ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Diseño e implementación de un prototipo de sistema de gestión de servicios básicos para una ciudad inteligente basado en procesamiento de eventos complejos

Yanez Morocho, Luis Roberto

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Alulema Flores, Darwin Omar, PhD.

18 de agosto del 2023

Copyleaks

Plagiarism and AI Content Detection Report

Trabajo_Titulación_Yanez_Luis (1).pdf

Firmado digitalmente por: DARWIN OMAR ALULEMA FLORES

Scan details

Scan time:	Total Pages:	Total Words:
August 18th, 2023 at 14:50 UTC	82	20383

Plagiarism Detection

Types of plagiarism		Words
Identical	0.1%	27
Minor Changes	0.2%	39
Paraphrased	0.5%	92
Omitted Words	0%	0

AI Content Detection

Text coverage		Words
AI text	0%	0
Human text	100%	20383

Learn more

Plagiarism Results: (3)

T-ESPE-052383.pdf — 0.5%

https://repositorio.espe.edu.ec/bitstream/21000/31221/1/t-e...

David Tumbaco

1 “Implementación de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas” Cha...

Procesamiento de eventos complejos en tiempo real — 0.2%

https://ciberseguridad.com/herramientas/procesamiento-ev...

ciberseg1922

Saltar al contenido Ciberseguridad Noticias de ciberseguridad, ciberata...

Seguidor de Línea TCRT5000 Óptico Infrarrojo (Mod... — 0.1%

https://tiendadeelectronica.mx/tienda/seguidor-de-linea-tcrt...

Ir a la navegación Ir al contenido Buscar por: Buscar Inicio Tienda ...

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: **"Diseño e implementación de un prototipo de sistema de gestión de servicios básicos para una ciudad inteligente basado en procesamiento de eventos complejos"** fue realizado por el señor **Yanez Morocho, Luis Roberto;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 18 de agosto de 2023

Firma:

Ing. Alulema Flores, Darwin Omar, PhD.

C. C. 1002493334

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Auditoría

Yo, **Yanez Morocho Luis Roberto** con cédula de ciudadanía N° 1723012561, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Diseño e implementación de un prototipo de sistema de gestión de servicios básicos para una ciudad inteligente basado en procesamiento de eventos complejos",** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 18 de agosto del 2023

Yanez Morocho Luis Roberto

C. C. 1723012561

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Yo, **Yanez Morocho Luis Roberto** con cédula de ciudadanía N° 1723012561, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Diseño e implementación de un prototipo de sistema de gestión de servicios básicos para una ciudad inteligente basado en procesamiento de eventos complejos",** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 18 de agosto del 2023

Yanez Morocho Luis Roberto

C. C. 1723012561

Dedicatoria

Esta tesis está dedicada a:

A dios y a la virgen del quinche por permitirme culminar una de las etapas más importantes en mi vida sin sus bendiciones no hubiera sido posible cumplir este sueño.

A mi madre y mi padre por su apoyo incondicional, consejos, sacrificio y demostración de superación frente a las adversidades para entender que todo es posible.

A mis hermanos Danilo y Mauricio por ser una fuente de inspiración y enseñanzas desde pequeño hasta el día de hoy.

Esto es para ustedes familia, ¡Lo logramos!

Agradecimiento

Agradezco a Dios y a la Virgen del Quinche por haberme guiado, cuidado y hacerme comprender que en la vida los caminos difíciles son los que valen la pena, solo ellos saben todas las cosas que he sacrificado y he dejado de lado por cumplir este sueño. Los tiempos de dios son perfectos.

Mil gracias madre por tu apoyo incondicional por tener esa ilusión, ese sueño de verme crecer siempre por no abandonarme nunca y estar siempre para mí no me alcanzará esta vida ni la otra para agradecerte todo lo que has hecho por mí.

Gracias padre por enseñarme a forjar un carácter, para comprender que en la vida nada viene fácil y que todo esfuerzo tiene su recompensa.

Gracias a mi hermano Danilo por estar conmigo siempre a lo largo de esta etapa, por ayudarme en muchas situaciones de mi vida universitaria y darme ánimos cuando estuve a punto de dejar este logro.

Gracias a mi hermano Mauricio por sus enseñanzas que a través de sus vivencias y logros he podido tomar lo bueno y aprender de lo malo.

Finalmente, un agradecimiento a mis amigos que compartimos tantas vivencias a lo largo de esta etapa. A la universidad y sus docentes por brindarme la oportunidad de ser parte de esta gran institución y sobre todo a mi tutor Ing. Darwin Alulema por sus conocimientos, paciencia y confianza para el desarrollo del presente proyecto.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

El desarrollo tecnológico en las ciudades urbanísticas ha ido evolucionando debido al crecimiento económico, al transporte, servicios básicos como electricidad y agua potable todo esto para beneficio y mejora de calidad de vida de los habitantes. Por otra parte, la crisis sanitaria provocada por el COVID-19 ha evidenciado la necesidad del desarrollo de ciudades inteligentes que se encuentren disponibles los 365 días del año. Bajo estas consideraciones surge la propuesta del desarrollo de unos prototipos para la gestión de servicios básicos basado en el procesamiento de eventos complejos, este proceso no se lo realiza de manera local, sino que es realizado en la nube para que pueda ser configurado y coordinado con servidores remotos como el correo electrónico y más herramientas esenciales para el monitoreo de estos prototipos. La arquitectura del sistema consta de tres capas: una capa física contenedora de los tres prototipos encargados de monitorear los valores de consumo de agua, electricidad y nivel de residuos estos envían los datos hacia un gateway central mediante una tecnología inalámbrica LoRa multipunto. En la capa lógica contiene el procesamiento de eventos complejos para el intercambio de información con los prototipos y una capa de aplicación para la visualización en tiempo real de cada prototipo desde cualquier parte del mundo. Finalmente se realiza el escenario de pruebas mediante una maqueta para comprobar el correcto funcionamiento de la arquitectura en la cual el sistema se ha evaluado con pruebas de funcionalidad, carga y usabilidad.

Palabras clave: Smart city, Procesamiento de Eventos Complejos, Modulación Long Range, Internet de las cosas.

Abstract

Technological development in urban cities has evolved due to economic growth, transportation, basic services such as electricity and drinking water, all for the benefit and improvement of the quality of life of the inhabitants. On the other hand, the health crisis caused by COVID-19 has highlighted the need to develop smart cities that are available 365 days a year. Under these considerations, the proposal for the development of some prototypes for the management of basic services based on the processing of complex events arises, this process is not carried out locally, but is carried out in the cloud so that it can be configured and coordinated with remote servers such as email and more essential tools for monitoring these prototypes. The system architecture consists of three layers: a physical layer that contains the three prototypes in charge of monitoring the values of water consumption, electricity and level of waste, these send the data to a central gateway through multipoint LoRa wireless technology. The logical layer contains the processing of complex events for the exchange of information with the prototypes and an application layer for real-time viewing of each prototype from anywhere in the world. Finally, the test scenario is carried out by means of a model to verify the correct functioning of the architecture in which the system has been evaluated with functionality, load and usability tests.

Keywords: Smart city, Complex Event Processing, Long Range Modulation, Internet of things.

Capítulo I. Marco Metodológico

Antecedentes

El aumento de la población urbanística a lo largo del mundo va de la mano con el crecimiento económico, la movilidad, el transporte, servicios básicos y el incremento de la calidad de vida de los ciudadanos por lo que siempre se busca la eficiencia en la gestión de los recursos, acelerar la urbanización global mediante el desarrollo de soluciones inteligentes utilizando conceptos como el internet de las cosas en las Smart Cities. Mediante un estudio realizado por la ONU en el año 2016 a nivel mundial el 54,6% de las personas viven en ciudades urbanas por lo que se tiene pronosticado que para el 2050 con el crecimiento de las ciudades y la población aumentará aproximadamente un 20% para vivir en ciudades urbanas donde estadísticamente el 64,1% resida en regiones con condiciones en crecimiento mientras que el 85,9% resida en ciudades y regiones de primer mundo (Bouskela, Casseb, & Bassi, 2016).

En una ciudad inteligente la transmisión y el almacenamiento de los datos provenientes de los sensores utilizados para medir el consumo eléctrico, consumo de agua potable, llenado y ubicación de tachos de basura para la gestión y administración de la información se utiliza redes LPWAN como son Sigfox y LoRa se tomará en cuenta los siguientes trabajos de Borja (Borja, 2021), (Ayala, 2022) y (Castillo P. , 2018).

Por otra parte, para investigaciones relacionadas con la elaboración de prototipos de consumo de agua se basará en (Castillo & Murillo, 2017) y (Ayala, 2022) mientras que para prototipos de consumo eléctrico y monitoreo de desechos de tachos de basura se basará en (Morocho & Quinapanta, 2016), (Zambrano, 2019) y (Acosta, 2020).

Para el procesamiento y análisis de la información debido a cambios que se dan en tiempo real se utiliza el procesamiento de eventos complejos el cual es la agrupación de varios eventos simples previamente procesados para su análisis con el objetivo de detectar una situación relevante para el entorno como en (Jay, Behrad, & Kao, 2015) y (Pilco & Mendez, 2021) para con esto determinar las posibles acciones que ejercerán los controladores, los usuarios que gestionen el sistema de la ciudad inteligente o la ruta de los camiones al momento de la recolección de los desechos en la ciudad.

Justificación e importancia

En la actualidad para verificar el consumo de agua potable en cada hogar en la ciudad de Quito de forma habitual un trabajador se acerca al medidor, registra la lectura, y emite el comprobante pago de igual manera para el consumo de energía eléctrica un trabajador se acerca al medidor para verificar la lectura y después de un cierto tiempo se recibe en cada hogar el comprobante de pago. Por otra parte, para la recolección de basura existen contenedores que se encuentran desbordados y el servicio de recolección de basura no pasa por esa ruta ya sea por daño del carro recolector o por algún otro imprevisto, mientras que existen otros carros recolectores que realizan rutas innecesarias en lugares donde los contenedores se encuentran vacíos por lo que podrían cubrir lugares o zonas donde sea necesario la recolección de basura para así reducir costos y optimizar recursos. La crisis sanitaria provocada por la pandemia del COVID-19 ha puesto de manifiesto la necesidad de acelerar el desarrollo de ciudades inteligentes en todo el mundo, para que este tipo de ciudades puedan ser aliadas en la lucha contra la pandemia. Una ciudad inteligente implica que estará disponible las 24 horas del día los 365 días del año. En una ciudad inteligente se espera que lo largo del año se encuentre apta para brindar distintos servicios por medio del municipio de la ciudad o entidades encargadas de los servicios ya que le economía tanto en el

día como en la noche ha incrementado para lo que es el consumo de la energía eléctrica, agua potable y recolección de los residuos. Esto viene dado tanto para ciudades y países en crecimiento como del primer mundo (Lagos & Benavides , 2021). El tráfico vehicular en las ciudades muchas veces es causado por conductores buscando lugares donde estacionarse de igual manera viene dado por los camiones de recolección de basura ya que realizan rutas en lugares o contenedores que se encuentran vacíos significando tiempo perdido, mayor consumo de combustible y aumento de emisiones de CO2. (Allauca Fajardo, 2020)

Para lograr el desarrollo de una ciudad inteligente se debe contar con el despliegue de una red de sensores que tenga la posibilidad de conectarse con el Internet para permitir el intercambio de información este intercambio se lo da gracias a una propuesta interesante de redes LPWAN que tiene como características principales ser de largo alcance, baja velocidad de transmisión y bajo consumo de energía por lo que Sigfox y LoRa han desplegado una red inalámbrica a nivel mundial, en el Ecuador tiene una amplia cobertura con lo que nos permitirá integrar los múltiples prototipos que vayan surgiendo. Estos prototipos al emplear servicios web y procesamiento de eventos complejos (CEP) nos van a permitir que sea un esquema escalable ya que permitirá integrar con otras tecnologías que existan o vayan surgiendo en el tiempo con lo que este proyecto generará un prototipo que puede llegar a ser replicado a gran escala ya que tendrá una arquitectura genérica que pueda ser usada por otras cuidades. Esta propuesta va ser importante porque va aportar al Plan Nacional de Desarrollo 2021-2025 “Creando oportunidades’’ dentro del eje económico propone impulsar un sistema económico con reglas claras con el cual quiere promover un entorno de negocios que atraiga las inversiones, la asociación público-privada y la generación de empleos con lo que con esta propuesta se quiere dinamizar las economías de las ciudades al incorporar una propuesta que

permitiría que otras ciudades puedan replicar los sensores, servicios web y la arquitectura realizada.

Alcance

El prototipo de sistema de gestión está aplicado a una Smart city el cual tiene la gestión de tres ámbitos como son la energía eléctrica, agua potable y recolección de residuos. Este prototipo estará compuesto de una red LPWAN LoRa se conecta a internet obteniendo todos los datos de estos servicios básicos para después ser leídos mediante servicios disponibles en la nube y puedan ser utilizados por los usuarios a través de la PC o dispositivos móviles. Este sistema está compuesto de sensores los cuales se encargarán de recopilar la información de los servicios básicos antes mencionados por lo que los sensores se encargarán de recopilar el nivel de basura en los contenedores, el caudal de agua, el consumo eléctrico que se vaya dando en el transcurso del día para posteriormente todos estos datos ser enviados a la nube de una red LPWAN. Presentará indicadores led en una maqueta para verificar si se encuentran llenos los tachos de basura, estos indicadores se encargarán de facilitar la visualización de los tachos para mejorar el tiempo, tráfico y reducir costos operativos que pueden presentarse por partes de los camiones recolectores de basura. Estos módulos a desarrollar estarán realizados mediante el controlador ESP32 con conectividad a una red Lora esto debido a la variedad de sensores disponibles y la economicidad en el mercado de los mismos ya que se trata de diseñar módulos lo más pequeños, portables y económicos.

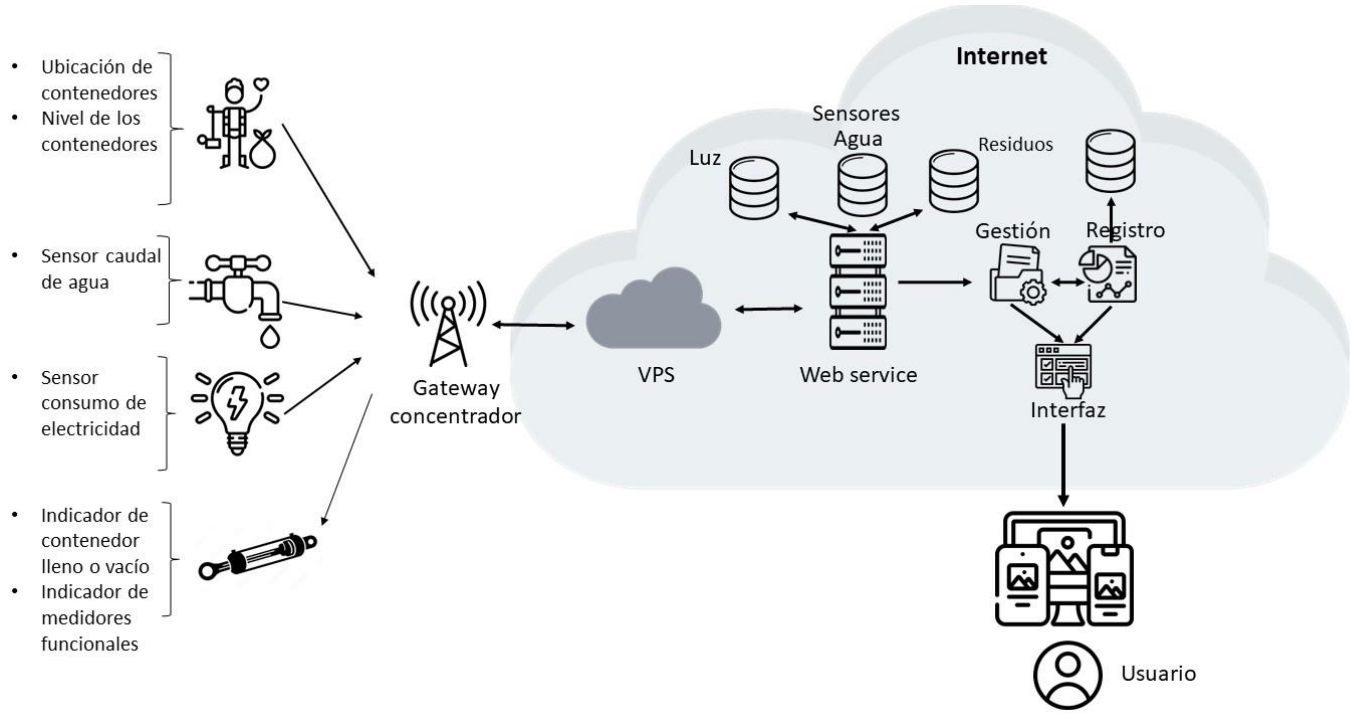
Los datos que vayan recolectando estos tres nodos serán transmitidos a un nodo central mediante conectividad LoRa en la cual este nodo central se encargará de subir a la nube todos los datos mediante la tecnología MQTT a un broker.

Todas las señales obtenidas de los sensores van hacer coordinados a un servicio web con el cual para la coordinación y toma de decisiones se implementará un algoritmo mediante el procesamiento de eventos complejos (CEP) el cual permitirá procesar, analizar y correlacionar en tiempo real, reduciendo la latencia en la toma de decisiones por parte de los controladores o los usuarios que estén supervisando esta ciudad inteligente ante las situaciones o eventos más relevantes para así filtrar los eventos intranscendentes tomados por los módulos de esta ciudad inteligente. Para saber cómo se monitorearán las variables del sistema esto dependerá de los datos que se obtengan de los prototipos y las situaciones que se simulen en la maqueta para ser presentados en una interfaz como se describe en la parte de actividades en la evaluación del sistema.

Todos los datos obtenidos de los prototipos estarán en la nube como se muestra en la figura 1, por lo que para tenerlo en nuestro sistema estará diseñado por algunas bases de datos para los sensores de cada prototipo, las cuales tendrán información como nombre del sensor, tipo, fecha y hora del mensaje recibido o enviado. En este caso se tendrán dos sistemas los cuales serán encargados de la gestión y registro de todos los servicios básicos antes mencionados y por último la interfaz para observar todos los datos como, por ejemplo, monitorear todas las variables recolectadas por los sensores, la ubicación de los medidores de electricidad, agua o donde se encuentran los contenedores de basura etc. Para que con esto los usuarios desde su computador o celular puedan acceder a la información. Este sistema será probado mediante una maqueta donde se simule situaciones reales que suceden en una ciudad en lo que tiene que ver con el consumo de agua potable, energía y recolección de residuos con lo cual verificaremos la toma de datos, funcionamiento del hardware, funcionamiento del sistema y la usabilidad de la interfaz para el despliegue de la información.

Figura 1

Arquitectura del sistema



Objetivos

Objetivo General

Diseñar e implementar un prototipo de sistema de gestión de servicios básicos para ciudades inteligentes basado en procesamiento de eventos complejos.

Objetivos Específicos

- Investigar los conceptos teóricos de: Web service, Procesamiento de eventos complejos, Smart City, Redes LPWAN.
- Diseñar una arquitectura escalable para la gestión de los servicios de una ciudad inteligente.
- Procesar los eventos complejos para conseguir una respuesta eficiente del sistema mediante una lógica de control.

- Diseñar un prototipo de hardware para monitorizar la energía eléctrica, uno de agua potable y uno de recolección de residuos, con capacidades de conexión a una red Lora.
- Evaluar el funcionamiento del hardware, usabilidad de la interfaz y rendimiento del sistema, mediante una maqueta, pruebas de usabilidad y pruebas de carga.

Estado del Arte

Metodología

Una vez desarrollado la justificación, alcance y objetivos del proyecto es fundamental realizar el análisis de trabajos desempeñados hasta la presente fecha, abordando mediante el estado del arte estudios que se encuentre dentro de las áreas de interés del proyecto, en este caso, sistemas de gestión de servicios básicos en aplicaciones IoT, procesamiento de eventos complejos y dispositivos con conexión a LoRa. Con lo cual se busca investigar a nivel mundial posibles soluciones o avances en investigaciones que tengan que ver con las áreas del proyecto, además conocer en qué momento fueron realizadas estas investigaciones. Para lo cual se elige realizar mediante un Mapeo Sistemático de la Literatura (SMS) la investigación de las áreas de interés con lo que ayudaría a encontrar la primicia de que partes no están siendo realizadas y sean vitales resolverlas a lo largo del proyecto.

Mapeo Sistemático de la Literatura. Según Gough et al., (2012) dice que el mapeo sistemático no necesariamente requiera encontrar las respuestas más acertadas a un problema o alguna pregunta en concreto, sino que indaga de manera sistemática todo lo que se ha publicado sobre el tema tomando en cuenta ciertos factores como la amplitud y como asido abordado el tema, pero no es necesario evaluar la metodología que han llevado los autores de cada trabajo. (Gough & Oliver, 2012)

Planteamiento de preguntas. Como se observa en la tabla 1 para investigar sobre como se ha desarrollado el tema con estudios similares desglozando el objetivo principal del

proyecto como son sistemas de gestión de servicios básicos y el procesamiento de eventos complejos.

Tabla 1

Preguntas de investigación planteadas para el Mapeo Sistemático

Pregunta de investigación	Motivación
RQ1: ¿Cuáles son las investigaciones desarrolladas alrededor de sistemas de gestión de servicios básicos para una smart city?	Conocer investigaciones que traten sobre el monitoreo del consumo luz, agua y recolección de residuos
RQ2:¿ Qué arquitecturas son las más utilizadas para el desarrollo de procesamiento de eventos complejos?	Conocer que arquitecturas pueden ser utilizadas para el desarrollo de procesamiento de eventos complejos.
RQ3:¿Qué tipo de servicios web existen para el desarrollo de sistemas de IoT ?	Conocer los servicio web disponibles para el desarrollo de sistemas de IoT
RQ4:¿Qué tecnologías y/o dispositivos existen para realizar el monitoreo de servicios básicos?	Conocer la base tecnológica para el desarrollo de prototipos de hardware.

Cadenas de búsqueda. Se utilizó la plataforma de búsqueda Web of Science para la búsqueda y recopilación de la información debido a que es una base de datos muy completa con investigaciones muy confiables alrededor del mundo. En la tabla 2 se muestra los términos claves utilizados en base a las preguntas planteadas anteriormente para colocarlos en el motor de búsqueda.

Tabla 2

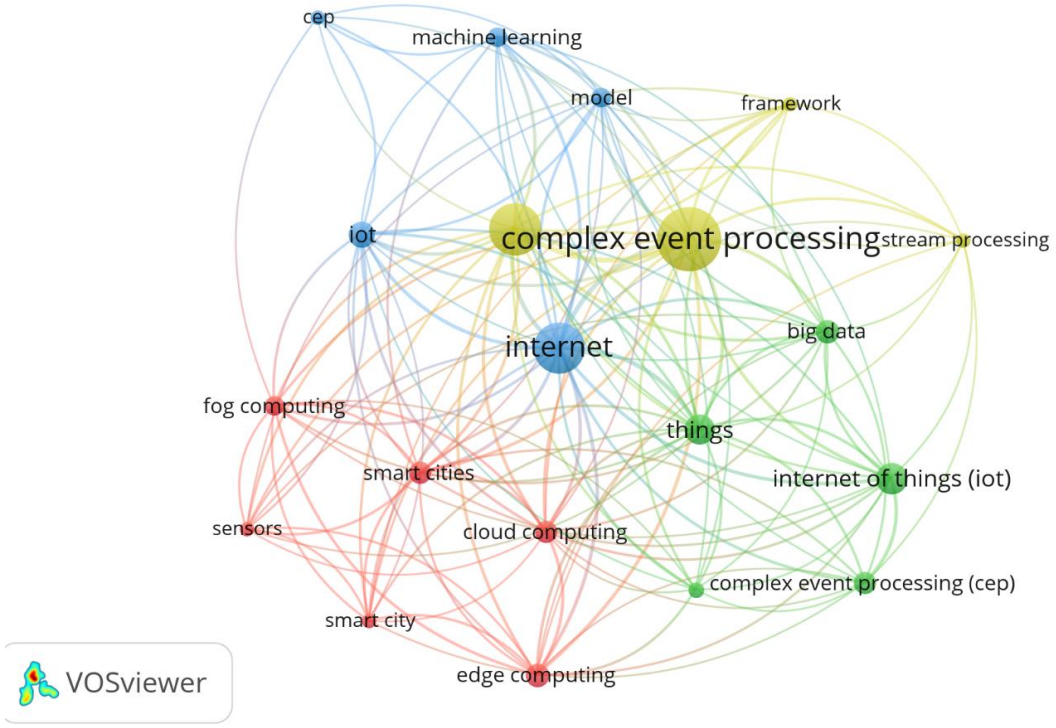
Cadenas de búsqueda

Término clave	Términos alternativos
RQ1: Sistemas de gestión de servicios básicos o Smart Cities	“Basic services management systems” OR “Services in Smart Cities”
RQ2: Procesamiento de eventos complejos	“Complex event processing” OR “CEP”
RQ3: Servicios Web	“Web Service” OR Rest
RQ4: Prototipos servicios básicos	“Prototip basic services” OR “Hardware for basic services”

A continuación, se muestra mediante el software VOSviewer la presentación en mapas de densidad de las palabras claves buscadas en Web of Science. En la figura 2 se muestra las palabras clave que han sido mencionados como mínimo 5 veces del listado de documentos obtenidos en la búsqueda por lo que se obtuvo en total de 18 palabras por lo que se puede observar que la palabra Procesamiento de eventos complejos fue las más mencionada y que tiene un fuerte enlace con las palabras como internet, IoT y Smart Cities.

Figura 2

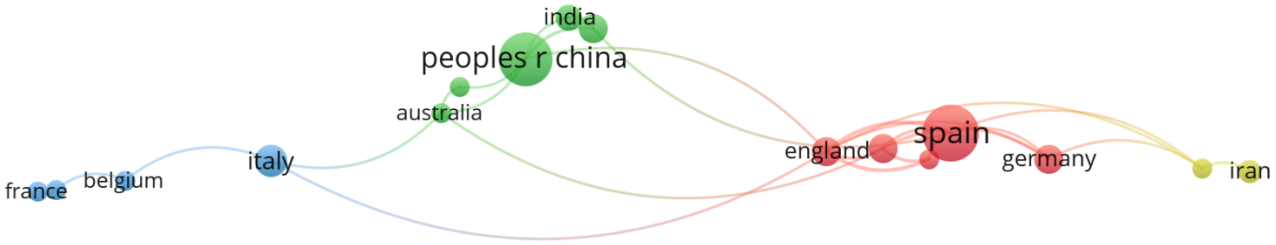
Densidad de búsqueda por palabras



En la figura 3 se tiene la presentación del mapa de densidad por países según los aportes en el tema de investigación a nivel mundial siendo China y España los de mayor número de publicaciones seguidos por Inglaterra, Alemania, Australia e Italia.

Figura 3

Densidad de búsqueda por países



Conclusiones. Se han obtenido respuesta a las preguntas planteadas anteriormente mediante el SMS, con su debida sustentación por artículos o trabajos de investigación que están dentro del área de interés del desarrollo del proyecto.

RQ1: ¿Cuáles son las investigaciones desarrolladas alrededor de sistemas de gestión de servicios básicos para una smart city?

Después de realizar un estudio de los trabajos recopilados sobre sistemas de gestión de servicios básicos en una Smart city, lo cual indica que estos tipos de sistemas son muy usados para tener una función inteligente en la administración tanto para la recolección de residuos como para monitorear distintas variables para poder brindar una mejor condición de vida para los habitantes (Muayad, 2021), el desarrollo de sistemas de gestión de servicios básicos han ido evolucionando cada vez más hasta poder monitorear la calidad del agua y ser monitoreado en tiempo real con lo que puede ser integrado con el sistema de gestión de agua urbana para así obtener una mayor eficiencia (Chen Y. , 2018). Además, para implementar estos sistemas inteligentes de medición se tebe tomar en cuenta 5 aspectos: la captación de los datos, el pretratamiento de los datos para obtener los consumos, la seguridad física de los dispositivos electrónicos, el almacenamiento y visualización de los datos (Fuentes , 2020).

RQ2:¿Qué arquitecturas son las más utilizadas para el desarrollo de procesamiento de eventos complejos?

El procesamiento de eventos complejos se infiere a partir de patrones de dinstintos eventos simples que se puedan dar según la información que se vaya dando en el espacio y en el tiempo, los cuales se basan su programación en lenguajes de procesamiento de eventos (EPL), los eventos complejos tiene características espaciales ya que tiene que detectar el área de trabajo en el que se desea aplicar el procesamiento de eventos por ejemplo se mide la calidad de aire mientras que para un ciudadano que camina en el día el estado actual de contaminación a su paso es fundamental mientras que para el ministerio de salud es necesario

el promedio de días contaminados a lo largo de un año por lo cual como vemos es importante detectar el área afectada por un evento complejo (Khazael & Tabatabaee, 2023). Para el desarrollo de la arquitectura del procesamiento de eventos complejos se utiliza la tecnología SND que significa Redes Definidas por Software que es básicamente donde el hardware se encuentra separado del software, una de las ventajas es que en lugar de programar manualmente varios dispositivos de hardware los desarrolladores pueden programar un controlador realizado en software para realizar las mismas funcionalidades (Clarke & Khazael, 2021).

RQ3:¿Qué tipo de servicios web existen para el desarrollo de sistemas de IoT ?

En el desarrollo de sistemas IoT es necesario la utilización de distintos servicios web para coordinar la información que se comparte mediante la conexión de dispositivos a través de redes cableadas e inalámbricas. Mediante Node-Red se crea el sistema y los usuarios mediante su interfaz de programación simple pueden configurar parámetros ambientales y así personalizar la interfaz según a las conveniencias del usuario (Sinchangreed & Egkarin, 2022).

También por otra parte se tiene servicios de cloud computing los cuales son encargados de permitirnos el acceso remoto al software, guardar y procesar los datos. Como por ejemplo un modelo de evaluación de computación en la nube en la cual se analiza datos de poblaciones (Hongqiang & Xinxin , 2020). Por otra parte, se tiene los servicios web cognitivos en los cuales a una maquina se le brindan capacidades de adquirir la información y procesarla ya sea mediante audio, lenguaje o imágenes igual que los seres humanos, como por ejemplo detectar la cantidad de basura mediante cámaras de seguridad para saber cuándo se realice la recolección de residuos (Coccoli & Fusco, 2021).

RQ4:¿Qué tecnologías y/o dispositivos existen para realizar el monitoreo de servicios básicos?

Para el diseño de un sistema de monitoreo del hogar en la que integre, electrodomésticos, prevención de desastres e instalaciones de vigilancia se ocupó servicios de cloud computing y servicios para conectar hardware y software como node red y MQTT para la comunicación e intercambio de datos mediante el método de publicación/suscripción, en este caso se realizó el control de interruptores de luces, aire acondicionado, extractores, puertas y la activación o desactivación de la alarma (Chen & Wen-Chiung , 2020). Por otra parte, en la actualidad el uso de medidores inteligentes para el monitoreo en tiempo real de la energía eléctrica está tomando una creciente y expansión exponencial en los cuales se puede ocupar distintas tecnologías inalámbricas como por ejemplo las redes LPWAN ya que tiene un gran alcance y un bajo consumo de energía que permite su instalación en cualquier tipo de lugares (Sánchez & Cano, 2021).

Revisión Sistemática de la Literatura. Para tener un estudio más completo de los trabajos revisados previamente se realiza una Systematic Literature Review (SLR), la cual es una técnica que evalúa, identifica y analiza trabajos investigativos para obtener el estado del arte del proyecto lo que le diferencia es que aborda un tema en específico mientras que el SMS obtiene una visión más amplia de la información (García, 2020). En la Tabla 3 se muestra las preguntas realizadas para la metodología SLR.

Tabla 3

Planteamiento de preguntas para una SLR

Pregunta de investigación	Motivación
RQ1: ¿Qué tecnologías, técnicas y herramientas se utilizan para el desarrollo de prototipos para monitorear los servicios básicos con funcionalidades IoT?	Conocer qué tecnologías, técnicas o herramientas existen para poder desarrollar prototipos de hardware para monitor orea los servicios básicos.
RQ2: ¿Qué tecnologías, técnicas y herramientas se usan para la gestión de procesamiento de eventos complejos?	Conocer que tipos de tecnologías, técnicas o herramientas se usan para la gestión de procesamiento de eventos complejos.
RQ3: ¿Qué tecnologías y/o técnicas son utilizadas para realizar Benchmarking de un sistema de gestión de servicios básicos?	Conocer qué tecnologías, técnicas o herramientas existen para realizar pruebas de rendimiento para poder medir y validar la funcionalidad de un sistema de gestión de servicios básicos.

Cadenas de búsqueda. En la tabla 4 se muestra las palabras claves ingresadas en el buscador de Web of Science.

Tabla 4

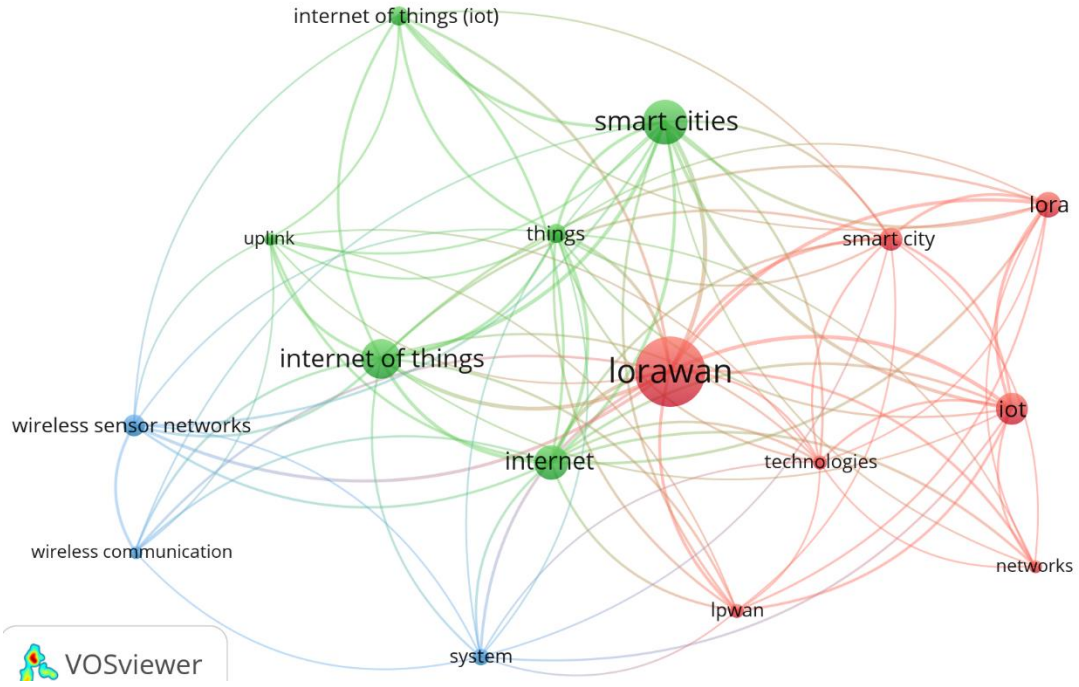
Términos clave utilizados en la SLR

Término clave	Términos alternativos
RQ1: Consumo de agua IoT OR consumo de energía IoT	“IoT water consuming” OR “IoT energy consuming”
RQ2: Procesamiento de eventos complejos OR LoRa Or Protocolos	“Complex event processing” OR “Protocol” “LoRa”
RQ3: Pruebas de rendimiento	“Benchmark” OR “Rest”

Como se muestra en la figura 4 mediante el software VOSviewer se tiene la densidad según las palabras claves con más búsquedas en publicaciones científicas relacionado al tema se tiene IoT, energy efficiency, protocol, optimization, Wireless sensor networks.

