry web server." https://howtoraspberrypi.com/install-nginx-raspbian-and-accelerate-your-raspberry-web-server/, 2018. [Online] Accedido: 26-Diciembre-2021.

[29] T. Jankov, "MariaDB vs MySQL, a Database Technologies Rundown." https://kinsta.com/blog/mariadb-vs-mysql/, 2020. [Online] Accedido: 26-Diciembre-2021.

[30] Python Wiki, "Python Overview." https://wiki.python.org/moin/BeginnersGuide/Overview, 2019. [Online] Accedido: 26-Diciembre-2021.

[31] Vijay Singh, "Flask vs Django in 2022: Which Framework to Choose?." https://hackr.io/blog/flask-vs-django, 2022. [Online] Accedido: 03-Abril-2022.

[32] Platzi, "¿Flask o Django para desarrollo web con Python?." https://platzi.com/blog/flask-django-desarrollo-web-python/, 2022. [Online] Accedido: 03-Abril-2022.

[33] Flask Project, "Foreword." https://flask.palletsprojects.com/en/2.0.x/foreword/, 2021. [Online] Accedido: 26-Diciembre-2021.

[34] Wikipedia, "Web Server Gateway Interface." https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface, 2021. [Online] Accedido: 26-Diciembre-2021.

[35] Gunicorn, "Gunicorn." https://gunicorn.org, 2021. [Online] Accedido: 26-Diciembre-

2021.

[36] J. Ellingwood and K. Juell, "How To Serve Flask Applications with Gunicorn and Nginx on Ubuntu 18.04." https://www.digitalocean.com/community/tutorials/how-to-serve-flask-applications-with-gunicorn-and-nginx-on-ubuntu-18-04, 2021. [Online] Accedido: 26-Diciembre-2021.

[37] Omed Habib, "A Performance Analysis of Python WSGI Servers: Part 2." https://www.appdynamics.com/blog/engineering/a-performance-analysis-of-python-wsgi-servers-part-2/, 2022. [Online] Accedido: 03-Abril-2022.

[38] Song, E. Y. y Lee, K., "Understanding ieee 1451-networked smart transducer interface standard - what is a smart transducer?," IEEE Instrumentation Measurement Magazine, vol. 11, no. 2, pp. 11–17, 2008, doi:10.1109/MIM.2008.4483728.

[39] Potter, D., "Implementation of a plug and play sensor system using ieee p1451.4," en SIcon/01. Sensors for Industry Conference. Proceedings of the First ISA/IEEE. Sensors for Industry Conference (Cat. No.01EX459), pp. 162–166, 2001, doi:10.1109/SFICON.2001.968522.

[40] Rana, R., Bergmann, N., y Trevathan, J., "Towards plug-and-play functionality in low-cost sensor network," en 2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing, pp. 265–270, 2011, doi:10.1109/ISSNIP.2011.6146545.

[41] Ciancetta, F., Bucci, G., Fiorucci, E., y Landi, C., "A Plug-n-Play wireless sensor network based on Web service for monitoring climatic parameters," VECIMS 2010 - 2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems, Proceedings, pp. 72–76, 2010, doi:10.1109/VECIMS.2010.5609340.

[42] Yi, W.-Y., Leung, K.-S., Leung, Y., Meng, M.-L., y Mak, T., "Modular sensor system (mss) for urban air pollution monitoring," en 2016 IEEE SENSORS, pp. 1–3, 2016, doi:10.1109/ICSENS.2016.7808924.

[43] Mikhaylov, K. y Petajajarvi, J., "Design and implementation of the plug&play enabled flexible modular wireless sensor and actuator network platform," Asian Journal of Control, vol. 19, 2017, doi:10.1002/asjc.1492.

[44] Huang, C.-M., Hsieh, Y.-J., Lai, W.-L., Liu, Y.-J., Juan, C.-Y., Chen, S.-Y., Chen, C.-Y., Chue, J.-J., Yang, C.-C., y Wu, C.-M., "A modular wireless sensor platform and its applications," en 2017 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–4, 2017, doi:10.1109/ISCAS.2017.8050611.

[45] RAK Wireless, "WisBlock." https://store.rakwireless.com/pages/wisblock, 2021. [Online] Accedido: 22-Junio-2021.

[46] Upswift, "Device Management." https://www.upswift.io/platform/linux-iot-device-management, 2021. [Online] Accedido: 22-Junio-2021.

[47] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer." https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf, 2019. [Online] Accedido: 27-Octubre-2021.

[48] Pinout.xyz, "W1-GPIO - One-Wire Interface." https://pinout.xyz/pinout/1_wire. [Online] Accedido: 28-Octubre-2021.

[49] Elec Freaks, "Ultrasonic Ranging Module HC-SR04." https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf, 2019. [Online] Accedido: 27-Octubre-2021.

