

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



**“Diseño e implementación de un dispositivo electrónico
como herramienta de apoyo a la educación escolar en
STEAM”**

Memoria de Titulación presentada por

Guillermo Hernán Becerra Ojeda

como requisito parcial para optar al título de

Ingeniero Civil Electrónico

Profesora Guía
Dra. María José Escobar
Profesor Correferente
Dr. Gonzalo Carvajal

Abril - 2023

TÍTULO DE LA MEMORIA:

Diseño e implementación de un dispositivo electrónico como herramienta de apoyo a la educación escolar en STEAM

AUTOR:

Guillermo Hernán Becerra Ojeda

TRABAJO DE MEMORIA, presentado en cumplimiento parcial de los requisitos para el título de Ingeniero Civil Electrónico de la Universidad Técnica Federico Santa María.

Dra. María José Escobar

Dr. Gonzalo Carvajal

Valparaíso, Abril de 2023.

a mi madre Sara

AGRADECIMIENTOS

Quiero iniciar este espacio agradeciendo a mis seres queridos, familia y amigos que han sabido acompañarme con su cariño, entusiasmo e incansable aliento para que diera cumplimiento y término formal a este proceso académico y profesional. A mi madre Sara, quien me diera todos los medios para entregarme por entero a recorrer el camino universitario, no sólo en lo académico, sino también en lo deportivo, político, cultural y en cuanto aportara a mi formación dentro y fuera del aula, por su incansable empuje y preocupación por mí. Agradezco a mi hija Isabel y su madre Carolina por vivir juntos el camino desde los primeros años universitarios y los primeros pasos de Isabel armonizando el crecimiento de todos. Pilares importantes para mí fueron mi abuela, Olga, mi docente por excelencia y a mis padrinos René y Marietta, con su apoyo emocional y su tiempo. Quiero cerrar este primer espacio reconociendo a Fernanda, ya que sin ella no habría sido posible dar este último paso; fue quien me acompañó con su amor, paciencia y entusiasmo, a pesar de vivir tiempos difíciles.

Agradezco especialmente a la profesora María José Escobar, por su guía, interés, paciencia y profusos conocimientos entregados, no sólo durante la guía de este trabajo de título, sino durante toda mi carrera universitaria; desde el primer peldaño en Introducción a la Ingeniería, hasta las asignaturas de postgrado, en las actividades de investigación, como su ayudante y todas esas conversaciones que aportaron a mi crecimiento profesional y personal. Es para mí un modelo a seguir, en cuanto destaca por su integridad humana, vocación de servicio, tanto pública como institucional, sumando a de toda su capacidad y vasto conocimiento.

Deseo expresar mi gratitud para con los siguientes profesores: Juan Yuz Eissmann, mi tutor, quien en diferentes espacios me enseñó a enfrentar problemas con actitud y responsabilidad; a Milan Derpich Musa, por sus excelentes y desafiantes clases, ejemplo del amor por el conocimiento y su transmisión; a Daniel Rodríguez Schulz, por su buscar siempre formas creativas de enseñar

electrónica y demostrar que el rigor y la alegría son compatibles; y a todos las y los profesores que desempeñan su labor con responsabilidad y cariño, por enseñar también fuera de la sala con su ejemplo, por el tiempo dedicado a escucharnos y compartir su guía genuinamente interesada en la formación profesional y humana de nosotros, sus estudiantes. Un reconocimiento adicional va para el profesor Gonzalo Carvajal, miembro de la comisión examinadora, por su real interés en el tema, sus valiosos aportes sin mayor pretensión que el sincero deseo de aportar a un mejor trabajo. Le agradezco también la voluntad presta a revisar y participar de los exámenes con diligencia.

Tengo que agradecer también a Diego Pandolfa, amigo y colega, por sus invaluable aportes en las discusiones sobre el diseño y posibilidades de implementación del dispositivo fruto de este trabajo. Hemos aprendido juntos en este y tantos otros proyectos.

No puedo dejar de reconocer y agradecer a las siguientes instituciones que contribuyeron directa o indirectamente a la realización de este trabajo. La fundación Fundación Gabriel & Mary Mustakis, por otorgarme la beca Aporte País que me permitió realizar una pasantía en investigación de la enseñanza de STEAM en la Universidad de Newcastle, Australia. A la Fundación Ecoscience, con quienes colaboramos por varios años en la enseñanza del STEAM en Chile y gracias a quienes se materializó este proyecto con la posibilidad de impactar realmente en la educación en Chile durante la pandemia. A la universidad por mantener las instalaciones impecables y hacer gestión para que tuviésemos un lugar con las condiciones para alcanzar el más alto grado de saber posible.

Finalmente, pero no menos importante, quiero destacar mi agradecimiento a mi compañeros de ruta y amigos, Boris Vidal, José Rojas, Hans Lehnert, Felipe Arriagada, Rodrigo González, Constanza Lorca y Alejandro Gajardo. Juntos vivimos las asumidas largas horas de estudio, pero también otras tantas alegrías y pesares, nos vimos crecer y así seguiremos haciendo en los años por venir. Vayan a ellos tanto mi gratitud como mis mejores deseos de éxito en la vida.

CONTENIDO

AGRADECIMIENTOS	I
RESUMEN	VII
ABSTRACT	IX
ACRÓNIMOS	XI
1. INTRODUCCIÓN	1
1.1. Contexto	1
1.2. Descripción del Problema	3
1.3. Objetivos y Alcance	5
2. PRELIMINARES	6
2.1. Educación STEAM	6
2.2. Internet of Things	8
2.3. Diseño de equipos electrónicos	9
2.4. Trabajos relacionados	10
2.4.1. Ámbito educativo	10
2.4.2. Productos similares	12
3. DEFINICIÓN DE REQUERIMIENTOS	15
3.1. Características Funcionales	15
3.1.1. Requerimientos Generales	15
3.1.2. Experiencias de aprendizaje pivotaes	16
3.1.3. Requerimientos Deseables	21
4. DISEÑO DE PCB	23
4.1. Etapas de Diseño en software CAD	23
4.2. Diseño por módulos	27
4.2.1. Módulos de Energía	27

4.2.2. Módulo del Microcontrolador	28
4.2.3. Módulo de Sensores	31
4.2.4. Módulo de Actuadores Embebidos	32
4.3. Diseño realizado	33
5. FABRICACIÓN Y MONTAJE	35
5.1. Adquisición de componentes mediante BOM	35
5.2. Orden de Fabricación de PCB	37
5.3. Montaje	39
5.3.1. Preparación	39
5.3.2. Métodos de montaje	41
5.4. Costos Directos del Producto	42
6. DESARROLLO DE FIRMWARE	44
6.1. Configuración del ambiente de desarrollo	44
6.2. Diseño del flujo principal	46
6.3. Diseño por Módulos	46
6.3.1. Setup	46
6.3.2. Loop	48
6.3.3. Reconnect	48
6.3.4. Callback	48
6.3.5. Beeping	48
6.3.6. Módulos de iniciación de hardware	49
6.4. Bibliotecas utilizadas	49
7. PRUEBAS EXPERIMENTALES	51
7.1. Pruebas eléctricas	51
7.2. Pruebas funcionales	52
7.3. Pruebas con Ubidots	56
7.3.1. Valores de medición reportados	56
7.3.2. Recibir valores de actuación	57
CONCLUSIONES Y TRABAJO FUTURO	59
7.4. Conclusiones	59
7.5. Trabajo futuro	60
A. ESQUEMÁTICOS DEL DISPOSITIVO	62
B. PLANOS DE FABRICACIÓN	72
C. ORDEN DE FABRICACIÓN	74

D. LISTADO DE COMPONENTES: BOM COMPRADO	77
D. CÓDIGO FIRMWARE	81
7.6. Librería IoTkit.h	81
7.7. Librería IoTkit.cpp	83
7.8. Código principal	88
REFERENCIAS	102
COPYRIGHT	105

RESUMEN

En el presente trabajo de título, se aborda el diseño e implementación de un dispositivo electrónico en hardware y firmware, como un producto mínimo viable de una herramienta docente para la enseñanza del Internet de las Cosas en el paradigma educativo STEAM.

El objetivo principal de este trabajo es disminuir dos barreras que dificultan la enseñanza y el aprendizaje del STEAM-IoT, una metodológica y una económica. Por la parte metodológica, se definen requerimientos de diseño para que el dispositivo facilite la introducción del tema a estudiantes de forma práctica. Por la parte económica, se define una restricción sobre costo directo unitario que puede tener el producto.

Para cumplir con este objetivo, se documenta la definición de requerimientos funcionales, desde las experiencias de aprendizaje sugeridas hasta las funcionalidades deseables del producto. Se lleva a cabo el diseño electrónico del dispositivo en software EDA, la adquisición de componentes y la orden de fabricación de las placas PCB en proveedores internacionales, el montaje manual de la placa, la programación del firmware y la realización de pruebas experimentales.

Como resultado de este trabajo, se logra un dispositivo que cumple con los requisitos de diseño y cuyo costo directo unitario asciende a cuarenta y cinco mil pesos, monto que representa menos de la mitad de la cota máxima definida. El dispositivo incluye conectividad inalámbrica, sensores y actuadores embebidos, interfaz de usuario y la posibilidad de conectar hardware externo para ampliar sus posibilidades de aplicación. El firmware permite el reporte periódico autónomo de los sensores a la nube y el control remoto de los actuadores, lo que habilita a los usuarios resolver problemas con IoT aplicando las experiencias de aprendizaje desde los primeros pasos de uso.

ABSTRACT

This work focuses on the design and implementation of an electronic device in hardware and firmware, as a minimum viable product for a teaching tool aimed at the Internet of Things (IoT) in the STEAM educational paradigm.

The main objective of this work is to reduce two barriers that hinder the teaching and learning of STEAM-IoT, one methodological and one economic. On the methodological side, design requirements are defined to facilitate the practical introduction of the subject to students. On the economic side, a restriction is placed on the direct unit cost that the product may have.

To achieve this objective, the work documents the definition of functional requirements, from suggested learning experiences to desirable product functionalities. The electronic design of the device is carried out using EDA software, components are acquired, and PCB boards are ordered from international suppliers. The manual assembly of the board, firmware programming, and experimental testing are also carried out.

The result of this work is a device that meets the design requirements, with a direct unit cost of forty-five thousand Chilean pesos, which is less than half of the maximum defined value. The device includes wireless connectivity, embedded sensors and actuators, a user interface, and the possibility of connecting external hardware to expand its application possibilities. The firmware allows the autonomous periodic reporting of sensors to the cloud and remote control of the actuators, enabling users to solve IoT problems by applying learning experiences from the first steps of use.

ACRÓNIMOS

PMV Producto Mínimo Viable. Dice relación con el diseño o definición de un producto cuyas características son las mínimas necesarias para validar hipótesis de un modelo de negocios de modo que permita levantar evidencia mediante un ejercicio empírico, minimizando el esfuerzo y los costos implicados. Se encuentra frecuentemente en la literatura como MVP (léase *emvipi*, del inglés *Minimum Viable Product*).

PCB Placa de Circuitos Impresos, sigla que proviene del inglés *Printed Circuit Board*.

MCU Microcontrolador, del inglés *Micro Controller Unit*.

STEAM Ciencia, Tecnología, Ingeniería, Artes y Matemáticas. De la sigla en inglés: *Science Technology Engineering, Arts and Math*. El término STEAM se trata de un modelo de aprendizaje interactivo y constructivista, basándose en el trabajo colaborativo y el desarrollo de proyectos [1].

IoT Internet de las Cosas, del inglés *Internet of Things*.

MQTT Protocolo de comunicación usado en Internet of Things, se traduce como Cola de mensajes telemetría y transporte del inglés *Message Queuing Telemetry Transport*.

API Interfaz de programación de aplicaciones, del inglés *Application Programming Interface*.

GPIO Pin de entrada o salida de propósito general, del inglés *General Purpose Input Output pin*.

CI Circuito Integrado.

DIY “Hazlo tu mismo”, del inglés *Do It Yourself*.

LDR Fotorresistor; del inglés *Light Dependent Resistor*.

LED Diodo emisor de luz; del inglés *Light-Emitting Diode*.

EDA Automatización de diseño electrónico; del inglés *Electronic Design Automation*.

CAD Desarrollo asistido por computador; del inglés *Computer-Aided Design*.

SMD Dispositivo de montaje superficial; del inglés *Surface Mounted Device*.

TH Dispositivo de montaje a través de orificio; del inglés *Through Hole*.

Breakout En el contexto de los dispositivos electrónicos hace referencia a la disposición espacial, dimensiones y funciones asociadas a los terminales de un dispositivo.

INTRODUCCIÓN

En este capítulo se exponen el contexto y la descripción del problema que motiva el trabajo de título, seguidamente, se describen los objetivos propuestos junto con el alcance esperado. La sección del contexto se divide en dos partes, la primera entrega una mirada general de la importancia de la educación STEAM y la segunda relata el origen de la oportunidad de hacer una contribución concreta en este campo.

1.1. Contexto

Importancia de la Educación STEAM

El desarrollo de la economía digital ha generado una creciente demanda de capital humano formado en Tecnologías de Información (TI), demanda que no ha podido ser satisfecha por la oferta de talentos a nivel global [2] y la brecha sigue aumentando [3] [4]. En Chile un 31 % de las vacantes en TI no logran ser cubiertas, en Latinoamérica esta cifra asciende al 48 % [5] y se estima un crecimiento de la demanda de talentos TI de un 7,7 % para 2022 en dicha región. Esta brecha se explica no solamente porque la economía digital en su crecimiento requiere más volumen de capital humano, sino que además está requiriendo nuevos roles que no se condicen con carreras universitarias tradicionales. Si bien la variedad de oferta de formación en carreras TI en educación superior, tanto oficial (reconocida por el ministerio de educación) como no oficial también ha aumentado, resulta necesario que la educación primaria y secundaria pueda preparar y orientar a niñas, niños y jóvenes para seguir -con éxito- una trayectoria vinculada a carreras STEAM en educación superior y así lograr mejorar la oferta de talentos TI al mercado y de científicas/os a la academia. En educación STEAM, es posible realizar experiencias de aprendizaje activo que estimulen la curiosidad, la exploración y el hacer, para el logro de la formación de competencias y pensamiento de alto nivel y la promoción de una valoración positiva sobre la utilidad de las Ciencia, Tecnología, Ingeniería y Matemáticas, de manera integrada y no por silos.

Colaboración con Fundación Ecoscience

En el año 2019 la Fundación Ecoscience solicita a Becerra diseñar un taller de Internet de las Cosas (en adelante IoT) para un programa educativo de difusión de la ciencia y la tecnología con jóvenes que dicha organización planificaba en su estrategia STEAM anual. El taller solicitado sería un piloto para probar el concepto y analizar continuidad y escala para actividades futuras de la fundación.

El taller propuesto por Becerra consideró algunas experiencias de aprendizaje mediante el uso de un kit de electrónica básico, que incluía una protoboard de 800 puntos, una placa DEVKIT V1 de DOIT [6] basada en la ESP32, cables y componentes básicos como resistencias, leds, pulsadores, etc. El taller fue aceptado y se realizaron una serie de capacitaciones en su uso a profesoras y profesores de la Fundación.

Al siguiente año, en 2020 la fundación escala el programa IoT y solicita a Becerra cotizar 100 kits de apoyo docente, que recogiera puntos de mejora basadas en la experiencia del pilotaje del año anterior y las buenas prácticas que se pudieran recomendar. Algunos de los aspectos de mejora clave identificados estaban relacionados con la sensación de frustración emergidas en pasos de armado de los circuitos y desarrollo de código previo a la realización de algunos experimentos y de la introducción al pensamiento sistémico y algorítmico. Por lo cual, al momento pensar una nueva versión del kit solicitado se consideró diseñar un kit que permita ofrecer experiencias de aprendizaje en las que profundizar de manera gradual en los aspectos técnicos, posibilitando la generación de un espacio experimental desde los primeros pasos de uso, es decir, sin tener que armar necesariamente circuitos ni desarrollar código, pero con las capacidades funcionales para permitir al estudiante hacerlo en la medida que desarrolle tanto las habilidades como el interés.

1.2. Descripción del Problema

Se identifican dos grandes barreras al acceso de educación STEAM en educación escolar, una de ellas es de orden económica y la otra metodológica. En primer lugar, la barrera económica, ya que la tecnología es costosa. En segundo lugar, la barrera metodológica, ya que es importante tener la habilidad de enseñar y utilizar la tecnología. Ambas barreras están relacionadas con la necesidad de contar con los recursos adecuados para enseñar STEAM de manera práctica, lo que implica la aplicación, análisis y eventual diseño de soluciones, en lugar de simplemente identificar y repetir conceptos.

Relacionado con la barrera de acceso económica

Actualmente existe una amplia oferta de equipamiento (dispositivos y kits) en el mercado que permiten el apoyo docente para la aplicación de experiencias STEAM en contextos escolares, no obstante aquellos que tienen un mayor foco en el área TI resuelven una de las barreras de acceso a la vez. En este sentido se identifican dos grupos, en el primero se encuentran aquellos dispositivos o kits que facilitan mucho la curva de aprendizaje, pero debido a su precio son restrictivos para las establecimientos de educación escolar, especialmente para la educación pública y particular subvencionada en Chile; además presentan limitaciones técnicas al momento de querer realizar experiencias más avanzadas. El segundo grupo se caracteriza porque sus equipos presentan un menor costo y por ende son más accesibles económicamente, sin embargo tienen una curva de aprendizaje más empinada, lo que dificulta su uso y adopción por la complejidad técnica que presentan.

Relacionado con la barrera de acceso metodológica

Durante años de experiencia en enseñanza y diseño de actividades de aprendizaje, el autor ha identificado un problema común al intentar enseñar robótica y programación a estudiantes: la gran cantidad de conceptos complejos que se deben aprender para entender el funcionamiento de un sistema electrónico, como algoritmos, lógica, señales, corrientes, voltajes y potencias, velocidad, fuerza y torque, frecuencia, y otros que en etapa escolar generalmente no están dominados y no resultan intuitivos, especialmente a la escalada de magnitud en que requieren ser utilizados. Esto lleva a plantear la siguiente pregunta: ¿Cómo permitir al estudiante implementar una solución práctica que conduzca a poner en valor el aprendizaje STEAM en virtud de la utilidad o beneficio que otorga en algo tangible y cercano a su contexto, evitando que la experiencia educativa esté expuesta a la pérdida de interés por intimidación ante la gran cantidad de conceptos involucrados.

Por lo tanto, el problema a resolver en este trabajo de título consiste en diseñar un dispositivo electrónico que permita ofrecer una forma más accesible y práctica de enseñar STEAM, con foco en IoT, para que las y los estudiantes puedan adquirir competencias y pensamiento de alto nivel, y valorar positivamente la utilidad de la ciencia, la tecnología, la ingeniería, artes y las matemáticas, experimentando menor fricción metodológica y por un costo acotado y accesible.

1.3. Objetivos y Alcance

El trabajo de título tiene por objetivo generar un Producto Mínimo Viable¹ de una herramienta de apoyo al aprendizaje STEAM-IoT, mediante el diseño de una tarjeta electrónica de desarrollo y su implementación en hardware y software, para facilitar la introducción práctica al tema permitiendo versatilidad funcional y un costo acotado².

Para cumplir con el objetivo planteado, se definen las siguientes criterios de cumplimiento:

Contribución al Objetivo	Criterio de cumplimiento
Facilitar la introducción práctica al aprendizaje en STEAM-IoT	El dispositivo consta de hardware embebido necesario para realizar experiencias de aprendizaje predefinidas desde el primer uso sin necesidad de adquirir, conectar o configurar partes adicionales.
Versatilidad funcional	Equipo cuenta con GPIOs disponibles y al menos un bus de comunicación en breakout, de modo tal que se puedan incorporar partes adicionales como nuevos sensores o actuadores.
Costo acotado	El costo directo del equipo no debe superar los cien mil pesos chilenos.

Cuadro 1.1: Criterios de cumplimiento del objetivo.

En términos simples, el dispositivo a desarrollar puede describirse como una tarjeta de desarrollo similar a un Arduino UNO en cuanto a componerse de un microcontrolador programable con GPIOs que permiten una gran versatilidad, pero incluyendo el hardware necesario embebido para que pueda ser empleada en experiencias de aprendizaje prácticas desde una primera sesión.

El trabajo considera el desarrollo del código necesario en forma de ejemplos específicos o un firmware general, que permita a estudiantes realizar las experiencias de enseñanza-aprendizaje con sólo energizar la placa, sin necesidad de tener conocimiento previo en circuitos electrónicos o programación. El producto en su conjunto a entregar ha de permitir al estudiante profundizar en ambas disciplinas en la medida que se adquieren nuevos conocimientos y se despierte su interés.

¹También llamado MVP por sus siglas en inglés *Minimum Viable Product*

²La Fundación Ecoscience comunica presupuesto para lote de 100 unidades en su solicitud de servicios en 2020 para implementar su proyecto educacional en IoT. El monto resultante corresponde a cien mil pesos por unidad, al aplicar corrección monetaria entre noviembre de 2020 y febrero de 2023, equivalente al 22,2 %, se obtiene un monto actualizado de ciento veintidós mil doscientos pesos chilenos [7]

PRELIMINARES

En este capítulo se presentan varios elementos conceptuales que sirven como base y orientación para el diseño del producto a desarrollar. En primer lugar, se aborda la educación STEAM como un paradigma educativo. En segundo lugar, se explica el concepto de Internet of Things (IoT), el cual se plantea como una disciplina tecnológica que permite proponer experiencias de aprendizaje STEAM. El IoT ofrece soluciones tanto a problemas cotidianos del entorno cercano al estudiante como a desafíos operacionales a nivel empresarial en diversas industrias, lo que hace que sea un enfoque muy adecuado para las habilidades y competencias de interés de este trabajo, tal como se describe en la sección 2.1. Finalmente, se presenta la disciplina del diseño de equipos electrónicos, y en particular el diseño de circuitos impresos, ya que es esencial para el desarrollo del producto que es el objetivo de este trabajo.

2.1. Educación STEAM

La palabra STEAM corresponde a la sigla anglosajona de *Science, Technology, Engineering, Arts and Math* que se traduce al español como Ciencia, Tecnología, Ingeniería, Artes (o Arquitectura en algunos casos) y Matemática. Es un paradigma educativo que integra disciplinas, tiene un enfoque que promueve el aprendizaje colaborativo e involucra necesariamente la aplicación práctica de las materias tratadas [8].

En educación, el paradigma STEAM cada vez toma más protagonismo como una forma de motivar y formar a estudiantes en las disciplinas que lo conforman [9], [10], [11], [12], [13]. Esto se debe a que, en la actualidad, habilidades como brindar soluciones creativas a problemas emergentes, la capacidad de trabajar en equipo y colaborativamente o lograr ver los desafíos de manera holística y no por silos, son más necesarias y valoradas que el memorizar contenidos o replicar procedimientos. El enfoque STEAM en educación ha demostrado ser efectivo para motivar la creatividad, gracias a que permite la exploración y el equivocarse.

Enfoque en determinadas competencias digitales

Si bien el objetivo de este trabajo apunta al desarrollo de una herramienta tecnológica que sirva de apoyo docente para el aprendizaje STEAM en su definición amplia, el proyecto tiene -por deseo y por diseño del autor- énfasis en la promoción de formación de competencias digitales asociadas al empleo de la tecnología para resolver problemas del entorno inmediato de los estudiantes. Específicamente se identifican algunas habilidades, competencias y valoraciones de especial interés formativo, las que a continuación se listan siendo éstas inclusivas pero no taxativas:

- la capacidad de adquirir información del ambiente a partir de sensores electrónicos y según ésta llevar a cabo acciones a través de actuadores;
- comprender el potencial de la tecnología en la generación de automatismos en tareas de monitoreo, acciones repetitivas o que requieran de conocimiento experto, de duraciones prolongadas entre otras características;
- comprender tanto el potencial como las limitaciones que otorgan un determinado conjunto de sensores y actuadores a la solución de un problema; ser capaz de idear y ofrecer soluciones basadas en tecnología a problemas prácticos del entorno;
- identificar el potencial que tienen los datos como fuente de análisis para la comprensión de un fenómeno;
- valorar a la ciencia y la matemática como fuente de conocimiento para entender el mundo físico que nos rodea y a la tecnología e ingeniería como medios eficaces de hacer uso fructífero de ese conocimiento de manera práctica;
- valorar la creatividad como la capacidad de identificar oportunidades, proponer soluciones en diferentes niveles de acción;
- valorar y ser capaz de practicar la colaboración como una forma de trabajar conducente a conseguir mejores resultados que los posibles sin dicha actitud y capacidad.

El logro de la formación de las habilidades y competencias descritas en el texto precedente depende mayormente de la metodología de enseñanza y del espacio docente, por lo que, respecto de este proyecto de título se consideran una consecuencia deseable, mas no un resultado a obtener dentro del alcance,

ya que no se dispone de los recursos ni el plazo para medirlo. De todos modos, el diseño de las experiencias de enseñanza-aprendizaje pivotaes definidas dentro del alcance del PMV se guían u orientan para ser conducentes a la formación de dichas habilidades y competencias.

2.2. Internet of Things

Si bien no hay una definición única del término, existe consenso sobre que el término *Internet of Things* hace referencia a dotar con la capacidad de cómputo y conectividad en red a objetos, permitiéndoles generar, consumir e intercambiar datos, sin la necesidad de intervención humana o siendo ésta mínima.

El dotar de estas capacidades a objetos que normalmente no clasificarían como computadores trae múltiples beneficios, como la implementación de automatismos, la adquisición de datos de múltiples naturalezas y fuentes para un mejor entendimiento del entorno y mejor toma de decisiones, operabilidad remota e inter-operabilidad, redundancia, entre otros.

Los principios conceptuales del IoT no son nuevos, sin embargo, las mejoras tecnológicas y los avances económicos han habilitado condiciones para hacer que la implementación de aplicaciones IoT sea cada vez más accesible. Las principales condiciones habilitadores son: la ubicuidad de la conectividad, la amplia adopción del protocolo IP en las redes, la miniaturización electrónica y de fabricación en general, avances en analítica de datos, el incremento de la computación en la nube y la economía computacional [14].

El IoT involucra electrónica, programación, sensores, actuadores, conectividad y ciencia e ingeniería en general. Dada la amplia aplicabilidad de esta disciplina es que se presenta ideal como herramienta educativa, ya que permite la implementación de proyectos de menor envergadura en contextos cotidianos y partir de éstos proyectar el alcance de su potencial. De esta forma se contribuye a la valoración de las disciplinas STEM y especialmente de la tecnología y la ingeniería como elementos clave para la solución de problemas de importancia en el mundo de hoy.