Figura 4

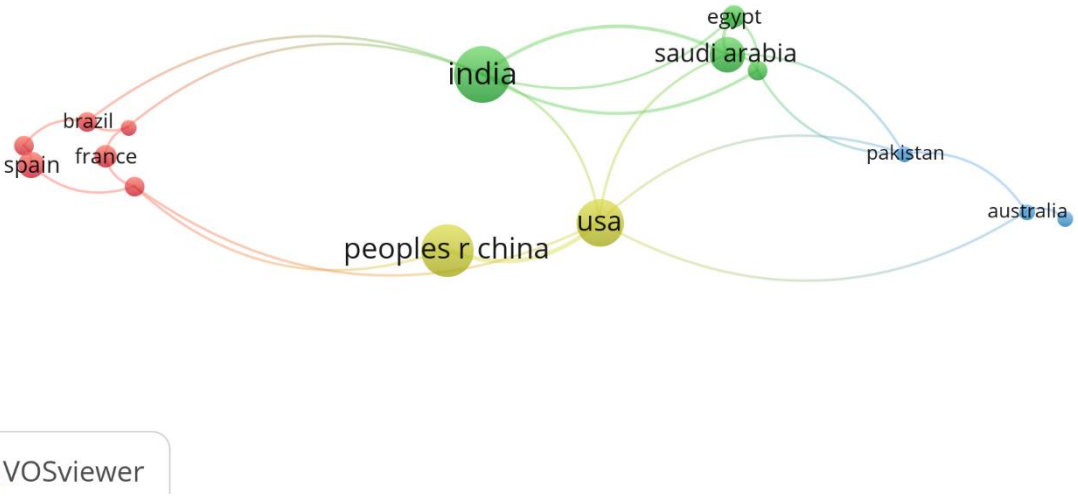
Densidad de búsqueda por palabras



En la figura 5 se tiene el mapa de densidad para los países con más publicaciones realizadas por las investigaciones relaciones al tema como se observa China, Estados Unidos e India tienen predominancia sobre publicaciones alrededor del mundo.

Figura 5

Densidad de búsqueda por países



Conclusiones. La elaboración del SLR mediante la búsqueda de información por medio de los términos claves da como resultado las siguientes respuestas.

RQ1: ¿Qué tecnologías, técnicas y herramientas se utilizan para el desarrollo de prototipos para monitorear los servicios básicos con funcionalidades IoT?

Para el desarrollo de prototipos para monitorear servicios básicos se tiene diversos estudios que muestran que microcontroladores y tecnologías inalámbricas se ocupan para la recepción y transmisión de datos por ejemplo para la recolección de residuos se realiza un contenedor inteligente para predecir el estado del contenedor ya sea su nivel o pronosticar el nivel de aire para este caso se usó para la recolección de datos un Arduino y para la transmisión un ESP8266 mediante Wi-Fi y este prototipo está conectado a la nube mediante Google cloud para poder acceder desde cualquier parte del mundo. (Antonino, Yasin, & Rahman, 2020). Por otra parte, se realiza la medición de corriente, voltaje, potencia activa,

potencia reactiva, potencia aparente, factor de potencia variables que son importantes para medir el consumo y la calidad de la energía eléctrica suministrada para este caso se ocupó dos Esp32 LoRa Heltec para la transmisión de datos a larga distancia y un Esp32 para la toma de datos de los sensores mientas que para subir los datos a la nube se ocupó la tecnología MQTT. (Guevara & Yamir, 2022). Para la medición de un sistema de riego en un jardín y permitir un ahorro del 34% de agua se utiliza sensores de temperatura y humedad del suelo mediante un Esp32 y un chip LoRa RFM95W el cual está conectado al Esp32 para la transferencia de datos mediante LoRa y para mostrar los datos en un Dashboard se ocupa un la tecnología MQTT mediante el Esp32 que tiene tecnología wifi estos dos últimos proyectos están realizados en un servidor local como la PC por lo cual no pueden ser transportados con facilidad a otras partes (Glória, Dionisio, & Simões, 2020).

RQ2: ¿Qué tecnologías, técnicas y herramientas se usan para la gestión de procesamiento de eventos complejos?

Sugieren que la creación, la teoría y la implementación de lenguajes de patrones para asociar eventos de fuentes de datos distantes constituyen el campo de la correlación de eventos. El procesamiento de flujo, la programación reactiva y los eventos complejos (CEP) son solo algunos ejemplos de las muchas familias de técnicas de correlación de eventos que se han desarrollado como resultado de extensas actividades de investigación y desarrollo en respuesta a la naturaleza cada vez más impulsada por eventos de los sistemas en este caso se realizó como es el caso en el la industria de la salud se ocupó Espertech una plataforma de procesamiento de eventos complejos es de código abierto y está disponible en Java su lenguaje de programación es muy parecido a SQL con lo que se analizó los datos de instancias, el diagnóstico temprano y el tratamiento efectivo de los pacientes para sus enfermedades (Rahamani & Babei, 2020)**.** Además, el CEP analiza grandes cantidades de información en ciertas anotaciones de un entorno monitoreado ya sea por sensores o por otros

dispositivos que tomen datos en campo o en el ámbito físico ya sea el caso para la monitorización del tráfico en este caso sea realizó la programación de los eventos complejos en un servidor web privado el cual está programado en java mediante las condiciones para saber en qué partes y áreas de la ciudad existe mayor flujo de tránsito en las diferentes horas del día (Târnaucă & Comnac, 2013)

RQ3: ¿Qué tecnologías y/o técnicas son utilizadas para realizar Benchmarking de un sistema de gestión de servicios básicos?

No se habla mucho sobre pruebas comparativas y de rendimiento, pero algunos documentos describen procedimientos cruciales para determinar si un sistema puede brindar la confiabilidad deseada. Comienzan señalando lo crucial que es realizar validaciones en tiempo real y en entornos de prueba lo más realistas posible para identificar las fortalezas y fallas del sistema. (Chawla, Babu, & Gawande, 2021). Finalmente, se menciona una plataforma de prueba para este tipo de testing denominado Gatling, en el cual se puede definir ciertas condiciones para evaluar el sistema o la arquitectura realizada, y así poder verificar la cantidad máxima de solicitudes por segundo que se podría manejar sin ninguna molestia en cuanto a lo que percibe el usuario, ya que la calidad de funcionamiento del sistema es algo primordial y por eso es importante al sistema colocarlo a cierto estrés. (Lyaskov, Spasov, & Petrova, 2017)

Capítulo II. Marco Conceptual

Internet de las cosas

En el internet de las cosas es importante la interconexión en un sistema de la capa física con la capa lógica o también puede ser servicios virtuales para un fin común que es mejorar las prestaciones para los usuarios mediante las tecnologías de la información.

El procesamiento de los datos es una parte fundamental ya que estos datos deben estar guardados anteriormente para su procesamiento por lo que para llegar a eso se debe tener grandes capacidades de comunicación, pero a la vez la seguridad y confianza que los datos subidos hacia el internet puedan estar a salvo de ataques cibernéticos para seguridad de los usuarios (Unión Internacional de Telecomunicaciones, 2012).

En el internet de las cosas su principal característica es que todos los elementos que conforman el sistema puedan conectarse para enviar y recibir información con esto logramos comunicarnos alrededor del mundo con tiempos de respuesta pequeños y sin la necesidad de tener un usuario remotamente que realice la acción que se quiera realizar. Una vez con todo el entorno conectado a una gran nube, la vida cambiará radicalmente porque no se tendrá que preocupar por ningún tipo de demanda, emergencia o desastre, porque cada dispositivo sabrá qué hacer.

Características del IoT

Las características fundamentales que conforman el IoT, son:

- Interconectividad: Todo puede estar conectado a Internet.
- Servicios relacionados con objetos: Puede proveer de servicios al entorno relacionado a determinado objeto, dentro de la función que desempeñe el objeto.
- Heterogeneidad: Los dispositivos IoT pueden pertenecer a diferentes bases de hardware y software.

- Cambios dinámicos: En IoT los dispositivos pueden aumentar o disminuir dinámicamente ya que estos pueden estar en reposo, prendidos o apagados.
- Escala enorme: El número de dispositivos en volumen podría ser incluso mayor al número de dispositivos conectados actualmente a Internet.

En la figura 6 se muestra cada una de las etapas de la arquitectura para una interconexión IoT.

Figura 6

Arquitectura para el internet de las cosas (IoT)

Nota. Figura tomada de (Benitez, 2016)

Smart Cities

Durante la última década el concepto de Smart Cities o ciudades inteligentes está de moda en el ámbito político por lo que los administradores de las ciudades deben brindar soluciones que hagan a una ciudad acogedora y que facilite la vida de los habitantes ya que una ciudad no es únicamente una estructura física sino también una inmensa red de ciber conexiones encaminadas a la optimización recursos. Cabe mencionar que una ciudad

inteligente la define el Grupo Temático sobre Ciudades Inteligentes y Sostenibles creado por la Unión Internacional de Telecomunicaciones como:

"Una Ciudad Inteligente y Sostenible es una ciudad innovadora que aprovecha las Tecnologías de la Información y la Comunicación (TIC) y otros medios para mejorar la calidad de vida, la eficiencia del funcionamiento y los servicios urbanos y la competitividad, al tiempo que se asegura de que responde a las necesidades de las generaciones presente y futuras en lo que respecta a los aspectos económicos, sociales, medioambientales y culturales".

(FG-SSC, 2015)

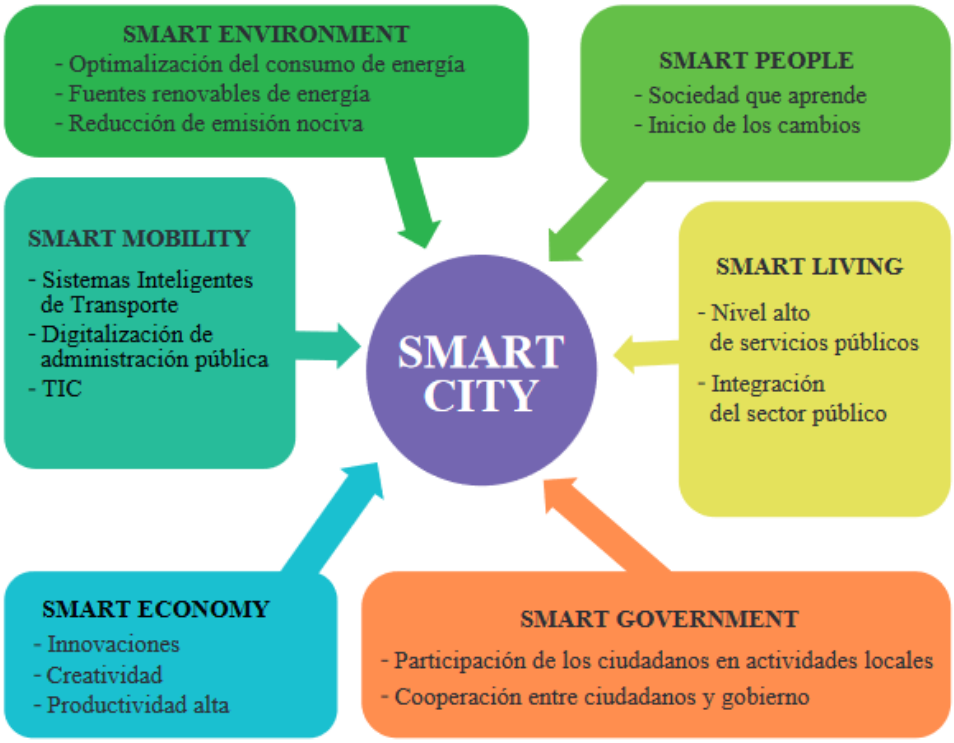
Características de una Smart City

A continuación, en la figura 7 se admite que las ciudades pueden definirse como Smart o ciudades inteligentes, si poseen los siguientes elementos dentro de su estructura:

- Economía (smart economy): Se determina por la inclusión de soluciones innovadoras y adaptaciones a condiciones susceptibles al cambio
- Transporte y comunicación (smart mobility): Complementar el transporte tradicional como soluciones tecnológicas para aprovechar la infraestructura existente
- Medioambiente (smart environment): Optimizar el consumo de energía para así disminuir las emisiones nocivas para el medio ambiente.
- Calidad de vida (smart living). Crear un entorno acogedor y amigable para los habitantes con acceso a una amplia infraestructura con diferentes servicios para facilitar el nivel de vida de los habitantes
- Gestión y administración inteligente (smart governance): Se requiere la cooperación entre autoridades y usuarios de la ciudad para la creación de un conveniente sistema de gestión y administración.

Figura 7

Características de una Smart City



Nota. Figura tomada de (Sikora, 2017)

Procesamiento de eventos complejos (CEP)

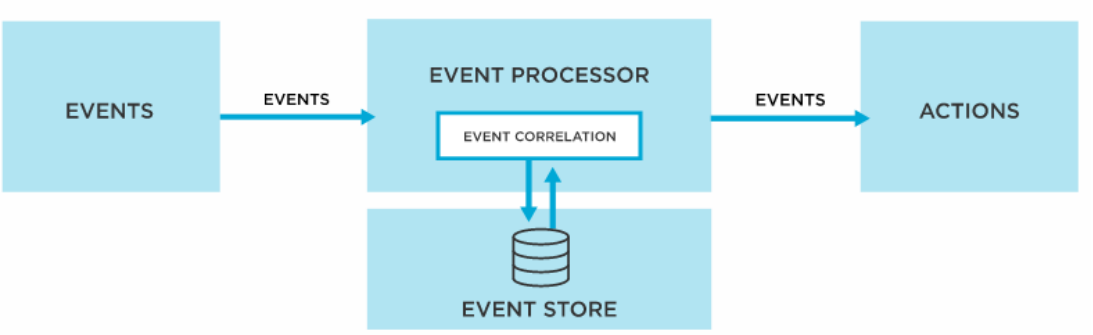
El procesamiento de eventos complejos (CEP) es el uso de la tecnología que permite analizar datos en tiempo real antes de ser almacenarlos en una base de datos como se observa en la figura 8. Se utiliza para rastrear y examinar flujos de datos que identificarán vínculos de causa y efecto entre eventos en tiempo real para sacar conclusiones sobre eventos específicos. Por ejemplo, una forma sencilla de comprender el CEP para el caso de una ciudad inteligente es la recolección de residuos, por lo que se toma distintos eventos simples que podrían ser la temperatura, niveles del contenedor, la fecha y hora. La acción final que se procedería es la recolección de la basura, esto dependerá de los eventos simples que serían

los días hábiles de trabajo, la hora de recolección para disminuir el tráfico y el nivel para saber si es necesario o no recoger la basura. (Ayllon, 2017)

En el caso para rastrear instalaciones industriales mediante el CEP hace esto gracias a mediciones numéricas sin procesar. Algunos ejemplos de este método son un aumento de temperatura y una gran cantidad de humo posteriormente a eso realice la acción de cerrar alguna válvula.

Figura 8

Funcionamiento del CEP



Nota. Figura tomada de (TIBCO, 2019)

Características del CEP

Las principales características del CEP son las siguientes:

- Recopilar una variedad de información y datos simultáneamente.
- Identificar e informar en tiempo real si existe algún evento inusual de las variables que se estén monitoreando.
- Recopilar en bases datos únicamente información relevante para así desechar la información irrelevante cuando se monitorea las variables.
- Disminuir el tamaño de la información en el caso que se requiera descargar y procesar la información obtenida.

Redes LPWAN

Como su nombre lo indica, Low Power and Wide Area Network, llamada LPWAN es una tecnología de comunicación que envía una pequeña cantidad de datos a lo largo de varios kilómetros con bajo consumo de energía. Una aplicación generalizada de esta tecnología son las redes de sensores inalámbricos conocidas como WSN. Según (Raj & Raman, 2017) los beneficios que trae la tecnología de comunicación LPWAN son:

- Conexiones de media a alta densidad de dispositivos de red (reemplazando las comunicaciones basadas en celulares para aplicaciones con bajas tasas de transferencia de datos).
- Aplicaciones que requieren baterías de larga duración estimadas en hasta 10 años.

LoRaWAN

LoRaWAN es un estándar global centrado en el desarrollo de redes LPWAN IoT de área amplia y baja potencia. En el mercado de telecomunicaciones de IoT, coexisten varias tecnologías, estándares y protocolos de comunicación, lo que requiere múltiples soluciones para sensores y actuadores de baja potencia. En general, la capa física de LoRa permite enlaces de comunicación de largo alcance para transmisiones de comunicación, mientras que LoRaWAN tiene un impacto en los siguientes parámetros:

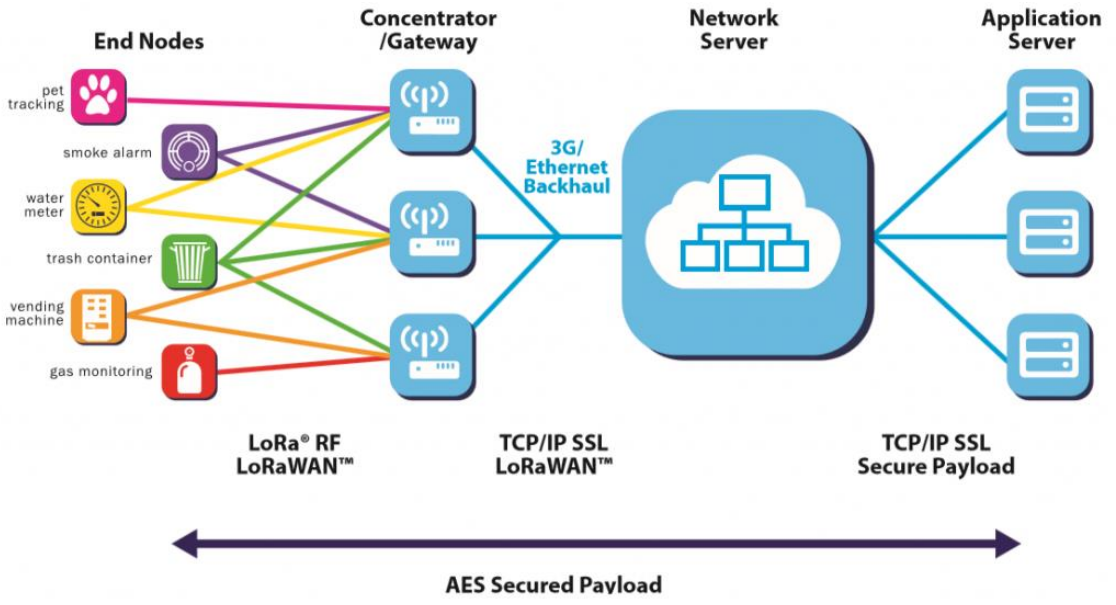
- Duración de la batería del nodo transmisor
- Capacidad de la red
- QoS, Calidad de Servicio
- Nivel de seguridad
- Aplicaciones para servicios web

Estructura de una red LoRaWan

Una red LoRaWAN está conformada por elementos principales como nodos finales, Gateways (pasarelas), servidor de red y servidores de aplicaciones como se observa en la Figura 9. La gran cantidad de datos transmitidos por un nodo final son recibidos por el Gateway o pasarela. Los datos recibidos por cada Gateway son empaquetados y reenviados al servidor de la red a través de distintas tecnologías como por ejemplo la red celular, vía Ethernet, WiFi o satélites. La configuración de software en el Gateway posibilita el reenvío de cualquier paquete de datos recibidos al servidor de red. (Quimbita, 2018)

Figura 9

Estructura de una red LoRaWan



Nota. Figura tomada de (Lora-Alliance, 2018)

Servidor

Un servidor se refiere a una máquina remota que trabaja con una arquitectura cliente-servidor, es decir, se dedica a atender las peticiones de los dispositivos clientes, esta respuesta se genera porque el servidor tiene un programa funcionando de manera continua, dispuestos a responder los pedidos realizados por los dispositivos clientes, estos pedidos se envían a través de redes LAN o internet, los servidores pueden almacenar archivos, aplicaciones, datos y servicios que deben ser respondidos por los servidores de acuerdo con el contenido requerido por los clientes, los servidores también pueden implementar servicios web que son una serie de código que reside en el servidor y se ejecuta al realizar solicitudes o respuestas. (Cases, 2014) El servidor puede tener los siguientes aspectos:

- Cliente y servidor no necesariamente tiene que estar en la misma máquina sino pueden estar en máquinas físicas distintas.
- El cliente interactúa con la interfaz del servidor por lo que no necesariamente necesita conocer el programa que contiene el servidor.
- El servidor presta a todos sus clientes las mismas funcionalidades.
- El servidor se encuentra funcionando de forma perpetua, para no interrumpir las peticiones de los clientes.

Servicios Web

Según Microsoft un servicio Web es *"Una unidad de la lógica del negocio que proporciona datos y servicios para otras aplicaciones. Las aplicaciones acceden a los Web services vía protocolos Web como HTTP y SOAP y formatos de datos universales como XML, sin necesidad de preocuparse de cómo cada Web Service es implementado"*. (Gallegos, 2011) Por lo que se dice que son funciones que se encuentran albergadas en un servidor en la que diferentes servicios o dispositivos la utilizan, lo que permite el intercambio de mensajes entre el

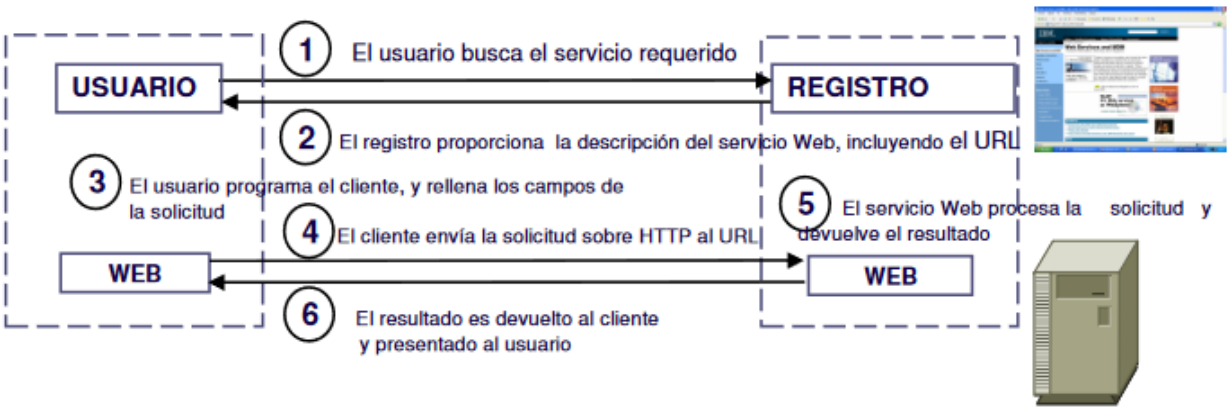
cliente y el servidor así tengan distintos lenguajes de programación cada uno. Los servicios web tienen ciertas características como las siguientes:

- Se trabaja con facilidad y desde cualquier lugar ya que casi todos los dispositivos soportan HTTP.
- Alto grado de escabilidad ya según el proyecto vaya aumentando solo se deben aumentar estos servicios web.
- El programa que contiene los servicios web puede estar muy remotamente de donde serán consumidos estos servicios.

En la Figura 10 se muestra la descripción de un servicio web.

Figura 10

Descripción del servicio web



Nota. Figura tomada de (Gallegos, 2011)

Bases de Datos

Es un conjunto de datos que tienen un origen parecido que son escritos de una forma ordenada para su posterior análisis o procesamiento. El procesamiento de esta información dependerá de la finalidad que tenga el usuario u organización. (Grapheverywhere, 2021)

Los datos almacenados en las bases de datos son los encargados de recopilar la información obtenida por los sensores que se encuentran en el campo estos datos son de gran

importancia debido a que permite realizar gráficas de cada componente o por ejemplo de cómo va variando los datos según en el tiempo y con un correcto análisis estadístico con modelos predictivos, solventar problemas existentes o que antes no hubieran sido tratados a tiempo, una base de datos se encarga de guardar datos del pasado y presente por ejemplo de los prototipos realizados en el presente proyecto para predecir el consumo de agua, consumo de luz de cada hogar y en qué sectores de la ciudad se tiene un mayor consumo de agua o luz para que así se realice un correcto análisis y a futuro en proyectos que se vayan realizando por las empresas que proporcionan estos servicios puedan tomar las mejores decisiones acorde a los datos obtenidos.

MQTT

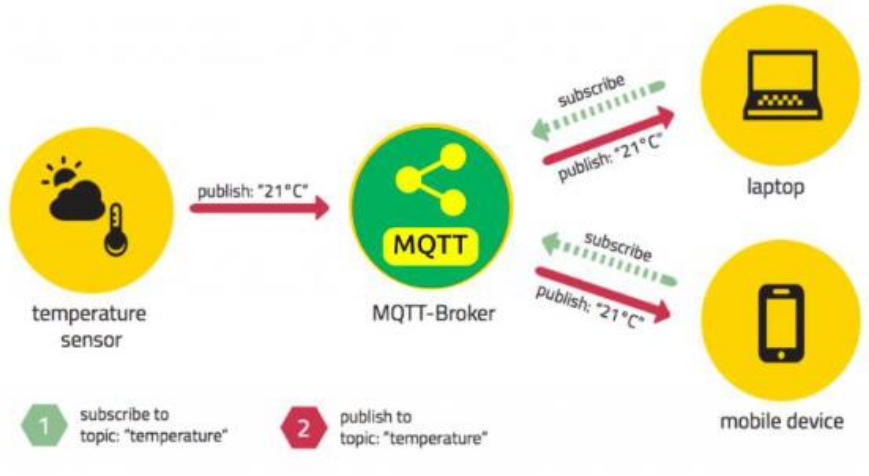
Para llevar a cabo el proceso de comunicación e intercambio de información entre los dispositivos de la red, se puede utilizar HTTP y se puede construir un modelo de publish/subscribe. Sin embargo, existen estándares diseñados para ser mucho más livianos que HTTP, que funcionan de manera eficiente en redes inalámbricas y con requisitos de conectividad intermitente. MQTT funciona de forma simple los mensajes o dispositivos a enviar se clasifican según un “tópico” el cual es el tema del mensaje, es decir a quien va dirigido o de quien se desea recibir el mensaje como se muestra en la figura 11, están conectados en forma de una red estrella en la que el nodo central es un Broker el cual gestionará el tráfico de los mensajes que le lleguen de cada tópico y a manera de un Hub los encaminará a los dispositivos que estén suscritos, utilizar MQTT trae muchos beneficios como son los siguientes (Mahedero, 2020).

- Requiere un ancho de banda mínimo y puede implementarse en redes con alta latencia o baja calidad.
- En la actualidad para dispositivos IoT se ocupa este protocolo por su efectividad.

- Consume muy poca energía y es muy recomendable para dispositivos alimentados por batería.
- Existen intermediarios, llamados brokers, que son servidores delegados para la gestión del tráfico de mensajes.

Figura 11

Transmisión MQTT



Nota. Figura tomada de (Pietro, 2019)

Broker MQTT

El Broker MQTT es un intermediario el cual es un servidor de mensajes que se encarga de recibir los mensajes y repartirlos utilizando el modelo publish/ subscribe lo cual permite ocupar la comunicación MQTT como se observa en la Figura 8. Debido a que los clientes envían regularmente paquetes de datos y esperan la confirmación del servidor o broker, el canal siempre está activo. Una de las mayores ventajas de un broker es que en el caso que el publicador o el subscriptor se desconecten entre sí, el servidor guarda el mensaje y espera a que el cliente se conecte nuevamente para el envío y recepción de los datos como se muestra

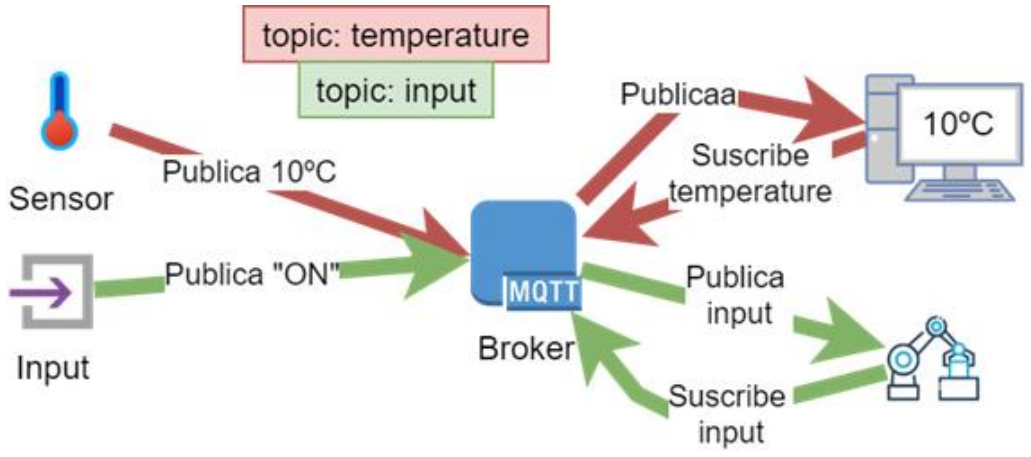
en la figura 12. Suele utilizarse para dispositivos IoT ya que permite el intercambio de mensajes entre dispositivos con recursos limitados como puede ser la potencia o el ancho de banda. (Biswajeeban & Attila , 2021)

El broker puede estar instalado en la maquina personal o instalado en un servidor remoto en la nube. En el presente trabajo se utilizará el broker Mosquitto, las características de los broker son las siguientes:

- Optimizan recursos del servidor donde se encuentren instalados ya que son ligeros.
- Son escalables fácilmente lo cual permite gestionar todo el tránsito de mensajes.
- Remite la información según el tópico asignado como un hub.
- Permite encriptar mediante SSL/TLS lo cual hace que se pueda autenticar al momento de remitir los mensajes y sea mucho más seguro.

Figura 12

Estructura MQTT



Nota. Figura tomada de (Intriago, 2020)

Sensores

En los sistemas automatizados, los sensores son herramientas que pueden detectar magnitudes físicas y proporcionar una señal útil para el operador, como voltaje o corriente, en relación con la cantidad de la magnitud física medida. Los sensores se encuentran en la capa física donde se puede detectar las condiciones del proceso y medir variables propias del proceso que se esté monitoreando (Kaschel & Pinto, 2020). Por ejemplo, en este trabajo se está monitoreando el nivel, caudal, corriente, voltaje entre otros, todos estos valores son tomados por el microcontrolador a utilizar ya sea mediante sus entradas analógicas o digitales.

Como se mencionó existe una gran diversidad de sensores que son utilizados para la medición de variables y en la tabla 5 se definen los sensores que se consideran más acertados para el proyecto.

Tabla 5

Sensores utilizados en el proyecto

Nombre del elemento	Voltaje de alimentación	Tipo
Sensor de caudal Yf-201	5 V	Digital
Sensor de corriente SCT013	5 V	Analógico
Sensor de voltaje zmpt101b	5 V	Analógico
Sensor ultrasónico HC-SR04	5 V	Analógico

Microcontroladores

Un microcontrolador es un dispositivo electrónico que puede realizar operaciones lógicas para llevar a cabo una determinada tarea. Cada microcontrolador soporta un lenguaje de programación diferente, para lo cual el usuario debe conocer estos lenguajes para programar estos dispositivos. La CPU, la memoria, los puertos de entrada y salida, así como la conexión disponible en estos microcontroladores, varían entre ellos. Las aplicaciones de IoT

con frecuencia necesitan operar en tiempo real o como mínimo, responder instantáneamente. Por eso es preferible implementar un microcontrolador con una arquitectura de 32 bits y una variedad de opciones de conexión. (Asanza, 2021)

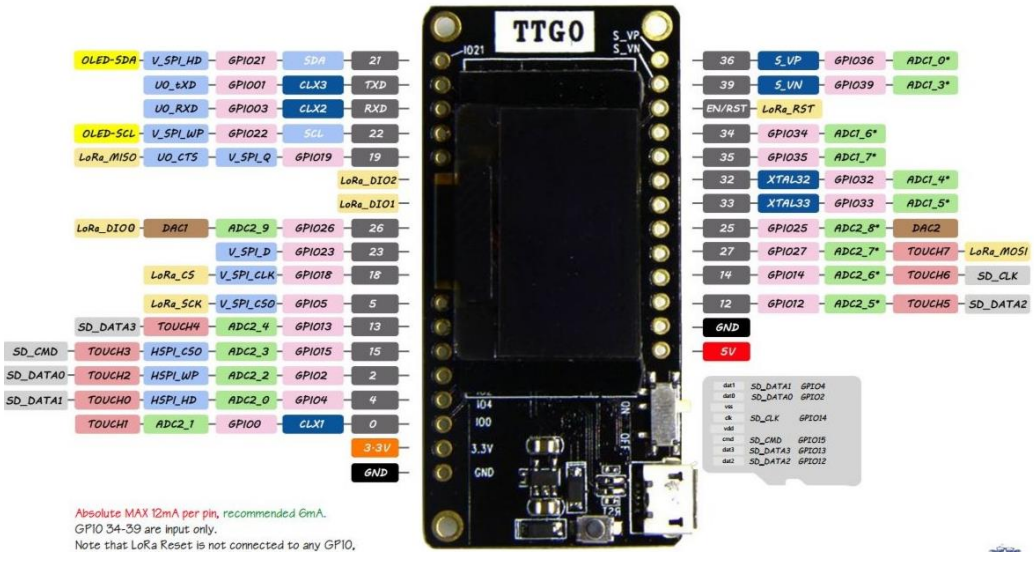
Las tarjetas de programación más conocidas se han revidado y se caracterizan a continuación ya que son funcionales para aplicaciones IoT.

