SISTEMA PARA LA RECOLECCIÓN DE DATOS DE DISPOSITIVOS UTILIZANDO LA RED LORA

Cabezas Moreno Roberto Carlos
López Alvarado Carmen Elisa
Rivera Pascasio Dennys Alberto
Ungo Muñoz Rodrigo Antonio

Asesora: Aldana Calderón Elisa Cristina

Universidad José Simeón Cañas,
Facultad de Ingeniería y
Arquitectura

ealdana@uca.edu.sv

El proyecto aborda el tema de la recolección de dispositivos que estén instalados en un área extensa, como lo es el campus de la Universidad Centroamericana José Simeón Cañas (UCA), utilizando la red LoRa. La red LoRa fue instalada en la UCA utilizando las bases propuestas por previas investigaciones enfocadas en la propagación y funcionamiento de la tecnología LoRa.

El objetivo de este proyecto es el desarrollo de un sistema informático que permita la recolección de datos de diferentes dispositivos instalados en la UCA a través de la red LoRa y presentarlos al público general a través de un dashboard en una aplicación web y móvil.

Para lograr dicho objetivo se utilizó la metodología de desarrollo SCRUM, que consiste en el uso de sprints, los cuales son cortos tiempos de desarrollo en los cuales el equipo desarrolla una funcionalidad del sistema. Esta metodología es iterativa, lo que significa que los sprints se repiten cada cierto tiempo (usualmente dos semanas). SCRUM también utiliza épicas, las cuales representan los módulos generales de la aplicación, como el backend y el frontend; y las historias de usuario, las cuales representan cada funcionalidad del sistema, como la obtención de datos con LoRa. En la metodología SCRUM se realizan cierta cantidad de historias de usuario cada sprint de dos semanas de forma repetida hasta finalizar con el sistema.

El resultado final fue un sistema completo que recolecta los datos de dispositivos instalados en el campus de la UCA (en este proyecto se utilizaron dos dispositivos), y presenta los datos al público en un dashboard con gráficas que permiten una mejor lectura y análisis de la información. Este sistema consiste en un backend, un frontend y una base de datos, además de una aplicación móvil como alternativa a la versión web.

Palabras claves—LoRa, LoRaWAN, SCRUM.

I. INTRODUCCIÓN

LoRa (abreviatura de Long Range), es una técnica de modulación de espectro ensanchado derivada de la tecnología Chirp de Espectro Ensanchado (Chirp Spread Spectrum o CSS). En los últimos años, se ha visto un aumento en el interés en el uso de la técnica de comunicación LoRa; esto debido a la habilidad de esta red de permitir la comunicación de datos entre sistemas a larga distancia (de 10 a 20 kilómetros), mientras utiliza un bajo consumo eléctrico [1]. Dentro del contexto que concierne a su uso, la tecnología LoRa conforma la capa física del sistema; es decir, los módulos encargados de la transmisión de datos.

Por su parte, la especificación LoRaWAN es el protocolo que vuelve posible la conexión entre dispositivos, conformando la capa MAC aplicada a los dispositivos físicos [2]. La figura 1 ilustra la arquitectura de LoRa y LoRaWAN mientras la figura 2 muestra el funcionamiento en conjunto de ambas tecnologías.

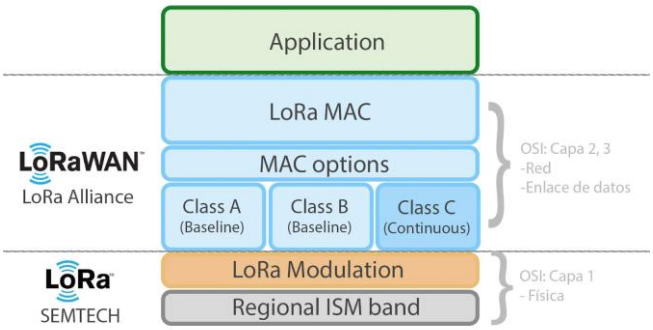

Fig. 1 Modelo OSI de capas sobre las que se implementan LoRa y LoRaWAN [3].