Ubidots

Ubidots es una plataforma de IoT en la nube que permite a sus usuarios, conectar y visualizar datos de dispositivos y sensores en tiempo real[15]. Permite a emprendedores probar prototipos y escalar a grandes proyectos con clientes industriales.

Esta plataforma ofrece una licencia no comercial llamada Ubidots STEM que permite bajo un límite de tráfico de datos diario, el uso gratuito de hasta tres dispositivos por usuario sin necesidad de suscripción de pago. Su objetivo es contribuir a la misión de hacer que el IoT sea más fácil de usar y accesible para más personas. Es por ello que se utilizará como interfaz digital del dispositivo a diseñar e implementar.

En cuanto a opciones de conectividad tiene habilitadas las tecnologías WiFi y Ethernet, Bluetooth, Sigfox, LoRaWAN y más. La plataforma también ofrece una API abierta y flexible que permite a los desarrolladores integrar fácilmente sus propias aplicaciones con la plataforma. Con Ubidots, los usuarios pueden crear paneles personalizados y gráficos interactivos para visualizar sus datos, así como definir alertas y notificaciones basadas en eventos. Además, Ubidots ofrece capacidades avanzadas de análisis de datos, incluyendo machine learning y análisis predictivo, que permiten a los usuarios descubrir patrones y tendencias en sus datos para tomar decisiones informadas.

MQTT

La sigla MQTT significa “Cola de mensajes telemetría y transporte” (del inglés *Message Queuing Telemetry Transport*). Es un protocolo de mensajería que utiliza el modelo de comunicación de publicación o suscripción en la capa de aplicación y requiere una conexión TCP permanente para la comunicación de dispositivos de IoT con recursos limitados y un ancho de banda restringido. Ofrece beneficios como ser ligero y eficiente, escalable, fiable, seguro y ampliamente compatible. El protocolo MQTT se ha convertido en un estándar para la transmisión de datos de IoT debido a que puede comunicar datos IoT de manera eficiente; admite la mensajería entre dispositivos a la nube y la nube al dispositivo [16]. Este será el protocolo de comunicación a utilizar en la implementación del firmware del producto.

2.3. Diseño de equipos electrónicos

Dada la naturaleza tecnológica y digital del producto a desarrollar, éste ha de ser un equipo electrónico, y por ende el conocimiento en prácticas de diseño e implementación de circuitos impresos es uno de los tópicos técnicos importantes a considerar en este trabajo.

El diseño de equipos electrónicos es una disciplina que abarca muchas áreas, exige un alto nivel de especialización y puede considerarse incluso un arte, pues requiere no solo de la expertise técnica del diseñador, sino también de su capacidad de proponer soluciones creativas en la intersección de las diferentes

alternativas de solución a las que se puede enfrentar. El diseño de equipos electrónicos abarca áreas del conocimiento como teoría de redes eléctricas, campos electromagnéticos, componentes electrónicas de lógica, potencia, comunicaciones, accionamiento, sensorizado, memoria, visualización, entre otras. También, se sirve de conocimientos en mecánica, en tanto conectores, botones y el equipo en sí mismos sufren esfuerzos mecánicos, lo que debe ser considerado en el diseño para evitar quiebres, desajustes o mal funcionamiento; la fabricación de un equipo involucra procesos de soldadura y calor; el diseño e implementación de un equipo electrónico requiere de proyectar dimensiones físicas tanto de componentes a montar y su orientación, los espacios a dejar libres necesarios para el montaje y las restricciones en las dimensiones y formas del equipo mismo. Esta disciplina requiere también conocer sobre fabricantes y proveedores de componentes electrónicas en la industria, así como proveedores de servicios de fabricación y montaje de circuitos impresos.

Herramienta de diseño CAD: EasyEDA

El software EasyEDA asiste el diseño equipos electrónicos al igual que EagleCAD o Altium, sin embargo este destaca por tener una licencia de uso gratuito, dar acceso a los usuarios para publicar y consumir bibliotecas de componentes [17] y la integración con otras empresas del mismo grupo económico que prestan servicios complementarios, como la venta de componentes, la fabricación de PCB, el montaje, testeo y despacho de PCBs.

2.4. Trabajos relacionados

Existe basta literatura científica respecto del uso del enfoque STEAM en entornos educativos escolares y sus beneficios, en esta sección se resumirán tres trabajos de interés conceptual. A continuación, se introducirán algunos productos que actualmente se encuentran en el mercado en educación STEAM y sirven de referencia para el trabajo técnico a desarrollar.

2.4.1. Ámbito educativo

¿Qué es la educación STEM y por qué es importante? [18]

El artículo se enfoca en explorar la definición de STEM a través de una revisión histórica de este tipo de educación desde sus inicios, presentando sus fundamentos y su impacto en las grandes innovaciones mundiales desde el siglo XX. El autor argumenta la integración de STEM, en primer lugar, en la ingeniería enfocada en avances tecnológicos útiles para la defensa nacional, el avance de la carrera espacial y para el desarrollo industrial de las grandes potencias. Luego, se exploran los diferentes hitos que llevaron a su inclusión

formalmente como enfoque educativo a partir de 1896, con el Acto de Morrison y la creación de algunas universidades norteamericanas. Se profundiza en la definición de las disciplinas que componen el acrónimo STEM, con especial énfasis en la tecnología e ingeniería, para luego brindar una explicación sobre la importancia del rol del educador para el logro de los objetivos de aprendizaje STEM en educación primaria y secundaria.

Metodología de Aprendizaje Basado en Proyectos (ABP) [19]

Este informe presenta, en sus diferentes capítulos, un currículum educativo para tercero y cuarto medio en Chile, a través de la metodología de ABP. En el primer capítulo, se describen los objetivos, el marco de referencia legal y algunas consideraciones sobre el fenómeno de la enseñanza junto con datos sobre el grado de inserción escolar de Chile. A continuación se explican las definidas “Habilidades para el Siglo XXI”, resumidas en: las herramientas necesarias para los estudiantes, las metodologías de aprendizaje, las formas de pensar y las maneras de vivir, como foco formativo central del currículum. El capítulo tres explica la metodología de ABP y su relación con la educación STEM; luego se exponen los estándares que debe idealmente contemplar un proyecto de educación bajo esta metodología. Los capítulos siguientes incluyen herramientas para el diseño, evaluación y desarrollo de proyectos educativos -formatos, cuadros informativos, entre otros-. El documento culmina con un listado de 32 proyectos concretos por áreas de aprendizaje.

Robótica y Proyectos STEAM: Desarrollo de la creatividad en las aulas de Educación Primaria [20]

El capítulo presenta los resultados de un estudio realizado a alumnos de quinto y sexto grado de educación primaria, enfocado en la observación y medición del aumento de la creatividad a través de la integración de proyectos STEAM y robótica. Las autoras inician con las definiciones medulares del tema -STEAM, robótica, constructivismo educativo, modelos EAC, entre otros-. Se defiende una visión constructivista del conocimiento, reforzada por estos modelos educativos que permiten a los estudiantes hallar respuestas y soluciones a los problemas presentados. Luego, se hace mención a las diferentes ventajas que presenta la vinculación entre STEAM y la robótica educativa en el proceso de aprendizaje de estudiantes de estas edades, dada su etapa de desarrollo cognitivo. En particular, se resalta la cooperación social, el interés y la motivación, el fomento de la creatividad, las habilidades para la resolución de problemas, entre otros. Seguidamente, se explica la metodología aplicada para realizar el estudio y sus objetivos, el robot diseñado y construido por los estudiantes, para luego presentar las fases del proyecto, sus resultados y las conclusiones.

2.4.2. Productos similares

Sphero

Como su nombre lo indica, Sphero es una esfera robot, programable, tiene capacidad de movimiento direccionable, posee diversos sensores y puede ser te-leoperada mediante un dispositivo móvil inteligente. Cuenta con una interfaz física amigable dada su sencillez esférica, lo cual la hace ideal para entornos de educación primaria. Entre sus capacidades cuenta con una brújula, un acelerómetro y giroscopio, encoders para sus motores, una pantalla de 8x8 leds y comunicación infrarroja. Posee una carcasa plástica que le da durabilidad, cuenta con una batería que otorga una autonomía de hasta 2 horas y se carga por inducción [21].

Es un excelente dispositivo para iniciar a niñas y niños de edades tempranas en educación STEAM, destacando por su facilidad de uso y lo entretenido que resulta aprender con él. Sin embargo, presenta limitaciones a la hora de querer conectar nuevos sensores o actuadores, al estar el robot dentro de una esfera plástica. Esto tiene como consecuencia que las experiencias de aprendizaje están limitadas por las capacidades de fábrica y el desarrollo de aplicaciones IoT se ve impedido. El precio unitario bruto de la versión Sphero Bolt alcanza los 170 mil pesos chilenos, y cuenta con una versión *mini* con funciones reducidas por 90 mil unidades de la misma moneda.

Little bits

Este producto consiste en una serie de bloques electrónicos unitarios, funcionales y conectables eléctrica y mecánicamente entre sí asistidos por imanes. Su diseño busca hacer más entretenido y amigable el aprendizaje y uso de circuitos electrónicos. Los bloques de construcción son circuitos impresos rectangulares de dimensiones similares a una moneda grande, los que en sus costados tienen conectores plásticos que facilitan el ajuste mecánico, la correcta polaridad eléctrica y la distinción cognitiva de las funciones de cada bloque al usar diferentes colores en el plástico según el tipo de bloque [22].

El producto permite armar diversos circuitos combinando los diferentes bloques unitarios, como sensores, actuadores, unidades de procesamiento, interfaces de conectividad y baterías. Este producto destaca por la versatilidad y facilidad de armado. Existen múltiples kits disponibles a la venta, uno introductorio se puede obtener por 68 mil pesos en el mercado nacional. Una desventaja económica de este producto, es que para poder aumentar el alcance de los proyectos hay que comprar solo kits de expansión de la marca, con conjuntos de bloques funcionales definidos (no se venden bloques por unidad), los

cuales presentan valores similares o mayores al kit introductorio. Además, no existen tantos módulos funcionales como componentes o módulos electrónicos disponibles.

Tarjetas de desarrollo con protoboard

Existe una gran número de tarjetas de desarrollo electrónico basadas en microcontroladores y microprocesadores con las cuales se pueden diseñar y probar, desde prototipos de productos extremadamente sencillos hasta sistemas muy complejos. Ejemplo de marcas o familias de estas placas de desarrollo son: IntelEdison, RaspberryPi, Arduino, DFRobot, Adafruit, Sparkfun, LattePanda, Particle, Teensy, micro:bit, entre muchos otros.

En esta categoría existe una exorbitante diversidad de productos en cuanto a prestaciones y precios, pero todas comparten dos características principales: primero la de tener una curva de aprendizaje más compleja al requerir de conocimiento técnico; y segundo, ser el tipo de producto que lidera en versatilidad en cuanto al rango de aplicaciones que se pueden trabajar con ellas.

Arduino MKR Family Los productos de la empresa Arduino son ampliamente conocidos en el gremio ingenieril, entre hobbistas y en el mundo educacional, en parte porque la compañía desarrolló hardware de propósito general con diferentes niveles de prestaciones y costos, lo que la posicionó muy bien en el mercado. En los últimos años la empresa ha desarrollado líneas de productos especializadas en diferentes ámbitos, ejemplos de ellas son: la línea Pro de alta gama con foco en la Industria o los productos en el segmento de educación. Arduino MKR es una línea de productos de hardware, específicamente tarjetas de desarrollo que comparten las siguientes características: factor de forma (dimensiones y disposición de conectores); alta conectividad integrada (GSM, GPS, WiFi, Narrowband IoT, Lo-Ra Network, entre otros) -lo que hace a estos dispositivos ideales para proyectos de IoT-; cargador y conector de batería tipo jst; y algún complemento de hardware embebido que habilite a la tarjeta para desarrollar proyectos sin necesidad trabajo y costo adicional [23]. Existen también variadas ¹shields en la familia MKR que amplían aún más las prestaciones de la tarjeta; al compartir el factor de forma es posible combinar diferentes modelos de tarjetas y shields, otorgando prestaciones para un muy amplio espectro de proyectos.

Se ha de tener en especial consideración como referente a la familia de productos Arduino MKR, ya que sus tarjetas presentan las características de versatilidad, conectividad, autonomía y rapidez de inicio identificadas como

¹Shield, palabra del inglés que significa *escudo o protección*. En jerga electrónica se usa para referirse a hardware adicional que se superpone al hardware principal, agregando funcionalidades adicionales.

parte de las necesarias para el diseño del dispositivo de este Trabajo de Título. No obstante, estas tarjetas carecen de una interfaz de usuario que permita interacciones y operabilidad por parte de los usuarios y el hardware adicional embebido es limitado. Una tarjeta Arduino MKR WiFi 1010 en Chile tiene un precio aproximado de 47 mil pesos.

Los tópicos a trabajar y trabajos relacionados presentados aportan una base conceptual metodológica y una referencia de mercado que sirven como guía para conducir el diseño e implementación de un dispositivo de apoyo a la educación STEAM-IoT pretendido en este trabajo de título.

DEFINICIÓN DE REQUERIMIENTOS

En el presente capítulo se identifican y describen los requerimientos para llevar a cabo el diseño del dispositivo. En la primera sección se describen las características funcionales en dos niveles, primero con lineamientos generales y segundo a partir de experiencias de aprendizaje específicas propuestas para desarrollar con estudiantes utilizando el dispositivo. En la segunda sección de este capítulo se establecen especificaciones técnicas a partir de lo descrito en la sección funcional.

3.1. Características Funcionales

3.1.1. Requerimientos Generales

En esta sección se describen las consideraciones que define el autor en base a su experiencia docente previa enseñando STEAM-IoT a jóvenes escolares.

La carga de un nuevo código al dispositivo debe ser simple

Descargar un nuevo código al dispositivo debe considerar conectarlo a un computador mediante un cable USB, y hacer click en la interfaz para iniciar la carga. Esto es importante porque existen tarjetas de desarrollo de bajo costo como el modelo *DevKit Dev Rev1* (basada en el microcontrolador ESP32) que requieren la activación manual del modo bootloader, ejecutando una rutina sobre los pulsadores mecánicos embebidos en la placa. Para lograr hacer simple la carga de un nuevo código para el usuario, se debe incluir hardware de puente entre USB y el puerto serial del microcontrolador. Se debe incluir también un puerto microUSB para conectar el cable.

Interruptor de encendido-apagado y pulsador de reinicio.

Si bien no es estrictamente necesario, se define incluir interruptor de encendido y apagado, ya que en caso de incluir el uso de batería, es más conveniente apagar el dispositivo mediante el uso de un interruptor que tener que desconectar la batería cada vez. También se considera la incorporación del uso de un pulsador para reiniciar el equipo, lo que aporta mucho a la comodidad de volver a ejecutar el programa según se requiera, sin tener que desenergizarlo.

Sensores deben entregar mediciones en escala de unidades conocidas

Resulta conveniente usar elementos conocidos previamente por los estudiantes en las actividades a realizar, especialmente variables físicas del entorno como la temperatura o la luminosidad del ambiente. Sin embargo al usar transductores simples y baratos, como un LDR DIY para medir cambios en la intensidad de la luz, los valores obtenidos son números enteros proporcionales a un voltaje. Esto requiere tener que explicar una serie de conceptos antes de llegar al punto de poner en valor el uso de la tecnología para simplificar o mejorar una determinada actividad, como por ejemplo el monitoreo automático del nivel de temperatura en la habitación gatillando una alarma cuando se pasa cierto umbral. Por lo tanto un requerimiento funcional para el diseño del dispositivo, es que los sensores a incorporar entreguen las mediciones en una escala conocida, como celcius para la temperatura, y no requieran un paso de calibración inicial.

Incluir interfaz de usuario

El uso de interruptores, pulsadores y potenciómetros es muy común y conveniente en aplicaciones reales de equipos electrónicos, como el ajuste de volumen, un selector de modo de operación, o botones para navegación de menús. Se define incluir dos interruptores, dos pulsadores y un potenciómetro.

Protección de sobrecorriente

Siempre es importante en un diseño considerar aquellos casos en que las cosas salen mal, sobretodo si el usuario final no es experto, es curioso y es un niño o niña. Por ello se define incluir un fusible cortacorriente a la entrada de alimentación del dispositivo.

3.1.2. Experiencias de aprendizaje pivotaes

A continuación se describen diferentes experiencias de aprendizaje que permiten introducir conceptos de monitoreo y automatización remota, mediante sensores y actuadores e interfaz web; a partir de dichas experiencias se desprenden requerimientos de diseño para el dispositivo. Las experiencias se plantean

en orden, de modo que lo aprendido en experiencias anteriores se entiende como punto de partida de la siguiente. En este apartado se describen experiencias de aprendizaje que se han de incluir en el producto tanto a nivel de hardware como de software, y otras que no se desarrollarán como parte del alcance de este proyecto, pero sí se considerarán sus requerimientos en el diseño del dispositivo de modo que puedan implementarse mediante la conexión de componentes de externas y el desarrollo de software adicional a modo de desafíos propuestos.

Cabe mencionar que la parte que corresponde a servicios e interfaz web no está considerada dentro del alcance de este proyecto, sin embargo se dispone de la plataforma de IoT industrial y Educación STEAM *Ubidots* para resolver la parte web de la solución como un servicio configurable a nivel de usuario.

En una primera instancia, el o la estudiante debe familiarizarse con el dispositivo (reconocer sus partes y energizarlo), la interfaz de programación (abrir un código de ejemplo y cargarlo) y la interfaz de Ubidots (iniciar sesión y configurar el dispositivo). Por lo tanto, la primera experiencia debe motivar el interés práctico, pero ser una aplicación sencilla para no introducir demasiada complejidad adicional a la exposición a nuevas herramientas.

Monitorear la temperatura de la habitación

Una aplicación sencilla y cercana, es un sistema de monitoreo de la temperatura de la habitación. Esta experiencia permite familiarizarse con conceptos como la tasa de muestreo y cuestionarse rangos de operación. En esta experiencia el o la estudiante debe lograr energizar y cargar al dispositivo un código de ejemplo que muestrea la temperatura y la reporta al servidor de ubidots. Para ello se debe modificar ligeramente el código para ingresar las credenciales de una red WiFi disponible.

Al final de esta experiencia, el o la estudiante debe ser capaz de:

- Conocer cómo energizar y encender el dispositivo.
- Cargar un nuevo programa desde IDE en computador.
- Modificar el código propuesto para dar un nombre al dispositivo (opcional).
- Modificar el código propuesto para ingresar las credenciales de la red WiFi.
- Ingresar al Dashboard de la experiencia en Ubidots.
- Interpretar las mediciones reportadas e indagar sobre su pertenencia temporal.

Requerimiento técnico a incorporar en el diseño: Sensor de temperatura.

Monitorear la temperatura de la habitación - realimentación visual al usuario

Tomando como punto de partida la experiencia anterior, el o la estudiante debe lograr un cargar un código de ejemplo que es capaz de recibir un valor numérico definido en ubidots y encender un una fuente de luz con una intensidad proporcional al valor del número recibido.

En ubidots es posible agregar múltiples elementos visuales configurables por el usuario. Particularmente en esta experiencia se utilizará un deslizador (o slider), que consiste en un elemento que se desplaza sobre un tramo de línea y entrega un valor numérico que corresponde a la proporción entre la posición del elemento y un rango numérico definido por el usuario.

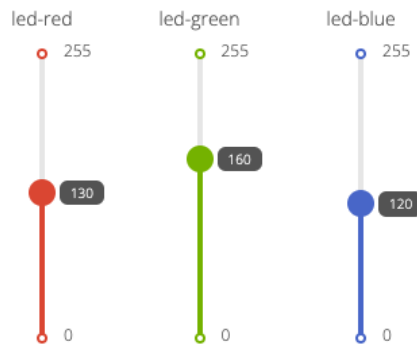


Figura 3.1: Deslizadores para selección de niveles de intensidad de LED RGB en Ubidots.

En esta experiencia se busca dar realimentación al usuario de la temperatura registrada, mediante la operación de un LED RGB según la definición de escenarios definidos según a la temperatura de la habitación.

1. Implementar deslizadores para que un operario modifique la intensidad de los colores del LED en el dispositivo de manera remota según observación del valor reportado en el dashboard.
2. Definir eventos con reglas lógicas en ubidots de modo que si la temperatura medida supera un umbral se presente al usuario un color rojo de máxima intensidad y que si la medición está bajo el umbral se presente al usuario el color azul en su máxima intensidad.
3. Hacer que el umbral de temperatura del ejercicio anterior sea definido por una variable sintética cuyos valores estén en un rango de 10 a 40 grados Celcius, proporcional al valor de la medición de voltaje del potenciómetro disponible en el dispositivo.

4. Crear dos variables sintéticas para mostrar al usuario un color que cumpla con lograr una proporcionalidad directa entre el valor de temperatura medido y la intensidad de color rojo, e inversa entre la medición y el color azul.

Al final de esta experiencia, el o la estudiante debe ser capaz y conocer los conceptos de:

- Modificar el Dashboard de la experiencia en Ubidots para agregar un deslizador numérico.
- Crear variables sintéticas y saber ocuparlas para implementar abstracciones sobre valores medidos a unidades de variables físicas y viceversa (a valores de entrada para actuadores).
- Utilizar deslizadores para controlar el comportamiento del LED en la placa
- Implementar automatismos de actuación sobre el sistema según mediciones del ambiente.

Requerimiento técnico a incorporar: LED RGB.

Monitorear la luminosidad en gráficos de tendencia y alertas

La luminosidad disponible en el ambiente es un factor importante para el correcto desempeño de las actividades cotidianas, especialmente a la hora de leer, escribir u observar una pizarra en una sala de clases. En esta actividad se propone al estudiante medir, registrar y graficar la intensidad lumínica del ambiente y configurar alertas sonoras cuando la luminosidad baje de cierto rango. Para ello se dispone del código de ejemplo a cargar en el dispositivo y se solicita crear una variable para recibir las mediciones de luminosidad en Ubidots e incorporar un gráfico de tendencia al Dashboard de la experiencia. El o la estudiante deberá configurar una regla para que cuando la luminosidad observada baje de cierto umbral una chicharra se encienda en señal de alerta. Esta experiencia se puede extender incluyendo un umbral superior y un cambio de frecuencia del tono a emitir por la chicharra.

Al final de esta experiencia, el o la estudiante debe ser capaz y conocer los conceptos de:

- Modificar el Dashboard de la experiencia en Ubidots para agregar un gráfico de tendencia con los datos observados.
- Utilizar deslizadores para controlar el comportamiento de la chicharra en la placa, tanto para intensidad en intensidad como frecuencia.

Requerimiento técnico a incorporar: Sensor de intensidad lumínica y chicharra.

Caso de uso: proponer un escenario donde aplicar lo aprendido

En esta experiencia se solicita a las y los estudiantes combinar las experiencias anteriores y proponer sus propios casos de aplicación donde puedan poner en práctica sus nuevos conocimientos.

Otras Experiencias Planteadas

Las experiencias de aprendizaje descritas anteriormente buscan introducir los nuevos conceptos gradualmente al mismo tiempo que ponen en valor el uso de la tecnología en contextos conocidos. Por ello, se promueve la simplicidad y un enfoque incremental top-down, es decir, la estudiante parte como usuario de alto nivel observando la utilidad del uso de la tecnología, y poco a poco va profundizando en el funcionamiento de los sistemas. A partir de generar una buena experiencia con la menor frustración posible, se espera despertar interés en querer saber cómo funciona lo que se está usando y así profundizar en el código del dispositivo y también en el hardware, tanto en aquellas componentes que vienen embebidas, como en la incorporación de nuevas para expandir el potencial de aplicaciones a implementar. En este contexto, se plantean brevemente otras experiencias que se podrían implementar con el dispositivo adicionando hardware externo.

- Conectar un servomotor que mueva una rendija de ventilación en función de la temperatura. Esta experiencia y la siguiente sirve para introducir los conceptos de realizamentación y lazo cerrado. Dado los largos transientes asociados a la temperatura de una habitación, el uso de un canal de red para conectar controlador (reglas en ubidots) no es un problema.
- Conectar un módulo relé y encender un ventilador o termocalefactor en función de la temperatura.
- Incorporar un sensor de movimiento para encender una luz de alerta o asistencia con accionada por un relé.
- Conectar una bomba de agua y un sensor de humedad en la tierra para monitorear y automatizar el riego de una planta en macetero. Configurar notificaciones automáticas por correo, mensaje o llamada ante determinados eventos, tales como el inicio o término del riego.
- Control de Invernadero. Usando los sensores de temperatura, humedad del ambiente, luminosidad, humedad en la tierra, automatizar ventilación, calefacción, riego y luz artificial para mejorar factores de cultivo de frutas y verduras.

- Estación de carga solar, esta es una experiencia más avanzada y desafiante. Conectar dos servomotores, y un sensor de radiación UV, construir estructura apropiada para mover un panel solar (de dimensiones razonables, 12cmx15cm de 12V) en dos grados de libertad usando los servomotores. Programar un seguidor de sol mediante las mediciones de radiación solar. Conectar un cargador de baterías y una batería lipo de 3.7V. Probablemente el consumo energético del dispositivo y los servomotores sea mayor a la energía a almacenar en la batería en una determinada ventana temporal. Sin embargo, ilustra el potencial de aplicación de tecnología y permite presentar un desafío bien contextualizado para motivar el interés del estudiante.
- Controlar actuadores de una dispositivo con mediciones de otro dispositivo. Esta experiencia resulta particularmente interesante para motivar el trabajo en equipo e ilustrar la capacidad del trabajo en una modalidad IoT donde las reglas de negocio se configuran en una interfaz y los equipos se convierten en legos, logrando un potencial de escalabilidad muy grande.

Mencionar que si bien son experiencias simuladas por los volúmenes de agua o el tamaño de los objetos, el pensamiento sistémico de fondo es el mismo para un caso real y esto pone en valor el uso de la tecnología para resolver problemas cotidianos y productivos, como lo son estar a gusto en un espacio o hacer uso eficiente de los recursos naturales.

3.1.3. Requerimientos Deseables

Para expandir el potencial de uso del dispositivo a más aplicaciones se propone incluir más componentes de hardware. El incorporar hardware por parte del usuario normalmente es una actividad laboriosa que debe ser supervisada, propensa a errores como pérdida de los componentes o quema de los mismos por sobrecarga o mal conexión y consume tiempo valioso del espacio docente en un taller. Disponer de más hardware, aumenta las combinaciones disponibles de funcionalidades para dar cobertura a más aplicaciones. A continuación, se describen tres módulos que se piensa pueden nutrir en este sentido al dispositivo a diseñar. No obstante, se declaran como módulos deseables por lo que se tendrán en consideración para el diseño del hardware, pero su incorporación en firmware y funcionamiento no constituye un factor estricto para medir el logro del alcance del proyecto, pues las experiencias de aprendizaje planteadas anteriormente se pueden realizar sin ellos.