Ttgo Esp32 LoRa V1.0

Es una tarjeta programable compatible con arduino la cual contiene un Esp32 para el procesamiento de los datos. Permite una comunicación inalámbrica mediante Wifi y la red LPWAN LoRa mediante el circuito integrado SX1276. Este módulo contiene una pantalla OLED de 0.96” la cual es configurable mediante la librería adafruit en arduino. En la tabla 6 se especifica más a fondo.

Figura 13

TTGO ESP32 LoRa V1.0



Nota. Figura tomada de (Wurst, 2021)

Tabla 6

Características de la tarjeta TTGO ESP32 LoRa V1.0

TTGO ESP32 LoRa V1.0	
Master Chip	ESP32 (240MHz dual core+1)
Lora Chip	SX1276
Frecuencia	868/915 MHz
Wi-Fi	Protocolo: 802.11 velocidad de hasta 150 Mbps)
Memoria	4 Mb
Alimentacion	5V USB – 3.7 V
Voltaje de salida	3.3 V
Medida Display	0.96” OLED
Temperatura de operación	-40 a 90º C
Dimensiones	51.43 x 20.01 mm

Nota. Tabla elaborada en base a la información proporcionada por (Wurst, 2021).

Heltec Esp32 LoRa V2

Es una placa de desarrollo IoT basado en el microcontrolador ESP32 con un chip de comunicación LoRa SX1278 o SX1276 dependiendo de la región como se observa en la figura 14. Mediante el protocolo LoRa Wan permite una comunicación de largo alcance a un bajo consumo de energía además posee una pantalla OLED de 0.96” para presentar la información que se requiera. A continuación, en la tabla 7 se especifica las características de la tarjeta más a fondo. (Heltec, 2020)

Tabla 7

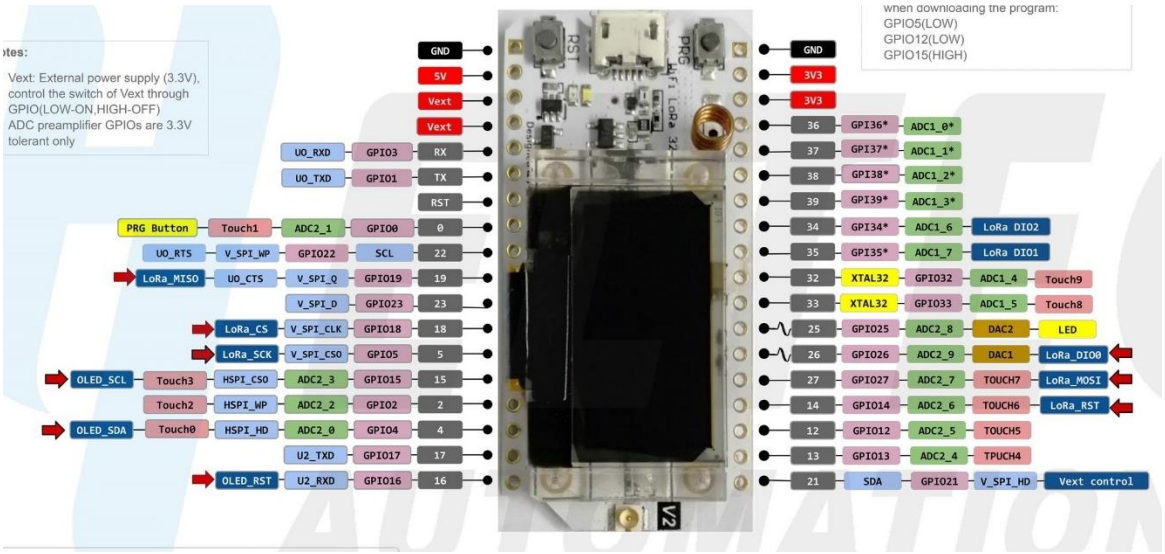
Características de la tarjeta Heltec ESP32 LoRa

Heltec ESP32 LoRa V2	
Master Chip	ESP32 (240MHz dual core+1)
Lora Chip	SX1276/ SX1278
Frecuencia	470/510 MHz, 868/915 MHz
Wi-Fi	Protocolo: 802.11, velocidad de hasta 150 Mbps
Memoria	8 Mb (64M-bits)
Alimentacion	5V USB – 3.7 V
Voltaje de salida	3.3 V
Medida Display	0.96” OLED
Temperatura de operación	-20 a 70º C
Dimensiones	51.00 x 25.50 mm

Nota. Tabla elaborada en base a la información proporcionada por (Heltec, 2020).

Figura 14

Heltec ESP32 LoRa V2



Nota. Figura tomada de (Heltec, 2020)

Capítulo III. Diseño

Requisitos de diseño

Para el diseño de la arquitectura se ha seguido el modelo mencionado en el capítulo uno, esta arquitectura tiene 3 niveles que son una capa física, una capa lógica y una capa de aplicación. En la parte de la capa física se tiene los tres prototipos con cada uno de sus sensores correspondientes para medir el consumo eléctrico, el consumo de agua y el nivel de residuos de un contenedor de basura, estos dispositivos serán los nodos que posteriormente se comunicarán con un Gateway o concentrador para ser subida la información a la nube. En el nivel lógico se detalla la metodología usada y el diseño del algoritmo donde está implementado el procesamiento de eventos complejos para saber en qué fecha y cual asido el consumo para notificar al usuario mediante correo y posteriormente notificar el corte de servicio en caso de no pagar por otra parte, para la recolección de residuos se notificará al recolector siempre y cuando cada cierta el contenedor se encuentre mayor al 75% para proceder a la recolección de la basura. Finalmente, en la capa de aplicación se aborda el diseño de la interfaz para monitorear en tiempo real los tres prototipos antes mencionados.

Requisitos de la arquitectura

En la tabla se muestra los requisitos que debe cumplir cada uno de los elementos dentro de cada nivel de la arquitectura del sistema de gestión de servicios básicos.

Tabla 8

Requisitos de la arquitectura del sistema

Requisitos	Descripción
Requerimientos que debe cumplir la arquitectura en la Capa Física.	• Los tres prototipos deben ser independientes tanto para el consumo de agua, luz y nivel de residuos. • Los datos de consumo se deben guardar en la memoria del Arduino. • Los sensores utilizados deben ser lo menormente posible invasivos. • Simular situaciones en una maqueta para el monitoreo de los datos. • Los prototipos deben tener conexión a Lora para la transmisión de los datos. • Se debe tener un concentrador para recibir la información de los tres nodos.
Requerimientos que debe cumplir la arquitectura en la Capa Lógica.	• El proyecto debe estar en un servidor remoto para que el cliente puede acceder en cualquier parte. • Evita el consumo excesivo de recursos del ordenador para ejecutarse en un VPS en la nube. • Se basa su funcionamiento en el procesamiento de eventos complejos. • Se debe contar con envíos de correos a los usuarios. • Permite la notificación con fecha y hora a los usuarios para pagar sus tarifas de consumo y notificar en el caso de corte.

Requisitos	Descripción
	• Notifica al recolector si es necesario o no la recolección de los residuos tomando en cuenta los días laborables de la semana. • El Gateway concentrador debe comunicarse con la capa lógica mediante MQTT.
Requerimientos que debe cumplir la arquitectura en la Capa de Aplicación.	• La interfaz debe contar con monitoreo en tiempo real. • Permite la implementación del algoritmo principal, así como la conexión a una base de datos en la nube. • Se debe tener una base de datos para cada prototipo • La base de datos debe ser incremental debido a la acumulación de datos que puede existir de cada nodo. • Se debe monitorear por separado los tres prototipos. • Cada componente analógico que se tenga en los prototipos se debe mostrar gráficos históricos de su progreso • La interfaz puede ser vista en cualquier cliente si se instala los programas requeridos de esta capa.

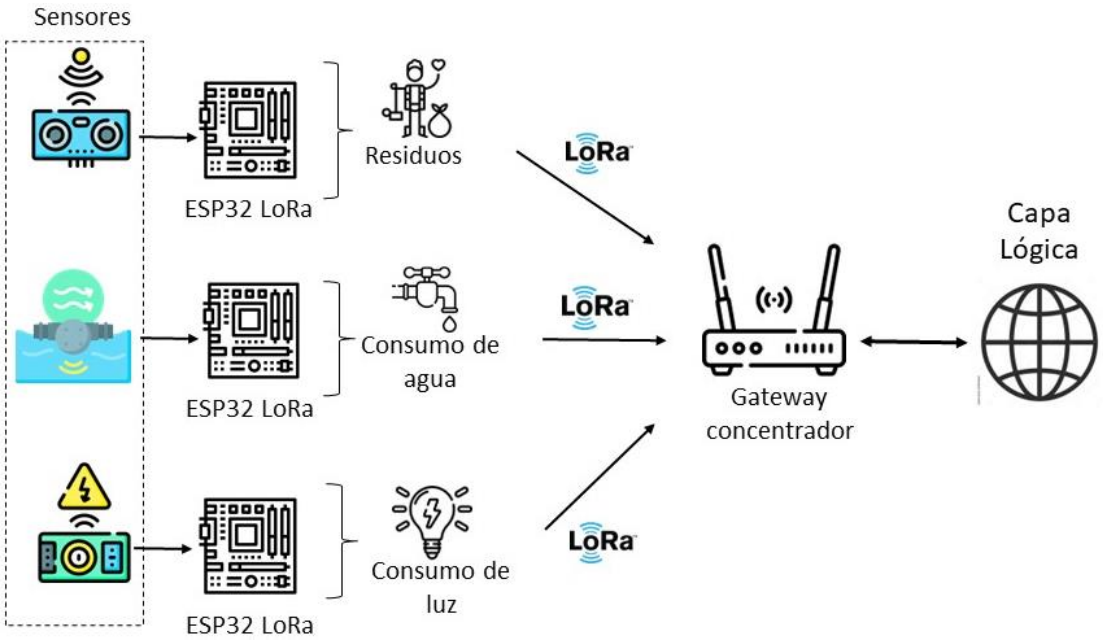
Para el diseño de la arquitectura se empieza con el nivel físico pues contiene los tres prototipos que se encargan de monitorear las variables para posteriormente en la capa lógica ser procesados mediante un algoritmo de procesamiento de eventos complejos para al final en la capa de aplicación mostrar los datos en un Dashboard.

Capa física

En la capa física de la arquitectura propuesta se encuentra los sensores y microcontroladores en este caso son dos para cada prototipo uno es el Arduino nano y para la transmisión de datos se ocupará un Esp32 LoRa y un Gateway o concentrado que de igual manera será un Esp32, en el cual llegará la información proveniente de los tres nodos como se observa en la figura 15, los sensores estarán colocados en el entorno físico para medir las variables de nivel, corriente y caudal para los prototipos antes mencionados. Para el prototipo de residuos se contará con un sensor ultrasónico para la medición del nivel de basura, mientras que para el prototipo de consumo de agua se contará con un sensor de flujo para la medición del caudal de agua que pase por la tubería y por último para el consumo de luz se ocupará dos sensores uno de corriente no invasivo y otro de voltaje para con estos datos obtener el consumo en kWh de las cargas que se coloquen. El Arduino nano va adquirir las señales provenientes de los sensores mediante sus entradas análogas y digitales dependiendo del sensor que se coloque en cada prototipo, posteriormente con la programación realizada procesará los datos de los sensores para así obtener los valores de nivel del contenedor, consumo de agua (m^3) y consumo de luz (kWh). El Arduino nano mediante comunicación serial enviará los datos al Esp32 LoRa, una vez los datos se encuentre ahí estos se enviarán mediante la comunicación inalámbrica LoRa al concentrador o Gateway. Finalmente, el Gateway enviará la información obtenida de los prototipos mediante el protocolo MQTT al broker Mosquitto y el cual se conectará a la capa lógica para el manejo y programación de los distintos eventos que se puedan dar.

Figura 15

Estructura general de la capa física



Características de los elementos del sistema

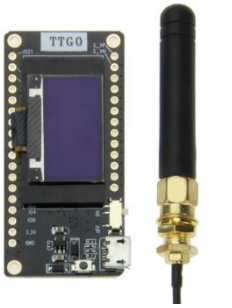
Para el diseño de los prototipos del proyecto se utiliza distintos tipos de sensores por lo cual se enlista en las siguientes tablas los elementos que conforman cada uno de los prototipos. Los microcontroladores que se ocuparon son unos Esp32 LoRa cabe mencionar que se utilizó uno de la marca Heltec y los demás de la marca Ttgo, esto se lo realizó por la disponibilidad en el mercado y el tiempo de llegada al país de estos módulos, pero las funcionalidades son las mismas únicamente cambia la posición de los puertos. Además, se debe tener en cuenta la frecuencia que funcionan estos módulos porque cada región tiene su frecuencia de trabajo para este caso en Ecuador funcionan a una frecuencia de 915 MHz.

Diseño prototipo consumo de luz

En el prototipo de luz se ocupó distintos sensores y microcontroladores en la tabla 9 se muestra cada uno de los elementos que conforman el prototipo con sus respectivas características.

Tabla 9

Listado de componentes prototipo consumo de luz

Nombre	Imagen	Características
Zmpt101b		• Sensor de voltaje • Salida de tipo análoga • Voltaje de alimentación: 3.3V ~ 5V DC • Corriente nominal de entrada y salida: 2mA • Voltaje máximo de medición 250VAC
SCT013-100A		• Sensor de corriente • Tipo de salida análoga • Voltaje de alimentación: 3.3V ~ 5V DC • Corriente de entrada: 0-100A AC. • Temperatura de operación: -25°C ~70°C • Modo de salida: 0~50mA
Arduino nano		• Microcontrolador: ATmega328 • Voltaje de alimentación: 7-12 VDC fuente externa y por USB 5VDC • E/S digitales: 14 • E/S análogas: 8 • Memoria: 32 KB • EEPROM: 1 KB
Ttgo Lora **Esp32**		• Master chip; ESP32 (240 MHZ dual core) • Chip Lora: SX1276 • Frecuencia: 868/915 MHz • Memoria: 4Mb • Alimentación: 5V USB – 3.7 V Externo. • Display: 0.95” OLED • Puertos: 29 GPIO

Circuito de acondicionamiento para el sensor de corriente SCT013. Este tipo de sensor funciona como un transformador de corriente en el cual tiene una relación de transformación de 1/2000 ya que nos entrega 50 mA por 100 A. Por lo cual se necesita calcular la resistencia de carga que irá conectada en paralelo a la salida del sensor ya que este nos entrega una señal de corriente y el Arduino solo admite señales de voltaje.

Para calcular la resistencia de carga conocemos la corriente que admite el sensor en este caso hasta 100 A por lo que debemos obtener la corriente pico y con esto mediante la fórmula de relación de transformación obtenemos la resistencia de carga.

$$I_{RMS} = \frac{I_{pico}}{\sqrt{2}}$$

$$I_{pico} = I_{RMS} * \sqrt{2} = 100 * \sqrt{2} = 141.42\ A$$

$$\frac{N_p}{N_s} = \frac{I_s}{I_p}$$

$$I_s = \frac{1 * 141.42}{2000} = 0.0707\ A$$

Finalmente toca máxima la resolución al valor máximo de voltaje admitido por la tarjeta en este caso la entrada del Arduino nano admite 5 V.

$$R_{carga} = \frac{\frac{A_{REF}}{2}}{I_{pico}}$$

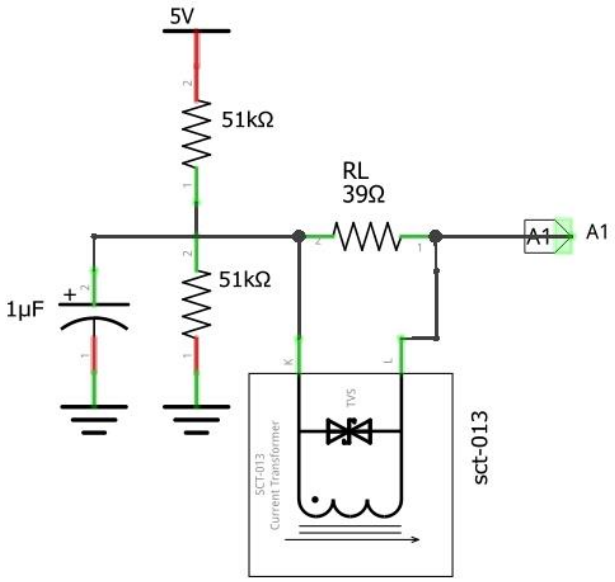
$$R_{carga} = \frac{\frac{5}{2}}{0.0707} = 35.36\ \Omega$$

En el mercado no se encuentra resistencias de este valor por lo que se eligió una cercana de $39\ \Omega$, ya que no habría problema porque no vamos a medir cargas que tenga una corriente que llegue a los 100 A. Finalmente mediríamos valores con esta resistencia de 2.5 V a -2.5 V el problema es que Arduino no registra señales negativas asique tenemos que modificar

la señal para que varíe en el rango de 0 a 5 V añadiéndole un offset mediante un divisor de tensión para sumarle 2.5 V, en la figura 16 se muestra el circuito de acondicionamiento.

Figura 16

Circuito de acondicionamiento sensor de corriente

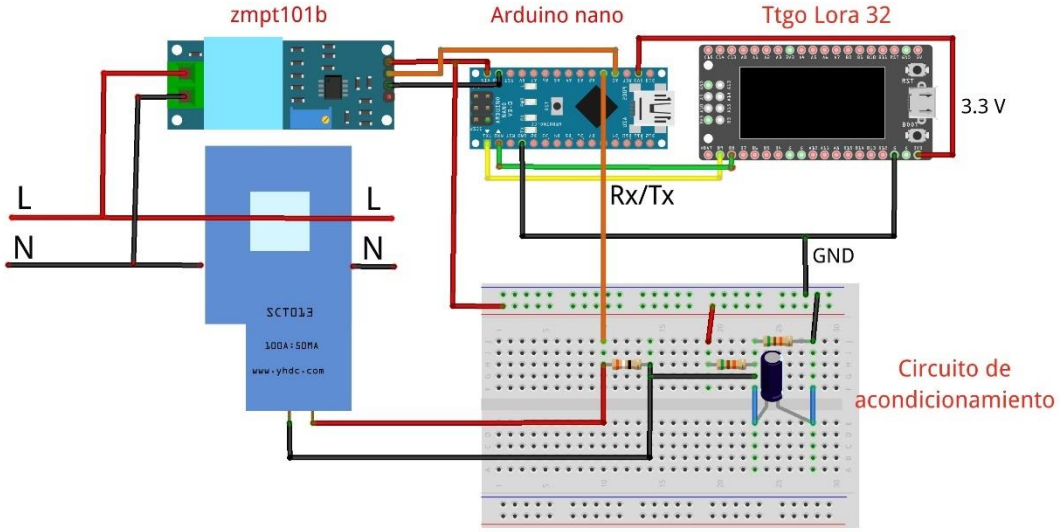


Para el desarrollo del prototipo se ocupó dos tipos de sensores uno de corriente (SCT013) como se acaba de observar y uno de voltaje (zmpt101b). Estos sensores son los encargados de medir estas dos variables para posteriormente obtener el consumo de luz mediante la programación del Arduino con las fórmulas de potencia y voltaje. En el Arduino nano se tomó la consideración de guardar los valores de consumo en la memoria EEPROM ya que puede existir la posibilidad que se suspenda el servicio eléctrico y se borren los valores de consumo. Posteriormente mediante comunicación serial se transfiere los datos de voltaje, corriente y consumo al Esp32 LoRa para que este envíe al Gateway mediante LoRa los valores procesados, este módulo se lo escogió debido a que tiene una pantalla OLED y nos facilitará la presentación de los valores en el encapsulamiento de los prototipos y facilidad del usuario para

observar los valores en tiempo real, en la figura 17 se observa el diagrama general del prototipo.

Figura 17

Diagrama general de conexión del prototipo para el consumo de luz



Diseño prototipo consumo de agua

En el prototipo de agua se ocupó un sensor de flujo y 2 microcontroladores en la tabla 10 se muestra cada uno de los elementos utilizados que conforman el prototipo con sus respectivas características.

Tabla 10

Listado de componentes prototipo consumo de agua

Nombre	Imagen	Características
YF-S201		• Sensor de flujo • Tipo de salida digital • Voltaje de funcionamiento: 5V – 18V DC. • Flujos en el rango de 1 a 30 l/min. • Rosca externa de ½ pulgada. • Presión de trabajo máximo: 1.75 Mpa
Arduino nano		• Microcontrolador: ATmega328 • Voltaje de alimentación: 7-12 VDC fuente externa y por USB 5VDC • E/S digitales: 14 • E/S análogas: 8 • Memoria: 32 KB • EEPROM: 1 KB
Heltec Esp32 V2		• Master chip; ESP32 (240 MHZ dual core) • Chip Lora: SX1276 • Frecuencia: 868/915 MHz • Memoria: 8 Mb • Alimentación: 5V USB – 3.7 V Externo. • Display: 0.96” OLED • Puertos: 29 GPIO

Para el desarrollo del prototipo para monitorear el consumo de agua se utilizó un sensor de flujo YF-S201 de ½”, ya que las tuberías de los hogares tienen esta medida en cual mediante la cantidad de flujo en el tiempo que pase se calculará el consumo de agua, este sensor tiene tres cables rojo y negro para su alimentación y amarillo para la salida de pulsos la

cual es una onda cuadrada donde la frecuencia es proporcional al caudal que ingresa. El factor de conversión de frecuencia varia de acorde a los fabricantes en este caso el fabricante nos proporciona la siguiente fórmula.

$$f(Hz) = k * Q(L/min)$$

El factor k el fabricante recomienda un valor de 7.5 pero en este caso para la calibración del sensor se debe realizar algunas mediciones de prueba para contar los pulsos de acorde al volumen con la siguiente expresión se obtiene

$$k = \frac{n^o pulsos}{Volumen * 60}$$

Para la obtener el consumo de agua o volumen partimos del caudal una vez calibrado el sensor ya que el caudal es la variación del volumen con respecto al tiempo como se observa en las siguientes expresiones.

$$Q = \frac{\Delta V}{\Delta t}$$

$$V = Vo + Q\Delta t$$

Donde:

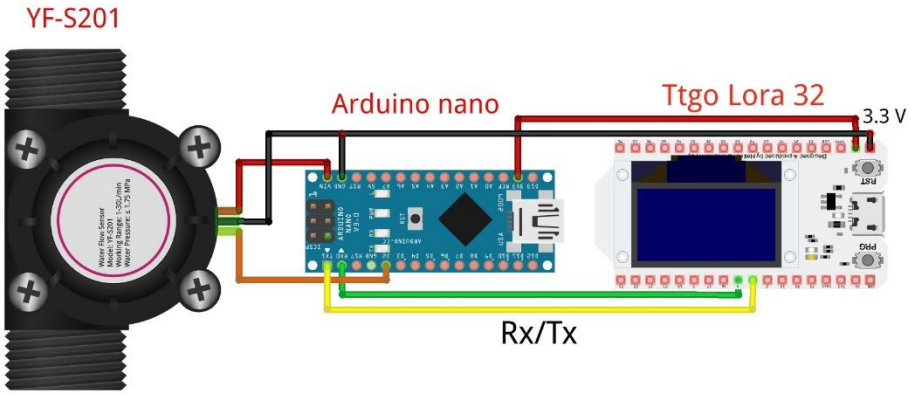
$$Vo = volumen\ inicial$$

$$\Delta t = variación\ del\ tiempo\ en\ segundos$$

Una vez obtenido los valores de volumen, caudal y consumo se procede a enviar mediante comunicación serial al Esp32 LoRa, el cual tiene una pantalla OLED en el que se presentará estos valores para que el usuario puede observar en tiempo real los valores que este consumiendo. El Esp32 LoRa mediante comunicación LoRa enviará los datos al Gateway para que este pueda subirlos a la nube posteriormente. En la figura 18 se observa el diagrama general de conexión del prototipo.

Figura 18

Diagrama general de conexión del prototipo para el consumo de agua



Diseño prototipo nivel de residuos

Para el desarrollo del prototipo de nivel de residuos se ocupó un sensor ultrasónico y un microcontrolador en la tabla 11 se muestra los elementos utilizados para la construcción del prototipo.

Tabla 11

Listado de componentes para el prototipo nivel de residuos

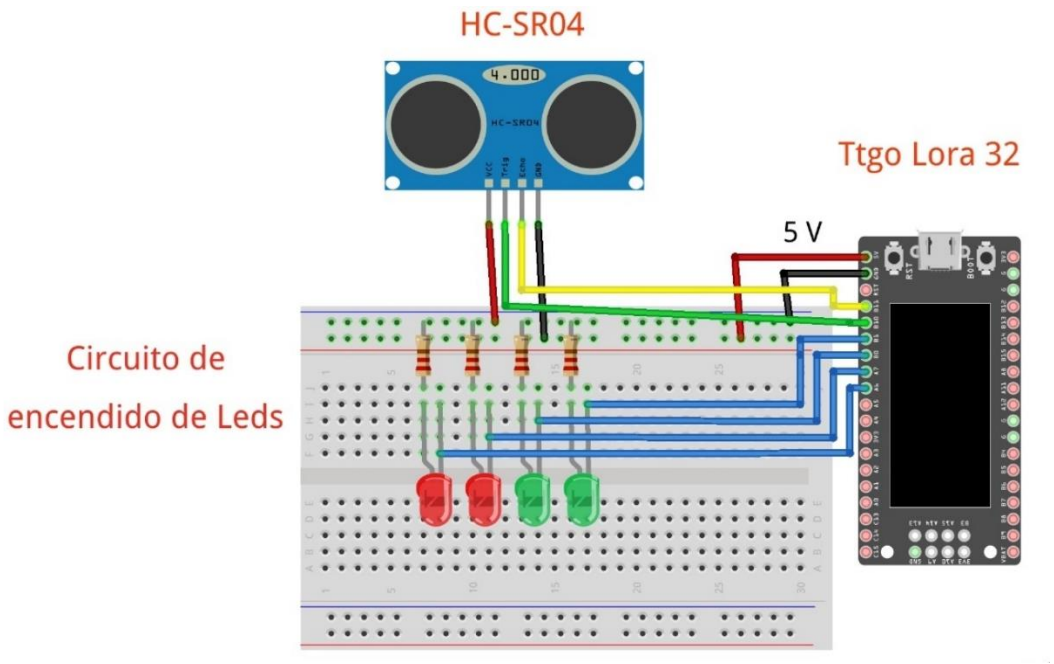
Nombre	Imagen	Características
HC-SR04		• Sensor de flujo • Tipo de salida digital • Voltaje de funcionamiento: 5V – 18V DC. • Flujos en el rango de 1 a 30 l/min. • Rosca externa de ½ pulgada. • Presión de trabajo máximo: 1.75 Mpa
Diodo led		• Paso de corriente de un solo sentido. • Polarizado emite un haz de luz. • Voltaje de funcionamiento de aproximadamente 2 V.
Ttgo Lora Esp32		• Master chip; ESP32 (240 MHZ dual core) • Chip Lora: SX1276 • Frecuencia: 868/915 MHz • Memoria: 4Mb • Alimentación: 5V USB – 3.7 V Externo. • Display: 0.95” OLED • Puertos: 29 GPIO

En el prototipo de residuos se ocupa un sensor ultrasónico HC-SR04 el cual se encargará de medir la distancia, para el monitoreo de residuos se plantea cinco niveles como son nivel a 0%, 25%, 50%, 75% y 100% para los cuales también se indicarán mediante leds indicadores, los dos leds verdes para los niveles hasta el 50% y los dos leds rojos para los niveles 75% y 100%. En este caso se ocupa únicamente un Esp32 LoRa el cual se encargará

de realizar el procesamiento de los datos, esto debido a que no necesariamente tiene que estar midiendo los niveles constantemente como en el caso de los prototipos de agua y luz ya que cuando se envía los datos mediante LoRa hacia el Gateway únicamente ahí es cuando se activa la tarjeta para realizar las mediciones, pero en el caso del contenedor de basura no es necesario ya que se enviarán datos cada diez segundos, ya que así está configurado el Gateway por lo que censar el nivel cada este cierto tiempo es prudente y no se necesita otro microcontrolador lo que nos ahorra costos y el prototipo es más pequeño para su instalación. Finalmente, estos datos de distancia obtenidos por el sensor ultrasónico serán enviados por el Esp32 LoRa hacia el Gateway para su posterior envío a la nube, en la figura 19 se observa el diagrama general de conexión del prototipo de nivel de residuos.

Figura 19

Diagrama general de conexión del prototipo de nivel de residuos

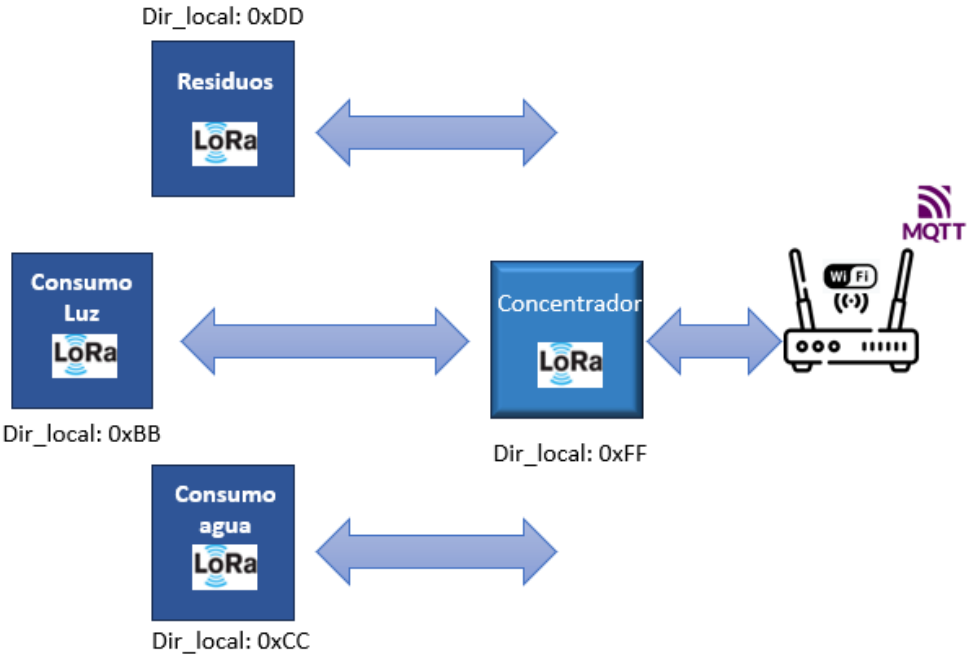


Diseño Gateway

Para el diseño del Gateway se ocupa la arquitectura multipunto de LoRa ya que se tiene tres prototipos los cuales serán nuestros nodos, para el proyecto se ocupa una tarjeta Ttgo Esp32 Lora como concentrador o Gateway al cual llegará los datos de nuestros tres prototipos, se ocupa esta tarjeta debido a que es más barato que adquirir un Gateway, por lo que para funcionalidades del proyecto es suficiente. Para lograr la comunicación es necesario realizar un direccionamiento para cada prototipo y para el nodo master o concentrador ya que no se puede enviar simultáneamente los datos de los tres prototipos a la vez ya que únicamente estos módulos Esp32 Lora son de únicamente un canal y habría un error al momento de decodificar los datos por consiguiente se procedió a dar un tiempo de cinco segundos para la transmisión del prototipo de agua y cinco segundos para el prototipo de luz, pero para el prototipo de residuos únicamente se dieron tres segundos debido a que únicamente toma el nivel que se encuentre en ese momento y no sea tan largo el tiempo de espera entre prototipos para el envío de los datos, por último estos módulos Esp32 incorporan Wifi lo que permitirá enviar los mensajes del Gateway al broker Mosquitto. En la figura 20 se puede observar el esquema de comunicación utilizado para el envío y recepción de los datos mediante Lora.

Figura 20

Esquema de comunicación de los datos

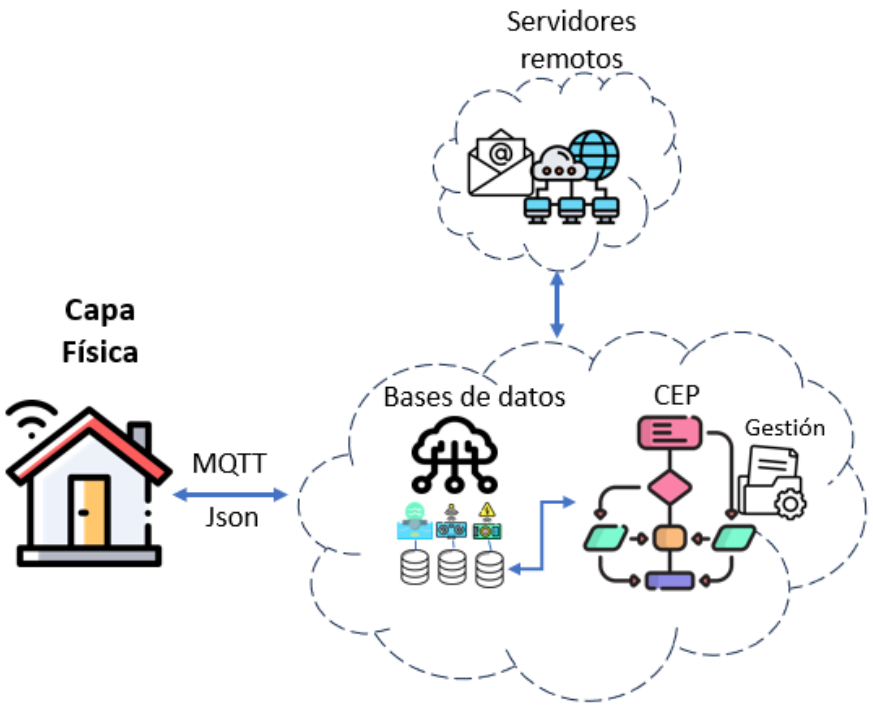


Capa Lógica

La capa lógica corresponde al back-end del proyecto, en esta se encuentra toda la lógica de funcionamiento del sistema de gestión de servicios básicos, en esta capa se encuentra las base de datos, el procesamiento de eventos complejos y la conexión a servidores remotos como el correo electrónico para la notificación de los eventos que se vayan dando en los prototipos, en la figura 21 se puede observar la arquitectura genérica de la capa lógica donde la capa física envía los datos en formato JSON a la capa lógica mediante MQTT, para que esta gestione, y guarde toda la información que sea necesaria.

