UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES

TEMA:

Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil.

AUTOR:
Gil Cevallos, Mario Javier

Trabajo de titulación previo a la obtención del grado de
MAGISTER EN TELECOMUNICACIONES

TUTOR:
Ing. Bohórquez Escobar, Celso Bayardo, PhD.

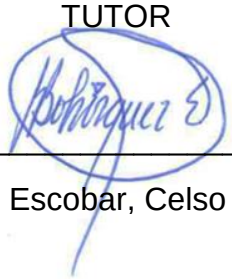
Guayaquil, Ecuador
13 de marzo del 2025

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

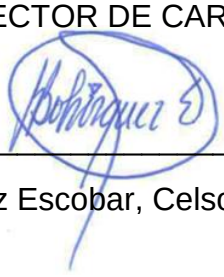
Certificamos que el presente Trabajo de Integración Curricular fue realizado en su totalidad por el Sr. **Gil Cevallos, Mario Javier**, como requerimiento para la obtención del Título de Magíster en Telecomunicaciones.

TUTOR



Ing. Bohórquez Escobar, Celso Bayardo. PHD.

DIRECTOR DE CARRERA



Ing. Bohórquez Escobar, Celso Bayardo. PHD.

Guayaquil, 13 de marzo del 2025

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Gil Cevallos, Mario Javier**

DECLARO QUE:

El trabajo de Integración Curricular **Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil,** previo a la obtención del Título de **Magíster en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Integración Curricular referido.

Guayaquil, 13 de marzo del 2025

EL AUTOR

_____

Gil Cevallos, Mario Javier

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Gil Cevallos, Mario Javier**

Autorizo a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del Trabajo de Integración Curricular: **Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

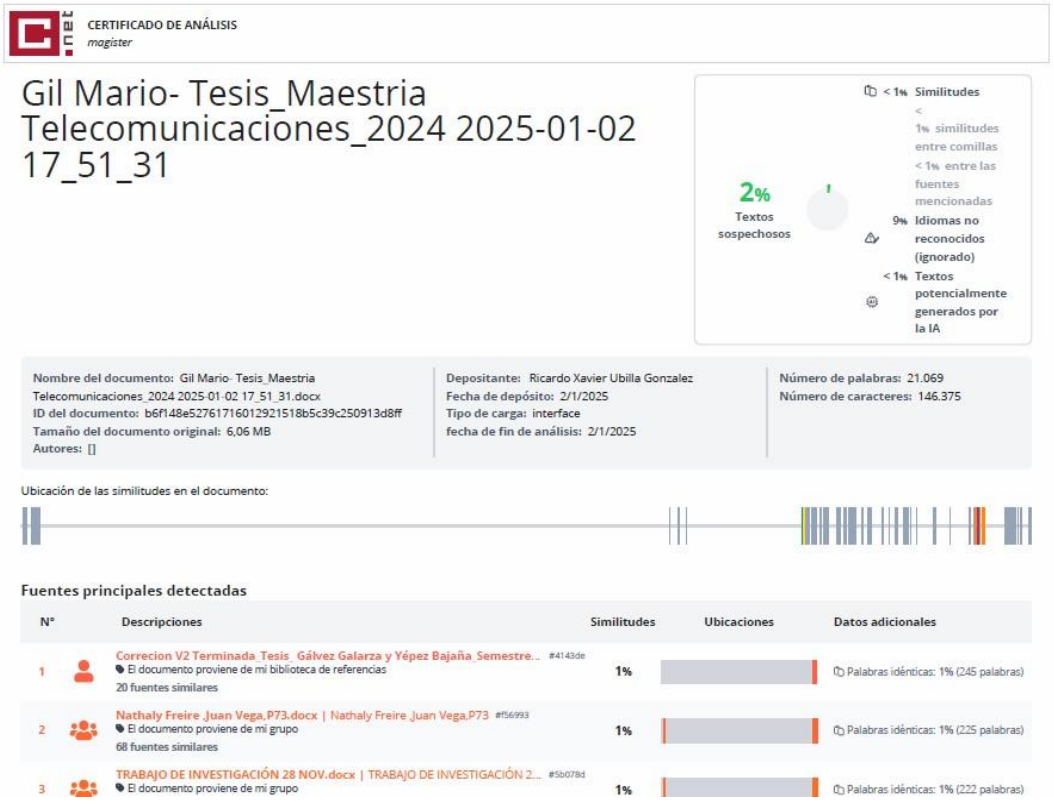
Guayaquil, 13 de marzo del 2025

EL AUTOR

_____

Gil Cevallos, Mario Javier

REPORTE DE COMPILATIO



CERTIFICADO DE ANÁLISIS
magister

Gil Mario- Tesis_Maestria Telecomunicaciones_2024 2025-01-02 17_51_31

2% Textos sospechosos

< 1% Similitudes
< 1% similitudes entre comillas
< 1% entre las fuentes mencionadas
9% Idiomas no reconocidos (ignorado)
< 1% Textos potencialmente generados por la IA

Nombre del documento: Gil Mario- Tesis_Maestria Telecomunicaciones_2024 2025-01-02 17_51_31.docx
ID del documento: b6f148e5276171601292151 8b5c39c250913d8ff
Tamaño del documento original: 6,06 MB
Autores: []

Depositante: Ricardo Xavier Ubilla Gonzalez
Fecha de depósito: 2/1/2025
Tipo de carga: interface
fecha de fin de análisis: 2/1/2025

Número de palabras: 21.069
Número de caracteres: 146.375

Ubicación de las similitudes en el documento:

Fuentes principales detectadas

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	Correcion V2 Terminada_Tesis_Gálvez Galarza y Yépez Bajaña_Semestre... #4143de El documento proviene de mi biblioteca de referencias 20 fuentes similares	1%		Palabras idénticas: 1% (245 palabras)
2	Nathaly Freire_Juan Vega.P73.docx | Nathaly Freire_Juan Vega.P73 #f56993 El documento proviene de mi grupo 68 fuentes similares	1%		Palabras idénticas: 1% (225 palabras)
3	TRABAJO DE INVESTIGACIÓN 28 NOV.docx | TRABAJO DE INVESTIGACIÓN 2... #5b078d El documento proviene de mi grupo	1%		Palabras idénticas: 1% (222 palabras)

Reporte COMPILATIO del estudiante Gil Cevallos Mario Javier de la MAESTRÍA EN TELECOMUNICACIONES del Sistema de Posgrado con el tema **Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil**, mismos que se encuentra al 2% de coincidencias.

Ing. Bayardo Bohórquez Escobar, PHD.
DOCENTE-TUTOR

AGRADECIMIENTO

Agradezco ante todo a Dios, quien es el que me ha permitido seguir en este mundo junto a mi preciada familia, a mi querida esposa Marianella Robles y mis amados hijos Dereck, Paula y Francisco, quienes han sido mi fuente de inspiración y fuerza, para seguir adelante en esta meta tan ansiada, que constituye como uno de mis objetivos de vida.

Agradezco también al ingeniero Bayardo Bohórquez, quien ha sido un excelente docente y tutor a lo largo de mi carrera, el cual me ha ayuda hasta lo último para poder culminar con mis estudios.

DEDICATORIA

Este trabajo se lo dedico primeramente a Dios por darme la salud y también a mi apreciada familia por acompañarme a lo largo de esta travesía; con esto espero demostrar y ser un ejemplo de constancia para mis hijos en el futuro, para que no se rindan y siempre cumplan todas sus metas en la vida que tiene presentes.

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
SISTEMA DE POSGRADO
MAESTRÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f. _____

Ing. Bohórquez Escobar, Celso Bayardo. PhD.

TUTOR

f. _____

Ing. Ubilla González, Ricardo Xavier. MSc.

REVISOR

f. _____

Ing. Bohórquez Heras, Diana Carolina. MSc.

REVISOR

f. _____

Ing. Bohórquez Escobar, Celso Bayardo. PhD.

DIRECTOR DEL PROGRAMA

ÍNDICE GENERAL

INDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 2:

Capítulo 3:

Resumen

El presente trabajo de integración curricular se basa en el diseño de un sistema IoT para el monitoreo de interferencia y ruido en la ciudad de Guayaquil, utilizando una red de nodos sensores con tecnología LoRaWAN y ESP32. Mediante el desarrollo de la interfaz principal para el monitoreo de interferencia y ruido en la ciudad de Guayaquil mediante el uso del entorno de programación de Arduino IDE. También se realiza el diseño de un entorno gráfico utilizando el software LabVIEW para gestionar las variables recibidas en la plataforma IoT Ubidots y a su vez se evalúa el desempeño, viabilidad y estimar el costo del diseño de una red de sensores inalámbricos utilizando el módulo ESP32. Este estudio establecerá un sistema IoT con una red de nodos sensores con tecnología LoRaWAN con ESP32, que permitirá el monitoreo constante de la contaminación y ruido en la ciudad de Guayaquil, brindando una información con mayor precisión, que servirá para tomar decisiones y ejecutar medidas públicas idóneas para reducir dichos problemas ambientales. La metodología utilizada es la bibliográfica de la literatura existente sobre sistemas IoT para el monitoreo del entorno, nodos sensores, tecnologías LoRaWAN y ESP32.

Palabras claves: ESP32, Lora, IoT, Arduino IDE, sensores, monitoreo.

ABSTRACT

This curricular integration work is based on the design of an IoT system for monitoring interference and noise in the city of Guayaquil, using a network of sensor nodes with LoRaWAN and ESP32 technology. Through the development of the main interface for monitoring interference and noise in the city of Guayaquil using the Arduino IDE programming environment. The design of a graphical environment is also carried out using the LabVIEW software to manage the variables received on the Ubidots IoT platform and in turn the performance, feasibility and estimate of the cost of the design of a wireless sensor network using the ESP32 module are evaluated. This study will establish an IoT system with a network of sensor nodes with LoRaWAN technology with ESP32, which will allow constant monitoring of pollution and noise in the city of Guayaquil, providing more accurate information, which will be used to make decisions and implement public measures. suitable to reduce these environmental problems. The methodology used is bibliographical one of the existing literatures on IoT systems for environmental monitoring, sensor nodes, LoRaWAN and ESP32 technologies.

Keywords: ESP32, Lora, IoT, Arduino IDE, sensors, monitoring.

Capítulo 1: Descripción general del trabajo de titulación

1.1 Introducción

El desarrollo exponencial de las tecnologías de la información y la comunicación han revolucionado la manera en la que son abordadas y vigiladas las condiciones urbanas. Debido a esto, el Internet de las cosas ha sido desarrollado como una piedra angular para las smart cities (ciudades inteligentes), permitiendo la conexión de distintas fuentes de información para su manejo en tiempo real. En esencia, el IoT lidera el ambiente urbano, al posibilitarles a las ciudades alcanzar desde un manejo más eficiente del tráfico vehicular, hasta un control medioambiental integral. En términos sencillos, los sistemas IoT en ciudades hacen que su población no solo consuma de forma más inteligente los recursos, sino que también vivan en un lugar más sustentable y seguro (Callejón, 2022).

Con una densidad de población tan alta, las ciudades, como Guayaquil, se enfrentan a uno de los desafíos más problemáticos: las interferencias electromagnéticas y un alto nivel de ruido de fondo ambiental. Además del hecho de que esos fenómenos estudian el bienestar de los residentes locales, dañan su salud, provocan estrés, problemas de sueño, enfermedades cardiovasculares, también influyen en la operación de infraestructuras críticas, como sistemas de transporte, telecomunicaciones, servicios de emergencia, etc. Por lo tanto, la tecnología desarrolla un sentido en tiempo real de las soluciones que serán eficientes y adoptará todos los desafíos relacionados (Huertas, 2022).

Dado el escenario descrito anteriormente, la tecnología LoRaWAN es la solución ideal para implementar una red de sensores en una ciudad. Esta tecnología es adecuada para la implementación de una multitud de nodos, ya que es capaz de enviar datos a distancias largas con un mínimo consumo de energía. En otras palabras, se puede usar en una ciudad donde los sensores necesitan ser dispersados en grandes cantidades pero con la sostenibilidad de la red en mente. Con la implementación que utiliza el microcontrolador ESP32. Como resultado, se puede monitorear la interferencia y el ruido en la ciudad de Guayaquil de manera rentable debido a un sistema robusto, flexible y escalable (Maldonado & Tigrero, 2023).

El presente proyecto plantea el uso de la combinación de las tecnologías LoRaWAN y ESP32 para la generación de una red de nodos sensores, que sean capaces de ofrecer los datos exactos y en tiempo real. A través de la información recabada, se generará el recurso adecuado para que la autoridad y el resto de los responsables tomen decisiones informadas que les permitan administrar y manejar la variabilidad de los niveles de ruido y de la interferencia en la ciudad, lo cual les facilitará un manejo más efectivo y proactivo del medio ambiente.

1.2 Antecedentes

En este contexto, el monitoreo ambiental de los factores de las ciudades se ha vuelto crucial, ya que es necesario amortiguar estos y muchos otros impactos negativos del rápido crecimiento urbano. Los problemas críticos en esta área incluyen el de la contaminación acústica y las

interferencias electromagnéticas. Ambos afectan tanto el bienestar de las personas como la eficiencia de las infraestructuras tecnológicas. En cuanto a los últimos, la densidad de señales electromagnéticas se ha disparado ya que, por ejemplo, en Guayaquil, Ecuador, el número de dispositivos electrónicos se multiplicó y la infraestructura de las redes de comunicación se amplió. Como resultado, los monitores de ambas categorías necesitan ser más avanzados (Gómez, 2022).

Las redes de sensores inalámbricos han demostrado ser una herramienta muy útil para la gestión de un entorno una vez ilustrado. En efecto, estas permiten, monitorear datos en tiempo real acerca de las condiciones de una ciudad sin necesidad de grandes costos para instalarlos. Entre todos los protocolos de dicho tipo, el LoRaWAN ha comenzado a cobrar popularidad gracias a su habilidad de trabajo en bandas de frecuencia no licenciadas, lo que disminuye los costos asociados al despliegue de la red, y aislamiento de señal muy importante, especialmente en una ciudad. El atractivo menor principal, el consumo energético, significa que los nodos sensores pueden operar por mucho tiempo sin la duplicación frecuente, lo que también hace el sistema más sostenible (Andrade & Brigss, 2023).

Además, el microcontrolador ESP32 es un punto de referencia en la implementación de dispositivos de IoT debido a su versatilidad. Durante su operación, el microcontrolador puede implementar varios protocolos de comunicación, lo que le permite funcionar sin problemas en una red masiva y, al mismo tiempo, procesar datos en tiempo real. Además, el bajo precio de

este dispositivo lo convierte en un competidor preferido para la implementación de sistemas de monitoreo en ciudades extensas, como Guayaquil. Después de eso, es posible conectar estos sistemas a servicios de almacenamiento masivo de datos en la nube, como Ubidots, para no solo recopilar datos, sino también analizarlos y presentarlos en una forma humana. De este modo, el uso de esta tecnología facilita la toma de decisiones (Loor & Macías, 2024).

En cuanto a los estudios y trabajos en el contexto local, son los proyectos previos realizados en Guayaquil, y como se ha demostrado en los estudios anteriores, tanto la contaminación acústica como las interrupciones por interferencia electromagnética también son graves problemas que deben abordarse urgentemente. Los estudios llevados a cabo también han revelado que la falta de provisión de equipos de monitoreo adecuado en tiempo real impide en gran medida a las autoridades tomar medidas correctivas en un sistema tan sistemático y adecuado. Por esta razón, la propuesta del sistema IoT ofrece a la ciudad la oportunidad de aumentar la calidad de vida a través del monitoreo permanente y subsiguiente mejora del entorno seguro (Morrón & Eduardo, 2024).

1.3 Definición del Problema

El constante desarrollo urbano en expansión y la actividad industrial en la ciudad de Guayaquil son las responsables de una mayor contaminación del aire y el ruido, perjudicando la salud de la población y el ecosistema. La falta de un sistema unificado adecuado para recopilar estos datos ambientales de

manera continua y en tiempo real, genera que la información no sea precisa y oportuna haciendo imposible implementar medidas correctivas efectivas.

1.4 Justificación del Problema

La creación de un sistema de Internet de las cosas (IoT) para el monitoreo de la contaminación y el ruido en Guayaquil no solo proporcionará al gobierno municipal datos esenciales para un mejor enfoque de sostenibilidad medio ambiental. También, mejorará la calidad de vida de sus ciudadanos. La capacidad de monitorear activamente permitirá tomar decisiones inmediatas e informadas a las autoridades empresariales y gubernamentales para abordar los problemas que surjan. Esto convierte a Guayaquil en una ciudad atenta a la innovación y al bienestar de sus ciudadanos, incluido el medio ambiente.

1.5.1 Objetivo general

Diseñar un sistema IoT para el monitoreo de interferencia y ruido en la ciudad de Guayaquil, utilizando una red de nodos sensores con tecnología LoRaWAN y ESP32.

1.5.2 Objetivos específicos

1) Desarrollar la interfaz principal para el monitoreo de interferencia y ruido en la ciudad de Guayaquil utilizando el entorno de programación de Arduino IDE.
2) Diseñar un entorno gráfico en el software LabVIEW para manejar las variables recibidas en la plataforma de Ubidots.

3) Evaluar la factibilidad y costo del diseño de una red de sensores inalámbricos con el módulo ESP32.

1.7 Hipótesis

La implementación de un sistema IoT con una red de nodos sensores con tecnología LoRaWAN con ESP32, permitirá el monitoreo constante de la contaminación y ruido en la ciudad de Guayaquil, brindando una información con mayor precisión, que servirá para tomar decisiones y ejecutar medidas públicas idóneas para reducir dichos problemas ambientales.

1.8 Metodología de Investigación

- Se realizará una búsqueda bibliográfica de la literatura existente sobre sistemas IoT para el monitoreo del entorno, nodos sensores, tecnologías LoRaWAN y ESP32.
- Se llevará a cabo el diseño de los nodos sensores y la red LoRaWAN, incluyendo la selección de componentes, la creación de esquemas de circuitos y el desarrollo de la interfaz para los ESP32.

Capítulo 2: Fundamentación Teórica

2.1 Concepto y definición del IoT

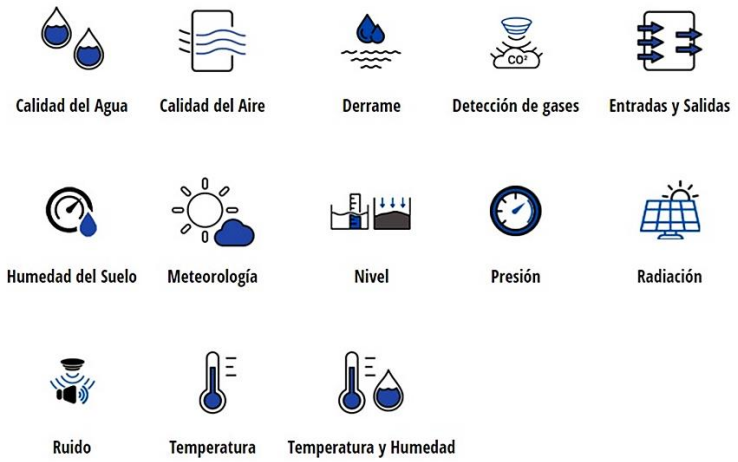
La definición de Internet de las Cosas o denominada con sus siglas IoT, se refiere a miles de millones de dispositivos interconectados a la nube repartidos por todo el mundo, así como a lo largo de las tecnologías que posibilitan su comunicación. Una cantidad variable de aparatos electrónicos desde artefactos de uso ordinario, como refrigeradores inteligentes, hasta complejos mecanismos industriales son capaces de recopilar, procesar y compartir avisos por medio de redes digitales. La creación detrás de la invención del IoT radica en el que los aparatos pueden comunicarse no solo entre sí, sino además con las personas como se observa en la figura 2.1, propulsando un nivel de interconexión no visto antiguamente en la vida rutinaria y los asentamientos industriales (Ferro & Rodríguez, 2021).

Figura 2.1: *Internet de las cosas presentes en distintos ámbitos.*

Nota: IoT aplicado en diferentes campos residencias e industriales para beneficio de la sociedad. Fuente: (Cooplalonja, 2022)

Cada uno de los miles de aparatos que constituyen el IoT tiene sensores así como programas, y componentes modernos les habilitan compartir enseres con demás aparatos y mecanismos por medio de la red. Los sensores actúan como recolectores de indicaciones del medio ambiente, como cambio de temperatura, medición de humedad, cantidad de aire que está en el espacio, o movimiento, en la figura 2.2 se observan los sensores inalámbricos más usados en el IoT. Los programas se encargan de procesar esta información, suele ser terminar en acciones automatizadas o codiciados. El IoT favorece tanto a acrecentar la eficiencia de trabajar, como a suscitar una cantidad de conocimientos significativa para elegir medidas de manera correcta y puntual (Toscano, 2024).

Figura 2.2: *Clases de sensores inalámbricos utilizados en IoT.*



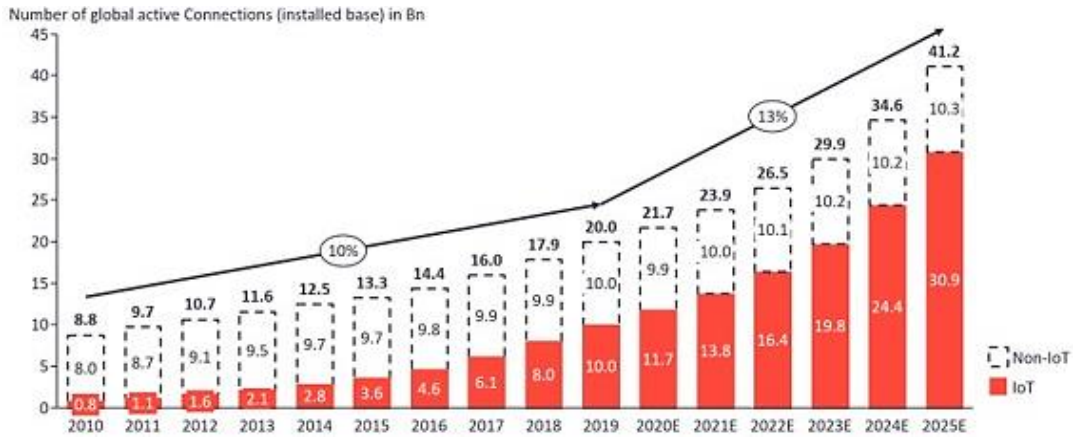
Nota: Distintos tipos de sensores no cableados para variedad de proyectos IoT. Fuente: (Flexitron, 2022)

2.1.1 Evolución de IoT

Los desarrollos tempranos del Internet de las Cosas IoT tienen sus raíces en el año de 1980. Al mismo tiempo, sucedió una parte del desarrollo activo de las posibilidades para conectar equipos electrónicos a través de

redes. Uno de los sucesos más significativos de estos tiempos es el lanzamiento de la red ARPANET. Se instaló como un proyecto pionero realizado por el Departamento de Defensa Estadounidense. En general, controlar procedimientos en Internet ha sido fundamental para el desarrollo del fenómeno en general, un hecho que también es cierto para el IoT. En efecto, ARPANET mostró cómo los sistemas de comunicación digital pueden vincular varios instrumentos situados en diferentes lugares en condiciones óptimas. De hecho, al mismo tiempo, los logros de la microelectrónica y la informática se produjeron en esta época. Gracias a la miniaturización de los sensores y la optimización de las tecnologías de comunicación, era posible que los dispositivos se incorporaran a redes más complejas con procesos masivos. Estas premisas mostraban cómo los objetos no solo se conectaban entre sí, sino que podían actuar de manera independiente en condiciones óptimas, sentaban las bases para el surgimiento del fenómeno moderno conocido como IoT, en la figura 2.3 se observa el crecimiento del IoT a lo largo de los años (Agüero et al., 2024).