Reloj RTC con pila

Un RTC corresponde a una componente electrónica que permite a un sistema mantener seguimiento del tiempo indistintamente de si el equipo se desenergizó por un tiempo o estuvo apagado o no. RTC viene del inglés *Real Time Clock* y se traduce como Reloj de tiempo real. Su funcionamiento normalmente involucra un pila o batería adicional dedicada exclusivamente a mantener el reloj encendido. El RTC tiene la capacidad de ser configurado, estableciendo una hora y fecha, y luego mantenerse actualizado independiente del estado del resto del equipo. Se propone incluir un módulo RTC en el diseño del dispositivo.

Batería y cargador

Con el objetivo de permitir una operación *off-grid*, es decir, que pueda funcionar indistintamente de tener disponible una fuente de alimentación eléctrica externa, se propone incluir una batería Lipo de 3.7V para alimentar el dispositivo. De manera complementaria, para simplificar el manejo de componentes externas al usuario, se propone incorporar al equipo un cargador de baterías.

Memoria SD

Con el objetivo de permitir una operación *offline*, lo que quiere decir sin conexión a internet, se propone incluir un medio de almacenamiento de información con capacidad para guardar miles de datos de manera no volátil, lo que quiere decir que no se pierdan cuando el equipo se desenergice. Para ello se propone incorporar un slot y el circuito necesario para controlar una tarjeta SD.

Combinando estos tres módulos, es posible operar el dispositivo en un modo *offline y off-grid*, ya que éste podría operar sin tener disponible alimentación eléctrica externa y ni WiFi. De esta forma, el equipo puede funcionar alimentado por la batería, las mediciones pueden guardarse tabuladas en la memoria SD, correctamente etiquetadas con una marca temporal otorgada por el RTC.

DISEÑO DE PCB

En este capítulo se presentan tres secciones, primero una descripción de las etapas generales para ejecutar el diseño del dispositivo, segundo la presentación de los módulos funcionales, y tercero el resultado del trabajo realizado en la forma de un plano esquemático y el plano de capas de la PCB.

4.1. Etapas de Diseño en software CAD

En general, al realizar el diseño de una PCB en software CAD especializado para esta tarea, se trabajan dos documentos: el plano esquemático y las capas físicas. Ambos documentos están íntimamente relacionados a partir de la definición de nodos comunes entre componentes y la selección de la versión de empaquetado de las mismas en el documento esquemático, lo que determina las conexiones eléctricas necesarias a realizar entre las componentes por un lado, y por el otro el área y forma a utilizar por dichas componentes en el documento. El software asiste para guiar al diseñador para que no se quede sin conectar los nodos necesarios, y respete reglas de diseño previamente establecidas. No obstante, si bien es una gran ayuda, no es infalible y el cuidado de los detalles es muy importante para obtener un buen diseño.

Diseño del plano esquemático

Para poder realizar el trabajo de manera ordenada y sistemática, lo primero es definir módulos funcionales para dividir el diseño en bloques que se pueden ejecutar de uno en uno. Ejemplos de estos módulos son: un módulo de alimentación de energía, un módulo de comunicaciones, interfaz de usuario, uno o varios módulos de sensorizado, etc.

Una vez definidos los módulos, se plantea un circuito inicial que resuelva el propósito del módulo. En este trabajo el circuito inicial se obtuvo principalmente de dos maneras. La primera consiste en combinar módulos de circuitos

previamente conocidos por el diseñador, como módulos de amplificación, contadores, accionamiento de actuadores por medio de transistores, etc, teniendo el cuidado de ajustar los valores de las componentes según los niveles de operación necesarios para el contexto de aplicación del diseño. La segunda forma empleada para plantear un circuito inicial consiste en buscar un circuito integrado diseñado para cumplir el propósito del módulo y en los niveles de operación requeridos, y replicar el plano propuesto por el fabricante en la hoja de datos, revisando la sección de aplicaciones típicas, para luego realizar ajustes en las componentes acompañantes de ser necesario.

Selección de componentes

Una vez definida una primera versión del plano de un módulo, lo que se obtiene es un modelo. Como siguiente paso, se requiere especificar las componentes a emplear en términos no sólo de sus parámetros eléctricos resultantes del modelo sino que también de su forma física (empaquetado específico) y rangos de operación. Dado que las componentes se adquieren de proveedores y fabricantes en particular, los programas de diseño -en este caso EasyEDA- admiten registrar los números de parte tanto de fabricante como de proveedor, por ejemplo, Texas Instruments como fabricante y Digikey como proveedor. Esto resulta muy útil en la etapa de compra de materiales, según se explica en la sección 5.1.

Para la selección de componentes se tuvo presente las siguientes consideraciones generales:

- Cotizar en proveedores reconocidos, que oferten productos de fabricantes reconocidos. Esto es importante porque de esta forma se asegura la calidad de las componentes, por ejemplo con varianzas mínimas o acotadas de los parámetros funcionales, tales como ganancias eléctricas, o dimensiones físicas.
- En el caso de ser una componente no genérica que depende de uno o pocos proveedores, verificar que no esté descontinuada, ya que si en el futuro se quiere replicar el equipo sin tener que rehacer el diseño, debe haber continuidad de oferta por parte del proveedor.
- Verificar que el stock sea alto, relacionada al punto anterior.
- Comparar los precios con productos sustitutos para elegir aquella de mejor precio dentro un mismo nivel de calidad.
- Revisar si tiene disponibilidad inmediata para no retrasar el proyecto por espera de entrega de la fábrica.

- Verificar que el producto se venda por unidad en lotes mínimos menores que la cantidad necesaria para el proyecto.
- Verificar que el formato de entrega sea el necesario para la técnica de montaje. Por ejemplo, si se va a ocupar una máquina de montaje automática, las componentes deben venir en rollos perforados para poder ser cargados en la máquina.
- Estandarizar el uso de empaquetados y valores de resistencias. De esta forma se compran unidades de una misma componente, en vez de comprar menos unidades de dos distintas. Esto hace permite aumentar la eficiencia de costos por materiales (compra en volumen) y de montaje (trabajo por lotes).

También se tuvo presente las siguientes consideraciones específicas:

- Para seleccionar resistencias, una vez definidos sus valores de resistividad, se calcula la potencia máxima a disipar y se le suma un factor de seguridad. Luego se procede a buscar en el catálogo del proveedor (no del fabricante) -en este caso digikey- según los parámetros definidos.
- Para seleccionar condensadores una vez definidos sus valores de capacitancia del modelo, se analizan los niveles de operación de voltaje y luego se procede a buscar en el catálogo del proveedor, según los parámetros definidos.
- En el caso de la selección de los interruptores se realizó la búsqueda considerando minimizar tamaño, que fueran SMD, y por el menor costo posible.
- En el caso del potenciómetro se buscó que este tuviese una perilla vertical de al menos un centímetro de largo para que resultase fácil y cómoda de operar con los dedos directamente. También se definió que fuera de montaje TH para mejorar su anclaje y así evitar que la soldadura se quiebre por esfuerzos mecánicos, dado que el largo de la perilla permite ejercer torque sobre la base del potenciómetro.
- En el caso de los circuitos integrados ESP32 y FTDI, correspondientes al microcontrolador y al puente de comunicación USB/Serial respectivamente, la búsqueda fue directa porque son de fabricante único y estas componentes por su grado de especialización se eligieron juntas según conocimiento y experiencia previa.
- El criterio de selección de los IC del módulo de alimentación consistió en implementar una marca reconocida aunque tuviera un alto costo, ya que

estos dispositivos deben ser de alta confiabilidad para asegurar un bajo ruido en el voltaje de alimentación y así no afectar el normal funcionamiento del microcontrolador y otros IC que realizan procesamiento, como algunos sensores, el puente USB/serial o el módulo para integrar una tarjeta SD.

- Los sensores de temperatura y sensor de luz se seleccionaron mediante una investigación bibliográfica y foros especializados, ya que se definieron los siguientes requerimientos. Primero, que fueran operables por el bus de comunicación I2C de modo de se pudieran incorporar más dispositivos sin necesidad de usar más GPIOs; el compromiso aceptado fue limitar la tasa de muestreo máxima posible a más de medio segundo por sensor. Segundo, que entregasen mediciones en unidades del sistema internacional. Tercero, que tuvieran al menos un decimal de precisión sobre la unidad de medición.

Una vez definida la componente a usar en un módulo, se busca en el catálogo de componentes disponible en la librería pública de EasyEDA por número de parte y se reemplaza en el diseño esquemático, asegurándose que el par símbolo y empaquetado sea correcto. Se debe verificar que el diseño del empaquetado corresponda con las especificaciones sobre las dimensiones físicas definidas por el fabricante en la hoja de datos, junto con la asociación de pares entre los terminales del símbolo y el empaquetado sea correcta; si estas verificaciones no se realiza se corre el riesgo de observar fallas en el proceso de montaje y por ende en el correcto funcionamiento del equipo.

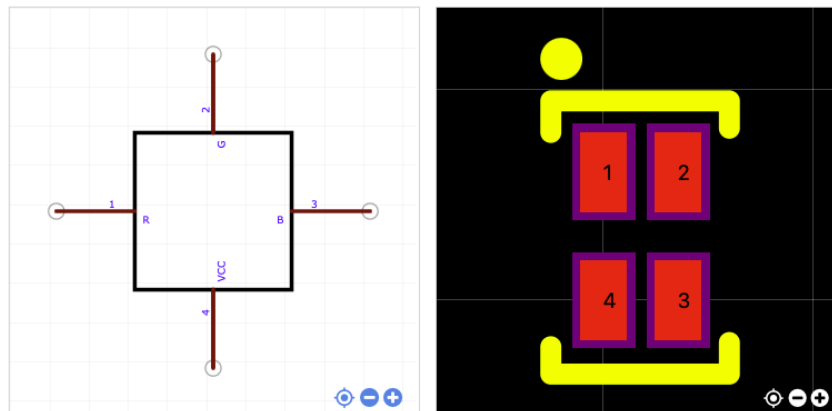


Figura 4.1: Ejemplo de par símbolo:empaquetado de la componente LED RGB. Note la correspondencia entre los terminales numerados del símbolo (imagen de la izquierda), con los terminales numerados de la huella del empaquetado (a la derecha).

Diseño de capas físicas

Esta etapa requiere conocimiento técnico, experiencia y tiempo ya que se trabaja de manera iterativa. Se inicia creando el documento del tablero (*board* en inglés). Al hacerlo, automáticamente se emplazan los modelos de la huella de los empaquetados de las componentes. Como siguiente paso, es necesario definir las dimensiones de área basal de la PCB a implementar, ya que esto tiene impacto en la distribución posible de las componentes, así como en el costo final de la PCB (mientras más grande, mayor será el costo). Además, al usar dimensiones bien pensadas, es posible optimizar el uso del espacio panelizando la fabricación por lotes. Antes de iniciar la distribución espacial de las componentes en el tablero propiamente tal, es necesario revisar el gestor de las reglas de diseño, en el cual se definen parámetros como el ancho mínimo del trazado de una pista o la mínima distancia de espaciado entre ellas. De este modo, el programa EDA puede asistir al diseñador verificando y alertando oportunamente al diseñador si en algún lugar de la PCB no se cumplen dichas reglas; de este modo se pueden corregir los errores oportunamente antes de generar los archivos de fabricación.

4.2. Diseño por módulos

En esta sección se presenta la división del trabajo por módulos, indicando las principales consideraciones de diseño que tuvieron lugar en cada caso. A continuación, se presentan los siete módulos trabajados en prosa, los planos esquemáticos se encuentran en los anexos al final de este documento.

4.2.1. Módulos de Energía

Este módulo se encarga de recibir la alimentación eléctrica al dispositivo y proveer energía a los otros seis módulos según sus necesidades en particular. Este módulo se divide en cuatro submódulos: La etapa de alimentación hacia la PCB, la conversión a 5V, la conversión a 3,3V y la etapa de carga de la batería (indicada en los requerimientos deseables).

Alimentación hacia la PCB

Se compone de un puerto microusb para recibir alimentación y el bus de datos USB. Se dispuso un diodo en el terminal de voltaje $V+$ para restringir contracorrientes en caso de falla del equipo y así evitar dañar el computador al que estuviera conectado. Se incluyó un interruptor de encendido y un fusible como etapas previas entre el diodo y la siguiente etapa del módulo de alimentación, la conversión a 5V. También se incluyó un mecanismo de autoselección

de fuente de alimentación para que el dispositivo se provea de la energía de la batería en caso de esta estar disponible y se desconecte el cable USB.

Conversión a 5V

Esta etapa se encuentra alimentada por el submódulo anterior y su función es proveer 5V a las componentes que se alimentan a esta tensión (como el puente USB-Serial) y al breakout de la placa para conectar otros dispositivos externos. El corazón de este módulo es el circuito integrado TPS61220 de Texas Instruments, el cual permite elevar la tensión a 5V independientemente de si la placa se alimenta a 4.5-5V desde un USB o a 3.3-3.8V desde una batería de Litio. El de base plano implementado corresponde al sugerido por el fabricante. Se eligió este CI ya que cuenta con buena reputación por su calidad y tiene un rango de operación tal que permite una corriente máxima de 2A, suficiente para el contexto de aplicación del dispositivo.

Conversión a 3.3V

Similar a la etapa anterior, ésta se encuentra alimentada por el submódulo que le precede y su función es proveer 3,3V a las componentes que se alimentan a esta tensión (principalmente la EPS32 y sus relacionados, como componentes lógicas y pullups) y al breakout de la placa para conectar otros dispositivos externos. Este submódulo se sustenta en el regulador lineal LM1117, también de Texas Instruments, el cual permite una corriente máxima de 800mA. El plano de base utilizado corresponde al sugerido en la hoja de datos del fabricante.

Carga de la batería

Como una funcionalidad deseable, se considera la opción de tener una batería que de autonomía al dispositivo. Para ello se implementó un cargador de baterías del fabricante Microchip, el cual destaca por adherir a las políticas y buenas prácticas para cargar baterías desde puertos USB, además de su bajo costo y pequeño tamaño. Incluye el control de LEDs para indicar estados de operación, como carga en proceso, batería cargada. En el diseño también se incluye un conector JST hembra de 2 pins, que es el conector que normalmente traen las baterías de litio para aplicaciones DIY de 3.7V.

4.2.2. Módulo del Microcontrolador

La selección del microcontrolador a utilizar fue objeto del trabajo realizado en la asignatura Proyecto de Titulación, en el cual se determinó el uso del microcontrolador ESP32-WROOM-32D de Espressif Systems. Esta compañía fabrica y vende sus modelos y ha posicionado su familia ESP32, que destaca

por incluir embebidos en el chip los módulos de comunicación WiFi y Bluetooth. Más adelante en la sección 4.2.2 se presentan algunas las especificaciones técnicas que destacan a este microcontrolador para su empleo en este proyecto.

En este módulo de diseño se trabajan dos submódulos, el primero corresponde a la alimentación y asignación de pins del microcontrolador a los diferentes componentes funcionales, y el segundo a la conexión del puente de comunicación USB-Serial, el cual permite descargar el firmware desde el computador al dispositivo, así como enviar datos al computador para debug o monitoreo de la ejecución del programa. En este segundo submódulo se define utilizar el puente FTDI FT232 recomendado para la ESP32, al cual se le conectan los canales de datos D_+ y D_- de la entrada USB. Mediante un arreglo de transistores es posible inducir el estado de Boot en la ESP32 a través de sus pins IO0 y Enable los cuales tienen una resistencia de pull-up interna, gestionados desde los pins del FTDI RTS y DTR.

Una tarea no mencionada en las etapas generales de diseño de la sección anterior, que requiere cuidado y tiempo, corresponde al análisis de los pins de entrada salida disponibles en el breakout del microcontralor en términos de los periféricos que cada uno tiene disponible, ya que se deben analizar los requerimientos de los módulos a diseñar y asignarlos a los módulos según lo requieran. Esta tarea es fundamental para obtener un uso eficiente de los GPIOs disponibles, y requiere un entendiendo profundo de los requerimientos funcionales, del funcionamiento técnico del microcontrolador y algo de intuición sobre el uso potencial o futuro que pueda darse al equipo. Observe la distribución diseñada en la figura 7.7 en la sección de anexo correspondiente.

Microcontrolador ESP32 WROOM32D

Los MCU son esencialmente sistemas embebidos con tres grandes componentes principales y se diseñan en diferentes arquitecturas, pero la más importante es la unidad de procesamiento que automatiza lo que se quiere realizar, se acompaña de un segundo elemento, la memoria para guardar el programa y sus las variables, y el tercer elemento es el conjunto de periféricos que permiten interacciones de entrada y salida de información.

Si bien un equipo electrónico en general consta de varias dimensiones o módulos de diseño -a saber, alimentación (energía), procesamiento, comunicaciones, interfaz de usuario, entre otros- el equipo a diseñar no presenta restricciones categóricas ni cuantitativas en estos tópicos, salvo la premisa general de conseguir un producto de bajo costo respecto de los productos sustitutos. Los principales requerimientos funcionales del equipo a diseñar son la conectividad WiFi y Bluetooth, la capacidad de interacción con sensores y actuadores mediante puertos de comunicación y puertos de propósito general (tanto embebidos en el equipo como externos) que permitan desarrollar experiencias

STEAM, la capacidad de almacenar datos en una memoria externa (como una tarjeta microSD), además el equipo debe ser reprogramable.

ESP32 corresponde a una serie de módulos MCU desarrollados por Espressif Systems a lo largo de los últimos ocho años, incluyendo variaciones en algunas especificaciones de interés en cada lanzamiento [24].

Estos dispositivos se caracterizan por ser un todo en uno. Incluyen una gran cantidad de funcionalidades y características que les otorgan mucha versatilidad, una muy importante y distintiva es la conectividad Bluetooth y WiFi embebidas, lo que les ha posicionado en un amplio rango de aplicaciones, especialmente en IoT. Un compromiso que trae consigo esta gran cantidad de funcionalidades de fábrica, es que el dispositivo consume más potencia relativo a otros MCU bajo un mismo test de desempeño, esto ya que tiene que alimentar todo el circuito, aún cuando no se estén utilizando todos los módulos. En este trabajo se utiliza el modelo ESP32 WROOM-32D (en adelante ESP32) que corresponde a un modelo de 2017 aún vigente.

Desde la hoja de datos oficial del dispositivo, versión 2.3 del año 2022, se observan las siguientes características relevantes:

- Es utilizable en un amplio rango aplicaciones, desde redes de sensores de bajo consumo, hasta tareas muy demandantes como codificación de voz o transmisión de música y decodificación mp3.
- Cuenta con una antena embebida para WiFi y Bluetooth de 2.4GHz. Opcionalmente, se puede adquirir la versión que trae un terminal para conectar una antena externa de mejor alcance. Incluye también los periféricos y memorias necesarias para que sea transparente al programador el hardware.
- Dispone de una gran cantidad de periféricos y módulos de interfaz, como SPI, I2C, PWM especializados para el control de LEDs y motores, interfaz para SD, sensor capacitivo, bus CAN, entre otros.
- Trae integrada una memoria Flash SPI de 4MB.

La ESP tiene muy buenas prestaciones en memoria. Tiene una memoria Flash (con SPI integrado lo que quiere decir que no usa GPIOs adicionales) y una SRAM. Adicionalmente tiene dos memorias de 8KB SRAM integradas en módulos RTC (del inglés *Real Time Clock*, reloj en tiempo real).

- Memoria Flash SPI integrada de 4MB
- Memoria SRAM 520KB para variables de programa.

La ESP32 con Bluetooth y WiFi integrados en el empaquetado. El WiFi maneja el protocolo 802.11 b/g/n que alcanza hasta los 150Mbps en 2.4[GHz],

banda base disponible en todos los routers domésticos. Un punto en contra de este MCU es que funciona sólo en redes personales o domésticas y no maneja redes categoría enterprise, lo que puede ser un problema en establecimientos educacionales con una gestión más desarrollada de las redes internas. En el caso del bluetooth, maneja un protocolo v4.2 y LE, expandiendo el rango de aplicaciones e integraciones posibles.

En términos generales, la ESP32 presenta 34 GPIOs bidireccionales con la posibilidad de multiplexar a cada pin al cada terminal del empaquetado a discreción. Este dispositivo es un SoC (System on a Chip) muy completo ya que tiene periféricos de muchos tipos, algunos de ellos no son relevantes para este trabajo y solo se mencionarán, como el acelerador de hardware de algoritmos generales, controlador TWAI, Contrador de Pulso, Control Remoto por Infrarojo, I2S, un sensor de efecto Hall Sensor y JTAG (no se espera que usuarios finales hagan debug a nivel de registros).

Se puede programar con Arduino IDE. La interfaz de programación del MCU tiene un alto impacto en el correcto uso dispositivo por parte de los usuarios finales, aún cuando el IDE no es parte constitutiva de ningún MCU en sí mismo. ESP21 cuenta con paquetes de soporte oficiales de los fabricantes para el Arduino IDE. Este software puede descargarse gratuitamente y tiene versiones nativas para Windows, macOS y Linux, además de una versión online y una versión ejecutable para Windows. Cuenta con amplia documentación y comunidades online, además de ser un IDE conocido por ser simple de usar.

4.2.3. Módulo de Sensores

En este módulo se incorporan tres sensores, los cuales miden seis variables físicas del ambiente. Los tres sensores se conectan al bus de comunicación I2C, por lo que el diseño de sus planos consisten principalmente en conectar los canales SDA y SCL, condensadores de estabilización de voltajes de referencia y resistencias de pull-up y pull-down en los pins de selección del número de identificación del dispositivo en el bus (parámetro *address*). En todos los casos se dispuso para este fin, en el plano, resistencias de 0 Ohms, para dejar la huella disponible en el circuito, y permitir tomar la decisión de usar pull-up o pull-down al momento de montar.

Sensor de Intensidad Lumínica

Se seleccionó para este diseño el sensor TSL2561 del fabricante TAOS (Texas Advanced Optoelectronic Solutions). El TSL2561 posee un fotodiodo de banda ancha (rango de luz visible e infrarrojo), y un CI CMOS que sólo res-

ponde en banda infrarroja. Entrega las mediciones en unidades de lux, lo cual evita tener que convertir los datos y hacer procesos de calibración. Este sensor destaca por aproximar la respuesta de sensibilidad a la del ojo humano, tener la opción de programar interrupciones según un umbral definido por el usuario, modo de comunicación a 400kHz por I2C, ganancia análoga programable, filtra el ripple entre 50 y 60 Hz aportado cuando la frecuencia del suministro eléctrico se replica en la incandescencia de la iluminación artificial.

Sensor de Temperatura, Presión y Humedad

Para este módulo se eligió el sensor marca Bosch modelo BME280. Este sensor destaca por manejar dos interfaces de comunicación digital (I2C y SPI), un consumo de corriente de $3.6[\mu A]$ cuando se leen las tres variables a $1[Hz]$ y un rango de operación realmente versátil: $[-40, 85]gr.Celsius$; $[0, 100] \%$ de humedad relativa y $[300, 1100]$ hecto Pascales.

4.2.4. Módulo de Actuadores Embebidos

LED RGB

Se implementó un LED RGB de alto brillo con pantalla difusa y de montaje superficial (SMD). Para obtener una buena potencia de iluminación se optó por buscar un modelo que funcione a 5V, lo que implica que no se alimenta directamente desde un pin de la ESP32. El accionamiento se realiza a través de transistores MOSFET N-Channel, con capacidad de corriente de $310[mA]$ bajo un máximo de tensión de $60[V]$.

Chicharra

Para este submódulo se buscó una chicharra piezoeléctrica sin oscilador interno, para bajar el coste. Esto implica que el pin de accionamiento desde la ESP debe tener disponible el periférico PWM. Para obtener una potencia de sonido máxima, se decide alimentar desde la fuente de 5V y así no consumir potencia ni introducir ruido al voltaje de alimentación de la ESP (regulador de 3V3 de 800mA de corriente máxima). Se define que el pin de accionamiento será el IO0 mediante un transistor MOSFET N-Channel. Este transistor admite corrientes de hasta $2,8[A]$ a un máximo de $20[V]$. Note que dado que el consumo de potencia de la chicharra es mayor al de cada color del LED RGB anterior, se debe usar un MOSFET de mayor capacidad de corriente. Este transistor es tiene un mayor precio.

4.3. Diseño realizado

El resultado del trabajo de diseño de la PCB realizado en el software EasyEDA se presenta a continuación en las figuras 4.2 y 4.3, en las que se muestran el plano esquemático y el diseño de la placa respectivamente.

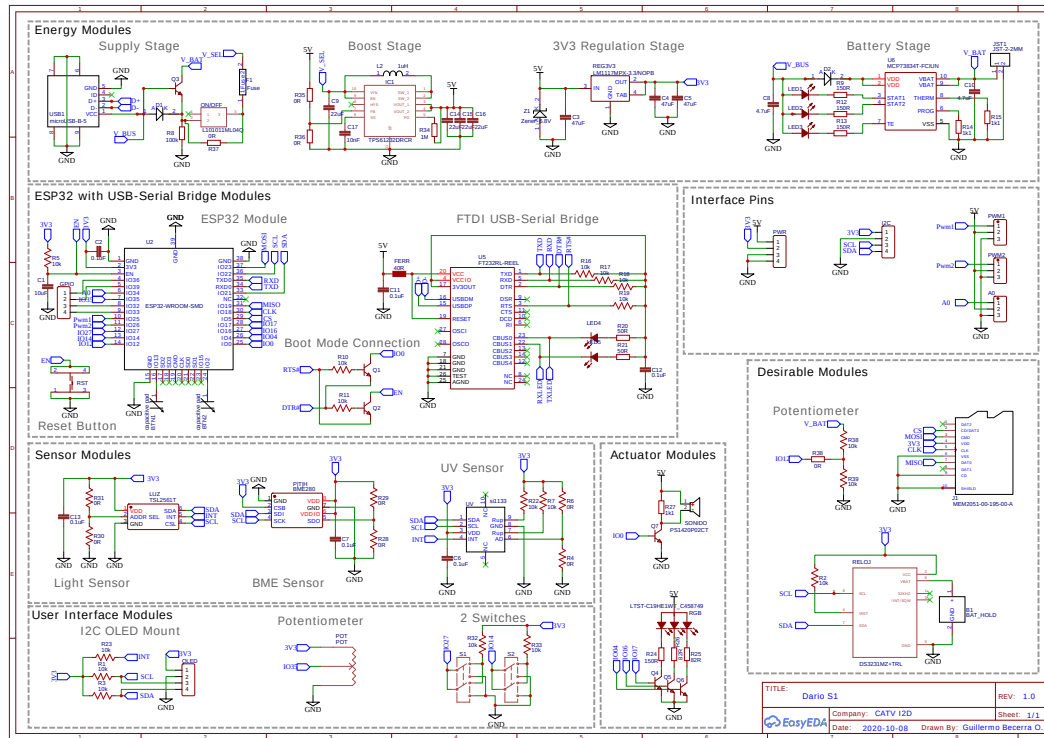


Figura 4.2: Plano esquemático del diseño realizado.