Figura 21

Arquitectura Genérica para la Capa Lógica



Diseño del Procesamiento de eventos complejos (CEP)

El diseño del procesamiento de eventos complejos es la unión de distintos eventos simples que se vayan dando en los prototipos, para así con esos eventos simples realizar una acción en tiempo real por lo que se ha tomado las siguientes consideraciones para cada prototipo como se observa en la tabla 12. La programación del algoritmo será realizada en Node-red.

Tabla 12

Consideraciones para el diseño del CEP

Nombre	Eventos simples	Eventos complejos
Prototipo consumo de luz	• Consumo en kWh • Consumo en dólares • Fecha • Hora • Se realizó o no se realizó el pago	• Enviar una notificación mediante correo al consumidor con el consumo y valores a pagar cada determinado día del mes. • Enviar una notificación de corte en el caso de no pagar después de determinados días con el consumo y valores a pagar para restablecer el servicio.
Prototipo consumo de agua	• Consumo en m^3 • Consumo en dólares • Fecha • Hora • Se realizó o no se realizó el pago	• Enviar una notificación mediante correo al consumidor con el consumo y valores a pagar cada determinado día del mes. • Enviar una notificación de corte en el caso de no pagar después de determinados días, con el consumo y valores a pagar para restablecer el servicio.
Prototipo nivel de residuos	• Nivel del contenedor • Fecha • Hora • Días de la semana	• Enviar una notificación por correo pasado las cinco de la tarde para saber si es necesario hacer la ruta siempre y cuando: - El nivel es mayor o igual al 75% - Días laborables de lunes a viernes. • Caso contrario notificará que no es necesario realizar la ruta.

Capa de aplicación

En la capa de aplicación se tiene la interfaz donde se puede observar y monitorear en tiempo real los tres prototipos de gestión de servicios básicos. Donde se podrá observar los valores de voltaje, corriente, caudal, consumo de agua o luz y nivel de residuos además se tendrá gráficos históricos de los consumos diarios que se vayan dando. Para la distribución de las pantallas de tendrá tres secciones que son las siguientes:

Información: Es esta parte se tendrá todos los datos del medidor como puede ser el número de contrato, la persona dueña del medidor, la dirección donde se encuentre el medidor y los valores de consumo que se vaya teniendo.

Monitoreo: En esta parte constará de la visualización de las variables de los prototipos como el caudal, voltaje, corriente, consumos de agua o luz, niveles del contenedor de basura y gráficos históricos.

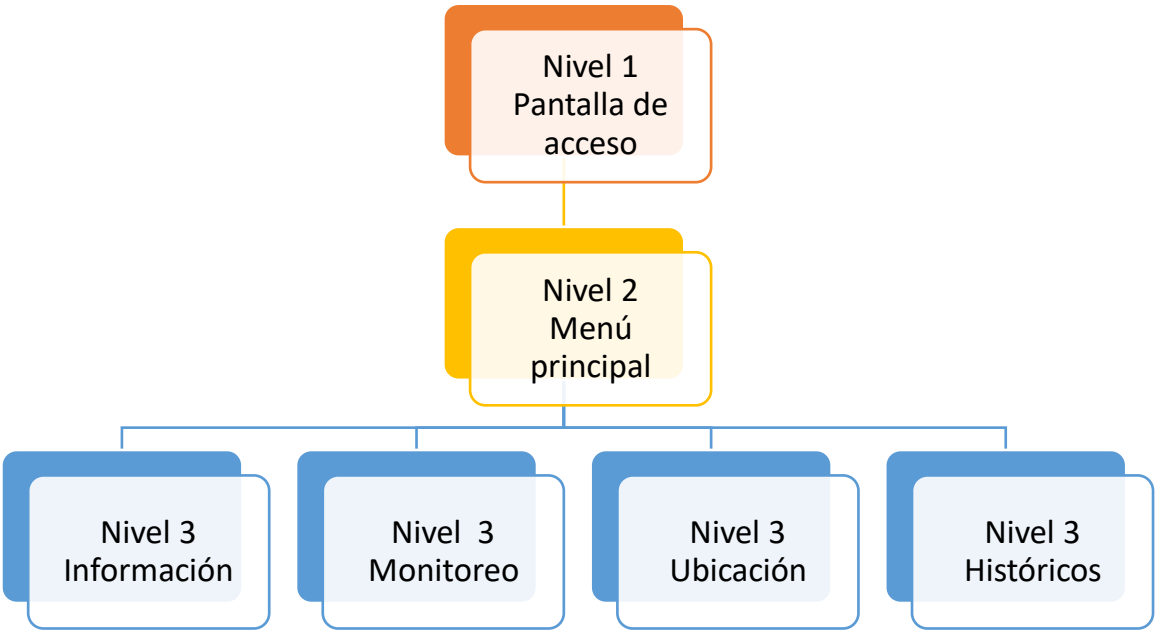
Ubicación: En esta parte se tendrá la visualización de la ubicación de los prototipos mediante Google Maps donde además se tendrá la información de las calles y las coordenadas de latitud y longitud para saber dónde exactamente se encuentren los prototipos.

Históricos: Gráficos del comportamiento de las variables monitoreadas a lo largo del tiempo.

Esta interfaz está realiza mediante node red ya que se encuentra suscrita a los tópicos con cada nombre característico enviado por el Gateway además esta interfaz podrá ser desplegada desde cualquier lugar ya que se encuentra instalada en un servidor privado en la nube. Las normativas recomiendan el uso de colores con tonos grises para evitar la fatiga visual, el diseño de la distribución de las pantallas será el mismo para todas las pantallas de los prototipos únicamente cambiará las variables que se encuentren monitoreando en la figura 22 se muestra la distribución jerárquica de las pantallas.

Figura 22

Distribución jerárquica de las pantallas en la interfaz



Capítulo IV. Implementación

Una vez formulado los requerimientos del diseño de la arquitectura de cada una de las capas del proyecto se procede a definir las herramientas y tecnologías utilizadas para cumplir con los objetivos del proyecto el cual es un sistema de gestión para los servicios básicos, en este capítulo se irá detallando mediante diagramas, librerías y configuraciones que se llevó a cabo para la implementación de la arquitectura en el cual engloba un conjunto de tecnologías como la utilización de LoRa, servidores web, almacenamiento de datos y monitoreo en tiempo real de los prototipos realizados.

Implementación de los prototipos

Para el desarrollo de los prototipos se ocupó distintas tecnologías que se detallarán a continuación, para cada uno de los prototipos se realizó una calibración de los sensores para que los valores mostrados sean lo más posible parecidos al valor real igualmente se detallará la programación para la conexión multipunto mediante Lora y la configuración del Gateway para recibir y enviar los datos a la nube.

Implementación prototipo consumo de agua

Calibración sensor de caudal

Para la calibración del sensor de caudal YF-S201 se calculó la constante k mediante la formula vista en el capítulo anterior por lo que se utiliza un recipiente con medidas en litros como patrón para la medición de cuantos pulsos emite el sensor cada cierto volumen y posteriormente a eso realizar un promedio con los datos obtenidos. En la figura 23 se observa el código utilizado en Arduino para la lectura de los pulsos.

Figura 23

Código calibración sensor de caudal

```
volatile long NumPulsos;
int PinSensor=2;

void ContarPulsos()
{
  NumPulsos++;
  }
void setup() {
  Serial.begin(9600);
  pinMode(PinSensor,INPUT);
  attachInterrupt(0,ContarPulsos,RISING);
  interrupts();
}
void loop() {
 Serial.print("Número de Pulsos = ");
 Serial.println(NumPulsos);
 delay(100);
}
```

Para realizar las pruebas se utiliza mediciones con cinco y nueve litros para cada una de estas se realiza seis lecturas para obtener así el número de pulsos en el monitor serial como se observa en las figuras 24 y 25 respectivamente.

Figura 24

Numero de pulsos para 5 litros

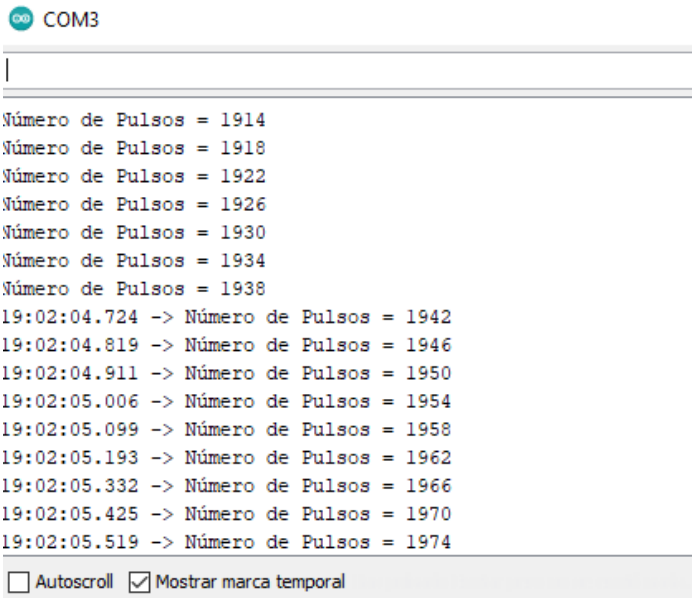
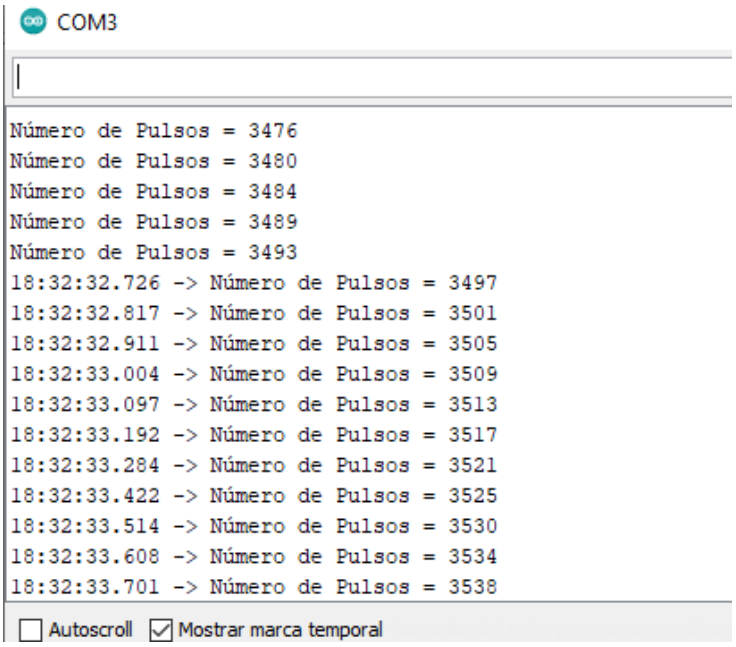


Figura 25

Numero de pulsos para 9 litros



En la tabla 13 se muestra todos los valores obtenidos de cada una de las mediciones realizadas con lo que para cada valor se obtiene la constante K. Finalmente sacamos un promedio de este valor K para posteriormente colocar en la programación del prototipo para obtener un mínimo error en las mediciones.

Tabla 13

Cantidad de pulsos por el sensor de flujo para cada prueba

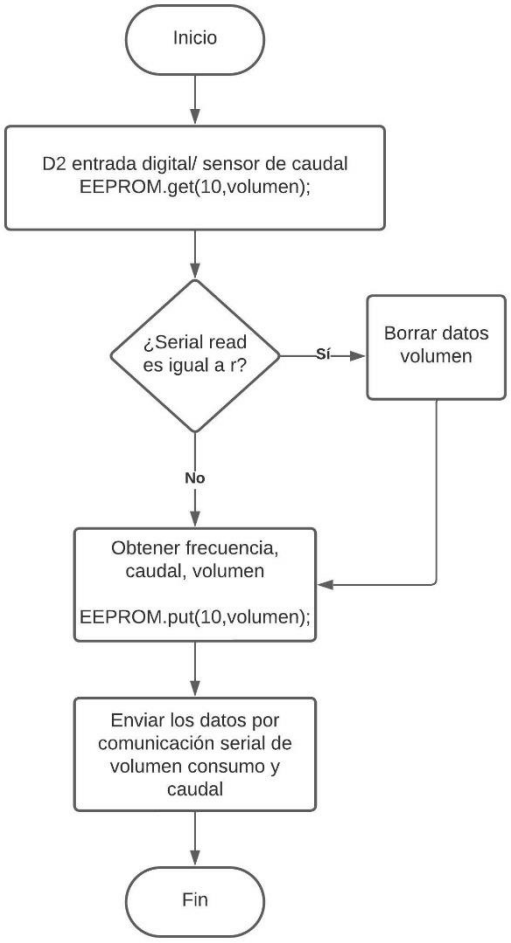
Volumen	Cantidad	N. Pulsos	K
Litros	9	3534	6,544444444
Litros	9	3497	6,475925926
Litros	9	3568	6,607407407
Litros	9	3554	6,581481481
Litros	9	3530	6,537037037
Litros	9	3532	6,546056053
Litros	5	1942	6,473333333
Litros	5	1948	6,488888847
Litros	5	1946	6,486666667
Litros	5	1949	6,496666667
Litros	5	1951	6,503333333
Litros	5	1949	6,496666667
Promedio			**6,520296296**

Diagrama de flujo para la programación del prototipo

En la figura 26 se muestra el diagrama de flujo utilizado para la toma de datos mediante el Arduino nano ya que toda la gestión de los servicios básicos está realizada en el servidor en la nube, los prototipos únicamente están encargados de la toma de datos y guardar en la memoria EEPROM el valor del consumo para que en el caso si es que se va la energía el valor del consumo se mantenga para finalmente enviarlo por comunicación serial al Esp32 LoRa y este se encargue del envío de los datos al Gateway.

Figura 26

Diagrama flujo consumo de agua



Programación Arduino nano consumo de agua

En la parte de la programación se incluye la librería EEPROM para más adelante poder guardar los valores de volumen y se crea e inicializa las variables que intervienen en la programación también se coloca el factor de conversión k obtenido en la etapa de calibración del sensor como se observa en la figura 27.

Figura 27

Inclusión de librerías y creación de variables

```
#include <EEPROM.h>
volatile int NumPulsos;
int PinSensor=2;
float factor_conversion=6.5203;
//int ResetAgua=LOW;
float volumen = 0;
float volumenm3=0;
float consumo=0;
long dt=0;
long t0=0;
```

En la figura 28 se observa la creación de dos funciones la uno es para aumentar el número de pulsos en la variable de la frecuencia y la función Obtfrecuencia () para obtener la frecuencia mediante interrupciones, el puerto D2 del Arduino nano permite realizar interrupciones por esa razón se coloca el sensor en este pin.

Figura 28

Creación de funcionas para obtener la frecuencia

```
void ContarPulsos()
{
  NumPulsos++;
  }

int Obtfrecuencia()
{
  int frecuencia;
  NumPulsos=0;
  interrupts();
  delay(1000);     // muestras en un segundo
  noInterrupts();
  frecuencia=NumPulsos;    //Hz pulsos por segundo
  return frecuencia;

  }
```

Finalmente, se llama las funciones creadas anteriormente para aplicar las fórmulas vistas en la parte de diseño para la obtención del caudal y el volumen en metros cúbicos para

al final enviar en una cadena de caracteres por comunicación serial los valores de caudal, volumen y el consumo como se muestra en la figura 29.

Figura 29

Obtención de las variables y envió por comunicación serial

```
void loop()
{
  if (Serial.available()) {
    if(Serial.read()=='r')volumen=0;//restablecemos el volumen si recibimos 'r'
  }

 float frecuencia=Obtfrecuencia();
 float caudal_L_m=frecuencia/factor_conversion;
 dt=millis()-t0;
 t0=millis();
 volumen= volumen+(caudal_L_m/60)*(dt/1000);
 volumenm3=volumen*0.001;
 consumo=volumenm3*0.31;
 EEPROM.put(10,volumen);

String Sl = String(caudal_L_m) + "/" + String(volumenm3) + "/" + String(consumo);
Serial.println(Sl);
 delay(1000);
}
```

Ensamblado del prototipo de consumo de agua

El prototipo de consumo agua tiene unas dimensiones de 110x110x70 mm, además cuenta con una protección IP55 que según la norma IEC 60529 tiene una protección moderada frente al polvo y agua. Los elementos utilizados cumplen con todas las características vistas en el capítulo anterior en la parte de diseño en la figura 30 se tiene una vista interior del prototipo con todos los elementos mencionados y en la figura 31 se muestra la implementación final del prototipo por la parte de afuera donde se puede apreciar la pantalla Oled para la presentación de los datos en tiempo real y su puerto tipo C para la alimentación a 5V de este prototipo.

Figura 30

Ensamblado parte interior prototipo consumo de agua

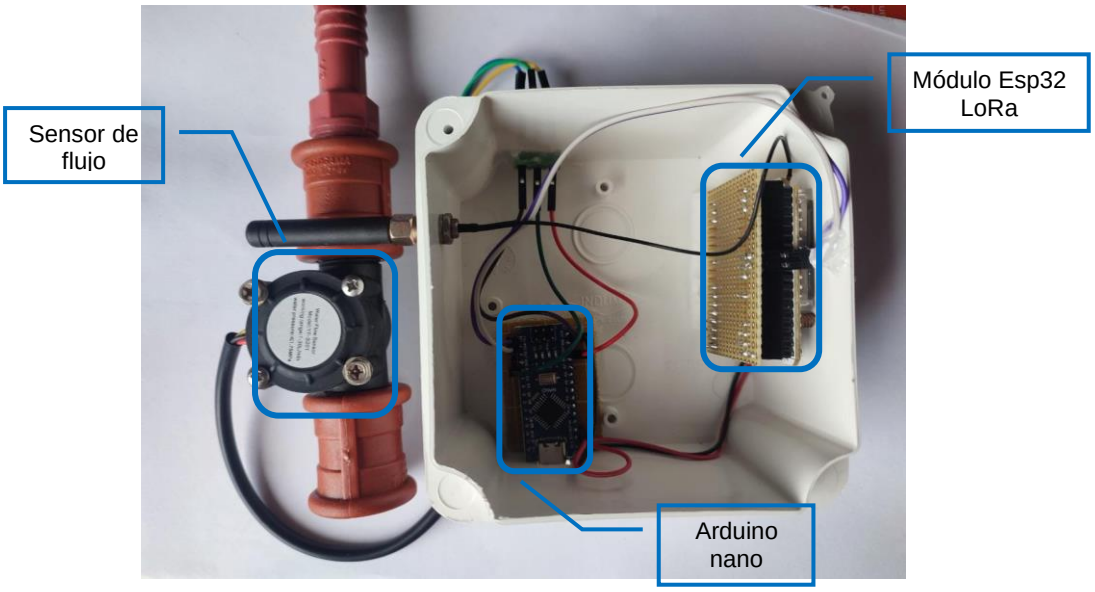


Figura 31

Ensamblado parte exterior prototipo consumo de agua

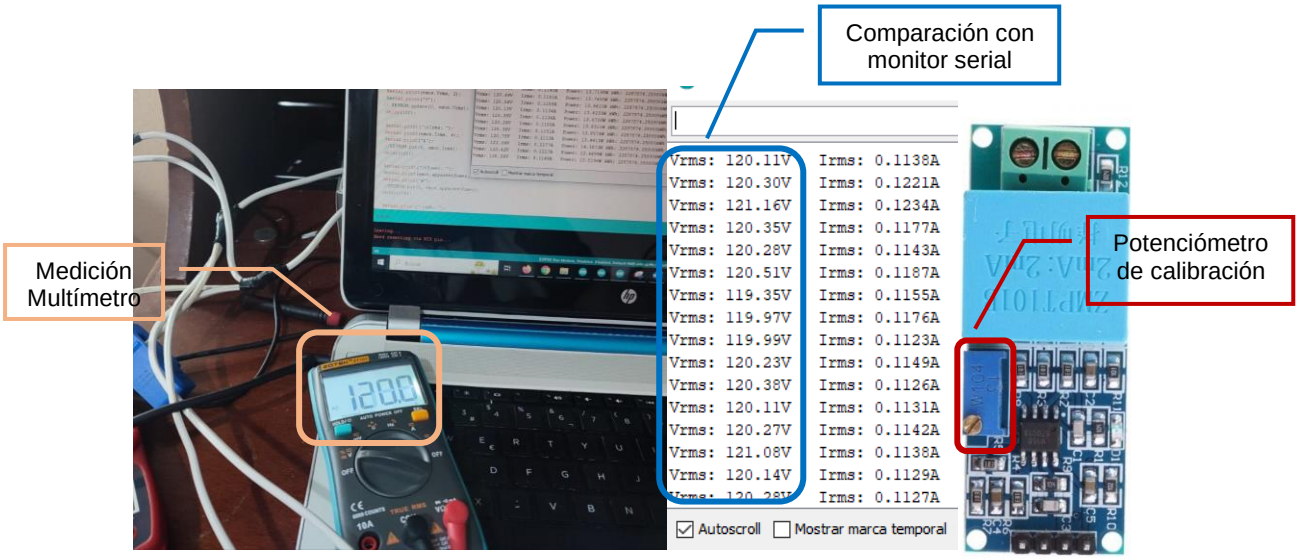
Implementación prototipo consumo de luz

Calibración sensor de voltaje y corriente

Para la calibración del sensor de voltaje Zmpt101b se debe mover el potenciómetro hasta que coincida el valor de voltaje medido en el multímetro con el valor que se muestra en el monitor serie del Arduino como se muestra en la figura 32.

Figura 32

Calibración del sensor de voltaje



En el sensor de corriente SCT-013 después de realizar el circuito de acondicionamiento se tiene una notable variación en las mediciones de corriente por lo cual se realiza una calibración mediante software por lo que se aplica un filtro promedio tomando diez datos de la entrada analógica A1 y obteniendo el promedio antes de realizar cualquier cálculo como se observa en la figura 33.

Figura 33

Filtro promedio para sensor de corriente SCT-013

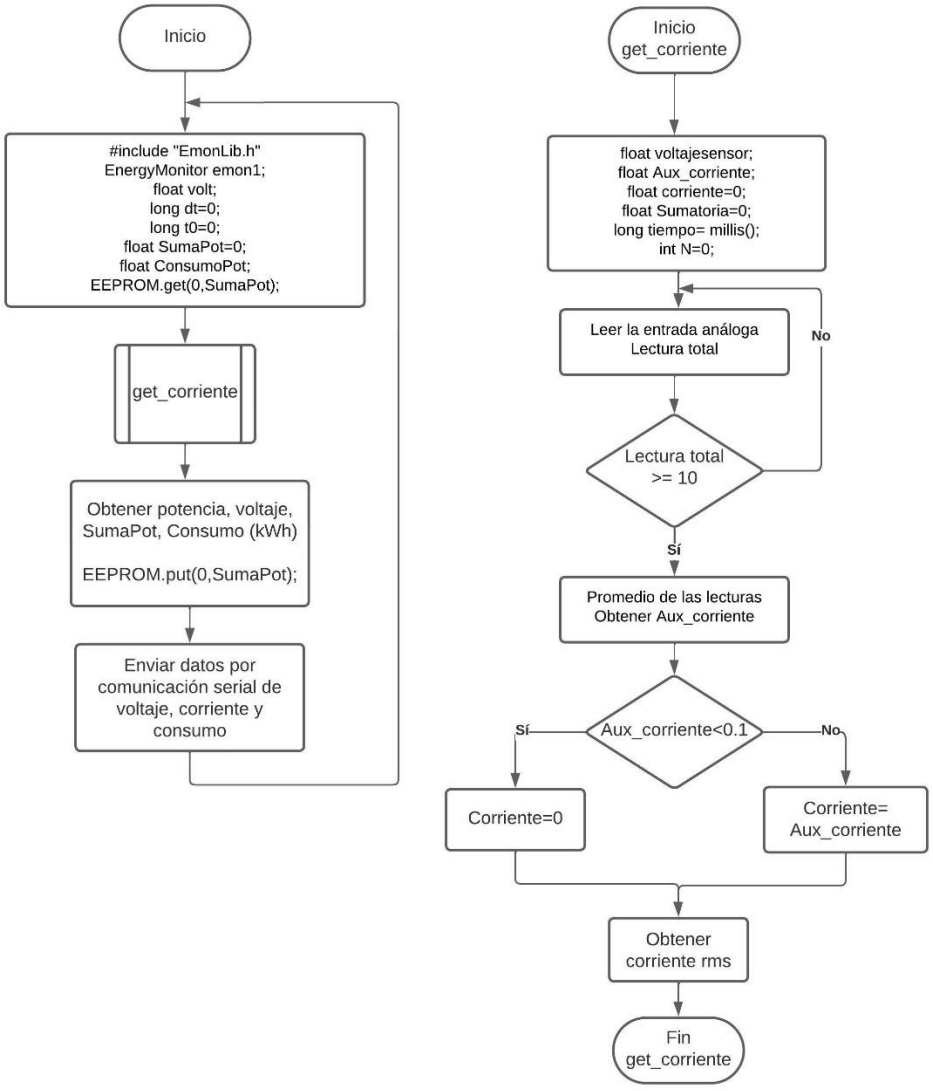
```
Total = Total - Lecturas[i];
//Agrega una lectura a la posición actual dentro del vector
Lecturas[i] = analogRead(Al);
//Realiza la sumatoria entre lecturas
Total = Total + Lecturas[i];
 i = i + 1;

  if (i >= 10){
  i = 0;
  Promedio = Total / 10;
```

Diagrama de flujo para la programación del prototipo

En la figura 34 se muestra el diagrama de flujo el cual indica el proceso llevado para para la toma de datos mediante el Arduino nano, en esta parte se tiene la función get_corriente() en la cual la sumatoria de los cuadrados para el número de muestras nos da la corriente para con esto obtener la potencia y el consumo en kWh para finalmente guardar en la memoria EEPROM el consumo para evitar que se borre el dato en el caso de un corte de luz.

Figura 34

Diagrama de flujo consumo de luz



Programación Arduino nano consumo de luz

En la parte de la programación se incluye las librerías EEPROM y EmonLib para guardar los valores de consumo y obtener el valor de voltaje rms por medio del sensor Zmpt101b, se instancia las variables que intervienen en la programación como se observa en la figura 35.

Figura 35

Inclusión de librerías y variables a utilizar

```
  #include <EEPROM.h>
//Librería de Monitor Energético
#include "EmonLib.h"
EnergyMonitor emonl;             // Instancia de la librería EmonLib
float SumaPot=0;
float ConsumoPot;
long dt=0;
long t0=0;
int Lecturas[10]; //Vector de lecturas.
int Val, i = 0, Total = 0, Promedio = 0;
float vrms;
```

En la figura 36 se tiene la programación de la función get_corriente(), donde se procesa la señal que envía el sensor SCT-013, para el caso del Arduino nano en las entradas analógicas admite un voltaje de hasta 5V y una resolución de 10 bits (1024 niveles), es por eso que se coloca el voltaje dividido para la resolución del microcontrolador en este caso 1023, posterior a eso se realiza la suma de los cuadrados de la corriente y se obtiene su valor rms.

Figura 36

Función get_corriente

```
float get_corriente()
{
while(millis()-tiempo<1000) // duracion 0.5 segundos (30 ciclos de 60 hz)
  {
      voltajesensor=Promedio*(5.0/1023.0)-2.502;
       Aux_corriente=voltajesensor*56;// voltaje por por factor de conversion
     if(Aux_corriente < 0.1 )// para eliminar el ruido
     {
      corriente=0;

      }
       else
      {
      corriente=Aux_corriente;
      }
      Sumatoria=Sumatoria+ sq(corriente); //sum cuadrados
      N = N+1;
      delay(1);
     }
    }
    Sumatoria= Sumatoria*2;     // RMS onda completa
    corriente= sqrt((Sumatoria)/N); //ecuacion del RMS
    return(corriente);
  }
```

Finalmente se tiene el tiempo de procesamiento del Arduino con la función millis() la cual devuelve el tiempo en milisegundos desde que la placa se encendió o empezó a ejecutar algún programa, con esto nos permite obtener el consumo en kWh para al final guardar en la memoria EEPROM y enviar por el serial los valores de voltaje, corriente y consumo como se observa en la figura 37.

Figura 37

Valores de consumo de luz y envío serial de los datos

```
  dt=millis()-t0;
  t0=millis();
  SumaPot=SumaPot+P*(dt/1000);
  ConsumoPot= (SumaPot)/3600000; //consumo en kWh

  EEPROM.put(0,SumaPot);

String Sl = String(Vrms) + "/" + String(Irms) + "/" + String(ConsumoPot,6);

Serial.println(Sl);
delay(1000);
```

Ensamblado del prototipo de consumo de luz

El prototipo de consumo de luz tiene unas dimensiones de 95x150x50 mm. Los elementos utilizados cumplen con todas las características vistas en el capítulo anterior en la parte de diseño, en la figura 38 se muestra una vista interior del dispositivo con la interconexión de todos sus elementos mientras que en la figura 39 se muestra la implementación final del prototipo en la parte exterior con su pantalla Oled para la visualización en tiempo real de los valores de voltaje, corriente y consumo en kWh. Además, cuenta con una entrada Jack 3.5 mm para la facilidad de conexión del sensor de corriente y su puerto tipo C para la alimentación a 5V de este prototipo

Figura 38

Implementación parte interior prototipo consumo de luz

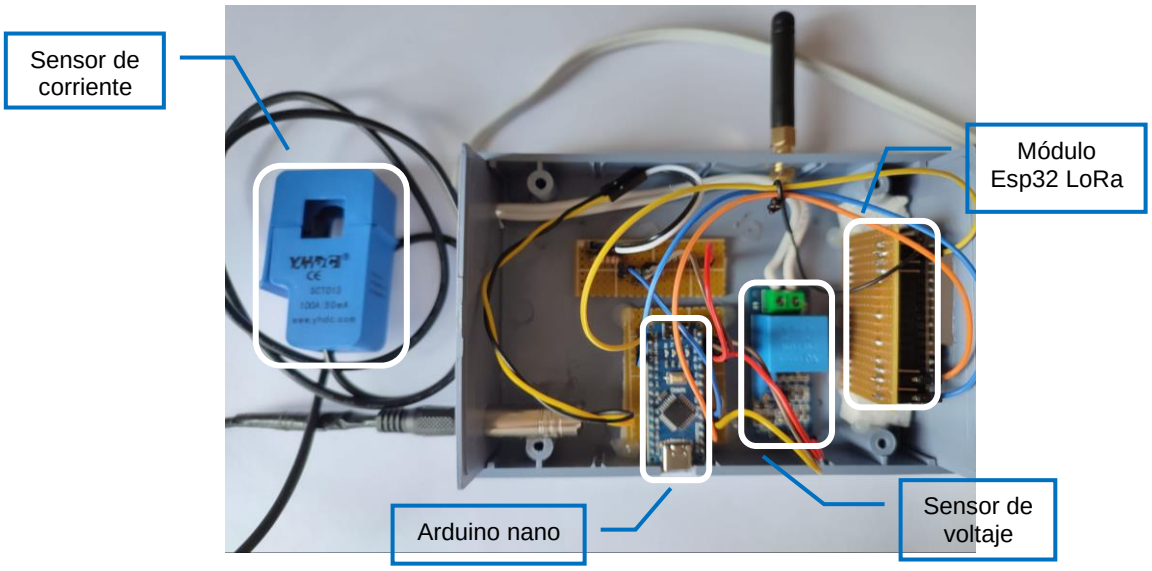
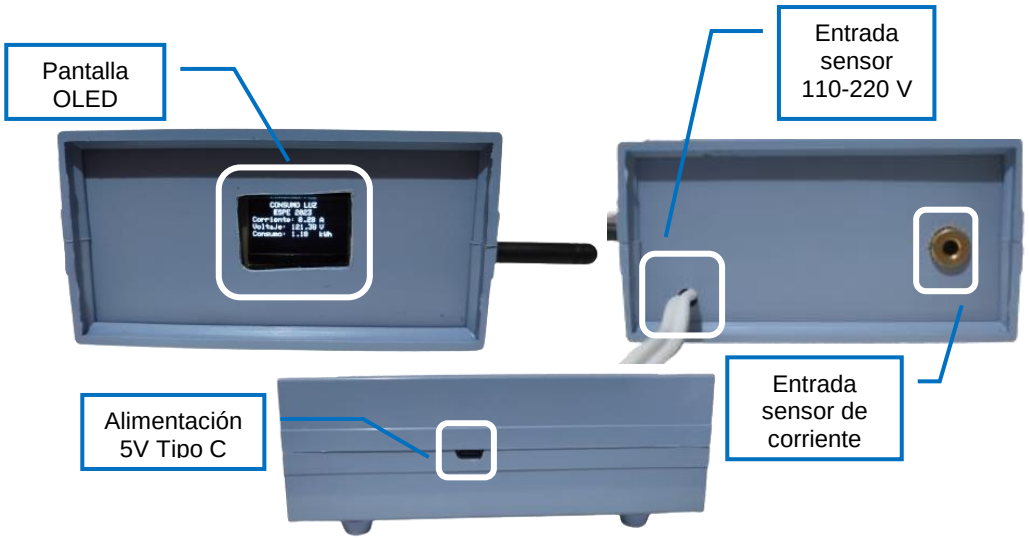


Figura 39

Implementación parte exterior prototipo consumo de luz



Implementación tarjetas Esp32 LoRa

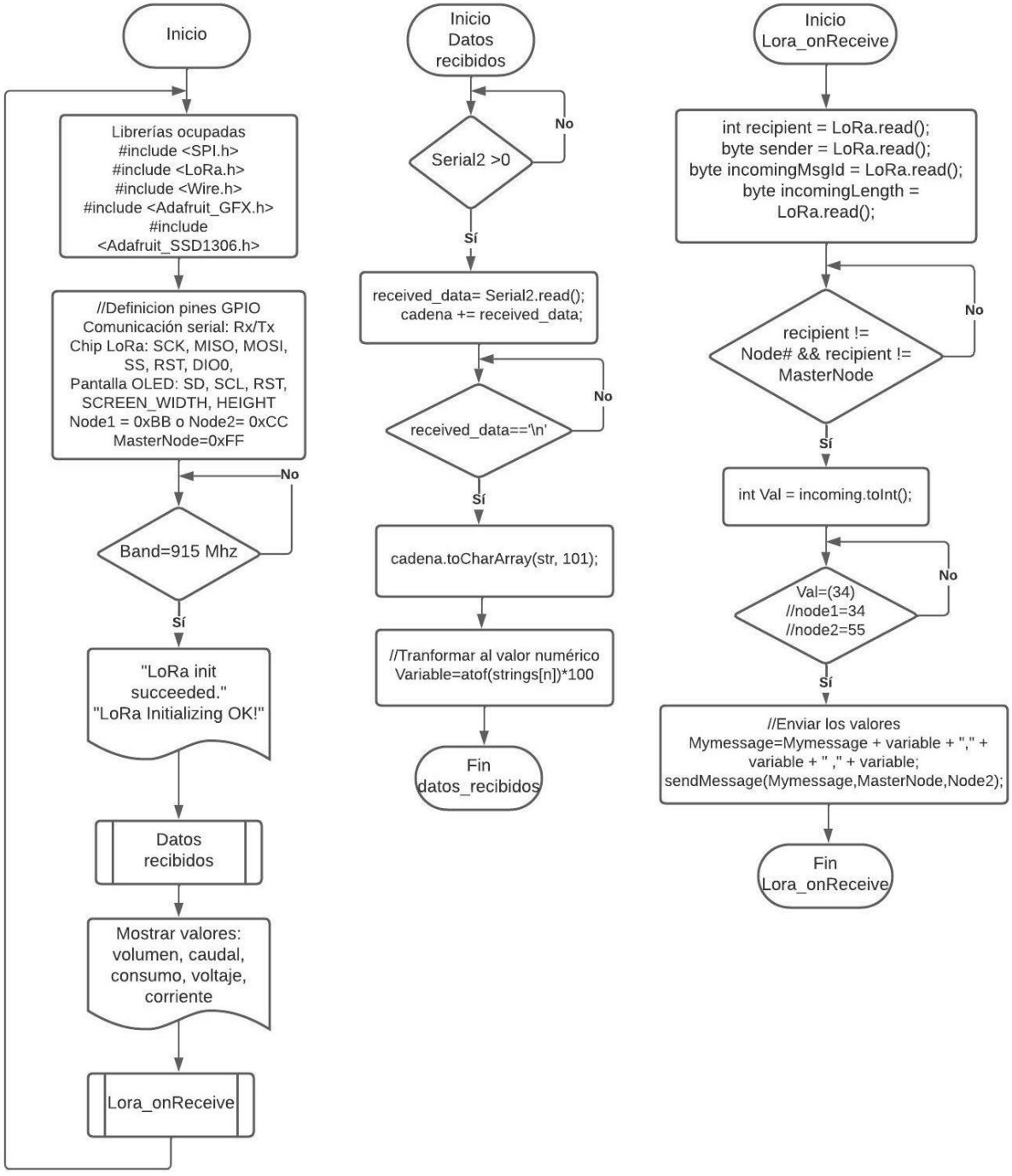
En la implementación de las tarjetas Esp32 con conexión a LoRa para la programación y configuración en los prototipos de agua y luz se tiene la misma, únicamente cambia las variables que reciben por parte del Arduino nano que se detalló anteriormente su funcionamiento. Estas tarjetas Esp32 tiene dos funcionalidades en el proyecto, la una es enviar los datos al Gateway o concentrador mediante la conexión inalámbrica LoRa y la otra aprovechar su pantalla OLED para la presentación de los datos en las cajas de los prototipos, para que el usuario pueda observar en tiempo real el monitoreo de las variables.