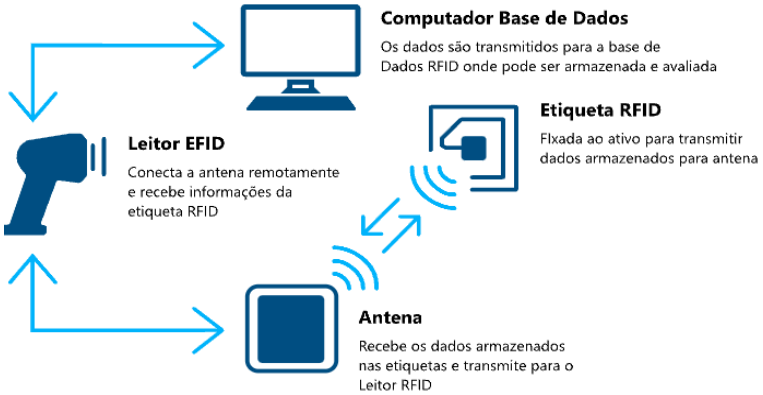
Figura 2.3: *Incremento del uso del IoT desde el 2010 hasta el 2025.*



Nota: Ritmo de incremento estadístico en el número de dispositivos conectados al internet. Fuente: (Martínez, 2022)

Uno de esos hitos fue la introducción de las Redes de Sensores inalámbricos durante la década de los 2000. Estos marcos de red permitieron la creación de sistemas que podrían monitorear el entorno y recopilar datos en tiempo real. Por lo tanto, su campo de aplicación se extendió a la agricultura de precisión, monitoreo medioambiental, y la gestión de infraestructuras. Al ser fundamentalmente inalámbricos, los sensores pudieron funcionar sin necesidad de conexiones físicas, lo que incrementó su flexibilidad y escalabilidad; ambos fueron aspectos críticos que contribuyeron significativamente al crecimiento del IoT en sus etapas tempranas. Un otro hito vital durante este período fue la adopción más generalizada de la tecnología de RFID como se observa en la figura 2.4. Descubrimos esta comprensión, principalmente para el seguimiento de objetos en el comercio minorista y la logística. Fue un desarrollo vital ya que ya no se permitió una mejor gestión de inventarios y activos. permitió rastrear artículos a lo largo de la cadena de suministro global. La capacidad de los dispositivos RFID para comunicar sin contacto físico resultó ser una jugada crítica en lo que concierne los avances y la trazabilidad de los procesos (Ulloa et al., 2021).

Figura 2.4: *Utilidad de la tecnología RFID.*



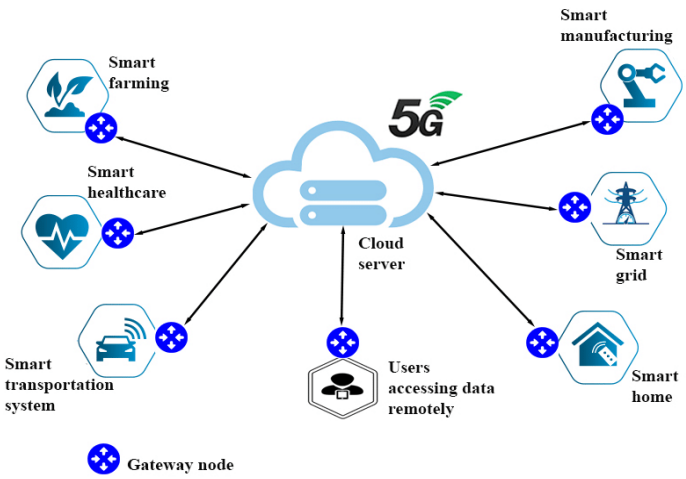
Nota: Proceso y uso de la tecnología RFID en la actualidad para distintos tipos de elementos. Fuente: (Vinicius, 2022)

En la última década, la computación en la nube y el procesamiento en la nube, como se indica anteriormente, y la creciente disponibilidad de Internet de alta velocidad debido a la popularización de la banda ancha han alterado profundamente el paisaje del Internet de las cosas. Estos desarrollos han permitido un procesamiento y almacenamiento de datos a gran escala que facilita la implementación de soluciones IoT a gran escala. En el presente, millones de dispositivos IoT producen y comparten datos de forma regular, y estos datos son disfrutados de inmediato por servidores remotos produciendo sistemas más inteligentes y efectivos. La computación en la nube ha sido particularmente central en este aspecto, ya que limpia la necesidad de infraestructura monumental y de cara y ofrece un acceso remoto y flexible a los datos. El desarrollo del IoT no se ha detenido, y las corrientes modernas indican una integración creciente con la inteligencia artificial y el aprendizaje automático. En conjunto, estas tecnologías permiten que los dispositivos IoT no solamente recopilen datos, sino que igualmente aprendan de ellos, generando así sistemas cada vez más autosuficientes en minutos y decides. Un caso de aplicación común de la IA en los vehículos inteligentes: un auto equipado con sensores y IA puede analizar los niveles de tráfico en tiempo movible y seleccionar la mejor trayectoria para su conductor mientras contribuye a la seguridad del tráfico (Chica & Leonardo, 2024).

Otra revolución pendiente que se abordará es la conectividad 5G. Para proporcionar capacidades de datos más grandes y tasas de transmisión mucho más necesarias, 5G significa menos retardo/latencia. Esto permitirá la creación de aplicaciones mucho más profundas y críticas para IoT, como el

control en línea de la maquinaria en una fábrica, la cirugía remota en el campo médico, el vehículo sin conductor y la operación aérea en el centro urbano. La conjunción del IoT y 5G proporcionará una variedad de oportunidades, desde la recalificación de las infraestructuras de las metrópolis que se llaman a sí mismas “metrópolis inteligentes” como se observa en la figura 2.5 de ejemplo, hasta la administración informada de los servicios públicos y los servicios públicos y la optimización en fábricas automatizadas la producción. Asimismo, las áreas inteligentes serán otro escenario del futuro de IoT. Útiles IoT en metrópolis conocida como smart-cities serán la solución de pauta para utilizar proyectos orientados a los residentes, desde la administración de tráfico, agua y electricidad hasta la mejora de la buena vida de los residentes. A su vez, conocido como Industria 4.0, la fábrica automatizada aprovechará la utilidad de IoT para ligar maquinaria y procesos y disfrutarla más eficiente, recortando los períodos de inactividad y aumentando la eficacia de producción. Conexos al paramento del bienestar, la industria sanitaria está llevando a cabo una transformación digital gracias al IoT (Martín, 2021).

Figura 2.5: *Modelo de red 5G aplicado al IoT.*

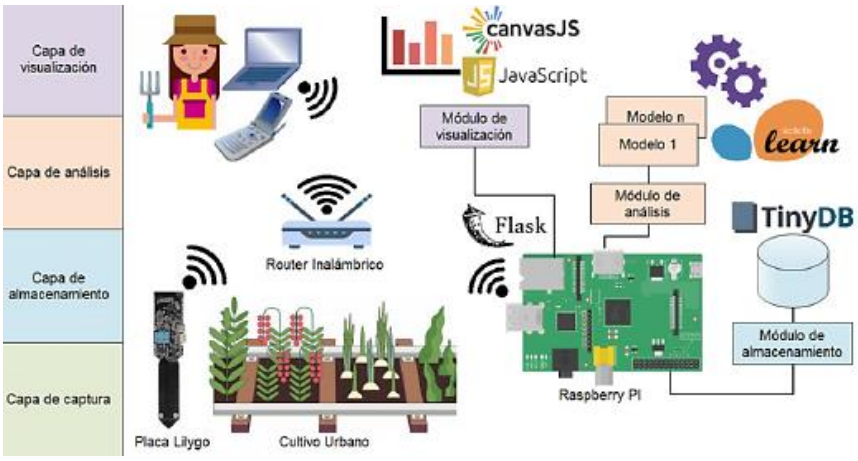


Nota: Distintos elementos interconectados al servicio de la nube mediante el uso de la tecnología 5G. Fuente: (Wazid et al., 2020)

2.1.2 Beneficios de IoT en la gestión urbana

El uso de tecnologías IoT en la gestión urbana está finalmente haciendo posible una transformación en la forma en que las ciudades utilizan y administran sus recursos. En base al uso del IoT, las ciudades pueden encontrar soluciones más eficientes y sostenibles para el desarrollo del entorno citadino. Por un lado, la capacidad de conectar los dispositivos y los sistemas directamente a través de redes integradas facilita a las autoridades locales la capacidad de monitorear y administrar el uso de agua, electricidad y otros recursos esenciales en tiempo real como se observa 2.6 por ejemplo. Por otro lado, también les permite corregir de manera más rápida y efectiva cualquier sistema defectuoso o en mal estado. Por ejemplo, gracias a la interconexión, las ciudades pueden identificar y reparar fugas, set backs en el sistema eléctrico y cualquier otro tipo de problemas que puedan surgir de la noche a la mañana. Definiendo según el avance de las tecnologías, en lugar de tener que esperar a que ceda algo (Bernardi, 2024).

Figura 2.6: *Beneficios del IoT para el monitoreo del clima.*



Nota: Elementos que conforman un red IoT para el monitorio de la climatología del entorno. Fuente: (Chanchí et al., 2022)

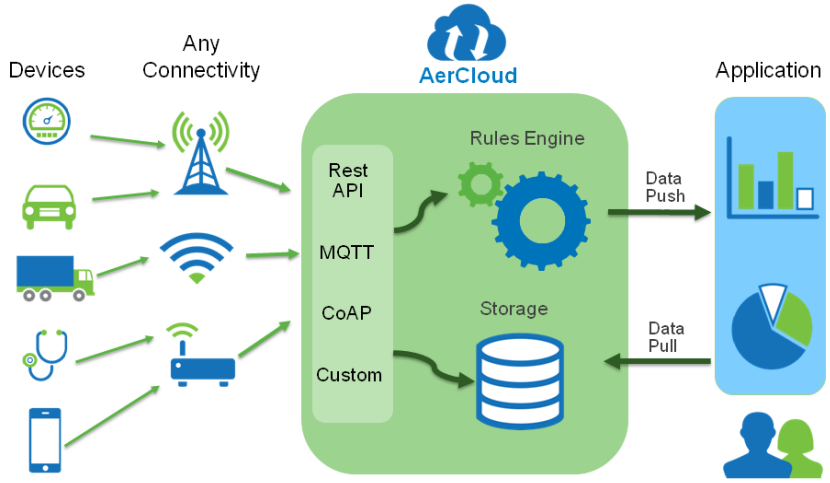
Además, el IoT también es vital para ayudar a mejorar la calidad de vida de los ciudadanos. Al reunir datos sobre sus respectivas operaciones diarias y recopilar y analizar datos en tiempo real, las ciudades logran responder y ofrecer servicios de manera más ágil y personalizada. También en la implementación dirigida a la seguridad pública, las ciudades poseen cámaras de seguridad de tipo inteligentes, a su vez también constan con sensores en los carriles y calles, donde se pueden monitorear la actividad delictiva, para que en el caso de que se detecte dicha actividad, enviar alertas automáticas a las autoridades, si algo ocurre algo. Por ende, no solo se mantiene a salvo a las personas, sino que las organizaciones pueden tomar medidas proactivas para lidiar con situaciones problemáticas similares de incidencia de enfermedades. Otra área benefactora son las medidas relacionadas con el tráfico. Los semáforos están conectados y se ajustan de acuerdo con el tráfico registrado en los sistemas de gestión de sus respectivas redes (Pérez, 2022).

2.1.3 Arquitectura de IoT

La arquitectura de IoT comienza con la capa de detección, una parte importante para la operatividad del sistema en general. Esta capa incluye una serie de sensores y dispositivos que recopilan información del medio ambiente y demás objetos físicos. Por ejemplo, sensores de temperatura, detectores de movimiento, módulos GPS, etc. como se observa en la figura 2.7. trabajan con esmero para capturar datos en tiempo real. Dado que estos sensores se han construido con una alta sensibilidad y belleza, los datos que recolectan se pueden confiar en su seguridad y exactitud. Los datos generados en esta

etapa actúan como la base para todos los demás enfoques de operación en IoT porque hace que la capa de detección sea esencial en este caso (León et al., 2024).

Figura 2.7: *Elementos que conforman de manera general el IoT.*



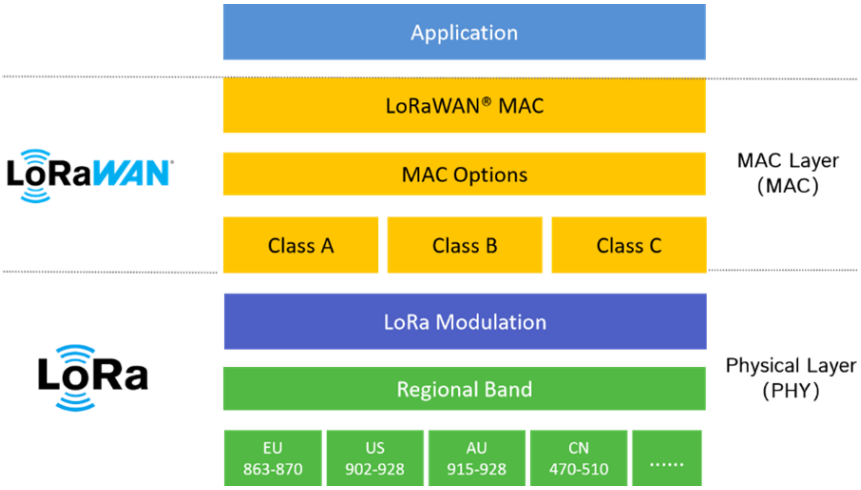
Nota: Protocolos de comunicación y las plataformas que forman parte del IoT. Fuente: (Jecrespom, 2024)

Dentro de la estructura de leo, la capa de red se establece. Esta capa lleva datos capturados por la capa de detección a otras partes del sistema. Las tecnologías de comunicaciones cubiertas por esta capa son Wi-Fi, Bluetooth y redes celulares, todas estas permiten una transmisión sin problemas de la información. Nos proporciona, por lo tanto ofrece un puente entre el mundo físico y el ámbito digital, lo que hace que la información disfrute de, una fluidez y seguridad de producción. La capa de red debe ser resistente y viable para afrontar la gran cantidad de datos generados por los dispositivos IoT, muchas veces incorporando protocolos de seguridad avanzados para proteger la integridad y confidencialidad de los datos en tránsito (Montaño, 2021).

2.2 Descripción general de la tecnología LoRaWAN

LoRaWAN, el cual significa Long Range Wide Area Network, es un protocolo WAN de bajo consumo pensado para permitir la comunicación inalámbrica entre los dispositivos en distancias considerable como se observa en la figura 2.8. Esta incluye aplicaciones en la Internet de las cosas cuando sus dispositivos, tales como sensores, no poseen una fuente de energía eléctrica directa pero tienen que comunicar los datos en una área mucho más grande. Utilizando LoRaWAN, se puede desarrollar una red de IoT que permita una eficiente transmisión de poco volumen de datos con bajo consumo de energía. En general, el propósito de LoRaWAN es permitir una comunicación de largo alcance y bajo consumo entre una incontable cantidad de dispositivos, siendo entonces compatible con múltiples aplicaciones de IoT (Espinosa et al., 2021).

Figura 2.8: *Funcionamiento de la tecnología LoRa y LoRaWAN.*



Nota: Explicación de las aplicaciones, modulaciones y bandas de conectividad en base a la región. Fuente: (Venco, 2022)

Otra de las características clave que define a la tecnología LoRaWAN es su capacidad de alcance en la comunicación, con distancias de hasta 10

millas en líneas de vista. La transmisión a distancia es posible gracias a las técnicas de modulación del espectro ensanchado, que permiten una señal resiente basada en un espectro más reducido. Asimismo, el tipo de dispositivos que operan en protocolos LoRaWAN suelen tener una larga vida útil. La tasa de las baterías puede tardar hasta 10 años, por lo que un sensor es ideal para un despliegue de sensor de largo alcance y desplazado en el tiempo. La comunicación bidireccional permite que los dispositivos manden información y se envíen de forma automática. Asimismo, LoRaWAN opera en bandas de radiofrecuencia sin licencia, es decir de 433 MHz y 900 MHz, lo que reduce los costos (Finistrosa, 2024).

En comparación con otras tecnologías inalámbricas como el estándar de conectividad de red inalámbrica local, Wifi y las comunicaciones engranadas por Bluetooth, LoRaWAN presenta una serie de ventajas que lo convierten en la tecnología más adecuada a utilizar. En particular, LoRaWAN supera a sus contrapartes derrotando en la distancia cubierta por señales por Wifi y Bluetooth. Otra ventaja es la sensibilidad del receptor que LoRaWAN maneja, con la capacidad de recibir señales más débiles desde lejos. Además, LoRaWAN presenta una mejor gestión de la seguridad, incluido de 128 bits mientras que las redes Wifi utilizan comúnmente de 64 bits de larga clave provista por la clave de autenticación. Por lo tanto, la tecnología mencionada anteriormente es altamente efectiva y segura para garantizar la comunicación (Castellanos, 2020).

2.2.1 Arquitectura de LoRaWAN

Los otros componentes de la arquitectura incluyen nodos finales, puertas de enlace y servidores de red. Los nodos finales o nodos sensores son los propios sensores y actuadores y son los que realmente recopilan datos y los envían. Las puertas de enlace, a veces referidas como el centro o el puente, operan entre los nodos y los servidores y reciben datos de los nodos finales y los envían y reciben mediante LoRaWAN a los propios sensores y otros dispositivos. Por último, los servidores de red procesan los datos de las puertas de enlace y los envían a los servidores de aplicación a los que se conectan de manera centralizada. Los servidores de aplicación en sí administran cómo los usuarios finales ven el proceso. Esto en estrella asegura que haya comunicación y la red es gestionada eficazmente (Toapanta, 2024).

LoRaWAN tiene protocolos y estándares en su arquitectura que permiten el proceso y la cobertura de datos con eficacia. Es un protocolo LPWAN que opera en la capa física de la modulación Lora. Esto permite conectividad y cuanto más baja es la tasa de bits y la poca potencia que se necesita para conectarse. LoRaWAN simétrica estos protocolos para que los dispositivos puedan hacerlo. LoRaWAN opera en capas, cada uno con una función en la cadena de comunicación. La capa física es el propio Lora; a continuación, LoRaWAN es la capa de red del OSI como se observa en la figura 2.9 y opera de niveles 2 y 3. La arquitectura en estrella permite la interacción entre los nodos finales y las puertas (Peñaloza & Yupanqui, 2022).

Figura 2.9: *Modelo de las capas de interconexión del OSI.*

Nota: Descripción general de los 7 niveles que conforman el modelo OSI en la interconexión. Fuente: (Stackscale, 2023)

2.2.2 Bandas de frecuencia y velocidades de datos

Estas asignaciones de la frecuencia de LoRaWAN varían según la región, de modo que se ajusta a las regulaciones locales para el funcionamiento eficiente y sin interferencia. La frecuencia operativa para el LoRaWAN, por ejemplo, en Europa, es de alrededor 868 MHz. América asigna 915 MHz mientras que Asia usa una variedad de 433 MHz completa. Estas asignaciones permiten al LoRaWAN ofrecer una cobertura robusta y confiable según las necesidades específicas de cada región. Las frecuencias de 433 a 928 MHz son un rango ideal para la tecnología Lora ya que logran equilibrar la cobertura y la penetración de interiores. La banda esenciales para áreas urbanas compactadas y regiones rurales de escasa infraestructura de comunicación (Ortiz, 2024).

LoRaWAN tiene opciones de velocidad de datos utilizadas que se denominan factores de esparcimiento, que permiten enviar información conforme a las circunstancias de la aplicación. Existen diversos velocidades

de datos desde un mínimo de 300 bps a un máximo de 37.5 kbps. La tecnología de modulación Lora le permite ser fijada ya que el espectro ensanchado capacita para la comunicación de largo alcance a pesar de poseer un ancho de banda pequeño. esto es muy importante en la Internet de las cosas. Corresponde tener en cuenta un ejercicio clave entre el alcance y la capacidad de la velocidad de datos. A mayor velocidad de datos, menos alcance y viceversa. 450 bps es ideal para corto alcance, 50 kbps es apropiado, mientras que modulación FSK es óptima para corto alcance (Chasiluisa, 2020).

Figura 2.10: *Frecuencias utilizadas en cada continente.*

Nota: Distintos tipos de frecuencias empleadas por la tecnología Lora dependiendo la ubicación geográfica. Fuente: (Lora, 2022)

2.2.3 Seguridad en LoRaWAN

Los métodos de cifrado utilizados por LoRaWAN también son de alta calidad para asegurar la comunicación dentro de la red. Un método ampliamente reconocido utilizado es el Estándar de Cifrado Avanzado – Advanced Encryption Standard, un algoritmo de cifra de clave simétrica

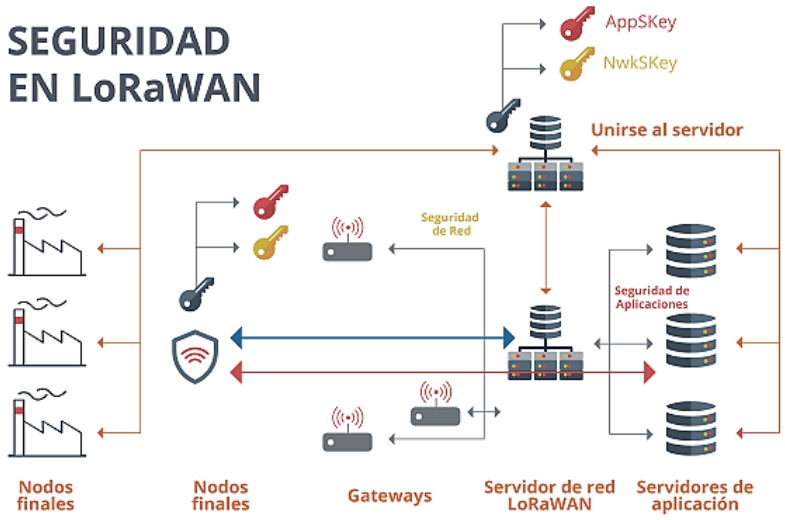
fácilmente implementado y confiable. El protocolo de cifrado emplea varios modos de funcionamiento de AES como el modo Contador y el Código de Autenticación de Mensajes basado en cifrado), para cifrado e integridad, respectivamente. Por estas técnicas de cifrado, los mensajes entre los dispositivos finales y el servidor de red son confidenciales y protegidos. Dada la variedad de estos métodos de cifrado, LoRaWAN proporciona un fuerte entorno de protección para la transmisión de datos en redes de baja energía de área amplia (Giraldo, 2024).

LoRaWAN logra la autenticación garantizando la integridad y la autenticidad del mensaje. De esta manera, la red solo permitirá que los dispositivos de confianza tengan acceso a ella. Para hacerlo, el protocolo define un Código de integridad del mensaje que genera de la AppKey. La AppKey se comparte entre la red y el dispositivo final de destino. Cuando un dispositivo intenta unirse a la red, el servidor de red verifica el Mensaje Integrity Code para garantizar que el mensaje recibido sea auténtico. En otras palabras, la red controla si la fuente de la solicitud es confiable. Luego, la red devuelve al nuevo dispositivo una respuesta de unión para que este se comunique de manera autenticada. Este proceso asegura no solo la autenticación, sino también un canal autorizado entre los dispositivos y la red como se observa en la figura 2.11 (Sanabria & Bravo, 2020).

