UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

Diseño e implementación de prototipo IoT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web

Trabajo de titulación previo a la obtención del

Título de Ingeniero Electrónico

AUTORES: IVÁN ANDRÉS MACIAS MORA

JAVIER ANTONIO LOOR BARREZUETA

TUTOR: PhD. VICTOR HUILCAPI SUBÍA

Guayaquil – Ecuador

2024

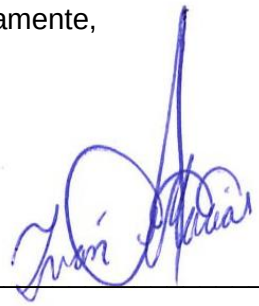
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Iván Andrés Macias Mora con documento de identificación N° 0917387854 y Javier Antonio Loor Barrezueta con documento de identificación N° 0922081583; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 6 de julio de 2024

Atentamente,



Iván Andrés Macias Mora
0917387854

Javier Antonio Loor Barrezueta
0922081583

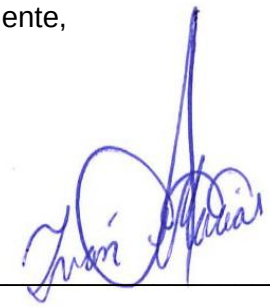
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Iván Andrés Macias Mora con documento de identificación N° 0917387854 y Javier Antonio Loor Barrezueta con documento de identificación N° 0922081583, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del proyecto de investigación: “Diseño e implementación de prototipo Iot para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web”, el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 6 de julio de 2024

Atentamente,



Iván Andrés Macias Mora
0917387854

Javier Antonio Loor Barrezueta
0922081583

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Victor Huilcapi Subía con documento de identificación N°0917206294, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE PROTOTIPO IOT PARA EL MONITOREO DE PARÁMETROS AMBIENTALES Y DE CONSUMO DE ENERGÍA APLICADOS A DATACENTERS UTILIZANDO HARDWARE DE BAJO COSTO Y SERVICIOS WEB, realizado Iván Andrés Macias Mora con documento de identificación N° 0917387854 y Javier Antonio Loor Barrezueta con documento de identificación N° 0922081583, obteniendo como resultado final el trabajo de titulación bajo la opción tesis que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 6 de julio de 2024

Atentamente,

PhD. Victor Huilcapi Subía
CI: 0917206294

AGRADECIMIENTO

El principal agradecimiento a Dios que me ha guiado y me dio la fortaleza, sabiduría y capacidad para alcanzar esta meta. A mis padres que han sabido formarme con buenos valores y hábitos, gracias por apoyarme a lo largo de mi carrera para seguir adelante y tenga un mejor futuro. A mis hermanos por sus estímulos y buenos concejos en mis años de estudio. A mi esposa por su apoyo y amor incondicional, creer en mi capacidad, por sus palabras de aliento para que no decaiga y cumpla mis ideales.

El camino no ha sido sencillo hasta ahora, pero gracias a todos por sus aportes y fomentar en mí el deseo de superación, hoy estoy convirtiendo este sueño en realidad.

IVÁN ANDRÉS MACÍAS MORA

AGRADECIMIENTO

Quiero expresar mi más profundo agradecimiento primero a Dios por iluminar mi camino guiar mis pasos y darme las bendiciones e inteligencia necesaria para culminar este proyecto de tesis y sobre todo darme el valor y fuerza en cada momento para seguir adelante hasta poder hacer este sueño realidad.

Gracias infinitas también a mis padres por su apoyo incondicional y moral y sobre todo por su fe en mí, Ellos han sido un pilar fundamental en este logro por estar ahí incluso en los momentos más difíciles.

Y a mi amada esposa la cual su ayuda fue fundamental porque estuvo conmigo incluso hasta en los momentos más turbulentos. Y es que este proyecto no fue nada fácil, pero estuvo motivándome y ayudándome hasta donde sus alcances se lo permitían.

Sin ellos todo esto no habría sido posible, su amor, comprensión, guía y sacrificio han sido la luz que guio mi camino a través de este viaje académico.

JAVIER ANTONIO LOOR BARREZUETA

RESUMEN DEL PROYECTO

Año	Alumnos	Tutor de Proyecto de titulación	Proyecto de titulación
2024	Iván Andrés Macias Mora y Javier Antonio Loor Barrezueta	PhD. Victor Huilcapi Subía	Diseño e implementación de prototipo IoT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web.

El principal objetivo de este proyecto de titulación es el diseño e implementación de prototipo IoT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web. Con el desarrollo de este tipo de prototipos se abre el camino para futuras investigaciones que pueden ser desarrolladas por estudiantes, docentes e investigadores de la carrera de ingeniería en electrónica, en la cual se puede desarrollar aplicaciones de IoT enfocadas en el equipamiento informático y electrónico, específicamente en los datacenters.

Se desarrolló un prototipo IoT que consiste en varios sensores de monitoreo de parámetros ambientales como temperatura, humedad de ambiente, humedad de suelo, y otros parámetros importantes como el consumo eléctrico, estos parámetros son importantes para un correcto funcionamiento y evitar interrupciones en las operaciones y servicios, prevenir sobrecalentamientos de los componentes, incendios e inundaciones, que hagan que el datacenter deje de funcionar y así evitar grandes pérdidas económicas.

Para desarrollar este prototipo IoT se hizo uso de un microcontrolador ESP32 el cual se encargará de procesar todos los datos que se irán obteniendo gracias a los sensores, dichos datos serán enviados a través de una red inalámbrica y el uso de un servidor MQTT, serán almacenados y tratados en servidores web en la nube como Ubidots. Se realiza la evaluación y análisis de los diferentes datos obtenidos de los sensores y mediante un sistema vía mails de Ubidots alertar de acuerdo con los rangos con los que se esté manejando cada sensor.

Como conclusión del trabajo de esta investigación los estudiantes de la carrera de ingeniería en electrónica de la Universidad Politécnica Salesiana sede Guayaquil, profesores e investigadores del área de electrónica serán beneficiados ya que tienen a su disposición un prototipo IoT en el cual podrán realizar prácticas de laboratorio y plantearán sus conocimientos teóricos del internet de las cosas aplicadas a la ingeniería.

ABSTRACT

Year	Students	Degree Project Tutor	Technical Degree Project
2024	Iván Andrés Macias Mora y Javier Antonio Loor Barrezueta	PhD. Víctor Huilcapi Subía	Design and implementation of an IoT prototype for monitoring environmental parameters and energy consumption applied to datacenters using low-cost hardware and web services

The main objective of this degree project is the design and implementation of IoT prototype for the monitoring of environmental and energy consumption parameters applied to datacenters using low-cost hardware and web services. With the development of this type of prototypes, the way is opened for future research that can be developed by students, teachers, and researchers of the electronics engineering career, in which IoT applications can be developed focused on computer and electronic equipment, specifically in datacenters.

An IoT prototype was developed that consists of several monitoring sensors of environmental parameters such as temperature, ambient humidity, soil moisture, and other important parameters such as electricity consumption, these parameters are important for proper operation and avoid interruptions in operations and services, prevent overheating of components, fires and floods, which make the datacenter stop working and thus avoid large economic losses.

To develop this IoT prototype, an ESP32 microcontroller was used, which will be responsible for processing all the data that will be obtained thanks to the sensors, such data will be sent through a wireless network and the use of an MQTT server, will be stored and treated in web servers in the cloud such as Ubidots. The evaluation and analysis of the different data obtained from the sensors is conducted and through a system via Ubidots emails alert according to the ranges with which each sensor is being handled.

As a conclusion of the work of this research, the students of the electronics engineering career of the Salesian Polytechnic University Guayaquil headquarters, professors and researchers in the area of electronics will benefit since they have at their disposal an IoT prototype in which

they will be able to carry out laboratory practices and will raise their theoretical knowledge of the internet of things applied to engineering.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los centros de datos de la información o datacenters son los espacios adecuados donde se sitúan los equipos de telecomunicaciones como routers, switches, servidores, cables de fibra óptica, cableado estructurado, fuentes de poder, baterías, generadores, lo cual representa la estructura medular del área de sistemas o telecomunicaciones en una empresa.

El teletrabajo en épocas de pandemia demostró los desafíos de TI que enfrentan las empresas en situaciones de emergencia. Con este fin, los datacenters se han convertido en el centro de la infraestructura comercial y se planean varias actualizaciones a corto y mediano plazo. Gracias a la conectividad, la economía global puede seguir creciendo, por lo que cualquier esfuerzo actual, como la realización del prototipo propuesto, servirá para aumentar la disponibilidad y el procesamiento de datos, lo cual será prioridad máxima.

Con el desarrollo de sistemas basados en el internet de las cosas donde se utilice sensores para la monitorización de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web, como son: temperatura, humedad de ambiente, humedad de suelo, y otros parámetros importantes como el consumo eléctrico, se logra asegurar su correcto funcionamiento y evitar interrupciones en las operaciones y servicios, prevenir sobrecalentamientos de los componentes, incendios e inundaciones, etc.

En relación a este tema, se propuso un prototipo IoT de monitorización de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web que sea utilizado como objeto de práctica y experimentación a los futuros ingenieros electrónicos de la carrera de ingeniería en electrónica de la Universidad Politécnica Salesiana sede Guayaquil, con la finalidad de realizar prácticas de aplicación de IoT y seguir desarrollando e innovando hacia el futuro.

1 El problema

1.1 Descripción del problema

Muchas empresas medianas y grandes disponen de datacenters para que sus negocios funcionen sin inconvenientes en su parte operativa y técnica, en ocasiones tienen escasos sistemas de monitorización ambiental o de consumo energético dentro de ellos, causando pérdidas económicas considerables, debido que uno de sus servidores o equipos deje de funcionar por mal funcionamiento climático o de energía, o algún tipo de sobrecalentamiento de los componentes.

El control adecuado de los parámetros ambientales como temperatura, humedad de ambiente, humedad de suelo, y otros parámetros importantes como el consumo eléctrico, control de luminarias y acceso al datacenter son importantes monitorizar constantemente y de ser posible disponer de un sistema de alerta que indique vía correo electrónico algún mal funcionamiento a nivel eléctrico o ambiental dentro de un datacenter, para que los tiempos de respuesta para correcciones y solución sea más efectiva.

1.2 Antecedentes

La necesidad de monitorizar los datacenters en empresas medianas y grandes para un adecuado y mejor control de las diferentes variables (temperatura, humedad de ambiente, humedad de suelo, consumo eléctrico, control de luminarias) incentivó al desarrollo de este prototipo IoT de bajo costo, el cual mediante sensores y electrónica se pueden monitorizar que los parámetros ambientales y de consumo de energía en los datacenters sea el correcto.

Con la realización de este proyecto se puede lograr una mejora a nivel de tecnología dentro de los diferentes datacenters en las pequeñas empresas, y también sirve como guía de aprendizaje para los futuros ingenieros electrónicos de la Universidad Politécnica Salesiana sede Guayaquil y así seguir investigando, desarrollando e implementando, para que el país alcance un mejor estatus tecnológico.

1.3 Importancia y alcance

Con esta propuesta, las empresas en general se benefician de estudios y prototipos realizados por estudiantes e investigadores de la carrera de ingeniería electrónica de la Universidad Politécnica de Salesiana sede Guayaquil para monitorear datacenters o centro de datos, al igual que se benefician los estudiantes de la carrera de ingeniería electrónica donde ponen en práctica sus conocimientos adquiridos en diversas disciplinas utilizando el Internet de las Cosas.

En este contexto, se presentó este trabajo de investigación, que permite a los datacenters utilizar métodos de monitoreo para controlar la temperatura, la humedad ambiental, la humedad del suelo, el consumo de energía, la iluminación y otros parámetros ambientales

1.4 Delimitación

1.4.1 Delimitación temporal

El tiempo estimado para el diseño, implementación y pruebas de funcionamiento del prototipo es de 6 meses, desde agosto 2023 hasta enero del 2024.

1.4.2 Delimitación espacial

Las pruebas del sistema IoT de monitoreo ambiental para datacenters se realizó en primera fase en el domicilio del Sr. Iván Macías autor del proyecto de tesis, en la ciudad de Guayaquil.

Posteriormente el prototipo se entregó al laboratorio de la carrera de ingeniería electrónica de la Universidad Politécnica Salesiana sede Guayaquil para el uso de estudiantes de dicha carrera.

El testeo y análisis de los resultados se lo realiza en tiempo real de acuerdo con el diseño del prototipo IoT, la información que es obtenida mediante los sensores que se apliquen en el prototipo son administrados vía web en servidores de Ubidots y controlados mediante acceso remoto a través del internet.

1.4.3 Delimitación académica

Este trabajo de investigación donde el resultado es un prototipo IoT funcional y operativo es una herramienta fundamental para el desarrollo de las capacidades y destrezas de los futuros ingenieros electrónicos de la Universidad Politécnica Salesiana sede Guayaquil. Con este trabajo se pone en práctica los conocimientos adquiridos en materias como redes de computadora, electrónica digital, redes inalámbricas, comunicaciones digitales y medios de transmisión.

1.5 Beneficiarios de la propuesta

Este proyecto tiene como beneficiarios directos a las empresas que tienen datacenter y carezcan de sistemas de monitorización ambiental y energético, por lo general en pequeñas y medianas empresas no invierten en estas soluciones por sus altos costos de implementación, de operación y mantenimiento. También, se ven beneficiados los estudiantes de la carrera de ingeniería electrónica y de telecomunicaciones de la Universidad Politécnica Salesiana sede Guayaquil, puesto que se entrega una maqueta con los sensores para el desarrollo de prácticas.

1.6 Propuesta de solución

Se propuso un prototipo IoT para la monitorización ambiental como temperatura ambiente, humedad relativa, humedad de suelo, sensor de luz y sensores para monitorización de consumo energético para prototipo IoT aplicado a datacenters.

La propuesta del prototipo se realiza mediante el microcontrolador ESP32 el cual procesa todos los datos obtenidos mediante los sensores y serán enviados a través de una red inalámbrica al internet para que estos datos sean almacenados y enviados a servicios web en la nube como Ubidots. También que se alerta vía mail cuando los parámetros ambientales o energéticos salgan de rango y afecten a la funcionalidad de algún equipo del datacenter.

1.7 Innovación e impacto del proyecto

El desarrollo de este trabajo de titulación tiene un nivel de innovación alto, debido a la necesidad de evitar desconexiones de servicios, desperfectos en los equipos de telecomunicaciones dentro de un datacenter en las pequeñas y medianas empresas hace que este proyecto de diseño e implementación IoT pueda efectuarse para un uso a nivel comercial e industrial en el futuro.

1.8 Objetivos

1.8.1 Objetivo general

Diseñar e implementar un prototipo IOT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web.

1.8.2 Objetivo específico

- Diseñar una red inalámbrica para realizar la conexión del ESP32 y los sensores para la monitorización de parámetros ambientales y de control de consumo de energía en un datacenter aplicado al laboratorio de telecomunicaciones de la Universidad Politécnica Salesiana sede Guayaquil.

- Implementar sensores para monitorización ambiental como temperatura ambiente, humedad relativa, humedad de suelo, sensor de luz, y sensor para el consumo eléctrico para prototipo IoT aplicado a la monitorización de datacenters simulado en laboratorio de telecomunicaciones.

- Implementar servicios en la nube para el almacenamiento y monitoreo web de la información obtenida por los sensores mediante Ubidots.

- Alertar vía mail cuando los parámetros ambientales o energéticos salgan de rango y afecten a la funcionalidad de algún equipo del datacenter.

2 Fundamentos teóricos

2.1 Datacenters o Centro de Datos

Los centros de datos son ubicaciones físicas que las organizaciones utilizan para albergar aplicaciones y datos. Los centros de datos se construyen alrededor de una red de recursos informáticos y de almacenamiento para permitir aplicaciones compartidas y transmisión de datos. Los componentes principales del diseño de un centro de datos son enrutadores, conmutadores, plataformas, sistemas de almacenamiento, servidores y servidores de aplicaciones.

Los centros de datos modernos son muy diferentes a lo que eran no hace mucho tiempo. La infraestructura ha pasado de los servidores físicos locales tradicionales a las redes virtuales, que admiten aplicaciones y cargas de trabajo en clústeres de infraestructura física y entornos de múltiples nubes.

En esta era, los datos existen y están conectados a través de múltiples centros de datos, bordes y nubes públicas y privadas. El centro de datos debe poder comunicarse entre todos estos múltiples sitios (locales y en la nube). Incluso una nube pública es un conjunto de centros de datos. Cuando las aplicaciones se alojan en la nube, utilizan los recursos del centro de datos del proveedor de la nube (Cisco.com, 2024).

.

En TI empresarial, los centros de datos están diseñados para respaldar aplicaciones y actividades comerciales, que incluyen:

- Correo electrónico y uso compartido de archivos
- aplicaciones de productividad
- Gestión de relaciones con el cliente (CRM)
- Planificación de recursos empresariales (ERP) y bases de datos
- Big data, inteligencia artificial y aprendizaje automático
- Servicios de escritorio virtual, comunicación y colaboración (Cisco.com, 2024).

El diseño del centro de datos incluye enrutadores, conmutadores, firewalls, sistemas de almacenamiento, servidores y controladores de entrega de aplicaciones. Dado que estos componentes almacenan y administran aplicaciones y datos comerciales críticos, la seguridad del centro de datos es fundamental en el diseño del centro de datos. Juntos proporcionan la siguiente funcionalidad:

La infraestructura de red conecta servidores (físicos y virtualizados), servicios de centro de datos, almacenamiento y conectividad externa a las ubicaciones de los usuarios finales.

Infraestructura de almacenamiento. Los datos son el combustible del centro de datos moderno. Se utilizan sistemas de almacenamiento para albergar este valioso producto.

Recursos informáticos. Las aplicaciones son el motor del centro de datos. Estos servidores proporcionan potencia de procesamiento, memoria, almacenamiento local y conectividad de red para alimentar las aplicaciones (Cisco.com, 2024).
.

2.1.1 ¿Cómo funcionan los centros de datos?

Los servicios del centro de datos generalmente se implementan para proteger el rendimiento y la integridad de los componentes esenciales del centro de datos.

Equipos de ciberseguridad. Estos incluyen firewalls y protección contra intrusiones para proteger los centros de datos.

Garantía de entrega de la aplicación. Para mantener el rendimiento de las aplicaciones, estos mecanismos proporcionan recuperación y disponibilidad de las aplicaciones mediante conmutación por error automática y equilibrio de carga (Cisco.com, 2024).
.

2.1.2 ¿Qué hay dentro de las instalaciones de un centro de datos?

Los componentes del centro de datos requieren una infraestructura significativa para soportar el hardware y el software del centro. Estos incluyen subsistemas de energía, sistemas de energía ininterrumpida (UPS), ventilación, sistemas de enfriamiento, sistemas de extinción de incendios, generadores de respaldo y conexiones a redes externas (Cisco.com, 2024).

.

2.1.3 ¿Cuáles son los estándares de infraestructura del centro de datos?

El estándar más ampliamente adoptado para el diseño y la infraestructura de centros de datos es ANSI/TIA-942. Incluye estándares de certificación de cumplimiento ANSI/TIA-942 para garantizar el cumplimiento de una de las cuatro categorías de niveles del centro de datos clasificadas por niveles de redundancia y tolerancia a fallas.

Nivel 1: Infraestructura del sitio. Los centros de datos de nivel 1 brindan protección limitada contra eventos físicos. Tiene un único componente de capacidad y una única ruta de distribución no redundante.

Nivel 2: Infraestructura del sitio de componentes con redundancia. El centro de datos proporciona una protección mejorada contra eventos físicos. Tiene componentes de capacidad redundantes y una única ruta de distribución no redundante.

Nivel 3: Al mismo tiempo se protegen las estructuras del sitio. Estos centros de datos se encargan de todos los eventos físicos proporcionando componentes de alta capacidad y canales de distribución especiales. Cada componente se puede quitar o reemplazar sin interrumpir el servicio al usuario final. Nivel 4: la infraestructura del sitio es defectuosa. Estos centros de datos proporcionan altos niveles de tolerancia a fallas y redundancia. Se pueden mantener simultáneamente componentes adicionales y múltiples líneas de distribución independientes y las fallas individuales pueden extenderse por todo el edificio sin averías (Cisco.com, 2024).