En los anexos y en los archivos adjuntos se encuentra el plano original para estudio y análisis.

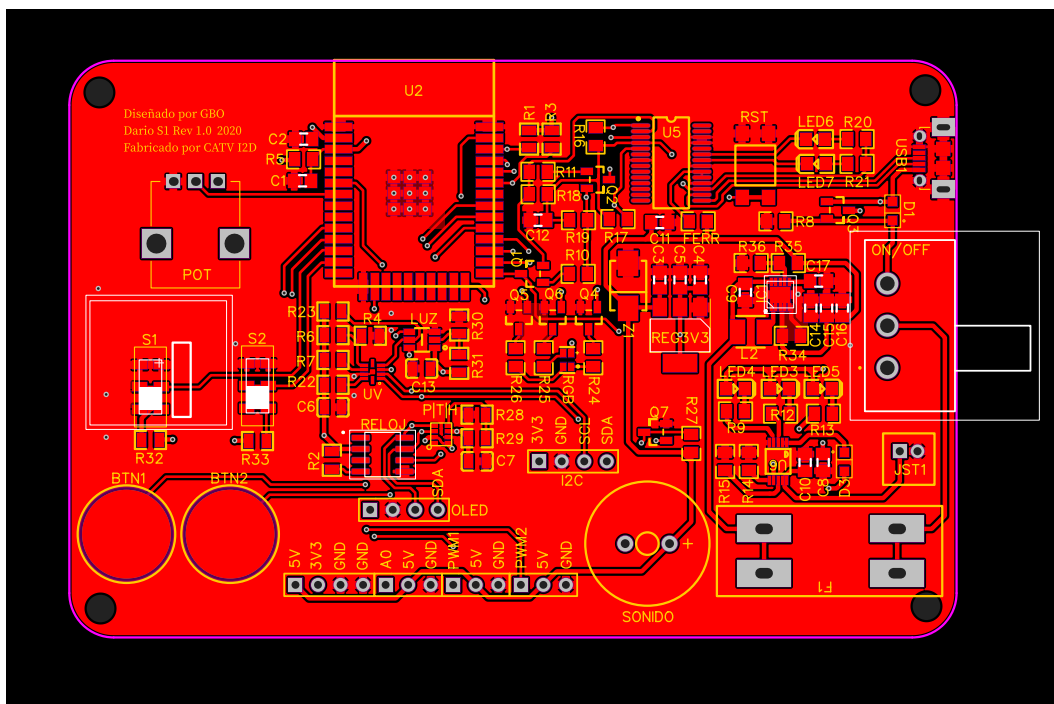


Figura 4.3: Vista del plano superior de la PCB diseñada en EasyEDA.

FABRICACIÓN Y MONTAJE

En este capítulo se describen las tres etapas ejecutadas para obtener el dispositivo en físico a partir del diseño realizado en el capítulo anterior. Una vez se define el cierre de la versión 1.0 del dispositivo, se procede a generar dos exportables: el listado de componentes (BOM) en base al archivo del plano esquemático y los archivos Gerber desde el plano de capas físicas. La generación automática de estos archivos es una de las principales ventajas del uso del software EDA.

5.1. Adquisición de componentes mediante BOM

El listado de materiales normalmente consiste en un archivo del tipo valores separados por coma (*.csv) en que se tabulan y describen las componentes a utilizar. Entre los principales atributos por ítem se encuentran:

- Identificador numérico único de la componente en el diseño, valor correlativo que parte en 1.
- Cantidad de veces que se usa la componente por dispositivo.
- Nombre distintivo otorgado por el diseñador para identificar la componente en el plano. Esta columna es importante ya que normalmente el proveedor de componentes entrega las componentes por lotes en contenedores, al cual se le adhiere una etiqueta con la referencia de nombre del diseñador, lo cual facilita el trabajo de montaje posterior.
- Valor eléctrico o característica funcional de la componente, por ejemplo: $4,7[\mu F]$, $1[k\Omega]$.
- Etiqueta de referencia de la instancia de la componente en el circuito. Por ejemplo, si una resistencia de $10[k\Omega]$ se utiliza 3 veces, las instancias por defecto que generará el software EDA serán R1, R2 y R3. Estas etiquetas normalmente (como es el caso de este trabajo) se incluyen en la capa de

impresión de serigrafía en la placa (incluida en los gerber files). De este modo se facilita el montaje y el análisis del circuito en la placa apoyándose en el plano esquemático, el cual también indica esta etiqueta.

- Huella del empaquetado, por ejemplo 0805, que designa las dimensiones y forma del área donde se va a soldar la componente a la placa, en el caso del ejemplo una resistencia de montaje superficial con forma de paralelepípedo.
- Número de parte del fabricante: serial única identificadora dentro de la oferta de productos del fabricante.
- Número de parte del proveedor: serial única identificadora dentro de la oferta de productos del proveedor.
- Nombre del proveedor (para saber de quién es el número de parte de proveedor indicado en la columna anterior).
- Opcionalmente se puede incluir: un link al producto en la página del proveedor, el nombre del autor de la componente en la biblioteca de componentes, el tipo de montaje, el precio, la cantidad de pines, etc.

Una vez exportado el archivo BOM en formato *.csv u otro formato de datos tabulados, éste se ha de importar directamente desde el sitio web del proveedor, indicando un múltiplo entero para repetir el BOM en función de la cantidad de dispositivos a fabricar; esta acción genera un carro de compras con las cantidades necesarias. A continuación, se describen algunas buenas prácticas adicionales:

- Considerar holgura en la cantidad a comprar. En este caso, desea fabricar 100 unidades, pero se cotizan 112, es decir se agregan componentes adicionales para absorber la merma del proceso de montaje y tener stock si se llega a requerir dar soporte al cliente por garantías de post venta o servicio técnico. En caso de tener un presupuesto restringido se puede bajar el número de componentes de holgura de las partes de mayor valor.
- Se recomienda hacer una verificación detallada sobre el stock disponible y efectivamente cargado al carro de compras.
- Analizar si alguna componente presenta un volumen cercano al quiebre del escalón de precios. Por ejemplo, el cálculo automático del carro de compras resulta en 95 unidades de un ítem y el valor unitario de esa componente tiene un valor de 0.5 dólares, al comprar de 1 a 99 unidades; mientras que el valor cambia a 0.4 dólares por unidad comprando entre 100 y 1000 unidades. Es más conveniente comprar 100 unidades dado

que el precio por esa cantidad corresponde a 40 dólares, mientras que al comprar 95 unidades el monto a pagar corresponde a 47,5 dólares. Esto tiene un impacto relevante cuando se busca minimizar el precio unitario de un equipo.

Una vez validado el carro de compras se puede proceder a realizar el pago, indicando la información administrativa necesaria, como la dirección e identificación tributaria.

5.2. Orden de Fabricación de PCB

Para solicitar a un proveedor la fabricación de las placas diseñadas se requiere exportar los archivos Gerber (el detalle de los documentos exportados se encuentra en el anexo B). Los archivos gerber son planos por capas de fabricación, separando los tipos de trabajo, como las capas de cobre superior e inferior, capas de serigrafía, plano de perforaciones y vías, planos de máscaras de soldadura, entre otras. Estos archivos gerbers se han de cargar juntos dentro de un fichero comprimido (normalmente con extensión *.zip) en el sitio web del proveedor. En este proyecto se escogió al proveedor JLCPCB ya que corresponden al mismo grupo de empresas que desarrolló y puso a libre disposición el software EasyEDA; de este modo se espera una alta compatibilidad de archivos y no recibir observaciones del equipo técnico de fabricación por incompatibilidad de formatos en alguna o varias capas.

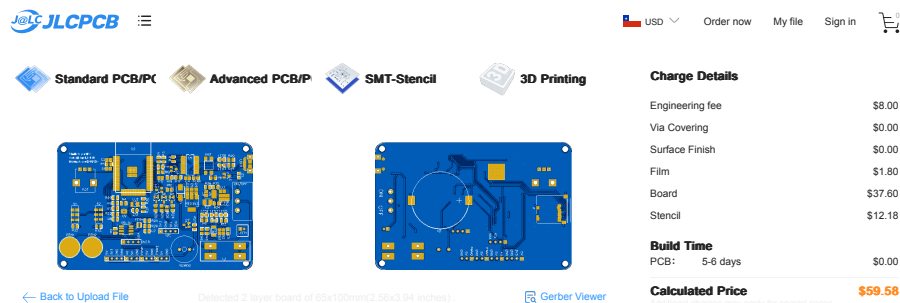


Figura 5.1: Cotización de fabricación de 125 placas PCB realizada en el sitio web del proveedor JLCPCB. Se vista completa de la solicitud de fabricación se encuentra disponible al final del documento en los anexos.

A continuación se describen las principales definiciones requeridas al momento de solicitar la fabricación de la placas PCB:

- Material del dieléctrico entre capas conductoras o sustrato de aislamiento. En este proyecto se seleccionó el material FR-4, designado así por la propiedad piroretardante del material y al laminado de epoxi reforzado

de vidrio tejido tipo 4 del sustrato [25]. Para circuitos de propósito general este el material de fabricación es el elegido por defecto ya que su precio no es elevado y tiene una alta resistencia mecánica entre otros beneficios técnicos [26]; sin embargo, se debe evaluar el uso de otros materiales cuando la PCB se diseña para estar expuesta a altas temperaturas o trabajar altas frecuencias (1GHz o superior) [27].

- La cantidad de capas, que en este caso corresponde a las dos caras visibles superior e inferior, sin capas intermedias.
- Las dimensiones de largo y ancho del modelo 100[mm]×65[mm]. Observación: Al momento de diseñar el plano de capas se realizó un ejercicio de cotización variando las dimensiones de manera iterativa para determinar un el mayor área posible sin que el precio por unidad aumentara en un escalón en la tabla de precios.
- Cantidad de placas a fabricar. Se seleccionan 125 unidades, 13 de margen de seguridad adicional que los lotes de componentes. Esto dado que una placa se puede estropear durante el montaje y se debe descartar, pero se pueden rescatar las componentes de mayor valor para reinstalarlas en otra unidad.
- Peso de cobre en capas superficiales. Este parámetro describe la masa de cobre en onzas por pie cuadrado de superficie, cuanto más cobre menos resistividad (DC) tendrá el plano para conducir y distribuir potencia. Los valores típicos son 0.5[Oz] y 1[Oz]. Este es un factor a tener en cuenta ya que la potencia disipada por la resistencia del plano lo hace en forma de calor, aumentando la temperatura del dispositivo [28]. Para este diseño se seleccionó 0.5 [Oz].
- Terminado superficial. Hace referencia a las partes de cobre expuestas, como el área para montar componentes, puntos de medición, vías (si quedan expuestas) o perforaciones con conexión al plano de tierra. La forma más popular de acaba se conoce como HASL, que viene del inglés *Hot Air Solder Leveling* y consiste en cubrir el cobre con soldadura mediante la inmersión de la placa en soldadura fundida quitando el exceso con cuchillas de aire caliente [29]. Para este proyecto se utilizó HASL libre de plomo. Una opción de acabado superficial muy usada en la industria es la denominada ENIG, del inglés *Electroless Nickel Immersion Gold* y considera la aplicación de una primera capa de Nickel sobre el cobre y una terminación en oro, que ayuda a prevenir la oxidación durante el tiempo de almacenamiento de la PCB y otorga una superficie plana, no obstante es una opción que encarece el producto y no presenta buenos resultados si

se necesita hacer un re-trabajo (como cambiar una componente y volver a soldar sobre esta terminación) [30].

- Cobertura de vías. Consiste en determinar si las vías se dejan expuestas o se aíslan con máscara anti-soldante, si los extremos se cubren para dejar una superficie plana soldable (en caso de no aislarse) y se se rellenan para aumentar conductividad eléctrica y térmica (muy útil para aplicar en planos de disipación de calor). Para este proyecto se seleccionó la opción más simple y de bajo costo, sin relleno y con máscara anti-soldante.

5.3. Montaje

Una vez recibidos los materiales y las placas PCB, se procede a montar el dispositivo. En esta sección se describen los métodos utilizados, las herramientas empleadas y los hallazgos encontrados tras el proceso.

5.3.1. Preparación

Recibidos los materiales, se realizó una revisión del listado de materiales, para corroborar modelos, cantidades y empaquetados. Esta revisión permite anticipar y mitigar riesgos antes de comenzar con el montaje.



Figura 5.2: Formato de entrega de las componentes del proveedor Digikey de fondo, mientras se sostiene placa PCB virgen en primer plano.

Inspección visual de placa PCB

Antes de iniciar el montaje o ensamblado, se realiza una inspección visual exhaustiva para detectar cualquier defecto de fabricación que pueda implicar un mal funcionamiento o ponga en riesgo la integridad del equipo o de quien realiza el trabajo. Este trabajo se realiza con una primer inspección general y

10. Soldadura en pasta.
11. Soldadura de estaño en rollo libre de plomo.
12. Pasta de soldar.

Los equipos e insumos se costearon dentro del precio de montaje en el costo directo.

5.3.2. Métodos de montaje

Montaje Manual

En este proyecto se utilizaron dos métodos de montaje. El primer método utilizado consiste en un montaje manual componente a componente, utilizando una estación soldadora con cautín, pinza chip-tool y pistola de calor. Primero se deposita soldadura en pasta en los pads de manera manual con una jeringa, a continuación se posiciona la componente en la placa y finalmente se aplica calor con la herramienta correspondiente. Este método sirve para realizar la prueba de validación eléctrica y funcional del dispositivo, pero no se recomienda para producto final, ya que la soldadura debe seguir una curva de temperatura específica para conseguir resultados duraderos.

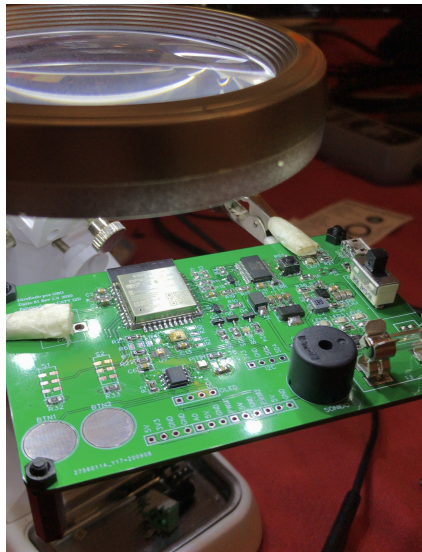


Figura 5.4: Montaje manual de componentes con cautín, pinzas y ayudante.

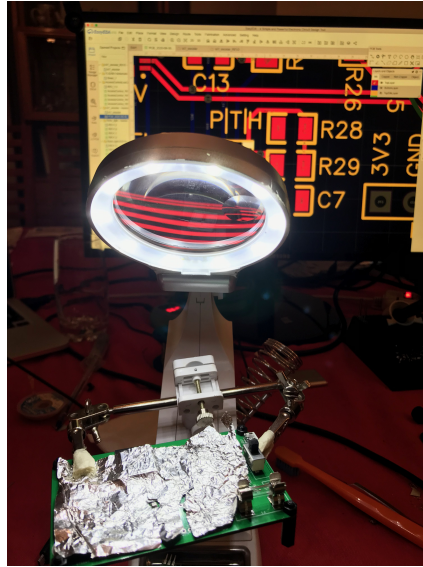


Figura 5.5: Montaje de sensor BME280 empleando pistola de calor y pantalla de aluminio para focalizar proteger el resto del circuito al flujo de aire caliente.

Montaje Semi Manual con plantilla y horno

El segundo método de montaje utilizado considera el uso de una plantilla metálica para aplicar soldadura en pasta a todos los puntos de montaje superficial de la placa de una sola vez, a continuación posicionar manualmente las componentes con pinzas sobre la PCB, para luego aplicar calor con un horno especializado, seleccionando una curva de temperatura que se ajuste a las definiciones de las componentes principales.

5.4. Costos Directos del Producto

Los costos directos de un producto corresponden al costo de materiales más la mano de obra. De acuerdo al alcance de este proyecto, no se consideran costos indirectos ni gastos generales de una organización que quisiera fabricar este producto, ya que éstos dependen en gran medida del tamaño de la organización, su ubicación y otros factores como convenio con proveedores o la posibilidad de reutilización de parte de una cadena de valor existente.

El costo de materiales fue de 3800USD brutos al realizar la compras en volumen para 100 unidades, considerando márgenes para merma. Esto incluye componentes, fabricación de PCB, costos de envío, honorarios e impuestos de aduana. El segundo factor del costo directo, el costo de montaje, se cotizó en \$1,500,000 CLP Netos para las 100 unidades entregadas. En este caso, se considera la tercerización del montaje como servicio, que incluye tanto la mano

de obra como el uso de infraestructura e insumos, tales como el horno, la soldadura y las otras herramientas descritas en la sección anterior.

Dados los montos indicados, el costo de directo de las 100 unidades asciende a \$4,521,000*CLP*¹ por lote, equivalente a un costo directo unitario de \$45,521.

¹A un precio del dólar referencial de 795 pesos chilenos con fecha 3 de abril de 2023

DESARROLLO DE FIRMWARE

En este capítulo se identifican y describen los módulos de software desarrollados y sus lineamientos de diseño. Se divide en cuatro partes, primero se presenta la configuración del ambiente de desarrollo, a continuación se plantea el diseño lógico de alto nivel del flujo principal, para luego describir los módulos más pequeños que permiten mejorar el el orden, control y entendimiento del código. La cuarta parte lista las librerías utilizadas.

Inicialmente se consideró desarrollar códigos de ejemplo para las diferentes experiencias de aprendizaje pivotaes. Sin embargo, en el ánimo de simplificar aún más los primeros pasos al usuario, se optó por hacer un sólo código que opere como firmware general del dispositivo. Este permite realizar las diferentes experiencias con la única necesidad de modificar cuatro parámetros en el primer uso: el nombre de la red WiFi y su contraseña, el nombre del dispositivo en Ubidots para diferenciarlo en el tópico MQTT y el token entregado por la plataforma para interactuar con su API.

6.1. Configuración del ambiente de desarrollo

El entorno de desarrollo utilizado para desarrollar el código corresponde al Arduino IDE versión 1.8.12 para macOS. Se debió instalar el paquete Arduino-ESP Core para poder compilar el código y probar su correcto funcionamiento, el cual tiene mantenimiento oficial por Espressfi Systems [31]. Observe a continuación la captura de pantalla del gestor de tarjetas de Arduino IDE en la que se indica el proveedor y versión del Arduino-ESP32 Core.



Figura 6.1: Instalación de compilador para tarjetas de desarrollo ESP32 disponibles

Junto con el compilador, se descarga una herramienta en python llamada ESPTOOL.PY la cual permite gestionar la comunicación y descarga del código compilado. El siguiente paso consiste en seleccionar cuatro parámetros para establecer el compilador y operar la comunicación entre el computador y el dispositivo. Estos se describen a continuación y se observan en la captura de pantalla de la figura 6.2.

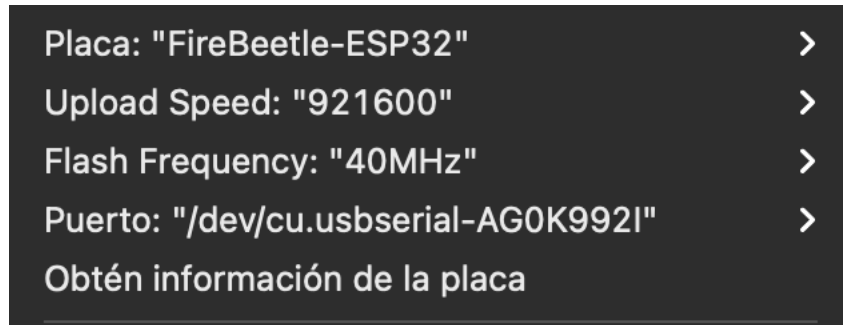


Figura 6.2: Configuración del ambiente Arduino IDE para el desarrollo del firmware en el menú de herramientas

El primer campo corresponde a la selección de una tarjeta específica para determinar el compilador, ya que el paquete Arduino-ESP trae consigo las consideraciones de diseño de hardware de una gran cantidad de tarjetas de desarrollo específicas en las que se usa este microcontrolador, como la DevKit Rev1, ESP32 Pico D4, etc. En este caso se utiliza el modelo FireBeetle, ya que esta tarjeta de desarrollo de DFRobot es muy sencilla y expone todos los pins directamente al breakout, lo que permite usarla de manera homóloga como si fuera el microcontrolador sin hardware adicional. Los siguientes pasos a configurar son el baudrate de carga y la frecuencia de trabajo de la memoria Flash, que en el caso de la ESP32 WROOM32D corresponde a 40[Mhz]. Finalmente se debe seleccionar el puerto COM donde se encuentra conectado el dispositivo.

6.2. Diseño del flujo principal

Usar el Arduino IDE supone utilizar dos métodos principales, `void setup()` y `void loop()`. El compilador garantiza que el método `setup` se ejecute solo una vez al inicio del programa y que el método `loop` se ejecute de forma periódica a continuación del primero. Por esta razón, en el método `setup` se configuran periféricos, como el puerto serial o puertos PWM y se instancian objetos de las librerías a utilizar, como el cliente WiFi. Por su parte en el `loop` se implementa la lógica de operación del firmware.

Como tarea base, el firmware debe tomar lecturas del ambiente usando los sensores embebidos y enviarlas a Ubidots, así como consultar por actualizaciones de los parámetros para el control de actuadores. Para esto se utiliza el protocolo de comunicación MQTT gracias a la librería PubSub ¹; el protocolo opera en la capa de aplicación sobre TCP y se debe tener el cuidado de mantener viva la comunicación realizando refrescos del canal permanentemente para que no expire. El firmware debe confirmar que la conexión se mantenga abierta para operar y en caso que se cierre, re-establecerla.

6.3. Diseño por Módulos

Para mantener el orden se divide el código en funciones o módulos. A continuación se presentan los módulos implementados:

6.3.1. Setup

Al iniciar el sistema, en el método `setup` se realiza la configuración de:

- Modo de lectura o escritura de pins en la ESP, especificando el número de GPIO correspondiente. En concreto pins de lectura de switches y pulsadores, y escritura para el control de LEDs y chicharra.
- Inicialización de osciladores para señales PWM para el control de intensidad de LEDs y chicharra.
- Inicialización de buses de comunicación I2C y Serial. El primero para lectura de sensores y control del OLED, el segundo para gestionar comunicación al Monitor Serial del IDE, hacer debug y realimentar al usuario en caso de fallas.
- Inicializar sensores BME y TSL, pantalla OLED y el cliente WiFi.

¹<https://www.arduino.cc/reference/en/libraries/pubsubclient/>

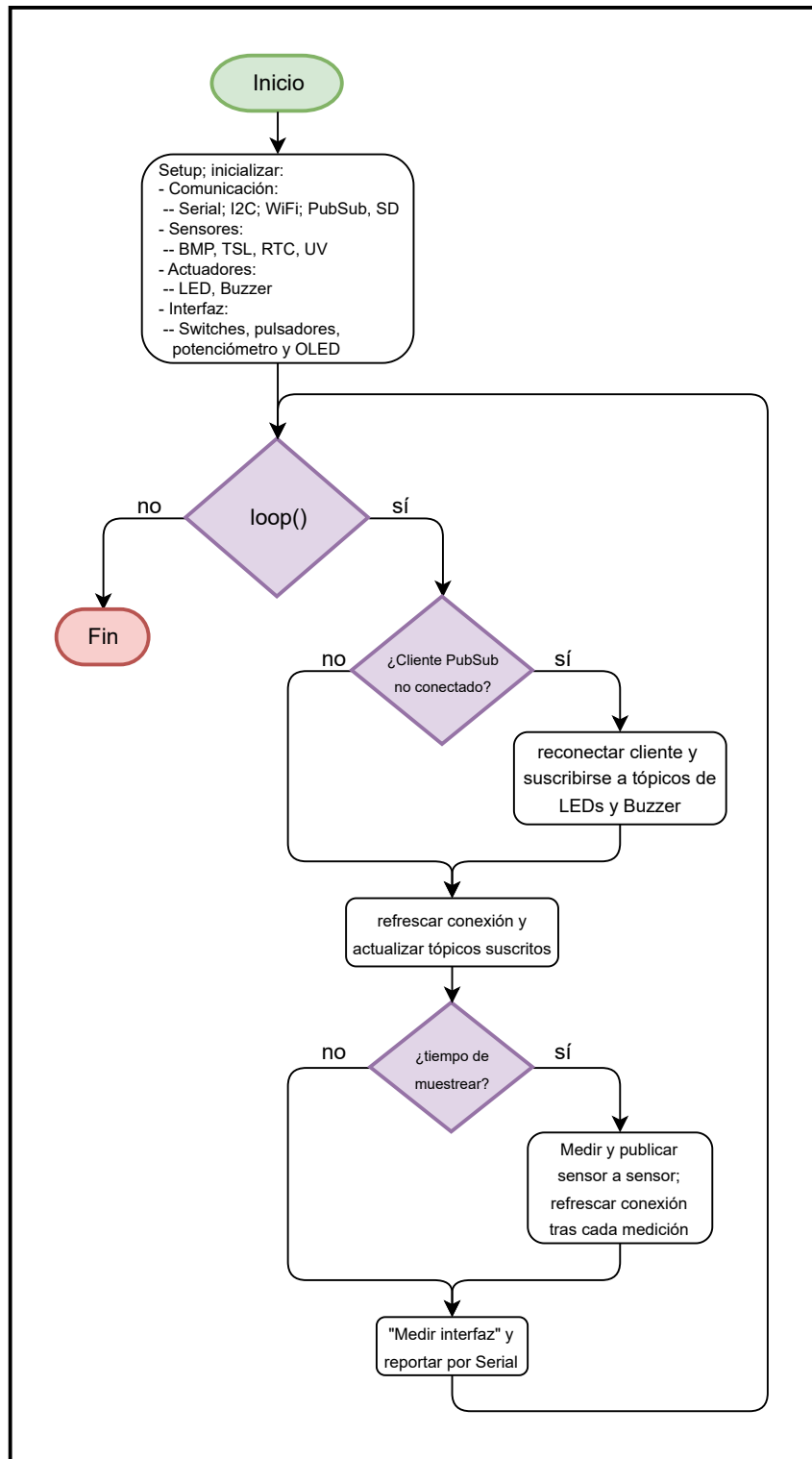


Figura 6.3: Diagrama de flujo de alto nivel que describe el flujo del código principal del firmware desarrollado

6.3.2. Loop

Una vez realizadas las inicializaciones y configuraciones iniciales, el ciclo loop parte verificando que la comunicación MQTT con el servidor de Ubidots esté viva. En la primera ejecución se espera que no lo esté, ya que en el setup solo se realizó la conexión al router mediante el cliente WiFi. Por tanto se llama al método `reconnect()`, el cual establece la conexión al servidor. El siguiente paso consiste en refrescar la conexión y actualizar los valores de actuación disponibles desde el servidor.