Además, en el marco de seguridad de LoRaWAN, tanto la integridad como la privacidad de datos son factores críticos. Para proteger sus datos cuando lo transmite, LoRaWAN utiliza la clave de sesión de red y la clave de

sesión de aplicación. Estas claves se utilizan para cifrar la carga útil y proteger la comunicación entre el dispositivo final y el servidor de red; mientras NwkSKey se asegura de que los datos no se alteren durante su transmisión; el otro AppSKey se asegura de que los datos a nivel de aplicación sean confidenciales. Por lo tanto, con estas claves de sesión implementadas, todos los datos deben ser, a priori y sin excepción, privados e inalterados, lo que mantiene a salvo la información transmitida (Bonilla, 2023).

Figura 2.11: *Frecuencias utilizadas en cada continente.*



Nota: Distintos tipos de frecuencias empleadas por la tecnología Lora dependiendo la ubicación geográfica. Fuente: (Incibe, 2023)

2.2.4 Aplicaciones de LoRaWAN

Cabe agregar que las ciudades inteligentes y el Internet de las Cosas dependen en gran medida de la tecnología LoRaWAN porque facilita las conexiones de largo alcance y las redes IoT en áreas con malas conexiones. LoRaWAN facilita la recolección y transmisión de datos desde varios sensores y dispositivos en la ciudad, como los medidores de consumo energético: las estaciones meteorológicas. El bajo consumo y la capacidad de conexión a

nivel de red se traducen directamente en ciudades más conectadas e inteligentes, lo que ahorra tiempo a sus residentes y disminuye su huella ambiental. Además, implementar una red de sensores de calidad del aire utilizando LoRaWAN es fundamental durante la ejecución del proyecto de la cadena de bloques de una ciudad inteligente (Santander, 2024).

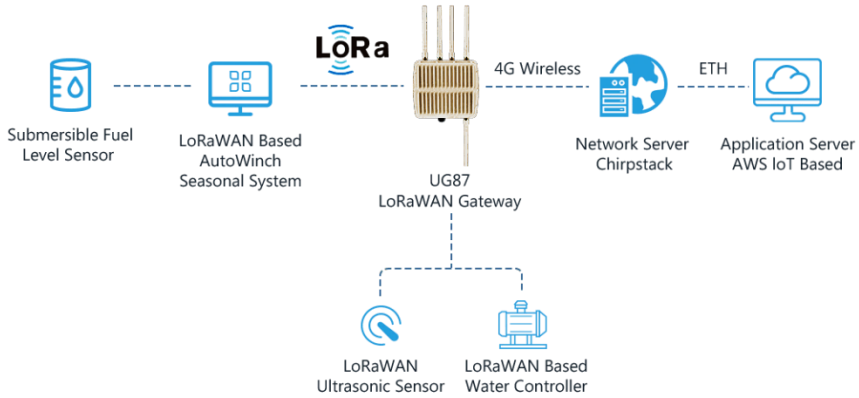
En la industria de la salud y el seguimiento de los activos relevantes, LoRaWAN hace posible rastrear los bienes relevantes, ya que es posible la tecnología de monitoreo de seguridad y condiciones de salud de los pacientes. Las instalaciones de atención médica usan dispositivos de rastreo GPS de LoRaWAN el cual mantiene la ubicación de los bienes relevantes, rastreando la ubicación y monitoreando el monitoreo de la temperatura y el movimiento. LoRaWAN mantiene la alineación de sus empleados, los seres como un botón de pánico que presenta las alertas en la ubicación de la vida. Esta herramienta se combina con otras herramientas en la industria de la salud para mejorar la alineación y la administración del monitoreo de la propagación de la exhibición de grandes posiciones, lo que aumenta la eficacia operativa y la respuesta a situaciones desesperadas (Zepeda, 2023).

2.2.5 Ventajas de LoRaWAN

Una característica de la tecnología que lo convierte en una de las oportunidades es la capacidad de comunicación de largo alcance. Los dispositivos pueden comunicarse hasta distancias de 15 km en el lado rural y 5 km en el urbano. Esta cobertura insuficiente lo convierte en la solución perfecta para implementaciones en agricultura como se observa en la figura

2.13, administración de infraestructura remota y monitoreo ambiental donde las redes tradicionales no son lo suficientemente eficientes. La mayor ventaja del alcance es que los nodos más aislados pueden establecer una comunicación sin ocasionar fallas, aumentando la eficiencia de todo el proceso y el alcance del IoT implementado (S. A. Sánchez, 2022).

Figura 2.13: *IoT implementado en la agricultura.*



Nota: Gestión de riego automatizado a través del control inalámbrico mediante el uso del IoT. Fuente: (Monolitic, 2020)

La segunda característica de la tecnología LoRaWAN es el consumo de energía. Esta tecnología mejora la vida útil de las baterías conectadas a los dispositivos. Con una sola carga de la batería, los dispositivos LoRaWAN se ejecutan durante muchos años. Por lo tanto, esta red es una solución viable para las baterías que admiten dispositivos inteligentes. Tales indicadores bajos de gasto energético resultan especialmente útiles en caso de que la implementación incluya la ubicación remota o la accesibilidad limitada, como sería el caso con el seguimiento del hábitat de la vida silvestre o las mediciones inteligentes, donde la batería requiere ser reemplazada regularmente lo cual sería extremadamente costoso y poco práctico. En

conjunto, la eficacia energética de LoRaWAN disminuye la demanda de esfuerzos renovados, también reduciendo los desechos electrónicos y contribuyendo al desarrollo sustentable del proyecto de IoT (Hernández & Marcos, 2024).

La escalabilidad y flexibilidad de LoRaWAN son cualidades críticas y son las que posibilitan que esta tecnología sea la elección adecuada en muchas aplicaciones de IoT. LoRaWAN se escala y adapta perfectamente a los cambios en la dinámica de una organización e intento de cumplir con una amplia gama de dispositivos y necesidades de transmisión de datos. Las redes de LoRaWAN pueden interactuar en diferentes anchos de banda, niveles de potencia, distancia y seguridad. De esta manera, garantiza un rendimiento óptimo en todas las condiciones porque diferentes dispositivos tienen requisitos normales. La vertical de escalabilidad de LoRaWAN significa que cuando crece la cantidad de redes, la cantidad de dispositivos se adapta para cumplir con la dinámica de una empresa (Jaramillo, 2021).

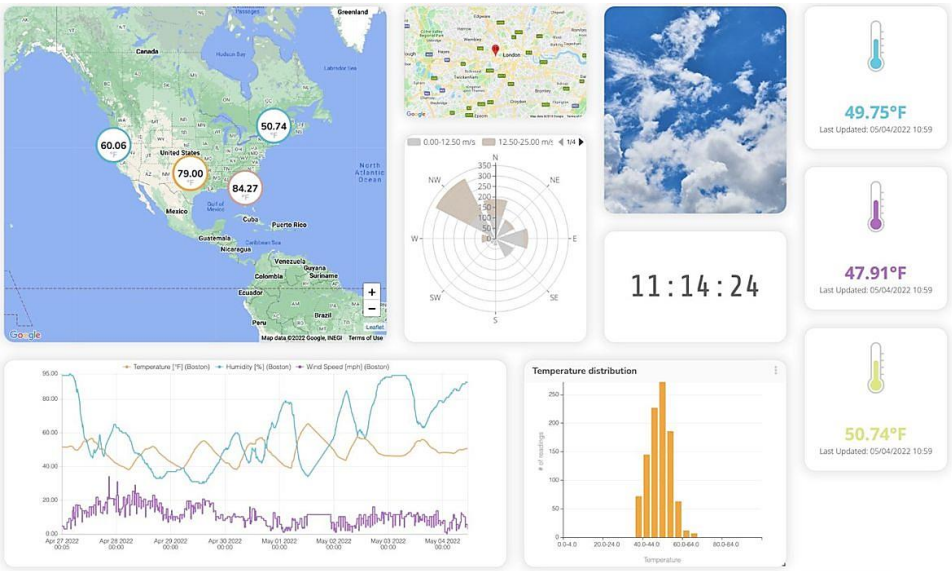
2.3 Descripción general de Ubidots

Ubidots es una plataforma de Internet de las Cosas que permite a las empresas y los usuarios finales compilar, analizar y observar las mediciones de sus dispositivos. Su función principal es transitar la integración de mediciones provenientes de varios sensores y artefactos de IoT en un entorno único en el que se puedan postprocesar y comprimir varias fuentes para muchas. La integración y el preanálisis son esenciales para múltiples

verticales de la industria que desean mejorar sus operaciones, desempeñar y adherirse decisiones a datos fidedignos al 100% (Iqbal & Manzoor, 2020).

Asimismo, Ubidots introduce apasionados y startuppers a diseñar nuevas mercancías de plataformas a partir de artefactos con conexión a Internet. Ubidots tiene sus orígenes en el arranque, donde identificar una necesidad con un sumario claro para posibilitar la administración desconexión de IoT más sencilla para las masas. Desde entonces, Ubidots ha permitido una funcionalidad robusta, que logró ser usada por corporaciones de distintos campos. Algunos de los ejemplos son monitoreo ambiental como se observa en la figura 2.14, producción inteligente, SCADA y más (Aghenta, 2020).

Figura 2.14: *Ubidots aplicado en el monitoreo de las condiciones ambientales del entorno.*



Nota: Sistema de rastreados de tipo flotantes para medir las condiciones ambientales de la zona. Fuente: (Tangarife, 2023)

Ubidots tiene muchas características y beneficios clave, la compilación y visualización de datos en tiempo real son críticas porque a los usuarios se les debe permitir ver y analizar datos al instante. Esta función les da la oportunidad de actuar de la misma manera. Los sistemas de alerta y notificación en tiempo real son críticos porque garantizan el monitoreo constante y la respuesta instantánea a los eventos. Además, la facilidad de uso es una de las características clave que atraen al máximo número de usuarios. Se asegura de que incluso las personas no muy expertas utilicen la aplicación. Al mismo tiempo, la capacidad de trabajar con muchos dispositivos y sensores es aún más crucial. Todas estas características hacen de Ubidots una aplicación poderosa y viable para cualquier proyecto (Casas et al., 2021).

2.3.1 Configuración de Ubidots

Crear una cuenta en Ubidots es el primer paso necesario para exprimir todo el potencial de esta poderosa plataforma de internet de las cosas. Ubidots es un software en el que los hombres pueden hacer programas de IoT rápido, gracias a una interfaz fácil e intuitiva. Además, Ubidots proporciona un servicio gratuito para probar diversas capacidades, sin un plan ya algoritmo claramente estructurado desde el principio. Para crear una cuenta, primero ve al sitio web de Ubidots Oficial y selecciona la opción de registro. Se pedirá que se ingresen varios datos básicos como: nombre, dirección de correo electrónico y contraseña segura. Una vez hecho esto, se tendrá acceso a la plataforma y podrá probar las capacidades que se ofrecen (Tolocka, 2024).

2.3.2 Recopilación y gestión de datos

Ubidots admite varios tipos de datos, lo que brinda a la plataforma la flexibilidad necesaria para aplicaciones de IoT de uso múltiple. La plataforma está diseñada para procesar y albergar datos de diferentes orígenes que incluyen sensores, dispositivos y bases de datos ajenas. Sin embargo, más que eso, Ubidots puede trabajar con datos numéricos, datos categóricos y otros tipos de datos más complejos como objetos JSON. Al alojar varios tipos de datos, Ubidots garantiza que los usuarios puedan fusionar e investigar datos de múltiples fuentes para aumentar sus capacidades analíticas y de monitoreo. Además, tener múltiples capacidades de datos aumenta la productividad del desarrollo de aplicaciones de IoT en todas las etapas. Es especialmente crítico para las industrias de la fabricación inteligente y el SCADA en la nube porque los usuarios pueden beneficiarse de la ejecución de la plataforma (Vargas et al., 2023).

En Ubidots, los datos se recopilan mediante métodos variados que están diseñados para satisfacer las necesidades de diferentes escenarios. Se utilizan RESTful API para datos, y se necesita para una transmisión sin errores de datos desde dispositivos a la plataforma Ubidots. Del mismo modo, Ubidots admite MQTT, que es un protocolo de mensajería de IoT “más adecuado para lo que llamamos vales únicos de datos”, específicamente, los datos recopilados a través de dispositivos con almacenamiento de ancho de banda fuera de línea o con restricciones. Estos métodos variados ofrecen a los usuarios la flexibilidad de elegir con qué medios desean recopilar datos, y esta capacidad integrada permite a la biblioteca recopilar datos con los

márgenes de error mínimos esperados. Además, Ubidots es compatible con la prevalencia de la tercera parte servicios y plataformas. Por lo tanto, los usuarios pueden agregar lecturas de varias fuentes a un solo sistema (Patricio, 2020).

2.3.3 Visualización de datos

Panel de control sección de Ubidots es una solución intuitiva que permite a los usuarios tomar decisiones sobre los datos recibidos de diferentes fuentes. Ubidots canaliza un camino sencillo para que los entusiastas de la tecnología y los emprendedores obtengan todos los productos y soluciones a medida conectados a Internet que les encantaría tener. Los paneles de control ofrecen un resumen visual de todo lo que está pasando, un centro de información en el que se puede ver una variedad de variables al mismo tiempo para decidir. Ubidots incluye alertas y notificaciones en tiempo real, la plataforma asegura que los usuarios estén informados sobre cualquier anomalía o cambio oportunamente. La personalización de widgets como se observa en la figura 2.15, también es un punto esencial en Ubidots, los elementos no solo se invierten alrededor de su eje central, también se pueden dimensionar y colorar según se desee. Esto no solo ofrece una experiencia más agradable multiplicando los resultados, pero un valor agregado que transforma la información (Aliaga & Serquén, 2022).

Los widgets Ubidots son elementos en un tablero que se configuran para presentar datos físicos en un formato visualizable, fácil y entendible. Los widgets pueden ser gráficos, tablas, medidores, indicadores y más,

dependiendo de sus necesidades y preferencias. Monitoreo en tiempo real es una de las características más destacadas que Ubidots puede ofrecer. Un sistema a tiempo ayuda al usuario a seguirlo en directo mientras le brinda una respuesta en el instante que se lleva a cabo un cambio la razón por la cual este tipo de soluciones son aplicables en supervisión de condiciones, fabricación inteligente y SCADA en la nube. Ubidots también admite alertas y notificaciones en tiempo real (Cuji et al., 2023).

Figura 2.15: *Tipos de widgets disponibles en Ubidots.*

Nota: Variedad de elementos para visualizar las variables en la plataforma de Ubidots para su análisis. Fuente: (Microside, 2020)

2.3.4 Medidas de seguridad y privacidad

El cifrado de datos es un aspecto central de la seguridad de los datos en la operación de plataformas IoT, como Ubidots. Una de las técnicas más importantes, es el cifrado de transmisión, Ubidots utiliza SSL/TLS, asegurando así que la información sensible no pueda ser accedida durante la

transmisión de dispositivos en la nube. Además, no se permite originalmente que los datos sean interceptados por algún tercero en la transmisión a leerse. La robusta implementación del cifrado tiene un rol más allá de garantizar la confidencialidad de la información, sino también asegura la integridad evitando que sea modificada por un tercero. En Ubidots, se pueden implementar diferentes niveles de acceso para la autenticación. Los administradores tienen la autoridad para configurar los mismos en cuanto a permisos para cada usuario específico. Se requiere la habilidad de auditar las acciones de los usuarios, que aumenta la seguridad mediante la detección de las actividades sospechosas (Flores & Guadalupe, 2024).

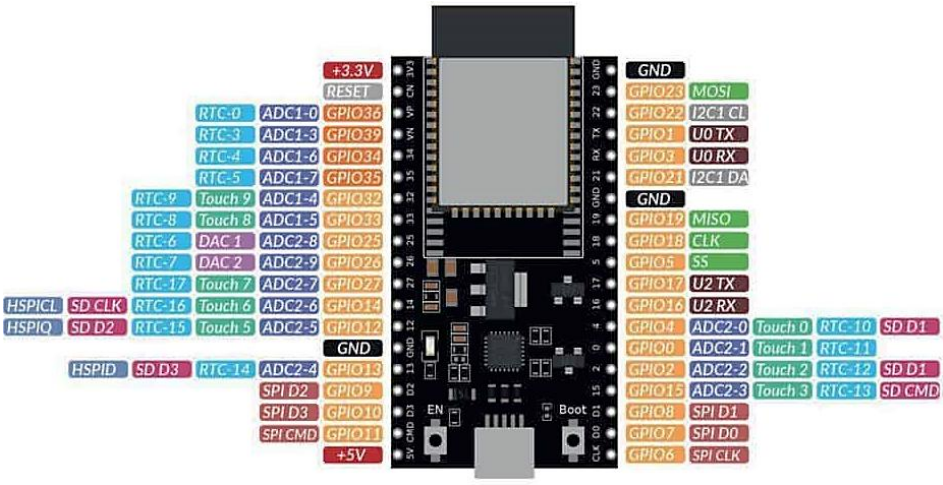
2.4 Descripciones técnicas del ESP32

Esta placa se destaca por su capacidad de procesamiento, debido a sus dos núcleos de CPU de hasta 240 MHz que permiten la administración de múltiples tareas en paralelo, lo cual la hace ideal para aplicaciones especializadas, complejas y con necesidades de grandes cantidades de datos en tiempo real; sus establecimientos de 32 bits que comparativamente con diseños de microcontroladores previos, la hace una opción mucho más eficiente. Gracias a sus 520 KB de RAM y el cache de hasta 4MB, este microcontrolador está diseñado para ejecutar aplicaciones muy grandes y de almacenar datos sin poner en peligro la eficiencia. También a la posibilidad de actualizaciones de firmware base en campo, esencial para dispositivos IoT (Eras et al., 2024).

2.4.1 Protocolos de comunicación de ESP32

El microcontrolador ESP32 como se observa en la figura 2.16 tiene una función Wi-Fi muy robusta, lo que lo hace muy funcionales para múltiples aplicaciones. El módulo tiene una velocidad de datos de transmisión de hasta 150 Mbps, lo que facilita una transmisión rápida y segura de datos. Para permitir un rango físico más amplio, el ESP32 también tiene una potencia de salida de 20 dBm aproximadamente en la antena. Esto lo hace ideal para su uso en un rango más amplio. Dichas características son útiles para dispositivos de IoT que necesitan una conexión Wi-Fi más fuerte y duradera. La CPU también puede ser usada a 240 MHz y tiene una CPU doble integrada (Belmonte, 2022).

Figura 2.16: *Pines de la placa ESP32.*



Nota: Pines analógicos y digitales que conforman el microcontrolador ESP32. Fuente: (Morales, 2022)

Aparte del impresionante rendimiento de la Wi-Fi, el ESP32 también es excelente con la comunicación Bluetooth, proporcionando Bluetooth classic y su versión de baja energía Bluetooth Low Energy. Al brindar soporte para ambas versiones, el ESP32 se puede usar en una variedad de aplicaciones,

incluso para transferencia de datos de alta velocidad y corto alcance y bajo costo en términos de energía. Al agregar la compatibilidad con Bluetooth al ESP32, la placa lo hace ideal para fabricar productos de IoT con una conexión inalámbrica constante. La conectividad BLE del chip tiene un papel adicional, ya que, al tener el chip conectado a un bajo consumo de energía, podría ser ideal para dispositivos como rastreadores de actividad física o dispositivos inteligentes en nuestro hogar (J. Sánchez, 2023).

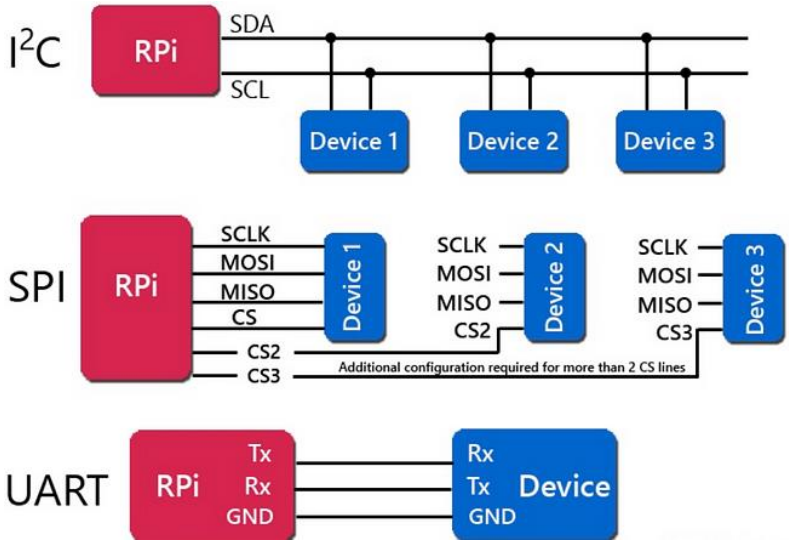
Su otra característica atractiva es la capacidad de admisión de comunicación de modo dual en varias funciones, es decir, Wi-Fi y Bluetooth de manera simultánea. El modo dual en un solo dispositivo permite a la ESP32 funcionar como un elemento central de ambos tipos de redes y mejorar la flexibilidad y adaptabilidad de los diversos sistemas de IoT. Por un lado, basado en la conexión Wi-Fi, el dispositivo puede conectarse a Internet alojando el servidor, y, por otro, puede encontrar dispositivos cercanos y sensores mediante la conexión Bluetooth. Así, dado que es un solo sistema integrado con esquema de modo dual y características flexibles, permite satisfacer los requisitos y mejorar el intercambio de datos y control a través de distintos tipos de protocolos de red (Andrade et al., 2022).

2.4.2 Integración de ESP32 en sistemas IoT

El ESP32, por otro lado, se destaca por ser compatible con varias plataformas IoT, mientras que el dispositivo se puede integrar con otras plataformas IoT con relativa facilidad. Su CPU de doble núcleo con un máximo de reloj de 240MHz significa que tiene un rendimiento potente y admite varios

protocolos de comunicación, incluidos Wi-Fi, Bluetooth estándar y BLE, que son esenciales para los sistemas de IoT. Además, el módulo puede interactuar con varios sensores y otros periféricos dado que tiene varios pines GPIO además de las interfaces periféricas integradas, como I2C y SPI como se observa en la figura 2.17. Esta compatibilidad adicional facilita a los desarrolladores la conexión del ESP32 a varias plataformas IoT, por lo que es versátil e ideal para varios proyectos de IoT (Tovar, 2024).

Figura 2.17: *Diferentes tipos de protocolos de comunicación en ESP32.*



Nota: Tres tipos de protocolos de comunicación más utilizados en los dispositivos electrónicos para comunicarse con el microcontrolador ESP32. Fuente: (Rufus, 2023)

Si hablamos de seguridad, es otro aspecto crucial al que prestar atención al elegir componentes para los sistemas IoT. El ESP32 también posee un avanzado set de funciones de este tipo. En primer lugar, el ESP32 admite un cifrado basado en hardware. Puedes tener la certeza de que tus datos son seguros y confiables. Incluso puede estar seguro de que su banco no es hackeado. Esta función garantiza que tu sistema no sea hackeado por

ciberdelincuentes u otras amenazas. Además, el dispositivo soporta Secure Boot y Flash Encryption. Esto significa que solo el firmware provisto por un fabricante autenticado se ejecutará en su dispositivo. Eso le garantiza datos seguros y confiables a través de cualquier aplicación de IoT que esté manejando.(Parra, 2023).