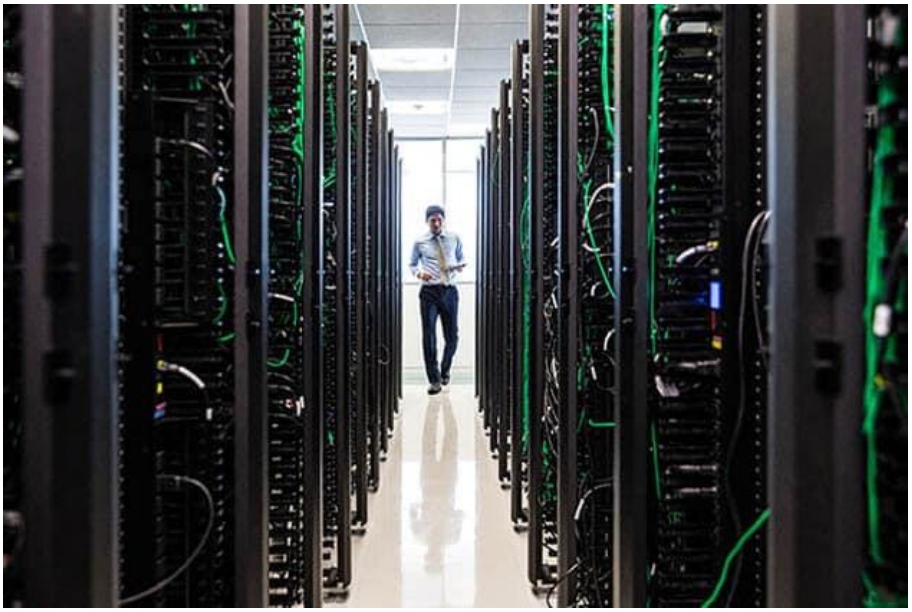


Figura 1 Datacenter
(Cisco.com, 2024)

2.2 IoT

Los conceptos centrados en Internet de las cosas (IoT), como la realidad aumentada, la transmisión de vídeo de alta resolución, los vehículos autónomos, el entorno inteligente, la atención sanitaria electrónica, etc., tienen ahora una presencia omnipresente. Estas aplicaciones requieren velocidades de datos más altas, gran ancho de banda, mayor capacidad, baja latencia y alto rendimiento. A la luz de estos conceptos emergentes, IoT ha revolucionado el mundo al proporcionar una conectividad perfecta entre redes heterogéneas (HetNets). El objetivo final de IoT es introducir la tecnología plug and play que proporcione al usuario final facilidad de operación, control de acceso remoto y capacidad de configuración (Shafique et al., 2020).

El término IoT se tornó relevante por primera vez en 1999 por Kevin Ashton. El término IoT gira en torno a la palabra "inteligencia". Es decir, la capacidad de adquirir y aplicar conocimientos de forma independiente. Por lo tanto, IoT se refiere a "cosas o dispositivos y sensores" autónomos que son inteligentes, direccionables de manera única en función de protocolos de comunicación, adaptables y con seguridad inherente. IoT se caracteriza por

tres visiones. Orientado a Internet: Esta visión se centra en las conexiones entre objetos. Orientado a las cosas: La visión se enfoca en objetos comunes. Orientado al conocimiento: La visión se enfoca en cómo se representa, almacena y organiza la información. Estas visiones allanaron el camino para la visión de IoT de la Unión Internacional de Telecomunicaciones (UIT). Todas las conexiones están ahora establecidas. En definitiva, el objetivo final es “conectar y utilizar objetos inteligentes”.

IoT ofrece muchas oportunidades comerciales que permiten a las empresas crear nuevas estrategias y modelos comerciales para implementar el concepto. No sólo oportunidades de negocio, sino también la realización de oportunidades de investigación eficientes e ingeniosas para académicos e investigadores de campos multidisciplinarios. Por lo tanto, combina estudios empresariales, habilidades de ingeniería, ciencias y humanidades, todo bajo un mismo paraguas. Además, IoT transforma el mundo en un mundo inteligente, donde todo es fácilmente accesible en menos tiempo y esfuerzo. En la figura 2 se muestra un diagrama de bloques que muestra aplicaciones inteligentes basadas en IoT y beneficios de IoT como tecnología de vanguardia.

En general, las empresas adoptarán una inversión inmediata o un enfoque de esperar y ver en función del nivel de madurez de las tecnologías de IoT específicas. Durante la fase de despliegue inicial, se debe dejar un margen para adaptarse a los cambios. Para ello, se deberían proponer soluciones de IoT abiertas basadas en software y hardware. En este sentido, una forma de reducir costes es utilizar teléfonos inteligentes que sirvan como nodos de IoT (Ashton, 2010).

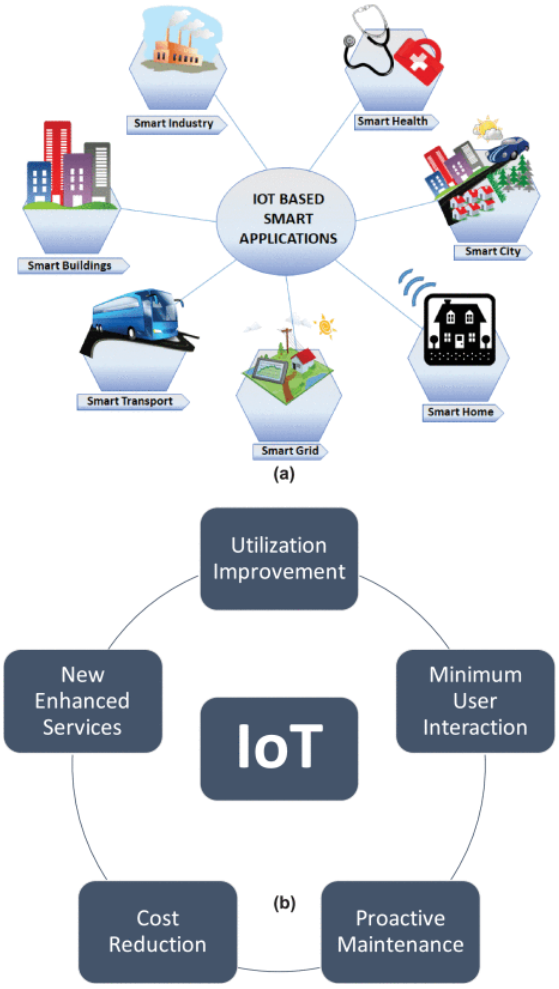


Figura 2 IoT

(Shafique et al., 2020)

2.3 Ubidots

Ubidots es una plataforma en la nube muy utilizada en proyectos de IoT. Es un proveedor de servicios en la nube para proyectos de IoT o dispositivos de IoT. Ubidots es una plataforma de Internet de las cosas (IoT) que permite a los innovadores y a las industrias crear prototipos y escalar proyectos de IoT a producción. Usamos la plataforma de Ubisoft para enviar datos desde nuestro dispositivo a la nube (Tiwari et al., 2022).

Ubidots es un proyecto que permite a miles de empresas y emprendedores crear soluciones IoT sin desarrollar software para el monitoreo remoto de sus dispositivos en campo. Ubidots logra esto proporcionando cuatro herramientas, comenzando con los paneles: donde la información creada por las herramientas en el campo cobra vida. Utilice gráficos, tablas, métricas, mapas y otras herramientas de visualización para gestionar tareas importantes desde paneles que se pueden crear con solo un clic (Es.ubidots.com, 2024).

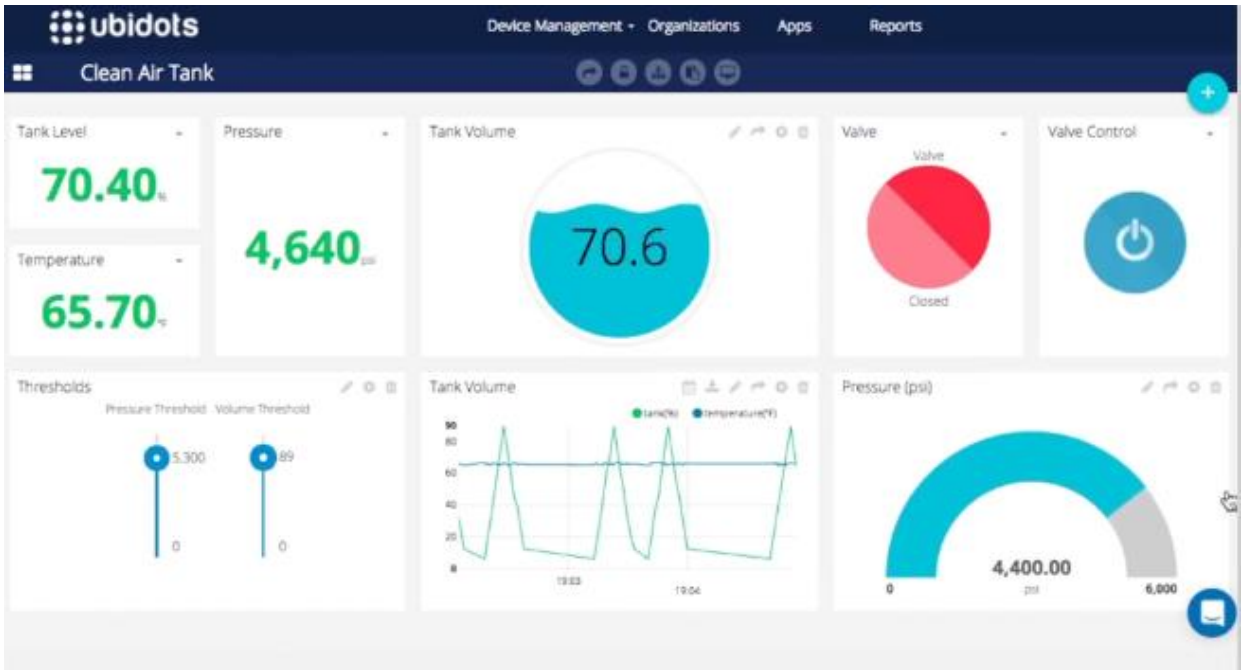


Figura 3 Interfaz de Ubidots
(Es.ubidots.com, 2024)

2.4 MQTT

MQTT (Message Queuing Telemetry Transport). Es un protocolo ligero pensado para la transmisión de pequeñas cantidades de información, fue desarrollo por Andy Stanford-Clark de IBM y Arlen Nipper de Arcom en 1999, y desde el 2014 es un estándar abierto.

MQTT es un protocolo M2M lo que quiere decir que está pensado para el intercambio de información entre dispositivos sin interacción humana. Requiere un ancho de banda bajo, pensado para funcionar sobre conexiones de baja calidad en dispositivos de bajo consumo energético, como por ejemplo diferentes sensores autónomos alimentados por pilas. Estas y más características hacen que MQTT sea un protocolo adecuado para IoT.

Se ejecuta sobre TCP/IP utilizando una topología PUSH/SUBSCRIBE. Existen dos tipos de sistemas: clientes y brókeres. Un bróker es el servidor con el que se comunican los clientes, recibe comunicaciones de unos y se las envía a otros. Los clientes no se comunican directamente entre sí, sino que se conectan con el bróker. Cada cliente puede ser un editor, un suscriptor o ambos. Este protocolo está controlado por eventos, lo que quiere decir que; la transmisión de datos no se realiza de manera periódica o continua. De esta manera el volumen de transmisión se mantiene mínimo. Un cliente sólo publica cuando hay información para enviar, y un bróker sólo envía información a los suscriptores cuando llegan nuevos datos.

En la figura 4 se puede apreciar la estructura del sistema. Los clientes se registran en el bróker y establecen comunicación basada en un mecanismo de publicación y suscripción de mensajes clasificados por temas (topics). Cuando el bróker recibe un mensaje de un determinado tema, lo distribuye entre todos los clientes suscritos a dicho tema. Los clientes registrados en el broker pueden publicar temas, suscribirse a uno o varios temas o publicar unos temas y suscribirse a otros, siendo el desarrollador quien define la estructura de la comunicación. Los mensajes tienen dos partes: el topic tema, que es el nombre con el que se identifica el mensaje, y la carga o payload que es la información que se incluye en el mensaje (PIZARRO PELÁEZ, 2020).

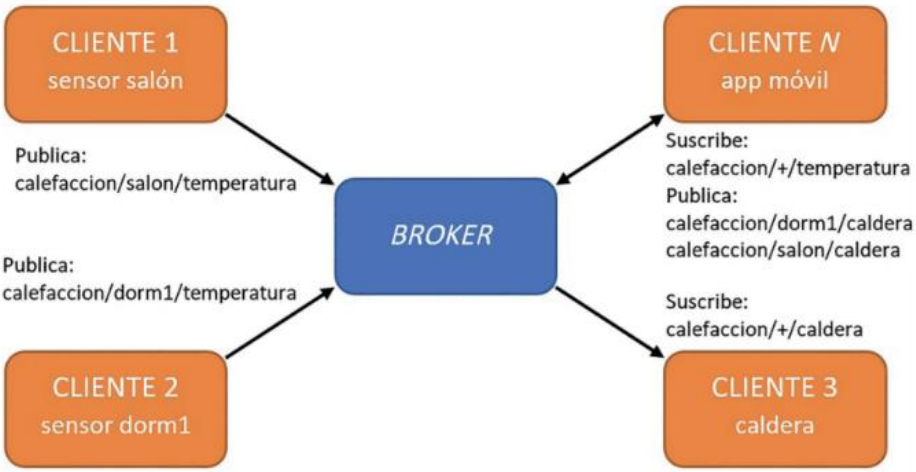


Figura 4 Arquitectura MQTT
(PIZARRO PELÁEZ, 2020)

2.5 Esp32

La conectividad inalámbrica a IoT mediante Wi-Fi o Bluetooth se ha hecho relativamente simple utilizando módulos y kits ESP32. El ESP32 es un SoC, que ofrece conectividad Wifi y Bluetooth de 2,4 GHz de frecuencia, está diseñado para una amplia gama de aplicaciones. Capaz de funcionar entornos industriales, puede adaptarse a los cambios en las condiciones externas. Está diseñado para dispositivos móviles, dispositivos electrónicos portátiles y aplicaciones IoT, logra un consumo de energía bajo. Está integrado con interruptores de antena incorporados, balun de RF, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía.

ESP32 es capaz de funcionar de forma fiable en entornos industriales, con una temperatura de funcionamiento que oscila entre –40 °C y +125 °C. Alimentado por circuitos de calibración avanzados, ESP32 puede eliminar dinámicamente imperfecciones de circuitos externos y adaptarse a cambios en las condiciones externas.

Diseñado para dispositivos móviles, electrónica portátil y aplicaciones de IoT, ESP32 logra un consumo de energía ultra bajo con una combinación de varios tipos de software propietario. ESP32 también incluye características de última generación, como sincronización de reloj de grano fino, varios modos de energía y escalado dinámico de energía.

ESP32 está altamente integrado con interruptores de antena incorporados, balun RF, amplificador de potencia, amplificador de recepción de bajo ruido, filtros y módulos de administración de energía. ESP32 agrega funcionalidad y versatilidad invaluables a sus aplicaciones con requisitos mínimos de placa de circuito impreso (PCB).

ESP32 puede funcionar como un sistema independiente completo o como un dispositivo esclavo de una MCU host, lo que reduce la sobrecarga de la pila de comunicaciones en el procesador de aplicaciones principal. ESP32 puede interactuar con otros sistemas para proporcionar funcionalidad Wi-Fi y Bluetooth a través de sus interfaces SPI/SDIO o I2C/UART (espressif.com, 2024).

Figura 5 Modulo ESP32
(espressif.com, 2024).

2.6 Sensores Ambientales

Los sensores ambientales realizan mediciones de alta precisión de diferentes parámetros climáticos, como, temperatura, humedad, calidad el aire, entre otros. Son de uso esencial para la toma de acciones preventivas y correctivas en diferentes áreas de interés, las más comunes son áreas verdes o zonas escolares (iotsens.com, 2024)

2.7 SNMP

El Protocolo simple de administración de red (SNMP) es un protocolo de red que se utiliza para la administración y monitoreo de todos los dispositivos conectados en redes de protocolo estándar de Internet. Recientemente, se ha mejorado la utilización y capacidad de SNMP para poder manejar el intercambio de datos a través de una red desde el nivel de control hasta el nivel de interfaz hombre-máquina. Algunos controladores, incluidos los microcontroladores, han actualizado sus funciones y cobertura para incluir SNMP en su protocolo de comunicaciones. El uso de SNMP también es compatible con el servidor Open Platform Communications (OPC), que desempeña un papel importante como servidor de intercambio de datos para varios controladores o sistemas con diferentes puertas de enlace y controladores (Nurmalia et al., 2023).

En el contexto de la informática industrial, la utilización de SNMP como protocolo de comunicación de datos para facilitar el intercambio de información entre dispositivos de campo, servidores OPC [2] e interfaces hombre-máquina (HMI) está motivada por varios factores. , particularmente en la era de la Industria 4.0. SNMP ofrece un medio estandarizado y confiable para acceder, monitorear y administrar dispositivos en un entorno industrial en red. Con su capacidad para recuperar datos específicos de dispositivos utilizando direcciones IP, SNMP permite una integración e interoperabilidad perfectas entre varios componentes de una configuración industrial. Al emplear SNMP, los profesionales de la industria pueden establecer una infraestructura de comunicación sólida y eficiente que admita procesos de monitoreo, control y toma de decisiones en tiempo real. Este protocolo ha demostrado ser invaluable para permitir la administración remota de dispositivos, la resolución de problemas y la recopilación de datos, lo que contribuye a mejorar la eficiencia operativa, el mantenimiento predictivo y la

productividad general mejorada en sistemas industriales alineados con los principios de la Industria 4.0 (Nurmalia et al., 2023).

Los diferentes protocolos de comunicación de datos tienen sus propias ventajas y desventajas, que deben considerarse cuidadosamente en función de los requisitos y características específicos del sistema industrial. Un protocolo ampliamente utilizado es SNMP, que ofrece simplicidad y facilidad de implementación. Proporciona capacidades eficientes de gestión de red, incluida la supervisión, configuración y detección de fallos de dispositivos. Sin embargo, SNMP puede no ser adecuado para entornos con recursos muy limitados debido a su sobrecarga y limitaciones de escalabilidad. Por otro lado, protocolos como Message Queuing Telemetry Transport (MQTT), son livianos y están diseñados para aplicaciones de bajo consumo y ancho de banda, lo que los hace ideales para la implementación de IoT (Nurmalia et al., 2023).

3 Marco metodológico

3.1 Tipo de investigación

El proyecto de tesis se basa en una investigación exploratoria y experimental. Exploratorio porque el problema no ha sido estudiado antes, sugiriendo que apenas está comenzando, con el objetivo de comprenderlo mejor y responder a las preguntas qué, por qué y cómo. Experimental, porque los datos se obtendrán a través de experimentos, en este caso particular se obtendrán a través de sensores, el propósito es la prevención en tiempo real y alertar al usuario final.

3.2 Diseño de investigación

El diseño de la investigación es experimental ya que se va a establecer una relación entre causa y efecto del problema. Por ejemplo, se monitorea el efecto de un parámetro ambiental como la temperatura sobre el correcto funcionamiento de los equipos de cómputos y electrónicos en el datacenter. Es un diseño muy práctico, ya que contribuye a la resolución de un problema.

3.3 Enfoque de la investigación

El enfoque dentro de esta investigación es mixto, es decir tanto cualitativo como cuantitativo. Cualitativo, ya que se realiza una recopilación de información de diferentes fuentes, como, libros, artículos científicos, tesis de grado, entre otras publicaciones, y luego de esto se realiza un análisis cualitativo de la información obtenida. Cuantitativo, porque, con la recopilación de datos obtenidos a través de los diferentes sensores se realizarán estadísticas y se determinaran los rangos en los que su funcionamiento es correcto, caso contrario, se dará a conocer el error.