En la figura 40 se detalla el funcionamiento de esta tarjera mediante un diagrama de flujo, donde se tiene las siguientes partes:

- Inicialización de los pines del chip LoRa SX1276 y de la comunicación serial.
- Asignación de las direcciones para los prototipos de agua, luz y nodo concentrador donde se tiene las siguientes direcciones Node1=0xBB y Node2=0xCC y MasterNode=0xFF respectivamente.
- Decodificación de los datos que llegan por comunicación serial mediante los pines RX/TX.
- Mostrar en la pantalla OLED los datos recibidos y envío de los datos al nodo concentrador mediante la comunicación inalámbrica LoRa.

Figura 40

Diagrama de funcionamiento del Esp32 para el consumo de agua y luz



Programación Esp32 para los prototipos de agua y luz

En la programación de los prototipos se utiliza el entorno Arduino en ambos módulos la programación es idéntica, únicamente cambia las direcciones y variables a enviar al nodo concentrador. Para explicar el código se utiliza la programación del prototipo de luz como se observa en la figura 41 se inicializa los puertos del chip, los puertos de la comunicación serial y los puertos de la pantalla OLED. La ventaja de estos módulos es que cualquier GPIO se le puede asignar los puertos Tx/Rx, se asignó unos puertos diferentes debido a que la pantalla OLED ocupa los puertos asignados por el fabricante para la comunicación serial.

Figura 41

Inicialización de los puertos e inclusión de librerías

```
//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
//define lora and display
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26
#define BAND 915E6
#define RXp2 13
#define TXp2 12
//OLED pins
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

Se asigna las direcciones del nodo concentrador o master y la dirección del nodo del prototipo. Además, se coloca las variables que se utilizan en el prototipo como se muestra en la figura 42.

Figura 42

Declaración de variables

```
//variables lora
String outgoing;              // outgoing message
byte msgCount = 0;            // count of outgoing
byte MasterNode = 0xFF;
byte Node1 = 0xBB;

String Mymessage = "";
String incoming = "";

String cadena;
char *strtok(char *str1, const char *str2);
char *resultado = NULL;
char str[100];
char received_data;
char strings[5][10]={0};

float Vrms, Irms, power, consumo;
int Vrms1, Irms1, consumo1;
float Vrms12, Irms12, consumo12;
int cont = 0;
```

La ventaja de estos módulos es que cualquier GPIO se le puede asignar los puertos Tx/Rx, se asignó unos puertos diferentes debido a que la pantalla OLED ocupa los puertos designados por el fabricante como se muestra en la figura 43 para la comunicación serial.

Figura 43

Inicialización del segundo puerto serial

```
void setup()
{
   Serial.begin(115200,SERIAL_8N1);


 Serial2.begin(9600,SERIAL_8N1,RXp2,TXp2);

  //reset OLED display via software
pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);
```

En la figura 44 se muestra la función para la decodificación de los datos enviados serialmente por el arduino nano, donde la posición de las variables en el string tiene el mismo orden que se envió desde el arduino nano.

Figura 44

Función para decodificación de los datos enviados serialmente

```
if(Serial2.available()>0){
      delay(2);
      received_data= Serial2.read();
      cadena += received_data;
      if(received_data == '\n'){
        cadena.toCharArray(str, 101);
        char lim[] = "/";
        char *resultado = NULL;
        resultado = strtok (str, lim);
        while(resultado != NULL){
          strcpy(strings[index],resultado);
          index++;
          resultado = strtok(NULL, lim);
        }
        Vrms = atof(strings[0])*100;
        Irms = atof(strings[1])*100;
        consumo = atof(strings[2])*1000000;
        cadena = "";

        Vrms12 = Vrms/100;
        Irms12 = Irms/100;
        consumo12 = consumo/1000000;
```

Una vez obtenidos los valores se muestra en pantalla y se envían al nodo concentrador mediante la función onReceive como se muestra en la figura 45.

Figura 45

Visualización de los datos en pantalla y envío de los datos al nodo concentrador

```
display.clearDisplay();
 display.setCursor(25,0);
 display.println("CONSUMO LUZ");
 display.setCursor(30,10);
 display.setTextSize(1);
 display.print("ESPE 2023");
 display.setCursor(0,20);
 display.print("Corriente:");
 display.setCursor(65,20);
 display.print(Irms12);
 display.setCursor(95,20);
 display.print("A");
 display.setCursor(0,30);
 display.print("Voltaje:");
 display.setCursor(55,30);
 display.print(Vrms12);
 display.setCursor(95,30);
 display.print("V");

onReceive(LoRa.parsePacket());
```

Por último, se tiene la función para envío de los datos por LoRa en la cual el nodo concentrador tiene un valor predeterminado para cada nodo por ejemplo para el prototipo de luz que es el nodo 1 se tiene un valor de 34 para el nodo 2 un valor de 55, si es igual a este valor se procederá a enviar los datos como se muestra en la figura 46.

Figura 46

Función para envío de los datos mediante LoRa

```
if (recipient != Node1 && recipient != MasterNode) {
  ;
  return;
}
  Serial.println(incoming);
  int Val = incoming.toInt();
  if(Val == 34)
  {
  Mymessage=String(Mymessage + Vrms12 + "," + Irms12 + "," + consumo12);
  sendMessage(Mymessage,MasterNode,Node1);
  delay(100);
  Mymessage = "";
  }
```

Implementación Prototipo nivel de residuos

Para el prototipo de nivel de residuos se utiliza únicamente la tarjeta Esp32, la cual se encargará del procesamiento de los datos del sensor y el envío de la información al Gateway mediante comunicación inalámbrica LoRa. Únicamente se ocupa este microcontrolador ya que no es necesario que esté midiendo el prototipo constantemente las 24 horas del día, sino que es más que necesario que mida cada cierto tiempo, con esto reducimos costos y medidas del prototipo.

Diagrama de flujo del prototipo de residuos

En la figura 47 y 48 se detalla el funcionamiento del prototipo mediante un diagrama de flujo en el cual consta la inclusión de las librerías, inicialización del chip lora SX1276 y los pines de la pantalla OLED. Además, se tiene la asignación de la dirección del prototipo que es Node3= 0xDD y la dirección del nodo concentrador que es MasterNode=0xFF a donde se enviarán los datos mediante comunicación inalámbrica lora.

Figura 47

Diagrama de flujo función principal e indicador de nivel del prototipo de residuos

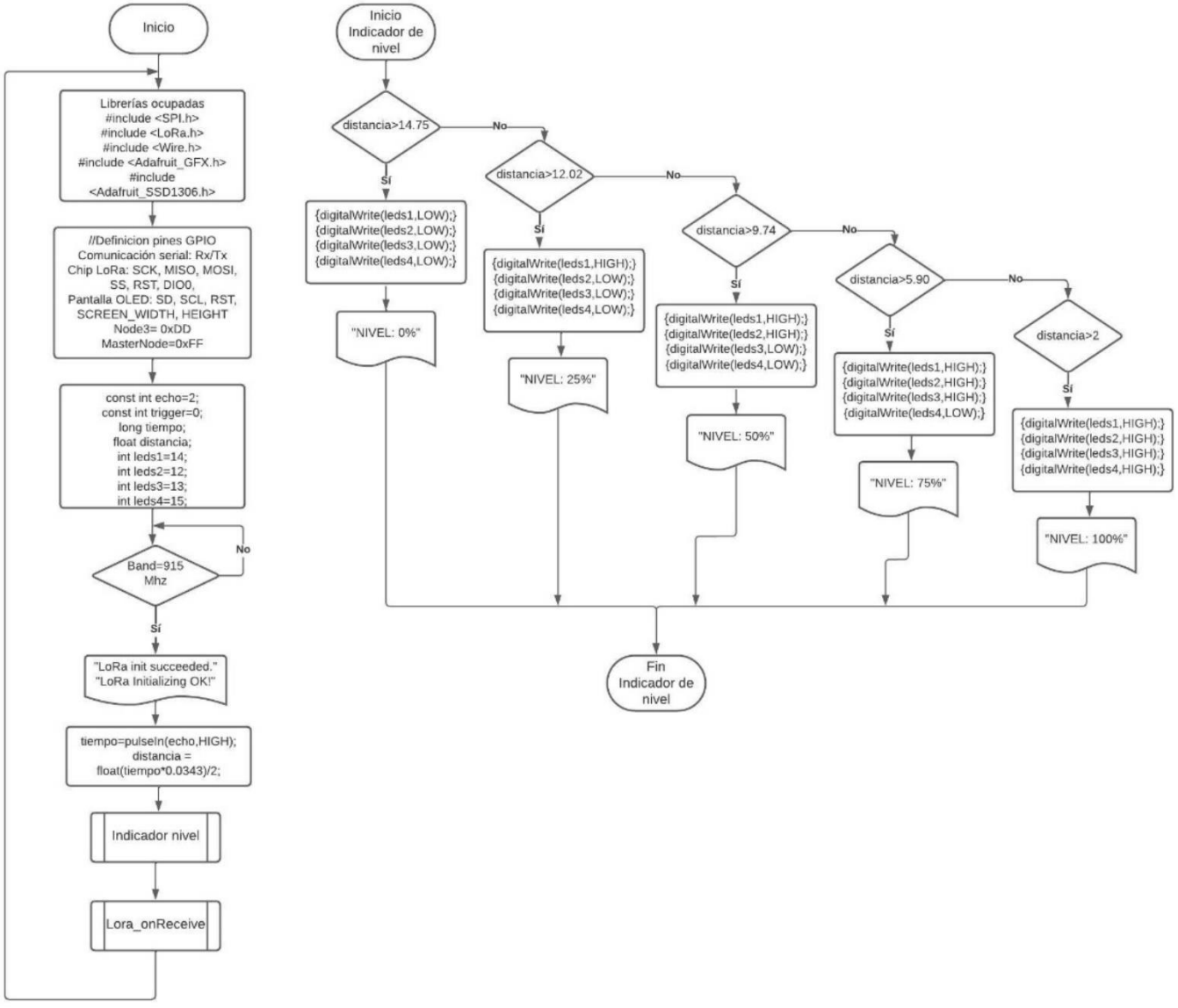
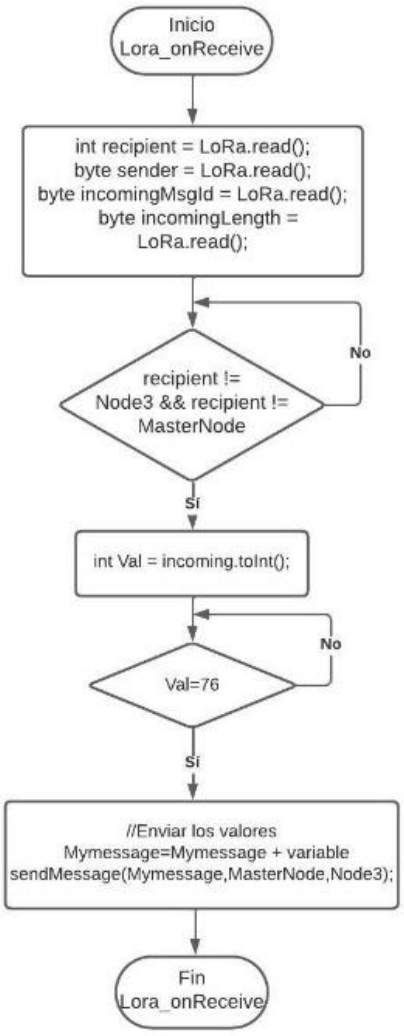


Figura 48

Diagrama de flujo prototipo de residuos función envío de datos por LoRa



Programación prototipo nivel de residuos

Se inicializa los puertos para el chip lora, pantalla oled y se incluye las librerías como se muestra en la figura 49.

Figura 49

Inicialización de puertos prototipo de residuos

```
//Libraries for OLED Display
#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
//OLED pins
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 12
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
```

En la figura 50 se muestra la declaración de las variables que intervienen en la programación del prototipo y también se coloca la dirección del nodo master y la dirección del prototipo que es Node3.

Figura 50

Declaración de variables del prototipo de residuos

```
 const int echo=2;
const int trigger=0;
long tiempo;
float distancia;
int leds1=14;
int leds2=12;
int leds3=13;
int leds4=15;
//variables lora
String outgoing;
byte msgCount = 0;
byte MasterNode = 0xFF;
byte Node3 = 0xDD;
```

En la figura 51 se muestra la función medir que calcula la distancia mediante los pines trigger y echo del sensor ultrasónico para finalmente la función onReceive enviar los datos al nodo concentrador.

Figura 51

Cálculo de la distancia y envío de los valores

```
void loop()
{
  medir();

 Serial.print("Distancia: ");
  Serial.print(distancia);      //I
  Serial.print("cm");
  Serial.println();
onReceive(LoRa.parsePacket());
void medir()
{
  digitalWrite(trigger,LOW);
  delayMicroseconds(2);
  digitalWrite(trigger,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger,LOW);

  tiempo=pulseIn(echo,HIGH);//us=microsegundos
  distancia = float(tiempo*0.0343)/2;
```

Para el envío de los datos mediante lora al igual que en los prototipos anteriores el nodo master envía un valor predeterminado que en este caso 76 para saber si es ese prototipo como se muestra en la figura 52.

Figura 52

Envío de los datos del prototipo de residuos

```
if (recipient != Node3 && recipient != MasterNode) {
  ;
  return;
}
  Serial.println(incoming);
  int Val = incoming.toInt();
  if(Val == 76)
  {
indicador_nivel();
  Mymessage= String(distancia);
  sendMessage(Mymessage,MasterNode,Node3);
  delay(100);
  }
```

Finalmente se tiene la función indicadora de nivel donde se encienden los leds del prototipo y se muestra en la pantalla los valores de nivel del basurero. En la figura 53 se muestra dos casos de la función.

Figura 53

Función indicador de nivel prototipo de residuos

```
void indicador_nivel()
{
  if(distancia>14.75) //distancia maxima
  {
   {digitalWrite(leds1,LOW);}
   {digitalWrite(leds2,LOW);}
   {digitalWrite(leds3,LOW);}
   {digitalWrite(leds4,LOW);}

 display.clearDisplay();
    display.setCursor(30,0);
    display.setTextSize(1);
  display.println("RESIDUOS");
  display.setCursor(30,10);
  display.print("ESPE 2023");
  display.setTextSize(2);
  display.setCursor(0,30);
   display.print("NIVEL: 0%");
  display.display();
```

```
  else if(distancia>12.02) // primer nivel
    {
     {digitalWrite(leds1,HIGH);}
     {digitalWrite(leds2,LOW);}
     {digitalWrite(leds3,LOW);}
      {digitalWrite(leds4,LOW);}

     display.clearDisplay();
     display.setTextSize(1);
     display.setCursor(30,0);
   display.println("RESIDUOS");
   display.setCursor(30,10);
   display.print("ESPE 2023");
   display.setTextSize(2);
   display.setCursor(0,30);
    display.print("NIVEL: 25%");
   display.display();
```

Ensamblado del prototipo nivel de residuos

El prototipo de nivel de residuos tiene unas dimensiones de 110x110x70 mm, además cuenta con una protección IP55 que según la norma IEC 60529 tiene una protección moderada frente al polvo y agua. Los elementos utilizados cumplen con todas las características vistas en el capítulo anterior en la parte de diseño, en la figura 54 se muestra una vista de la parte interior con la interconexión de todos los elementos mencionados anteriormente mientras que en la figura 55 se muestra el ensamblado en la parte exterior del prototipo donde se observa su pantalla Oled en la cual se visualiza los niveles en porcentaje del tacho y los indicadores led para conocer a simple vista en qué nivel se encuentra los residuos. En la sección APÉNDICES, con el nombre “A: Manual de Usuario” se tiene todas las consideraciones utilizadas para los prototipos.

Figura 54

Implementación parte interior prototipo nivel de residuos

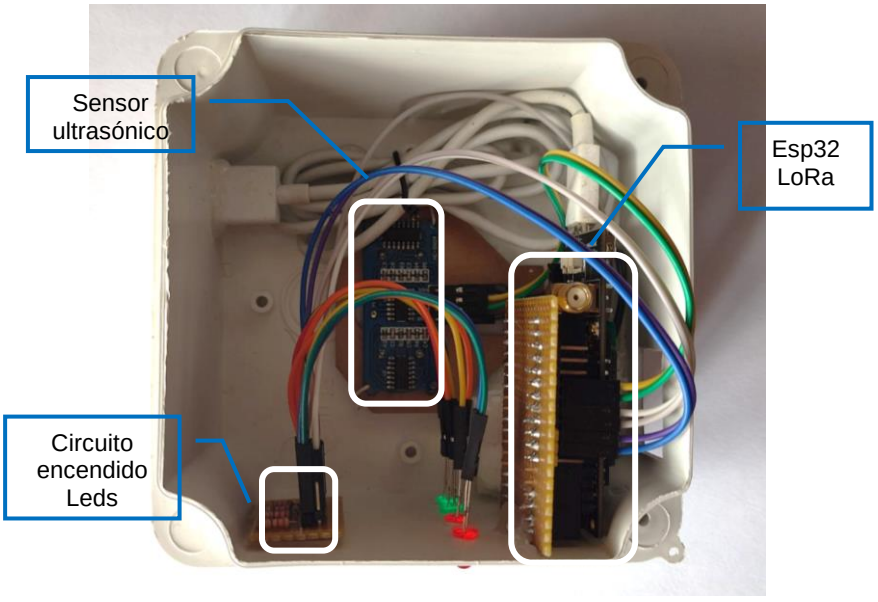


Figura 55

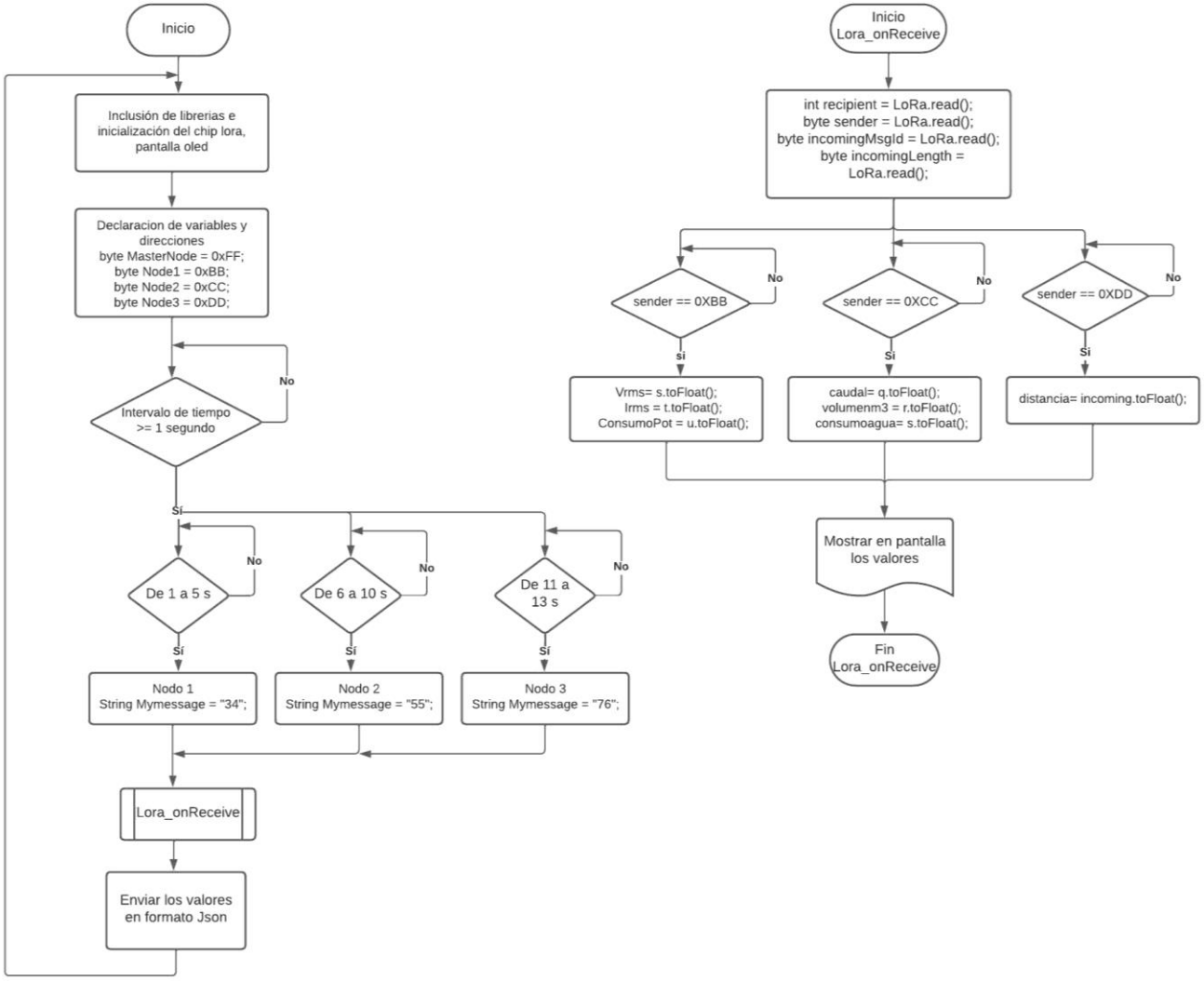
Implementación parte exterior prototipo nivel de residuos

Implementación Gateway

En la implementación del gateway o concentrador se utiliza un Esp32 Lora de la marca Ttgo, en el cual llegan los datos de los tres nodos donde el Node1 es el prototipo de luz, Node 2 el prototipo de agua y Node 3 prototipo de residuos. Este nodo concentrador una vez recibido los datos los envía al broker Mosquitto mediante comunicación mqtt. A continuación, mediante el diagrama de flujo de la figura 56 se indica el funcionamiento del gateway.

Figura 56

Diagrama de flujo para el funcionamiento del gateway



Programación gateway

Se presenta a continuación porciones de código donde se muestra las distintas partes de la programación como por ejemplo las librerías e inicialización de puertos que se han colocado en la programación como se muestra en la figura 57.

Figura 57

Librerías y definición de los puertos

```
#include <ArduinoJson.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <SPI.h>
#include <LoRa.h>
//Libraries for OLED Display
#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 14
#define DIO0 26

//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 915E6
```

Como se muestra en la figura 58, es importante colocar las direcciones de los nodos de los cuales se va a recibir la información ya que con esto podemos direccionar la información que ingrese al gateway y no haya problemas al envío de los datos.

Figura 58

Direcciones de cada nodo

```
byte MasterNode = 0xFF;
byte Node1 = 0xBB;
byte Node2 = 0xCC;
byte Node3 = 0xDD;
```

Se asigna los datos para conectarse a la red local donde sea realizará las pruebas y también la dirección IP para que puede suscribirse al broker de mosquitto como se muestra en la figura 59.

Figura 59

Datos para conectarse a la red local

```
//******MQTT****
const char* ssid = "XTRIM_LUIS YANEZ";
const char* password = "1723012561@";
const char* mqtt_server = "192.168.1.22";
WiFiClient esp32;
PubSubClient client(esp32);

char data_vrms[12]="";
char data_irms[12]="";
char data_consumo[12]="";
char data_caudal[12]="";
char data_volumen[12]="";
char data_consumoagua[12]="";
String messageData;
```

En la figura 60 se muestra cada que tiempo el gateway tomará los datos de cada nodo, en esta parte el gateway envía el valor característico de cada nodo como se ha visto anteriormente por ejemplo para el nodo 1 es el 34, el nodo el 55 y el nodo 3 el 76. Aquí se observa que el nodo 1 y 2 tomará datos en un tiempo de 5 segundos individualmente y el nodo 3 tomará datos en un tiempo de 3 segundos y así sucesivamente se repetirá la recepción de los datos.

Figura 60

Tiempos de recepción de los datos de cada nodo

```
 if ( Secs >= 14 )
{
  Secs = 0;
}
if ( (Secs >= 1) && (Secs <= 5) )
{
String Mymessage = "34";
sendMessage(Mymessage,MasterNode, Node1);
Mymessage = "";
}
    if ( (Secs >= 6 ) && (Secs <= 10))
{
String Mymessage = "55";
sendMessage(Mymessage,MasterNode, Node2);
Mymessage = "";
}
   if ( (Secs >= 11) && (Secs <= 13))
{
String Mymessage = "76";
sendMessage(Mymessage,MasterNode, Node3);
Mymessage = "";
}
```

En esta parte se tiene el envío de los datos hacia el broker mosquitto en formato Json ya que se facilita en node red identificar cada variable para ser tratados posteriormente como se muestra en la figura 61.

Figura 61

Envío de los datos en formato Json

```
StaticJsonDocument<192> doc;

doc["Dispositivo"] = "ESP32";
doc["Anho"] = 2023;
doc["Empresa"] = "ESPE";
doc["caudal"] = caudal;
doc["volumen"] = volumenm3;
doc["consumoagua"] = consumoagua;
 doc["voltaje"] = Vrms;
doc["corriente"] = Irms;
doc["consumoluz"] = ConsumoPot;
doc["distancia"] = distancia;

String output;

serializeJson(doc, output);
Serial.print("Publish message: ");
Serial.println(output);
client.publish("tesis/esp32", output.c_str());
```

Para la recepción de los datos se decodifica de acorde a la dirección del nodo que ingresen por ejemplo en la figura 62 se muestra el caso para el nodo 1, que como se vió anteriormente es la dirección 0XBB y según la posición en la que se envió los datos desde los prototipos se va obteniendo cada valor. La misma configuración se tiene para los demás nodos únicamente se cambia la dirección y las variables.

Figura 62

Decodificación de los valores ingresados

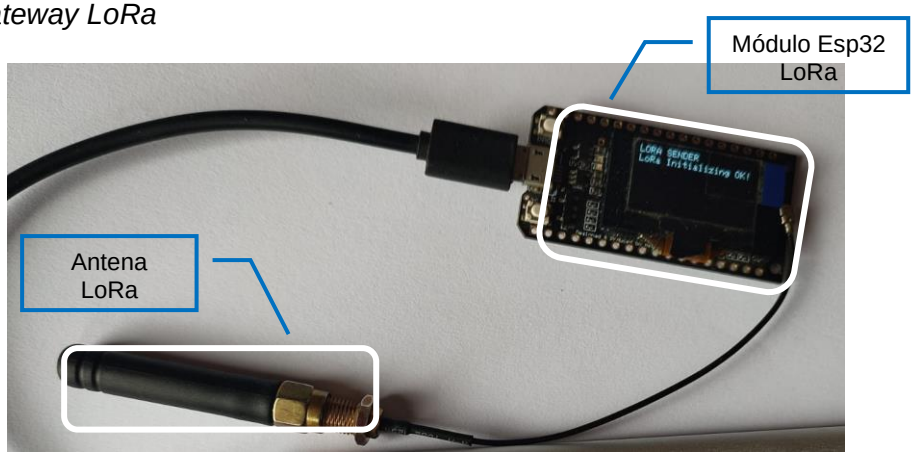
```
if( sender == 0XBB )
{
 s = getValue(incoming, ',', 0); // Vrms
 t = getValue(incoming, ',', 1); // Irms
 u = getValue(incoming, ',', 2); // consumo