2.4.3 Consumo y eficiencia de energía de ESP32

Varias técnicas de recopilación de energía también ayudarían a abordar aún más la eficiencia energética de los sistemas basados en ESP32. Uno puede integrar paneles solares para garantizar una fuente ilimitada de energía en aplicaciones de campo, lo que reduce eficazmente la dependencia total de la energía de la batería. El uso de dispositivos piezoeléctricos que generan electricidad bajo estrés mecánico o vibraciones es otra técnica efectiva. Esta abordar sería relevante para las áreas de alto desplazamiento o las condiciones en las que hay movimientos continuos, como las instalaciones industriales. Los generadores termoeléctricos que convierten las diferencias de temperatura en electricidad también harían que una aplicación basada en ESP32 sea más favorable. También normaliza su fuente de energía en condiciones inestables. Incorporar típicas técnicas de recopilación de energía en sistemas que aumentarían su sostenibilidad, lo que seguiría impactando el medio ambiente (Romero, 2023).

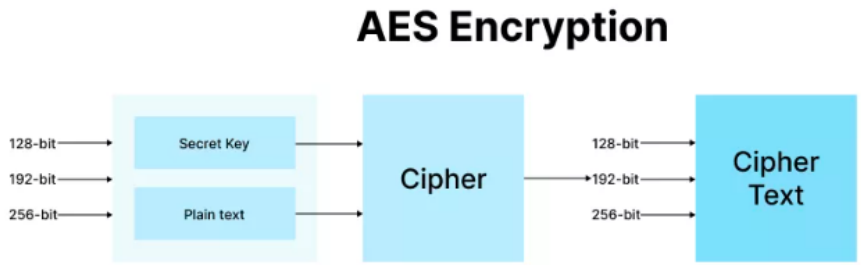
Optimizar la vida útil de la batería de los dispositivos basados en ESP32 requiere energía eficiente y una gestión cuidadosa del software. Por lo tanto, los desarrolladores deberían hacer un uso extensivo de los diferentes

modos de suspensión proporcionados por el microcontrolador, lo que aseguraría que el dispositivo pase la corriente activa en su estado de baja potencia. Además, es beneficioso implementar ciertos ciclos de trabajo, es decir, alternar entre el estado de trabajo y de suspensión para utilizar la energía solo cuando sea necesario para el funcionamiento. Una programación adecuada para reducir la cantidad de proyección no facilitada y aprovechar las interrupciones de hardware para “despertar” el dispositivo solo cuando sea necesario asegurar que el dispositivo funcione durante un largo período con una sola carga. Al usar todas estas técnicas juntas, los dispositivos ESP32 alimentados por batería pueden usar aplicaciones a largo plazo en ubicaciones lejanas o de difícil acceso (Chunga & Niacata, 2023).

2.4.4 Características de seguridad de ESP32

Debido a los fuertes estándares de cifrado, el ESP32 microcontrolador garantiza la seguridad de los datos en varios otros casos. Entre otros, el ESP32 admite las prácticas de cifrado de la industria, como AES el cual se observa en la figura 2.18, RSA y SHA. El verdadero cifrado es necesario para proteger la integridad y confidencialidad de los datos. En otras palabras, esos métodos ayudan a garantizar los datos confidenciales en el envío y almacenamiento de datos, por lo que el ESP32 es un método compatible para la comunicación segura. Además, la disponibilidad de la aceleración de hardware de tareas criptográficas garantiza un mejor rendimiento, ya que las operaciones son más rápidas y menos caras en volumen. Por lo tanto, el hecho de que el microcontrolador se centre en la seguridad y el cifrado demuestra la práctica del ciclo de vida (Niño & Rimarachin, 2023).

Figura 2.18: *Protocolo de encriptación de tipo AES.*



Nota: Cantidad de bits que abarca y dispone la encriptación de tipo AES.
Fuente: (Mehta, 2023)

Otra característica de seguridad importante del ESP32 es su proceso de arranque seguro. El ESP evita que el código no autorizado se ejecute en su dispositivo utilizando esta función crítica. Un proceso de arranque seguro garantiza que solo se permita ejecutar el firmware que está firmado criptográficamente por una autoridad confiable. Esta es una de las medidas de seguridad más críticas para proteger sus dispositivos de actualizaciones de firmware maliciosas que pueden ser un peligro para su sistema. La verificación de la autenticidad y la integridad del firmware antes de la ejecución puede considerarse una excelente forma de defensa. Y el proceso de arranque seguro es esa capa defensora inicial (Jiménez & Chávez, 2023).

Capítulo 3: Aportes de la investigación

En el presente capítulo se detalla todos los componentes técnicos implicados en la operación del sistema, desde la interfaz de usuario diseñada en Arduino IDE, el entorno gráfico en LabVIEW y la implementación de los nodos sensores en la plataforma IoT Ubidots hasta el adecuado desarrollo de diagramas de conexión en el caso de la red de los nodos sensores que permiten una red estructurada de mínimo acumulamiento y pérdida de datos. Asimismo, se tiene en cuenta un estudio de costos e ingresos. En cuanto a el aspecto técnico económico, la propuesta resulta ser compatible con la capacidad de proporcionar datos exactos y en tiempo real sobre los niveles de contaminación y ruido.

3.1 Análisis de las partes que integran el sistema IoT propuesto

En este subcapítulo se proporciona una descripción general de los componentes involucrados en el sistema. Cada componente mencionado a continuación tiene una función específica en el dispositivo desde el registro de los datos hasta la transmisión inalámbrica de los mismos. Por lo tanto, a continuación, se muestra una descripción general de algunas de las características y funciones asociadas con los dispositivos, tales como el microcontrolador ESP32, el módulo LoRa, los sensores de CO2 y ruido, y la pantalla OLED, los cuales son fundamentales para el monitoreo.

ESP32 TTGO LoRa32 OLED V2.1.6

Funcionamiento: El corazón del proyecto, este microcontrolador como se observa en la figura 3.1 es responsable de todas las funciones inalámbricas

a través de LoRa utilizadas y controla la pantalla OLED en la que se representarán los datos. También es responsable de la recolección de datos de los sensores de CO2 y ruido y Los envía mediante LoRa después de ser procesados en él.

Descripción: En una placa ESP32, posee un microprocesador con conectividad Wi-Fi y bluetooth, un transceptor Lora y una LCD OLED 128×64 píxeles (SSD1306). El ESP32 fue diseñado para permitir que la placa actúe como un nodo IoT versátil y autosuficiente. Lo ideal sería utilizar un ESP32 como un sistema de monitoreo con comunicación de larga distancia. El ESP32 también tiene múltiples E/S analógicas y digitales y soporta numerosos protocolos de comunicación, tales como: UART, I2C y SPI.

Figura 3.1: *Módulo ESP32 de controlador TTGO LoRa32 con pantalla OLED V2.1.6.*

Nota: ESP32 con tecnología Lora para mayor alcance en una red de nodos.
Fuente: (Lilygo, 2024)

Módulo LoRa (SX1276)

Funcionamiento: Comunicación inalámbrica de largo alcance y bajo consumo; tecnología de modulación Lora. Este módulo se emplea en este

sistema para enviar datos de sensores a un receptor a larga distancia; esto permite la monitorización remota sin la necesidad de infraestructura Wi-Fi o celular.

Descripción: El módulo transceptor SX1276 de Semtech está diseñado para la banda ISM, que es generalmente 915 MHz en esta configuración. Brinda transmisión fiable a distancias que pueden exceder los 10 km en condiciones óptimas (Guerrero, 2022).

Sensor de CO_2 MH-Z19B

Funcionamiento: Medición de concentración y detección de dióxido de carbono presente en el aire hasta en partes por millón (ppm), ideal para cuantificar la aptitud de calefacción o refrigeración en interiores o exteriores.

Descripción: Con una capacidad específica para CO2, mide las lecturas de CO2 en un rango de 0 a 5000 ppm con un margen de error bajo de ±50 ppm + 5% de la lectura (Borodinecs et al., 2022).

Figura 3.2: *Sensor de dióxido de carbono modelo MH-Z19B.*

Nota: Sensor encargado de monitorear el grado de contaminación del entorno. Fuente: (Emariete, 2020)

Sensor de Ruido MAX4466

Funcionamiento: Nivel de ruido ambiental en el área circundante y convierte el nivel de presión sonora en una señal eléctrica. Los datos recopilados en

este dispositivo pueden servir como base de datos para inferir la intensidad del sonido proveniente de la toma, lo que podría interpretarse como una forma de contaminación acústica

Descripción: El MAX4466 es un micrófono amplificado, y la sensibilidad de sonido ambiental seleccionada es suficiente para obtener una precisión adecuada para el experimento. A su vez, se adjunta un micrófono de condensador electrónico proyectado para medir cualquier cambio en la presión sonora, produciendo un cambio correspondiente de voltaje que el amplificador integrado en el módulo amplifica (Picazo, 2021).

Figura 3.3: *Sensor de ruido modelo MAX4466.*

Nota: El sensor incorpora un micrófono el cual le permite detectar los niveles de ruido presentes en el ambiente. Fuente: (Destec, 2022)

3.2 Descripción del funcionamiento de los elementos del sistema propuesto de IoT

En este subcapítulo, son presentados y analizados los diagramas de flujo que describen las etapas principales de trabajo del sistema de monitoreo ambiental diseñado en el marco del proyecto. Los diagramas presentan un desglose del flujo lógico y secuencias operativas que rodean el trabajo de los sensores integrados, los procesos de comunicación, procesamiento de datos y su visualización. Los diagramas de flujo examinan el proceso de

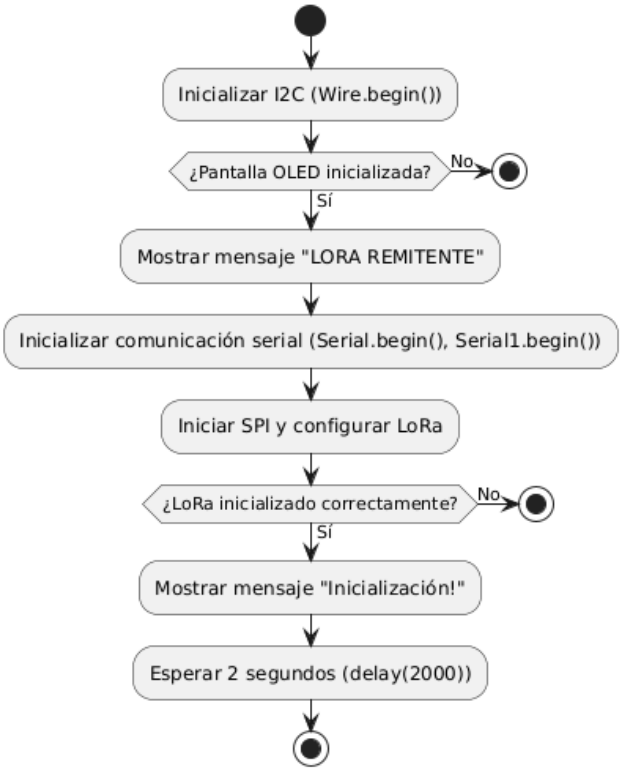
inicialización de cada componente, incluyendo el módulo LoRa SX1276, la pantalla OLED, los sensores de CO2 y ruido, el ciclo de recolección del dato y su procesamiento, el flujo de transmisión del dato al receptor LoRa, así como el esquema de manejo de error condicional y de control que garantiza la operación estable y continua del sistema. Cada diagrama proporciona una descripción detallada de las interacciones internas entre unidades funcionales del dispositivo y documenta los procesos individuales y su integración en el flujo del programa principal. Por lo tanto, el capítulo presenta una representación exacta de la lógica implementada y establece la base para su análisis técnico en el contexto del proyecto.

3.2.1 Diagrama de flujo del setup del nodo sensor emisor Lora

En diagrama de flujo de la figura 3.4 y anexos se describe el proceso de inicialización del sistema. Se configuró la comunicación I2C para que el dispositivo interactúe con la pantalla OLED. Se comprueba si la inicialización de la pantalla se ejecuta correctamente. En caso afirmativo, se imprime la etiqueta LORA REMITENTE en el dispositivo de visualización, lo que significa que el sistema está encendido y listo. Luego, se configura la comunicación serial para la comunicación y transmisión de datos de depuración y los puertos serie estándar del hardware ESP32 mediante Serial.begin() y Serial1.begin(). Procedemos entonces a configurar el módulo Lora en el sistema. A través de la interfaz SPI, inicializamos los pines y configuramos la tasa de transmisión de datos en el módulo a 915 mm para cerrar esta transmisión. Si las etapas anteriores se ejecutan sin problemas, se activa Inicialización en el dispositivo OLED, lo que significa que la comunicación LoRa está habilitada y lista para

enviar y recibir mensajes. Finalmente, se presenta un retraso de 2 segundos (delay(2000)) para garantizar que todos los sistemas estén completos antes de que el dispositivo comience a funcionar.

Figura 3.4: *Diagrama de flujo de ejecución del setup.*



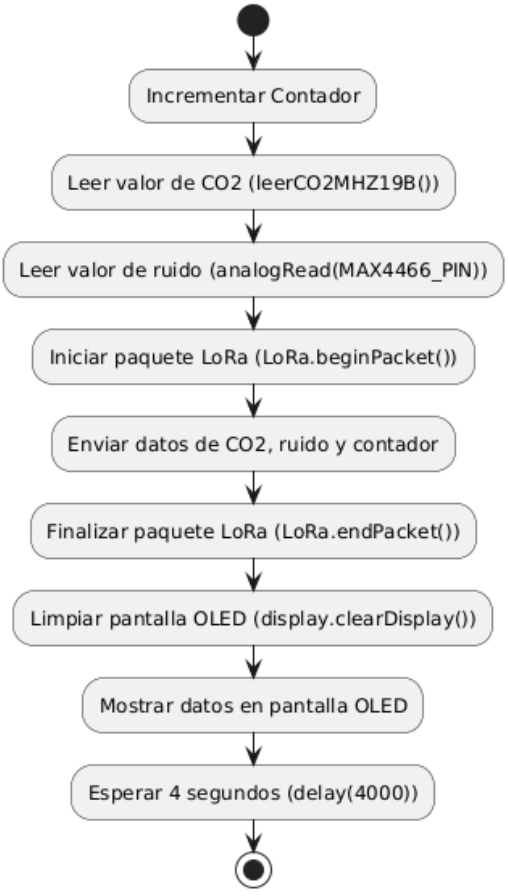
Elaborado por: El Autor

3.2.2 Diagrama de flujo del loop del nodo sensor emisor Lora

En el diagrama de la figura 3.5, se describe el ciclo operativo continuo del dispositivo. Cada vez que el ciclo llega al final del flujo, el contador del dispositivo se incrementa en 1, el contador es solo por los efectos de rastreo para permitir que el creador del dispositivo sepa el número de ciclos o transmisiones que el dispositivo ha realizado. En el flujo, se leen los valores de los sensores de CO 2 y ruido. Se lee el valor de CO 2 mediante la función

MH-Z19B sensor-object read, y el ruido se lee utilizando el análisis series sensor del sensor MAX4466: analogRead. Después de tener ambos valores o mediciones, el sistema inicia un paquete LoRa con el arreglo LoRa: beginPacket y envía los datos de CO 2, ruido, contador, escribiendo la info con LoRa: print.m MathLibrary, luego se cierra el paquete LoRa con LoRa-endPacket, se limpia la pantalla del consumo medio de oxígeno digital con función display.clearDisplay y se muestra la info de los datos de entrada para que el creador m pueda ver y ponga los datos en tiempo real. Después de la emisión, el trabajo que se hace se permite una pausa antes de 4 segundos (delay(4000)) y se hace o se repite.

Figura 3.5: *Diagrama de flujo de ejecución del loop.*

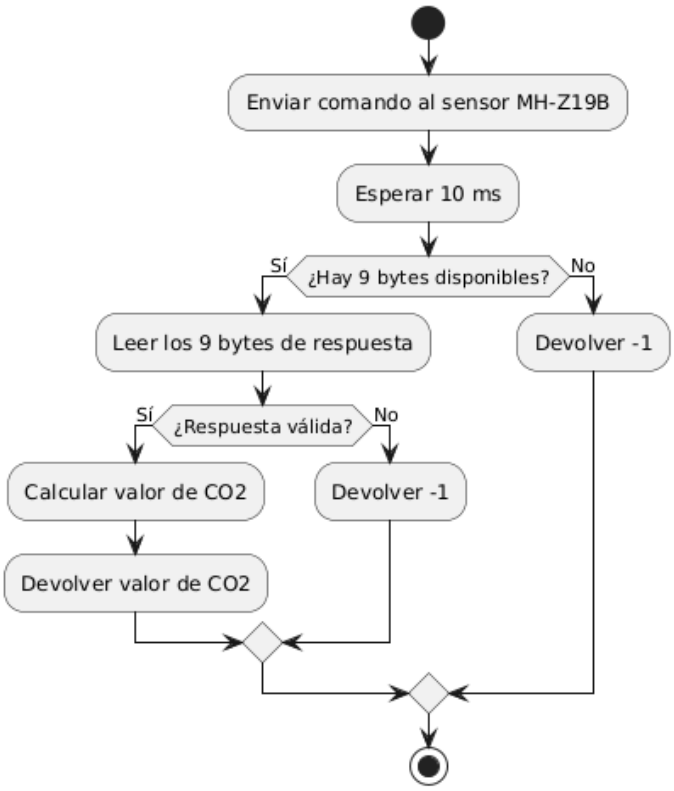


Elaborado por: El Autor

3.2.3 Diagrama de flujo de lectura del sensor de CO2 (MH-Z19B) del nodo sensor emisor Lora

El diagrama de flujo de la figura 3.6, como su nombre lo indica, ilustra el proceso de lectura de los datos del sensor de CO 2, MH-Z19B. La función se inicia con un comando enviado al sensor, solicitando una medición de CO 2. Se espera que el sensor procese y devuelva la respuesta durante 10 ms. Luego, el sistema comprueba si 9 bytes de datos han sido recibidos. En caso afirmativo, la respuesta ha sido completad y es válida, de lo contrario, la función devuelve -1. Si 9 bytes se reciben, el sistema valida que la información es correcta, y el valor de ppm de CO 2 es calculado y retornado, de lo contrario, la función vuelve a devolver -1.

Figura 3.6: *Diagrama de flujo de ejecución del sensor de CO2 (MH-Z19B).*

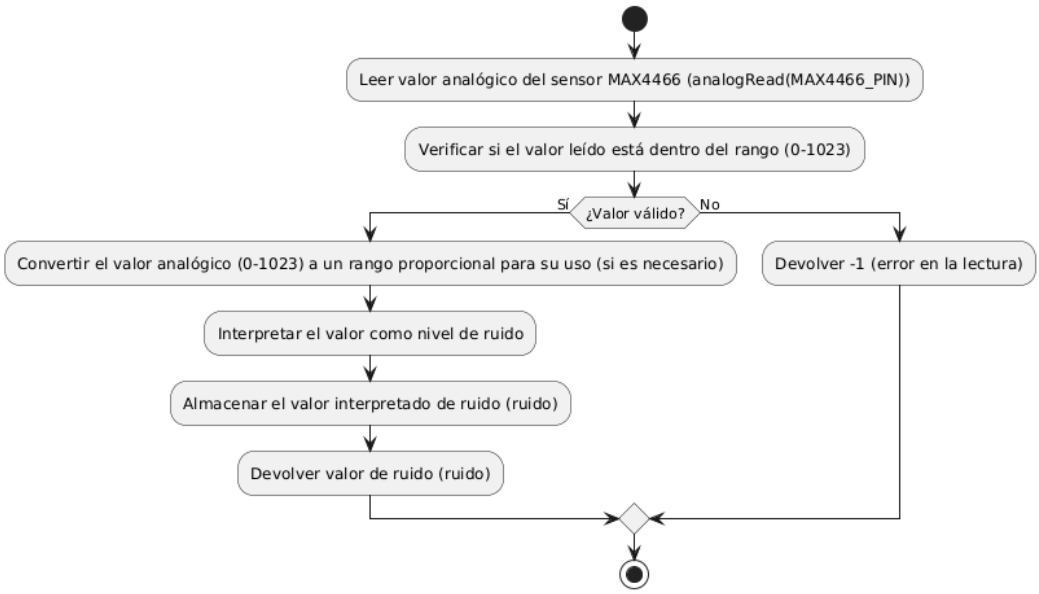


Elaborado por: El Autor

3.2.4 Diagrama de flujo de lectura del sensor de ruido MAX4466 del nodo sensor emisor Lora

En el diagrama de flujo de la figura 3.7 se observa el proceso seguido por el sistema para leer el sensor de ruido MAX4466, el cual es un micrófono amplificado. En primer lugar, el sistema lee el valor analógico de ruido que devuelva el sensor a través del pin asignado del ESP32; esto se hace con la función analogRead() y esta devuelve un número que representa el valor analógico de intensidad de sonido que se capta con el micrófono. Este valor leído se regresa como el resultado de la función para usarse posteriormente en el sistema, ya sea para incluirse como dato en los que serán enviados por el módulo LoRa o en el que será impreso en la pantalla OLED.

Figura 3.7: *Diagrama de flujo de ejecución del sensor de ruido MAX4466.*



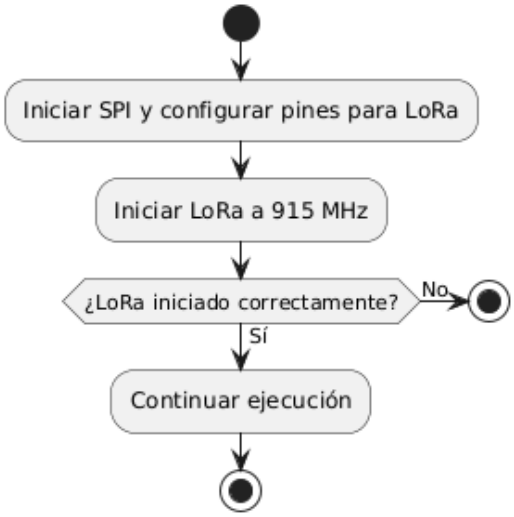
Elaborado por: El Autor

3.2.5 Diagrama de flujo de la inicialización del nodo sensor emisor Lora

En el diagrama de flujo de la figura 3.8 se observa el “main” archivo de arranque que inicializa el LoRa módulo. En primer lugar, inicializa el SPI bus

y asigna los pines requeridos para la comunicación con el LoRa módulo. Entonces intenta inicializar el LoRa módulo con el 915MHz frecuencia. La banda ISM 915 MHz es una de las bandas ISM más comunes para aplicaciones de comunicación a larga distancia. Si la inicialización es exitosa, el sistema sigue su ejecución a lo largo del programa y es capaz de enviar y recibir paquetes entre sí. Sin embargo, si el LoRa inicialización falla, el sistema apaga y se detiene porque la red LoRa es necesaria para la comunicación entre dispositivos.