3.4 Metodología de investigación

Se realiza diseño e implementación de prototipo IoT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo y servicios web, luego de la construcción del prototipo se puso a prueba la maqueta por varios días, esto se lo realiza con el fin obtener una cantidad de datos adecuada y poder continuar con el siguiente paso.

El siguiente paso es analizar los datos obtenidos por los sensores de humedad, temperatura y humedad de suelo, consumo energético, mismos que a través de un microcontrolador como el ESP32 y WIFI serán enviados a Ubidots con el fin de ser procesados y gestionados de manera tal que se pueda recibir alertas vía mail que sirvan en la ayuda de toma de decisiones para la monitorización y prevención adecuada de los equipos que estén dentro del datacenter.

3.4.1 Configuración de acceso a Ubidots:

Para acceder a la plataforma de Ubidots desde el ESP32 se requiere de los siguientes pasos:

1. Crear cuenta, ya sea gratuita o de pago.
2. Añadir al proyecto (Vscode, PlatformIO), la librería de conexión MQTT para ESP32 https://github.com/ubidots/esp32-mqtt
3. Copiar TOKEN de la plataforma y añadirla al código.

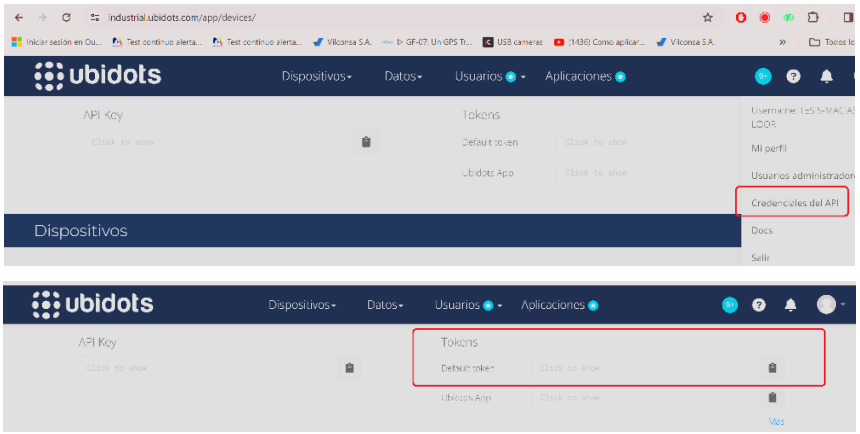


Figura 6 Configuración para ingreso a Ubidots

4. Nombrar al dispositivo y agregar todas las variables requeridas en el código tal y cual se mostrarán el apartado de dispositivos/datos.

```
const char *UBIDOTS_TOKEN = "BBFF-5QU3hWY5QpFiDYOKGAOSMBNXHhTq7J";
const char *DEVICE_LABEL = "esp32-tesis";
const char *VARIABLE_LABEL_TEMP1 = "temperatura_area1";
const char *VARIABLE_LABEL_TEMP2 = "temperatura_area2";
const char *VARIABLE_LABEL_HUMD1 = "humedad_area1";
const char *VARIABLE_LABEL_HUMD2 = "humedad_area2";
const char *VARIABLE_LABEL_CURRT1 = "corriente_rack1";
const char *VARIABLE_LABEL_CURRT2 = "corriente_rack2";
const char *VARIABLE_LABEL_CURRT3 = "corriente_rack3";
const char *VARIABLE_LABEL_POWER1 = "potencia_rack1";
const char *VARIABLE_LABEL_POWER2 = "potencia_rack2";
const char *VARIABLE_LABEL_POWER3 = "potencia_rack3";
const char *VARIABLE_LABEL_ENERG1 = "energia_rack1";
const char *VARIABLE_LABEL_ENERG2 = "energia_rack2";
const char *VARIABLE_LABEL_ENERG3 = "energia_rack3";
const char *VARIABLE_LABEL_LIGHT = "light";
const char *VARIABLE_LABEL_RAIN = "rain";
```

Figura 7 Creación de variables

5. Utilizar las funciones de conexión al servidor de ubidots (posteriormente a la conexión wifi con acceso a internet), encolado y envío de datos.

```
ubidots.setCallback(callback);
ubidots.setup();
ubidots.reconnect();
```

```
if (!ubidots.connected()){
  ubidots.reconnect();
```

```
ubidots.add(VARIABLE_LABEL_HUMD1, humidity_measured[0]);
ubidots.add(VARIABLE_LABEL_HUMD2, humidity_measured[1]);
ubidots.add(VARIABLE_LABEL_TEMP1, temperature_measured[0]);
ubidots.add(VARIABLE_LABEL_TEMP2, temperature_measured[1]);
ubidots.publish(DEVICE_LABEL);
ubidots.add(VARIABLE_LABEL_CURRT1, currentRMS_measured[0]);
ubidots.add(VARIABLE_LABEL_POWER1, power_measured[0]);
ubidots.add(VARIABLE_LABEL_ENERG1, energy_measured[0]);
ubidots.publish(DEVICE_LABEL);
ubidots.add(VARIABLE_LABEL_CURRT2, currentRMS_measured[1]);
ubidots.add(VARIABLE_LABEL_POWER2, power_measured[1]);
ubidots.add(VARIABLE_LABEL_ENERG2, energy_measured[1]);
ubidots.publish(DEVICE_LABEL);
ubidots.add(VARIABLE_LABEL_CURRT3, currentRMS_measured[2]);
ubidots.add(VARIABLE_LABEL_POWER3, power_measured[2]);
ubidots.add(VARIABLE_LABEL_ENERG3, energy_measured[2]);
ubidots.publish(DEVICE_LABEL);
ubidots.add(VARIABLE_LABEL_LIGHT, light_measured);
ubidots.add(VARIABLE_LABEL_RAIN, rain_measured);
ubidots.publish(DEVICE_LABEL);
```

Figura 6 Configuración de variables para envío de datos

3.4.2 Variables a monitorear:

Las variables a monitorear en el sistema son **temperatura, humedad** (tanto para la area1 como para la área2), **corriente**, **potencia**, **energía** (rack1, rack2 y rack3), iluminación (En porcentaje) y Nivel de agua (en porcentaje según la máxima altura del sensor).
Las variables configuradas desde el código al momento del primer envío se añadirán a los datos del dispositivo, no es necesariamente crear variable por variable dentro de ubidots.

3.4.3 Ejemplo de configuración de Variable:

Primero requerimos de la fuente de información para esta variable, serían los sensores DHT22 estos miden tanto la temperatura como la humedad, se utiliza la librería de Arduino para hacer la comunicación con el sensor dado que el sensor es digital, realizamos la obtención de la información mediante el protocolo OneWire.

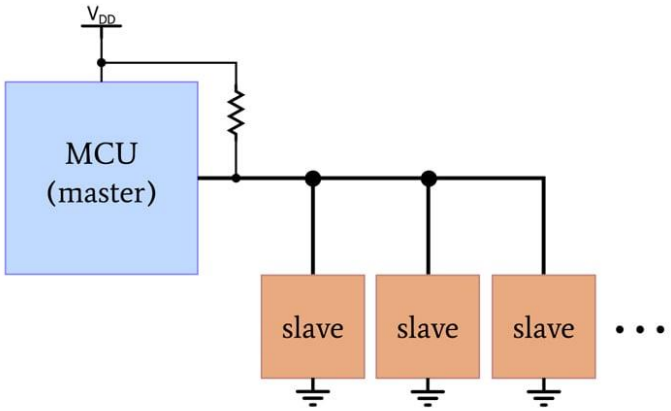


Figura 9 Envío de datos hacia los sensores DTH22

En nuestra tarjeta utilizamos 2 buses independientes con la posibilidad de añadir más sensores.

Figura 10 Conexión de sensor digital de temperatura y humedad relativa

En la programación primero definimos los puertos a los cuales están conectados los dos buses OneWire y definimos el tipo de sensor en este caso el DHT22.

```
12  // **** Sensores de Temperatura y Humedad *****
13  #define DHT1_PIN 13
14  #define DHT2_PIN 14
15  #define DHTTYPE DHT22
```

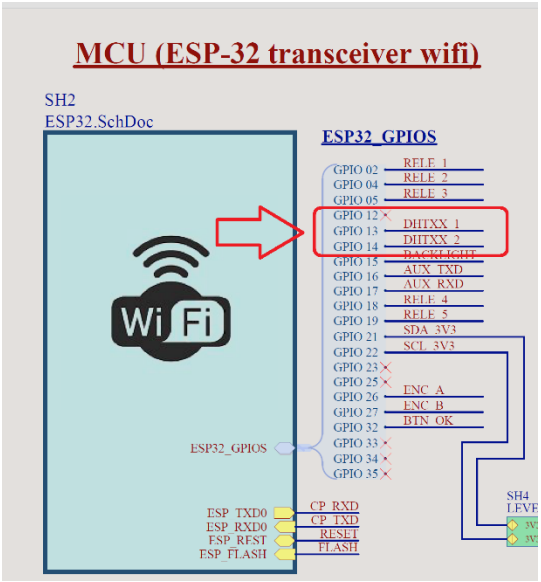


Figura 11 Definición de los puertos donde va el sensor

Creamos las variables en donde vamos a almacenar las lecturas de los sensores junto con las instancias de los objetos para cada sensor para posteriormente utilizar las funciones de

lectura de temperatura y humedad.

```
88    float temperature_measured[2];
89    float humidity_measured[2];
```

```
119   DHT_Unified dht1(DHT1_PIN, DHTTYPE);
120   DHT_Unified dht2(DHT2_PIN, DHTTYPE);
```

Figura 12 Creación de las variables de temperatura y humedad

En el Setup inicializamos los sensores y definimos las métricas a medir, para esto creamos dos instancias más de sensor.

```
298   dht1.begin();
299   dht2.begin();
300   sensor_t sensorDht1;
301   dht1.temperature().getSensor(&sensorDht1);
302   dht1.humidity().getSensor(&sensorDht1);
303   sensor_t sensorDht2;
304   dht2.temperature().getSensor(&sensorDht2);
305   dht2.humidity().getSensor(&sensorDht2);
```

Figura 13 Definición de las métricas a medir

Una vez inicializado los sensores en nuestro bucle principal Loop dentro de la función read_sensors(), procedemos con la lectura y guardamos en las respectivas variables creadas tanto para la temperatura como para la humedad (ambos sensores).

```
369    read_sensors();
```

```
228   sensors_event_t eventDht1;
229   sensors_event_t eventDht2;
230 -
231   dht1.temperature().getEvent(&eventDht1);
232   if (!isnan(eventDht1.temperature)){
233     temperature_measured[0] = eventDht1.temperature;
234   }
235   dht1.humidity().getEvent(&eventDht1);
236   if (!isnan(eventDht1.temperature)){
237     humidity_measured[0] = eventDht1.relative_humidity;
238   }
239   dht2.temperature().getEvent(&eventDht2);
240   if (!isnan(eventDht2.temperature)){
241     temperature_measured[1] = eventDht2.temperature;
242   }
243   dht2.humidity().getEvent(&eventDht2);
244   if (!isnan(eventDht2.relative_humidity)){
245     humidity_measured[1] = eventDht2.relative_humidity;
246   }
247
```

Figura 14 Se guarda información en las variables

Ya para este punto la data estará guardada en las variables (en este caso un vector) **temperatura_measured[0]** y **temperatura_measured[1]** estando lista para asignar los datos a las variables de Ubidots y proceder con el envío de la información.

En los demás sensores es el mismo procedimiento teniendo en cuenta utilizar la librería respectiva para el sensor correspondiente o lectura analógica, muestreo, y relación matemática que esté de por medio antes de la obtención de la métrica que requerimos, esto dependerá del sensor que estemos utilizando.

3.5 Proyectos de investigación vinculados

Sistema De Monitoreo de Variables Medioambientales Usando Una Red de Sensores Inalámbricos y Plataformas De Internet De Las Cosas

Autores: Manuel Quiñones, Víctor González-Jaramillo, Rommel Torres, Miguel Jumbo

Universidad Técnica Particular de Loja

Diseño de una solución basada en fuentes de Energía Solar y el Internet de las Cosas (IoT) para el control del consumo de energía eléctrica de los servicios hoteleros
Autores: Valdiviezo Calero, Vladimir Alejandro
Pontificia Universidad Católica del Ecuador

Desarrollo de un dispositivo de monitoreo del estado operativo para equipos electrónicos con comunicación a plataforma IOT y manejo de alertas para prevenir posibles daños.
Autores: Campos Alvarado Coraima Carolina, Bayas Salazar Karen Isabel
Universidad de Guayaquil

3.6 Descripción de la propuesta

A continuación, se muestra la descripción resumida de la propuesta planteada en este trabajo de investigación.

En la figura 6 se observa un esquema resumido del prototipo de monitorización de datacenters.

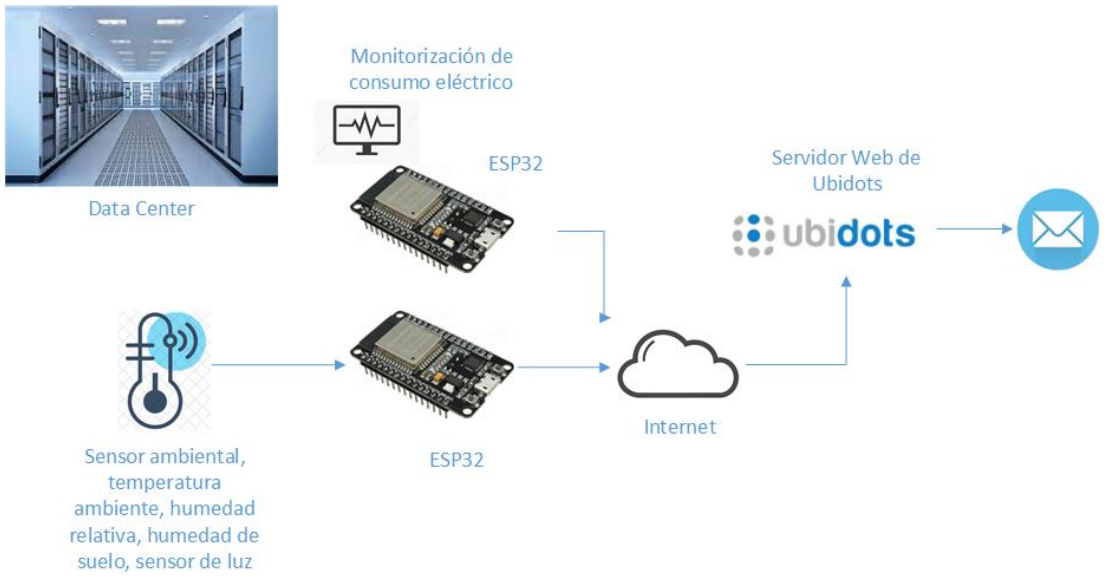


Figura 15 Prototipo Iot de monitorización de datacenter

3.7 Diseño electrónico del prototipo

A continuación, se muestran los diseños electrónicos del prototipo IoT:

En la figura 16 se muestra el pinout de conexiones del ESP32

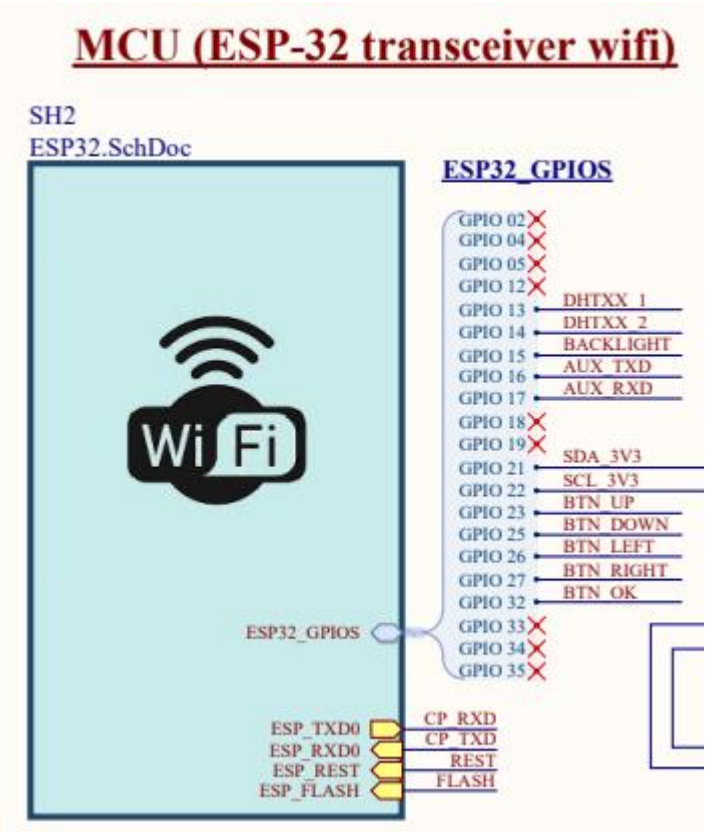


Figura 16 Conexiones ESP32

En la figura 17 se muestra el esquemático de del convertidor USB a UART.

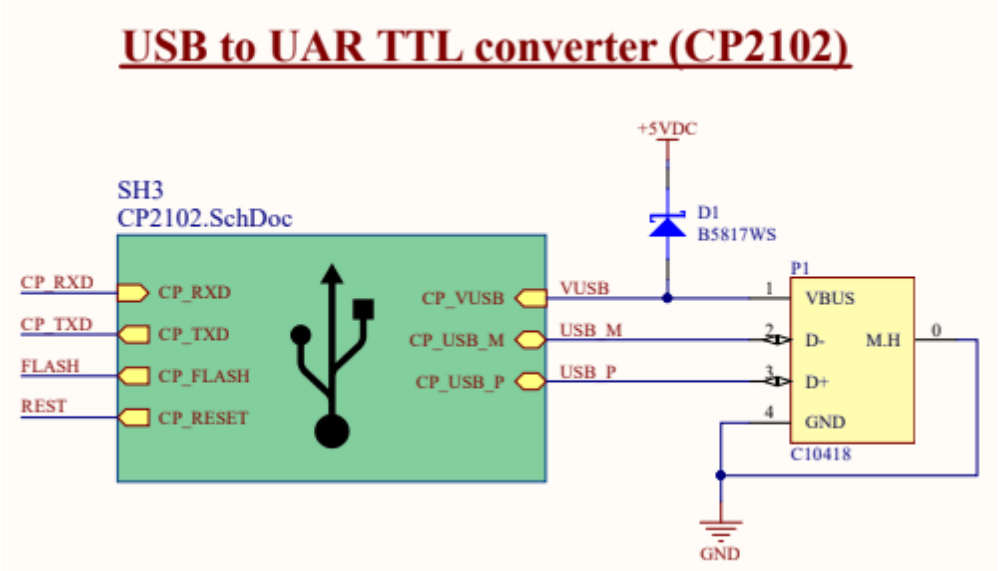


Figura 17 Conexión USB a UAR (CP2102)

En la figura 18 se observa el esquemático de la conexión HMI, display LCD de 20 x 4.

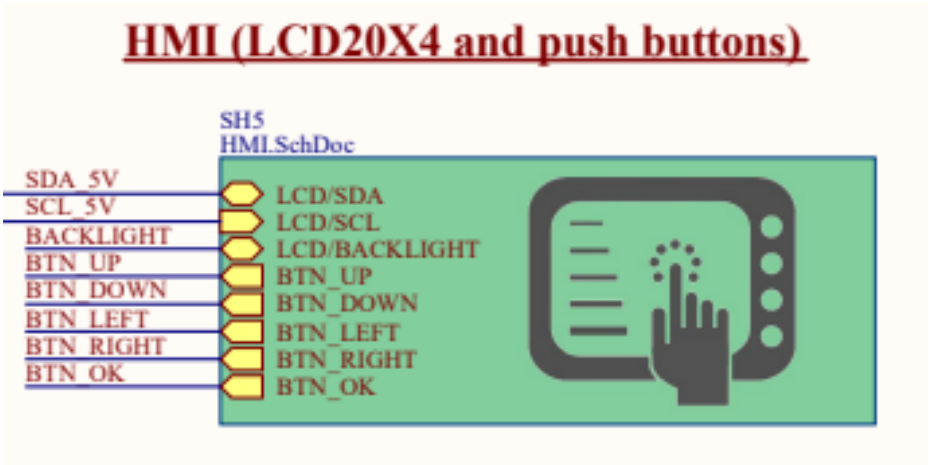


Figura 18 Conexión HMI

En la figura 19 se observa el esquemático de conexiones del convertidor analógico digital.