A continuación se verifica que haya transcurrido el tiempo necesario para volver a muestrear. Se define un periodo de muestreo de 15 segundos ya que la capa gratuita de Ubidots tiene un límite transacción de datos, y con este periodo se puede trabajar toda una clase de 90 minutos sin inconvenientes. En caso de que sea tiempo de medir, se operan los sensores y se publican los datos en los tópicos correspondientes a cada variable. Esto permite visualizarlos posteriormente en el dashboard en Ubidots

6.3.3. Reconnect

Se encarga de establecer la comunicación con la API de Ubidots, según la IP y token definidos en las macros que el usuario debe completar. Junto con abrir el canal de comunicación, el dispositivo debe suscribirse a los tópicos de interés, que en este caso corresponden a los valores de control de actuadores definidos por el usuario en la interfaz de Ubidots, o bien por una regla automática configurada por el mismo usuario en la plataforma. De este modo, cada vez que verifique la conexión, el servidor informará si hay nuevos valores publicados en los tópicos suscritos. En caso de haberlos se llamará a la función `callback()`.

6.3.4. Callback

Este método decodifica los valores recibidos, desarmando el mensaje, de modo de obtener solo el valor de interés. Transforma el dato original tipo string a valor numérico, actualiza la variable asociada correspondiente y lo implementa en el actuador que haya sido actualizado.

6.3.5. Beeping

Este método permite controlar el periférico PWM para ajustar la frecuencia del sonido, la intensidad sonora y el tiempo de duración del sonido a emitir por el buzzer. Se estos tres factores se requieren como parámetros de entrada, junto con el pin del buzzer

6.3.6. Módulos de iniciación de hardware

Para legibilidad del código se implementaron métodos globales sencillos con la inicialización requerida en cada caso. A continuación se listan los nombres de los métodos void respectivos.

- la pantalla OLED, `OLED_init()`
- el cliente WiFi, `WIFI_init()`
- el sensor de temperatura y presión, `BMP_init()`
- el sensor luz, `TSL_init()`
- el reloj de tiempo real, `RTC_init()`

6.4. Bibliotecas utilizadas

Una biblioteca de programación, también llamadas comúnmente librerías por su equivalente anglosajona, *Library*, es un conjunto de código predefinido que proporciona una funcionalidad específica para ser utilizada por otros programas. Son muy utilizadas por desarrolladores para ahorrar tiempo y esfuerzo, ya que en lugar de tener que escribir todo el código desde cero para cada nueva aplicación o proyecto, se pueden utilizar bibliotecas existentes para realizar tareas comunes.

En la siguiente tabla se presentan las bibliotecas utilizadas en este proyecto.

nº	Nombre	Propietario	Uso
1	iot-kit	Desarrollo Propio	Define Macros generales como nombre de pins de la ESP y direcciones I2C de los sensores utilizados. Define métodos para el control del hardware como el sensor de luz y el RTC.
2	WiFi.h	Arduino	Pertiene instanciar y gestionar un cliente WiFi.
3	PubSubClient.h	Github Nicholas O'Leary	Mediante un cliente WiFi permite gestionar comunicación en capa de aplicación con protocolo MQTT.
4	SPI.h	Arduino	Permite gestionar el periférico de comunicación SPI.
5	Metro.h	Github Thomas Ouellet Fredericks	Facilita el control de flujo de programa para eventos recurrentes con periodicidad temporal.
6	Adafruit_BMP280.h	Adafruit	Facilita la comunicación I2C con el sensor de temperatura y presión según especificaciones del fabricante.
7	Adafruit_TSL2561_U.h	Adafruit	Facilita la comunicación I2C con el sensor de luz según especificaciones del fabricante
8	Adafruit_Sensor.h	Adafruit	Es una biblioteca que estandariza el uso de sensores para minimizar el impacto al cambiar de módulos. Facilita el uso unidades en Sistema Internacional de los sensores.
9	Adafruit_GFX.h	Adafruit	Es un Driver para el control de la pantalla OLED.
10	Adafruit_SSD1306.h	Adafruit	Facilita la gestión del módulo para leer y escribir en una tarjeta SD.
11	RTCLib.h	Adafruit Industries	Permite iniciar y leer fácilmente módulos de Reloj de Tiempo Real.
12	Servo.h	Arduino	Facilita el comando de los periféricos PWM para controlar servomotores.

Cuadro 6.1: Bibliotecas de utilizadas en el desarrollo del Firmware de este proyecto.

PRUEBAS EXPERIMENTALES

En este séptimo y último capítulo, se presentan las principales pruebas del funcionamiento empírico del producto desarrollado.

En primera instancia se ejecutaron pruebas eléctricas, como mediciones de conductividad y niveles de tensión; en segundo lugar se realizan pruebas funcionales, como la carga del firmware para probar el puente de comunicación y el funcionamiento del microcontrolador; y finalmente pruebas punta a punta, en las cuales se toma información del ambiente a través de sensores y se envía a Ubidots, así como se definen valores en la interfaz de esta plataforma y se reciben conformes en el dispositivo mediante el uso de una conexión MQTT.

7.1. Pruebas eléctricas

En esta sección se presenta una tabla resumida con valores de medición de tensión en diferentes puntos de los módulos de alimentación: estos son el voltaje de entrada al sistema, es decir, el voltaje de alimentación observado en el conector microusb; el voltaje a la salida del boost de 5[V], el voltaje de salida del regulador de 3,3[V]; el voltaje de entrada a la ESP32 (que en teoría el es el mismo que el anterior). Observe la siguiente tabla 7.1.

nº	Descripción	Valor Teórico [V]	Valor Experimental [V]	Error[V]	Error [%]
1	Voltaje de entrada USB	5.0	5.06	0.06	1.2
2	V_{s1} regulador 5V	5.0	4.52	0.48	9.6
3	V_{s2} regulador 3V3	3.3	3.28	0.02	0.6
4	V_{in} ESP32	3.3	3.28	0.02	0.6

Cuadro 7.1: Tabulación de mediciones eléctricas realizadas comparadas con valores esperados

Las principales observaciones levantadas son las siguientes:

- Se logra que la etapa de alimentación logre un punto de operación que permite el funcionamiento del equipo.
- El voltaje de salida del boost de 5[V] se esperaba presentase el voltaje teórico incluso teniendo un voltaje de entrada igual o menor a los 5[V], cuestión que no ocurre y merece un análisis detallado para plantear una mejora en el trabajo futuro. No obstante el error observado en la medición supera el valor esperado, llegando a ser casi del 10 %, el equipo no presentó mal funcionamiento lógico en el puente de comunicación Serial-USB, punto que se tratará más adelante en el documento.
- El voltaje de salida del regulador de 3,3[V] entrega el voltaje esperado con un error menor al 1 %, lo cual permite la correcta operación de los equipos aguas abajo en el circuito, especialmente del microcontrolador ESP32 y los sensores BME280 y TSL.

7.2. Pruebas funcionales

En esta sección se presentan evidencias de la ejecución de pruebas funcionales que validan el correcto funcionamiento del hardware y firmware implementado, junto con el cumplimiento de los requerimientos funcionales.

Prueba de compilación del firmware

Se compiló el código desarrollado usando Arduino IDE para el dispositivo FireBeetleESP32. Una vez terminado el proceso se obtuvo el siguiente mensaje en la consola:

```
El Sketch usa 757910 bytes (57%) del espacio de almacenamiento de programa. El máximo es 1310720 bytes.
Las variables Globales usan 39828 bytes (12%) de la memoria dinámica, dejando 287852 bytes para las variables locales.
El máximo es 327680 bytes.
```

Se observa que el firmware ocupa poco más de la mitad de memoria de programa, lo cual es un buen parámetro ya que denota que aún se dispone de memoria para incorporar nuevas funcionalidades. Además la ausencia de errores en la compilación del código indica que las directivas para el control de periféricos involucrados en los requerimientos (I2C, PWM, ADC, WiFi) existen en las librerías utilizadas.

Comunicación USB-Serial entre computador y ESP32 mediante circuito integrado FTDI.

En esta prueba se descarga el código compilado del firmware para probar el envío de datos desde el computador al dispositivo. También, se abre el monitor del puerto Serial para observar que se reciban mensajes de acuerdo con el código compilado.

Con el dispositivo conectado al computador mediante un cable USB, se verificó que el dispositivo FTDI es reconocido por el gestor de puertos del equipo de origen. Posteriormente se ejecuta la rutina de descarga del código compilado desde la interfaz Arduino IDE, y se abre el monitor serial a 115200 baudios, mismo parámetro que el indicado en el código para inicializar la comunicación en el periférico del microcontrolador. Observer la figura 7.1, se reciben los mensajes esperados, lo que denota la correcta recepción del firmware y la ejecución del método `setup()`.

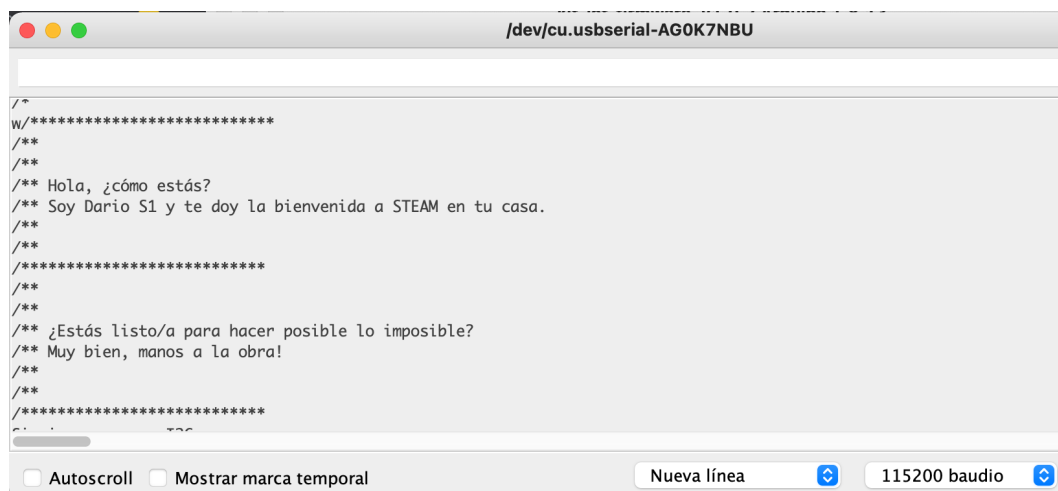


Figura 7.1: Captura de pantalla del monitor del puerto Serial con mensaje de saludo recibido desde Dario S1

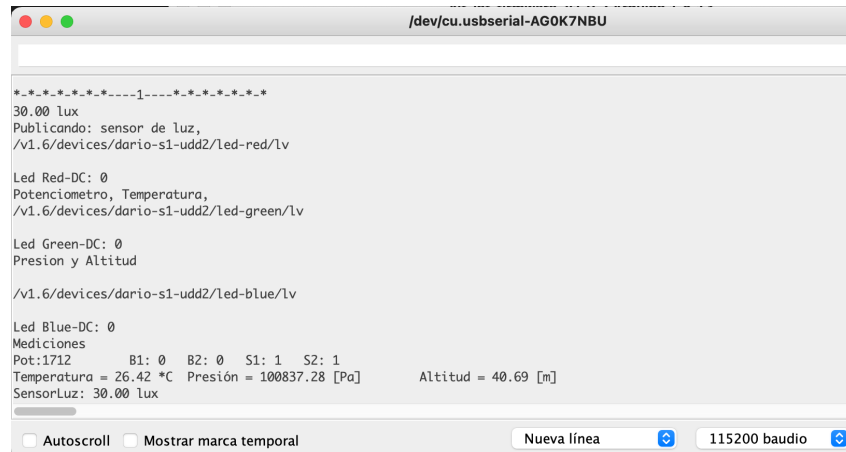


Figura 7.3: Captura de pantalla del monitor del puerto Serial con reporte de mediciones realizadas, tanto de variables ambientales como del estado de la interfaz de usuario (potenciómetro, interruptores y pulsadores).

La medición de altura (metros sobre el nivel del mar), no corresponde a la medición directa de dicha variable, si no a una estimación sencilla en base a la medición de la presión atmosférica.

7.3. Pruebas con Ubidots

En esta sección se presenta un ejemplo de Dashboard construido en Ubidots, a partir de la generación de componentes visuales como gráficos, tablas o elementos numéricos, a partir de los datos recibidos desde el dispositivo a través del protocolo MQTT. Para esto se crea un nuevo dispositivo en Ubidots mediante la interfaz gráfica, en la cual se define un nombre para el dispositivo y el sistema define un token, tipo llave única para que el dispositivo pueda publicar datos en el tópico: `PREFIJO_API/NOMBRE_DISPOSITIVO/NOMBRE_VARIABLE`.

Como se puede inferir del texto precedente, el siguiente paso es crear variables, para ello se necesita indicar al sistema el nombre de la variable y si será tipo *raw* (cruda, es decir el dato se almacena directo del dispositivo tal como llega) o sintética (en la cual el sistema tiene que aplicar un procesamiento a definir mediante una fórmula).

7.3.1. Valores de medición reportados

Observe el Dashboard implementado en Ubidots en la figura 7.4, éste presenta los datos recibidos durante los 66 minutos de monitoreo del experimento. En la parte superior se encuentran de izquierda a derecha:

- indicador visual de la medición del protenciómetro con relleno progresivo entre valor mínimo y máximo, 0 y 4095 que corresponden a los niveles de los 12 bits de resolución del conversor análogo digital, en el rango de voltaje 0 a 3.3[V] en que opera la ESP32.
- indicador visual de la medición de temperatura en grados Celsius con forma de termómetro.
- histograma con cinco clases de los valores de temperatura observados durante el periodo.
- valor numérico de la última medición de intensidad lumínica en lúmenes.
- histograma con cinco clases de los valores de luminosidad observados durante el periodo.

En la parte media e inferior izquierda se dispuso de dos gráficos de tendencia de las mediciones en el tiempo. El primero de ellos, que se encuentra a altura media, representa las curvas de temperatura (en rojo) y presión (azul) en el tiempo, en el cual se observa por un lado un aumento de 4 grados al inicio del registro (dado que a esa hora comienza a dar sol directo a la habitación donde se realizó el experimento) y por otra parte se observa una tendencia a la baja del valor de presión atmosférica, pero a nivel de segunda cifra significativa, lo que se atribuye a la naturaleza cíclica de los procesos atmosféricos y no al

desplazamiento del equipo en el espacio, ya que este no se movió durante el experimento. Esto muestra la calidad y precisión del sensor BMP280 de Bosch implementado en el diseño. El segundo gráfico de tendencia, ubicado en la parte inferior izquierda, representa la curva de

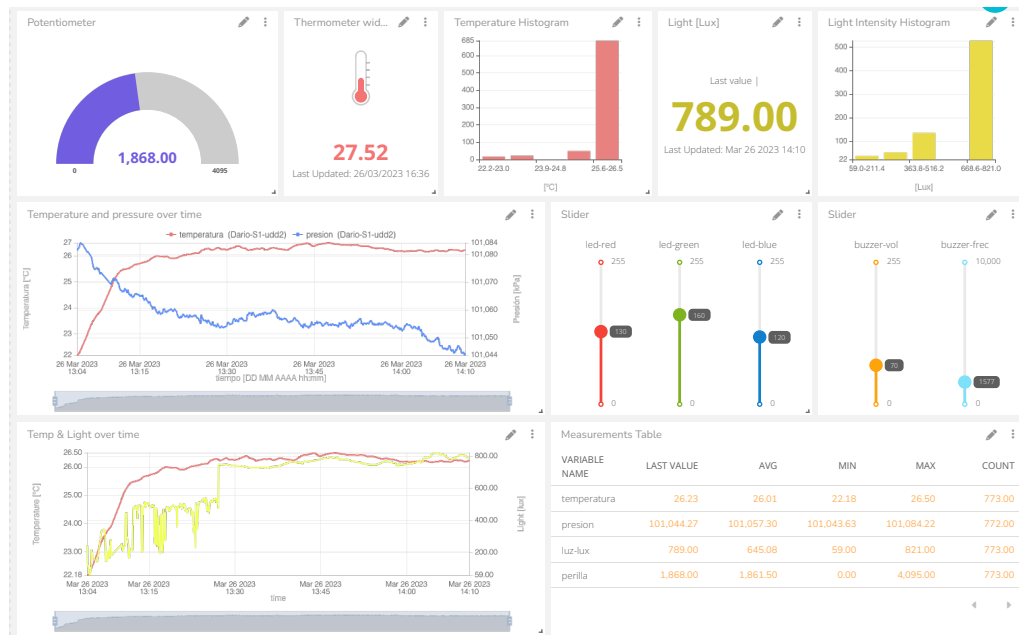


Figura 7.4: Dashboard implementado en Ubidots.

7.3.2. Recibir valores de actuación

En el dashboard de la figura 7.4 se observan cinco deslizadores. Cada uno determina un valor numérico entre un rango predefinido y está asociado a una variable del dispositivo. Al recorrer los deslizadores de izquierda a derecha se tiene que los tres primeros corresponden al control de intensidades de los LEDs rojo, verde y azul, en un rango de 0 a 255 (ya que el ciclo de trabajo de la PWM que los controla tiene como parámetro un número de 1 byte), y los dos últimos corresponden al volumen (también entre 0 y 255) y la frecuencia del sonido a emitir por la chicharra. El valor del deslizador de frecuencia se configuró para moverse en un rango de 0 a 10.000, ya que la frecuencia máxima definida por el fabricante de la chicharra indica 10kHz como frecuencia máxima.

Ubidots se encarga de publicar el nuevo valor definido por el deslizador en el tópico MQTT asociado a la variable correspondiente. Por otra parte, el firmware determina que el dispositivo al iniciar la conexión MQTT se suscriba a los tópicos asociados a las cinco variables del LED RGB y la chicharra, y se encarga también de revisar si hay nuevos valores publicados en alguno de los

tópicos a los que está suscrito cada vez que se publica una nueva medición. Si recibe un valor nuevo, el firmware modifica el registro necesario para que se vea reflejado el valor en los actuadores.

Para comprobar el correcto funcionamiento de la recepción de valores de actuación se modifica la posición de los deslizadores en el Dashboard esperando vez modificaciones coherentes en los actuadores del dispositivo. Tras operar los deslizadores uno a uno, separados por intervalos de 5 segundos, como modificaciones consecutivas (tan rápido como la operación del cursor permite a un usuario) se observan las modificaciones esperadas en el dispositivo, tras un rango de entre 1 y 5 segundos de tiempo transcurrido entre la modificación y la observación del efecto en los actuadores.

CONCLUSIONES Y TRABAJO FUTURO

7.4. Conclusiones

En este proyecto de título se logra diseñar e implementar un Producto Mínimo Viable de apoyo a la educación STEAM con foco en IoT, alcanzando los tres criterios definidos para el cumplimiento del objetivo dentro del alcance proyectado.

Se logra desarrollar un producto que permite facilitar la introducción al STEAM-IoT ya que contiene el hardware y firmware necesarios para que estudiantes puedan comenzar a realizar experiencias prácticas de aprendizaje con sólo energizar el dispositivo, sin necesidad de armar circuitos, ni desarrollar software en un inicio. Dentro de las prestaciones incluidas, se destacan: la conectividad WiFi y Bluetooth onboard, los sensores de variables ambientales por su amplio rango de operación, precisión, entrega de mediciones en unidades internacionales y ausencia de necesidad de calibración, la simplicidad para cambiar el firmware gracias al puerto USB-Serial, los actuadores LED RGB y chicharra y la interfaz de usuario.

El producto desarrollado cuenta con versatilidad funcional y no está limitado a las experiencias de aprendizaje pivotaes, ya que cuenta con GPIOs analógicos y digitales reservados, el bus I2C y acceso a las etapas de alimentación onboard disponibles desde el breakout del dispositivo. De este modo, se pueden conectar módulos externos y ampliar las posibilidades de experimentación. En cuanto al firmware, el dispositivo es reprogramable fácilmente desde el Arduino IDE y cuenta con una librería de desarrollo disponible.

El costo directo del producto se estimó en \$45,521CLP, lo cual cumple con creces el criterio de no superar los \$122,200CLP definidos inicialmente en el

alcance como cota superior para considerar el producto accesible. Es necesario enfatizar que este monto se logra realizando el proceso de adquisición y montaje por lotes de 100 unidades o más. Esto no significa que el valor indicado deba o pueda ser un precio de venta final, ya que corresponde a costo directo, es decir, materiales y mano de obra, pero no están incluidos otros costos de un penitencial negocio que ofrezca el producto, así como tampoco incluye el IVA de venta correspondiente.

El trabajo realizado consideró etapas de estudio de productos disponibles en el mercado, tecnologías de interés de implementación, definición de requerimientos y especificaciones técnicas, diseño de circuitos a nivel de planos esquemáticos y disposición por capas en PCB, selección y adquisición de componentes y ordenar la fabricación de la PCB en proveedores internacionales, montaje del dispositivo con dos procesos de soldadura, desarrollo de firmware y realización de pruebas experimentales para validar el diseño.

Si bien el alcance del trabajo de título llega al diseño e implementación, el producto realizado logró impactar en territorio nacional gracias a la colaboración con la fundación Ecoscience, ya que permitió realizar talleres STEAM-IoT a distancia en tiempos de pandemia. La fundación envió los dispositivos a las casa de los estudiantes impactando a 302 estudiantes de cursos de sexto, séptimo y octavo básico, primero, segundo y tercero medio y formando a 19 docentes y se realizaron 128 horas de clase en 2021.

7.5. Trabajo futuro

Si bien el dispositivo diseñado cumple con los parámetros de producto mínimo viable, durante el montaje y pruebas experimentales se identificaron puntos de mejora a considerar para un nuevo diseño. Para el diseño del hardware:

- Agregar al breakout pins para conectar directamente una protoboard de 400 u 800 puntos debajo del dispositivo, alimentando sus líneas verticales con las etapas de 3.3 y 5V.
- Quitar el sensor UV de la placa y dejar su pin disponible para conectarlo de manera externa al dispositivo. Esto debido a que la placa no se debe exponer al sol y el componente es difícil de soldar.
- Incluir un banco de condensadores para estabilizar mejor las etapas de alimentación, ya que al conectar servomotores o componentes que operen peaks de corriente, podrían afectar el correcto funcionamiento del dispositivo.

- Mejorar el diseño del selector automático de fuente de alimentación entre USB y batería.
- Cambiar el fusible por un CI, que sea automático, más pequeño y que no tenga la necesidad de ser reemplazado.
- Rotar 180 grados e invertir polaridad el potenciómetro para que quede en un sentido de giro intuitivo, es decir, girar la perilla en sentido horario hace crecer el valor medido.
- Aumentar a 10z la densidad de los planos de la PCB al momento de realizar la orden de fabricación, para intentar bajar la temperatura de la placa y no generar artefactos en sensor de temperatura onboard.

Para el diseño del firmware:

- Reescribir el firmware para facilitar la comprensión del mismo.
- Modificar el código para no usar el método `yield()` que no es recomendado, y controlar el flujo con un timer mediante el método `millis()`.
- Analizar el consumo energético de la activación del WiFi y sensores para determinar una frecuencia de actualización óptima entre consumo de energía, uso de datos y un muestro suficiente para las aplicaciones de interés.

Para mejorar el producto en el ámbito docente, se proyecta desarrollar nuevas experiencias de aprendizaje que conduzcan al estudiante a profundizar en los aspectos técnicos como la programación del firmware y la expansión circuital.

A. ESQUEMÁTICOS DEL DISPOSITIVO

En la figura 7.5 se presenta el plano esquemático completo del diseño. En las figuras sucesivas, desde la 7.6 a la 7.13 se muestran los módulos por separado; note que la numeración de las componentes cambia respecto del plano principal, ya que al generar nuevas instancias de cada componente el software las identifica con números incrementales (diferentes). La numeración que se puede observar en la serigrafía de la PCB corresponde a la indicada en el plano principal (7.5).

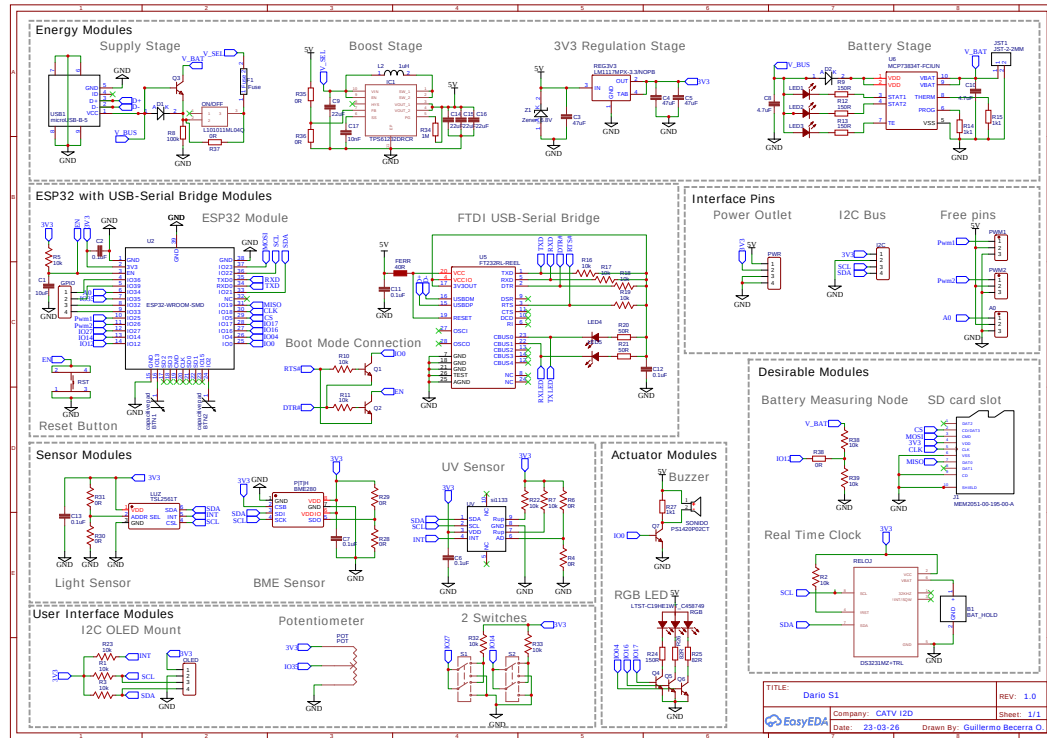


Figura 7.5: Plano esquemático global con todos los módulos del dispositivo.