Figura 3.8: *Diagrama de flujo de ejecución de inicialización del emisor.*



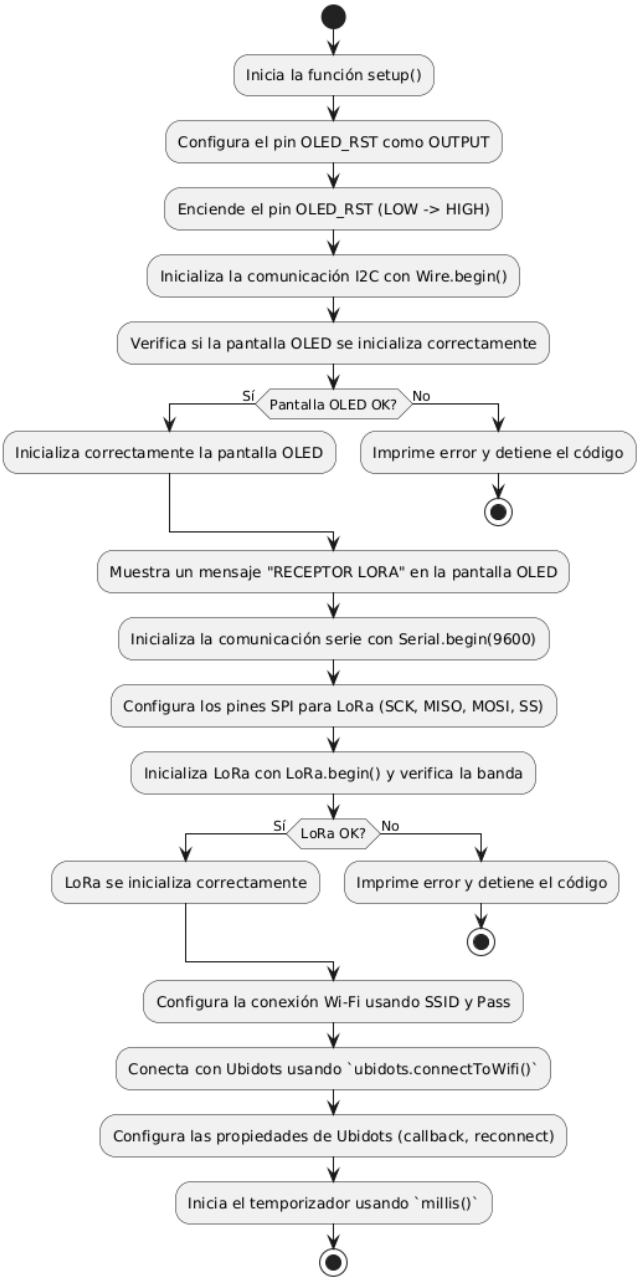
Elaborado por: El Autor

3.2.6 Diagrama de flujo del setup del nodo receptor Lora

En el diagrama de flujo de la figura 3.9 se observa la función de configuración inicial, se ejecutará solo una vez, lo que se usó al principio de la secuencia del programa. Comenzará con la progresión de los componentes del sistema. El primero es que OLED_RST, que es el pin de reinicio de la pantalla OLED, Se configurará y se creará como salida OUTPUT, luego se

pondrá activo para empezar la pantalla. Después, debe crearse la comunicación I2C con wire.begin(). Cabe destacar que LoRa solo puede hacerlo; Por lo tanto, la conexión a la pantalla OLED es esencial para controlar LoRa. Después, El sistema mostrará un mensaje de error en monitor serie y se desconecta si la pantalla no se crea.

Figura 3.9: *Diagrama de flujo de ejecución del setup del receptor.*



Elaborado por: El Autor

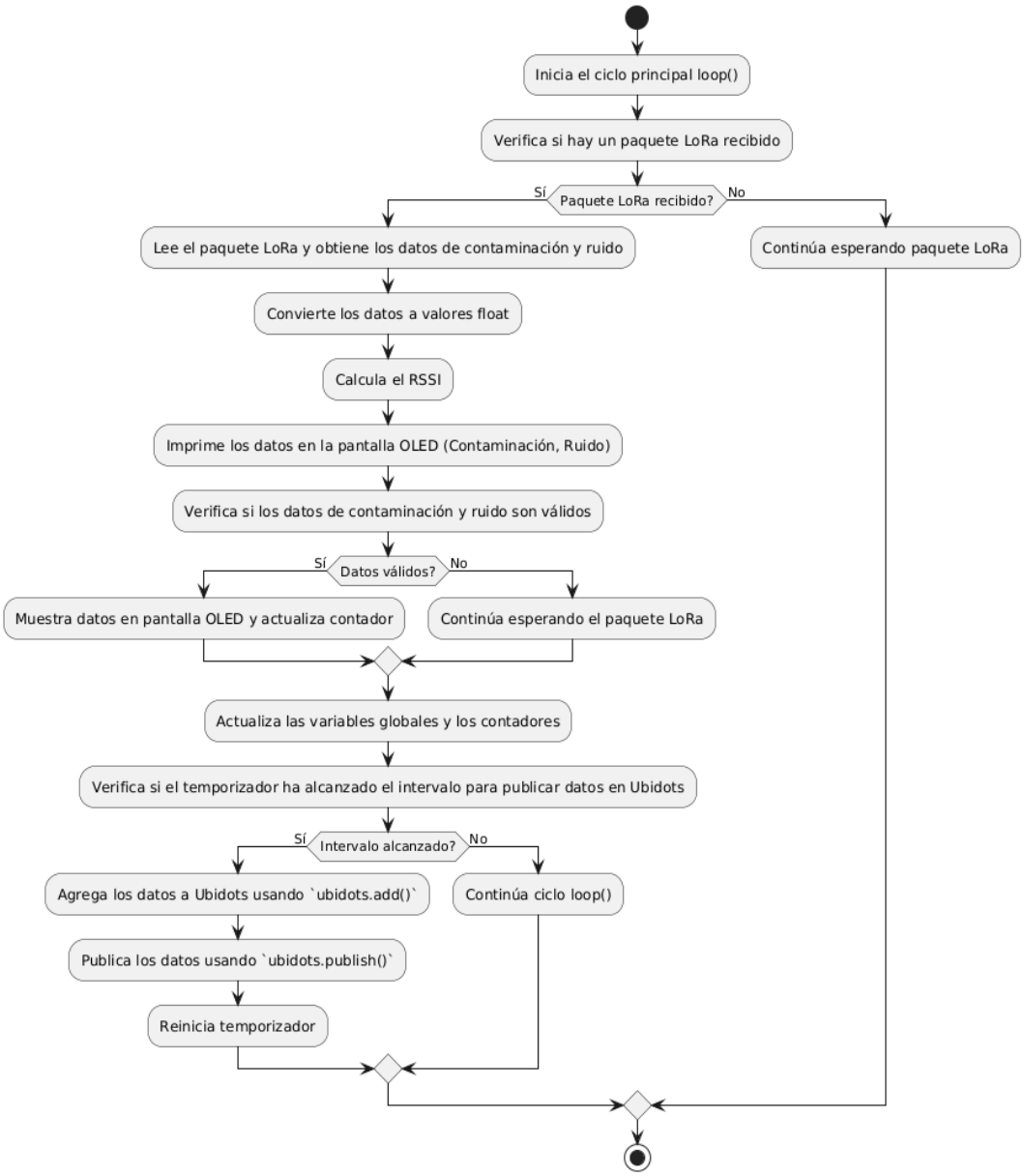
El sistema equivaldrá que la pantalla se arrancó y comenzará a escribir “RECEPTOR LORA” en la pantalla OLED. Luego, se configurará la comunicación serial con serial.begin(9600), La razón de la comunicación serié es para -para facilitar la resolución de problemas y para proporcionar una interfaz para los datos. Se comienza con un módulo LoRa y Se configura el módulo SPI LoRa los pines requeridos que se deben usar para que el Monitor llegue al microcontrolador, incluidos: SCK, MOSI, MISO y SS. Además de la comunicación serial, si la inicialización de LoRa no fue exitosa, entonces se mostrará y se saldrá del Monitor; el módulo LoRa está listo para funcionar.

3.2.7 Diagrama de flujo del loop del nodo receptor Lora

En el diagrama de flujo de la figura 3.10 se observa que la función loop se ejecuta para siempre cuando se enciende el dispositivo. Primero, comprueba si ha habido un paquete Lora utilizando la función de verificación de datos entrantes con serializado available(). Si no hay un paquete, el programa esperará hasta que haya un nuevo paquete. Si un paquete Lora se recibe, el sistema lee el paquete los datos y convierte cada uno en valor flotante. Luego se calcula el RSSI o la potencia de señal recibida. Todas estas lecturas se imprimen en la pantalla OLED para que el usuario las vea en tiempo real. Luego, el sistema verifica si los datos recibidos son válidos mirando los datos para asegurarse de que no son 0. raíces. Si los datos son válidos, los muestra en la pantalla y agrega 1 a los contadores utilizados para más cálculos. Si los datos son incorrectos, descarta el paquete Lora y procede a esperar el siguiente paquete. Sin embargo, si hay un paquete válido, el sistema espera hasta que se completen otros pasos antes de los intervalos

de tiempo. El sistema verifica si el temporizador que permite la publicación en ubidots ha pasado. Si la publicación es posible se realizan función add y función publish y luego reinicie el temporizador. A continuación, el programa espera a que pase el temporizador. Si no se pueden publicar datos, el programa esperará y se ejecutará como de costumbre.

Figura 3.10: *Diagrama de flujo de ejecución del loop del receptor.*

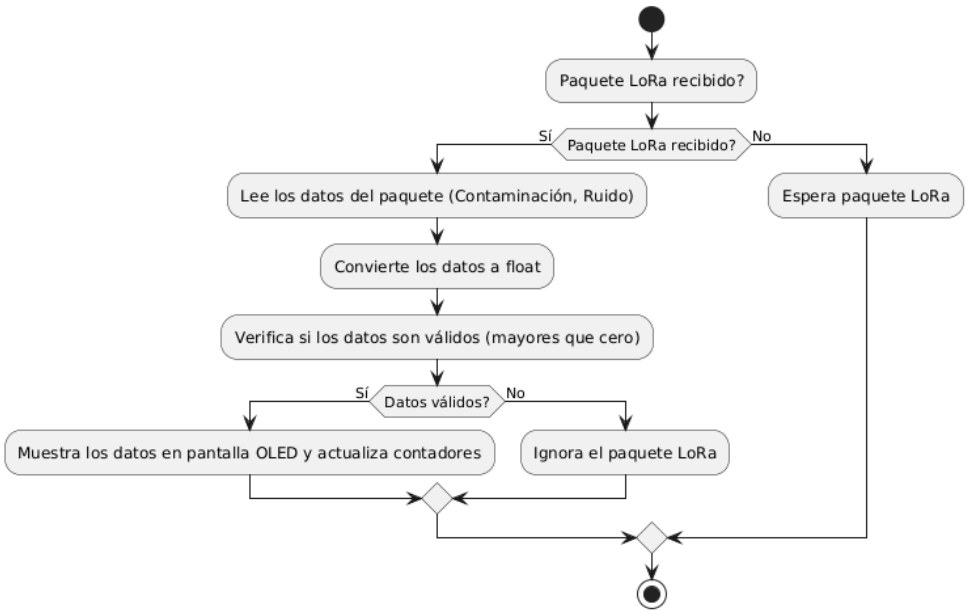


Elaborado por: El Autor

3.2.8 Diagrama de flujo del procesamiento de los datos LoRa del nodo receptor Lora

En el diagrama de flujo de la figura 3.11 se centra en qué hacer con los datos que se reciben a través de LoRa y cómo se manejan. Una vez que el sistema recibe un paquete LoRa, en primer lugar, el sistema lee los datos del paquete. Estos datos se refieren a la contaminación y la información del sensor de ruido. Luego, los datos se convierten en formato flotante, se pueden leer a continuación, analizar y procesar. Después de eso, el sistema verifica si los datos recibidos son válidos o, de hecho, ¿son más grandes que 0? 0. En otras palabras, esta etapa elimina los datos no sensatos, es decir, no se manejan datos erróneos. Si los datos son válidos, al sistema enviarlos a la pantalla OLED, mostrarlos y actualizar los contadores internos que rastrean el progreso del sistema.

Figura 3.11: *Diagrama de flujo del procesamiento de los datos LoRa del receptor.*

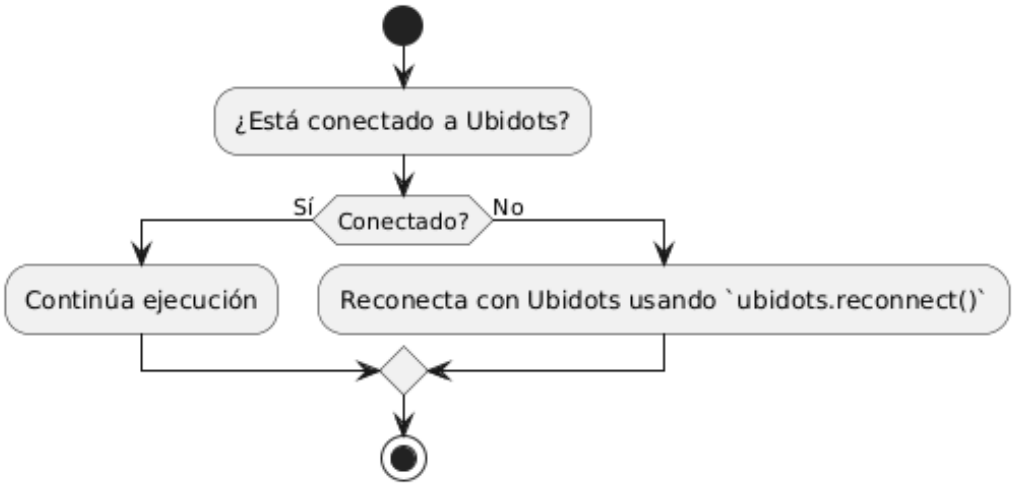


Elaborado por: El Autor

3.2.9 Diagrama de flujo detallado de reconexión con Ubidots del nodo receptor Lora

En el diagrama de flujo de la figura 3.12 se ilustra cómo se ocupan las conexiones a la plataforma de recopilación y visualización de datos en Ubidots. En la Figura 3, se verifica si el dispositivo está conectado a Ubidots. Si el dispositivo está conectado, el programa continúa con su ejecución normal: procesa los datos y realiza publicaciones tal como se detalla anteriormente en esta sección. En caso de que no esté conectado, lo que puede deberse a que el Wi-Fi es defectuoso o la duración de la sesión ha expirado, el sistema intenta reconectarse tan pronto como sea posible. Así, la función ubidots.reconnect () se llama. Como resultado, el sistema puede mantener la conectividad por sí misma sin intervención humana. Esta funcionalidad es especialmente valiosa en escenarios de Internet de las Cosas, donde la conectividad e la red a menudo es intermitente.

Figura 3.12: *Diagrama de flujo de reconexión con Ubidots del receptor.*

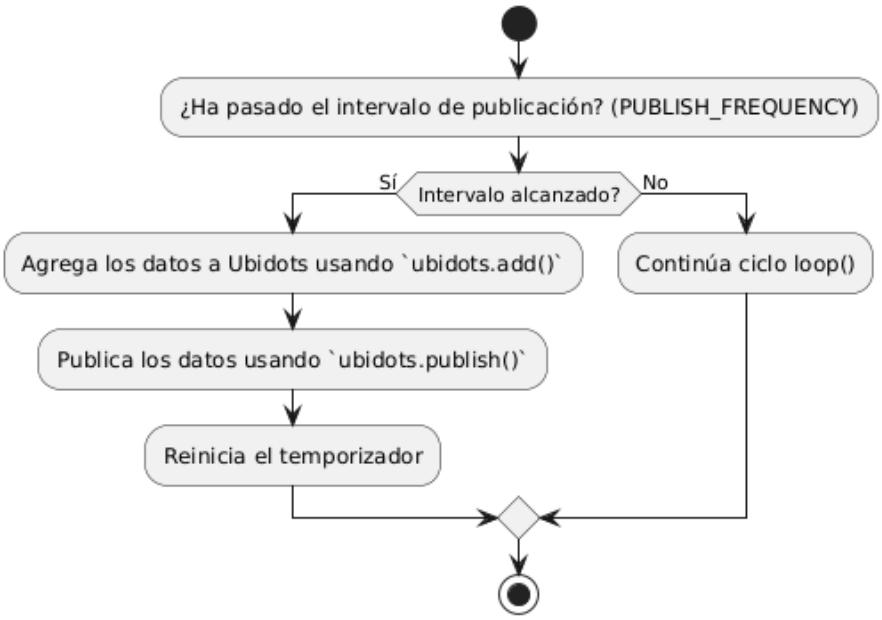


Elaborado por: El Autor

3.2.10 Diagrama de flujo de la publicación de datos a Ubidots del nodo receptor Lora

En el diagrama de flujo de la figura 3.13 se describe cómo y cuándo los datos se publican en Ubidots. La lógica seguirá esperando hasta que el temporizador se cumpla, siguiendo la constante PUBLISH_FREQUENCY. Cuando se cumpla, el temporizador se cerrará y los datos de contaminación y ruido se agregarán a Ubidots. Para determinarlo, ubidots_add se ocupará de los datos y luego se publicarán con el uso de ubidots.publish. Finalmente, cuando los datos se envíen, el temporizador volverá a encenderse para volver a enviar los datos. Por otro lado, si todavía no está cubierto el horario, la lógica se cerrará e intentará nuevamente en el ciclo loop. Esto asegurará que los datos se envíen regularmente sin sobrecargar la red con envíos constantes a la vez.

Figura 3.13: *Diagrama de flujo de la publicación de datos a Ubidots del nodo receptor.*



Elaborado por: El Autor

3.3 Interconexiones entre los dispositivos y los elementos del sistema IoT.

En la tabla 3.1 a continuación, se resume la conexión entre los distintos componentes del sistema basado en ESP32, detallando los pines utilizados, el tipo de pin, y el voltaje de alimentación de cada uno. Dichos componentes son el módulo LoRa para la comunicación inalámbrica, la pantalla OLED 128×64 para la visualización de datos y los sensores de CO 2 y sensor de ruido para la medición del ambiente, y el ESP32 TTGO LoRa32 OLED V2.1.6 en sí, que es el cerebro del sistema. La finalidad de la siguiente tabla es detallar cada conexión para que la implementación del hardware se realice de manera correcta, permitiendo la comunicación entre los sensores, proyección de datos y transmisión por LoRa.

Tabla 3.1: *Tensiones y corrientes de los elementos del sistema IoT.*

Elemento	Voltaje de operación	Corriente de operación
ESP32 TTGO LoRa32 OLED V2.1.6	3.3V / 5V	~200mA (con LoRa y OLED activados)
LoRa Module (SX1276)	3.3V	10mA (modo de espera) / 120mA (transmisión)
Pantalla OLED 128x64 (SSD1306)	3.3V	~20mA
Sensor de CO2 MH-Z19B	5V	~100mA (en funcionamiento)
Sensor de Ruido MAX4466	3.3V / 5V	~10mA

Elaborado por: El Autor

La tabla 3.2 a continuación describe las conexiones detalladas entre los componentes y los pines correspondientes del ESP32. Cada componente vital, incluido el módulo LoRa, la pantalla OLED 128×64, el sensor de CO_2 y el sensor de ruido, está elaborado con el tipo de pin utilizado en el ESP32 y su conexión en la placa.

Tabla 3.2: *Tipos de pines empleados para el sistema IoT.*

Componente	Tipo de Pin en ESP32	Pin/Conexión en la placa
LoRa Module (SX1276)	SPI (MISO, MOSI, SCK, NSS)	MISO (19), MOSI (27), SCK (5), SS (18)
Pantalla OLED 128x64 (SSD1306)	I2C (SDA, SCL)	SDA (21), SCL (22)
Sensor de CO2 MH-Z19B	UART (TX, RX)	RX (25), TX (34)
Sensor de Ruido MAX4466	Pin analógico	GPIO (35)

Elaborado por: El Autor

3.4 Programación del nodo receptor wifi Lora del proyecto IoT.

En la figura 3.14 se observa el código donde se tiene tres partes clave: las bibliotecas indispensables y las constantes necesarias para la conexión con Ubidots y la red Wi-Fi. En primer lugar, las bibliotecas. La primera resulta imprescindible para la comunicación SPI o Interfaz Periférica Serial, es decir, establece el enlace bidireccional entre el módulo LoRa y ESP32, mientras que la segunda se encarga de procesar la comunicación LoRa, es decir, el protocolo de transmisión de largo alcance y bajo consumo. En cambio, además del archivo de configuración, se incluye “UbidotsEsp32Mqtt.h”, una biblioteca que contenga las funciones necesarias para conectar el ESP32 a la

plataforma de Ubidots para que los códigos enviados puedan transmitirse y visualizarse. Tras la inclusión de los archivos, el código define algunas constantes esenciales. UBIDOTS_TOKEN preserva el token de la cuenta de Ubidots del usuario, es decir, el token de acceso a Leyla, similar a la contraseña, y un texto que identifica las credenciales para dar permiso de enviar códigos a su cuenta. WIFI_SSID y WIFI_PASS dan el nombre a la red Wi-Fi y su contraseña, al enlace por el cual el dispositivo se conectará a Internet. Finalmente, el código presenta los nombres de las variables (DEVICE_LABEL, VARIABLE_LABEL_1, VARIABLE_LABEL_2, etc.) que manejarán el equipo y la información, como “Contaminación” o “Ruido”.

Figura 3.14: *Configuración del entorno de Ubidots y credenciales wifi.*

```
#include <SPI.h>
#include <LoRa.h>
#include "UbidotsEsp32Mqtt.h"

const char *UBIDOTS_TOKEN = "xxxxxxxxxxxxx";
const char *WIFI_SSID = "xxxxxxxxxxxx";
const char *WIFI_PASS = "xxxxxxxxxxx";
const char *DEVICE_LABEL = "ESP32";
const char *VARIABLE_LABEL_1 = "Contador 1";
const char *VARIABLE_LABEL_2 = "Contaminacion 1";
const char *VARIABLE_LABEL_3 = "Ruido 1";
const char *VARIABLE_LABEL_4 = "Contaminacion 2";
const char *VARIABLE_LABEL_5 = "Ruido 2";
const char *VARIABLE_LABEL_6 = "Contador 2";
```

Elaborado por: El Autor

En la figura 3.15 se observa el código de configuración del intervalo de tiempo de publicación de datos, se declara un temporizador para monitorear dicho intervalo, se indica el pin analógico del cual se leerán los datos, y se configura la instancia de Ubidots. El PUBLISH_FREQUENCY es un define que se establece en 6000 milisegundos, lo que significa que la información en PUBLISH_INTERVAL se enviará a la plataforma de Ubidots cada 6 segundos. La variable timer es un variable de tipo unsigned long que almacenará el

tiempo en milisegundos desde el último envío, lo que permitirá saber si el intervalo ha pasado. Por otro lado, analogPin se establece en 34, lo que indica en qué pin analógico del ESP32 se conectará el sensor, el cual en este caso será algún sensor para leer datos ambientales. La instancia de la clase Ubidots se nombra Ubidots y se recibe como parámetro la variable UBIDOTS_T , el cual es el encargado de autenticar al ESP32 y permitirle enviar los datos a la cuenta del usuario en Ubidots. Adicionalmente, se incluyen las bibliotecas. La biblioteca es necesaria para habilitar la comunicación I2C, protocolo con el que es conectada la pantalla oled. La biblioteca se usa para manipular con facilidad una pantalla oled. La última biblioteca, es específica para manejar pantallas oled de 128*64 píxeles basadas en el controlador SSD1306.