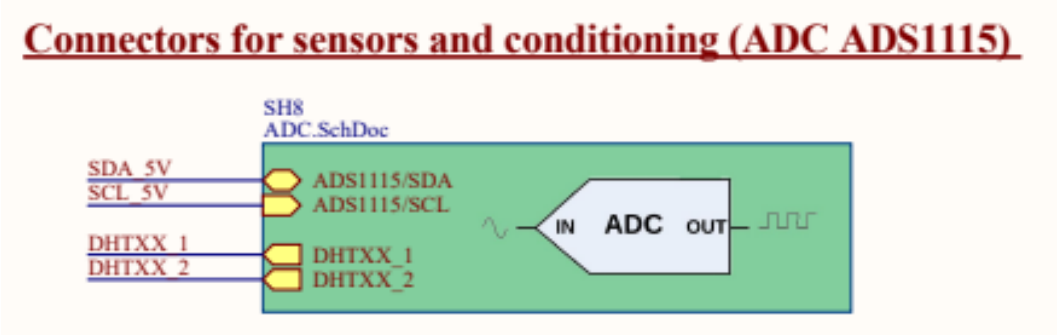


Figura 19 Conexión ADC

En la figura 20 se observa el esquemático del PSU.

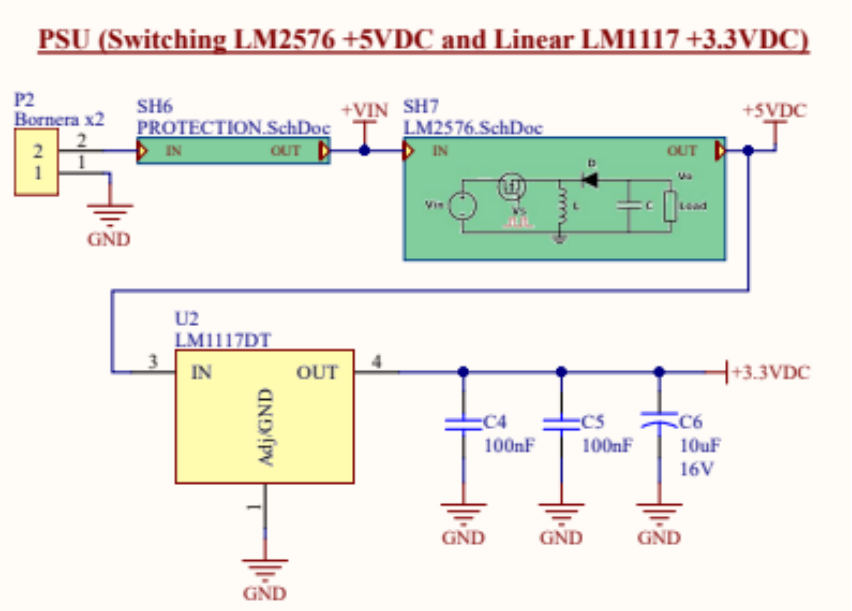


Figura 20 Conexión PSU

Las conexiones GPIO se muestran en la figura 21

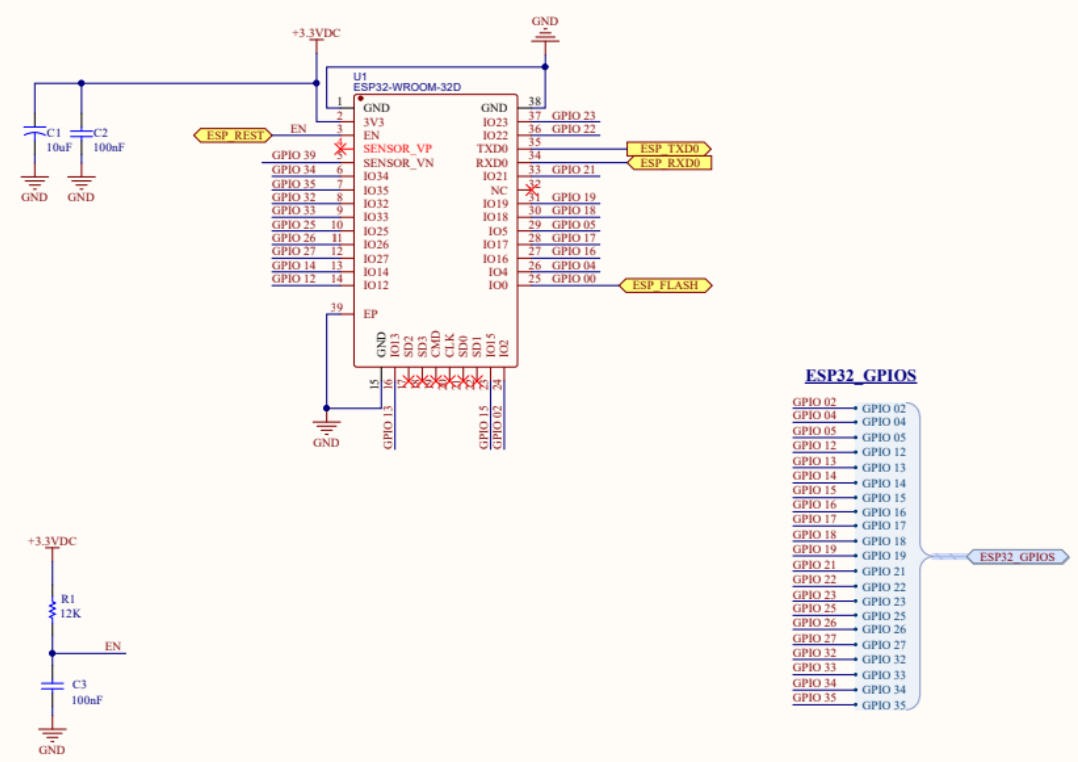


Figura 21 Conexiones GPIO del ESP32

En la figura 22 se observa el esquemático del módulo CP2102

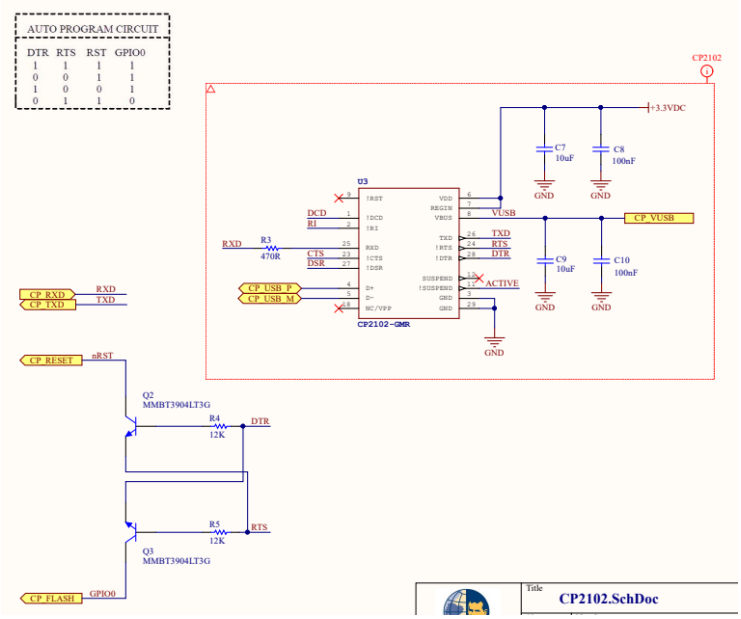


Figura 22 CP2102

En la figura 23 y 24 se observa el diagrama de protección del pcb y el level shifter, respectivamente.

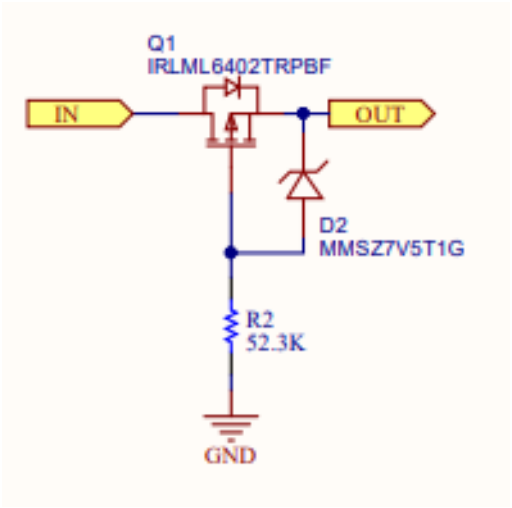


Figura 23 Diagrama para protección

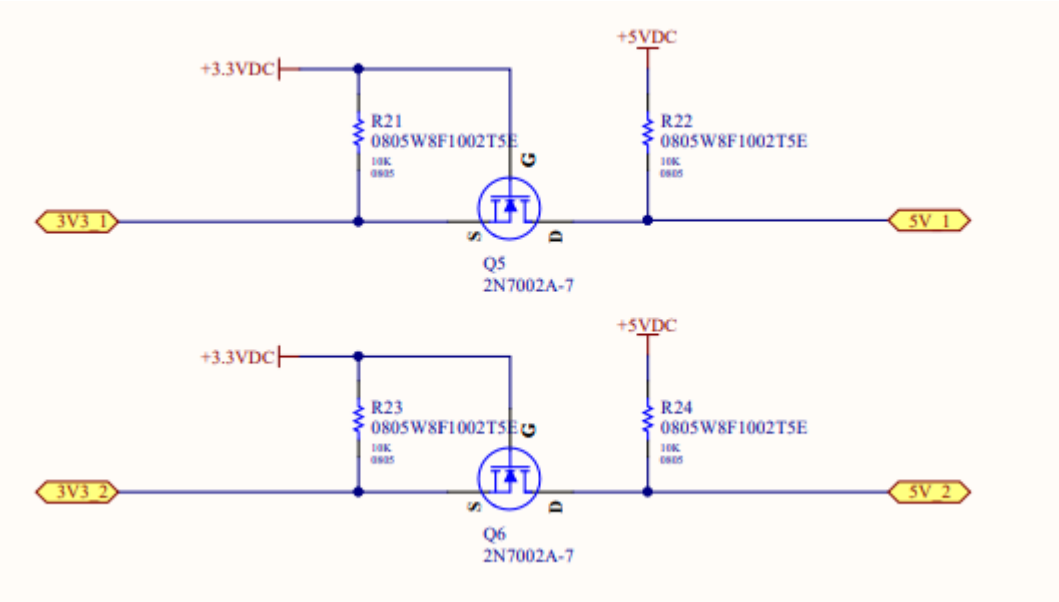


Figura 24 Level Shifter

En la figura 25 se observa el esquemático del HMI

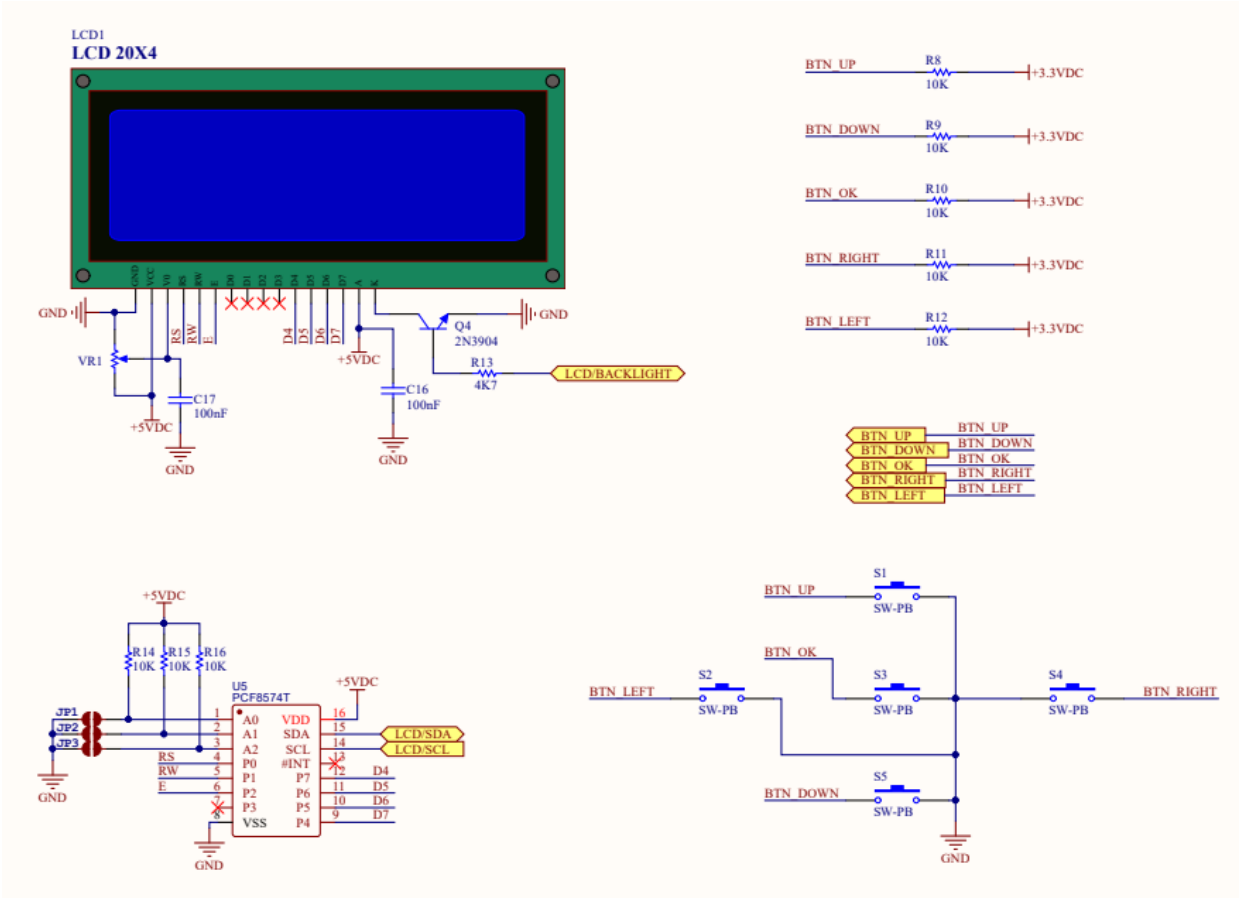


Figura 25 HMI

En la figura 26 se observa el esquemático del módulo LM2576

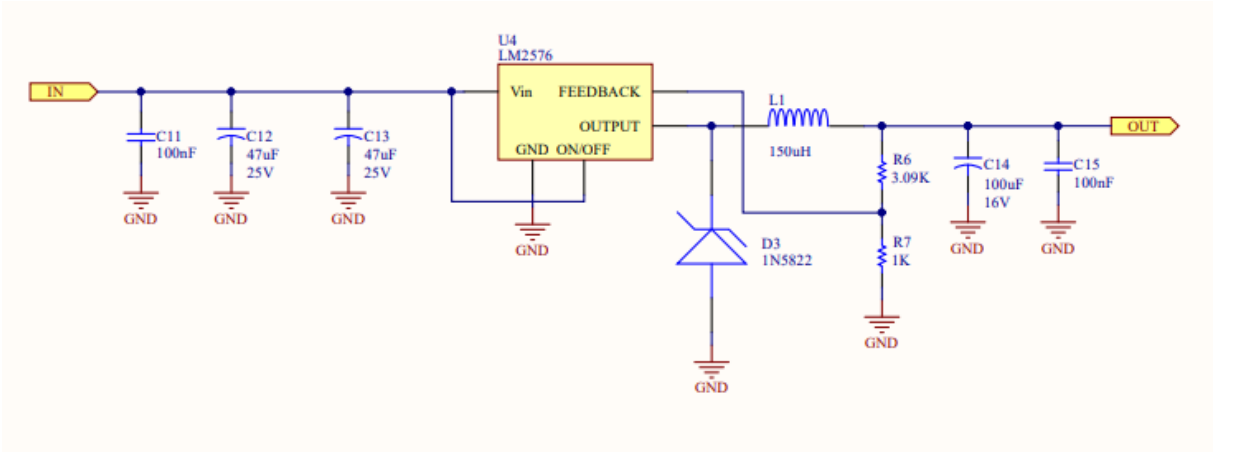


Figura 26 LM2576

3.8 Diseño en 3D de prototipo Iot de monitorización de datacenters

En la figura 27 se observa el diseño en 3D de la pcb y el display lcd.

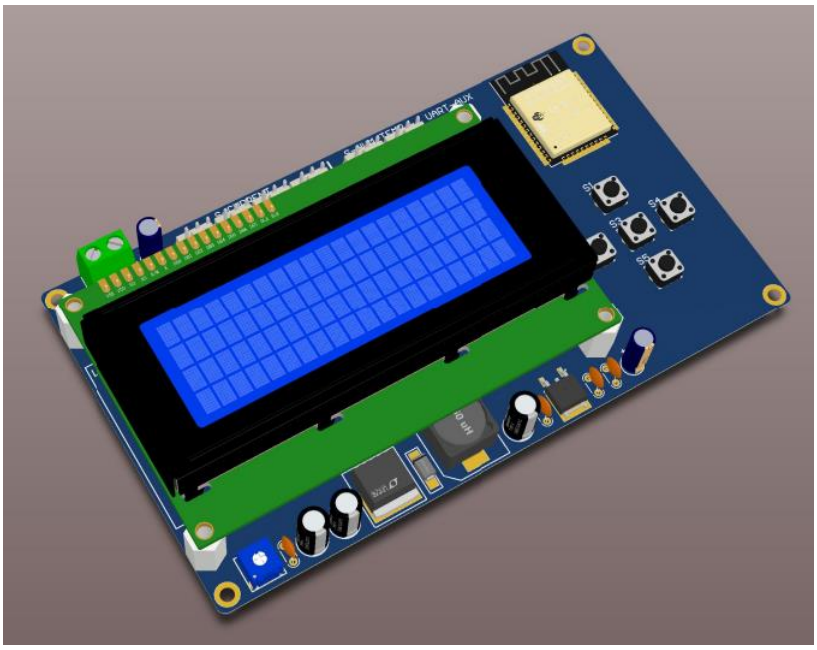


Figura 27 Diseño 3D de la pcb y la pantalla lcd

En la figura 28 se observa el diseño de la pcb con el case utilizando en Autodesk 360

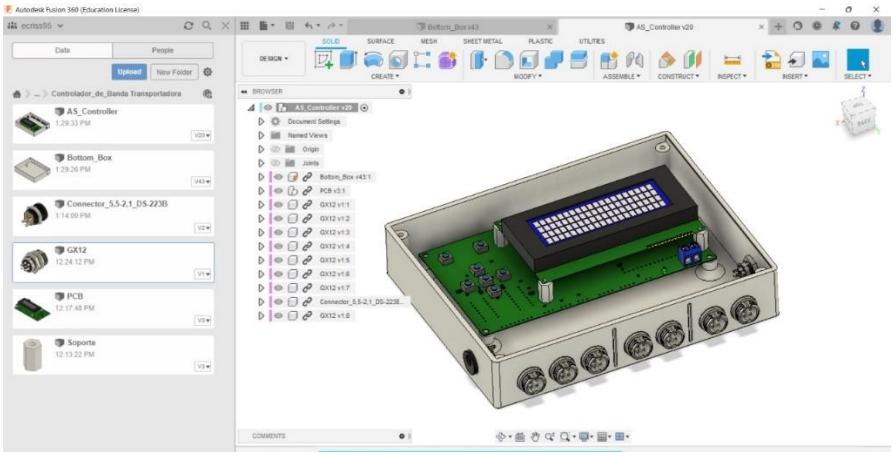


Figura 28 diseño 3D en AutoDesk 360

En las figuras 29 y 30 se observa los diseños en 3D del case del prototipo y la maqueta con los sensores, respectivamente.