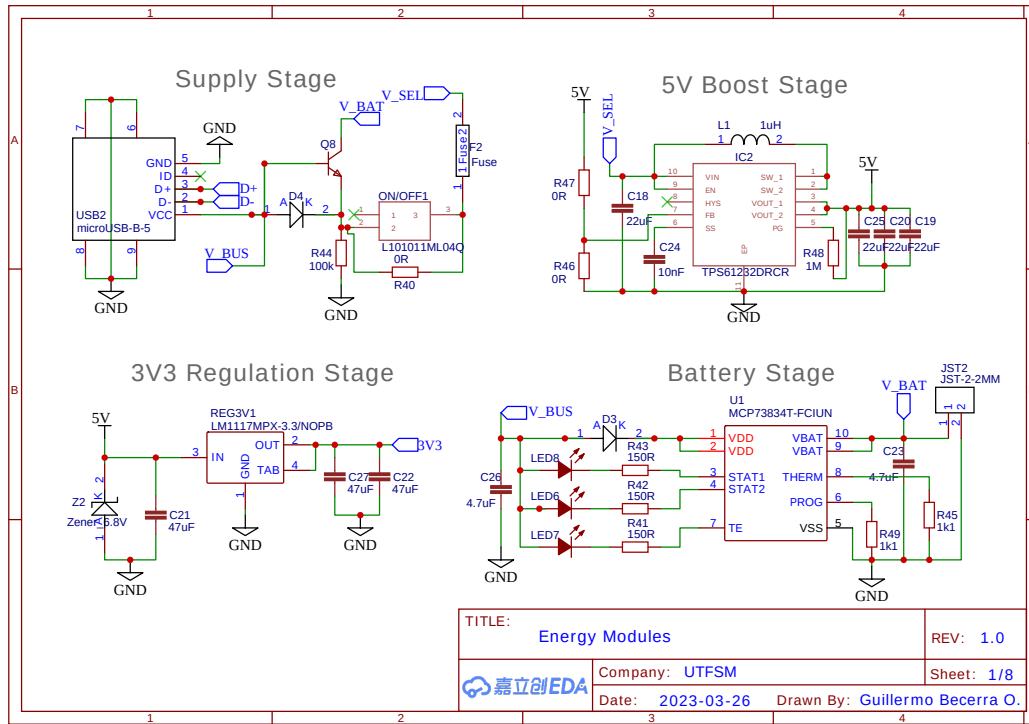


Figura 7.6: Módulos de alimentación del dispositivo.

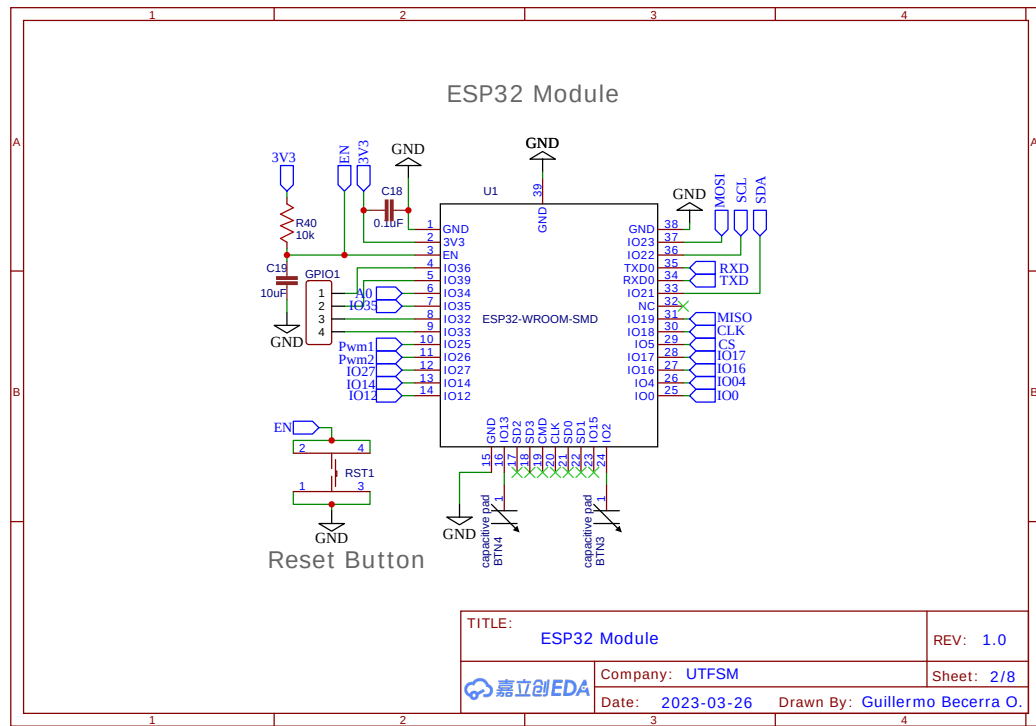


Figura 7.7: Módulo esquemático del microcontrolador ESP32-WROOM

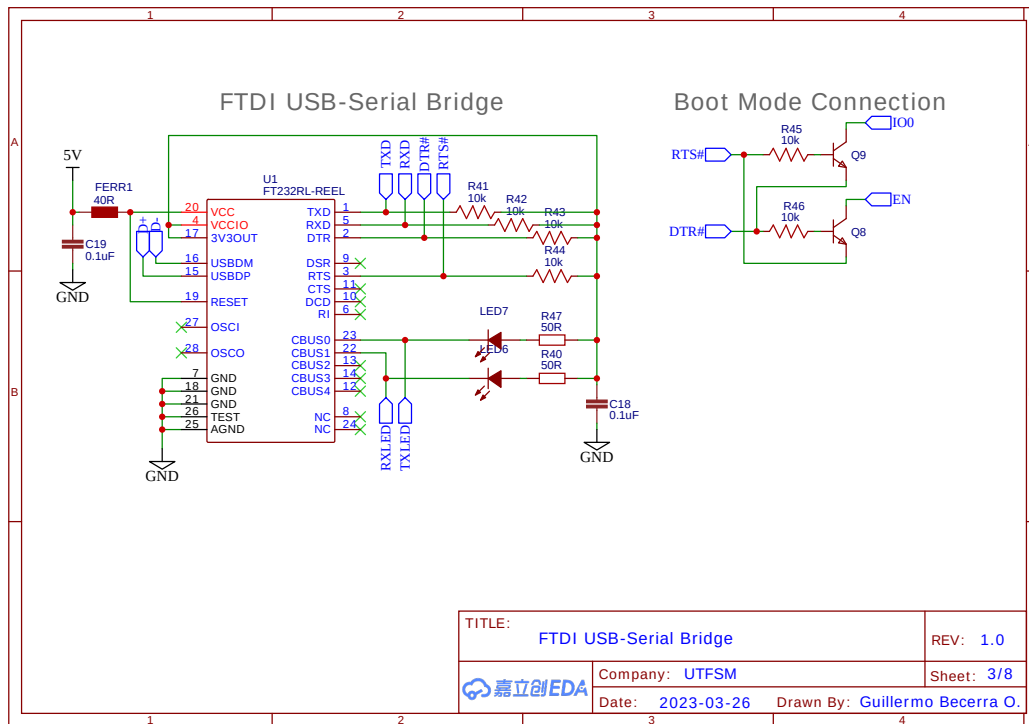


Figura 7.8: Módulo esquemático del puente de comunicación USB-Serial FTDI para comunicar el dispositivo a un computador mediante un cable USB.

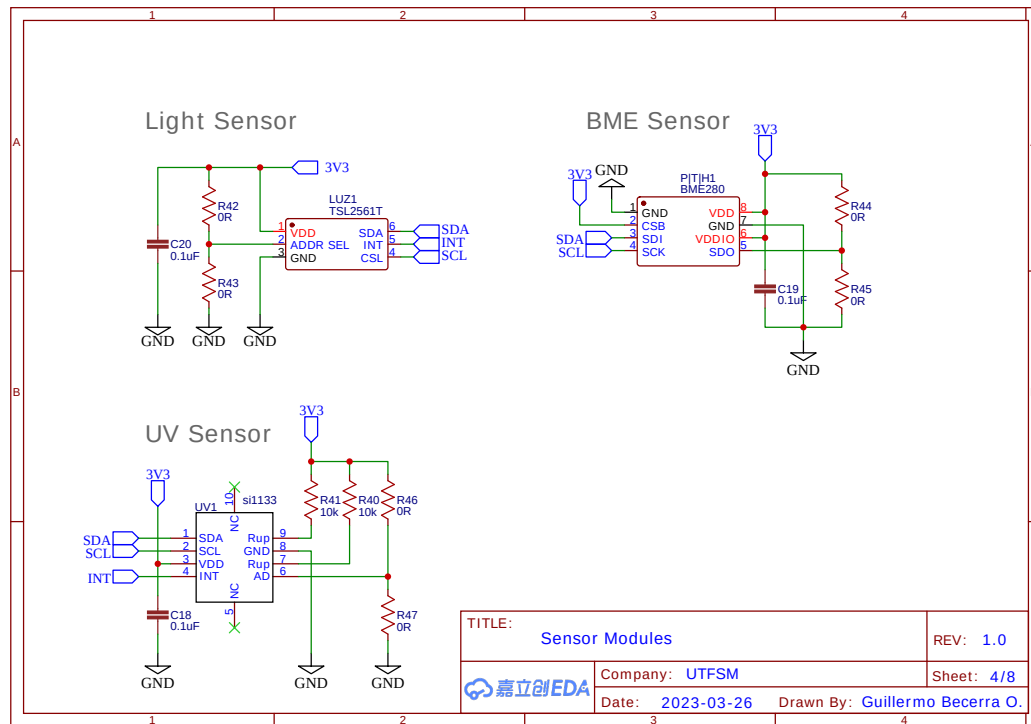


Figura 7.9: Módulos esquemáticos de los sensores para medir de luminosidad, presión, temperatura, humedad, y radiación UV.

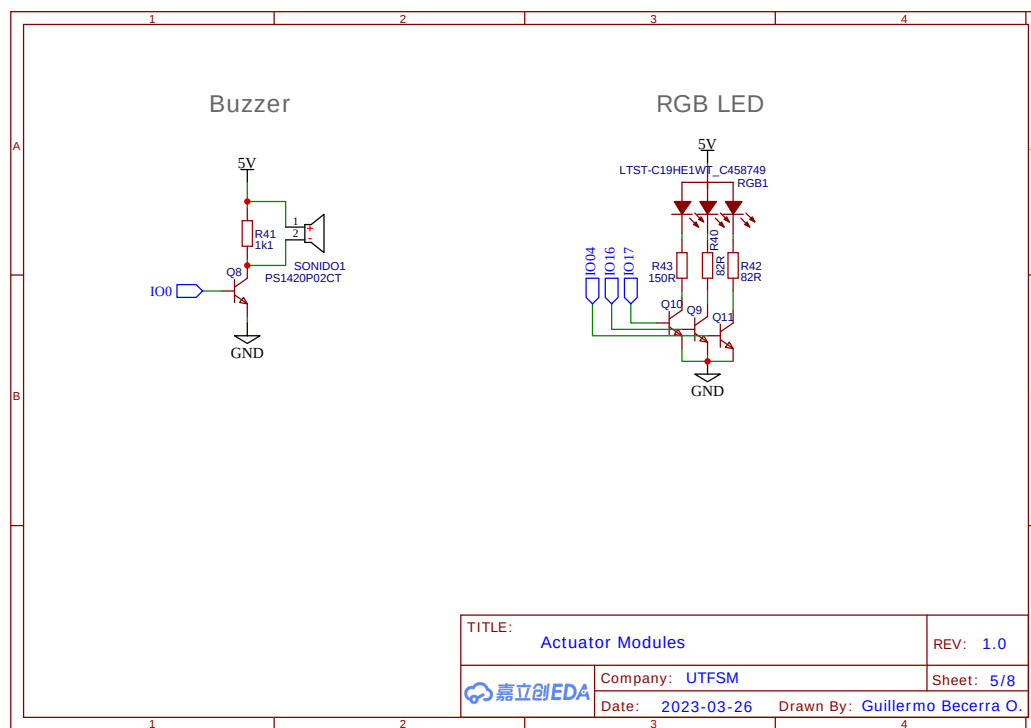


Figura 7.10: Módulos esquemáticos de los actuadores chicharra (buzzer) y LED RGB.

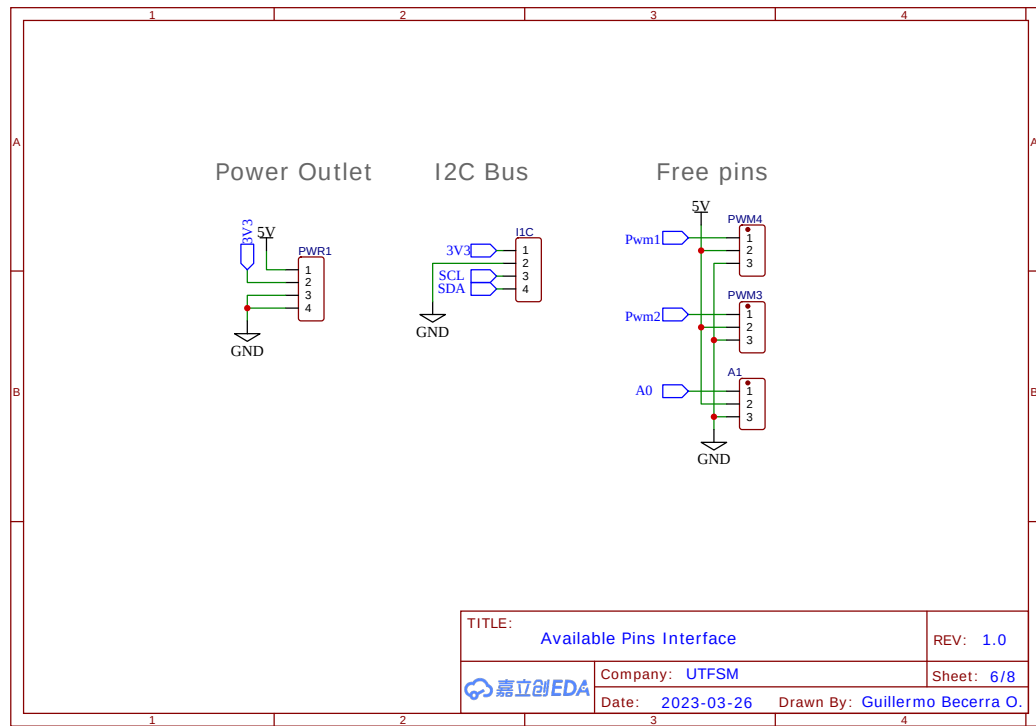


Figura 7.11: Módulos esquemáticos de los conectores para alimentación en 5V y 3.3V, el bus de comunicación I2C y los pins disponibles de la EPS con PWM GPIO25 y GPIO26, junto el un pin de lectura analógica GPIO34 (o A0 en el código).

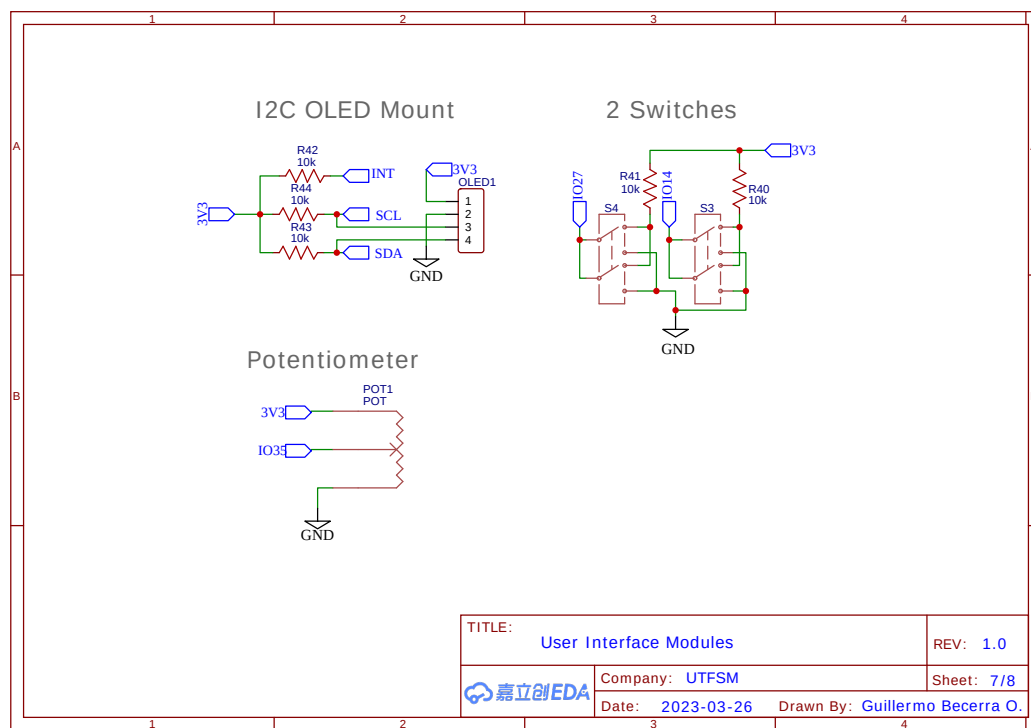


Figura 7.12: Módulos esquemáticos de los módulos de la interfaz de usuario del dispositivo: conexión de la pantalla OLED, interruptores y potenciómetro.

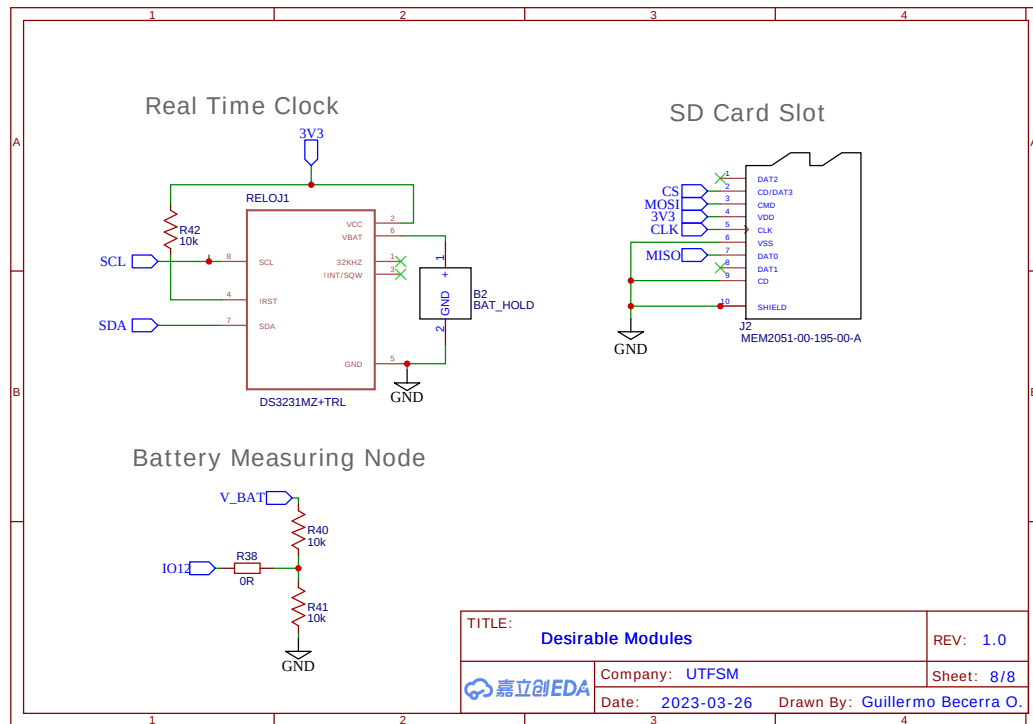


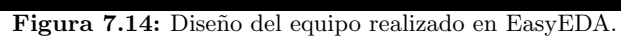
Figura 7.13: Módulos esquemáticos de módulos deseables (no esenciales): Reloj de tiempo real, puerto para tarjeta microSD y nodo de medición de voltaje de carga de la batería.

B. PLANOS DE FABRICACIÓN

A partir del del documento *Iot-kit-PCB.brd* que corresponde al diseño de circuito impreso realizado en EasyEDA (ver figura 7.14), se exportan los archivos Gerber cuyo nombre es el mismo que el documento origen, con las siguientes extensiones:

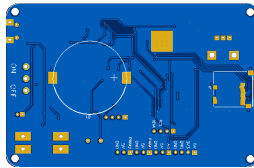
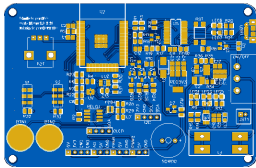
- DRL: Drill Through
- GKO: Board outline layer
- GBL: Bottom Layer
- GBP: Bottom Solder Mask
- GBO: Bottom Silkscreen Layer
- GBS: Bottom Solder Mask Layer
- GME: Mechanical Layer
- GTL: Top Layer
- GTP: Top PasteMask Layer
- GTO: Top Silkscreen Layer
- GTS: Top Solder Mask Layer

Estos archivos de fabricación se encuentran en la carpeta de Anexos.



C. ORDEN DE FABRICACIÓN

Standard PCB/PCB Advanced PCB/PCB SMT-Stencil 3D Printing



Back to Upload File

Detected 2 layer board of 65x100mm(2.56x3.94 inches)

Gerber Viewer

Base Material

FR-4 Aluminum Copper Core Rogers PTFE Teflon

Layers

1 2 4 Precision PCB 6 8 10 12 14 16 18 20

Dimensions

100 65 mm

PCB Qty

125

Product Type

Industrial/Consumer electronics Aerospace Medical

PCB Specifications

Different Design

1 2 3 4

Delivery Format

Single PCB Panel by Customer Panel by JLCPCB

PCB Thickness

0.4 0.6 0.8 1.0 1.2 1.6 2.0

PCB Color

Green Purple Red Yellow Blue White Black

Silkscreen

White

Surface Finish

HASL(with lead) LeadFree HASL ENIG

High-spec Options

Outer Copper Weight

1 oz 2 oz

Via Covering

Tented Untented Plugged

Confirm Production file

No Yes

Remove Order Number

No Yes Specify a location

Flying Probe Test

Random Test Fully Test

Gold Fingers

No Yes

Castellated Holes

No Yes

Charge Details

Engineering fee	\$8.00
Via Covering	\$0.00
Surface Finish	\$0.00
Film	\$1.80
Board	\$37.60
Stencil	\$12.18

Build Time

PCB:	5-6 days	\$0.00
------	----------	--------

Calculated Price

Additional Charges may apply for special cases \$59.58

SAVE TO CART

Shipping Estimate

\$188.18

DHL Express Worldwide	4-7 business days
Weight	4.31kg

Advanced Options

PCB Remark 



PCB Assembly



Free Assembly for your PCB order



Stencil

Order together with PCB

Framework	No <input checked="" type="button" value="Yes"/>
Step Stencil	<input checked="" type="button" value="No"/> Yes
Dimensions	470*370(Valid area 290mm*190mm, \$11.20 ,1.5kg) 
Stencil Side	<input checked="" type="button" value="Top+Bottom(On Single Stencil)"/> Top Bottom
	Top & Bottom(On Separate Stencil)
Stencil Qty	1
Polishing Process	<input checked="" type="button" value="Sanding"/> Etching Electropolishing
Fiducials	<input checked="" type="button" value="No Fiducial"/> Etched Through Etched Half into board
Confirm Production file	<input checked="" type="button" value="No"/> Yes
Engrave Text	No <input checked="" type="button" value="Yes"/>
Stencil Remark 	Dario S1 Rev 1.0 

D. LISTADO DE COMPONENTES: BOM COMPRADO



Review Order

Web ID 312554006 Access ID 18668 Salesorder ID 65806677
Submitted 9/10/2020 3:19 PM

Order Details

SHIPPING METHOD

DHL Express International

CUSTOMER NUMBER

12251562

EMAIL ADDRESS

GUILLERMO.BECERRA@C
ATV.CL

BACKORDER SHIPPING METHOD

DHL Express International

PAYMENT METHOD



SHIPPING ADDRESS

GUILLERMO BECERRA
CATV ELECTRONICA SPA
BALMACEDA 1830,
DEPTO 102
CERRO RECREO
VINA DEL MAR 20000-1
CHILE
Telephone: 56990995967

BILLING ADDRESS

GUILLERMO BECERRA
CATV ELECTRONICA SPA
ARLEGUI 333 OF217
VINA DEL MAR V 25714-48
CHILE

Invoice and Shipping Information

INVOICE STATUS

75879817 Shipped DHL Express 9/10/2020 Track - [Download](#) [Request](#)
International 4990344693 [Packlist](#) [Return/Report](#)
[Issue](#)

Cart 75879817

Cart

INDEX	QUANTITY	IMAGE	PART NUMBER	DESCRIPTION	CUSTOMER F
1	112		240-2389-1-ND	FERRITE BEAD 40 OHM 0805 1LN	FERR
2	112		311-1.00MCRCT-ND	RES 1M OHM 1% 1/8W 0805	R34
3	334		P1.10KCCT-ND	RES SMD 1.1K OHM 1% 1/8W 0805	R14,R15,R27
4	223		311-51.0CRCT-ND	RES 51 OHM 1% 1/8W 0805	R21,R20
5	223		A129736CT-ND	RES 82 OHM 1% 1/8W 0805	R25,R26
6	334		YAG3359CT-ND	RES SMD 100K OHM 1% 1/8W 0805	R37,R36,R8
7	112		P300KDACT-ND	RES 300K OHM 0.1% 1/8W 0805	R35
8	112		1904-1023-1-ND	RF TXRX MOD WIFI TRACE ANT SMD	U2
9	112		160-2162-1-ND	LED RGB DIFFUSED CHIP SMD	RGB
10	119		478-7802-1-ND	DIODE SCHOTTKY 40V 500MA 0805	D2,D1