Figura 3.15: *Configuración de frecuencia de envío de datos a Ubidots.*

```
const int PUBLISH_FREQUENCY = 6000;
unsigned long timer;
uint8_t analogPin = 34;
Ubidots ubidots(UBIDOTS_TOKEN);

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Elaborado por: El Autor

En la figura 3.16 se definen constantes que asignan ciertos números de pines en un ESP32 para la comunicación con el módulo LoRa, así como la frecuencia de operación del módulo en hertzios. Specific pin numbers on the ESP32 are first assigned as SCK 5, MISO 19, and MOSI 27, #define SCK 5, #define MISO 19, y #define MOSI 27 se designan los números 5, 19 y 27 del

ESP32 para los pines de comunicación SPI que se conectan al PCMcia. SCK es el pin de comunicación SPI, mientras que MISO o master in Slave out permite recibir datos desde el mando esclavo y MOSI manda datos al módulo LoRa enumeración. Define SS 18 es el pin de selección de chip y es un pin digital nivel bajo que activa el módulo cuando se designa en low. Define RST 23 se establece para permitir resetear el módulo LoRa si es necesario, y #define DIO0 26 se configura como trabajando con interrupciones, lo que es vital para continuar recibiendo y enviando datos sin interrupciones de microcontroladores. Por último, #define BAND 915E6 establece la frecuencia del módulo LoRa en 915 MHz.

Figura 3.16: Configuración de parámetros de antena Lora.

```
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
#define BAND 915E6
```

Elaborado por: El Autor

En la figura 3.17 es para definir qué pines serían específicos para los botones dependiendo del tipo de microcontrolador que se esté utilizando, lo cual es beneficioso ya que permitirá que el código pueda ser implementado en plataformas de hardware diferentes. El #if defined(ESP32 es una verificación para comprobar si se está compilando para un ESP32. En este caso, el BTN_A, BTN_B y BTN_C se definen como los pines 15, 32 y 14 en el ESP32 que serían tres botones físicos. Posteriormente se tiene la definición #define WIRE Wire, lo cual asigna el objeto Wire que se usará para la

comunicación I2C. Luego, si la tabla es compilada para una placa de Arduino STM32 Feather se define lo siguiente: PA15, PC7 y PC5 serían los pines respectivos para los botones BTN_A, BTN_B y BTN_C en la STM32 Feather. Y #define WIRE Wire que binder al objeto WIRE que es para la comunicación I2C. Si no se cumple una de las dos condiciones previas se define que si el microcontrolador no es de estas dos clases se haga que los pines 9, 6, 5 sean los pines para los botones BTN_A, BTN_B y BTN_C; esto hace que se implemente la capacidad de usar otros botones y demás en otros microcontroladores además de que si no se compila con ninguno de los dos primeros se compilará y definirá los pines para las otras microcontroladoras.

Figura 3.17: *Configuración de pines de cada uno de los elementos del ESP32 Lora.*

```
#if defined(ESP32)
  #define BUTTON_A 15
  #define BUTTON_B 32
  #define BUTTON_C 14
  #define WIRE Wire
#elif defined(ARDUINO_STM32_FEATHER)
  #define BUTTON_A PA15
  #define BUTTON_B PC7
  #define BUTTON_C PC5
  #define WIRE Wire
#else
  #define BUTTON_A  9
  #define BUTTON_B  6
  #define BUTTON_C  5
  #define WIRE Wire
#endif
```

Elaborado por: El Autor

En la figura 3.18 se definen varias variables como en esta sección del código, para contener los valores relacionados con los sensores de contaminación y ruido y varios contadores para seguir cuántos mensajes se revieron, primero, se inicializa las variables Contaminacion11, Ruidos11,

Contaminaciones22, Ruidos22, contadorvalores1, contadorvalores11, contadorvalores22, y contadorvalores2 como enteros y se asume un valor inicial de 0. Posteriormente, en el programase ejecuta un objeto display de la clase Adafruit_SSD1306 que representa una pantalla OLED de 128×64 píxeles y se l bus I2C (&WIRE) con la que se comunicará y mostrará información en la pantalla. Finalmente, se declara una variable LoRaData del tipo String que se utilizará para almacenar los datos que se revieron través de la comunicación LoRa. Los valores de los sensores que pueden ser la de los niveles de contaminación y ruido se almacenan de forma temporal y luego se procesan o se muestran en la pantalla.

Figura 3.18: *Configuración de variables de conteo de datos.*

```
int Contaminacion11 = 0;
int Ruidos11 = 0;
int contadorvalores1 = 0;
int contadorvalores11 = 0;
int Contaminaciones22 = 0;
int Ruidos22 = 0;
int contadorvalores22 = 0;
int contadorvalores2 = 0;
Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &WIRE);
String LoRaData;
```

Elaborado por: El Autor

En la figura 3.19 se define callback, el cual es un llamador de eventos, utilizado para recibir mensajes de MQTT. Si se recibe algún mensaje en el dispositivo, esta función se ejecutará automáticamente, recibirá tres parámetros: topic es un tema en el que se publicó un mensaje; payload es el mensaje en sí, pero en formato de byte; length es la longitud del mensaje en términos de byte. En esta función, primero se imprime el texto “Message arrived [” en el monitor en serie; después, infringe el nombre del topic, lo que ayuda a rastrear en qué canal se ha publicado el mensaje. Después de eso,

esta función entra en un bucle; se forzará a la cantidad de bytes en payload. En este bucle, todos los bytes que están en payload, serán transformados a char por char: “(char)payload[i]”. Por último, se imprime el contenido del mensaje en el monitor en serie. Además, después de un ciclo completo, envía “\n”, un salto de línea, para hacer la impresión más legible. Por lo tanto, esta función ayuda a ver y verificar los mensajes que el dispositivo está recibiendo de MQTT y realizar algún trabajo basado en el mensaje.

Figura 3.19*: Configuración de mensajes de recepción datos.*

```
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();
}
```

Elaborado por: El Autor

En la figura 3.20 se observa la función setup se ejecuta una vez al inicio del programa por lo cual configura los componentes iniciales del sistema, primero define la salida del pin OLED_RST mediante pinMode(OLED_RST, OUTPUT), después realiza un ciclo de reset en la pantalla OLED, bajando el pin, esperando 20ms y subiendo el pin nuevamente para resetear correctamente la pantalla, inicializa la comunicación I2C mediante el objeto Wire e indica a que pines se encuentra conectado los pines de datos y reloj, luego trata de inicializar la pantalla OLED llamando al método display.begin(), el cual tiene por parámetros el tipo de display y conexión, voltaje de alimentación y dirección I2C. Si la inicialización falla if(!display.begin(...) entonces imprime el error por serial monitor SSD1306 allocation failed y entra

en un ciclo for donde el programa se queda frenado y sin poder continuar si no tiene la pantalla OLED correctamente configurada.

Figura 3.20: *Configuración de parámetros de pantalla.*

```
void setup() {
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) {
    Serial.println(F("Asignación de SSD1306 fallo"));
    for(;;);
  }
```

Elaborado por: El Autor

En la figura 3.21 se realiza una serie de configuraciones y actividades iniciales para hacer que la pantalla y el dispositivo funcionen. Elimina cualquier texto o imagen previa en la pantalla OLED usando display.clearDisplay(). Luego establece su color en blanco usando display.setTextColor(WHITE), el tamaño del texto es 1 con display.setTextSize(1), y posiciona el cursor en 0,0 en la pantalla OLED con display.setCursor(0,0). Escríbase “RECEPTOR LORA ” en la pantalla OLED con display.print(“RECEPTOR LORA ”) e imprímalo con display.display(). Después de configurar la pantalla, inicialice la comunicación serial a una velocidad de 9600 baudios con Serial.begin(9600).

Figura 3.21: *Configuración de mensaje de funcionamiento de receptor.*

```
display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0,0);
display.print("RECEPTOR LORA ");
display.display();
Serial.begin(9600);
Serial.println("Prueba receptor LoRa");
SPI.begin(SCK, MISO, MOSI, SS);
LoRa.setPins(SS, RST, DIO0);
```

Elaborado por: El Autor

En la figura 3.22 se intenta inicializar el módulo LoRa con la constante BAND que contiene la frecuencia. Si esta inicialización falla, se imprime en el monitor serial el mensaje Starting LoRa failed! y el programa entra en un bucle sin fin con while(1); lo cual detiene la ejecución del código. En caso de que no se produzca ningún error, se envía un mensaje de error al monitor serial ¡Error al iniciar LoRa!. Luego se inicializa la pantalla OLED con el mensaje LoRa Inicializando! en la posición (0, 10) y se actualiza la pantalla con display.display(). El programa espera un segundo con delay(1000). Se conecta al WiFI con las credenciales dadas por WIFI_SSID y WIFI_PASS con ubidots.connectToWifi() y se establece un callback en Ubidots con ubidots.setCallback(callback). Ubidots se inicializa con ubidots.setup() y de ser necesario, se reconecta a Ubidots con ubidots.reconnect(). Finalmente, se guarda el tiempo actual en la variable timer con millis() y el cual permite controlar la frecuencia de las publicaciones más adelante.

Figura 3.22: *Configuración de mensaje de error de antena Lora.*

```
  if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  Serial.println("¡Error al iniciar LoRa!");
  display.setCursor(0,10);
  display.println("LoRa Inicializando!");
  display.display();
  delay(1000);
  ubidots.connectToWifi(WIFI_SSID, WIFI_PASS);
  ubidots.setCallback(callback);
  ubidots.setup();
  ubidots.reconnect();
  timer = millis();
}
```

Elaborado por: El Autor

En la figura 3.23 se observa el código, LoRa.parsePacket se verifica llamando a la función desde loop(). Si se ha recibido un paquete, es decir, si

packetSize es mayor que 0, se imprime el mensaje paquete recibido en el monitor serial. Se establece un bucle while LoRa.available() que se ejecuta mient LoRa en el buffer. Dentro de este bucle, se lee el contenido del paquete utilizando LoRa.readString, que almacena el contenido del paquete como una cadena de caracteres en la variable LoRaData. Entonces, LoRaData se muestra en el monitor serial. Este procedimiento permite la recepción y visualización de los datos enviados por otros dispositivos LoRa.

Figura 3.23: *Configuración de receptor de paquetes Lora.*

```
void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.print("paquete recibido ");
    while (LoRa.available()) {
      LoRaData = LoRa.readString();
      Serial.print(LoRaData);
    }
```

Elaborado por: El Autor

En la figura 3.24 se obtienen y convierten los valores de contaminantes y ruido de la cadena de datos que se envían por el módulo LoRa. En primer lugar, mediante la función substring(0,2) sobre la variable LoRaData se obtienen los primeros dos caracteres, que se guardan en la variable Contaminacion1 y luego se convierten a un tipo float con toFloat() en la variable Contaminaciones1 para guardar el valor numérico de la contaminación. Luego se repite el proceso anterior con el valor de ruido con Ruido1 convirtiéndolo en Ruidos1, y después con los valores de contaminación y ruido para el sensor 2, con los índices 7-9 para Contaminacion2 y 10-12 para Ruido2, convirtiéndolos en Contaminaciones2 y Ruidos2, respectivamente. Después se obtiene el valor del RSSI (Indicador

de Intensidad de Señal recibida) con la función LoRa.packetRssi(). Este valor corresponde a la calidad que se ha registrado con la señal recibida, se imprime valor en el monitor serial de la siguiente manera: con RSSI, teniendo el valor de rssi, lo que entrega un respaldo acerca de la intensidad de la señal recibida con objetivos de diagnóstico o ajuste de comunicación LoRa.

Figura 3.24: *Configuración de variables de string a float de los datos.*

```
String Contaminacion1 = LoRaData.substring(0,2);
float Contaminaciones1 = Contaminacion1.toFloat();
String Ruido1 = LoRaData.substring(3,5);
float Ruidos1 = Ruido1.toFloat();
String Contaminacion2 = LoRaData.substring(7,9);
float Contaminaciones2 = Contaminacion2.toFloat();
String Ruido2 = LoRaData.substring(10,12);
float Ruidos2 = Ruido2.toFloat();
int rssi = LoRa.packetRssi();
Serial.print(" with RSSI ");
Serial.println(rssi);
```

Elaborado por: El Autor

En la figura 3.25 se evalúa si los valores de Contaminaciones1 y Ruidos1 son mayores que 0 con la condición if. Si la condición se cumple, se aumenta el contador de mensajes contadorvalores1, que permite realizar un seguimiento del número de paquetes recibidos. Luego, se limpia la pantalla del display OLED display.clearDisplay() y se establece el cursor para mostrar información en posiciones diferentes. En la siguiente línea, se muestra el título “UCSG 2024 RECEPTOR 1” y “SENSOR DE Contaminacion 1” y, a continuación, el número de mensaje recibido, seguido del valor de contaminaciones y ruido Contaminaciones1 y Ruidos1, y, por último, el valor de rssi. Estos valores se visualizan en pantalla en lugares diferentes, por lo que se creará una pantalla con todas las variables colocadas. Luego de mostrar en display, los valores de contaminación y ruido se asignan a las

variables Contaminacion11 y Ruidos11 y el contador de mensajes en contadorvalores11 para tener una variable persistente con el último valor.

Figura 3.25: *Configuración de pantalla de datos del nodo sensor 1 recibido.*

```
if (Contaminaciones1 && Ruidos1 > 0) {
  contadorvalores1++;
  display.clearDisplay();
  display.setCursor(0,0);
  display.print("UCSG 2024 RECEPTOR 1");
  display.setCursor(0,20);
  display.print("SENSOR DE Contaminacion 1");
  display.setCursor(0,30);
  display.print("Mensaje #:");
  display.setCursor(75,30);
  display.print(contadorvalores1);
  display.setCursor(0,40);
  display.print("Te1: ");
  display.setCursor(25,40);
  display.print(Contaminaciones1);
  display.setCursor(60,40);
  display.print("Hu1: ");
  display.setCursor(85,40);
  display.print(Ruidos1);
  display.setCursor(0,50);
  display.print("RSSI:");
  display.setCursor(30,50);
  display.print(rssi);
  display.display();
  Contaminacion11 = Contaminaciones1;
  Ruidos11 = Ruidos1;
  contadorvalores11 = contadorvalores1;
}
```

Elaborado por: El Autor

En la figura 3.26 se observa la segunda parte del código, que evalúa si los valores de Contaminaciones2 y Ruidos2 son mayores que 0, para ello utiliza la condición: if (Contaminaciones2 && Ruidos2 > 0). Una vez que cumpla esa condición, se incrementa el contador de mensajes contadorvalores2, el cual sirve para llevar un registro de la cantidad de mensajes recibidos. Posteriormente procede a limpiar la pantalla del display OLED display.clearDisplay(), y también coloca el cursor en posiciones distintas para luego mostrar la información en la pantalla. Llama un mensaje donde se coloca el encabezado “UCSG 2024 RECEPTOR”, seguido de un segundo mensaje “SENSOR DE Contaminacion 2”, el tercer mensaje es el

número de mensaje recibido que está almacenado en el contador contadorvalores2, el cuarto mensaje son los valores de contaminación y ruido almacenados en las variables de Contaminaciones2 y Ruidos2, el quinto mensaje es la intensidad de la señal RSSI rssi. Esa información es presentada de forma organizada en el display con la finalidad de que el usuario pueda observar con claridad los datos revisados. Al actualizar el display, los valores de contaminación y ruido se almacenan en las variables Contaminaciones22 y Ruidos22, y el contador de mensajes se guarda en contadorvalores22, con estos valores realiza el registro persistente del valor más reciente para cada uno de los contadores.

Figura 3.26: *Configuración de pantalla de datos del nodo sensor 2 recibido.*

```
if (Contaminaciones2 && Ruidos2 > 0) {
  contadorvalores2++;
  display.clearDisplay();
  display.setCursor(0,0);
  display.print("UCSG 2024 RECEPTOR");
  display.setCursor(0,20);
  display.print("SENSOR DE Contaminacion 2");
  display.setCursor(0,30);
  display.print("Mensaje #:");
  display.setCursor(75,30);
  display.print(contadorvalores2);
  display.setCursor(0,40);
  display.print("Te2: ");
  display.setCursor(25,40);
  display.print(Contaminaciones2);
  display.setCursor(60,40);
  display.print("Hu2: ");
  display.setCursor(85,40);
  display.print(Ruidos2);
  display.setCursor(0,50);
  display.print("RSSI:");
  display.setCursor(30,50);
  display.print(rssi);
  display.display();
  Contaminaciones22 = Contaminaciones2;
  Ruidos22 = Ruidos2;
  contadorvalores22 = contadorvalores2;
}
```

Elaborado por: El Autor

En la figura 3.27 se observa la verificación de la conexión con Ubidots si debiese estar en línea, if(!ubidots.connected()), y ubidots.reconnect() si no lo está. A continuación, se evalúa si ha pasado lo suficiente tiempo desde el

último ciclo de publicación si lo están publicando. if (abs(millis() – timer) > PUBLISH_FREQUENCY, es decir, el programa publicará los datos para Ubidots después de un intervalo específico, definido por PUBLISH_FREQUENCY. Dentro del condicional si las condiciones de los valores de contaminación Contaminaciones11 y Contaminaciones22 son mayores que 0, ubidots.add() los datos relevantes, respectivamente, dando la etiqueta a las vars y los valores, mens_counter y contaminación, ruido.

Figura 3.27: *Configuración de proceso de envío de datos a Ubidots.*

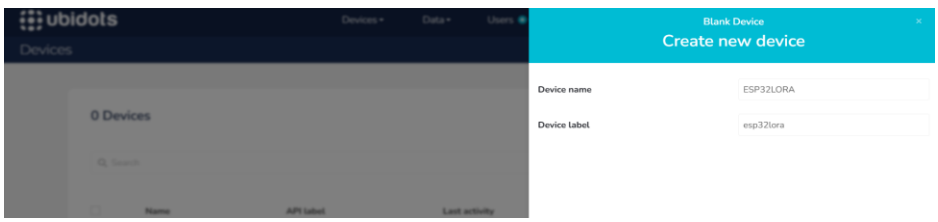
```
  if (!ubidots.connected()) {
    ubidots.reconnect();
  }