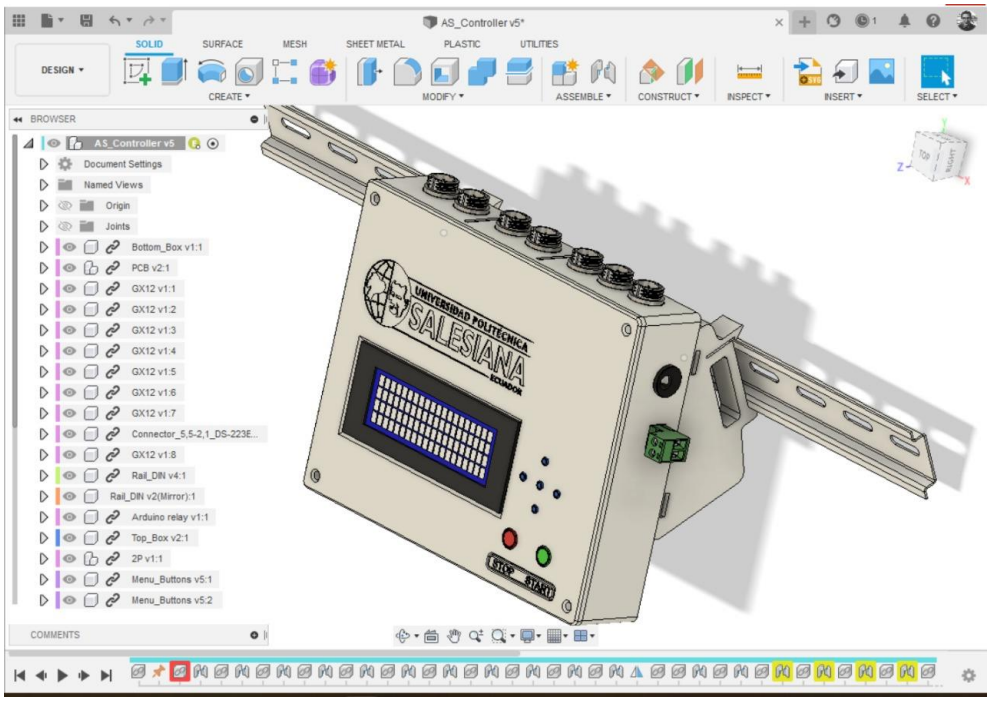


Figura 29 Diseño en 3D del case del prototipo

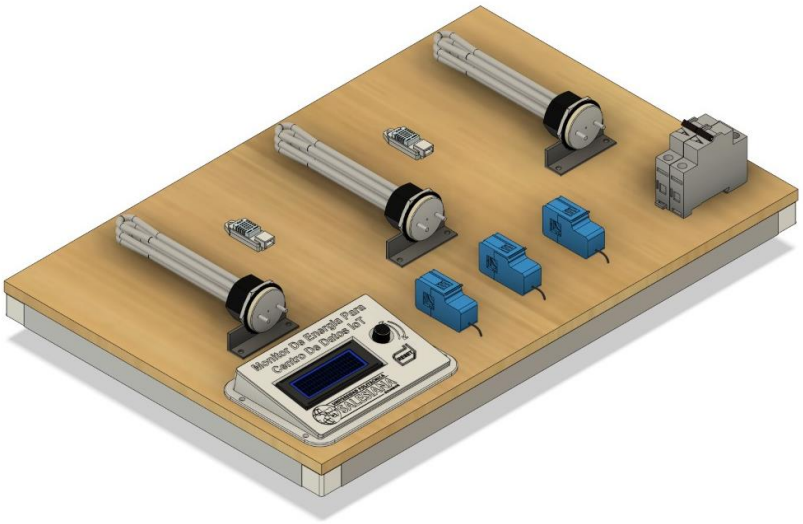


Figura 30 Diseño en 3D de la maqueta con los sensores

3.9 Elementos de prototipo de monitorización

En las figuras 31 al 35 se observan los elementos principales de la maqueta de monitorización IoT.

Figura 31 Resistencias para simular temperatura en datacenter

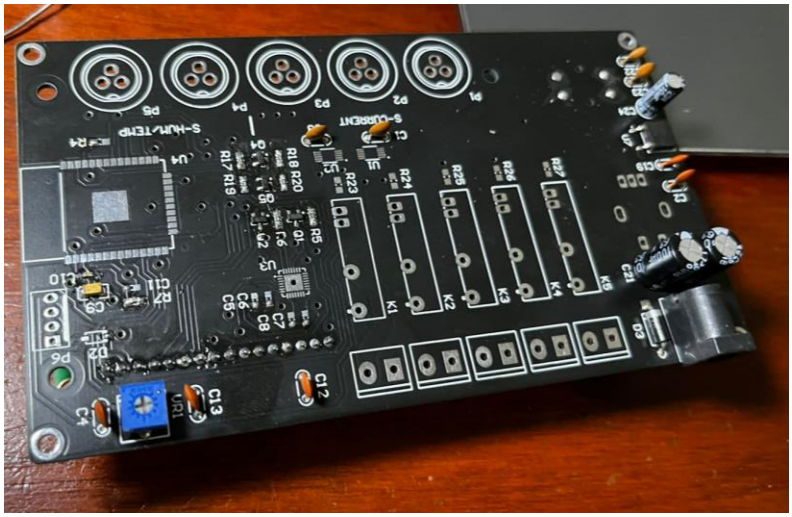


Figura 32 Placa PCB

Figura 33 Resistencia térmica

Figura 34 Pantalla LCD

Figura 35 Sensores de consumo eléctrico y breackers

3.10 Configuración de alertas vía email

En esta sección se detallan las configuraciones realizadas para las alertas vía email de las alarmas en la plataforma Ubidots. En la tabla 1 se detalla los umbrales máximo y mínimo de los sensores para las configuraciones de las alarmas vía email.

Tabla 1 Umbrales para alarmas

Descripciones	Niveles	Cantidad	Unidad
Corriente Rack 1	Mínimo	12	Amperios
	Máximo	13,8	Amperios
Corriente Rack 2	Mínimo	12	Amperios
	Máximo	13,8	Amperios
Corriente Rack 3	Mínimo	12	Amperios
	Máximo	13,8	Amperios
Potencia Rack 1	Mínimo	1355	Watts
	Máximo	1365	Watts
Potencia Rack 2	Mínimo	1355	Watts
	Máximo	1365	Watts
Potencia Rack 3	Mínimo	1355	Watts
	Máximo	1365	Watts
Humedad Zona 1	Mínimo	30	%
	Máximo	65	%
Humedad Zona 2	Mínimo	30	%
	Máximo	65	%
Temperatura Zona 1	Mínimo	30	° C
	Máximo	55	° C
Temperatura Zona 2	Mínimo	30	° C
	Máximo	55	° C
Light	Mínimo	20	
	Máximo	40	
Nivel de Agua	Mínimo	2	%
	Máximo	15	%

En la figura 36 se observa la configuración de los 12 eventos de alertas vía email cuando los umbrales superen o disminuyan en valor dependiendo de cada variable.

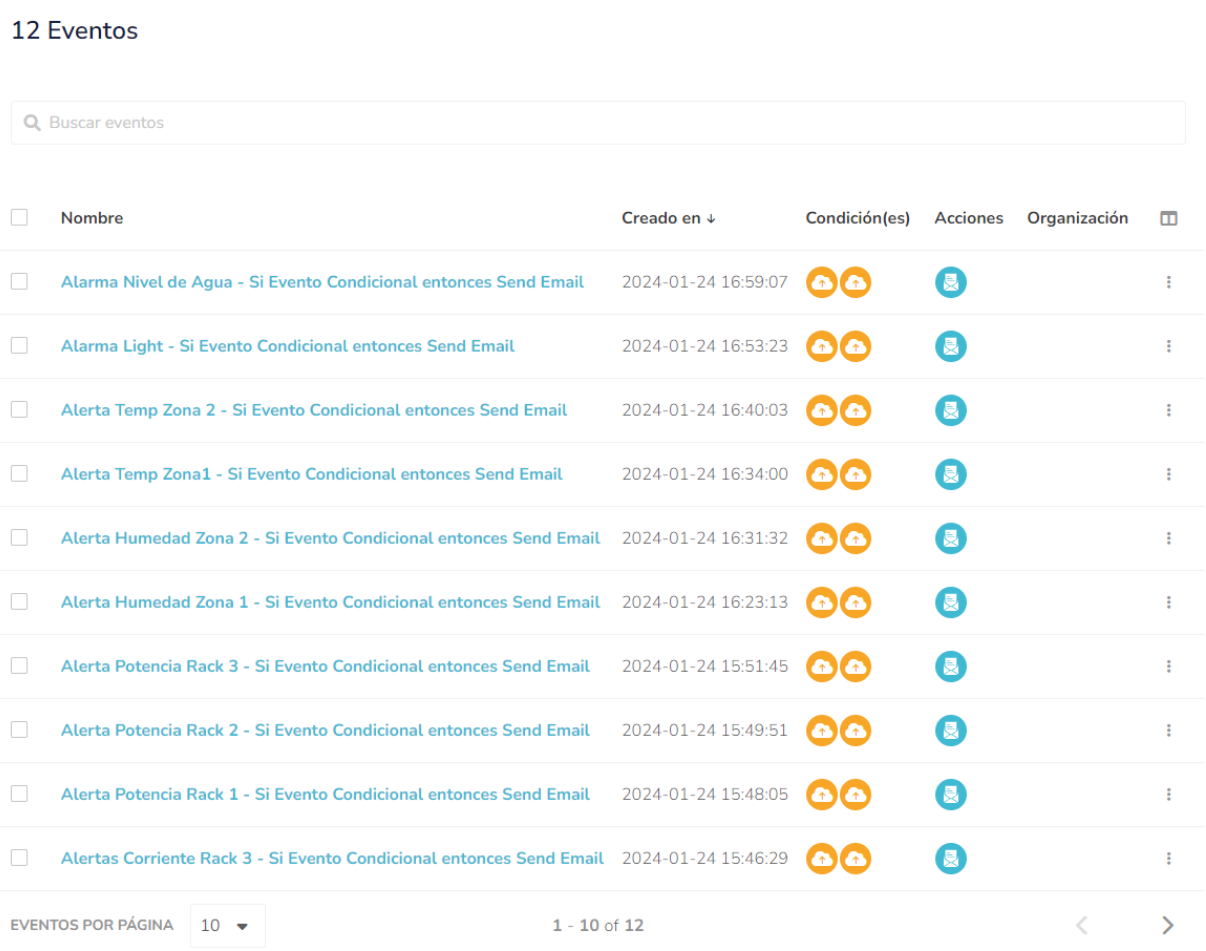


Figura 36 Configuración de eventos de alarmas vía email

En la figura 37 se observa la configuración de las condiciones que tienen que pasar para que la alarma se active.

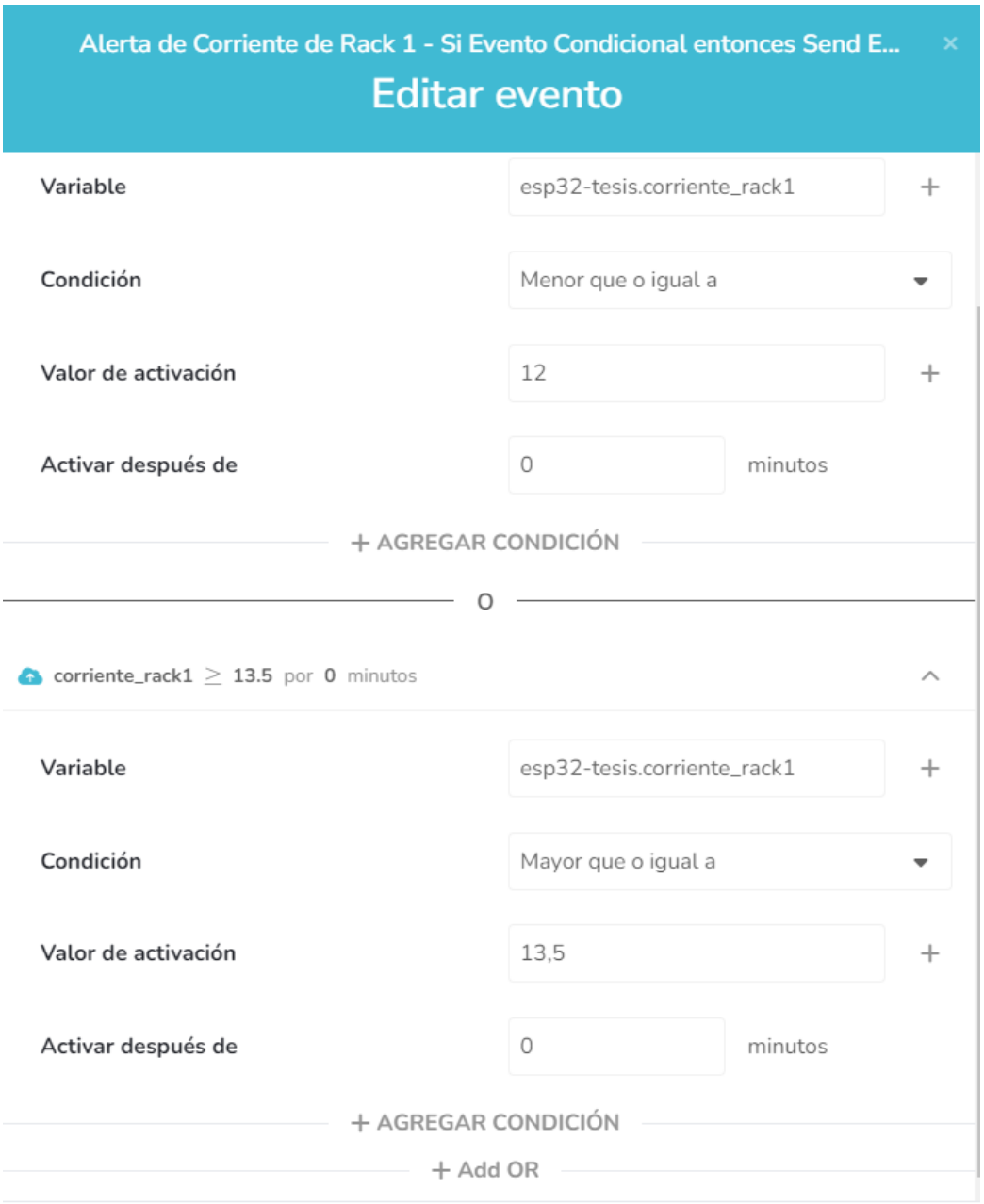


Figura 37 Configuraciones de las condiciones para envío de alertas vía email.

En la figura 38 se observa la configuración del texto del correo electrónico que se envia cuando se activa la alarma para ser enviada por email.

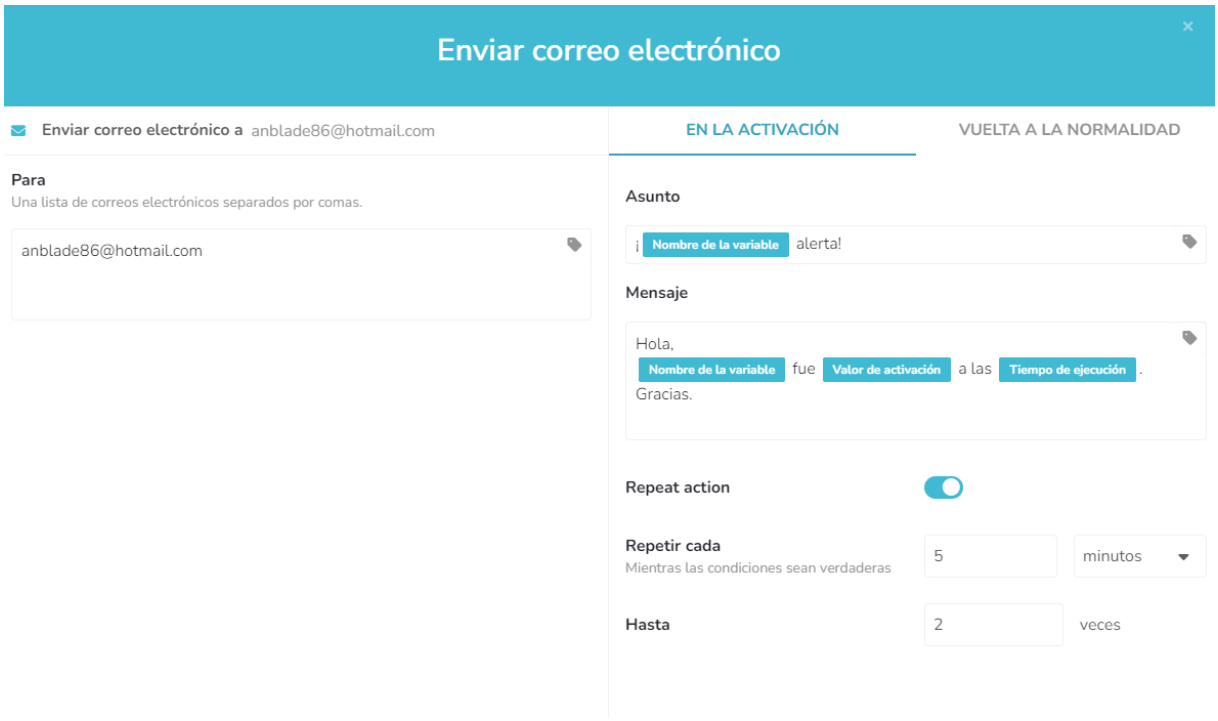


Figura 38 Configuración de mensaje de email.

4 Resultados

Detallamos los resultados obtenidos y las pruebas realizadas con el diseño e implementación del prototipo IoT de monitorización de datacenters.

En la figura 39 se observa la pantalla principal o Dashboard de las configuraciones en Ubidots.

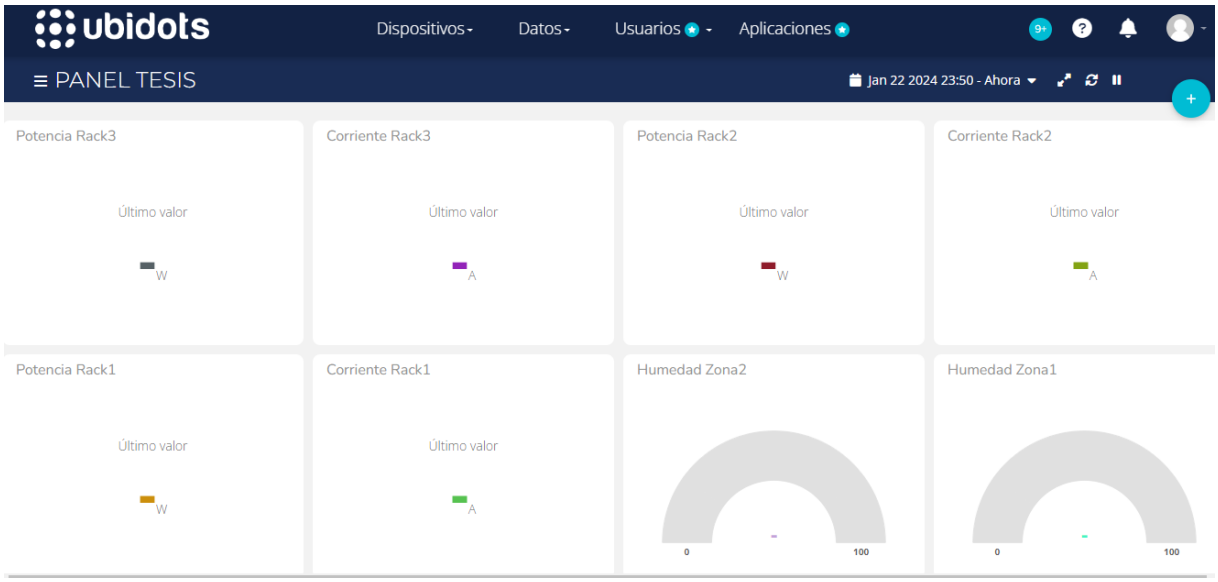


Figura 39 Pantalla principal de Ubidots

En la figura 40 se observa las variables obtenidas por el objeto denominado ESP32. Estas variables son obtenidas automáticamente al configurar el objeto ESP32, lo que significa que se ha validado con el token la comunicación MQTT de los sensores con el ESP32.