11	8		175-DS3231MZ+TRLCT-ND	IC RTC CLK/CALENDAR RELOJ I2C 8SOIC	21	8		679-1877-ND	SWITCH SLIDE SPDT 3A 30V	ON/OFF	
12	112		PTS647SK38SMTR2LFSCT-ND	TACT 4.5 X 4.5, 3.8 MM H, 2.5N, RST	22	112		SRP4020TA-1R0MCT-ND	FIXED IND 1UH 5A 27 MOHM SMD	L2	
13	112		296-44498-1-ND	IC REG BOOST 5V 4A 10VSON	IC1	23	112		507-1248-ND	FUSE GLASS 1.6A 250VAC 5X20MM	F1
14	112		LM1117MPX-3.3/NOPBCT-ND	IC REG LIN 3.3V 800MA SOT223-4	REG3V3	24	112		987-1277-ND	POT 10K OHM 1/32W PLASTIC LINEAR	POT
15	334		DMN65D8L-7DICT-ND	MOSFET N-CH 60V 310MA SOT23	Q6,Q4,Q5	25	245		475-2488-1-ND	LED ORANGE DIFFUSED 0805 SMD	LED2,LED5,L1
16	119		DMG2302UK-7DICT-ND	MOSFET N-CH 20V 2.8A SOT23	Q7,Q3	26	668		478-1395-1-ND	CAP CER 0.1UF 50V X7R 0805	C13,C12,C11,1
17	223		MMBT5551LT3GOSCT-ND	TRANS NPN 160V 0.6A SOT23-3	Q1,Q2	27	223		1276-1065-1-ND	CAP CER 4.7UF 16V X5R 0805	C10,C8
18	8		MCP73834T-FCI/UNCT-ND	IC BATT CNTRL LI-ION 1CEL 10MSOP	U6	28	112		399-17889-1-ND	CAP CER 0805 10NF 100V X7R 10%	C17
19	112		768-1007-1-ND	IC USB FS SERIAL UART 28-SSOP	U5	29	112		1276-6456-1-ND	CAP CER 10UF 10V X5R 0805	C1
20	1,670		P10KACT-ND	RES SMD 10K OHM 5% 1/8W 0805	R16,R22,R33,1	30	112		609-4618-1-ND	CONN RCPT USB2.0 MICRO B SMD R/A	USB1

31	112		445-2526-1-ND	BUZZER PIEZO 5V 14MM TH	SONIDO	41	223		CKN10723DKR-ND	SWITCH SLIDE DPDT 300MA 6V	S1,S2						
32	126		S7002-ND	CONN HDR 4POS 0.1 TIN PCB	I2C,OLED,PW	42	668		P0.0ACT-ND	RES SMD 0 OHM JUMPER 1/8W 0805	R4,R28,R29,R						
33	112		TSL2561-TCT-ND	SENSOR OPT 640NM AMBIENT 6SMD	LUZ	43	334		1276-1852-1-ND	CAP CER 47UF 6.3V X5R 0805	C4,C5,C3						
34	8		455-1734-1-ND	CONN HEADER SMD 2POS 2MM	JST1	44	446		1276-1822-1-ND	CAP CER 22UF 10V X5R 0805	C16,C14,C15,						
35	112		336-3477-1-ND	SENSOR OPT 625NM/525NM AMB 10DFN	UV	45	223		486-1172-ND	FUSE CLIP CARTRIDGE 10A PCB	FUSECLIP						
36	112		828-1063-1-ND	SENSOR PRESSURE HUMIDITY TEMP	P T H							Subtotal \$2,869.67					
											Copy to clipboard	Download .csv					
37	112		SMBJ5342B-TPMSCT-ND	DIODE ZENER 6.8V 5W DO214AA	Z1	Invoice 75879817											
												INDEX	QUANTITY	CANCELLED	QUANTITY SHIPPED	PART NUMBER	DESCRIPTION
38	446		311-150CRCT-ND	RES 150 OHM 1% 1/8W 0805	R13,R9,R12,R	1	112	0	112	240-2389-1-ND	FERRITE BEAD 40 OHM 0805 1LN						
39	8		BU2032SM-JJ-GCT-ND	BATT HOLDER COIN 20MM 1 CELL SMD	B1	2	112	0	112	311-1.00MCRCT-ND	RES 1M OHM 1% 1/8W 0805						
40	8		2073-MEM2051-00-195-00- ACT-ND	MICRO SD PUSH PUSH, OPEN, SMT, 1	J1	3	334	0	334	P1.10KCCT-ND	RES SMD 1.1K OHM 1% 1/8W 0805						
						4	223	0	223	311-51.0CRCT-ND	RES 51 OHM 1% 1/8W 0805						

D. CÓDIGO FIRMWARE

7.6. Librería IoTkit.h

ΩΩimmediateía,

```
1 /**
2  * @file iotkit.h
3  * @version 1.2.0
4  * @date 03/2023
5  * @author GBO
6  * @title IotKit for educations
7  * @brief ELO308 shorter version
8  * @code
9  *   delay(500);
10 * @endcode
11 */
12 // Kit-IoT
13 #ifndef iotkit_h
14 #define iotkit_h
15
16 // ESP32: Pins to use
17 #define RED_LED 17
18 #define GREEN_LED 16
19 #define BLUE_LED 4
20 #define BUZZER 0
21 #define SERVO1 25
22 #define SERVO2 26
23 #define SWITCH1 27
24 #define SWITCH2 14
25 #define BUTTON1 13
26 #define BUTTON2 2
27 #define UV_SENSOR 34 //GENERIC ANALOG SENSOR
28 #define POT_SENSOR 35
29 #define SDA 21
30 #define SCL 22
31 #define MISO 19
32 #define MOSI 23
33 #define CLK 18
34 #define CS 5
35
36 // Flags
37 #define LOG_DEBUG 2
38 #define LOG_INFO 1
39 #define LOG_NONE 0
40 #define LOG LOG_DEBUG
```

```

41
42 //OLED I2C 128x64px SSDD1306 controller
43 #define SCREEN_WIDTH 128 // OLED display width, in pixels
44 #define SCREEN_HEIGHT 64 // OLED display height, in pixels
45
46 // I2C ADDRESSES
47 #define DS3231_I2C_ADDRESS1 0x68 //CAREFULL, repeated with HWS12 and GY-88
48 #define DS3231_I2C_ADDRESS2 0x57
49 #define OLED_I2C_ADDRESS 0x3C
50 #define TSL2561_I2C_ADDRESS 0x39 //Light Sensor //Look sensorapi para
    ejemplo
51 #define TSL2561_I2C_ADDRESS 0x29 //sensor de color
52 //HW-611
53 // #define BMP280_ADDRESS 0x77
54 #define BMP280_ADDRESS_HW 0x76 //it's not 77, be carefull, address repeated
    with Gyros
55 //HW-S12
56 #define HW_S12_1 0x68 //presumably Gyro
57 #define HW_S12_2 0x76 //presumably BMP
58 //GY-88
59 #define GYROs1_ADDRESS 0x1E
60 #define GYROs2_ADDRESS 0x76
61 #define GYROs3_ADDRESS 0x68
62
63 #define VERSION "Kit-Iot1.2.0"
64
65 #include "SPI.h"
66 #include <WiFi.h>
67 #include <Metro.h>
68 // #include <Servo.h>
69 #include <RTCLib.h>
70 #include <PubSubClient.h>
71 // #include <Adafruit_GFX.h>
72 #include <Adafruit_Sensor.h>
73 #include <Adafruit_SSD1306.h>
74 #include <Adafruit_TSL2561_U.h>
75 #include <Adafruit_BMP280.h>
76
77 //TSL
78 /* Display some basic information on this sensor */
79 void displaySensorDetails(Adafruit_TSL2561_Unified *tsl);
80 /* Setup the sensor gain and integration time */
81 void configureSensor(Adafruit_TSL2561_Unified *tsl);
82 int tsl_measure(Adafruit_TSL2561_Unified *tsl);
83
84 // OLED
85 //void draw(void);
86 //void draw2(void);
87
88 //RTC functions
89 byte decToBcd(byte val);
90 byte bcdToDec(byte val);
91 /**
92  * @brief
93  * @param
94  * @param
95  * @return
96  */
97 void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek,
    byte dayOfMonth, byte month, byte year);

```

```

98  /**
99   * @brief
100   * @param
101   * @return
102   */
103  void readDS3231time(byte *second, byte *minute, byte *hour, byte *dayOfWeek,
104                    byte *dayOfMonth, byte *month, byte *year);
105  /**
106   * @brief
107   * @param
108   * @return
109   */
110  void displayTime();
111  /**
112   * @brief
113   * @param
114   * @return
115   */
116  float get3231Temp(byte *tMSB, byte *tLSB);
117  /**
118   * @brief
119   * @param
120   * @return
121   */
122  void get3231Date();
123
124  #endif

```

code/iotkit_h.ino

7.7. Librería IoTkit.cpp

```

1  /**
2   * @author: Guillermo Becerra
3   * @e-mail: g.becerra.o@icloud.com
4   *
5   * @company: GuilleD
6   * @e-mail: g.becerra.o@icloud.com
7   *
8   * @date: March, 2023
9   * @version: 1.2.0
10  *
11  * @description:
12  *   This schetch is a
13  *   DevNotes: Check the I2C address of BMP, in Adafruit library was
14  *   registered as 58, but by reading it from sensor, it was in fact 55.
15  */
16  /**
17   * Include Libraries
18   */
19
20  #include "Arduino.h"
21  #include "iotkit.h"
22  #include <Wire.h>
23
24

```

```

25 //-----%%-----%%-----%%-----%%-----%%-----%%-----
26 //                                     TSL561
27 //-----%%-----%%-----%%-----%%-----%%-----%%-----
28 // Requieres: #include <Adafruit_SSD1306.h> #include <Adafruit_TSL2561_U.
    h>
29 // ref: Library's example: sensorapi
30 /*
31     Displays some basic information on this sensor from the unified
32     sensor API sensor_t type (see Adafruit_Sensor for more information)
33 */
34 void displaySensorDetails(Adafruit_TSL2561_Unified *tsl)
35 {
36     sensor_t sensor;
37     tsl->getSensor(&sensor);
38     Serial.println("-----");
39     Serial.print ("Sensor:      "); Serial.println(sensor.name);
40     Serial.print ("Driver Ver:   "); Serial.println(sensor.version);
41     Serial.print ("Unique ID:    "); Serial.println(sensor.sensor_id);
42     Serial.print ("Max Value:     "); Serial.print(sensor.max_value);
        Serial.println(" lux");
43     Serial.print ("Min Value:     "); Serial.print(sensor.min_value);
        Serial.println(" lux");
44     Serial.print ("Resolution:    "); Serial.print(sensor.resolution);
        Serial.println(" lux");
45     Serial.println("-----");
46     Serial.println("");
47     delay(500);
48 }
49
50 /*
51     Configures the gain and integration time for the TSL2561
52 */
53 void configureSensor(Adafruit_TSL2561_Unified *tsl)
54 {
55     /* You can also manually set the gain or enable auto-gain support */
56     // tsl.setGain(TSL2561_GAIN_1X);      /* No gain ... use in bright
        light to avoid sensor saturation */
57     // tsl.setGain(TSL2561_GAIN_16X);     /* 16x gain ... use in low light
        to boost sensitivity */
58     tsl->enableAutoRange(true);           /* Auto-gain ... switches
        automatically between 1x and 16x */
59
60     /* Changing the integration time gives you better sensor resolution
        (402ms = 16-bit data) */
61     tsl->setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS);      /* fast but
        low resolution */
62     // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS); /* medium
        resolution and speed */
63     // tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_402MS); /* 16-bit
        data but slowest conversions */
64
65     /* Update these values depending on what you've set above! */
66     Serial.println("-----");
67     Serial.print ("Gain:          "); Serial.println("Auto");
68     Serial.print ("Timing:         "); Serial.println("13 ms");
69     Serial.println("-----");
70 } //configureSensor
71
72 int tsl_measure(Adafruit_TSL2561_Unified *tsl){
73     sensors_event_t event;

```

```

74     tsl->getEvent(&event);
75
76     /* Display the results (light is measured in lux) */
77     if (event.light){
78         Serial.print(event.light); Serial.println(" lux");
79     } else {
80         /* If event.light = 0 lux the sensor is probably saturated
81            and no reliable data could be generated! */
82         Serial.println("Sensor overload");
83     }
84     return event.light;
85 } //tsl.measure
86
87
88
89 //-----%%-----%%-----%%-----%%-----%%-----%%-----
90 //                                OLED
91 //-----%%-----%%-----%%-----%%-----%%-----%%-----
92 // Requieres: #include <Adafruit_GFX.h> #include <Adafruit_SSD1306.h>
93 // ref: https://randomnerdtutorials.com/esp32-ssd1306-oled-display-
94 //       arduino-ide/
95
96 //-----%%-----%%-----%%-----%%-----%%-----%%-----
97 //                                RTC
98 //-----%%-----%%-----%%-----%%-----%%-----%%-----
99 // Requieres: Wire.h // #define DS3231_I2C_ADDRESS1 0x68
100 // ref: https://handsontec.com/index.php/arduino-6-ds3231-real-time-clock
101 //       -module/
102
103 byte decToBcd(byte val)
104 {
105     return( (val/10*16) + (val%10) );
106 }
107 // Convert binary coded decimal to normal decimal numbers
108 byte bcdToDec(byte val)
109 {
110     return( (val/16*10) + (val%16) );
111 }
112
113 void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek,
114                   byte dayOfMonth, byte month, byte year)
115 {
116     // sets time and date data to DS3231
117     Wire.beginTransmission(DS3231_I2C_ADDRESS1);
118     Wire.write(0); // set next input to start at the seconds register
119     Wire.write(decToBcd(second)); // set seconds
120     Wire.write(decToBcd(minute)); // set minutes
121     Wire.write(decToBcd(hour)); // set hours
122     Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=
123 //       Saturday)
124     Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
125     Wire.write(decToBcd(month)); // set month
126     Wire.write(decToBcd(year)); // set year (0 to 99)
127     Wire.endTransmission();
128 }
129
130 void readDS3231time(byte *second,
131                    byte *minute,
132                    byte *hour,
133                    byte *dayOfWeek,

```

```

130 byte *dayOfMonth,
131 byte *month,
132 byte *year)
133 {
134     Wire.beginTransmission(DS3231_I2C_ADDRESS1);
135     Wire.write(0); // set DS3231 register pointer to 00h
136     Wire.endTransmission();
137     Wire.requestFrom(DS3231_I2C_ADDRESS1, 7);
138
139     // request seven bytes of data from DS3231 starting from register 00h
140     *second = byte(Wire.read() & 0x7f);
141     *minute = byte(Wire.read());
142     // *second = bcdToDec(Wire.read() & 0x7f);
143     // *minute = bcdToDec(Wire.read());
144     *hour = bcdToDec(Wire.read() & 0x3f);
145     *dayOfWeek = bcdToDec(Wire.read());
146     *dayOfMonth = bcdToDec(Wire.read());
147     *month = bcdToDec(Wire.read());
148     *year = bcdToDec(Wire.read());
149 }
150 void displayTime()
151 {
152     byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
153     // retrieve data from DS3231
154     readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month
155         , &year);
156     // send it to the serial monitor
157     Serial.print(hour, DEC);
158     // convert the byte variable to a decimal number when displayed
159     Serial.print(":");
160     if (minute<10)
161     {
162         Serial.print("0");
163     }
164     Serial.print(minute);
165     Serial.print(":");
166     if (second<10)
167     {
168         Serial.print("0");
169     }
170     Serial.print(second);
171     Serial.print(" ");
172     Serial.print(dayOfMonth, DEC);
173     Serial.print("/");
174     Serial.print(month, DEC);
175     Serial.print("/");
176     Serial.print(year, DEC);
177     Serial.print(" Day of week: ");
178     switch(dayOfWeek){
179     case 1:
180         Serial.println("Sunday");
181         break;
182     case 2:
183         Serial.println("Monday");
184         break;
185     case 3:
186         Serial.println("Tuesday");
187         break;
188     case 4:
189         Serial.println("Wednesday");

```

```

189     break;
190 case 5:
191     Serial.println("Thursday");
192     break;
193 case 6:
194     Serial.println("Friday");
195     break;
196 case 7:
197     Serial.println("Saturday");
198     break;
199 default:
200     Serial.println("Def-day case");
201     break;
202 }
203 }
204
205 float get3231Temp(byte *tMSB, byte *tLSB)
206 {
207     float temp3231;
208
209     //temp registers (11h-12h) get updated automatically every 64s
210     Wire.beginTransaction(DS3231_I2C_ADDRESS1);
211     Wire.write(0x11);
212     Wire.endTransmission();
213     Wire.requestFrom(DS3231_I2C_ADDRESS1, 2);
214
215     if(Wire.available()) {
216         *tMSB = Wire.read(); //2's complement int portion
217         *tLSB = Wire.read(); //fraction portion
218
219         temp3231 = (*tMSB & B01111111); //do 2's math on Tmsb
220         temp3231 += ( (*tLSB >> 6) * 0.25 ); //only care about bits 7 & 8
221     }
222     else {
223         //oh noes, no data!
224     }
225
226     return temp3231;
227 } // get3231Temp
228
229 void get3231Date() //another version
230 { /*
231     // send request to receive data starting at register 0
232     Wire.beginTransaction(DS3231_I2C_ADDRESS1); // 104 is DS3231 device
        address
233     Wire.write(0x00); // start at register 0
234     Wire.endTransmission();
235     Wire.requestFrom(DS3231_I2C_ADDRESS1, 7); // request seven bytes
236
237     if(Wire.available()) {
238         seconds = Wire.read(); // get seconds
239         minutes = Wire.read(); // get minutes
240         hours   = Wire.read(); // get hours
241         day     = Wire.read();
242         date    = Wire.read();
243         month   = Wire.read(); //temp month
244         year    = Wire.read();
245
246         seconds = (((seconds & B11110000)>>4)*10 + (seconds & B00001111)); //
            convert BCD to decimal

```

```

247     minutes = (((minutes & B11110000)>>4)*10 + (minutes & B00001111)); //
        convert BCD to decimal
248     hours   = (((hours & B00110000)>>4)*10 + (hours & B00001111)); //
        convert BCD to decimal (assume 24 hour mode)
249     day     = (day & B00000111); // 1-7
250     date    = (((date & B00110000)>>4)*10 + (date & B00001111)); // 1-31
251     month   = (((month & B00010000)>>4)*10 + (month & B00001111)); //msb7
        is century overflow
252     year    = (((year & B11110000)>>4)*10 + (year & B00001111));
253 }
254 else {
255     //oh noes, no data!
256 }
257
258 switch (day) {
259     case 1:
260         strcpy(weekDay, "Sun");
261         break;
262     case 2:
263         strcpy(weekDay, "Mon");
264         break;
265     case 3:
266         strcpy(weekDay, "Tue");
267         break;
268     case 4:
269         strcpy(weekDay, "Wed");
270         break;
271     case 5:
272         strcpy(weekDay, "Thu");
273         break;
274     case 6:
275         strcpy(weekDay, "Fri");
276         break;
277     case 7:
278         strcpy(weekDay, "Sat");
279         break;
280 }*/
281 } //get3231Date

```

Listing 7.1: Contenido del archivo de extensión **.cpp* que contiene el código fuente de la librería *iot-kit* desarrollada.

7.8. Código principal

```

1  #include "iotkit.h"
2
3  //Compiler directives
4  #define WIFLSTATE 1
5  #define LED_FADING 0
6  #define RTCLIB_COMP 0
7  #define TSLSTATE 1
8  #define BMP.STATE 1
9  #define DEBUG.FLAG 0
10 #define WEB.TIME 0
11 #define SENSOR_UV 0
12
13 //Macro parameters
14 #define LOOP_DELAY 15000

```

```

15 #define INIT_DELAY 300
16 #define LOOP_TIME 250
17 #define MQTT_TIME 5000
18 #define RECONNECT_TIME 1000
19
20 //WIFI
21 const char* ssid = "HUAWEI-a4XC"; // Complete your WiFi Network Name
    password here!
22 const char* password = "A2U97xPg"; // Complete your WiFi Password here
    !
23
24 /**
25  * Ubidots MQTT Broker Credentials
26  */
27 #define TOKEN "BBFF-MgPO0CCyLSrHGSRmSBA7sZJuBRuTAr" // Your Ubidots '
    TOKEN // Do not modify for this example
28 // Disp1: BBFF-Dd8rYdDHeWQonujJFprgPM7ZMKXG18
29 // Disp2: BBFF-MgPO0CCyLSrHGSRmSBA7sZJuBRuTAr
30
31 #define MQTT_CLIENT_NAME "DarioS1-ELO1" // MQTT client Name, please
    enter your own 8-12 alphanumeric character ASCII string;
32 //it should be a random and
    unique ascii string and different from all other devices
33 /**
34  * Ubidots MQTT labels
35  */
36 #define DEVICE_LABEL "dario-s1-udd2" // < ----- Write your
    device label here
37
38 #define VAR_LBL_LIGHT_SEN "luz-lux"
39 #define VAR_LBL_UV_SEN "uv"
40 #define VAR_LBL_TEMP_SEN "temperatura"
41 #define VAR_LBL_PRESSURE_SEN "presion"
42 #define VAR_LBL_PRESSURE_SEN "presion"
43 #define VAR_LBL_ALTURA_SEN "altura"
44 #define VAR_LBL_POT_SEN "perilla"
45 #define VAR_LBL_SWITCH1 "switch1"
46 #define VAR_LBL_SWITCH2 "switch2"
47 #define VAR_LBL_BUTTON1 "button1"
48 #define VAR_LBL_BUTTON2 "button2"
49 #define VAR_LBL_SERVO "servo"
50 #define VAR_LBL_BUZZER_FREQ "buzzer-frec"
51 #define VAR_LBL_BUZZER_VOL "buzzer-vol"
52 #define VAR_LBL_BUZZER_SWITCH "buzzer-switch"
53
54 #define VAR_LBL_LED_RED "led-red"
55 #define VAR_LBL_LED_GREEN "led-green"
56 #define VAR_LBL_LED_BLUE "led-blue"
57
58 /**
59  * Ubidots Broker IP
60  */
61 char mqttBroker[] = "industrial.api.ubidots.com";
62 #define API_BUILDER "/v1.6/devices/"
63
64 /**
65  * Auxiliar variables
66  */
67 char payload[150]; // buffer to save message
68 char topic[150]; // buffer to save topic

```

```

69 char str_sensor[10]; // buffer for sensor value
70 char buzzer_vol_mes = 0; // Volumen del buzzer cuando publica
71
72 Metro time_to_MQTT = Metro(MQTT.TIME);
73 Metro time_to_reconnect = Metro(RECONNECT.TIME);
74 Metro time_loop = Metro(LOOP.TIME); // PubSub client loop time; it is
    used to refresh the connection.
75
76 /**
77  * WiFi Client Connected to Ubidots
78  */
79 WiFiClient ubidots;
80 /**
81  * MQTT Client; this will make available the MQTT related functionalities
82  */
83 PubSubClient client(ubidots);
84
85 // setting PWM properties
86 //LED
87 const int freq = 5000;
88 const int led_freq = 5000;
89 const int ledChannel = 0;
90 const byte led_red_channel = 2;
91 const byte led_green_channel = 3;
92 const byte led_blue_channel = 4;
93 byte led_red_dc = 0;
94 byte led_green_dc = 0;
95 byte led_blue_dc = 0;
96
97 const int resolution = 8;
98 int buzzer_freq = 2000;
99 byte buzzer_vol = 0;
100 const int buzzer_channel = 1;
101
102 // OLED Screen
103 Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
104
105 //Sensors
106 Adafruit_BMP280 bmp; // uses I2C interface
107 Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561
    _ADDR_FLOAT, 12345);
108 Adafruit_Sensor *bmp_temp = bmp.getTemperatureSensor();
109 Adafruit_Sensor *bmp_pressure = bmp.getPressureSensor();
110 RTC_DS3231 rtc;
111
112 //Global Variables for measurements
113 long timezone = 1;
114 byte daysavetime = 1;
115 byte seconds, minutes, hours, day, date, month, year;
116 byte tMSB, tLSB;
117 float my_temp;
118
119 //-----%%-----%%-----%%-----%%-----%%-----
120 //                      SETUP
121 //-----%%-----%%-----%%-----%%-----%%-----
122 void setup() {
123     pinMode(BUTTON1, INPUT);
124     pinMode(BUTTON2, INPUT);
125     pinMode(SWITCH1, INPUT);
126     pinMode(SWITCH2, INPUT);

```

```

127 // configure LED PWM functionalities
128 //ledcSetup(ledChannel, freq, resolution);
129 // attach the channel to the GPIO to be controlled
130 ledcSetup(led_red_channel, led_freq, resolution);
131 ledcAttachPin(RED_LED, led_red_channel);
132
133 ledcSetup(led_green_channel, led_freq, resolution);
134 ledcAttachPin(GREEN_LED, led_green_channel);
135
136 ledcSetup(led_blue_channel, led_freq, resolution);
137 ledcAttachPin(BLUE_LED, led_blue_channel);
138
139 // configure BUZZer PWM functionalites
140 ledcSetup(buzzer_channel, buzzer_freq, resolution);
141 ledcAttachPin(BUZZER, buzzer_channel);
142
143 //Communication Config
144 Wire.begin();
145 Serial.begin(115200);
146 Serial.print("/\n*/\n*/\n*/\n*/\n*/\n*/\n*/\n*/\n*/\nw");
147 Serial.println("*****");
148 Serial.println("**");
149 Serial.println("**");
150 Serial.print("** Hola"); beep(buzzer_channel, 2000, 250, 255);
151 delay(500);
152 Serial.println([0xFFFD]c[U+FFFB][U+6FFD]);beep(buzzer_channel, 2000, 250, 255
);
153 delay(500);
154 Serial.println("/** Soy Dario SI y te doy la bienvenida a STEAM en tu
casa.");beep(buzzer_channel, 2500, 250, 255);
155 Serial.println("**");
156 Serial.println("**");
157 Serial.println("*****");
158 delay(1500);
159 Serial.println("**");
160 Serial.println("**");
161 Serial.println("[U+FFFD]Est[U+FFFD]so/a para hacer posible lo imposible?")
;
162 beep(buzzer_channel, 2000, 250, 125);
163 delay(2000);
164 Serial.println("/** Muy bien , manos a la obra!");
165 beep(buzzer_channel, 2500, 250, 125); delay(250);
166 beep(buzzer_channel, 3000, 250, 125);
167 Serial.println("**");
168 Serial.println("**");
169 Serial.println("/*****");
170 delay(1000);
171
172 #if BMP_STATE
173 BMP_init();
174 #endif
175
176 OLED_init();
177
178 #if RTCLIB_COMP
179 RTC.init();
180 #endif
181
182 TSL_init();
183 configureSensor(&tsl);