  if (abs(millis() - timer) > PUBLISH_FREQUENCY) {
    if (Contaminaciones11 && Contaminaciones22 > 0) {
      ubidots.add(VARIABLE_LABEL_1, contadorvalores11);
      ubidots.add(VARIABLE_LABEL_2, Contaminaciones11);
      ubidots.add(VARIABLE_LABEL_3, Ruidos11);
      ubidots.add(VARIABLE_LABEL_4, Contaminaciones22);
      ubidots.add(VARIABLE_LABEL_5, Ruidos22);
      ubidots.add(VARIABLE_LABEL_6, contadorvalores22);
      ubidots.publish(DEVICE_LABEL);
      timer = millis();
    }
  }
}
ubidots.loop();
```

Elaborado por: El Autor

3.5 Configuración de la plataforma de Ubidots para la recepción de datos del sistema IoT.

En el presente subcapítulo se lleva a cabo las configuraciones de la plataforma IoT para la recepción de los datos enviados por el sistema IoT, en la figura 3.28 se observa la creación del espacio de acopio de información.

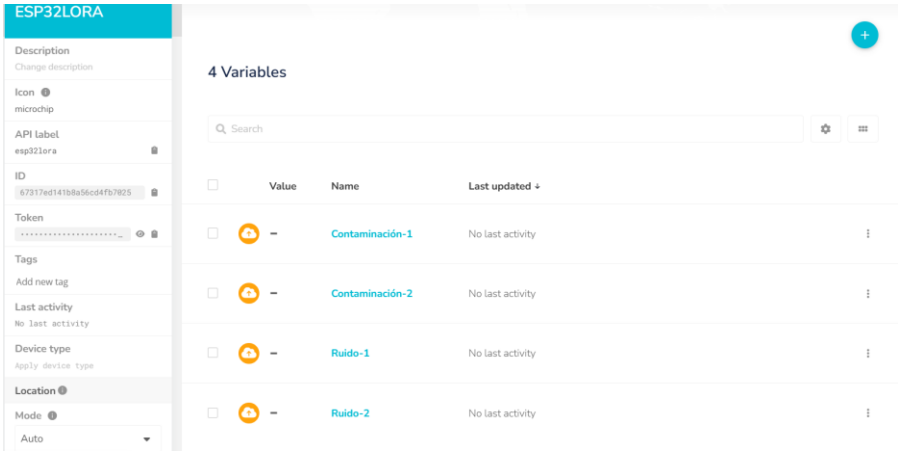
Figura 3.28: *Creación de proyecto en la plataforma de Ubidots.*



Elaborado por: El Autor

En la figura 3.29 se aprecia la creación de las cuatro variables para recibir la información sensada por cada uno de los nodos sensores emisores, tales como el sensor de contaminación y ruido, los cuales son transmitidos por el nodo receptor wifi Lora.

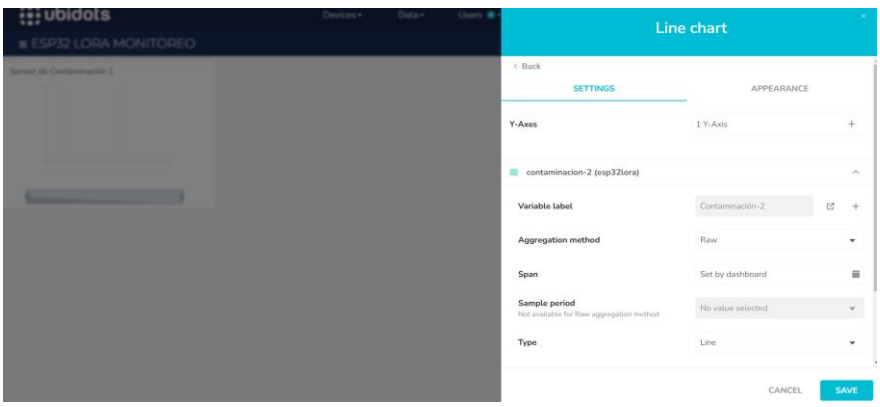
Figura 3.29: *Establecer las variables de recepción en la plataforma de Ubidots.*



Elaborado por: El Autor

En la figura 3.30 se observa la configuración de los diseños, tamaños y colores de los widgets asignados de cada una de las variables de recepción de datos del sistema IoT.

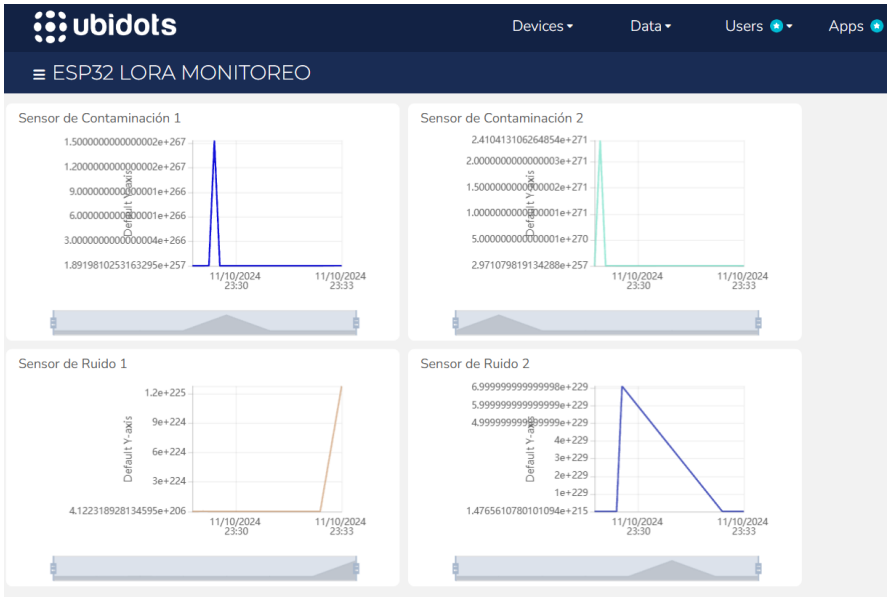
Figura 3.30: *Creación de widgets visuales en la plataforma de Ubidots.*



Elaborado por: El Autor

En la figura 3.31 se visualiza la ejecución de los widgets al momento de recibir los datos de los nodos sensores emisores, para su respectivo análisis e interpretación, según la necesidad del usuario.

Figura 3.31: *Funcionamiento de los widgets de variables en la plataforma de Ubidots.*



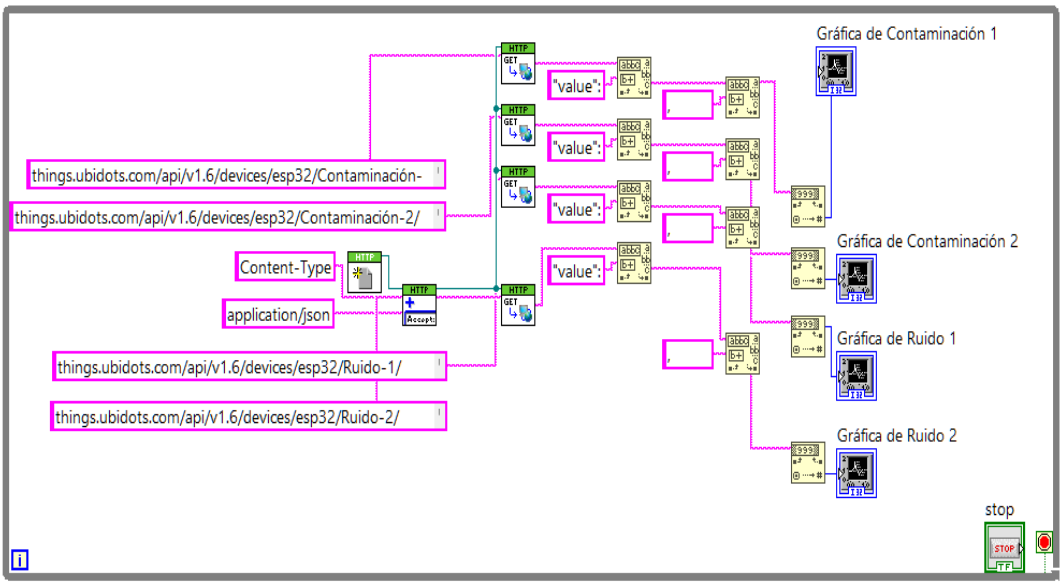
Elaborado por: El Autor

3.5 Programación de LabVIEW para recepción de datos de Ubidots.

La figura 3.32 presenta el diseño de los enlaces HTTP utilizados en la adquisición de datos en tiempo real que la plataforma Ubidots envía al nodo receptor. La configuración implica cuatro secciones en las que dos de ellas están destinadas a la adquisición de datos de contaminación del aire y otras dos para la adquisición de datos de los niveles de ruido. Cada una de estas variables tiene un token propio proporcionado por Ubidots y se transmite de manera segura para autenticar la conexión y permitir que los datos fluyan hacia la plataforma receptora. Además, el diseño implica un proceso crítico después de que los datos de las cuatro variables lleguen a Ubidots. Aquí, los

datos se remodelan para que sean legibles y utilizables dentro del campo de LabVIEW: cada variable es vista, analizada y procesada de acuerdo con los términos establecidos. De esta manera, esta configuración facilita una unión perfecta entre la captura y la interpretación de datos en el tiempo con LabVIEW para monitorear las condiciones de calidad aeróbica del coloreado del aire y el ruido en tiempo real.

Figura 3.32: *Estructura de código para recepción de datos en LabVIEW.*

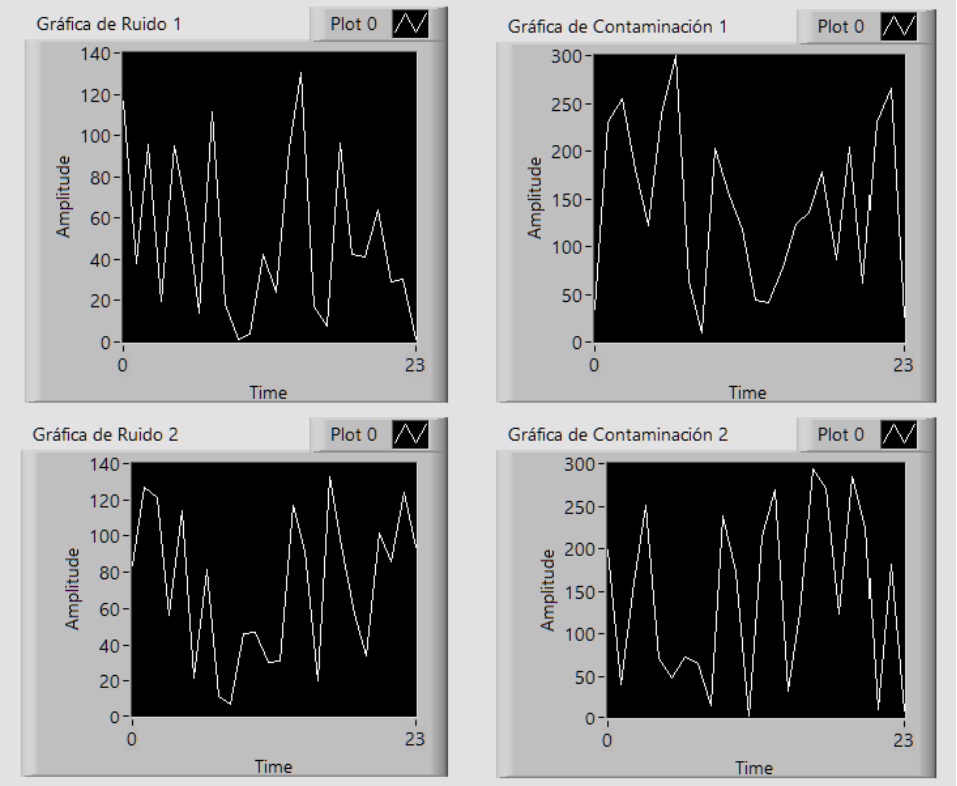


Elaborado por: El Autor

En la Figura 3.33 se muestra el diseño de la interfaz en LabVIEW para el monitoreo de variables como contaminación y ruido en tiempo real. La interfaz es didáctica y amigable, por lo que se pueden visualizar de forma clara y detallada datos estructurados, alertas, tendencias y niveles críticos tanto actuales como configurados para el monitoreo integral. En adición a lo anterior, la visualización y alarma gráfica funciona con un rastreo histórico y parámetros configurados en alarmas, lo cual permite determinar patrones y

actuar de forma rápida ante elementos críticos. Este sistema presenta trazabilidad de datos, pues quienes toman las decisiones pueden creer en la veracidad y valor de la información para la toma de decisiones.

Figura 3.33*: Diseño de la interfaz gráfica de LabVIEW.*



Elaborado por: El Autor

3.6 Análisis de costos del sistema IoT propuesto.

En cuanto a la Tabla 3.6, se refiere al Análisis de costos del sistema IoT propuesto, y proporciona una descripción general de los costos estimados de cada componente requerido para configurar el sistema IoT propuesto para monitorizar el ruido y las variables de contaminación ambiental. La lista incluye los elementos esenciales junto con sus costos unitarios en dólares estadounidenses y la cantidad de componentes obligatorios.

Tabla 3.3: *Costos de los elementos propuesto del sistema.*

Elemento	Costo Estimado Individual (USD)	Cantidad de elementos
ESP32 TTGO LoRa32 OLED V2.1.6	$50	3
Sensor de CO2 MH-Z19B	$5	2
Sensor de Ruido MAX4466	$4	2
Alimentación de 5 voltios	$7	3
Cables y Conectores	$10	2

Elaborado por: El Autor

En la Tabla 3.4 se aprecia el cronograma de actividades, que registran las actividades a realizarse en el marco de la instalación, configuración y puesta en funcionamiento del sistema propuesto. Con una duración de una semana, el presente esquematiza cada una de las actividades diarias a llevarse a cabo, desde la recepción de los componentes hasta la entrega formal del sistema final, permitiendo el desarrollo ordenado y la continuidad de cada una de las etapas involucradas.

Tabla 3.4: *Cronograma de actividades propuestas del sistema IoT.*

Día	Actividad
Día 1	Recepción de componentes y conexión del módulo LoRa.
Día 2	Instalación de pantalla OLED y sensor de CO_2.
Día 3	Instalación del sensor de ruido y programación del ESP32.
Día 4	Pruebas de comunicación LoRa y validación de sensores.
Día 5	Integración del sistema y pruebas de alcance LoRa.

Día 6	Instalación de alimentación y pruebas finales de funcionamiento.
Día 7	Documentación, ajustes finales y entrega del sistema.

Elaborado por: El Autor

Entre estas actividades, se encuentran: la conexión de los módulos LoRa, la instalación de la pantalla OLED y el LCD, seguido por el del sensor de ruido, mientras que el ESP32 ya procesa y emite datos. Las actividades de pruebas, incluida la comunicación, la validación de los sensores y la fuente de alimentación y las actividades de verificación del alcance son para la funcionalidad y capacidad apropiada. Posteriormente, la documentación, y la entrega del sistema de monitoreo ambiental.

Conclusiones y recomendaciones

Conclusiones

- Con el desarrollo de la interfaz en Arduino IDE, se logró implementar un sistema de monitoreo de interferencia y ruido en la ciudad de Guayaquil. La interfaz desarrollada permitirá visualizar y analizar dicha información en tiempo real, condición necesaria para determinar los patrones de interferencia que se suscitan en la ciudad y tener la capacidad de reaccionar ante ellos cuando suceden eventos disruptivos en la red.
- Utilizar LabVIEW para un entorno gráfico para el manejo de las variables de la plataforma IoT Ubidots es una excelente alternativa para representar visualmente los datos recolectados de los nodos sensores. En conjunto este diseño gráfico realza el uso y la toma de decisiones en el manejo de la interferencia y ruido entre los usuarios, resultando fácil y profesional.
- Los diagramas aseguraron una coordinación adecuada a nivel de módulo entre los dispositivos de emisión y recepción, lo que ha garantizado la compatibilidad entre los nodos correspondientes. Para la red de sensores en su conjunto, estos diagramas garantizaron la disposición de los elementos en lugares óptimos y promovieron una comunicación adecuada entre los dispositivos.
- Como resultado de la evaluación de la viabilidad y la estimación de los costos del diseño de la red de sensores inalámbricos mediante ESP32, el proyecto resulta ser técnico y económicamente viable. En el contexto

del bajo costo de los módulos ESP32 y la calidad de su capacidad de monitoreo y comunicación inalámbrica, el proyecto también es eficaz para la implementación a gran escala en áreas urbanas, como Guayaquil.

Recomendaciones

- Se sugiere como mejora futura, sería recomendable continuar mejorando la interfaz de usuario para ser más intuitiva y personalizable. Otras mejoras pueden incluir la posibilidad de proporcionar alertas automáticas o notificaciones al usuario cuando hay niveles elevados de interferencia y ruido. Además, sería viable investigar otras plataformas de programación para que la solución sea más escalable.
- Se recomienda que por el lado de funcionalidades avanzadas, es ideal implementar varias opciones en el entorno gráfico para el análisis predictivo de los datos recopilados sobre la interferencia y el ruido. Otra idea es agregar la opción para generar automáticamente informes basados en los datos recopilados. Ayudará a los monitores en tiempo real a laborar fácilmente.

Bibliografías

Aghenta, L. O. (2020). *Open source SCADA systems for small renewable power generation* [Masters, Memorial University of Newfoundland]. https://research.library.mun.ca/14333/

Agüero, R., Bote, M., Cano, M., & Felicia, S. (2024, mayo 1). *Investigación e innovación en ingeniería telemática: La sociedad científica de ingeniería telemática*. https://upcommons.upc.edu/handle/2117/407671

Aliaga, C. H., & Serquén, S. A. (2022). *Sistema de acceso y monitoreo remoto a un servidor en la nube para gestionar los datos del procesamiento de línea de harina en Chimú Agropecuaria—Trujillo*. http://repositorio.unprg.edu.pe/handle/20.500.12893/10456

Andrade, D. F., & Brigss, S. A. (2023). *Análisis de factibilidad y diseño de una red LoRaWAN en Cuenca para la comunicación de una red de sensores ambientales, administrada por el Instituto de Estudios de Régimen Seccional del Ecuador (IERSE)* [bachelorThesis, Universidad del Azuay]. http://dspace.uazuay.edu.ec/handle/datos/13741

Andrade, Vizuete, J. D. R., Olivero, A. P. M., & Sandra María Sosa Calero. (2022). Sistema de Monitoreo de temperatura y humedad en el hogar aplicando IoT de bajo costo: Home temperature and humidity monitoring system applying low-cost IoT. *Revista Científica Multidisciplinar G-nerando*, *3*(2), Article 2. https://revista.gnerando.org/revista/index.php/RCMG/article/view/39

Belmonte, T. (2022). *Diseño de una aplicación para la gestión de una estación meteorológica mediante protocolo de transporte de telemetría por mensaje de colas (MQTT) sobre modulo de radio frecuencia ESP32* [Thesis]. http://repositorio.umsa.bo/xmlui/handle/123456789/30672

Benítez, V. R. (2021). Sistema De Telemedición De Parámetros Del Agua Para Múltiples Puntos En Conductoras Con Redes Inalámbricas Lora. *Telemática*, *20*(2), Article 2. https://revistatelematica.cujae.edu.cu/index.php/tele/article/view/415

Bernardi, C. A. (2024). Avances sostenibles del transporte aéreo a través de la movilidad aérea urbana y la movilidad área avanzada: Integración de tecnologías emergentes. *In-Genium*, *no. 7*. http://sedici.unlp.edu.ar/handle/10915/167535

Bonilla, V. N. (2023). *Creación de un prototipo de monitoreo de temperatura del aire utilizando equipos de desarrollo de bajo costo y comunicación LoRaWAN: Implementación física de un prototipo de monitoreo de temperatura del aire utilizando equipos de desarrollo de bajo costo y comunicación LoRaWAN.* http://bibdigital.epn.edu.ec/handle/15000/24933

Borodinecs, A., Palcikovskis, A., & Jacnevs, V. (2022). Indoor Air CO2 Sensors and Possible Uncertainties of Measurements: A Review and an Example of Practical Measurements. *Energies*, *15*(19), Article 19. https://doi.org/10.3390/en15196961

Callejón, R. (2022). *Estudio de un sistema IoT para su aplicación en gestión de Smart Cities* [Bachelor thesis, Universitat Politècnica de Catalunya]. https://upcommons.upc.edu/handle/2117/373610

Carranza, S. (2021, noviembre 3). ESP32 + UBIDOTS: PRIMEROS PASOS. *TodoMaker*. https://todomaker.com/blog/esp32-ubidots-primeros-pasos/

Casas, V. P., Guzmán, H. A. M., Nolasco, J. J. M., Medina, J. A. P., & Sánchez, M. G. B. (2021). Diseño De Sensor Iot De Variables Climáticas Para Cultivo Aeropónico Aplicado A Lechuga (Iot Sensor Design Of Climate Variables For Aeroponic Cultivation Applied To Lettuce). *Pistas Educativas*, *43*(140), Article 140. https://pistaseducativas.celaya.tecnm.mx/index.php/pistas/article/view/2600

Castellanos, W. (2020, junio). *Evaluación técnica y financiera para la implementación de tecnología LoRaWAN en granjas de producción avícola*. https://repository.unimilitar.edu.co/items/c1efdca9-63db-4a49-b026-47e95cdee1b0

Chanchí, G.-E., Ospina, M.-A., & Saba, M. (2022). Sistema IoT para el monitoreo de variables climatológicas en cultivos de agricultura urbana. *Revista científica*, *44*(2), 257-271.

Chasiluisa, R. J. (2020). *Diseño y construcción de un sistema automático de control y monitorización del microclima de un invernadero para el Cantón Penipe-Chimborazo* [bachelorThesis, Riobamba: Universidad Nacional de Chimborazo]. http://dspace.unach.edu.ec/handle/51000/7098

Chica, M. A. V., & Leonardo, C. G. (2024). Estudio bibliográfico sobre las metodologías de big data y su aplicación en la movilidad inteligente. *ReCIBE, Revista electrónica de Computación, Informática, Biomédica y Electrónica*, *13*(1), Article 1. https://doi.org/10.32870/recibe.v13i1.349

Chunga, J. D., & Niacata, B. A. (2023). *Desarrollo de un prototipo para la medición de caudal de agua empleando un sistema IOT-RF.* http://repositorio.utc.edu.ec/handle/27000/11244

Cooplalonja. (2022, octubre 6). *Internet De Las Cosas: Definición Y Evolución - Coop La Lonja*. https://cooplalonja.com.ar/definicion-de-internet-de-las-cosas-segun-autores/

Cuevas, J. L. S., Castillo, M. M. G., Peña, L. J. M., Jácome, O. S., & Pacheco, A. E. V. (2023). Monitoreo De Temperatura Y Humedad Ambiental Con Lorawan: Iot En El Desafío Energético Y El Cambio Climático. *Revista Ipsumtec*, *6*(5), Article 5.

Cuji, C. L. R., Estrella, D. R. Ñ., Logroño, C. A. P., & Silva, W. D. C. (2023). Panel de monitoreo y control remoto de un proceso industrial utilizando la plataforma Ubidots y Node-Red. *Revista Científica Arbitrada Multidisciplinaria PENTACIENCIAS*, *5*(1), 576-592.

Destec. (2022). *Módulo sensor de sonido (MAX4466) [NB117]*. https://destecmex.com/producto/modulo-sensor-de-sonido-max4466-m3-110a/

Domínguez, E. A. (2024). *Sistema de información remoto para el control de la calidad de agua en el río Moche usando una plataforma IoT*. https://hdl.handle.net/20.500.14414/22114

Emariete. (2020, diciembre 27). *La biblia del sensor de CO2 MH-Z19B*. https://emariete.com/sensor-co2-mh-z19b/

Eras, A. P. P., Bueno, J. I. R., Gavilanes, M. D. G., & Núñez, E. F. H. (2024). Manual de Iniciación en el Uso y Aplicaciones Básicas de la Tarjeta ESP32.: Introduction Manual for the Use and Basic Applications of the ESP32 Card. *Revista Científica Multidisciplinar G-nerando*, *5*(2), Article 2. https://doi.org/10.60100/rcmg.v5i2.254

Espinosa, A., Ponte, D., Gibeaux, S., & González, C. (2021). Estudio de Sistemas IoT Aplicados a la Agricultura Inteligente. *Revista Plus Economía*, *9*(1), Article 1.

Ferrandez, F.-J., Maciá, L., Platero, M., & Silveira, D. (2021). *Plataforma IoT basada en paneles de monitorización (Dashboard)*. http://rua.ua.es/dspace/handle/10045/112947

Ferro, R., & Rodríguez, H. (2021, junio 30). *Análisis, definición e implementación de una plataforma de pruebas IoT para el Semillero de Investigación SCISEN*. https://repository.udistrital.edu.co/items/d8d25c2f-6ecb-4cc0-b4ce-5e312b91bd30

Finistrosa, R. (2024). *Gallinero IoT. Aplicación de la tecnología de redes LoRaWAN en el desarrollo de un sistema de ayuda a la selección de gallinas ponedoras*. https://hdl.handle.net/20.500.14468/24152

Flexitron. (2022). Sensores IoT - Smart Sensors. *MATRIX electrónica*. https://www.matrix.es/sensores

Flores, X., & Guadalupe, J. (2024). *Desarrollo de una red inalámbrica ad hoc para realizar tareas de monitoreo de temperatura y humedad*. http://repositorio.uppuebla.edu.mx:8080/xmlui/handle/123456789/490

Galvao, I. C. do N. (2024). *A utilização do RFID para automatizar fazendas com bovinos de corte*. https://repositorio.pucgoias.edu.br/jspui/handle/123456789/8038

Giraldo, M. A. (2024). *Pruebas de seguridad a nivel de radio (Espectro 915 MHz), en dispositivos LPWA (LoRa)*. https://ridum.umanizales.edu.co/handle/20.500.12746/6984

Gómez, Y. L. (2022). *Diseño de un prototipo multimodal utilizando tecnologías inalámbricas y móviles para monitorear en tiempo real la contaminación auditiva en el centro histórico de la ciudad de Riobamba.* http://dspace.espoch.edu.ec/handle/123456789/20804

Guerrero, C. S. (2022). *Evaluación de los retardos en redes LoraWAN multisalto con topología lineal.* [bachelorThesis, Quito : EPN, 2022.]. http://bibdigital.epn.edu.ec/handle/15000/22721

Hernández, C., & Marcos, M. (2024). *Implementación de un prototipo mediante Internet de las cosas para mejorar la gestión de recojo de residuos sólidos en contenedores soterrados, en el distrito de San Miguel*. https://repositorio.uch.edu.pe//handle/20.500.12872/940

Huacho, W. (2022). Diseño de un módulo de control basado en plataforma IOT para el monitoreo remoto de motores de inducción de baja potencia. *Universidad Continental*. https://repositorio.continental.edu.pe/handle/20.500.12394/12585

Huertas, K. N. J. (2022). Sistema de información de medición acústica para la configuración de espacios y el rendimiento laboral en la comisaría de Piura. *Repositorio Institucional - UCV*. https://repositorio.ucv.edu.pe/handle/20.500.12692/95514

Incibe. (2023, junio 15). *LoRaWAN y su aportación a las tecnologías IoT*. https://www.incibe.es/incibe-cert/blog/lorawan-y-su-aportacion-las-tecnologias-iiot

Iqbal, T., & Manzoor, S. (2020, octubre 8). *IoT based Renewable Energy Management and monitoring system for the First Passive House in Newfoundland*. Second International Conference on Enhanced Research and Industrial Application 2020, Yogyakarta, Indonesia. https://www.researchgate.net/publication/347439392_IoT_based_Renewable_Energy_Management_and_monitoring_system_for_the_Frist_Passive_House_in_Newfoundland

Jaramillo, J. M. (2021). *Estudio del diseño de un sistema IoT para el monitoreo de datos de uso, alertas y geolocalización en refrigeradores para ventas de una empresa de lácteos usando la red LoRaWAN*. https://tesis.pucp.edu.pe/repositorio//handle/20.500.12404/19040

Jecrespom. (2024, marzo 18). *Arquitecturas IoT*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/tag/arquitecturas-iot/

Jiménez, P. B., & Chávez, L. A. (2023). *Sistema IoT de vigilancia a través de la plataforma Telegram*. http://www.dspace.espol.edu.ec/handle/123456789/60566

León, E., Chaffo, F. B. F., & Chavez, J. L. (2024). Sistema que alerta la presencia de anomalías en logs de equipos IOT para prevenir el acceso no autorizado. *Universidad Peruana de Ciencias Aplicadas (UPC)*. https://repositorioacademico.upc.edu.pe/handle/10757/675786

Lilygo. (2024). *LoRa32 V2.1_1.6*. https://www.lilygo.cc/products/lora3

Loor, J. A., & Macías, I. A. (2024). *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* [bachelorThesis]. http://dspace.ups.edu.ec/handle/123456789/28127

Lora. (2022). *Conocimientos de antena*. https://www.loraantenna.com/es/antenna-knowledge/

Maldonado, J. M., & Tigrero, H. B. (2023). *Diseño e implementación de un sistema de monitoreo web de sensores por comunicación LORAWAN para planta de tratamiento de aguas terciarias* [bachelorThesis]. http://dspace.ups.edu.ec/handle/123456789/24120

Martín, M. (2021). *La transformación del Mundo de la Moda en base a los avances tecnológicos de la Industria 4.0*. https://uvadoc.uva.es/handle/10324/52121

Martínez, V. (2022, abril 5). *Historia y origen del IoT*. https://www.ui1.es/blog-ui1/historia-y-origen-del-iot

Mehta, J. (2023, marzo 10). RSA vs. AES Encryption: Know Key Differences. *CheapSSLWeb.Com Blog*. https://cheapsslweb.com/blog/rsa-vs-aes-encryption/

Microside. (2020). *Manual X-NODE LTE GNSS*. https://agelectronica.lat/pdfs/textos/X/XIDE-0007.PDF

Monolitic. (2020, noviembre 26). *Agrotech: Soluciones IoT para un uso inteligente de los recursos hídricos*. https://www.monolitic.com/agrotech-soluciones-iot-para-un-uso-inteligente-de-los-recursos-hidricos/

Montaño, M. A. (2021). *Redes definidas por software para redes móviles en arquitectura de IoT* [bachelorThesis]. http://dspace.utpl.edu.ec/handle/20.500.11962/28279

Morales, A. (2022, febrero 12). Esp32 características y pines. *PASIÓN ELECTRÓNICA*. https://pasionelectronica.com/esp32-caracteristicas-y-pines/

Morrón, D. L. R., & Eduardo, D. (2024). *Prototipo electrónico basado en IOT para prevención de inundaciones en el Embalse del Guájaro—Ecoguájaro* [Trabajo de grado - Maestría, Corporación Universidad de la Costa]. https://hdl.handle.net/11323/12999

Niño, N. Y., & Rimarachin, N. R. (2023). Evaluación de técnicas de cifrado para el intercambio de datos en internet de las cosas en el ámbito de la salud. *Repositorio Institucional - USS*. https://repositorio.uss.edu.pe/handle/20.500.12802/10593

Ortiz, J. O. (2024). *Diseño e implementación de un prototipo para localización de descargas atmosféricas mediante el método de multiestación utilizando tecnología Lorawan.* http://dspace.espoch.edu.ec/handle/123456789/21610

Parra, H. (2023). *Análisis Forense de Entornos IoT Caso de estudio:"Sistema Inteligente para la Captura de Datos de Medidores de Energía Eléctrica"*. http://redi.ufasta.edu.ar/jspui/handle/123456789/1016

Patricio, C. Á. (2020). *Implementación De Un Plc Siemens Logo En Plataforma De Aplicaciones IoT*. https://rinacional.tecnm.mx/jspui/handle/TecNM/6006

Peñaloza, J. M., & Yupanqui, R. (2022). Diseño de un sistema IoT de bajo costo basado en LPWAN para cultivos hidropónicos. *Universidad Privada de Tacna*. http://repositorio.upt.edu.pe/handle/20.500.12969/2390

Pérez, L. C. (2022). Inteligencia artificial y Big data en ciudades inteligentes. *instname:Universidad de Bogotá Jorge Tadeo Lozano*. https://doi.org/10/28702

Picazo, L. A. (2021). *Internet of Things: Control y monitorización de Smart Rural a través de tecnología LPWAN aplicado a la apicultura*. https://openaccess.uoc.edu/handle/10609/133506

Romero, W. R. (2023). Diseño e implementación de prototipo de adquisición de parámetros eléctricos para el consumo de energía de electrodomésticos en los hogares mediante IoT. *Repositorio Institucional - UTP*. http://repositorio.utp.edu.pe/handle/20.500.12867/8645

Rufus, C. (2023, marzo 19). Connecting the ESP32 World With I2C Communication. *Medium*.

https://medium.com/@ceavinrufus/connecting-the-esp32-world-with-i2c-communication-52ae13f07b6c

Sanabria, J. S., & Bravo, A. F. (2020). *Esquema de Seguridad de Datos Entre los Nodos y el Gateway en una Red LoRaWan*. http://hdl.handle.net/11349/25252

Sánchez, J. (2023). *Diseño e implementación de una app de monitorización de taquillas a través del módulo ESP32 y las tecnologías wifi, BLE y RFID*. http://dspace.umh.es/handle/11000/30197

Sánchez, S. A. (2022). *Diseño y simulación de un sistema de adquisición de datos y tracking del proceso productivo de café: Diseño y simulación del sistema de adquisición de datos en IoT con LoRaWAN para el proceso de secado de café.* [bachelorThesis]. http://bibdigital.epn.edu.ec/handle/15000/23255

Santander, C. L. (2024). *Modelo de negocio para la creación de una solución de mejoramiento en el alumbrado público de Nocaima*. http://hdl.handle.net/11349/41735

Stackscale. (2023, abril 27). *Modelo OSI: 7 capas y ciberataques habituales*. https://www.stackscale.com/es/blog/modelo-osi/

Tangarife, J. (2023, marzo 28). *Android-powered GPS Fleet Tracker*. Ubidots Blog. https://ubidots.com/blog/android-powered-gps-fleet-tracker/

Toapanta, D. I. (2024). *Diseño de la red lorawan para la estación de investigación de la Universidad Politécnica Salesiana de Ayora* [bachelorThesis]. http://dspace.ups.edu.ec/handle/123456789/27942

Tolocka. (2024, agosto 5). *Integra Ubidots y Micropython en IoT*. https://www.profetolocka.com.ar/2024/08/05/guia-completa-integrando-ubidots-con-micropython-para-proyectos-iot-1/

Tonato, C. E., & Sinche, S. (2022). Análisis comparativo entre arquitecturas de sistemas IoT. *Revista de Investigación en Tecnologías de la Información: RITI*, *10*(Extra 21), 55-70.

Toscano, J. (2024). *Implementación de prototipos de monitoreo remoto basados en ESP32 y página web: Implementación de un prototipo de un sistema de monitoreo ambiental de temperatura, humedad, presión y calidad del aire, basado en ESP32 y página web.* http://bibdigital.epn.edu.ec/handle/15000/25599

Tovar, J. E. (2024). *Repotenciación de un módulo interactivo a través de Arduino con comunicación de Radioenlace a IOT y Node Red para integración a Django en la visualización y almacenamiento de datos (SQL SERVER)* [B.S. thesis]. https://dspace.ups.edu.ec/handle/123456789/28131

Ulloa, F., Carrizo, D., & García, L. (2021). Alternativas de comunicación para redes de sensores AMI en Internet de las cosas para escenario energético en ciudades inteligentes. *Ingeniare. Revista chilena de ingeniería*, *29*(1), 158-167. https://doi.org/10.4067/S0718-33052021000100158

Vargas, J. R., Ramos, S. M., & Prieto, A. (2023). *Implementación de un sistema de monitoreo en un terminal portuario multipropósito mediante tecnología basada en industria 4.0* [Thesis, ESPOL. FIEC.]. http://www.dspace.espol.edu.ec/handle/123456789/57287

Venco. (2022, agosto 25). Qué es LoRa, cómo funciona y características principales. *Venco Electrónica*. https://www.vencoel.com/que-es-lora-como-funciona-y-caracteristicas-principales/

Vinicius, B. (2022, marzo 14). *RFID: Qué es y cómo funciona - 2024 - CPCON*. https://www.grupocpcon.com/es/rfid-que-es-y-como-funciona/

Wazid, M., Das, A. K., Shetty, S., Gope, P., & Rodrigues, J. (2020). Security in 5G-Enabled Internet of Things Communication: Issues, Challenges and Future Research Roadmap. *IEEE Access*, *PP*, 1-1. https://doi.org/10.1109/ACCESS.2020.3047895

Zepeda, M. del C. (2023). *El IoT aplicado al monitoreo de pacientes en el servicio de urgencias de unidades hospitalarias del sector público*. http://ri.uaemex.mx/handle/20.500.11799/139488

Glosario

1. **BLE:** Tecnología inalámbrica diseñada para dispositivos de bajo consumo y comunicación de corto alcance.
2. **Cifrado de flash:** Técnica de protección de datos almacenada en flash mediante algoritmos criptográficos.
3. **Conectividad 5G:** Quinta generación de sistema móviles de alta velocidad, baja latencia y capacidad de dispositivos masivos.
4. **Conectividad BLE:** Uso del estándar BLE para transmitir datos de manera eficiente y de bajo consumo.
5. **Coprocesador ULP:** Unidad integrada que proporciona tareas específicas con un consumo mínimo de energía.
6. **ESP32:** Microcontrolador de alto rendimiento con conectividad Wi-Fi y Bluetooth, ideal para aplicaciones de IoT.
7. **FSK:** Modulación de frecuencia de desplazamiento que representa datos mediante la modificación de la frecuencia de una señal portadora.
8. **HTTP:** Protocolo de Transferencia de Híper Texto, un protocolo de red que se utiliza en comunicación cliente-servidor a través de una red de computadoras.
9. Internet de las cosas IoT: Conexión de dispositivos tradicionales a Internet para recoger, analizar y compartir datos.
10. **JSON:** Objeto de notación JavaScript protocolo utilizado para recibir cuerpo en petición y gestionar objeto en respuesta, dado su sintaxis que lo hace fácil para las personas así como para las máquinas.
11. **LoRaWAN:** Sistemas de área amplia y de bajo consumo utilizados en redes de internet.

12. **LPWAN:** Red WAN de baja potencia para comunicarse a distancias grandes con poco poder.
13. **MQTT:** Protocolo de telemedida de mensajes en la forma de colas, ligero para comunicación en redes de baja capacidad con dispositivos de IoT.
14. **Protocolo WAN:** Series de reglas de negocios que permite comunicación de datos en redes de área amplia.
15. **RFID:** Identificación por radiofrecuencia es el uso de ondas de radio para identificar objetos y rastrearlos.
16. **RSA Algoritmo criptográfico:** Protocolo de cifrado de clave pública que garantiza la seguridad en comuniones mediante la dirección de la información en formato cifrado.
17. **SCADA:** Control y adquisición de datos en tiempo real regulados por un sistema que controla procesos industriales asegurando el funcionamiento de sistemas.
18. **SHA Algoritmo de hash seguro:** Tipo de algoritmo que provee la integridad y autenticidad de los datos.

Anexos

Código del nodo sensor emisor 1

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
#define MH_Z19B_RX 25
#define MH_Z19B_TX 34
#define MAX4466_PIN 35
#define BAND 915E6

Adafruit_SSD1306 display(128, 64, &Wire);
int Contador = 0;

void setup() {
 Wire.begin();
 if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) for (;;);
 display.clearDisplay();
 display.setTextColor(WHITE);
 display.setTextSize(1);
 display.setCursor(0, 0);
 display.print("LORA REMITENTE ");
 display.display();
```

```
  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, MH_Z19B_RX, MH_Z19B_TX);

  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DIO0);
  if (!LoRa.begin(BAND)) while (1);
  display.setCursor(0, 10);
  display.print("Inicializacion!");
  display.display();
  delay(2000);
}

void loop() {
  Contador++;
  int co2_ppm = leerCO2MHZ19B();
  int ruido = analogRead(MAX4466_PIN);

  LoRa.beginPacket();
  LoRa.print("CO2: ");
  LoRa.print(co2_ppm);
  LoRa.print(" ppm, Ruido: ");
  LoRa.print(ruido);
  LoRa.print(" Contador: ");
  LoRa.print(Contador);
  LoRa.endPacket();

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("UCSG 2024 EMISOR 1");
  display.setCursor(0, 20);
  display.print("Paquete LoRa enviado.");
  display.setCursor(0, 30);
```

```
  display.print("Contador: ");
  display.setCursor(60, 30);
  display.print(Contador);
  display.setCursor(0, 40);
  display.print("CO2: ");
  display.print(co2_ppm);
  display.print(" ppm");
  display.setCursor(0, 50);
  display.print("Ruido: ");
  display.print(ruido);
  display.display();

  delay(4000);
}

int leerCO2MHZ19B() {
  byte response[9];
  Serial1.write(0xFF);
  Serial1.write(0x01);
  Serial1.write(0x86);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x79);

  delay(10);

  if (Serial1.available() == 9) {
    for (int i = 0; i < 9; i++) {
      response[i] = Serial1.read();
    }
```

```
   if (response[0] == 0xFF && response[1] == 0x86) {
    int co2 = response[2] * 256 + response[3];
    return co2;
   }
 }
 return -1;
}
```