   Vrms= s.toFloat();
   Irms = t.toFloat();
   ConsumoPot = u.toFloat();
```

Elaboración del gateway

Para la implementación del gateway únicamente se utiliza una tarjeta Esp32 LoRa de la marca Ttgo v1.0 configurada y programada como se vió en el enunciado anterior ya que con esto nos ahorramos en la compra de un gateway por su precio alto y poca disponibilidad en el mercado ecuatoriano en la figura 63 se muestra el gateway implementado que puede ser alimentado con un valor de 5V.

Figura 63

Implementación gateway LoRa



Mosquitto Broker

Para la utilización de mosquitto broker es necesario habilitar entradas y salidas en nuestro servidor mediante el puerto 1883 para que el broker pueda gestionar la mensajería enviada por el gateway en formato Json. Con esto logramos una mejor estructuración y orden para el tratamiento de los datos en node red. El broker es el encargado que cada variable que se obtiene de los prototipos de consumo de agua, de electricidad y de recolección de residuos mediante el tópico colocado sea direccionado a la nube.

La instalación del broker Mosquitto fue realizada en el VPS contratado por lo que para acceder a este broker nos dirigimos a su página oficial y lo descargamos según la versión y sistema operativo que tenga nuestro computador para el caso del proyecto se utiliza un servidor con Windows. En la sección de APÉNDICES, con el nombre “B: Manual de instalación” se tiene las consideraciones y pasos a tomar en cuenta para su instalación como se muestra en la figura 64 mediante el cmd se observa la versión e instalación de mosquitto. En la figura 65 se muestra la habilitación de los puertos de entrada y salida para lograr la conexión con node-red y otros servicios ocupados en el proyecto.

Figura 64

Verificación de instalación de Mosquitto Broker

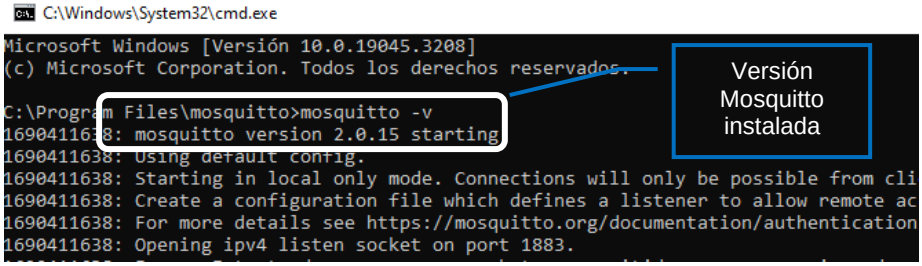
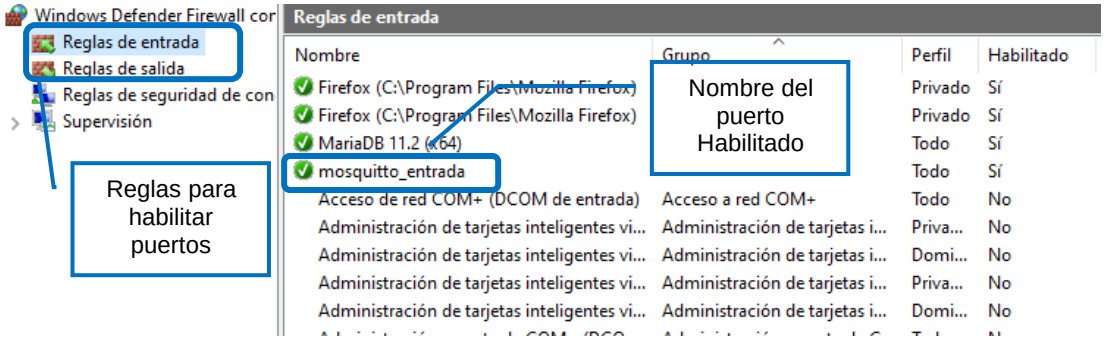


Figura 65

Puertos de entrada y salida abiertos



Programación del sistema de gestión de servicios básicos

Para la programación se utiliza Node-Red la cual es una herramienta basada en flujos que nos permite conectar dispositivos de hardware, Apis y servicios para realizar aplicaciones de IoT (Coro, 2022). En la instalación de la herramienta se adjunta algunos nodos para realizar acciones que requiere el proyecto o mediante el nodo function se los puede programar mediante el lenguaje JavaScript, las librerías empleadas son:

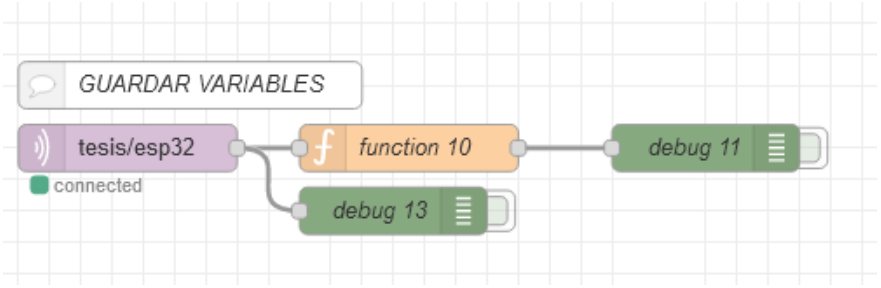
- Worldmap: Permite mostrar la ubicación mediante Google maps.
- Mysql: Permite comunicarse con una base de datos para crear, actualizar o eliminar datos de las tablas.
- Function: Permite programar mediante lenguaje JavaScript distintas acciones o configuraciones que se quieran realizar.

- Switch: Dirige los mensajes a una salida de acuerdo a un parámetro elegido.
- Change: Permite modificar las propiedades de un mensaje sin la necesidad de utilizar un nodo function
- Email: Permite el envío y recepción de mensajes a distintos servidores de email.

En la programación de los prototipos se presenta la utilización del tópico enviado por el gateway para guardar las variables en node red para poder procesarlos en el Dashboard y para guardar en la base de datos como se muestra en la figura 66.

Figura 66

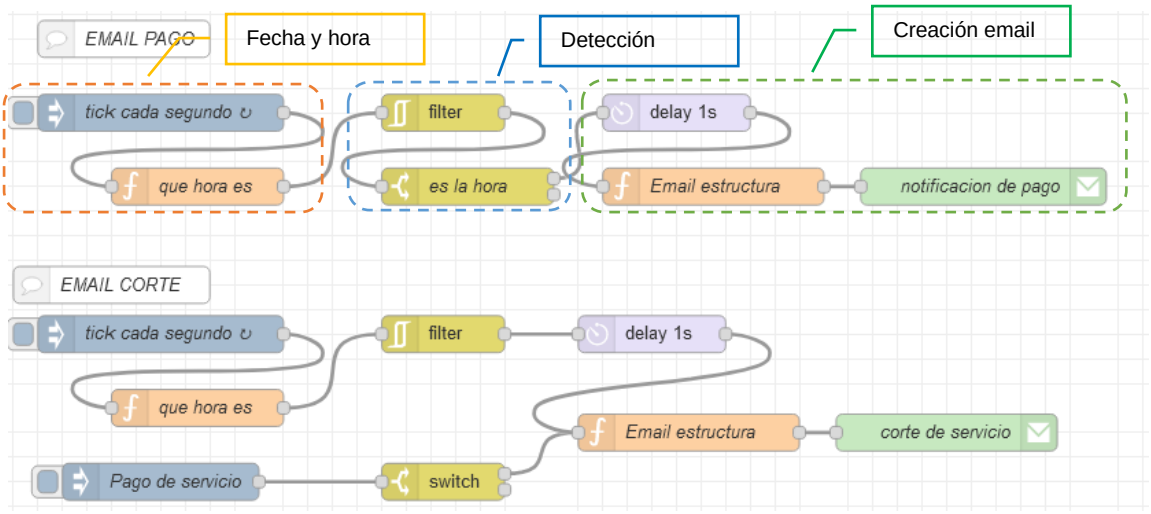
Variables guardadas en node red



Para la implementación del procesamiento de eventos complejos se utiliza el siguiente diagrama que será el mismo para el prototipo de agua y de luz. Este flujo enviará una notificación cada cierto día del mes, con los valores de consumo y los valores a pagar. En el caso que no se realice el pago después de un cierto tiempo llegará otro mensaje de notificación indicando el corte y los valores a pagar como se observa en la figura 67. Este diagrama consta de tres partes: generación de la fecha y hora, detección de la fecha y creación del email de notificación.

Figura 67

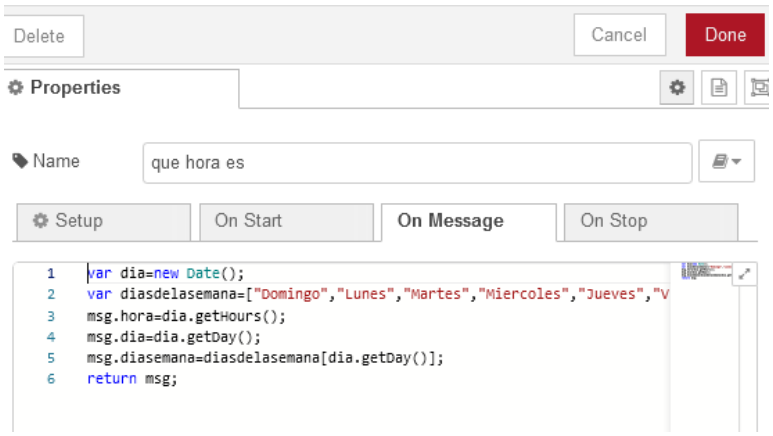
Procesamiento de eventos complejos para email de pago y corte.



Se utiliza un nodo function para crear la fecha y hora. Primero se establece un arreglo con los días de la semana de domingo a sábado y mediante el comando new Date(), obtenemos la fecha con la hora. Posteriormente guardamos en variables independientes la hora, los minutos y el día. Para los días de la semana mediante el arreglo creado en la línea 2 se relaciona el día para saber en qué día de la semana nos encontramos en la figura 68 se muestra el código realizado.

Figura 68

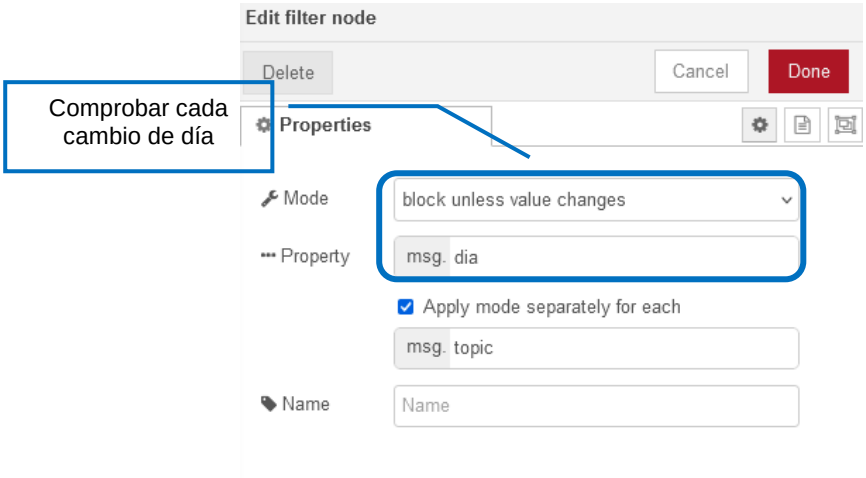
Creación de la fecha y hora



Para la segunda parte se debe detectar la fecha en la que se quiera enviar el email por lo que mediante el nodo filter tomamos el día y lo bloqueamos para que siga el flujo únicamente cuando cambie el día caso contrario nos enviaría emails cada segundo el día que se deba enviar. En la figura 69 se muestra la configuración realizada.

Figura 69

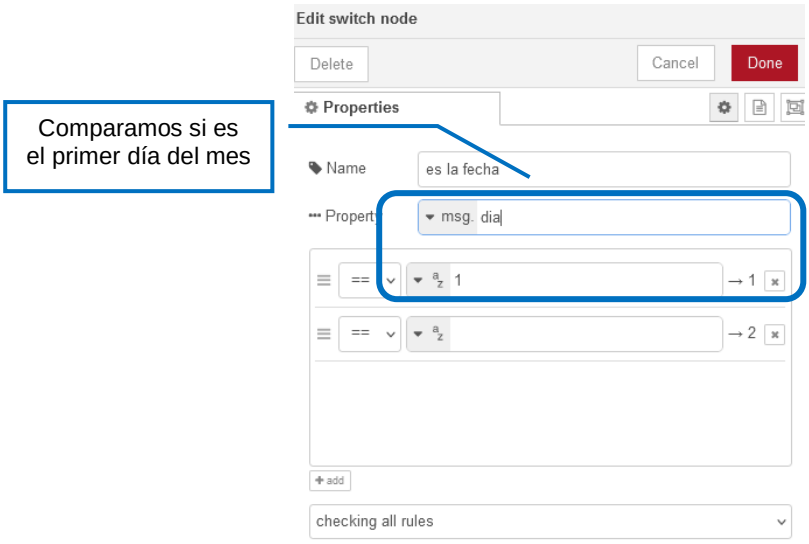
Detección del cambio de día



Mediante el nodo switch comprobamos si es que es el primer día de cada mes en el caso que sea que el flujo siga por la salida. En la figura 70 se muestra la configuración del nodo.

Figura 70

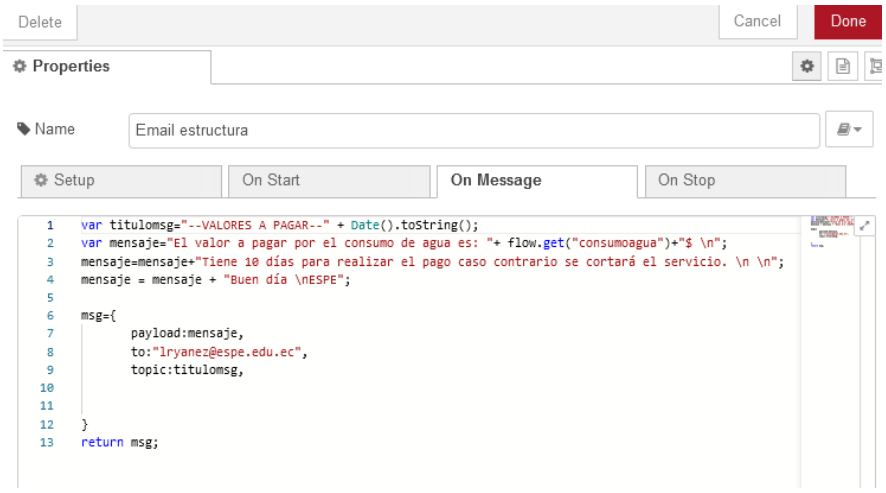
Detección del día a enviar la notificación



Finalmente se crea el mensaje a enviar al dueño del medidor con todas las credenciales de un mensaje como es el asunto, destinatario y contenido del mensaje como se muestra en la figura 71 se tiene el código.

Figura 71

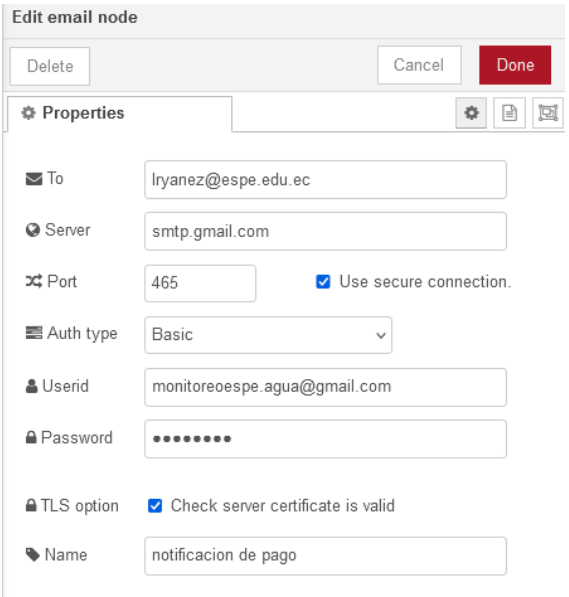
Creación del contenido del email



Para el envío de la notificación se debe configurar la cuenta de correo de donde se va enviar los mensajes. En el apéndice B se muestra las instalaciones y configuraciones realizadas, en el nodo email se coloca el destinatario y la cuenta del correo para enviar los mensajes. En la figura 72 se muestra la configuración del nodo.

Figura 72

Configuración nodo email

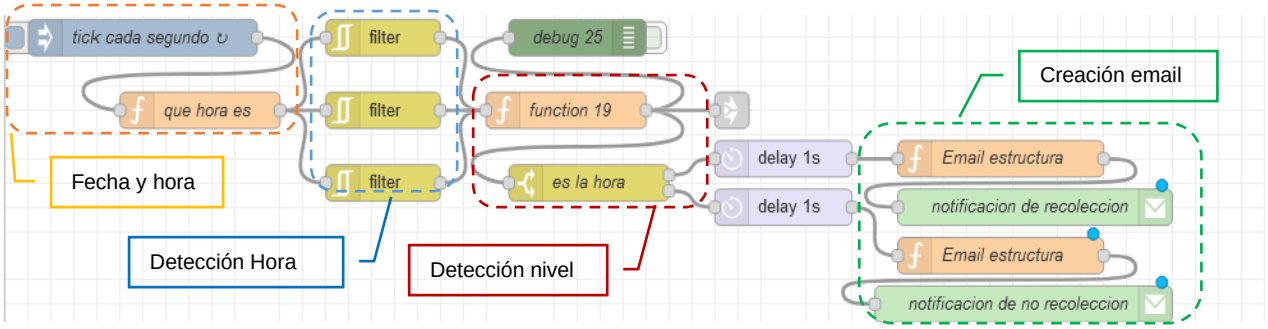


Para la notificación del corte del servicio tiene la misma estructura únicamente se agrega el pago del servicio mediante un nodo inyect simulando el pago para saber si se realizó o no se realizó el pago.

En el caso para el prototipo de residuos para enviar notificaciones al encargado de recolección, se utiliza el flujo de la figura 73 en la cual cada cierta hora de la tarde, verificando si son días laborables y si el tacho está con más del 75% notificará al recolector. Como se observa en la imagen tiene cuatro partes: generación de la fecha y hora, detección de la hora, comparación de niveles y la creación del email de notificación.

Figura 73

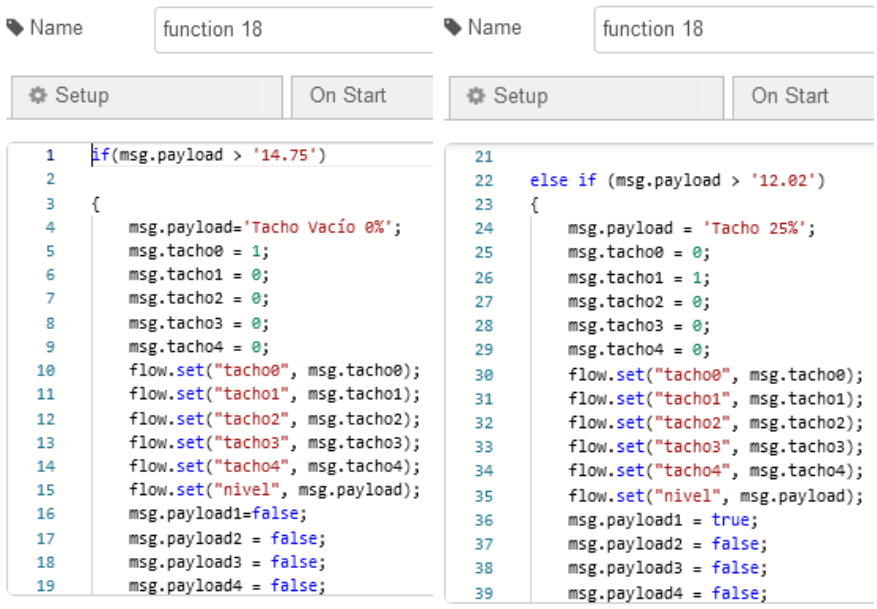
Procesamiento de eventos complejos prototipo residuos



La parte de la generación de fecha y hora con la detección de la hora tiene la misma configuración vista para el prototipo de consumo de agua lo que cambia es la comparación de los niveles del tacho para saber si es necesario o no enviar la notificación por lo que primero se debe definir los niveles del tacho según la distancia. En la figura 74 se tiene el código para la detección de los niveles del tacho donde en la parte de calibración del sensor se obtuvo una distancia con el tacho vacío de 14.75 cm esta medida se la divide en cuatro partes para obtener los cuatro niveles del tacho.

Figura 74

Código detección de los niveles de basura



Una vez detectado cada nivel del tacho de basura se debe comparar tres cosas: primero si son días laborables, segundo la hora de envío del mensaje y tercero si el tacho se encuentra con un nivel superior al 75%. En la figura 75 se muestra el código utilizado donde se observa que si cumple las tres condiciones antes mencionadas se enviará un uno y en el caso que una de las tres no cumpla se enviará un cero.

Figura 75

Código de la comparación del nivel, hora y días laborables

Mediante el nodo Switch los valores enviados en la anterior comparación los direccionamos a las salidas del nodo si se tiene un uno los datos van por la salida 1 y si se tiene un cero se direcciona por la salida 2 como se muestra en la figura 76. Con esto podemos enviar el email si es que es necesario o no la recolección de la basura.

Figura 76

Direccionamiento para envío del email

Finalmente, como en el prototipo de agua se establece los emails para saber si es o no es necesario la recolección de la basura. En la figura 77 se tiene el código de la creación para el envío del email.

Figura 77

Código utilizado para envío de las notificaciones

Implementación base de datos

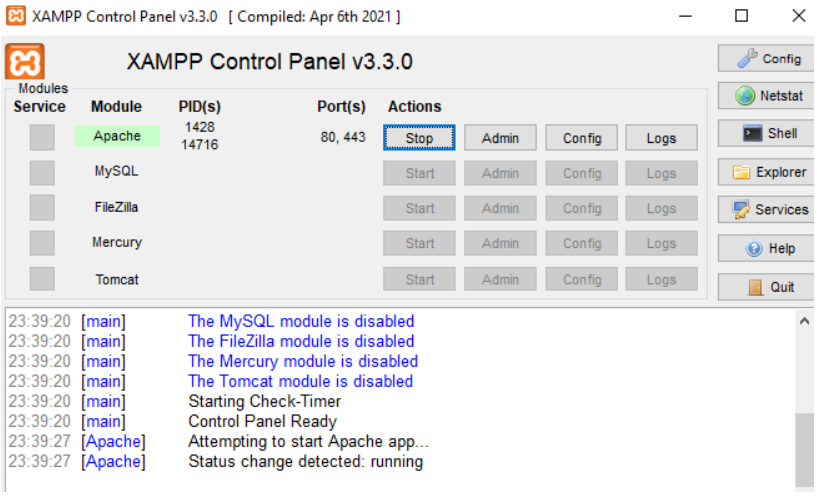
Para el proyecto se utiliza una base de datos estructurada que se encuentra ubicada en el servidor remoto de contabo. MariaDb es la base de datos utilizada ya que es una de las más populares a nivel mundial, la cual es prácticamente parecido a MySQL ya que se creó por el mismo desarrollador. En la gestión de estos datos se ocupa phpmyadmin.

Creación de base de datos

En el servidor se instala el paquete XAMPP en el cual se selecciona únicamente el servicio web apache que permitirá administrar la base de datos como se muestra en la figura 78.

Figura 78

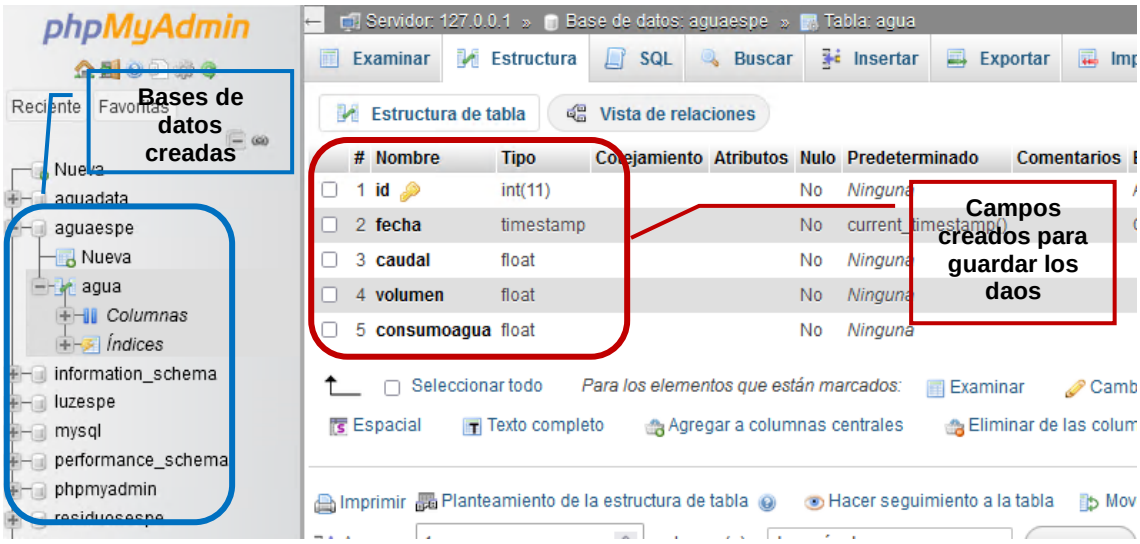
Paquete XAMPP



Para la gestión de la base de datos se utiliza PhpMyAdmin en la cual se crea tres bases de datos que son aguaespe, luzespe y residuosespe una para cada prototipo que tendrán los valores característicos enviados por cada prototipo como se observa en la figura 79.

Figura 79

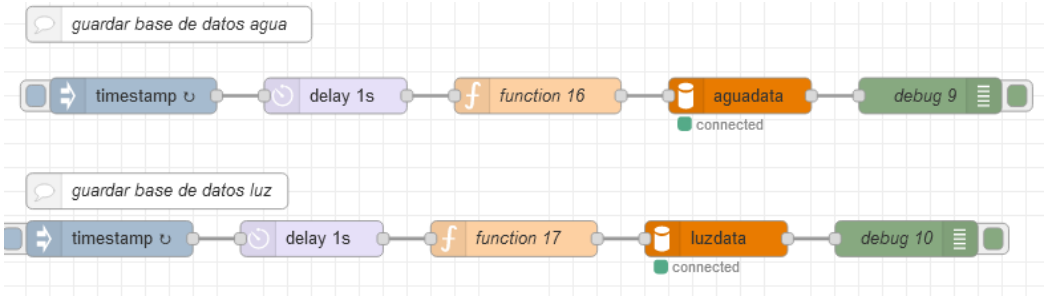
Interfaz de gestión de bases de datos phpMyAdmin



Una vez creada las bases de datos se procede mediante node red a insertar los datos provenientes de los prototipos como se muestra en la figura 80. En las cuales se puede configurar cada que tiempo se quiere que se guarde los datos para el presente proyecto está configurado para ser guardado cada hora.

Figura 80

Datos guardado en las bases de datos

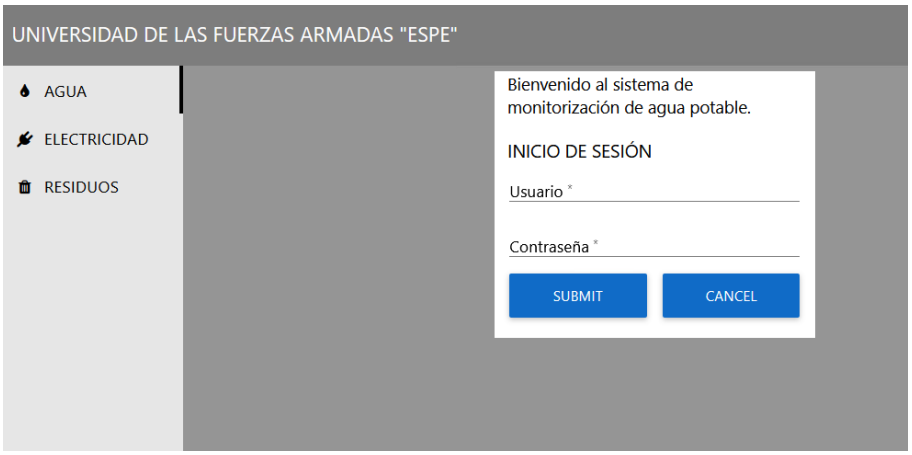


Implementación de la interfaz

En el capítulo anterior en la parte de diseño de la interfaz se mostró como estaría distribuída cada una de las pantallas y la jerarquía de las mismas. En cada uno de los prototipos se tendrá que colocar el usuario y contraseña para poder acceder a la información, esta ventana de inicio de sesión está validada para que se despliegue mensajes cuando la contraseña es incorrecta y cuando el usuario que se haya ingresado no sea el correcto en la figura 81 se muestra la pestaña de inicio de sesión que es la misma para los tres prototipos únicamente cambia de ventana entre prototipos. En la sección APÉNDICES, con el nombre “A: Manual de Usuario” se tiene el manejo de la interfaz y las credenciales para su ingreso.

Figura 81

Pestaña inicio de sesión de la interfaz



En el menú principal de cada uno de los prototipos se tiene las diferentes opciones para poder visualizar la información, monitoreo de las variables, la ubicación y los históricos como se muestra en la figura 82.

Figura 82

Menú principal del sistema de monitorización

En las ventanas de información de cada uno de los prototipos se presentará el número de los medidores, nombre de la persona dueña del medidor y los diferentes valores de

consumo. En las figuras 83, 84 y 85 se muestra la pestaña de información de los tres prototipos.

Figura 83

Pestaña de información prototipo consumo de electricidad

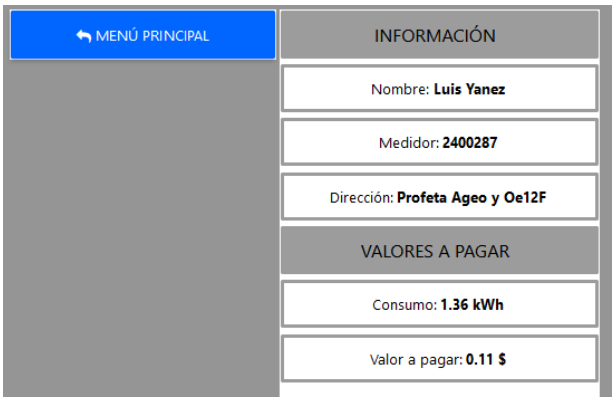


Figura 84

Pestaña de información prototipo consumo de agua

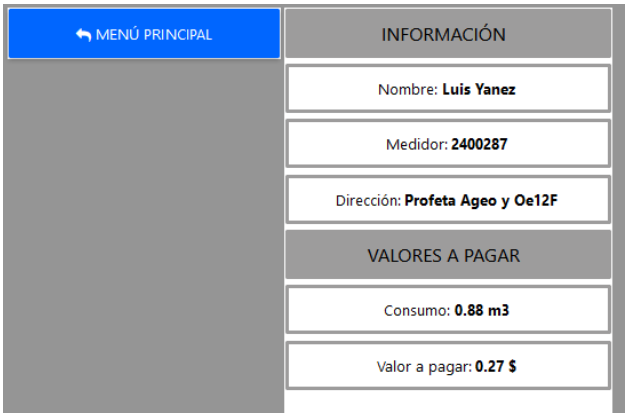
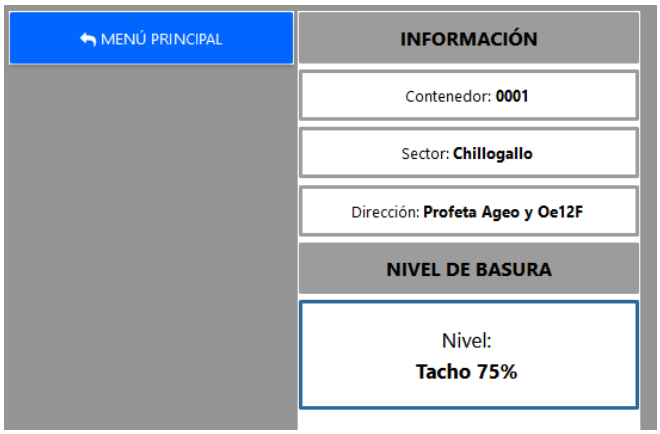


Figura 85

Pestaña de información prototipo nivel de residuos



En cada uno de los prototipos se visualizará todas las variables monitoreadas mediante los sensores de cada prototipo. En las figuras 86, 87 y 88 se muestra la pestaña de monitoreo de los tres prototipos.

Figura 86

Pestaña monitoreo prototipo consumo de electricidad

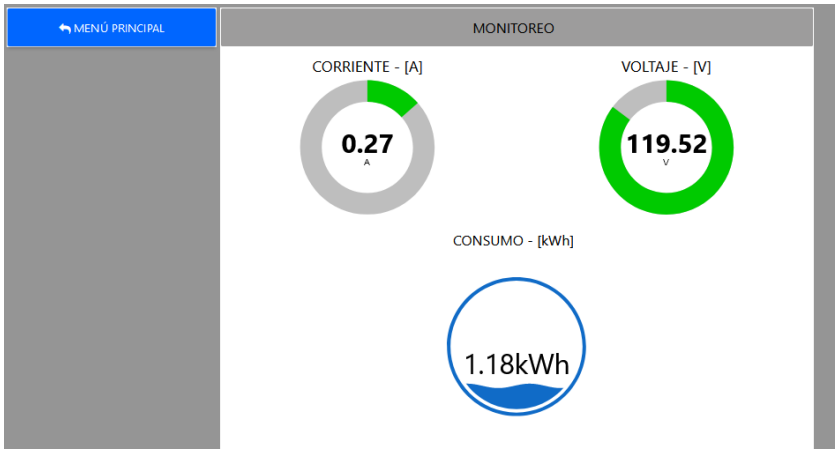


Figura 87

Pestaña monitoreo prototipo consumo de agua

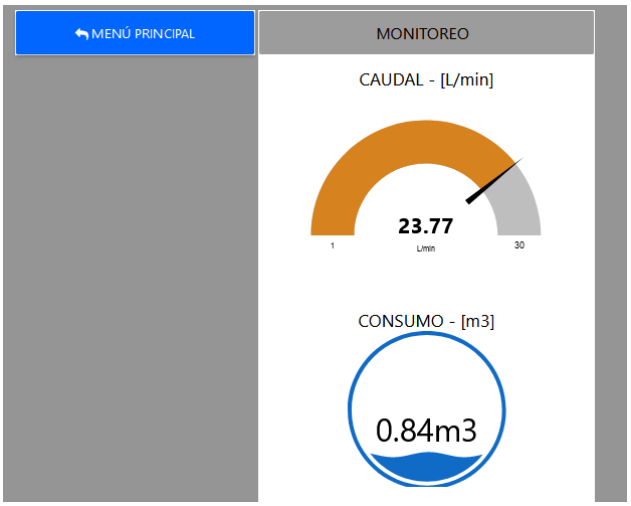
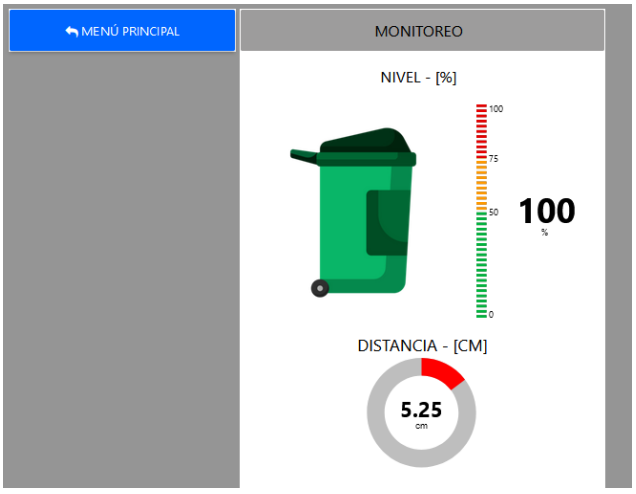


Figura 88

Pestaña monitoreo prototipo nivel de residuos



En la pestaña de ubicación mediante Google Maps se puede visualizar la ubicación exacta de cada prototipo, si se coloca el mouse encima del ícono de ubicación nos mostrará el nombre del dueño del medidor y el número del prototipo. En la figura 89 y 90 se tiene la pestaña de ubicación de los prototipos.

Figura 89

Pestaña de ubicación para los prototipos de electricidad y agua

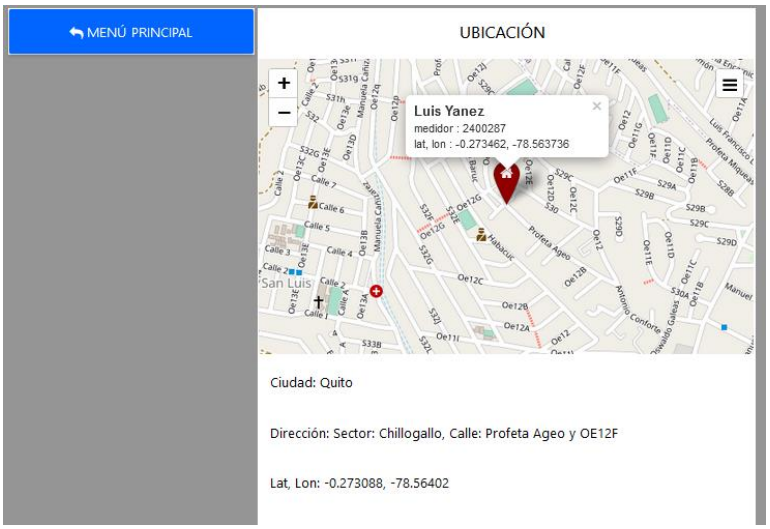
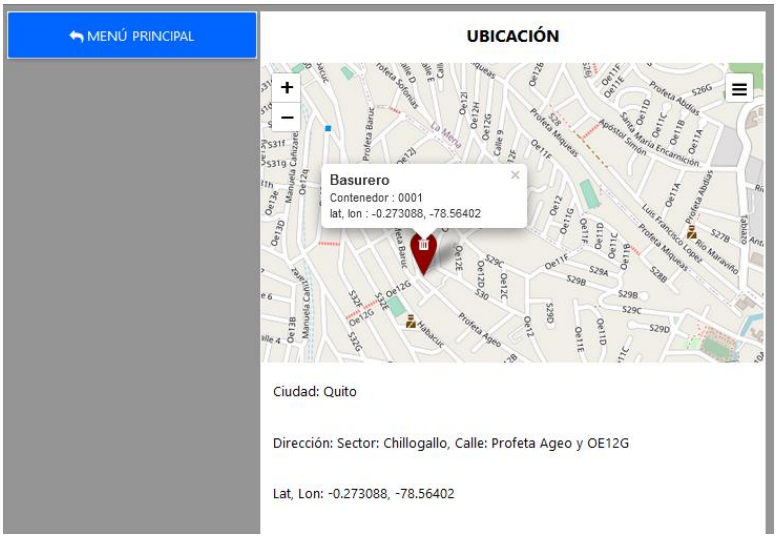


Figura 90

Pestaña de ubicación para el prototipo de nivel de residuos



Finalmente se tiene la pestaña de gráficos históricos donde se puede observar el comportamiento de las variables monitoreadas a lo largo del tiempo, en estas gráficas si se coloca el apuntador del mouse en la línea graficada se mostrará el valor que tomó la variable

en ese momento. En las figuras 91, 92 y 93 se muestra las pestañas de históricos de los prototipos.

Figura 91

Pestaña de históricos prototipo consumo de electricidad

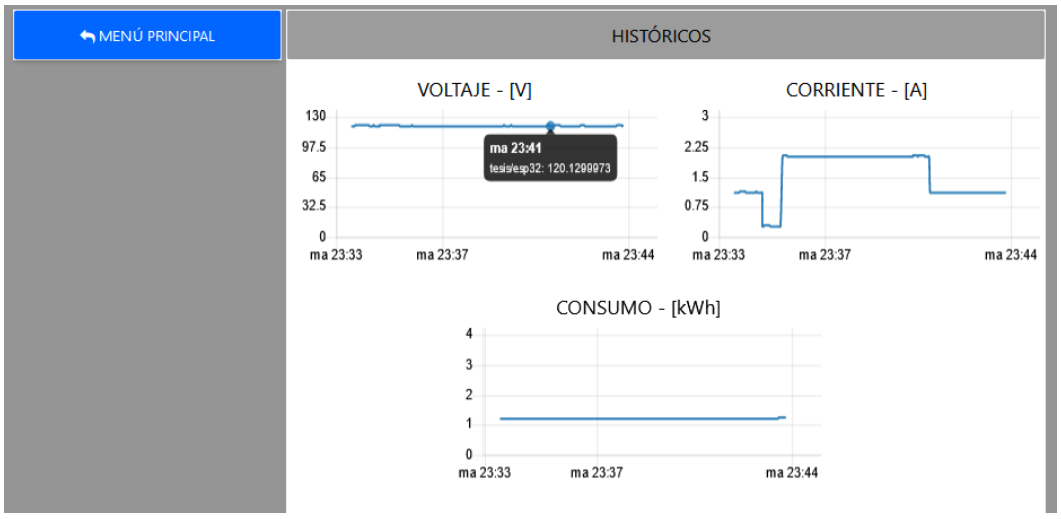


Figura 92

Pestaña de históricos prototipo consumo de agua

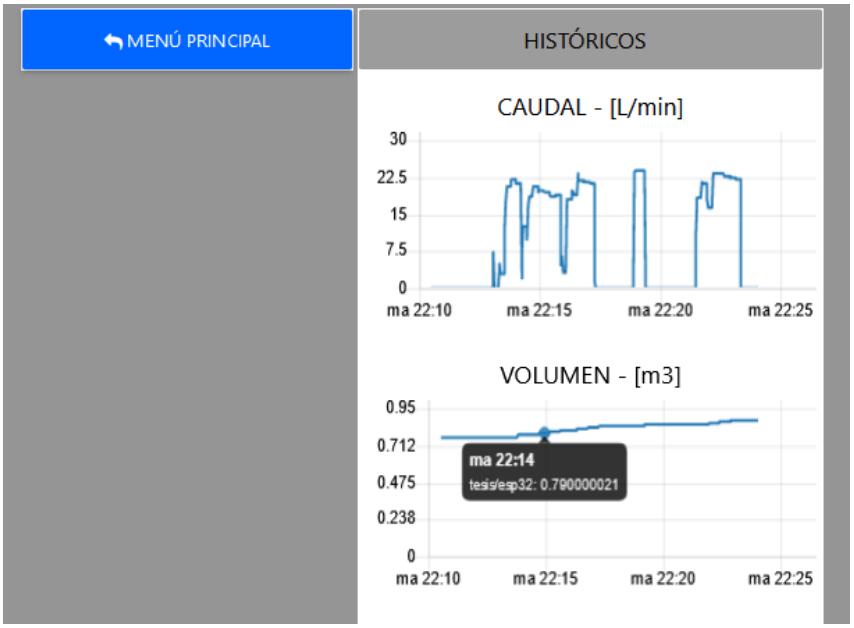


Figura 93

Pestaña de históricos prototipo nivel de residuos

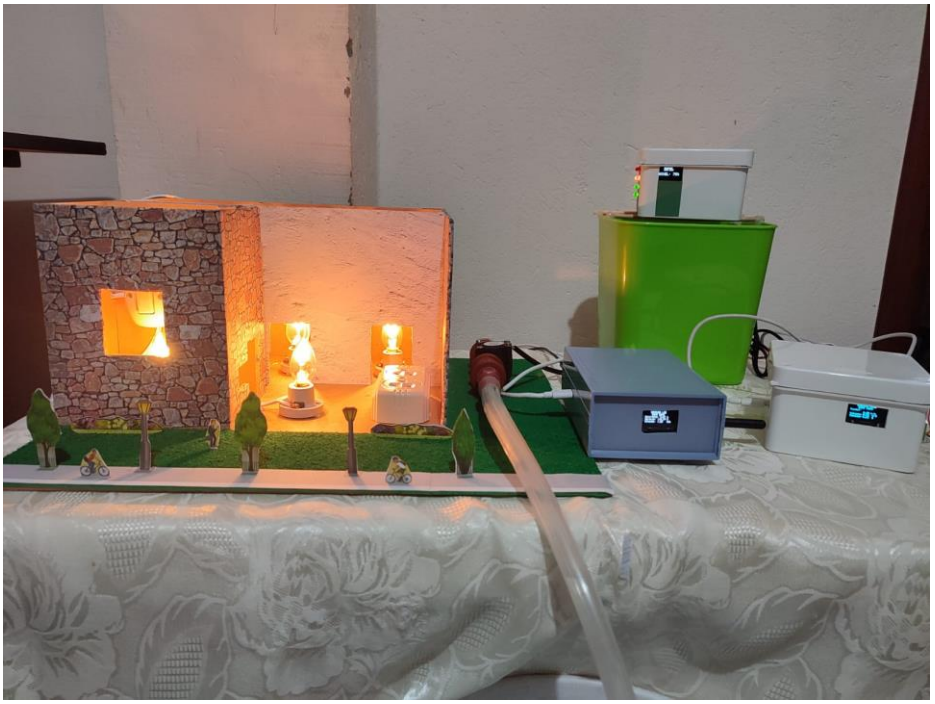
Capítulo V. Pruebas de validación

Pruebas del sistema

Para verificar el funcionamiento del sistema implementado se llevaron a cabo pruebas en una maqueta con cargas como lámparas de 7.5 W y un tomacorriente con dos entradas para conectar distintas cargas que se pueden encuentran en un hogar, mientras que para el prototipo de agua se conecta mediante una manguera al sensor de caudal para realizar mediciones en litros en un recipiente, con el que comprobaremos que las mediciones son parecidas a las simuladas en la maqueta y para el prototipo de recolección de residuos se utilizará un tacho pequeño donde se simulará el llenado de un contenedor de basura. La maqueta con la que se realizó estas pruebas se muestra en la figura 94.

Figura 94

Maqueta utilizada para las pruebas del sistema



Funcionamiento prototipo consumo de luz

Prueba de medición de voltaje AC

Para verificar que el prototipo mide valores reales de voltaje se tomaron 15 mediciones de voltaje en distintas horas del día en la maqueta las cuales deben variar en un rango de 110 V a 127 V. El multímetro utilizado muestra valores con una resolución de hasta un decimal en la tabla 14 y la figura 95 se verifica la medición de los voltajes.

Tabla 14

Mediciones de voltaje

Multímetro Zotek Zt101 [V]	Prototipo consumo de luz [V]	Error Absoluto [V]
119.8	119.6	0.2
119.7	119.5	0.2
119.9	119.7	0.2
119.6	119.4	0.2
120.5	120.4	0.1
120.6	120.5	0.1
121.4	121.2	0.2
122.5	122.3	0.2
122.2	122.0	0.2
121.6	121.5	0.1
123.5	123.3	0.2
123.1	122.9	0.2
122.4	122.3	0.1
122.6	122.4	0.2
122.8	122.7	0.1

Figura 95

Medición de voltajes

Prueba de medición de corriente AC

En las pruebas de corriente se realizaron 15 mediciones donde se coloca el sensor de corriente SCT-013 en la fase de donde se administra la energía, para estas mediciones se utilizaron distintas cargas para comprobar el valor real de la corriente mediante una pinza amperimétrica y comparándola con el prototipo realizado como se muestra en la tabla 15 y en la figura 96.

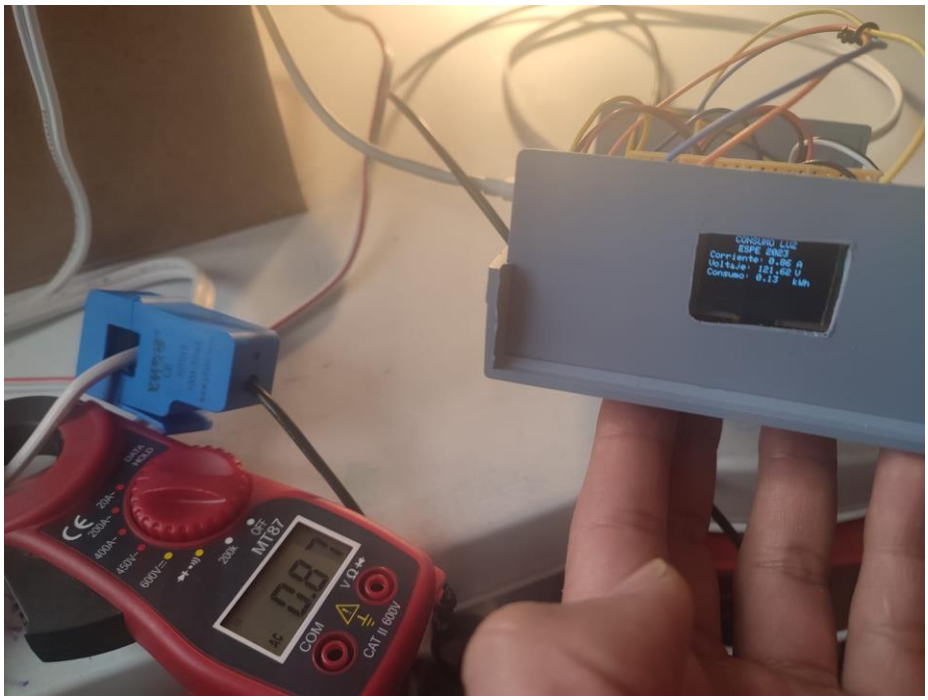
Tabla 15

Mediciones de corriente

Carga colocada [W]	Pinza MT87 amperimétrica [A]	Prototipo consumo de luz [A]	Error absoluto %
7.5	0.05	0.06	0.1
15	0.10	0.11	0.1
22.5	0.15	0.16	0.1
37.5	0.26	0.27	0.1
33	0.25	0.26	0.1
40.5	0.31	0.31	0
48	0.35	0.35	0
55.5	0.40	0.40	0
60	0.42	0.42	0
70.5	0.48	0.48	0
100	0.87	0.86	0.1
107.5	0.93	0.92	0.1
115	0.98	0.97	0.1
122.5	1.03	1.02	0.1
137.5	1.14	1.13	0.1

Figura 96

Mediciones de corriente



Pruebas de consumo

Las pruebas de consumo se realizaron durante 15 días en las cuales se mantuvo encendido en la maqueta una carga de 138 W durante un tiempo 3 horas cada día para comprobar que todos los días diera el mismo valor de consumo, en la tabla 16 se muestra los valores obtenidos.

Tabla 16

Datos de los valores de consumo

Fecha	Hora	Consumo acumulado [kWh]	Consumo diario [kWh]
18/07/2023	18:00-21:00	0.12	0.12
19/07/2023	18:00-21:00	0.52	0.40
20/07/2023	18:00-21:00	0.93	0.41
21/07/2023	15:00-18:00	1.33	0.40
22/07/2023	15:00-18:00	1.73	0.40
23/07/2023	15:00-18:00	2.13	0.41
24/07/2023	15:00-18:00	2.54	0.41
25/07/2023	10:00-13:00	2.96	0.42
26/07/2023	10:00-13:00	3.37	0.41
27/07/2023	10:00-13:00	3.79	0.42
28/07/2023	10:00-13:00	4.21	0.42
29/07/2023	10:00-13:00	4.62	0.41
30/07/2023	18:00-21:00	5.01	0.39
31/07/2023	18:00-21:00	5.42	0.40
1/08/2023	18:00-21:00	5.82	0.40

Los datos obtenidos son muy parecidos en las mediciones para los consumos diarios en los 15 días de recolección de datos. Pero hay una variación respecto a las horas, esto se lo hizo para verificar como afecta la caída de tensión a la potencia de las cargas que se coloquen como se observa en horas de la noche cuando hay más personas en el hogar la caída de tensión es mucho mayor porque se conecta un mayor número de dispositivos en el hogar

mientras que en la mañana la caída de tensión es mucho menor debido a que en el hogar se encuentra menos personas utilizando el servicio eléctrico. En la figura 97 se muestra el envío de las notificaciones de pago según un determinado número de días y con el valor a pagar.

Figura 97

Notificación de pago por el consumo eléctrico

Funcionamiento prototipo consumo de agua

En las pruebas de funcionamiento para el prototipo de agua al sensor de flujo se conectó a la tubería de agua mediante una manguera en la cual se utilizó durante 15 días para realizar mediciones y comprobar el consumo de agua. En la tabla 17 se muestra todos los valores obtenidos en el transcurso de los días mientras que en la figura 98 se muestra las mediciones realizadas con el prototipo mediante su pantalla OLED, en la cual se puede visualizar los valores de caudal y el consumo.

Tabla 17

Mediciones consumo de agua

Día	Hora de corte	Consumo acumulado [m3]	Consumo diario [m3]
18/07/2023	22:00	0.05	0.05
19/07/2023	22:00	0.11	0.06
20/07/2023	22:00	0.16	0.05
21/07/2023	22:00	0.22	0.06
22/07/2023	22:00	0.30	0.08
23/07/2023	22:00	0.36	0.06
24/07/2023	22:00	0.44	0.08
25/07/2023	22:00	0.52	0.08
26/07/2023	22:00	0.59	0.07
27/07/2023	22:00	0.68	0.09
28/07/2023	22:00	0.71	0.03
29/07/2023	22:00	0.83	0.12
30/07/2023	22:00	0.91	0.08
31/07/2023	22:00	1.02	0.11
1/08/2023	22:00	1.18	0.16

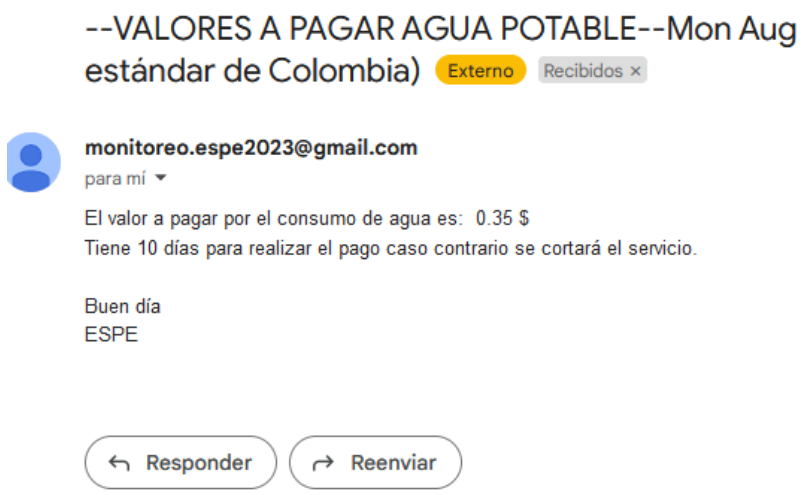
Figura 98

Pruebas prototipo consumo de agua

Los datos obtenidos como se observan son variados ya que estos dependerán del consumo de agua que se realice diariamente para este caso como solo se utilizó una maqueta los valores de consumo en m3 son muy pequeños pero el prototipo mide eficientemente ya que se comparó las mediciones obtenidas con recipientes delimitados en litros. En la figura 99 se muestra el envío de las notificaciones por el consumo de agua.

Figura 99

Notificaciones para el pago del consumo de agua



Funcionamiento prototipo nivel de residuos

En el monitoreo de los niveles de residuo se ocupó un pequeño tacho de basura para colocar el prototipo en el cual se simularon distintas situaciones que pueden suceder en la vida real por ejemplo que el nivel este vacío por lo tanto no sea necesario la recolección de la basura o por otra parte si el nivel está al 100% o mayor al 75% se enviará una notificación por correo informando que si es necesario realizar la recolección.

Como se muestra en la figura 100 se realiza la primera prueba cuando el nivel del tacho se encuentra en un 25% y está prendido el primer indicador led por lo que no se envía un correo de notificación ya que no es necesaria la recolección de basura.

Figura 100

Prueba prototipo de residuos al 25%

En la figura 101 se observa la prueba realizada para un 50% de llenado del tacho en el cual se muestra en la pantalla del prototipo y se encienden los dos primeros leds indicadores, pero no se envía ninguna notificación ya que no es necesaria la recolección.

Figura 101

Prueba prototipo de residuos 50%

En el tercer caso se tiene para un 75% o 100% de llenado del tacho en estos casos si es necesario la recolección de la basura por lo cual a partir de una cierta hora que se coloque tomará el nivel y se enviará una notificación al recolector para saber si es o no es necesario la recolección de la basura como se observa en la figura 102.

Figura 102

Prueba prototipo de residuos al 100%

Las notificaciones que llegan a los encargados de la recolección están configuradas para que sean en días laborables de lunes a viernes en una determinada hora que se quiera enviar, siempre y cuando el nivel de basura sea mayor al 75% como se muestra en la figura 103.

Figura 103

Notificación para la recolección de residuos

Monitoreo de los prototipos

Los tres prototipos están protegidos inicio de sesión en los cuales se debe ingresar un usuario y una contraseña, también se encuentra validado para que el caso que no se ingrese el usuario nos muestre un mensaje diciendo que el usuario no existe o en el caso que se ingrese el usuario, pero con una contraseña incorrecta igual nos desplazará un mensaje como se muestra en las figuras 104 y 105.

Figura 104

Inicio de sesión con contraseña incorrecta

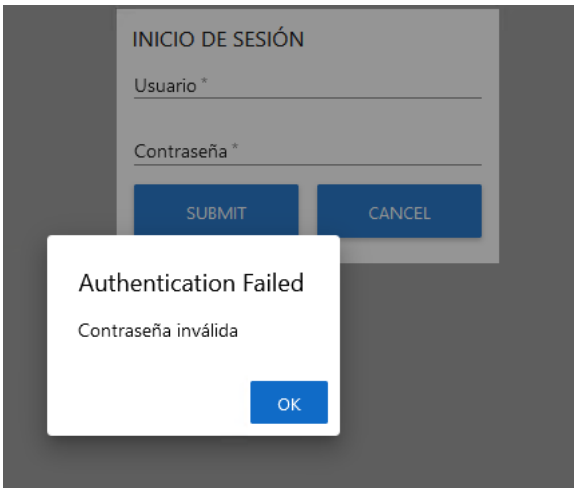
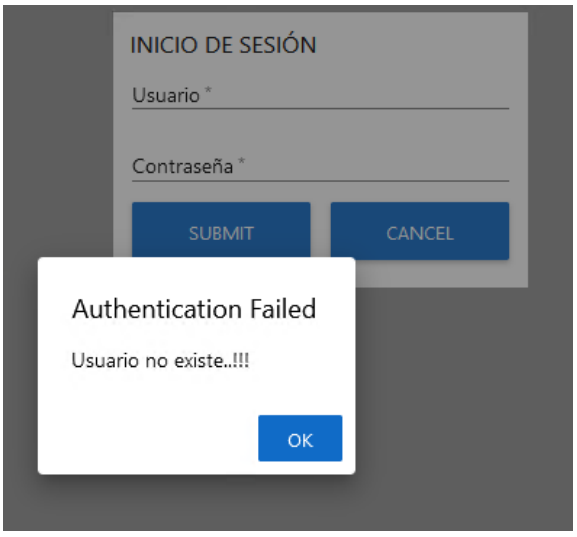


Figura 105

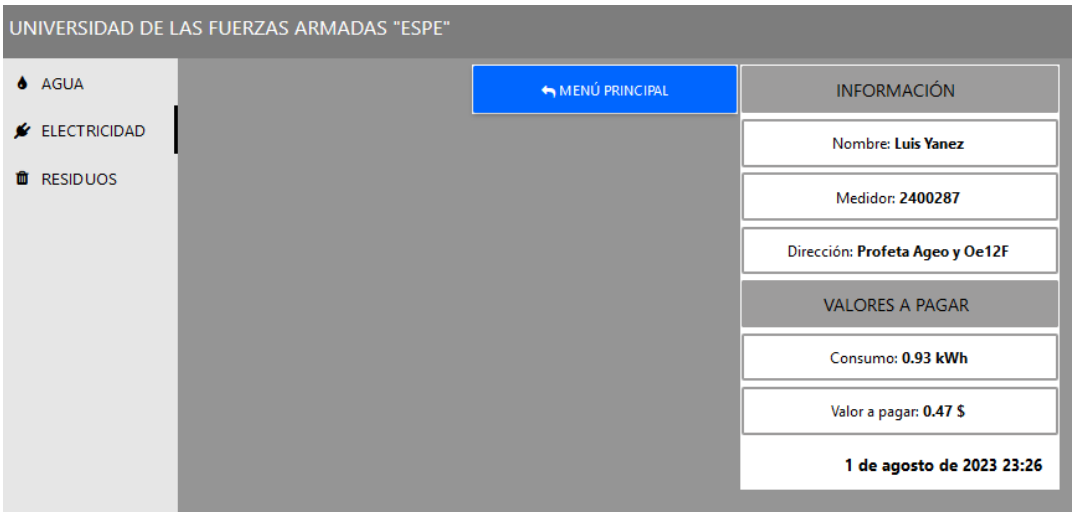
Inicio de sesión con usuario no existente



Los tres prototipos tienen un menú principal en los cuales se puede ingresar a la parte de información, monitoreo, ubicación, históricos. En la figura 106 se tiene la ventana de información del prototipo de consumo de luz para los tres prototipos tiene la misma distribución.

Figura 106

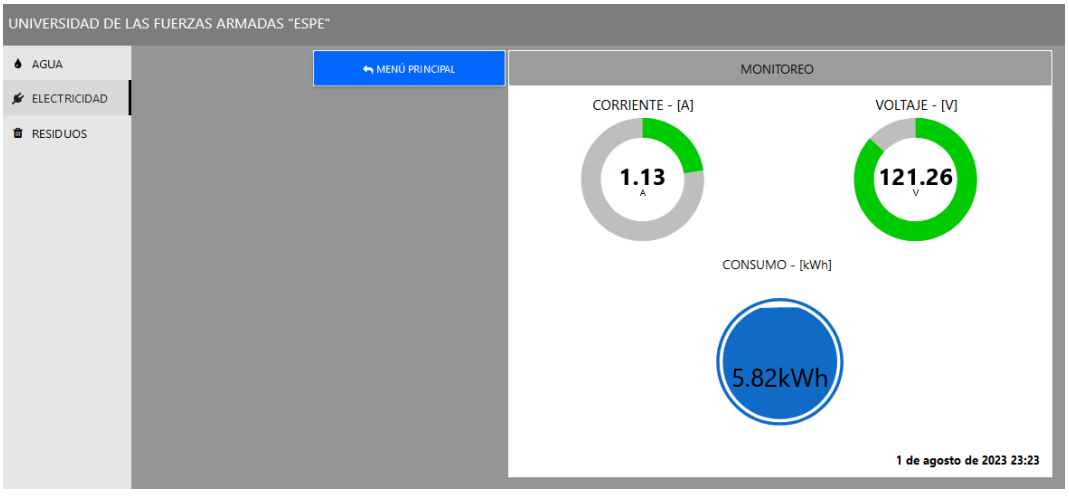
Pestaña de información del consumo eléctrico



Por otra parte, en la ventana de monitoreo se observa todas las variables que los prototipos registran, en las figuras 107 se muestra el monitoreo de las variables para el consumo eléctrico para los demás prototipos tiene la misma distribución.

Figura 107

Pestaña monitoreo prototipo consumo de luz



Finalmente se tiene la ventana de históricos de cada prototipo donde se muestra gráficas del comportamiento de las variables en el tiempo. También se tiene la ventana de ubicación en la cual se muestra el lugar exacto del prototipo mediante los datos de latitud y longitud como se observa en la figura 108 y 109 respectivamente.

Figura 108

Pestaña de históricos

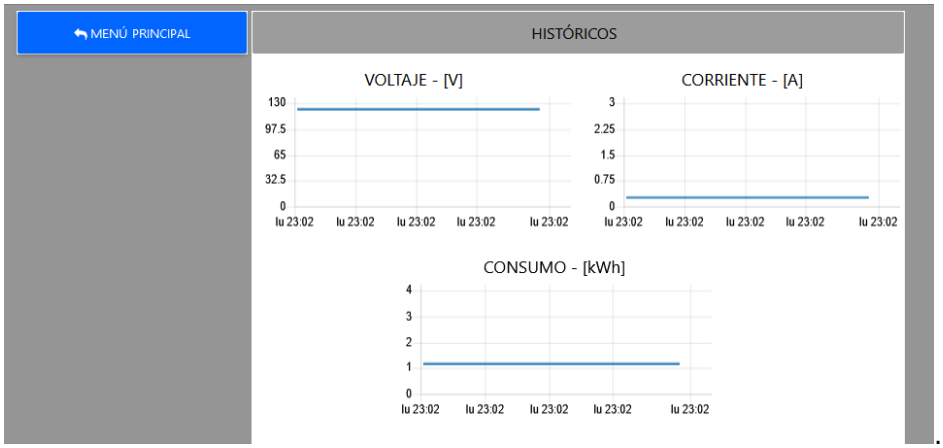
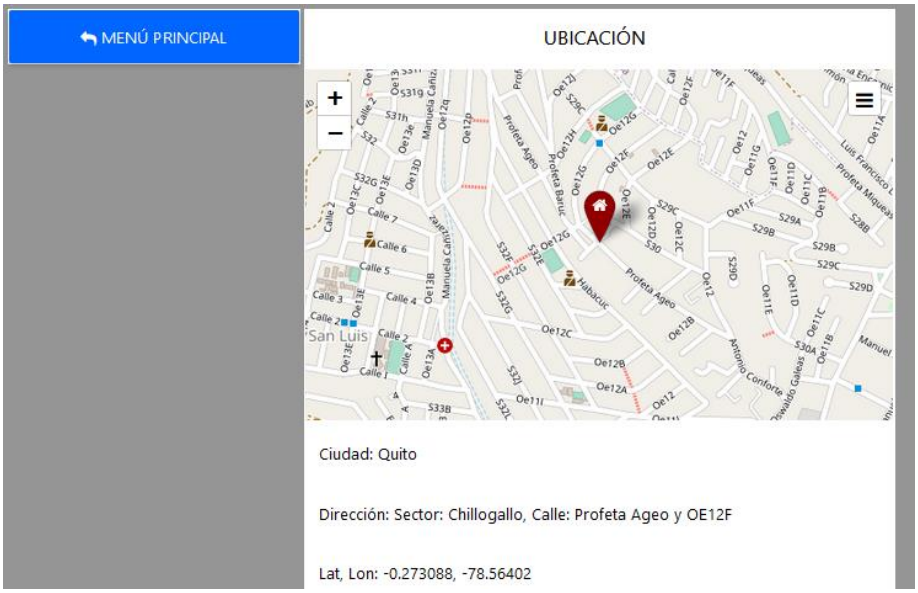


Figura 109

Ventana de la ubicación y dirección del prototipo



Pruebas de carga

Una vez realizado la arquitectura que cumple con los requisitos vistos en la parte de diseño y comprobado su funcionamiento es importante realizar pruebas de carga mediante la herramienta Gatling la cual estresa al sistema simulando peticiones por un número determinado de usuarios. Esta herramienta está basada en el lenguaje Scala en la cual se puede ir variando el número de usuarios mediante el software IDE IntelliJ IDEA para así saber hasta cuantos usuarios el sistema funciona correctamente.

El código consta de cuatro partes esenciales, en la primera parte consta la importación de los paquetes necesarios para el código como se muestra en la figura 110.

Figura 110

Importación de paquetes