```

```

184 //SD_init();
185
186 //RTC Section
187 // set the initial time here:
188 // DS3231 seconds, minutes, hours, day, date, month, year
189 //setDS3231time(30,42,16,5,13,10,16);
190
191 // WiFi Section
192 #if WIFLSTATE
193     WIFI.init();
194 #endif
195
196 // Web Time
197 #if WEB.TIME
198     configTime(3600*timezone, daysavetime*3600, "time.nist.gov", "0.pool.
199     ntp.org", "1.pool.ntp.org");
200     struct tm tmstruct ;
201     delay(2000);
202     tmstruct.tm_year = 0;
203     getLocalTime(&tmstruct, 5000);
204     Serial.printf("\nNow is : %d-%02d-%02d %02d:%02d:%02d\n", (tmstruct.
205     tm_year)+1900, (tmstruct.tm_mon)+1, tmstruct.tm_mday, tmstruct.tm_hour
206     , tmstruct.tm_min, tmstruct.tm_sec);
207     Serial.println("");
208 #endif
209 client.setServer(mqttBroker, 1883); // config ip and port MQTT Broker
210 client.setCallback(callback); // set callback function to attempt to
211     subscribe to topics
212
213 //OLED Setting
214 display.clearDisplay();
215 display.setCursor(0, 0);
216 display.println("TSL: ");
217 display.setCursor(64, 0);
218 display.println("lux");
219 display.display();
220
221 } // setup
222
223 //RTCLib
224 char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday",
225     "Thursday", "Friday", "Saturday"};
226 int potValue, uvValue;
227 boolean bool_inputs[4] = {0};
228
229 //-----%%-----%%-----%%-----%%-----%%-----%%-----
230 //                                LOOP
231 //-----%%-----%%-----%%-----%%-----%%-----%%-----
232 int n_mediciones = 0;
233 void loop(){
234     Serial.println("*****-----" + (String)(n_mediciones) +
235     "*****");
236     if (!client.connected()) { // PubSubclient conection check
237         reconnect();
238     }
239     client.loop(); // Keeps alive connection and checks for
240         new published values in suscribed topic.
241
242     if(time_to_MQTT.check() == 1){

```

```

236 // initialize buzzer parameters to provide the user auditive feedback
237 // of the code stage.
238 int temp_freq = 150;
239 byte freq_increment = 50;
240
241 //Data acquisition: Lumen and UV Intensity
242 int tsl_lux = tsl_measure(&tsl);
243 potValue = analogRead(POT_SENSOR);
244 uvValue = analogRead(UV_SENSOR);
245 buzzer_vol_mes = map(potValue, 0, 4095, 0, 255);
246 //Publicar
247 Serial.print("Publicando: ");
248 //TSL
249 dtostrf(tsl_lux, 4, 2, str_sensor); // double to string with format
250 sprintf(topic, "%s%s", API_BUILDER, DEVICE_LABEL); // for example "/
251 v1.6/devices/esp32-iot"
252 sprintf(payload, "{\"%s\":{\"value\": %s}}", VAR_LBL_LIGHT_SEN,
253 str_sensor); // for example {"sensor":{"value":1023}}
254 #if LOG > LOG_NONE
255 Serial.print("sensor de luz, ");
256 #endif
257 client.publish(topic, payload);
258 client.loop(); //—
259 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
260 freq_increment;
261 time_loop.reset();
262 while(time_loop.check() == 0) yield();
263
264 //POT
265 dtostrf(potValue, 4, 2, str_sensor); // double to string with format
266 sprintf(topic, "%s%s", API_BUILDER, DEVICE_LABEL); // for example "/
267 v1.6/devices/esp32-iot"
268 sprintf(payload, "{\"%s\":{\"value\": %s}}", VAR_LBL_POT_SEN,
269 str_sensor); // for example {"sensor":{"value":1023}}
270 #if LOG > LOG_NONE
271 Serial.print("Potenciometro, ");
272 #endif
273 client.publish(topic, payload);
274 client.loop(); //—
275 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
276 freq_increment;
277 time_loop.reset();
278 while(time_loop.check() == 0) yield();
279
280 //UV
281 #if SENSOR_UV
282 dtostrf(uvValue, 4, 2, str_sensor); // double to string with format
283 sprintf(topic, "%s%s", API_BUILDER, DEVICE_LABEL); // for example
284 "/v1.6/devices/esp32-iot"
285 sprintf(payload, "{\"%s\":{\"value\": %s}}", VAR_LBL_UV_SEN,
286 str_sensor); // for example {"sensor":{"value":1023}}
287 #if LOG > LOG_NONE
288 Serial.print("Sensor UV, ");
289 #endif
290 client.publish(topic, payload);
291 client.loop(); //—
292 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
293 freq_increment;
294 time_loop.reset();
295 while(time_loop.check() == 0) yield();

```

```

286 #endif
287 //Temperatura
288 dtostrf(bmp.readTemperature(), 4, 2, str_sensor); // double to string
    with format
289 sprintf(topic, "%s%s", APIBUILDER, DEVICE_LABEL); // for example "/"
    v1.6/devices/esp32-iot"
290 sprintf(payload, "{\n\"%s\":{\n\"value\": %s}}", VAR_LBL_TEMP_SEN,
    str_sensor); // for example {"sensor":{"value":1023}}
291 #if LOG>LOG_NONE
292     Serial.print("Temperatura, ");
293 #endif
294 client.publish(topic, payload);
295 client.loop(); //——
296 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
    freq_increment;
297 time_loop.reset();
298 while(time_loop.check() == 0) yield();
299
300 //Pressure
301 dtostrf(bmp.readPressure(), 4, 2, str_sensor); // double to string
    with format
302 sprintf(topic, "%s%s", APIBUILDER, DEVICE_LABEL); // for example "/"
    v1.6/devices/esp32-iot"
303 sprintf(payload, "{\n\"%s\":{\n\"value\": %s}}", VAR_LBL_PRESSURE_SEN,
    str_sensor); // for example {"sensor":{"value":1023}}
304 #if LOG>LOG_NONE
305     Serial.print("Presion ");
306 #endif
307 client.publish(topic, payload);
308 client.loop(); //——
309 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
    freq_increment;
310 time_loop.reset();
311 while(time_loop.check() == 0) yield();
312
313 //Altura
314 dtostrf(bmp.readAltitude(), 4, 2, str_sensor); // double to string
    with format
315 sprintf(topic, "%s%s", APIBUILDER, DEVICE_LABEL); // for example "/"
    v1.6/devices/esp32-iot"
316 sprintf(payload, "{\n\"%s\":{\n\"value\": %s}}", VAR_LBL_ALTURA_SEN,
    str_sensor); // for example {"sensor":{"value":1023}}
317 #if LOG>LOG_NONE
318     Serial.println("y Altitud");
319 #endif
320 client.publish(topic, payload);
321 client.loop(); //——
322 beep(buzzer_channel, temp_freq, 250, buzzer_vol_mes); temp_freq+=
    freq_increment;
323 time_loop.reset();
324 while(time_loop.check() == 0) yield();
325
326
327 }//if mqtt check
328
329 // Interface Reading
330 bool_inputs[0] = digitalRead(BUTTON1);
331 bool_inputs[1] = digitalRead(BUTTON2);
332 bool_inputs[2] = digitalRead(SWITCH1);
333 bool_inputs[3] = digitalRead(SWITCH2);

```

```

334 Serial.println(" Mediciones");
335 Serial.print("Pot:"+(String)(potValue)+"\t");
336 Serial.print("B1: " +(String)( bool_inputs[0])+"\t"+"B2: " +(String)(
    bool_inputs[1])+"\t");
337 Serial.println("S1: " +(String)( bool_inputs[2])+"\t"+"S2: " +(String)(
    bool_inputs[3]));
338
339 //char msg[50]=" ";
340 //snprintf(msg,sizeof msg,"Time: %0.2d:%0.2d:%0.2d",hour(),minute(),
    second());
341 //display.setCursor(0,48); //lines of 16px height
342 //display.print(msg);
343
344 //Serial.println("/-----BMP280-----/");
345 Serial.print(F("Temperatura = "));
346 Serial.print(bmp.readTemperature());
347 Serial.print(" *C \t");
348
349 Serial.print(F("Presi[U+FFD0] = "));
350 Serial.print(bmp.readPressure());
351 Serial.print(" [Pa] \t");
352
353 float alt_const = 1013.25;
354 Serial.print(F("Altitud = "));
355 Serial.print(bmp.readAltitude());
356 Serial.println(" [m]");
357
358 Serial.print("SensorLuz: ");
359 int tsl_lux = tsl_measure(&tsl);
360 display.fillRect(30, 0, 30, 10, BLACK);
361 //display.setCursor(30, 0); display.println(" ");
362 display.setCursor(30, 0); display.println((String)(tsl_lux));
363 display.display();
364
365 Serial.println("*****-----*****\n\n");
366 n_mediciones++;
367 delay(LOOP_DELAY);
368 } //Loop
369
370 //-----%%-----%%-----%%-----%%-----%%-----%%-----
371 // OLEDDATA
372 //-----%%-----%%-----%%-----%%-----%%-----%%-----
373 void OLEDDATA_init(){
374 // OLEDDATA setup
375 if(!display.begin(SSD1306_SWITCHCAPVCC, OLEDDATA_I2C_ADDRESS)) { // Address
    0x3D||0x3C for 128x64
376 Serial.println(F("SSD1306 allocation failed"));
377 for(;;);
378 }
379 delay(500);
380 display.clearDisplay();
381 display.setTextSize(1);
382 display.setTextColor(WHITE);
383 display.setCursor(0, 0); //x,y
384 // Display static text
385 display.println("Hello, world!");
386 display.display();
387 } //OLEDDATA_init
388
389 //-----%%-----%%-----%%-----%%-----%%-----%%-----

```

```

390 //                                     WIFI_INIT
391 //-----%%-----%%-----%%-----%%-----%%-----
392 void WIFI_init() {
393     Serial.println("/*****");
394     Serial.print("Conectando a: ");
395     Serial.println(ssid);
396     WiFi.begin(ssid, password);
397     while (WiFi.status() != WL_CONNECTED) {
398         delay(500);
399         Serial.print(".");
400     }
401     Serial.println("WiFi conectado!");
402     beep(buzzer_channel, 2500, 250, 125);
403     beep(buzzer_channel, 2500, 250, 125);
404     Serial.print("Direccion IP: ");
405     Serial.println(WiFi.localIP());
406     Serial.println("/*****\n\n\n");
407     delay(1500);
408 } // WIFI_init
409
410 //-----%%-----%%-----%%-----%%-----%%-----
411 //                                     TLS2561_INIT
412 //-----%%-----%%-----%%-----%%-----%%-----
413 #if TSLSTATE
414 void TSL_init() {
415     /* Initialise the sensor */
416     //use tsl.begin() to default to Wire,
417     //tsl.begin(&Wire2) directs api to use Wire2, etc.
418     if (!tsl.begin())
419     {
420         /* There was a problem detecting the TSL2561 ... check your
421         connections */
422         Serial.print("Oops, no TSL2561 detected ... Check your wiring or
423         I2C ADDR!");
424         while(1);
425     }
426     #if DEBUGFLAG
427         displaySensorDetails(&tsl);
428     #endif
429 } //TSL_init
430 #endif
431
432 //-----%%-----%%-----%%-----%%-----%%-----
433 //                                     BMP_INIT
434 //-----%%-----%%-----%%-----%%-----%%-----
435 void BMP_init() {
436     if (!bmp.begin(BMP280_ADDRESS_HW)) {
437         Serial.println(F("Could not find a valid BMP280 sensor, check wiring!
438         "));
439         while (1) delay(10);
440     }
441     /* Default settings from datasheet. */
442     bmp.setSampling(Adafruit_BMP280::MODE_NORMAL, /* Operating Mode. */
443                    Adafruit_BMP280::SAMPLING_X2, /* Temp. oversampling
444                    */
445                    Adafruit_BMP280::SAMPLING_X16, /* Pressure
446                    oversampling */
447                    Adafruit_BMP280::FILTER_X16, /* Filtering. */
448                    Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

```

```

445 } //bmp_init
446
447
448 //-----%%-----%%-----%%-----%%-----%%-----%%-----
449 //-----RECONNECT-----
450 //-----%%-----%%-----%%-----%%-----%%-----%%-----
451 void reconnect() {
452     // Loop until we're reconnected
453     while (!client.connected()) {
454         #if LOG>LOG.NONE
455             Serial.println("Abriendo conexion MQTT");
456         #endif
457         // Attempt to connect
458         if (client.connect(MQTT.CLIENT_NAME, TOKEN, "")) { // client name,
            user, pass
459             #if LOG>LOG.NONE
460                 Serial.println("Protocolo IoT conectado! \n Comenzando en un par
de segundos ;-)");
461             #endif
462             beep(buzzer_channel, 2500, 250, 125);
463             beep(buzzer_channel, 2500, 250, 125);
464             //delay(1500);
465
466             client.subscribe(String(String(API_BUILDER) + String(DEVICE_LABEL)
+ "/" + String(VAR_LBL_BUZZER_FREQ) + "/lv").c_str()); // "/v1.6/
devices/esp32-iot/light-switch/lv" // lv = last value
467             time_to_reconnect.reset();
468             while(time_to_reconnect.check() == 0) yield();
469             client.subscribe(String(String(API_BUILDER) + String(DEVICE_LABEL)
+ "/" + String(VAR_LBL_BUZZER_VOL) + "/lv").c_str()); // "/v1.6/
devices/esp32-iot/light-switch/lv" // lv = last value
470             time_to_reconnect.reset();
471             while(time_to_reconnect.check() == 0) yield();
472             //LED
473             client.subscribe(String(String(API_BUILDER) + String(DEVICE_LABEL)
+ "/" + String(VAR_LBL_LED_RED) + "/lv").c_str()); // "/v1.6/devices/
esp32-iot/light-switch/lv" // lv = last value
474             time_to_reconnect.reset();
475             while(time_to_reconnect.check() == 0) yield();
476             client.subscribe(String(String(API_BUILDER) + String(DEVICE_LABEL)
+ "/" + String(VAR_LBL_LED_GREEN) + "/lv").c_str()); // "/v1.6/
devices/esp32-iot/light-switch/lv" // lv = last value
477             time_to_reconnect.reset();
478             while(time_to_reconnect.check() == 0) yield();
479             client.subscribe(String(String(API_BUILDER) + String(DEVICE_LABEL)
+ "/" + String(VAR_LBL_LED_BLUE) + "/lv").c_str()); // "/v1.6/devices
/esp32-iot/light-switch/lv" // lv = last value
480             time_to_reconnect.reset();
481             while(time_to_reconnect.check() == 0) yield();
482
483             yield();
484
485     } else {
486         #if LOG>LOG.NONE
487             Serial.print("Failed, result code =");
488             Serial.print(client.state());
489             Serial.println(" try again in 2 seconds");
490         #endif
491         // Wait 2 seconds before retrying
492         time_to_reconnect.reset();

```

```

493     while(time_to_reconnect.check() == 0) yield();//ESP.wdtFeed();
494     time.loop.reset();
495     while(time.loop.check() == 0) yield();//ESP.wdtFeed();
496 }
497 yield();
498 }
499 }//RECONNECT
500
501 //-----%%-----%%-----%%-----%%-----%%-----
502 //                                CALLBACK
503 //-----%%-----%%-----%%-----%%-----%%-----
504 /**
505  * Function call when my subscribe topic is published
506  */
507 void callback(char* topic, byte* payload, unsigned int length) {
508     char p[length + 1]; // char array
509     memcpy(p, payload, length); // byte array to char array
510     p[length] = NULL; // string's end
511
512     #if LOG>LOG.NONE
513         Serial.println(); // linefeed // new line
514         Serial.print(topic); // print topic for debug
515         Serial.print("\t"); // tab
516         Serial.write(payload, length); // printing byte array for debug
517         Serial.println();
518     #endif
519     /**
520      * Execute for topic receive
521      */
522     if(String(topic).equals(String(String(APIBUILDER) + String(
523         DEVICE_LABEL) + "/" + String(VAR_LBL_BUZZER_VOL) + "/lv"))){ // "/v1
524         .6/devices/esp32-iot/threshold/lv"
525         byte temp = buzzer_vol;
526         buzzer_vol = String(p).toInt();
527         if(temp != buzzer_vol){
528             ledcWrite(buzzer_channel, buzzer_vol);
529         }
530     #if LOG>LOG.NONE
531         Serial.print("\nBuzzer-vol: "+(String)(buzzer_vol));
532     #endif
533 } //if Buzzer switch
534
535 if(String(topic).equals(String(String(APIBUILDER) + String(
536     DEVICE_LABEL) + "/" + String(VAR_LBL_BUZZER_FREQ) + "/lv"))){ // "/v1
537     .6/devices/esp32-iot/threshold/lv"
538     int temp = buzzer_freq;
539     buzzer_freq = String(p).toInt();
540     if(temp != buzzer_freq && buzzer_vol!=0){
541         ledcWriteTone(buzzer_channel, buzzer_freq);
542     }else if (temp != buzzer_freq && buzzer_vol==0){
543         ledcWriteTone(buzzer_channel, buzzer_freq);
544     }
545     ledcWrite(buzzer_channel, buzzer_vol);
546 }
547 #if LOG>LOG.NONE
548     Serial.print("\nBuzzer-freq: "+(String)(buzzer_freq));
549 #endif
550 } //if Buzzer freq
551
552 //LEDS

```

```

548 if (String(topic).equals(String(String(API_BUILDER) + String(
    DEVICE_LABEL) + "/" + String(VAR_LBL_LED_RED) + "/lv"))){ // "/v1.6/
    devices/esp32-iot/threshold/lv"
549     byte temp = led_red_dc;
550     led_red_dc = String(p).toInt();
551     if (temp != led_red_dc)
552         ledcWrite(led_red_channel, led_red_dc);
553     #if LOG>LOG_NONE
554         Serial.print("\nLed Red-DC: ");
555         Serial.println(led_red_dc);
556     #endif
557 } //if LED RED
558
559 if (String(topic).equals(String(String(API_BUILDER) + String(
    DEVICE_LABEL) + "/" + String(VAR_LBL_LED_GREEN) + "/lv"))){ // "/v1
    .6/ devices/esp32-iot/threshold/lv"
560     byte temp = led_green_dc;
561     led_green_dc = String(p).toInt();
562     if (temp != led_green_dc)
563         ledcWrite(led_green_channel, led_green_dc);
564     #if LOG>LOG_NONE
565         Serial.print("\nLed Green-DC: ");
566         Serial.println(led_green_dc);
567     #endif
568 } //if LED RED
569
570 if (String(topic).equals(String(String(API_BUILDER) + String(
    DEVICE_LABEL) + "/" + String(VAR_LBL_LED_BLUE) + "/lv"))){ // "/v1.6/
    devices/esp32-iot/threshold/lv"
571     byte temp = led_blue_dc;
572     led_blue_dc = String(p).toInt();
573     if (temp != led_blue_dc)
574         ledcWrite(led_blue_channel, led_blue_dc);
575     #if LOG>LOG_NONE
576         Serial.print("\nLed Blue-DC: ");
577         Serial.println(led_blue_dc);
578     #endif
579 } //if LED RED
580
581     yield();
582 }
583 //-----%%-----%%-----%%-----%%-----%%-----%%-----
584 //                                     BEEPING
585 //-----%%-----%%-----%%-----%%-----%%-----%%-----
586 void beep(byte Channel, int Frequency, int Time, int DutyCycle){
587     ledcWriteTone(Channel, Frequency);
588     ledcWrite(Channel, DutyCycle);
589     delay(Time);
590     ledcWriteTone(Channel, buzzer_freq);
591     ledcWrite(Channel, buzzer_vol);
592 } //beep
593
594
595
596 //-----%%-----%%-----%%-----%%-----%%-----%%-----
597 //                                     SD_INIT
598 //-----%%-----%%-----%%-----%%-----%%-----%%-----
599 /*
600 void SD_init(){
601     if (!SD.begin()){

```

```

602     #if DEBUG.FLAG
603         Serial.println(" Card Mount Failed");
604     #endif
605     return;
606 }
607 uint8_t cardType = SD.cardType();
608
609 if(cardType == CARD_NONE){
610     Serial.println("No SD card attached");
611     return;
612 }
613
614 Serial.print("SD Card Type: ");
615 if(cardType == CARD_MMC){
616     Serial.println("MMC");
617 } else if(cardType == CARD_SD){
618     Serial.println("SDSC");
619 } else if(cardType == CARD_SDHC){
620     Serial.println("SDHC");
621 } else {
622     Serial.println("UNKNOWN");
623 }
624
625 uint64_t cardSize = SD.cardSize() / (1024 * 1024);
626 Serial.printf("SD Card Size: %luMB\n", cardSize);
627 Serial.printf("Total space: %luMB\n", SD.totalBytes() / (1024 *
628 1024));
629 Serial.printf("Used space: %luMB\n", SD.usedBytes() / (1024 * 1024))
629 ;
629 } //SD_init
630 */
631
632 //-----%%-----%%-----%%-----%%-----%%-----%%-----
633 //                                     RTC_INIT
634 //-----%%-----%%-----%%-----%%-----%%-----%%-----
635 #if RTCLIB.COMP
636 void RTC_init(){
637     if (! rtc.begin()) {
638         Serial.println("Couldn't find RTC");
639         Serial.flush();
640         abort();
641     }
642 } // RTC_init
643 #endif
644
645 //-----%%-----%%-----%%-----%%-----%%-----%%-----
646 //                                     RTC.READ
647 //-----%%-----%%-----%%-----%%-----%%-----%%-----
648 //just copied from RTCLib example
649 void RTC.read(){
650     Serial.print("RTCway: ");
651     DateTime now = rtc.now();
652     Serial.print(now.year(), DEC);
653     Serial.print('/');
654     Serial.print(now.month(), DEC);
655     Serial.print('/');
656     Serial.print(now.day(), DEC);
657     Serial.print(" (");
658     Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);
659     Serial.print(") ");

```

```
660 Serial.print(now.hour());  
661 Serial.print(':');  
662 Serial.print((byte)(now.minute()));  
663 Serial.print(':');  
664 Serial.print((int)(now.second()));  
665 Serial.println();  
666 }//rtc_read
```

Listing 7.2: Código principal

REFERENCIAS

- [1] M. Maldonado, “El trabajo colaborativo en el aula universitaria.” *Laurus*, vol. 13, no. 23, pp. 263–278, 2007.
- [2] Wiley, “Digital Skills Gap Index 2021,” Tech. Rep., 2021. [Online]. Available: <https://dsgi.wiley.com/>
- [3] R. Corporation, “The global digital skills gap,” p. 41, 2021.
- [4] M. Intelligence, “Future of work report 2022,” Tech. Rep., 2021. [Online]. Available: https://learnmore.monster.com/future-of-work?_ga=2.3007350.712898892.1633625481-1953470270.1633625481
- [5] T. by PageGroup, “Talento TI - Competitividad STEM en América Latina,” Bogotá, Tech. Rep., 2021. [Online]. Available: <https://www.michaelpage.cl/estudios-y-tendencias/talento-ti>
- [6] Esp32-devkitc. [Online]. Available: <https://www.espressif.com/en/products/devkits/esp32-devkitc/overview>
- [7] I. N. de Estadística. Calculadora ipc. [Online]. Available: <https://calculadoraipc.ine.cl/>
- [8] D. Celis and R. González, “Aporte de la metodología steam en los procesos curriculares,” *Revista Redipe*, vol. 8, pp. 289–302, 2021. [Online]. Available: <https://revista.redipe.org/index.php/1/article/view/1405/1320>
- [9] B. Freeman, S. Marginson, and R. Tytler, *An international view of STEM education*, 08 2019.
- [10] S. Marginson, R. Tytler, B. Freeman, and K. Roberts, *STEM: Country comparisons*, 05 2013.
- [11] K. Thelwell. (2019) The role of stem in developing countries and potential benefits. Accessed: 05-Apr-2021. [Online]. Available: <https://borgenproject.org/stem-in-developing-countries/>
- [12] OECD, *Chile, in OECD Science, Technology and Innovation Outlook*. Paris: OECD Publishing, 2016. [Online]. Available: https://doi.org/10.1787/sti_in_outlook-2016-51-en
- [13] Corfo-FundaciónChile, “Preparando a Chile para la Sociedad del Conocimiento: Hacia una coalición que impulse la Educación STEAM,” Santiago, Tech. Rep., 2017. [Online]. Available: [content/uploads/2020/05/STEM.FCh_digital.pdf](content/uploads/2020/05/STEM.FCh_digital.pdf)

- [14] K. Rose, S. Eldridge, and L. Chapin, “The internet of things: An overview,” The Internet Society, Tech. Rep., 2015.
- [15] About ubidots. [Online]. Available: <https://ubidots.com/about>
- [16] ¿qué es mqtt? [Online]. Available: <https://aws.amazon.com/es/what-is/mqtt/#:~:text=El%20agente%20MQTT%20es%20el,mensaje%20y%20enviarles%20los%20mensajes>.
- [17] Easyeda. [Online]. Available: <https://easyeda.com/es>
- [18] D. W. White, “What is stem education and why is it important?” *Florida Association of Teacher Educators*, vol. 1, pp. 1–9, 2014. [Online]. Available: <http://www.fate1.org/journals/2014/white.pdf>
- [19] Unidad de Currículum y Evaluación, “Metodología de aprendizaje basado en proyectos,” *Ministerio de Educación de Chile*, 2019. [Online]. Available: https://www.curriculumnacional.cl/614/articles-140166_recurso_pdf.pdf.
- [20] R. C. Fernández and M. C. Romero, “Robótica y proyectos steam: Desarrollo de la creatividad en las aulas de educación primaria,” *Pixel-BIT Revista de Medios y Educación*, vol. 58, p. pp. 51–69, 2020. [Online]. Available: <https://recyt.fecyt.es/index.php/pixel/article/view/73672/49238>
- [21] Producto educativo: Sphero bolt. [Online]. Available: <https://sphero.com/collections/for-school/products/sphero-bolt-power-pack>
- [22] Producto educativo: Little bits. [Online]. Available: https://sphero.com/pages/littlebits?_pos=2&_sid=d85fd19f6&_ss=r
- [23] Familia de productos arduino mkr. [Online]. Available: <https://www.arduino.cc/en/hardware#mkr-family>
- [24] Comparación de los microcontroladores espressif. [Online]. Available: https://www.electroimc.com/es/doc/proyectos/microcontroladores/81_tabla-comparacion-microcontroladores-espressif-esp8266-esp32-esp32c6
- [25] Características del material fr-4 para el diseño de pcb. [Online]. Available: <https://resources.altium.com/es/p/fr4>
- [26] Una lectura obligada: Guía de materiales de pcb. [Online]. Available: <https://www.mokotechnology.com/es/pcb-material/>
- [27] Cómo elegir el material dieléctrico del pcb. [Online]. Available: <https://www.proto-electronics.com/es/blog/cmo-elegir-el-material-dielctrico-del-pcb>
- [28] Z. Peterson. What pcb copper thickness should you use? [Online]. Available: <https://resources.altium.com/p/what-pcb-copper-thickness-should-you-use>
- [29] Hot air solder leveling. [Online]. Available: https://en.wikipedia.org/wiki/Hot_air_solder_leveling
- [30] Electroless nickel immersion gold. [Online]. Available: https://en.wikipedia.org/wiki/Electroless_nickel_immersion_gold

[31] Arduino esp32 support page. [Online]. Available: https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html#

COPYRIGHT

Some sections and figures of this document are protected by copyright laws. Please contact the Universidad Técnica Federico Santa María, UTFSM or Guillermo H. Becerra O. prior to any non-personal use of this material.