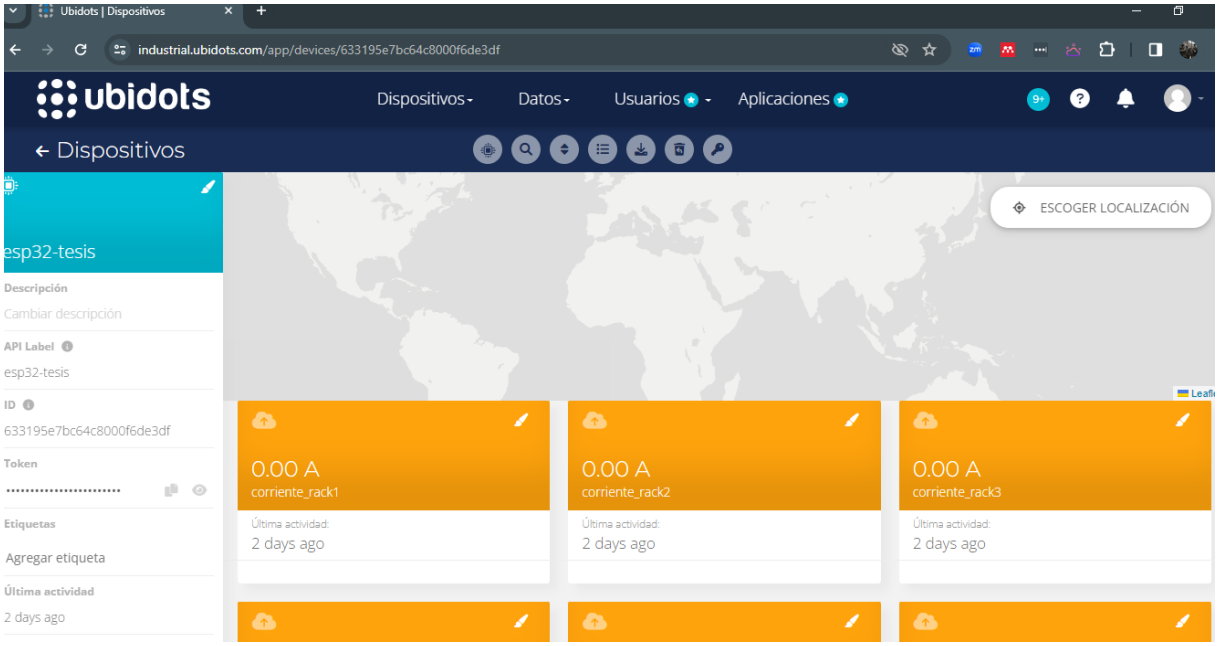


Figura 40 Variables en ubidots del proyecto IoT

En la figura 41 se observa el dispositivo esp32, el cual es el equipo principal de este proyecto de investigación. Se coloca el nombre de esp32 para tener referencia que se está trabajando con este elemento.

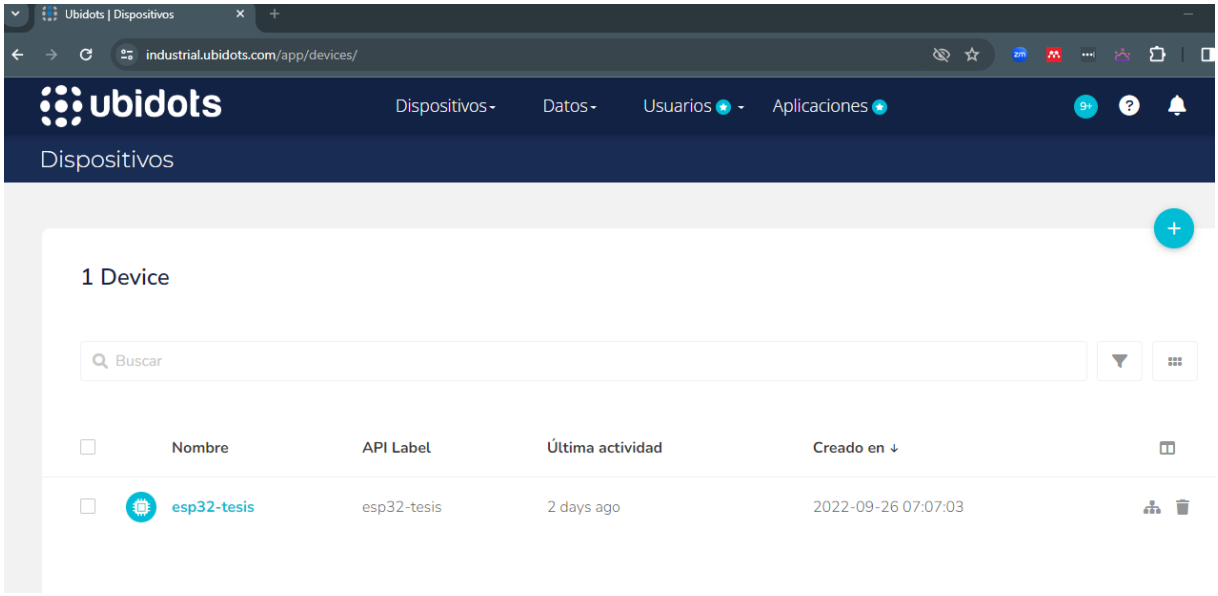


Figura 41 Dispositivo ESP32

En la figura 42 se observan todas las variables del dispositivo esp32. Las variables son de corriente, energía, humedad, luminosidad, etc.

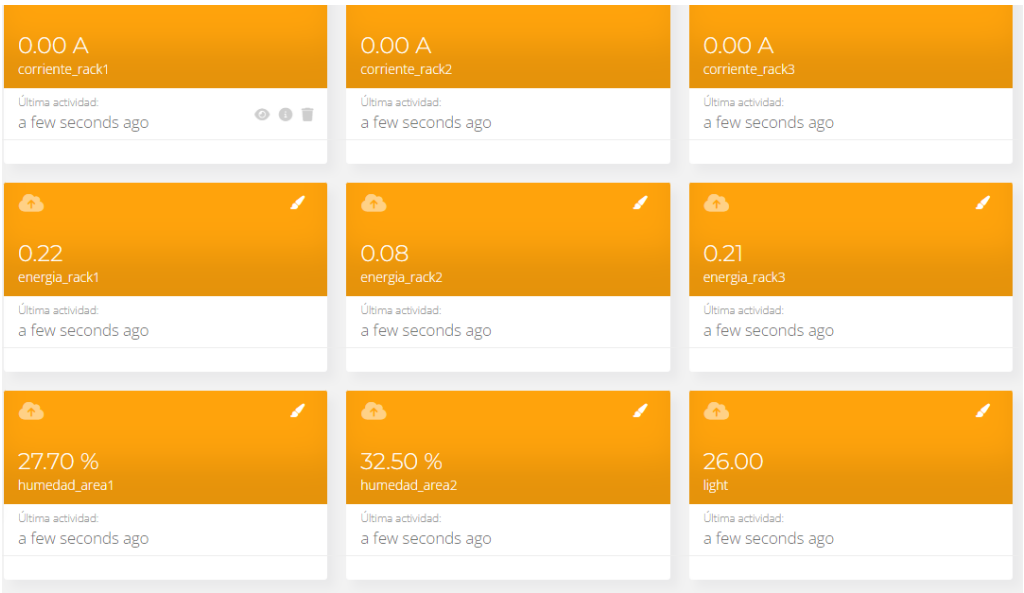


Figura 42 Variables del ESP32

En la figura 43 se observa los datos obtenidos durante las primeras pruebas con el prototipo IoT de monitorización.

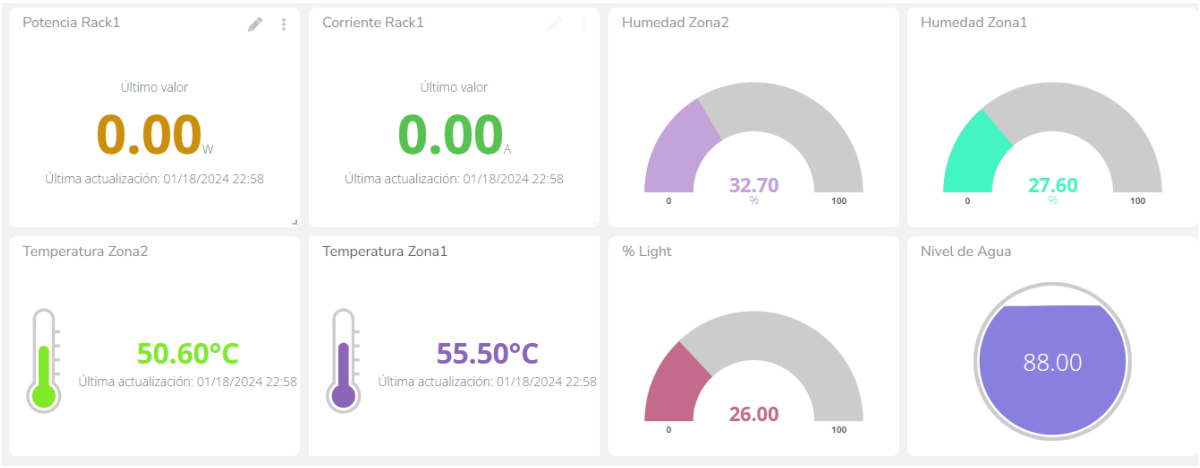


Figura 43 Datos medidos durante las pruebas

En la figura 44 se observa el Dashboard del sensor de nivel de agua.

Figura 44 Dashboard del nivel de agua

En la figura 45 se observa el sensor de cantidad de luz.

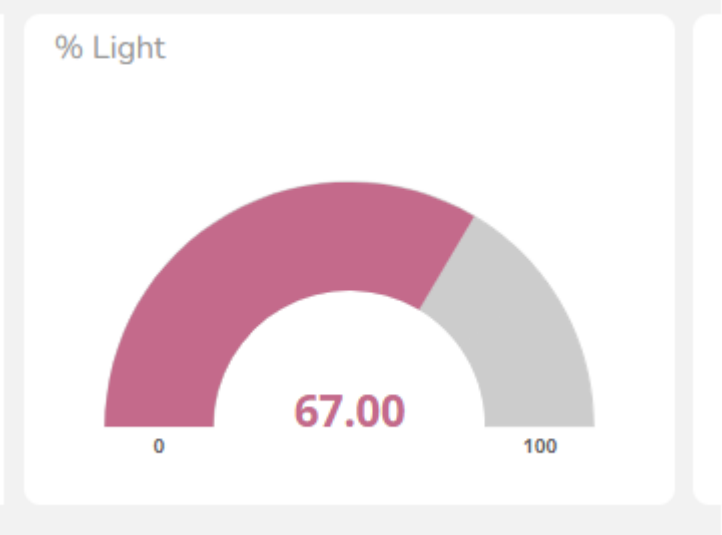


Figura 45 Dashboard de cantidad de luz

En las figuras del 46 al 49 se detallan los datos medidos de los sensores. En los Dashboard se observa los cambios durante las pruebas.

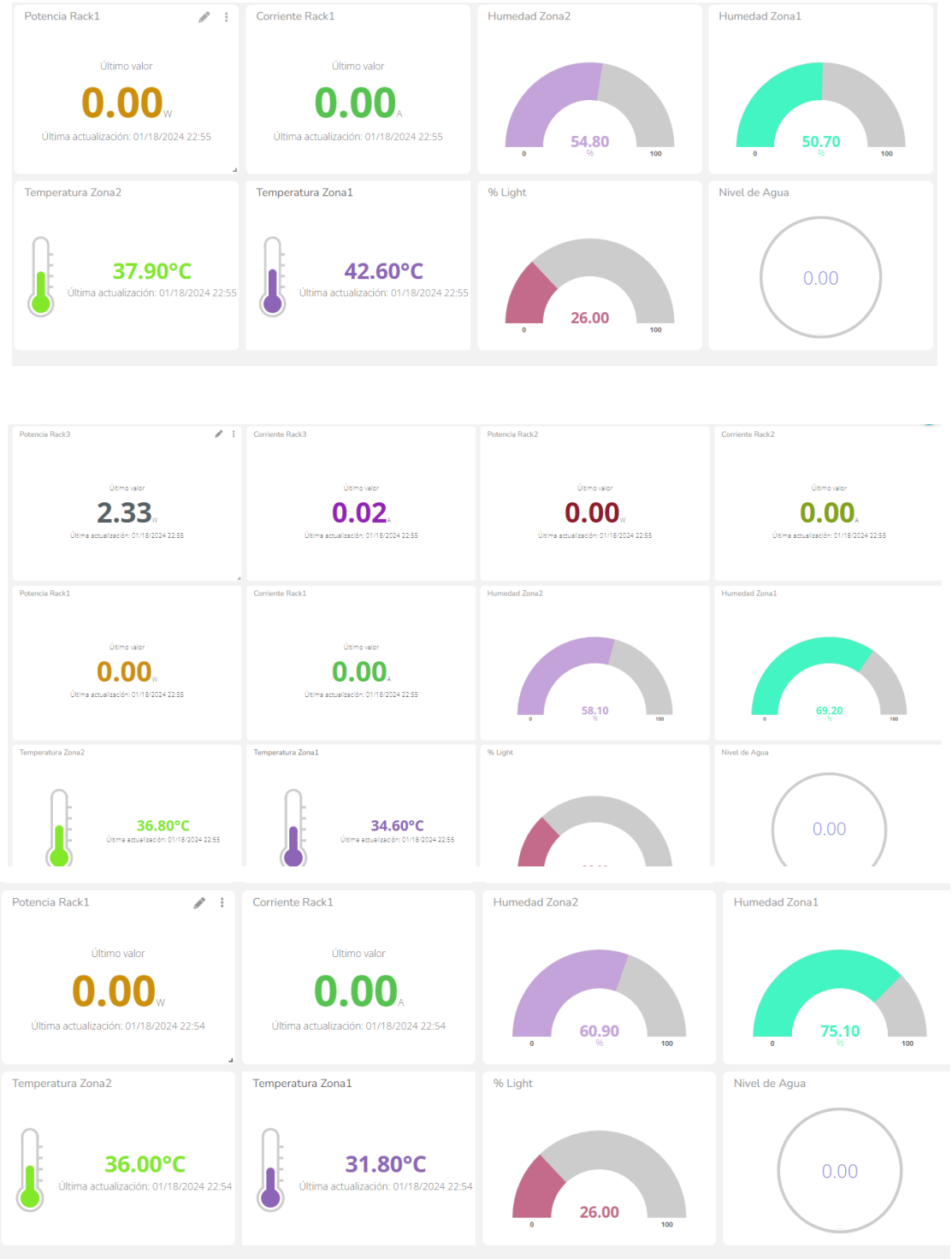


Figura 46 Dashboard con pruebas de sensores

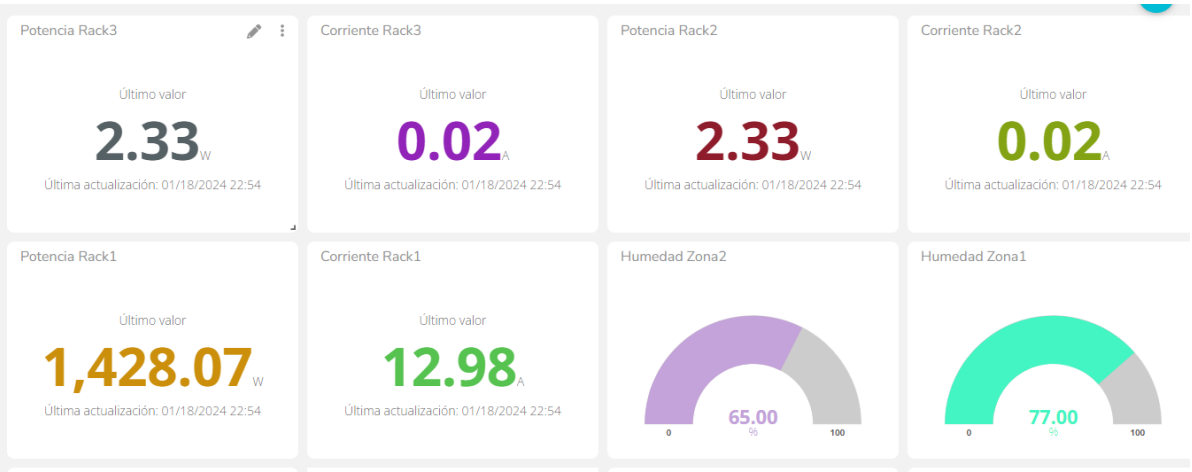


Figura 47 Dashboard de humedad de suelo, corriente y potencia por racks.

Figura 48 Dashboard de temperatura por zonas

Figura 49 Dashboard de consumo eléctrico en potencia.

4.1 Pruebas de maqueta del prototipo

En las figuras desde el numero 50 al 65 se observan las pruebas realizadas con la maqueta IoT y los sensores.