```
import io.gatling.core.Predef._
import io.gatling.http.Predef._
import io.gatling.jdbc.Predef._
```

En la segunda parte se define el protocolo para este caso HTTP y los atributos importantes que se tiene para el programa como se muestra en la figura 111.

Figura 111

Definición del protocolo y atributos

```
val httpProtocol = http
    .baseUrl("http://192.168.1.22:1880")
    .inferHtmlResources(BlackList("""".*\.js""", """.*\.css""")
    .acceptHeader("image/avif,image/webp,image/apng,image/svg
    .acceptEncodingHeader("gzip, deflate")
```

En la tercera parte se tiene la creación del escenario a la cual se le asigna un nombre y se especifica las peticiones que se van a realizar como se muestra en la figura 112.

Figura 112

Creación del escenario

```
    val scn = scenario("prueba4re")
.exec(http("request_0")
    .get("/")
    .headers(headers_0)
    .resources(http("request_1")
    .get("/vendor/vendor.js?v=3.0.2")
    .headers(headers_1),
        http("request_2")
    .get("/vendor/monaco/monaco-bootstrap.js?v=3.0.2")
    .headers(headers_2),
        http("request_3")
    .get("/red/red.min.js?v=3.0.2")
```

Finalmente, en la figura 113 se tiene la definición de los usuarios que se pretenden simular para las pruebas de estrés.

Figura 113

Definición del número de usuarios