[50] Vishay Semiconductors, "TEMT6000 Ambient Light Sensor." https://www.sparkfun.com/datasheets/Sensors/Imaging/TEMT6000.pdf, 2004. [Online] Accedido: 27-Octubre-2021.

[51] Michael Bartlett, "TEMT6000 Ambient Light Sensor Hookup Guide." https://learn.sparkfun.com/tutorials/temt6000-ambient-light-sensor-hookup-guide, 2016. [Online] Accedido: 28-Octubre-2021.

[52] Texas Instruments, "ADS101x Ultra-Small, Low-Power, I2C-Compatible, 3.3-kSPS, 12-Bit ADCs With Internal Reference, Oscillator, and Programmable Comparator." https://www.ti.com/lit/ds/symlink/ads1015.pdf, 2018. [Online] Accedido: 28-Octubre-2021.

[53] B. Nuttall, "Gpiozero Documentation." https://gpiozero.readthedocs.io/en/stable/index.html, 2022. [Online] Accedido: 06-Enero-2022.

[54] Flask Project, "Routing." https://flask.palletsprojects.com/en/2.1.x/quickstart/#routing, 2022. [Online] Accedido: 09-Abril-2022.

[55] RAKwireless Technology, "Compiling a Project." https://docs.rakwireless.com/Product-Categories/WisBlock/RAK11200/Quickstart/#compiling-a-project, 2022. [Online] Accedido: 10-Abril-2022.

[56] Adafruit, "Python CircuitPython." https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/python-circuitpython, 2022. [Online] Accedido: 09-Abril-2022.

[57] FreeDesktop.org, "Manual page for systemd unit." https://www.freedesktop.org/software/systemd/man/systemd.unit.html, 2022. [Online] Accedido: 21-Febrero-2022.

[58] FreeDesktop.org, "Manual page for systemd service." https://www.freedesktop.org/software/systemd/man/systemd.service.html, 2022. [Online] Accedido: 21-Febrero-2022.

[59] Unix Stack Exchange, "Why do most systemd examples contain WantedBy=multi-user.target?." https://unix.stackexchange.com/questions/506347/why-do-most-systemd-examples-contain-wantedby-multi-user-target, 2022. [Online] Accedido: 21-Febrero-2022.

[60] Yang, S.-H., Principle of Wireless Sensor Networks, pp. 7–47. London: Springer London, 2014, doi:10.1007/978-1-4471-5505-8_2.

[61] Labiod, H., Afifi, H., y Santis, C. D., eds., BluetoothTM: Architecture and Functions, pp. 75–108. Dordrecht: Springer Netherlands, 2007, doi:10.1007/978-1-4020-5397-9_3.

[62] Gumudavelli, S., Gurkan, D., Hussain, S. A., y Wang, R., "A network management approach for implementing the smart sensor plug and play," en 2010 IEEE Sensors Applications Symposium (SAS), pp. 261–264, 2010, doi:10.1109/SAS.2010.5439396.

[63] Mikhaylov, K. y Huttunen, M., "Modular wireless sensor and actuator network nodes with plug-and-play module connection," en SENSORS, 2014 IEEE, pp. 470–473, 2014, doi:10.1109/ICSENS.2014.6985037.

[64] No-IP, "Remote Access with Dynamic DNS." https://www.noip.com/remote-access, 2021. [Online] Accedido: 28-Junio-2021.

[65] Texas Instruments, "LM35 Precision Centigrade Temperature Sensors." https://www.ti.com/lit/ds/symlink/lm35.pdf, 2017. [Online] Accedido: 30-Junio-2021.

[66] Microchip, "PIC18FXX2 Data Sheet." https://www.mcielectronics.cl/website_MCI/static/documents/PIC18F452.pdf, 2006. [Online] Accedido: 01-Julio-2021.

[67] John Fader, "SMT." https://commons.wikimedia.org/wiki/File:Smt_closeup.jpg. [Online] Accedido: 25-Octubre-2021.

[68] Google Sheets, "Python Quickstart." https://developers.google.com/sheets/api/quickstart/python, 2021. [Online] Accedido: 30-Octubre-2021.

[69] Mozilla, "Trabajando con JSON." https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON, 2021. [Online] Accedido: 30-Octubre-2021.

[70] Pygsheets, "Pygsheets." https://pygsheets.readthedocs.io/en/stable/, 2016. [Online] Accedido: 30-Octubre-2021.

[71] MDN contributors, "HTML: HyperText Markup Language." https://developer.mozilla.org/es/docs/Web/HTML, 2021. [Online] Accedido: 23-Diciembre-2021.

[72] Pinout.xyz, "Raspberry Pi Pinout." https://es.pinout.xyz, 2022. [Online] Accedido: 04-Enero-2022.