Figura 50 Pruebas con sensores de temperatura y humedad

Figura 51 Pruebas con sensores de consumo energético

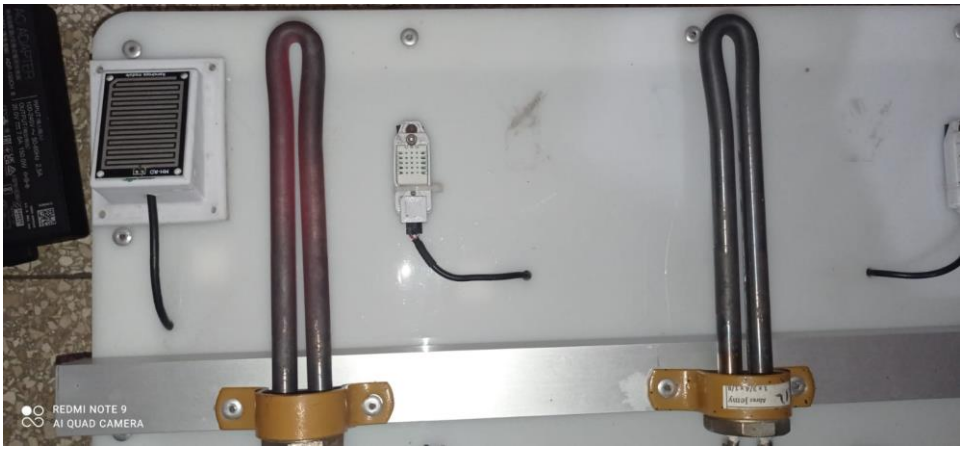


Figura 52 Pruebas con resistencia térmica 1

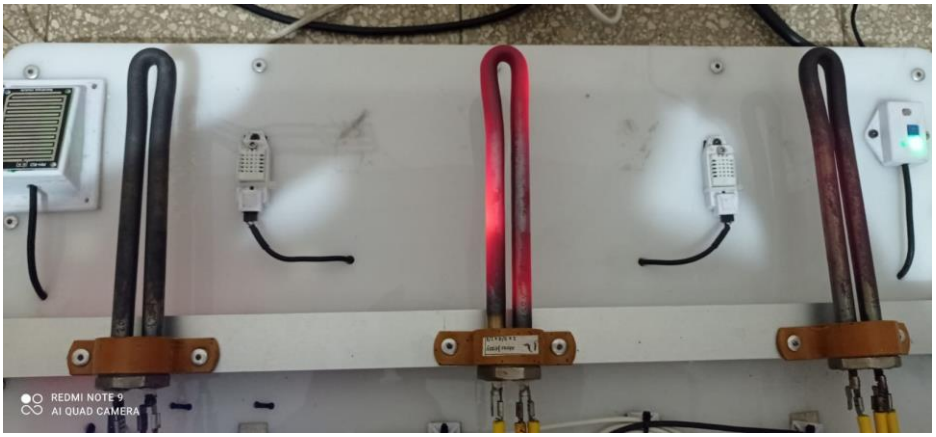


Figura 53 Pruebas con resistencia térmica 2

Figura 54 Pruebas con resistencia térmica 3

Figura 55 Router para conectividad inalámbrica

Figura 56 Maqueta conectada a red inalámbrica

Figura 57 Maqueta con conexiones

Figura 58 Conexiones eléctricas de maqueta

Figura 59 Pruebas con maqueta

Figura 60 Monitorización de Ubidots

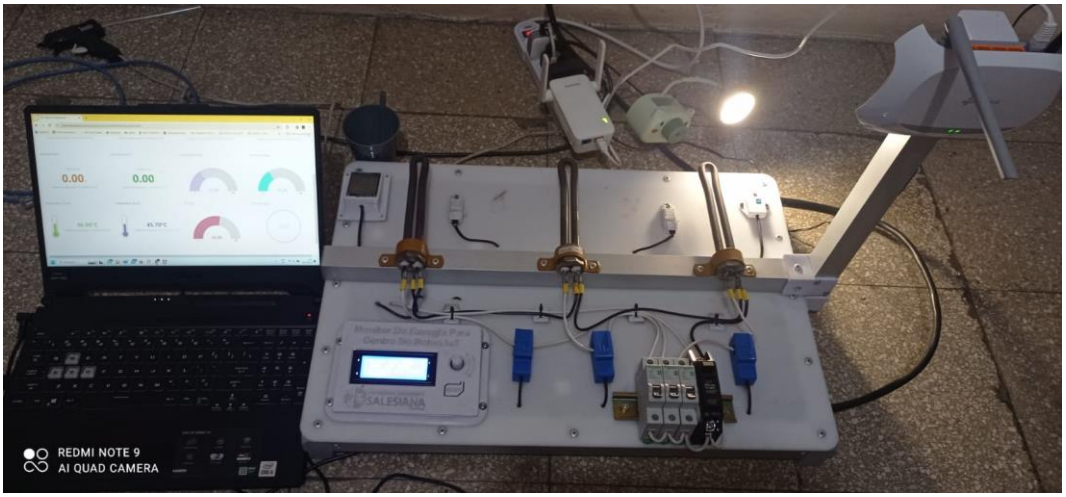


Figura 61 Pruebas de sensor de luminosidad

Figura 62 Sensor de luz

Figura 63 Visualización de censado de luz

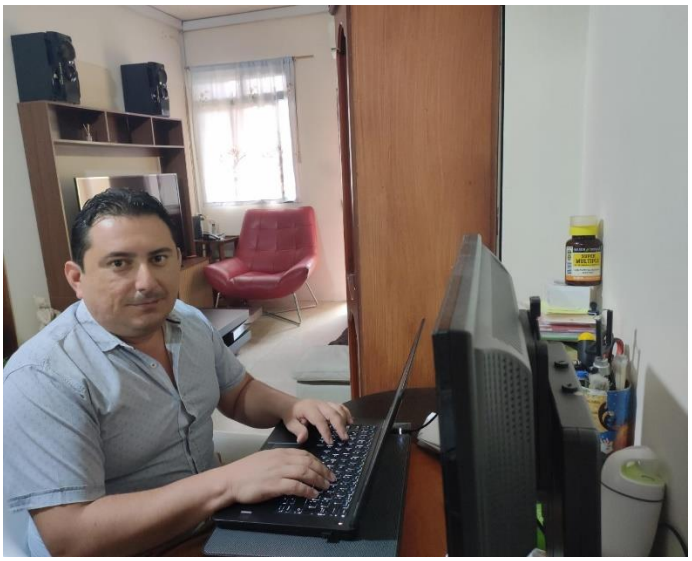


Figura 64 Programación de ESP32

Figura 65 Revisión de código de programación

4.2 Resultados de pruebas de alarmas con envíos mails.

En esta sección se observan las evidencias de los correos que llegan luego de que los parámetros medidos pasan el umbral configurado en Ubidots. En las figuras del 66 al 69 se observan las alarmas que llegan al correo.

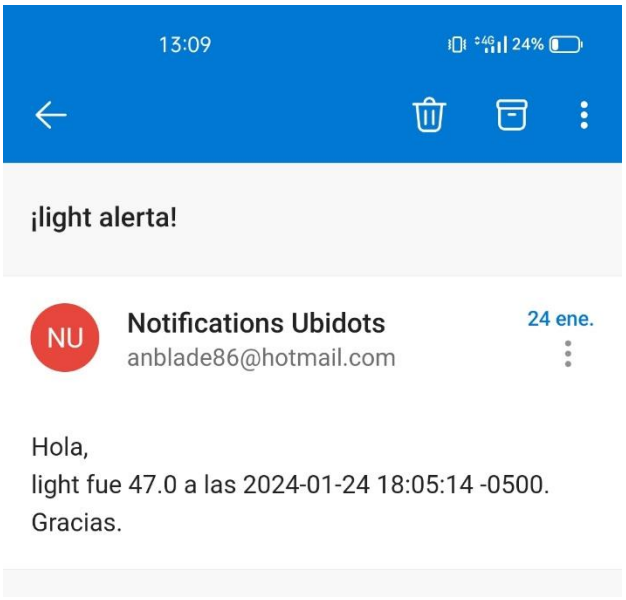


Figura 66 Alerta de luz

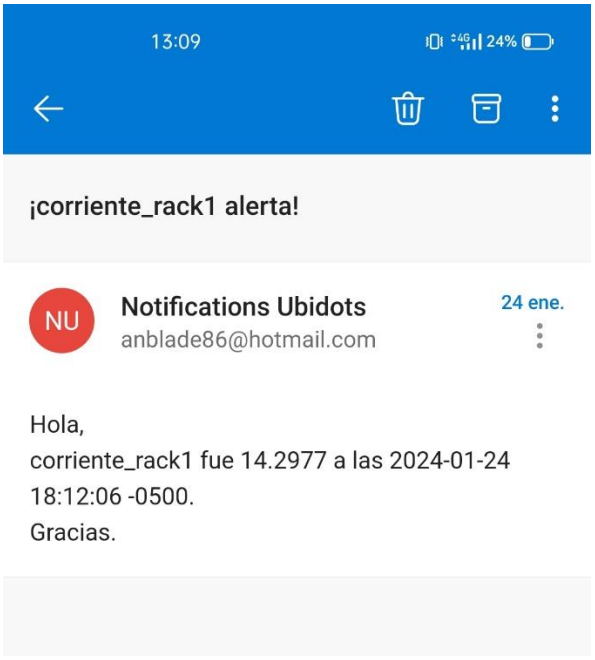


Figura 67 Alerta de corriente rack1

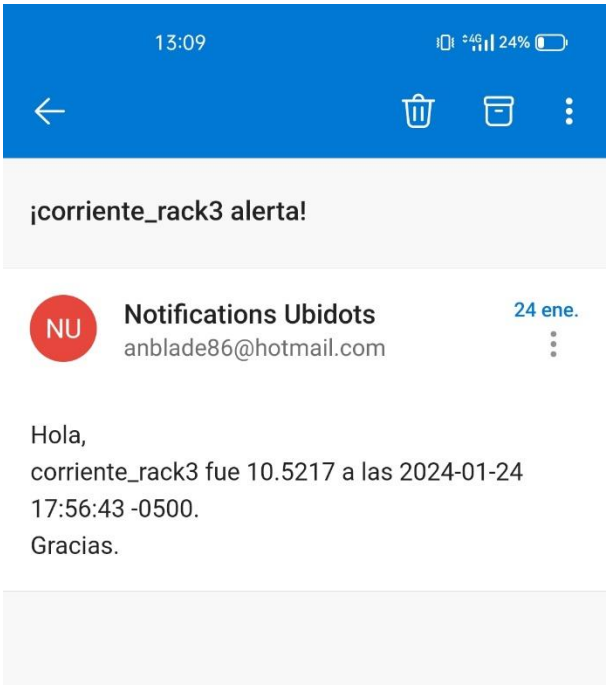


Figura 68 Alerta corriente rack3

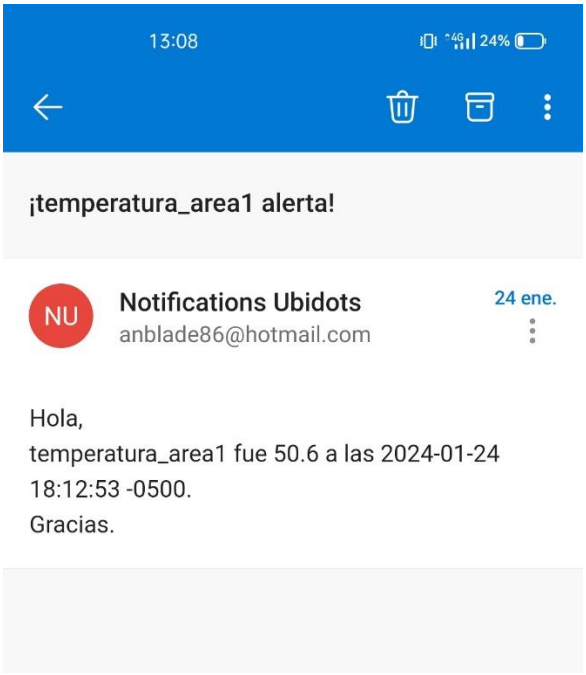


Figura 69 Alerta de temperatura

5 Conclusiones

Las conclusiones de este trabajo de investigación son las siguientes:

- Se realizó el diseño e implementación de un prototipo IOT para el monitoreo de parámetros ambientales y de consumo de energía aplicados a datacenters utilizando hardware de bajo costo como el ESP32, sensores y servicios web como Ubidots.

- Se realizó el diseño de una red inalámbrica para realizar la conexión del ESP32 y los sensores para la monitorización de parámetros ambientales y de control de consumo de energía en un datacenter aplicado al laboratorio de la carrera de ingeniería electrónica de la Universidad Politécnica Salesiana sede Guayaquil. Para este diseño se utiliza router inalámbrico que interconecta al ESP32 con la red de internet.

- Se implementó sistema de sensores para monitorización ambiental como temperatura ambiente, humedad relativa, humedad de suelo, sensor de luz, y sensor para el consumo eléctrico para prototipo IoT aplicado a la monitorización de datacenters simulado en laboratorio de telecomunicaciones.

- Se implementó servicios en la nube para el almacenamiento y monitoreo web de la información obtenida por los sensores mediante Ubidots.

- Se configuró servicios de alertas vía mail cuando los parámetros ambientales o energéticos salgan de rango y afecten a la funcionalidad de algún equipo del datacenter.

6 Recomendaciones

Las recomendaciones de este trabajo de titulación son las siguientes:

- Se recomienda situar la maqueta en laboratorio para prácticas en IoT para estudiantes de la carrera de ingeniería en electrónica de la Universidad Politécnica Salesiana sede Guayaquil.

- Se recomienda utilizar una licencia mensual para Ubidots para no perder los registros de los datos tomados durante las practicas.

- Se recomienda mejorar maqueta con más sensores que permitan la monitorización de varios parámetros como el acceso al datacenter, la presión atmosférica, calidad del aire dentro del datacenter, detección de gases, etc.

- Se recomienda utilizar plataformas web locales gratuitas como Grafana instaladas en una Raspberry Pi.

7 Bibliografía

Ashton, K. (2010). RELA TED C ONTENT RFID-Powered Handhelds Guide Visitors at Shanghai Expo Despite Sluggish Growth, Taiwan's RFID Industry Remains Committed Mobile RTLS Tracks Health-care Efficiency RFID Journal LIVE! 2010 Report, Part 2 That "Internet of Things" Thing. *That "Internet of Things" Thing-RFID Journal*. http://www.rfidjournal.com/article/print/4986

Cisco.com. (2024). *¿Qué es un centro de datos? - Cisco*. 2024. https://www.cisco.com/c/es_mx/solutions/data-center-virtualization/what-is-a-data-center.html

espressif.com. (2024). *ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems*. 2024. https://www.espressif.com/en/products/socs/esp32

Es.ubidots.com. (2024). *Acerca de Ubidots - Los valores que nos guían*. https://es.ubidots.com/about

iotsens.com. (2024). *IoTsens: Soluciones IoT, IA y Big Data*. https://www.iotsens.com/

Nurmalia, E., Bashar, L. B., & Joelianto, E. (2023). Performance of SNMP Protocol in Sensor Network Measurement and IoT Web-based HMI Features. *2023 8th International Conference on Instrumentation, Control, and Automation (ICA)*, 292–297. https://doi.org/10.1109/ICA58538.2023.10273086

PIZARRO PELÁEZ, J. (2020). *Internet de las cosas (IOT) con ESP. Manual práctico*. Ediciones Paraninfo, S.A. https://books.google.com.ec/books?id=B2oHEAAAQBAJ

Shafique, K., Khawaja, B. A., Sabir, F., Qazi, S., & Mustaqim, M. (2020). Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios. *IEEE Access*, *8*, 23022–23040. https://doi.org/10.1109/ACCESS.2020.2970118

Tiwari, A., Parate, N., Khamari, M., Jaiswal, A., Joshi, P., & Jadhav, P. (2022). IOT Based Health Care Monitoring and Facilitation. *2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22)*, 1–6. https://doi.org/10.1109/ICETET-

SIP-2254415.2022.9791720.

8 Anexos

8.1 Código de configuración en ESP32

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>
#include <LiquidCrystal_I2C.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <WiFi.h>
#include <ArduinoOTA.h>
#include "UbidotsEsp32Mqtt.h"

// **** Sensores de Temperatura y Humedad *****
#define DHT1_PIN 13
#define DHT2_PIN 14
#define DHTTYPE DHT22

// **** Encoder  ****
#define PIN_CLK 27
#define PIN_DT 26
#define PIN_SW 32

// **** BackLight LCD ****
#define BACKLIGHT_PIN 15

#define LIGHT_SENSOR_PIN 39
#define RAIN_SENSOR_PIN 36

// I2C device found at address 0x20 LCD
// I2C device found at address 0x48 ADS1115
// I2C device found at address 0x49 ADS1115
```

```
const unsigned int lcdColumns = 20;
const unsigned int lcdRows = 4;

const char* ssid = "TESIS-MACIAS-LOOR";
const char* password = "Jenticon22@@";
// const String wifi_ip = "192.168.1.3";
// const String wifi_mask = "255.255.255.0";
// const String wifi_gateway = "192.168.1.1";
// const String wifi_dns1 = "8.8.8.8";
// const String wifi_dns2 = "8.8.4.4";

const int PUBLISH_FREQUENCY = 4000; //5 segundos (Si es la versión gratis el tiempo se multiplica X4)

const char *UBIDOTS_TOKEN = "BBFF-5QU3hWY5QpFiDYOKGAOSMBNXHhTq7J";
const char *DEVICE_LABEL = "esp32-tesis";
const char *VARIABLE_LABEL_TEMP1 = "temperatura_area1";
const char *VARIABLE_LABEL_TEMP2 = "temperatura_area2";
const char *VARIABLE_LABEL_HUMD1 = "humedad_area1";
const char *VARIABLE_LABEL_HUMD2 = "humedad_area2";
const char *VARIABLE_LABEL_CURRT1 = "corriente_rack1";
const char *VARIABLE_LABEL_POWER1 = "potencia_rack1";
const char *VARIABLE_LABEL_CURRT2 = "corriente_rack2";
const char *VARIABLE_LABEL_POWER2 = "potencia_rack2";
const char *VARIABLE_LABEL_CURRT3 = "corriente_rack3";
const char *VARIABLE_LABEL_POWER3 = "potencia_rack3";
const char *VARIABLE_LABEL_ENERG1 = "energia_rack1";
const char *VARIABLE_LABEL_ENERG2 = "energia_rack2";
const char *VARIABLE_LABEL_ENERG3 = "energia_rack3";
const char *VARIABLE_LABEL_LIGHT = "light";
const char *VARIABLE_LABEL_RAIN = "rain";

const float FACTOR = 30.0; // 30A/1V
const float multiplier = 0.0625F; // ADC resolution
const float voltage_ac_in = 110;  // Voltage Network
```

```
const bool serial_print = true;
long timer_publish;

volatile int currentStateCLK;
volatile int lastStateCLK;
volatile int cont = 0;
int menu_number = 0;
String currentDir ="";

int cont_send_data_sensors = 0;

float temperature_measured[2];
float humidity_measured[2];

float currentRMS_measured[3];
float power_measured[3];
float energy_measured[3];

float light_measured;
float rain_measured;

long tiempo1;
long rawAdc1;
long minRaw1;
long maxRaw1;
bool flag1 = false;

long tiempo2;
long rawAdc2;
long minRaw2;
long maxRaw2;
bool flag2 = false;

long tiempo3;
long rawAdc3;
long minRaw3;
```

```
long maxRaw3;
bool flag3 = false;

Adafruit_ADS1115 ads1(0x48);
Adafruit_ADS1115 ads2(0x49);

DHT_Unified dht1(DHT1_PIN, DHTTYPE);
DHT_Unified dht2(DHT2_PIN, DHTTYPE);

Ubidots ubidots(UBIDOTS_TOKEN);

LiquidCrystal_I2C lcd(0x20,lcdColumns,lcdRows);

int getIpBlock(int index, String str){
  char separator = '.';
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex = str.length()-1;

  for(int i=0; i<=maxIndex && found<=index; i++){
    if(str.charAt(i)==separator || i==maxIndex){
      found++;
      strIndex[0] = strIndex[1]+1;
      strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }
  return found>index ? str.substring(strIndex[0], strIndex[1]).toInt() : 0;
}

IPAddress str2IP(String str){
  IPAddress ret( getIpBlock(0,str),getIpBlock(1,str),getIpBlock(2,str),getIpBlock(3,str));
  return ret;
}

void getCorrienteRak1(){
  if(!flag1){
```

```
    tiempo1 = millis();
    rawAdc1 = ads1.readADC_Differential_0_1();
    minRaw1 = rawAdc1;
    maxRaw1 = rawAdc1;
    flag1 = true;
  }
  if(millis() - tiempo1 < 1000){
    rawAdc1 = ads1.readADC_Differential_0_1();
    maxRaw1 = maxRaw1 > rawAdc1 ? maxRaw1 : rawAdc1;
    minRaw1 = minRaw1 < rawAdc1 ? minRaw1 : rawAdc1;
  }else{
    maxRaw1 = maxRaw1 > -minRaw1 ? maxRaw1 : -minRaw1;
    float voltagePeak = maxRaw1 * multiplier / 1000;
    float voltageRMS = voltagePeak * 0.70710678118;
    currentRMS_measured[0] = voltageRMS * FACTOR;
    flag1 = false;
  }
}

void getCorrienteRak2(){
  if(!flag2){
    tiempo2 = millis();
    rawAdc2 = ads1.readADC_Differential_2_3();
    minRaw2 = rawAdc2;
    maxRaw2 = rawAdc2;
    flag2 = true;
  }
  if(millis() - tiempo2 < 1000){
    rawAdc2 = ads1.readADC_Differential_2_3();
    maxRaw2 = maxRaw2 > rawAdc2 ? maxRaw2 : rawAdc2;
    minRaw2 = minRaw2 < rawAdc2 ? minRaw2 : rawAdc2;
  }else{
    maxRaw2 = maxRaw2 > -minRaw2 ? maxRaw2 : -minRaw2;
    float voltagePeak = maxRaw2 * multiplier / 1000;
    float voltageRMS = voltagePeak * 0.70710678118;
    currentRMS_measured[1] = voltageRMS * FACTOR;
```

```
    flag2 = false;
  }
}

void getCorrienteRak3(){
  if(!flag3){
    tiempo3 = millis();
    rawAdc3 = ads2.readADC_Differential_0_1();
    minRaw3 = rawAdc3;
    maxRaw3 = rawAdc3;
    flag3 = true;
  }
  if(millis() - tiempo3 < 1000){
    rawAdc3 = ads2.readADC_Differential_0_1();
    maxRaw3 = maxRaw3 > rawAdc3 ? maxRaw3 : rawAdc3;
    minRaw3 = minRaw3 < rawAdc3 ? minRaw3 : rawAdc3;
  }else{
    maxRaw3 = maxRaw3 > -minRaw3 ? maxRaw3 : -minRaw3;
    float voltagePeak = maxRaw3 * multiplier / 1000;
    float voltageRMS = voltagePeak * 0.70710678118;
    currentRMS_measured[2] = voltageRMS * FACTOR;
    flag3 = false;
  }
}

void callback(char *topic, byte *payload, unsigned int length){
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
```