```
setUp(scn.inject(atOnceUsers(500))).protocols(httpProtocol)
```

Pruebas de carga 10 usuarios

En la figura 114 se observa que se ha realizado con éxito las 10 peticiones en las cuales el 93% se hicieron en un tiempo menor a 800 ms mientas que el 7% restante en un tiempo entre 800 ms y 1200 ms mientras que en la figura 115 se puede observar el pico de peticiones por segundo a lo largo de la prueba en el software Gatling.

Figura 114

Prueba de carga con 10 usuarios

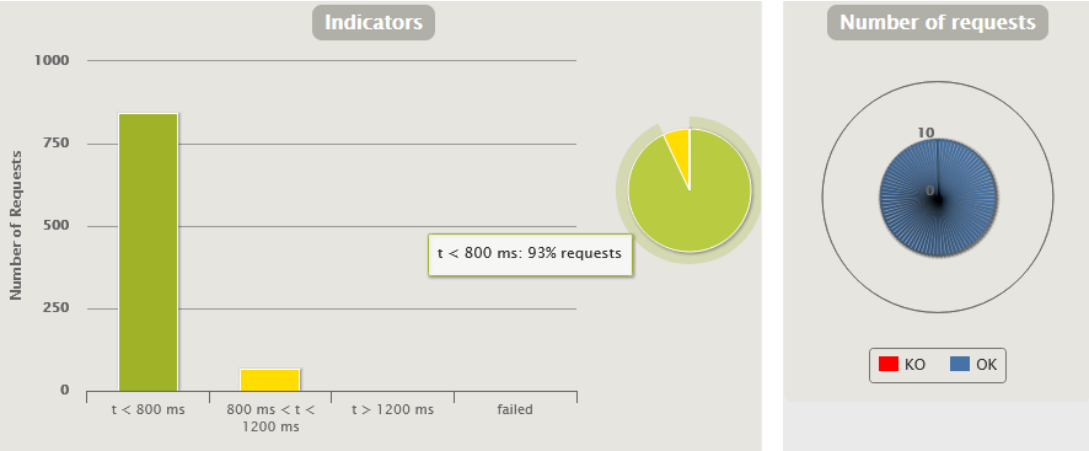
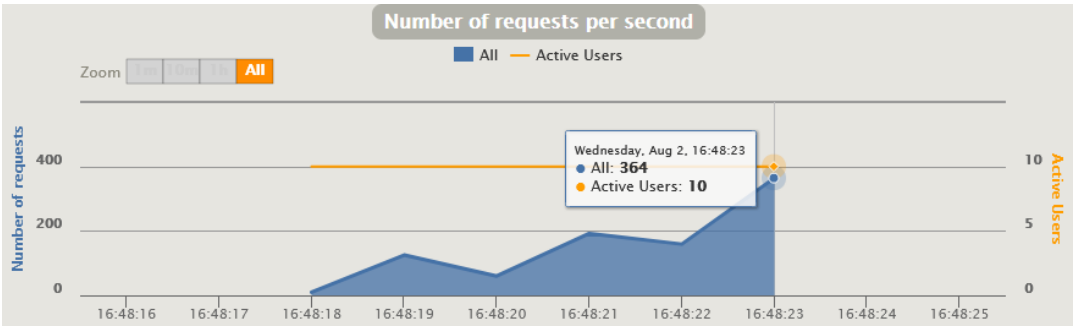


Figura 115

Pico de peticiones con 10 usuarios



Pruebas de carga 50 usuarios

En la figura 116 se observa que se ha realizado con éxito las 50 peticiones en las cuales el 55% se hicieron en un tiempo menor a 800 ms mientas que el 29% en un tiempo entre 800 ms y 1200 ms y el 16% restante en un tiempo mayor a 1200 ms mientras que en la figura 117 se puede observar el pico de peticiones por segundo a lo largo de la prueba con el número de usuarios activos.

Figura 116

Prueba de carga con 50 usuarios

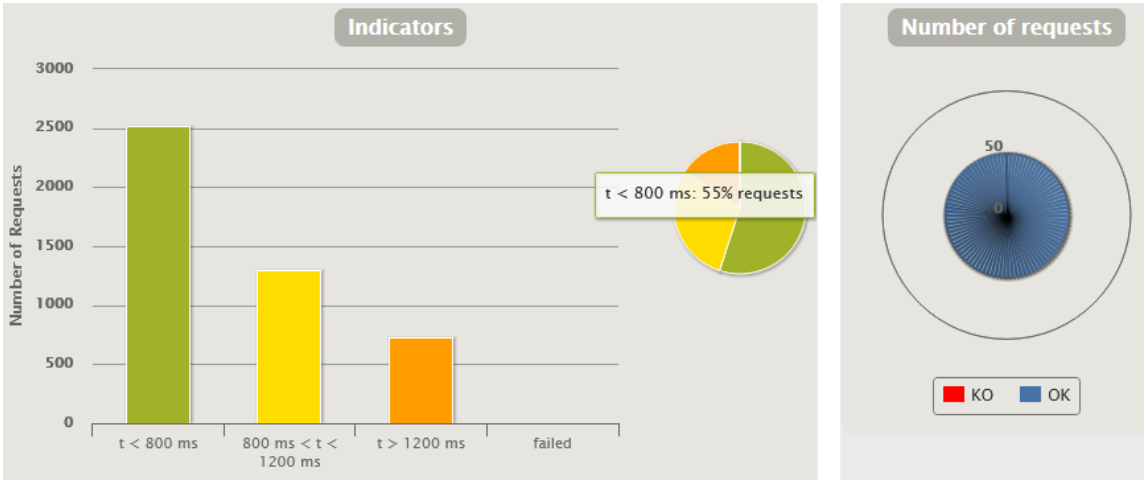
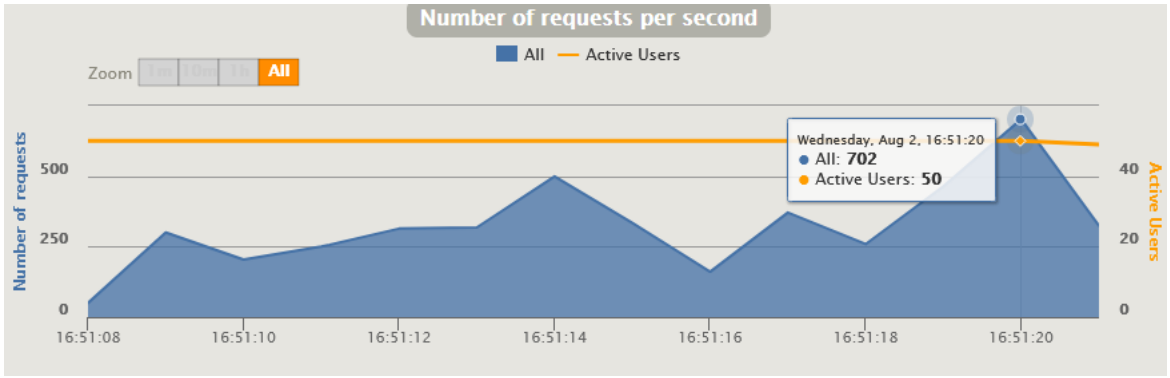


Figura 117

Pico de peticiones con 50 usuarios



Pruebas de carga 100 usuarios

En la figura 118 se observa que se ha realizado con éxito las 100 peticiones en las cuales el 17% se hicieron en un tiempo menor a 800 ms mientas que el 39% en un tiempo entre 800 ms y 1200 ms y el 45% restante en un tiempo mayor a 1200 ms mientras que en la figura 119 se puede observar el pico de peticiones por segundo a lo largo de la prueba con el número de usuarios activos.

Figura 118

Prueba de carga con 100 usuarios

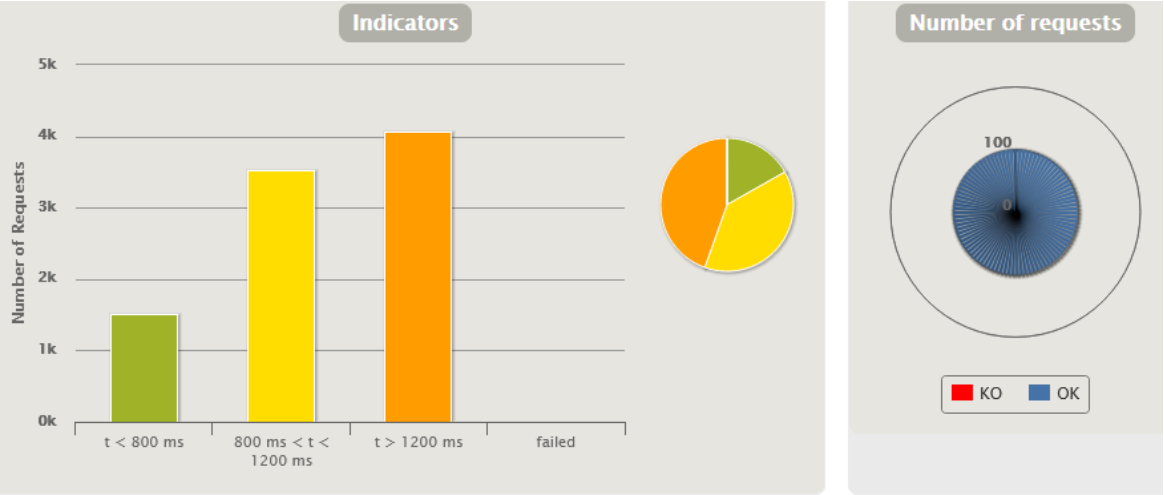
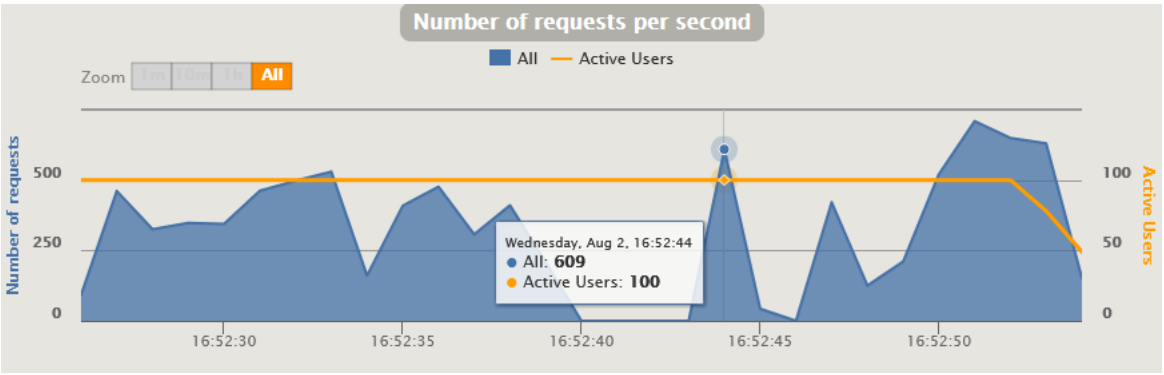


Figura 119

Pico de peticiones con 100 usuarios



Pruebas de carga 200 usuarios

En la figura 120 se observa que se ha realizado con éxito el 96% de las peticiones, pero con un tiempo mayor en las cuales el 8% se hicieron en un tiempo menor a 800 ms mientras que el 19% en un tiempo entre 800 ms y 1200 ms y el 69% restante en un tiempo mayor a 1200 ms, pero un 4% de las peticiones fallaron en la figura 121 se puede observar el pico de peticiones por segundo a lo largo de la prueba con el número de usuarios activos.

Figura 120

Prueba de carga con 200 usuarios

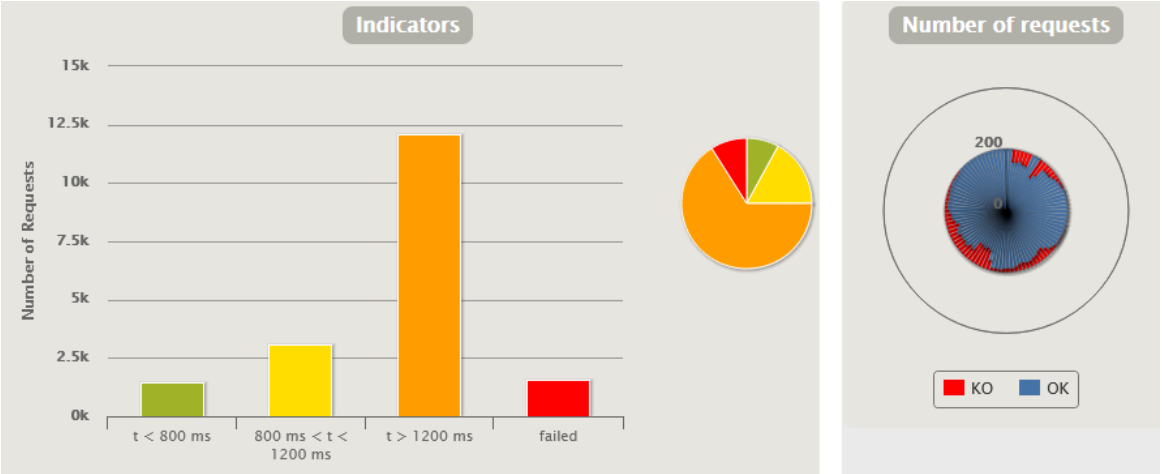
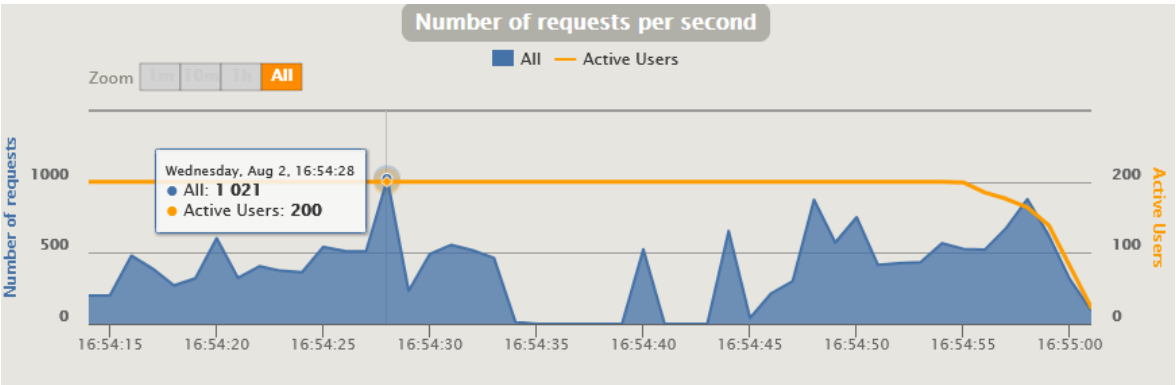


Figura 121

Pico de peticiones con 200 usuarios



Pruebas de carga 500 usuarios

En la figura 122 se observa que se ha realizado con éxito el 92% de las peticiones, pero con un tiempo mayor en las cuales el 2% se hicieron en un tiempo menor a 800 ms mientas que el 3% en un tiempo entre 800 ms y 1200 ms y el 87% restante en un tiempo mayor a 1200 ms, pero un 8% de las peticiones fallaron en la figura 123 se puede observar el pico de peticiones por segundo a lo largo de la prueba con el número de usuarios activos.

Figura 122

Prueba de carga con 500 usuarios

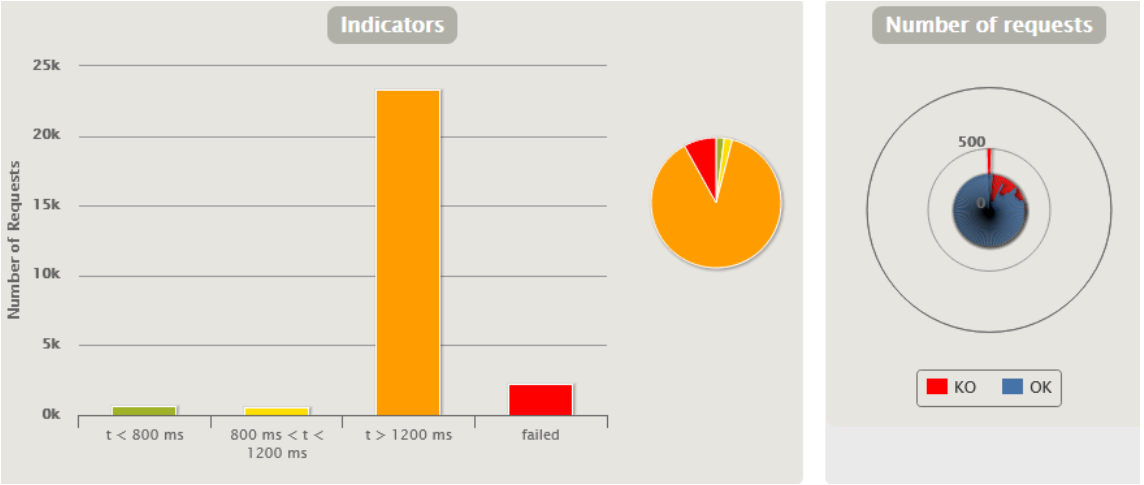
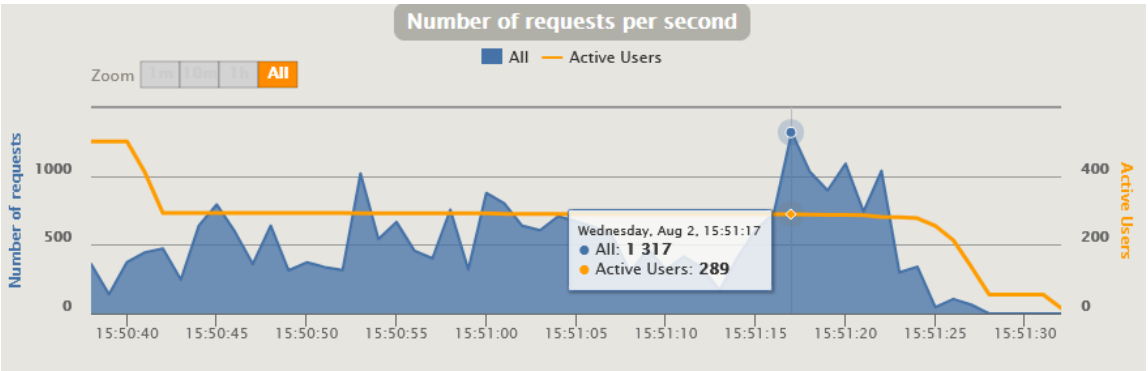


Figura 123

Pico de peticiones con 500 usuarios



Pruebas de carga 1000 usuarios

En la figura 124 se observa que a partir de los 1000 usuarios ya se observa un número significante de peticiones que fallaron las cuales representan el 71%, mientras que las que se realizaron con éxito se tiene el 1% en menos de 800 ms, el 2% con un tiempo entre 800 ms y 1200 ms y el 26% restante en un tiempo mayor a 1200 ms. En la figura 125 se puede observar el pico de peticiones por segundo a lo largo de la prueba con el número de usuarios activos.

Figura 124

Prueba de carga con 1000 usuarios

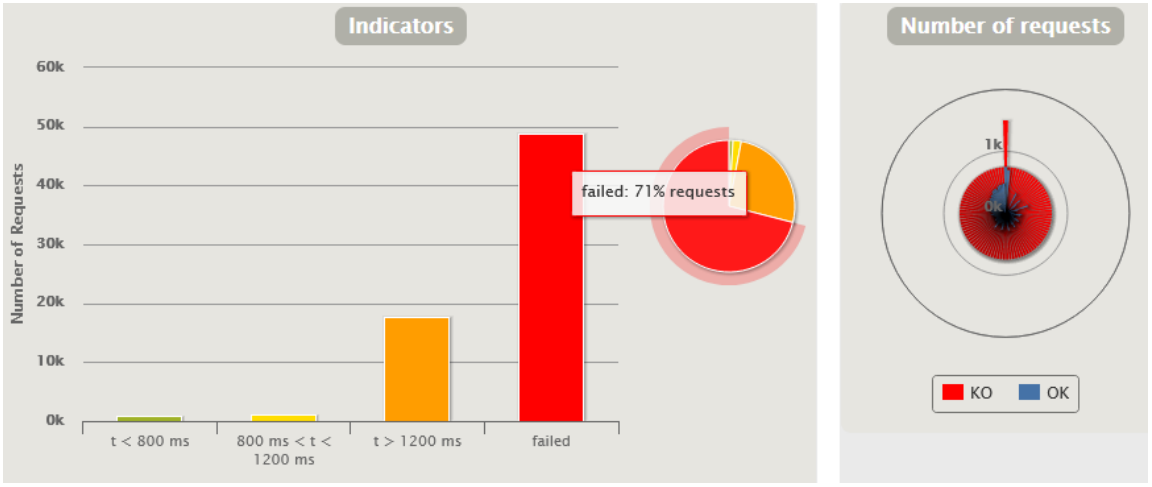
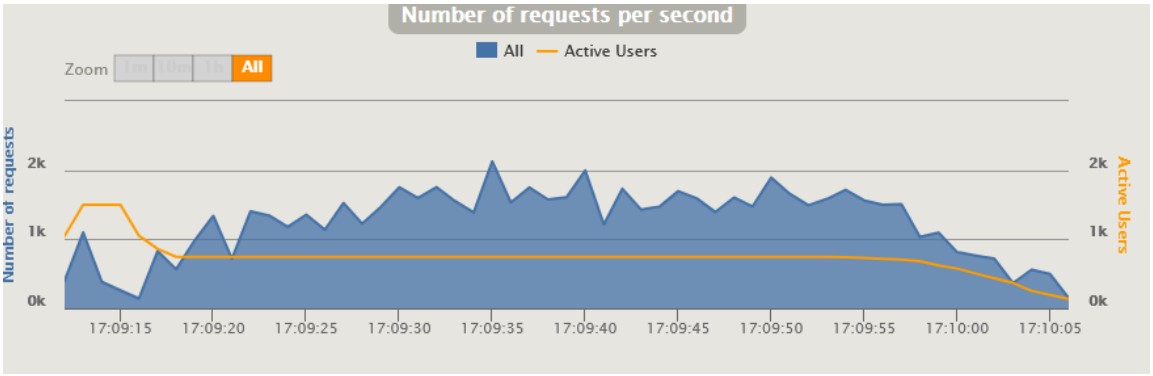


Figura 125

Pico de peticiones con 1000 usuarios



Finalmente se recopila los datos de todas las pruebas de carga utilizadas donde se puede observar que hasta 500 usuarios se realiza la mayor parte de peticiones, pero a partir de los 1000 usuarios ya el sistema colapsa y empieza a fallar casi todas las peticiones en la tabla 18 se muestra los valores obtenidos en las pruebas de carga.

Tabla 18

Datos de las pruebas de carga

Usuarios	Peticiones Totales	Máximas Pets/seg	Éxito [%]	Fallas [%]	Respuesta		
					t<800ms [%]	800 < t < 1200 ms [%]	t<1200ms [%]
10	910	364	100	0	93	7	0
50	4550	127	100	0	55	29	16
100	9100	254	100	0	17	39	45
200	18200	507	96	4	8	19	69
500	26780	1317	92	8	2	3	87
1000	3774	1340	49	71	2	6	42

Pruebas de usabilidad

Para evaluar la usabilidad de la interfaz y la experiencia del usuario frente a la arquitectura propuesta se utiliza el sistema de escalas de usabilidad (SUS). El test está conformado por diez cuestiones donde el puntaje en que están valoradas estas preguntas van del uno al cinco donde uno significa que no está de acuerdo con la pregunta y cinco que está de acuerdo con la pregunta. Para evaluar la experiencia de usuario se tomó una muestra de 10 personas las cuales interactuaron con la interfaz para observar el monitoreo de los prototipos realizados.

Las 10 personas que fueron seleccionadas para la aplicación de este test fueron entre estudiantes y graduados de la carrera de electrónica y también personas que no tienen que ver con una carrera técnica por lo que comprobaremos que tan amigable es la interfaz para cualquier persona que la utilice. Para la obtención de los resultados se debe considerar las siguientes directrices las preguntas impares al valor asignado por el usuario se le restará 1,

mientras que para las pares será de 5 menos el valor asignado por las personas que realicen el test, Finalmente se sumará todo y multiplicará por 2.5 para obtener el valor final para saber que tan amigable es el sistema con los usuarios. (Devin, 2017)

Como se observa en la tabla 19 el promedio del puntaje SUS fue de 88.75/100 con lo que se pudo verificar que el sistema es amigable con los usuarios. Además, se pudo observar que a las personas que se les realizó el test les llamó la atención que desde cualquier parte del mundo puedan monitorear los servicios básicos de un hogar.

Tabla 19

Resultados obtenidos de la prueba de usabilidad

Preguntas / Usuarios	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**Valor final sobre 100**
U1	5	2	5	1	5	1	4	1	4	2	92.5
U2	4	1	5	2	5	2	5	1	4	2	82,5
U3	5	1	5	2	4	1	5	2	4	2	87,5
U4	5	1	5	2	5	2	4	1	4	1	95
U5	5	1	4	3	5	1	5	1	5	3	87.5
U6	4	2	5	2	5	3	5	2	5	2	82,5
U7	5	2	5	2	4	3	5	1	5	1	87.5
U8	5	1	4	1	5	1	5	1	5	3	92.5
U9	3	2	5	1	4	1	5	1	5	2	87.5
U10	4	1	5	1	3	1	5	1	5	1	92.5
										Promedio	88,75

Conclusiones

El lenguaje de programación java permitió la implementación del procesamiento de eventos complejos, protocolos de comunicación y transmisión de datos entre las distintas capas del sistema de gestión de servicios básicos con lo que permite la creación de patrones de eventos complejos a partir de nodos function, operadores y nodos switch para el direccionamiento de los datos en el envío de notificaciones para servicios externos como el correo electrónico.

Mediante el protocolo MQTT se logra comunicar la capa física con la capa lógica utilizando el broker de mensajería Mosquitto con un tiempo de respuesta pequeño ya que MQTT cuando establece una conexión la mantiene abierta y se puede reutilizar siempre que sea necesario. Además, el envío de los datos en formato Json desde el gateway hacia el broker Mosquitto nos permite estructurar lo datos de mejor manera ocupando poco espacio de memoria, mejorando la agilidad en el procesamiento y la velocidad en el tráfico de red.

De acuerdo con el trabajo investigativo realizado sobre herramientas y tecnologías permitió el diseño y la implementación de una arquitectura basada en el procesamiento de eventos complejos tomando en cuenta factores físicos (sensores, controladores) o factores virtuales (protocolo mqtt, broker de mensajería, bases de datos) para el monitoreo de los servicios básicos en una ciudad inteligente en la cual se logró implementar una arquitectura de software confiable que se encuentra alojado en un servidor privado en la nube y que se puede acceder desde cualquier parte del mundo.

Como resultado de la elaboración de la arquitectura del sistema se utiliza una tecnología SDN (Software Defined Networking) ya que los prototipos únicamente son los encargados de medir o detectar algún cambio de las magnitudes físicas en el entorno y no es necesario programar cada prototipo manualmente para realizar algún cambio en el sistema,

sino que se puede programar el controlador central ubicado en la nube para así ofrecer más funcionalidades o mejoras en el sistema a futuro.

Mediante la implementación de la arquitectura basada en el procesamiento de eventos complejos en Node-Red permitió analizar en tiempo real cada uno de los prototipos, no únicamente desplegando la información de las variables sino tomando más de un evento a la vez como la hora, el día, la fecha y el valor de cada variable para así dependiendo de cada uno de estos eventos tomar un decisión automáticamente sin la necesidad que un operario realice alguna acción con lo que logramos mejorar la eficiencia en el sistema.

El presente proyecto se llevó a cabo para diseñar prototipos pequeños, portables y económicos para el monitoreo de los servicios básicos como electricidad, agua potable y recolección de residuos por lo que a partir de la implementación de estos mediante comunicación LoRa multipunto se obtiene un gran alcance, un bajo consumo de energía y que permite su instalación en cualquier tipo de lugares. La utilización de un Esp32 Lora como gateway concentrador logra bajar considerablemente los costos del proyecto en un 30% además que el sistema sigue siendo escalable y se puede conectar muchos más nodos a este gateway concentrador.

El presente proyecto con los resultados en las pruebas de carga se concluye la robustes del sistema alojado en el VPS establece que se podría admitir sin problemas hasta menos de 500 usuarios al mismo tiempo ya que superando este valor habrá peticiones fallidas y tiempos de respuesta altos, pero esto no es un gran problema debido a que la cantidad de datos que se envíen simultáneamente no serán tan elevados.

A partir de las pruebas de usabilidad mediante el sistema de escala SUS los puntajes obtenidos fueron de 88.75 sobre 100 con lo que se puede concluir que el sistema es amigable hasta para usuarios independientemente de conocimientos técnicos.

Recomendaciones

En la recepción de los datos en el gateway concentrador se debe colocar un tiempo prudente entre nodos para la lectura de los datos ya que el tiempo de respuesta de algunos sensores es alto y no se podría percibir de manera correcta el cambio de las variables en cada uno de los sensores utilizados.

Se recomienda dimensionar los elementos que se utilizarán para la construcción de los prototipos ya que con esto evitaremos problemas en las mediciones de las variables y también incremento innecesario en los costos del proyecto.

En las pruebas de carga del sistema se recomienda verificar si se encuentra instalado el jdk con la versión 1.8 o superior, con lo cual al momento de realizar las pruebas de carga en Intellij IDEA evitaremos problemas en la ejecución del código.

Trabajos Futuros

El gateway concentrador debe ser colocado en una caja dependiendo donde se lo vaya a instalar para protegerlo frente al agua y polvo.

Implementar protocolos de seguridad para los datos recopilados por los prototipos puesto que en la nube se encuentran expuestos a robos para con esto lograr confidencialidad de la información.

Implementar actuadores en los prototipos de consumo de agua o electricidad para que se pueda realizar el corte del servicio de forma remota además de colocar baterías externas en el prototipo de consumo de agua para que funcione en el caso de un corte de servicio eléctrico.

Realizar un sistema de facturación y contabilidad para que pueda ser complementado con este sistema de monitoreo para llevar un mejor control de los valores a pagar.

Utilizar una plataforma dedicada para IoT en la creación de HMI o Dashboard como Ubidots, Blink o AWS IoT para que pueda brindar más funcionalidades a los usuarios.

Bibliografía

Kaschel, H., & Pinto, E. (2020). *Análisis del estado del arte de los buses de campo aplicados al control de procesos industriales.* Universidad de Santiago de Chile, Santiago de Chile.

Acosta, E. (2020). *Sistema prototipo para el monitoreo inalámbrico de la cantidad de desechos de un contenedor de basura para ciudades inteligentes.* Quito : Escuela Politécnica Nacional.

Allauca Fajardo, B. D. (2020). *Desarrollo de un Sistema de Parqueadero Inteligente Mediante una Red LPWAN.* Sangolqui: ESPE.

Antonino, J., Yasin, S., & Rahman, S. (1 de Agosto de 2020). Waste Management and Prediction of Air Pollutants Using IoT and Machine Learning Approach. *Energies*. doi:https://doi.org/10.3390/en13153930

Asanza, V. (Septiembre de 2021). *Especificaciones del módulo ESP32* . Obtenido de https://vasanza.blogspot.com/2021/07/especificaciones-del-modulo-esp32.html

Ayala, J. J. (2022). *Implementación de un prototipo para la localización de personas de la tercera edad utilizando una aplicación móvil con sistema operativo android y sigfox.* Quito: Escuela Politécnica Nacional.

Ayllon, V. (2017). *CEP/ESP: Procesamiento y correlación de gran cantidad de eventos en arquitecturas SOA.* Sevilla: ISA Group.

Benitez, D. (2016). *Propuesta de arquitectura para IoT.* La Habana: Universidad Tecnológica de La Habana José Antonio Echeverría.

Biswajeeban, M., & Attila , K. (2021). Stress-Testing MQTT Brokers: A Comparative Analysis of Performance Measurements. *Energies 2021*. doi:https://doi.org/10.3390/en14185817

Borja, A. (2021). *Implementación de una red LPWAN para el control de consumo de agua potable utilizando tecnología LoRa.* Quito: Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica y Telecomunicaciones.

Bouskela, M., Casseb, M., & Bassi, S. (2016). *La ruta hacia las Smart Cities.* Banco Interamericano de Desarrollo (BID).

Cases, E. (11 de Junio de 2014). *Apache HTTP Server: ¿Qué es, cómo funciona y para qué sirve?* Obtenido de Agencia Marketing Digital en Barcelona: https://www.ibxagency.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/

Castillo, I., & Murillo, A. (2017). *Diseño de una red de comunicaciones para la medición y control del consumo de agua potable en tiempo real para la parroquia Yaruquíes.* Riobamba: Escuela superior politécnica de Chimborazo.

Castillo, P. (2018). *Monitorización de parámetros medioambientales mediante.* Jaén: Escuela Politécnica Superior de Linares.

Chawla, A., Babu, P., & Gawande, T. (29 de Marzo de 2021). Intelligent Monitoring of IoT Devices using Neural Networks. *IEEE Explorer*. doi:10.1109/ICIN51074.2021.9385543

Chen, F.-H., & Wen-Chiung , H. (10 de Mayo de 2020). Constructing Home Monitoring System with Node-RED. págs. 1701-1710. doi:https://doi.org/10.18494/SAM.2020.2686

Chen, Y. (18 de Febrero de 2018). Water quality monitoring in smart city: A pilot project. *Automation in Construction*, págs. 307-316. doi:https://doi.org/10.1016/j.autcon.2018.02.008

Clarke, S., & Khazael, B. (2021, Octubre 14). Complex Event Processing in Smart City Monitoring Applications. *IEEE Access*, pp. 143150 - 143165. doi:10.1109/ACCESS.2021.3119975

Coccoli, M., & Fusco, A. (3 de Septiembre de 2021). A cloud-based cognitive computing solution with interoperable applications to counteract illegal dumping in smart cities. *SpringerLink*, págs. 95-113. doi:https://doi.org/10.1007/s11042-021-11238-8

Coro, R. (26 de Septiembre de 2022). *Diseño e implementación de un sistema de monitoreo y gestión IoT de los puentes grúa de la empresa SEDEMI.* Quito: Repositorio ESPE. Obtenido de http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/35991/T-ESPE-052655.pdf?sequence=5&isAllowed=y

Devin, F. (25 de Febrero de 2017). *UXpañol*. Obtenido de https://uxpanol.com/teoria/sistema-de-escalas-de-usabilidad-que-es-y-para-que-sirve/

FG-SSC. (2015). *Grupo Temático sobre Ciudades Inteligentes y Sostenibles.* Ginebra: Unión Internacional de Telecomunicaciones.

Fuentes , D. (28 de Agosto de 2020). Smart water consumption measurement system for houses using IoT and cloud computing. *SpringerLink*. doi:https://doi.org/10.1007/s10661-020-08535-4

Gallegos, M. (2011). *Introducción a los Servicios Web.* Buenos Aires: Universidad Tecnológia Nacional.

García, F. (Marzo de 11 de 2020). Método para la revisión sistemática de literatura. *Recursos docentes de la asignatura Procesos y Métodos de Modelado para la Ingeniería Web y Web Semántica*. doi:10.5281/zenodo.3692796

Glória, A., Dionisio, C., & Simões, G. (4 de Marzo de 2020). Water Management for Sustainable Irrigation Systems Using Internet-of-Things. *Sensors*. doi:https://doi.org/10.3390/s20051402

Grapheverywhere. (2021). *Tipos de bases de datos | Clasificación por contenido y modelo*. Obtenido de https://www.grapheverywhere.com/tipos-bases-de-datos-clasificacion/

Guevara, N., & Y. B. (15 de Junio de 2022). Development of a low-cost IoT system based on LoRaWAN for monitoring variables related to electrical energy consumption. *HardwareX*. doi:https://doi.org/10.17632/bkhrks6x3m.2

Heltec. (2020). *Wifi Lora 32 Node Development Kit.* Heltec Automation.

Hongqiang , J., & Xinxin , W. (29 de Mayo de 2020). Service Oriented Cloud Computing Trusted. *Journal of Information Processing Systems*, págs. 1281-1292. doi:https://doi.org/10.3745/JIPS.03.0153

Intriago, J. (2020). *Funcionamiento MQTT*. Obtenido de iberasync.es: https://iberasync.es/protocolo-mqtt-en-iot/

Jay, L., Behrad, B., & Kao, H.-A. (2015). *A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems.* Estados Unidos: Elsevier.

Khazael, B., & Tabatabaee, H. (14 de Enero de 2023). Geospatial complex event processing in smart city applications. *Simulation Modelling Practice and Theory, 122*. doi:https://doi.org/10.1016/j.simpat.2022.102675.

Lagos, G., & Benavides , L. (2021). Ciudades inteligentes y su importancia frente al Covid-19. *Qualitas*, 101-115.

Lora-Alliance. (2018). *LoRa Alliance*. Obtenido de LoRa Alliance: https://www.lora-alliance.org

Lyaskov, M., Spasov, G., & Petrova, G. (17 de Noviembre de 2017). A practical implementation of smart home energy data storage and control application based on cloud services. *2017 XXVI International Scientific Conference Electronics (ET)*. doi:10.1109/ET.2017.8124387

Mahedero, F. (2020). *Desarollo de una aplicación IoT para el envío de imágenes mediante el protocolo MQTT.* Universidad Politécnica de Valencia, Valencia.

Morocho , V., & Quinapanta, J. (2016). *Diseño e implementación de un sistema multimodal que permita monitorear y controlar la recolección de desechos sólidos en tiempo real para Smart cities.* Riobamba: Escuela superior politécnica de Chimborazo.

Muayad, A. (03 de Octubre de 2021). Smart City in Urban Innovation: Concept, Management. *International Journal of Advanced Engineering Research and*, págs. 1-10.

Pietro, M. (2019). *Intro to MQTT.* Valencia: Universitat Politécnica de Valencia.

Pilco, B., & Mendez, J. (2021). *Diseño e implementación de un lenguaje especifico de dominio (DSL) para el procesamiento de eventos complejos (CEP) en sistemas ciberfísicos (CPS).* Quito: Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica, Automatización y Control.

Quimbita, M. A. (2018). *Evaluación de pasarela LoRa/LoRaWAN.* Valencia: Universidad Politécnica de Valencia.

Rahamani, A., & Babei, Z. (27 de Octubre de 2020). Event-driven IoT architecture for data analysis of reliable healthcare application using complex event processing. *SpringerLink*, págs. 1347-1360. doi:10.1007/s10586-020-03189-w

Raj, P., & Raman, A. (2017). *The Internet of Things.* Boca Raton: Taylor & Francis Group.

Sánchez, F., & Cano, A. (21 de Diciembre de 2021). Design and Implementation of a Smart Energy Meter Using a LoRa Network in Real Time. *Electronics*. doi: https://doi.org/10.3390/electronics10243152

Sikora, D. (1 de Junio de 2017). Factores de desarrollo de las ciudades inteligentes. *Revista Universitaria de Geografía*, págs. 135-152.

Sinchangreed, V., & Egkarin, W. (25 de Abril de 2022). Web services performance evaluation on single board computers for mobile applications and iot devices. *Suranaree journal of science and technology*.

Târnaucă, B., & Comnac, V. (4 de Julio de 2013). Traffic condition monitoring using complex event processing. *IEEE Xplore*. doi:10.1109/ICSSE.2013.6614645

TIBCO. (2019). *¿Qué es el procesamiento complejo de eventos (CEP)?* Palo Alto: TIBCO.

Wurst. (8 de Noviembre de 2021). *Github*. Obtenido de https://github.com/tobozo/ESP32-Chimera-Core/issues/67

Zambrano, R. (2019). *Diseño e implementación de una propuesta de arquitectura de sistema de gestión de consumo eléctrico en el hogar aplicando una plataforma de cloud computing y un algoritmo de machine learning.* Quito: Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica y Telecomunicaciones.

Acrónimos

- SMS. Mapeo Sistemático de la Literatura.
- SLR. Revisión Sistemática de la Literatura
- IoT. Internet de las Cosas
- CPE. Procesamiento de eventos complejos
- MQTT. Messege Queuing Telemetry Transport
- SUS. System Usability Scale
- SDN. Software Defined Networking
- LoRa. Long Range
- IT. Tecnologías de la Información
- VPS. Servidor Privado Virtual

Apéndices

Apéndice A: Manual de usuario.

Apéndice B: Manual de instalación.