Código del nodo sensor emisor 2

```
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26
#define MH_Z19B_RX 25
#define MH_Z19B_TX 34
#define MAX4466_PIN 35
#define BAND 915E6

Adafruit_SSD1306 display(128, 64, &Wire);
int Contador = 0;

void setup() {
 Wire.begin();
```

```
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) for (;;);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("LORA REMITENTE ");
  display.display();

  Serial.begin(9600);
  Serial1.begin(9600, SERIAL_8N1, MH_Z19B_RX, MH_Z19B_TX);

  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DIO0);
  if (!LoRa.begin(BAND)) while (1);
  display.setCursor(0, 10);
  display.print("Inicializacion!");
  display.display();
  delay(2000);
}

void loop() {
  Contador++;
  int co2_ppm = leerCO2MHZ19B();
  int ruido = analogRead(MAX4466_PIN);

  LoRa.beginPacket();
  LoRa.print("CO2: ");
  LoRa.print(co2_ppm);
  LoRa.print(" ppm, Ruido: ");
  LoRa.print(ruido);
  LoRa.print(" Contador: ");
  LoRa.print(Contador);
  LoRa.endPacket();
```

```
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("UCSG 2024 EMISOR 2");
  display.setCursor(0, 20);
  display.print("Paquete LoRa enviado.");
  display.setCursor(0, 30);
  display.print("Contador: ");
  display.setCursor(60, 30);
  display.print(Contador);
  display.setCursor(0, 40);
  display.print("CO2: ");
  display.print(co2_ppm);
  display.print(" ppm");
  display.setCursor(0, 50);
  display.print("Ruido: ");
  display.print(ruido);
  display.display();

  delay(5000);
}

int leerCO2MHZ19B() {
  byte response[9];
  Serial1.write(0xFF);
  Serial1.write(0x01);
  Serial1.write(0x86);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x00);
  Serial1.write(0x79);
```

```
  delay(10);

  if (Serial1.available() == 9) {
    for (int i = 0; i < 9; i++) {
      response[i] = Serial1.read();
    }
    if (response[0] == 0xFF && response[1] == 0x86) {
      int co2 = response[2] * 256 + response[3];
      return co2;
    }
  }
  return -1;
}
```

Código del nodo receptor wifi

```
#include <SPI.h>
#include <LoRa.h>
#include "UbidotsEsp32Mqtt.h"

const char *UBIDOTS_TOKEN = "xxxxxxxxxxxxx";
const char *WIFI_SSID = "xxxxxxxxxxxx";
const char *WIFI_PASS = "xxxxxxxxxxx";
const char *DEVICE_LABEL = "ESP32";
const char *VARIABLE_LABEL_1 = "Contador 1";
const char *VARIABLE_LABEL_2 = "Contaminacion 1";
const char *VARIABLE_LABEL_3 = "Ruido 1";
const char *VARIABLE_LABEL_4 = "Contaminacion 2";
const char *VARIABLE_LABEL_5 = "Ruido 2";
const char *VARIABLE_LABEL_6 = "Contador 2";

const int PUBLISH_FREQUENCY = 6000;
unsigned long timer;
```

```
uint8_t analogPin = 34;
Ubidots ubidots(UBIDOTS_TOKEN);

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 23
#define DIO0 26

#define BAND 915E6

#if defined(ESP32)
 #define BUTTON_A 15
 #define BUTTON_B 32
 #define BUTTON_C 14
 #define WIRE Wire
#elif defined(ARDUINO_STM32_FEATHER)
 #define BUTTON_A PA15
 #define BUTTON_B PC7
 #define BUTTON_C PC5
 #define WIRE Wire
#else
 #define BUTTON_A  9
 #define BUTTON_B  6
 #define BUTTON_C  5
 #define WIRE Wire
#endif

int Contaminacion11 = 0;
int Ruidos11 = 0;
int contadorvalores1 = 0;
```

```
int contadorvalores11 = 0;
int Contaminaciones22 = 0;
int Ruidos22 = 0;
int contadorvalores22 = 0;
int contadorvalores2 = 0;

Adafruit_SSD1306 display = Adafruit_SSD1306(128, 64, &WIRE);

String LoRaData;

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 setup() {
 pinMode(OLED_RST, OUTPUT);
 digitalWrite(OLED_RST, LOW);
 delay(20);
 digitalWrite(OLED_RST, HIGH);

 Wire.begin(OLED_SDA, OLED_SCL);
 if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) {
  Serial.println(F("Asignación de SSD1306 fallo"));
  for(;;);
 }

 display.clearDisplay();
 display.setTextColor(WHITE);
 display.setTextSize(1);
 display.setCursor(0,0);
```

```
  display.print("RECEPTOR LORA ");
  display.display();

  Serial.begin(9600);
  Serial.println("Prueba receptor LoRa");

  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DIO0);

  if (!LoRa.begin(BAND)) {
   Serial.println("Starting LoRa failed!");
   while (1);
  }
  Serial.println("¡Error al iniciar LoRa!");
  display.setCursor(0,10);
  display.println("LoRa Inicializando!");
  display.display();
  delay(1000);
  ubidots.connectToWifi(WIFI_SSID, WIFI_PASS);
  ubidots.setCallback(callback);
  ubidots.setup();
  ubidots.reconnect();
  timer = millis();
}

void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
   Serial.print("paquete recibido ");
   while (LoRa.available()) {
    LoRaData = LoRa.readString();
    Serial.print(LoRaData);
   }

   String Contaminacion1 = LoRaData.substring(0,2);
   float Contaminaciones1 = Contaminacion1.toFloat();
```

```
String Ruido1 = LoRaData.substring(3,5);
float Ruidos1 = Ruido1.toFloat();

String Contaminacion2 = LoRaData.substring(7,9);
float Contaminaciones2 = Contaminacion2.toFloat();

String Ruido2 = LoRaData.substring(10,12);
float Ruidos2 = Ruido2.toFloat();

int rssi = LoRa.packetRssi();
Serial.print(" with RSSI ");
Serial.println(rssi);

if (Contaminaciones1 && Ruidos1 > 0) {
 contadorvalores1++;
 display.clearDisplay();
 display.setCursor(0,0);
 display.print("UCSG 2024 RECEPTOR 1");
 display.setCursor(0,20);
 display.print("SENSOR DE Contaminacion 1");
 display.setCursor(0,30);
 display.print("Mensaje #:");
 display.setCursor(75,30);
 display.print(contadorvalores1);
 display.setCursor(0,40);
 display.print("Te1: ");
 display.setCursor(25,40);
 display.print(Contaminaciones1);
 display.setCursor(60,40);
 display.print("Hu1: ");
 display.setCursor(85,40);
 display.print(Ruidos1);
 display.setCursor(0,50);
 display.print("RSSI:");
 display.setCursor(30,50);
```

```
  display.print(rssi);
  display.display();
  Contaminacion11 = Contaminaciones1;
  Ruidos11 = Ruidos1;
  contadorvalores11 = contadorvalores1;
 }

 if (Contaminaciones2 && Ruidos2 > 0) {
  contadorvalores2++;
  display.clearDisplay();
  display.setCursor(0,0);
  display.print("UCSG 2024 RECEPTOR");
  display.setCursor(0,20);
  display.print("SENSOR DE Contaminacion 2");
  display.setCursor(0,30);
  display.print("Mensaje #:");
  display.setCursor(75,30);
  display.print(contadorvalores2);
  display.setCursor(0,40);
  display.print("Te2: ");
  display.setCursor(25,40);
  display.print(Contaminaciones2);
  display.setCursor(60,40);
  display.print("Hu2: ");
  display.setCursor(85,40);
  display.print(Ruidos2);
  display.setCursor(0,50);
  display.print("RSSI:");
  display.setCursor(30,50);
  display.print(rssi);
  display.display();
  Contaminaciones22 = Contaminaciones2;
  Ruidos22 = Ruidos2;
  contadorvalores22 = contadorvalores2;
 }
```

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

  if (abs(millis() - timer) > PUBLISH_FREQUENCY) {
   if (Contaminaciones11 && Contaminaciones22 > 0) {
    ubidots.add(VARIABLE_LABEL_1, contadorvalores11);
    ubidots.add(VARIABLE_LABEL_2, Contaminaciones11);
    ubidots.add(VARIABLE_LABEL_3, Ruidos11);
    ubidots.add(VARIABLE_LABEL_4, Contaminaciones22);
    ubidots.add(VARIABLE_LABEL_5, Ruidos22);
    ubidots.add(VARIABLE_LABEL_6, contadorvalores22);
    ubidots.publish(DEVICE_LABEL);
    timer = millis();
   }
  }
 }
 ubidots.loop();
}
```

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Gil Cevallos, Mario Javier** con C.C: **0925805525** autor del Trabajo de Integración **Curricular Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil,** previo a la obtención del título de **Magíster en Telecomunicaciones,** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de integración curricular para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de integración curricular, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 13 de marzo del 2025

_____

Gil Cevallos, Mario Javier

C.C: 0925805525

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE INTEGRACIÓN CURRICULAR			
TÍTULO Y SUBTÍTULO:	Diseño de un Sistema Iot para el monitoreo de interferencia y ruido mediante el uso de una red de nodos sensores con tecnología LoRaWAN y ESP32 en la ciudad de Guayaquil.		
AUTOR(ES)	Gil Cevallos, Mario Javier		
REVISOR(ES)/TUTOR(ES)	Bohórquez Heras, Diana Carolina; Ubilla González, Ricardo Xavier; Bohórquez Escobar, Celso Bayardo.		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil.		
FACULTAD:	Sistema de Posgrado		
CARRERA:	Maestría en Telecomunicaciones		
TITULO OBTENIDO:	Magister en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	13 de marzo del 2025	**No. DE PÁGINAS:**	104 p.
ÁREAS TEMÁTICAS:	Redes de nodos sensores, microcontroladores		
PALABRAS CLAVES/ KEYWORDS:	ESP32, Lora, IoT, Arduino IDE, sensores, monitoreo.		
El presente trabajo de integración curricular se basa en el diseño de un sistema IoT para el monitoreo de interferencia y ruido en la ciudad de Guayaquil, utilizando una red de nodos sensores con tecnología LoRaWAN y ESP32. Mediante el desarrollo de la interfaz principal para el monitoreo de interferencia y ruido en la ciudad de Guayaquil mediante el uso del entorno de programación de Arduino IDE. También se realiza el diseño de un entorno gráfico utilizando el software LabVIEW para gestionar las variables recibidas en la plataforma IoT Ubidots y a su vez se evalúa el desempeño, viabilidad y estimar el costo del diseño de una red de sensores inalámbricos utilizando el módulo ESP32. Este estudio establecerá un sistema IoT con una red de nodos sensores con tecnología LoRaWAN con ESP32, que permitirá el monitoreo constante de la contaminación y ruido en la ciudad de Guayaquil, brindando una información con mayor precisión, que servirá para tomar decisiones y ejecutar medidas públicas idóneas para reducir dichos problemas ambientales. La metodología utilizada es la bibliográfica de la literatura existente sobre sistemas IoT para el monitoreo del entorno, nodos sensores, tecnologías LoRaWAN y ESP32.			
ADJUNTO PDF:	☒ SI	☐ NO	
CONTACTO CON AUTOR/ES:	**Teléfono:** +593 958929084	E-mail: mario.gil@cu.ucsg.edu.ec	
CONTACTO CON LA INSTITUCIÓN: COORDINADOR DEL PROCESO DE UTE	**Nombre:** Ing. Bohórquez Escobar, Celso Bayardo PHD.		
	Teléfono: +593- 995147293		
	E-mail: celso.bohorquez@cu.ucsg.edu.ec		
SECCIÓN PARA USO DE BIBLIOTECA			
Nº. DE REGISTRO (en base a datos):			
Nº. DE CLASIFICACIÓN:			
DIRECCIÓN URL (tesis en la web):			