```
void read_sensors(){

  light_measured = map(4095-analogRead(LIGHT_SENSOR_PIN), 0, 4095, 0, 100);

  rain_measured = map(4095-analogRead(RAIN_SENSOR_PIN), 0, 4095, 0, 100);

  sensors_event_t eventDht1;
  sensors_event_t eventDht2;
-
  dht1.temperature().getEvent(&eventDht1);
  if (!isnan(eventDht1.temperature)){
    temperature_measured[0] = eventDht1.temperature;
  }
  dht1.humidity().getEvent(&eventDht1);
  if (!isnan(eventDht1.temperature)){
    humidity_measured[0] = eventDht1.relative_humidity;
  }
  dht2.temperature().getEvent(&eventDht2);
  if (!isnan(eventDht2.temperature)){
    temperature_measured[1] = eventDht2.temperature;
  }
  dht2.humidity().getEvent(&eventDht2);
  if (!isnan(eventDht2.relative_humidity)){
    humidity_measured[1] = eventDht2.relative_humidity;
  }

  getCorrienteRak1();
  power_measured[0] = voltage_ac_in * currentRMS_measured[0];
  energy_measured[0] = energy_measured[0] + (power_measured[0] * (1.0 / 60/60/1000));

  getCorrienteRak2();
  power_measured[1] = voltage_ac_in * currentRMS_measured[1];
  energy_measured[1] = energy_measured[1] + (power_measured[1] * (1.0 / 60/60/1000));

  getCorrienteRak3();
  power_measured[2] = voltage_ac_in * currentRMS_measured[2];
```

```
  energy_measured[2] = energy_measured[2] + (power_measured[2] * (1.0 / 60/60/1000));
}

void updateEncoder(){
  currentStateCLK = digitalRead(PIN_CLK);
  if (currentStateCLK != lastStateCLK  && currentStateCLK == 1){
    if (digitalRead(PIN_DT) != currentStateCLK) {
      cont -= 1;
      currentDir ="CCW";
    } else {
      cont += 1;
      currentDir ="CW";
    }
    if(cont<0)
      cont = 0;
    if(cont>=3)
     cont = 3;
  }

  lastStateCLK = currentStateCLK;
}

void setup(){
  pinMode(BACKLIGHT_PIN, OUTPUT);
  pinMode(PIN_CLK,INPUT_PULLUP);
  pinMode(PIN_DT,INPUT_PULLUP);
  pinMode(PIN_SW,INPUT_PULLUP);

  lcd.init();
  digitalWrite(BACKLIGHT_PIN, HIGH);
  lcd.setCursor(0,2);
  lcd.print("     Starting...    ");
  delay(3000);

  Serial.begin(57600);
  ads1.setGain(GAIN_TWO);    // ±2.048V  1 bit = 0.0625mV
```

```
  ads1.begin();
  ads2.setGain(GAIN_TWO);    // ±2.048V  1 bit = 0.0625mV
  ads2.begin();

  dht1.begin();
  dht2.begin();
  sensor_t sensorDht1;
  dht1.temperature().getSensor(&sensorDht1);
  dht1.humidity().getSensor(&sensorDht1);
  sensor_t sensorDht2;
  dht2.temperature().getSensor(&sensorDht2);
  dht2.humidity().getSensor(&sensorDht2);

  // WiFi.config(str2IP(wifi_ip), str2IP(wifi_gateway), str2IP(wifi_mask), str2IP(wifi_dns1),
str2IP(wifi_dns2));
  // WiFi.mode(WIFI_STA);
  // WiFi.begin(ssid, password);
  ubidots.connectToWifi(ssid, password);
  ubidots.setCallback(callback);
  ubidots.setup();
  ubidots.connect();

  int wifi_cont = 0;

  lcd.clear();
  lcd.setCursor(0,2);
  lcd.print(" Wifi Connecting... ");
  while (WiFi.status() != WL_CONNECTED){
    if(wifi_cont > 15){
      Serial.println();
      lcd.clear();
      lcd.setCursor(0,2);
      lcd.print("    Rebooting...  ");
      delay(1000);
      ESP.restart();
    }
```

```
  wifi_cont++;
  Serial.print(".");
  delay(500);
}
Serial.println();
WiFi.persistent(true);
lcd.clear();
lcd.setCursor(0,2);
lcd.print("  Wifi Connected!  ");

ArduinoOTA.setPort(3232);
ArduinoOTA.setHostname("Tesis-Macias-Loor");
ArduinoOTA
.onStart([]() {
  String type;
  if (ArduinoOTA.getCommand() == U_FLASH)
    type = "sketch";
  else
    type = "filesystem";
})
.onEnd([]() {
  ESP.restart();
})
.onProgress([](unsigned int progress, unsigned int total) {

})
.onError([](ota_error_t error){
  ESP.restart() ;
});

ArduinoOTA.begin();

lastStateCLK = digitalRead(PIN_CLK);
attachInterrupt(digitalPinToInterrupt(PIN_CLK), updateEncoder, CHANGE);
attachInterrupt(digitalPinToInterrupt(PIN_DT), updateEncoder, CHANGE);
lcd.clear();
```

```
  timer_publish = millis();
}

void loop(){

  read_sensors();

  if (!ubidots.connected()){
    ubidots.reconnect();
  }

  if (millis() - timer_publish > PUBLISH_FREQUENCY){ // triggers the routine every 3 seconds

    // Version Gratuita más lenta

    // if (cont_send_data_sensors >=4)
    //   cont_send_data_sensors = 0;

    // switch (cont_send_data_sensors){
    // case 0:
    //   ubidots.add(VARIABLE_LABEL_HUMD1, humidity_measured[0]);
    //   ubidots.add(VARIABLE_LABEL_HUMD2, humidity_measured[1]);
    //   ubidots.add(VARIABLE_LABEL_TEMP1, temperature_measured[0]);
    //   ubidots.add(VARIABLE_LABEL_TEMP2, temperature_measured[1]);
    //   ubidots.publish(DEVICE_LABEL);
    //   break;

    // case 1:
    //   ubidots.add(VARIABLE_LABEL_CURRT1, currentRMS_measured[0]);
    //   ubidots.add(VARIABLE_LABEL_POWER1, power_measured[0]);
    //   ubidots.add(VARIABLE_LABEL_CURRT2, currentRMS_measured[1]);
    //   ubidots.add(VARIABLE_LABEL_POWER2, power_measured[1]);
    //   ubidots.publish(DEVICE_LABEL);
    //   break;
    // case 2:
    //   ubidots.add(VARIABLE_LABEL_CURRT3, currentRMS_measured[2]);
```

```
//   ubidots.add(VARIABLE_LABEL_POWER3, power_measured[2]);
//   ubidots.add(VARIABLE_LABEL_ENERG1, energy_measured[0]);
//   ubidots.add(VARIABLE_LABEL_ENERG2, energy_measured[1]);
//   ubidots.add(VARIABLE_LABEL_ENERG3, energy_measured[2]);
//   ubidots.publish(DEVICE_LABEL);
//   break;
// case 3:
//   ubidots.add(VARIABLE_LABEL_LIGHT, light_measured);
//   ubidots.add(VARIABLE_LABEL_RAIN, rain_measured);
//   ubidots.publish(DEVICE_LABEL);
//   break;
// default:
//   break;
// }

// Version de Pago Rapida
if(ubidots.connected()){
  ubidots.add(VARIABLE_LABEL_HUMD1, humidity_measured[0]);
  ubidots.add(VARIABLE_LABEL_HUMD2, humidity_measured[1]);
  ubidots.add(VARIABLE_LABEL_TEMP1, temperature_measured[0]);
  ubidots.add(VARIABLE_LABEL_TEMP2, temperature_measured[1]);
  ubidots.publish(DEVICE_LABEL);
  ubidots.add(VARIABLE_LABEL_CURRT1, currentRMS_measured[0]);
  ubidots.add(VARIABLE_LABEL_POWER1, power_measured[0]);
  ubidots.add(VARIABLE_LABEL_CURRT2, currentRMS_measured[1]);
  ubidots.add(VARIABLE_LABEL_POWER2, power_measured[1]);
  ubidots.publish(DEVICE_LABEL);
  ubidots.add(VARIABLE_LABEL_CURRT3, currentRMS_measured[2]);
  ubidots.add(VARIABLE_LABEL_POWER3, power_measured[2]);
  ubidots.add(VARIABLE_LABEL_ENERG1, energy_measured[0]);
  ubidots.add(VARIABLE_LABEL_ENERG2, energy_measured[1]);
  ubidots.publish(DEVICE_LABEL);
  ubidots.add(VARIABLE_LABEL_ENERG3, energy_measured[2]);
  ubidots.add(VARIABLE_LABEL_LIGHT, light_measured);
  ubidots.add(VARIABLE_LABEL_RAIN, rain_measured);
  ubidots.publish(DEVICE_LABEL);
```

```
}

if(serial_print){
  Serial.print(F("Current1: "));
  Serial.print(currentRMS_measured[0]);
  Serial.print(F("[A]"));
  Serial.print(F(" | "));
  Serial.print(F("Power1: "));
  Serial.print(power_measured[0]);
  Serial.print(F("[W]"));
  Serial.print(F(" | "));
  Serial.print(F("Energy1: "));
  Serial.print(energy_measured[0]);
  Serial.println(F("[KW/h]"));

  Serial.print(F("Current2: "));
  Serial.print(currentRMS_measured[1]);
  Serial.print(F("[A]"));
  Serial.print(F(" | "));
  Serial.print(F("Power2: "));
  Serial.print(power_measured[1]);
  Serial.print(F("[W]"));
  Serial.print(F(" | "));
  Serial.print(F("Energy2: "));
  Serial.print(energy_measured[1]);
  Serial.println(F("[KW/h]"));

  Serial.print(F("Current3: "));
  Serial.print(currentRMS_measured[2]);
  Serial.print(F("[A]"));
  Serial.print(F(" | "));
  Serial.print(F("Power3: "));
  Serial.print(power_measured[2]);
  Serial.print(F("[W]"));
  Serial.print(F(" | "));
  Serial.print(F("Energy3: "));
```

```
    Serial.print(energy_measured[2]);
    Serial.println(F("[KW/h]"));

    Serial.print(F("Temperature1: "));
    Serial.print(temperature_measured[0]);
    Serial.print(F("[°C]"));
    Serial.print(F(" | "));
    Serial.print(F("Humdidity1: "));
    Serial.print(humidity_measured[0]);
    Serial.println(F("[%]"));

    Serial.print(F("Temperature2: "));
    Serial.print(temperature_measured[1]);
    Serial.print(F("[°C]"));
    Serial.print(F(" | "));
    Serial.print(F("Humdidity2: "));
    Serial.print(humidity_measured[1]);
    Serial.println(F("[%]"));

    Serial.print(F("Light: "));
    Serial.print(light_measured);
    Serial.print(F("[%]"));
    Serial.print(F(" | "));
    Serial.print(F("Rain: "));
    Serial.print(rain_measured);
    Serial.println(F("[%]"));
  }
  cont_send_data_sensors++;
  timer_publish = millis();
}
char buffer[20];
char data[6];
switch (menu_number){
case 0:
      switch (cont){
      case 0:
```

```
      lcd.setCursor(0,cont);
      lcd.print(">");
      lcd.setCursor(0,1);
      lcd.print(" ");
      lcd.setCursor(0,2);
      lcd.print(" ");
      lcd.setCursor(0,3);
      lcd.print(" ");

      lcd.setCursor(2,0);
      lcd.print("Light and rain");
      lcd.setCursor(2,1);
      lcd.print("Temp and Humd");
      lcd.setCursor(2,2);
      lcd.print("Current and Power");
      lcd.setCursor(2,3);
      lcd.print("Energy");
      if(!digitalRead(PIN_SW)){
        while (!digitalRead(PIN_SW)){
        }
        lcd.clear();
        menu_number = cont+4;
      }
      break;
  case 1:
      lcd.setCursor(0,0);
      lcd.print(" ");
      lcd.setCursor(0,cont);
      lcd.print(">");
      lcd.setCursor(0,2);
      lcd.print(" ");
      lcd.setCursor(0,3);
      lcd.print(" ");

      lcd.setCursor(2,0);
      lcd.print("Light and rain");
```

```
      lcd.setCursor(2,1);
      lcd.print("Temp and Humd");
      lcd.setCursor(2,2);
      lcd.print("Current and Power");
      lcd.setCursor(2,3);
      lcd.print("Energy");
      if(!digitalRead(PIN_SW)){
        while (!digitalRead(PIN_SW)){
        }
        lcd.clear();
        menu_number = cont;
      }
      break;
  case 2:
      lcd.setCursor(0,0);
      lcd.print(" ");
      lcd.setCursor(0,1);
      lcd.print(" ");
      lcd.setCursor(0,cont);
      lcd.print(">");
      lcd.setCursor(0,3);
      lcd.print(" ");

      lcd.setCursor(2,0);
      lcd.print("Light and rain");
      lcd.setCursor(2,1);
      lcd.print("Temp and Humd");
      lcd.setCursor(2,2);
      lcd.print("Current and Power");
      lcd.setCursor(2,3);
      lcd.print("Energy");
      if(!digitalRead(PIN_SW)){
        while (!digitalRead(PIN_SW)){
        }
        lcd.clear();
        menu_number = cont;
```

```
            }
            break;
        case 3:
            lcd.setCursor(0,0);
            lcd.print(" ");
            lcd.setCursor(0,1);
            lcd.print(" ");
            lcd.setCursor(0,2);
            lcd.print(" ");
            lcd.setCursor(0,cont);
            lcd.print(">");

            lcd.setCursor(2,0);
            lcd.print("Light and rain");
            lcd.setCursor(2,1);
            lcd.print("Temp and Humd");
            lcd.setCursor(2,2);
            lcd.print("Current and Power");
            lcd.setCursor(2,3);
            lcd.print("Energy");
            if(!digitalRead(PIN_SW)){
              while (!digitalRead(PIN_SW)){
              }
              lcd.clear();
              menu_number = cont;
            }
            break;
        default:
          break;
        }
        break;
    case 1:
        lcd.setCursor(0,0);
        dtostrf(temperature_measured[0], 4,2,data);
        sprintf(buffer,"  Temp1: %s[%cC]",data,223);
        lcd.print(buffer);
```

```
      lcd.setCursor(0,1);
      dtostrf(humidity_measured[0], 4,2,data);
      sprintf(buffer,"  Humd1: %s[%c]",data,37);
      lcd.print(buffer);
      lcd.setCursor(0,2);
      dtostrf(temperature_measured[1], 4,2,data);
      sprintf(buffer,"  Temp2: %s[%cC]",data,223);
      lcd.print(buffer);
      lcd.setCursor(0,3);
      dtostrf(humidity_measured[1], 4,2,data);
      sprintf(buffer,"  Humd2: %s[%c]",data,37);
      lcd.print(buffer);
      if(!digitalRead(PIN_SW)){
        long cont_exit = 0;
        while (!digitalRead(PIN_SW) and cont_exit <= 250){
          cont_exit++;
          delay(10);
        }
        if(cont_exit >=250){
          lcd.clear();
          menu_number = 0;
          while (!digitalRead(PIN_SW)){
          }
        }
      }
      break;
  case 2:
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(" Current and Power ");
      lcd.setCursor(0,1);
      dtostrf(currentRMS_measured[0], 4,1,data);
      sprintf(buffer,"I1:%sA",data);
      lcd.print(buffer);
      lcd.setCursor(9,1);
      dtostrf(power_measured[0], 4,1,data);
```

```
      sprintf(buffer,"P1: %sW",data);
      lcd.print(buffer);
      lcd.setCursor(0,2);
      dtostrf(currentRMS_measured[1], 4,1,data);
      sprintf(buffer,"I2:%sA",data);
      lcd.print(buffer);
      lcd.setCursor(9,2);
      dtostrf(power_measured[1], 4,1,data);
      sprintf(buffer,"P2: %sW",data);
      lcd.print(buffer);
      lcd.setCursor(0,3);
      dtostrf(currentRMS_measured[2], 4,1,data);
      sprintf(buffer,"I3:%sA",data);
      lcd.print(buffer);
      lcd.setCursor(9,3);
      dtostrf(power_measured[2], 4,1,data);
      sprintf(buffer,"P3: %sW",data);
      lcd.print(buffer);
      if(!digitalRead(PIN_SW)){
        long cont_exit = 0;
        while (!digitalRead(PIN_SW) and cont_exit <= 250){
          cont_exit++;
          delay(10);
        }
        if(cont_exit >=250){
          lcd.clear();
          menu_number = 0;
          while (!digitalRead(PIN_SW)){
          }
        }
      }
      delay(1000);
      break;
  case 3:
      lcd.setCursor(0,0);
      lcd.print("****** Energy ******");
```

```
      lcd.setCursor(1,1);
      dtostrf(energy_measured[0], 4,1,data);
      sprintf(buffer,"Energy1:%sKw/h",data);
      lcd.print(buffer);
      lcd.setCursor(1,2);
      dtostrf(energy_measured[1], 4,1,data);
      sprintf(buffer,"Energy2:%sKw/h",data);
      lcd.print(buffer);
      lcd.setCursor(1,3);
      dtostrf(energy_measured[2], 4,1,data);
      sprintf(buffer,"Energy3:%sKw/h",data);
      lcd.print(buffer);
      if(!digitalRead(PIN_SW)){
        long cont_exit = 0;
        while (!digitalRead(PIN_SW) and cont_exit <= 250){
          cont_exit++;
          delay(10);
        }
        if(cont_exit >=250){
          lcd.clear();
          menu_number = 0;
          while (!digitalRead(PIN_SW)){
          }
        }
      }
      break;
  case 4:
      lcd.setCursor(0,1);
      dtostrf(light_measured, 4,2,data);
      sprintf(buffer,"  Light: %s[%c]",data,37);
      lcd.print(buffer);
      lcd.setCursor(0,2);
      dtostrf(rain_measured, 4,2,data);
      sprintf(buffer,"  Rain: %s[%c]",data,37);
      lcd.print(buffer);
      if(!digitalRead(PIN_SW)){
```

```
        long cont_exit = 0;
        while (!digitalRead(PIN_SW) and cont_exit <= 250){
          cont_exit++;
          delay(10);
        }
        if(cont_exit >=250){
          lcd.clear();
          menu_number = 0;
          while (!digitalRead(PIN_SW)){
          }
        }
      }
      break;
  default:
    break;
  }

  if (ubidots.connected()){
    ubidots.loop();
  }
  ArduinoOTA.handle();
}

// ****** TEST LCD ***********

// #define BACKLIGHT_PIN 15

// // I2C device found at address 0x20 LCD

// const unsigned int lcdColumns = 20;
// const unsigned int lcdRows = 4;

// LiquidCrystal_I2C lcd(0x20,lcdColumns,lcdRows);

// void setup() {
//   pinMode(BACKLIGHT_PIN, OUTPUT);
```

```
//   Serial.begin(57600);

//   lcd.init();
//   digitalWrite(BACKLIGHT_PIN, HIGH);
//   lcd.backlight();
//   lcd.setCursor(0,2);
//   lcd.print("     Starting...    ");
//   delay(3000);
//   lcd.clear();
// }

// void loop() {

// }

// ****** SCAN I2C ***********
// void setup() {
//   Wire.begin();
//   Serial.begin(57600);
//   Serial.println("\nI2C Scanner");
// }

// void loop() {
//   byte error, address;
//   int nDevices;
//   Serial.println("Scanning...");
//   nDevices = 0;
//   for(address = 1; address < 127; address++ ) {
//     Wire.beginTransmission(address);
//     error = Wire.endTransmission();
//     if (error == 0) {
//       Serial.print("I2C device found at address 0x");
//       if (address<16) {
//         Serial.print("0");
//       }
```

```
//      Serial.println(address,HEX);
//      nDevices++;
//    }
//    else if (error==4) {
//      Serial.print("Unknow error at address 0x");
//      if (address<16) {
//        Serial.print("0");
//      }
//      Serial.println(address,HEX);
//    }
//  }
//  if (nDevices == 0) {
//    Serial.println("No I2C devices found\n");
//  }
//  else {
//    Serial.println("done\n");
//  }
//  delay(5000);
// }
```