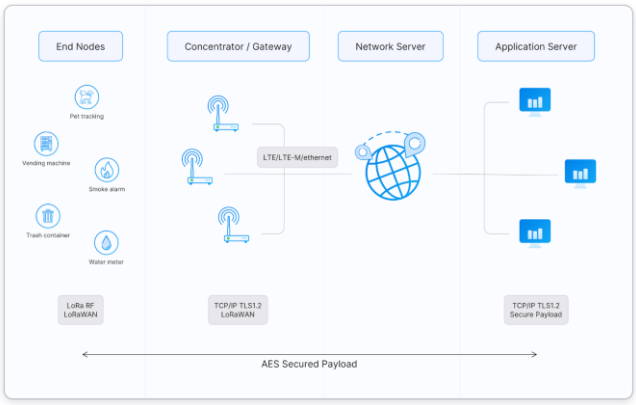

Fig. 2 Esquema de la red LoRaWAN [4].

El siguiente trabajo parte de la investigación de las tecnologías LoRa y LoRaWAN y su papel que desempeñan como medio físico e intermediario que permite realizar conexiones con diferentes dispositivos configurados para realizar mediciones y extracción de datos, aprovechando su sistema de seguridad basado en la criptografía, su largo alcance (considerando los diferentes obstáculos que pueda presentar de acuerdo al campo en donde se realiza la medición así como) y su bajo consumo de energía para el apoyo a la recolección de datos ocupados para el desarrollo de la solución informática planteada.

Asimismo, se abordará el beneficio obtenido de contar con un medio informático que permita el análisis y consumo de datos obtenidos a través de dispositivos configurados con la red LoRa y posicionados en el campus de la Universidad Centroamericana José Simeón Cañas (UCA), los cuales cumplen con una función medidora. Partiendo con una amplia investigación que permitirá conocer conceptos, características, así como ventajas y desventajas de trabajar con la red LoRa, el fin último será tanto buscar una solución informática óptima, escalable y adaptable, como el aprovechamiento de la red y sus funcionalidades para la obtención de información de manera concisa.

Se profundizará en conocer la manera de trabajo del gateway LoRaWAN, su configuración y las herramientas que brinda para realizar conexiones, crear aplicaciones y obtener datos mediante una configuración propia.

Como también, se plantea trabajar con tecnologías fáciles de manipular para su correcto funcionamiento y posterior

mantenimiento, brindando así un apoyo para la escalabilidad del proyecto mediante procedimientos estables abordados en la investigación, configuración, desarrollo e implementación que permitan una buena conexión con los dispositivos, así como un buen manejo de los datos obtenidos.

El objetivo general de este proyecto es el desarrollo de un sistema informático que permita la recolección, envío, almacenamiento y visualización de datos recibidos desde cualquier dispositivo ubicado en la UCA utilizando la red LoRa. Para lograr esto es necesario el desarrollo de una aplicación web que permita la recolección y la publicación de los datos de los sensores instalados en la UCA. Esta aplicación web es conformada por un backend web, el cual cumple la función de recibir los datos de todos los sensores y almacenarlos en la base de datos para que puedan ser consultados, y por un frontend web el cual consume los datos del backend y muestra los datos de todos los dispositivos en un dashboard con gráficas las cuales pueden ser utilizadas para el estudio de los datos enviados por todos los dispositivos del campus, así como la opción de mostrar en una tabla todos los datos de medición de los dispositivos y finalmente descargar todas las mediciones de cada dispositivo para su futuro estudio. Finalmente se propone el desarrollo de un prototipo de aplicación móvil para dispositivos Android que permite la visualización de los datos de los dispositivos en forma de gráficos, de forma similar al frontend web.

II. MATERIALES Y MÉTODOS

A. *Recolección de datos usando la red LoRa*

Para poder recolectar datos de diferentes dispositivos instalados en la UCA usando la red LoRa es necesario poseer dos cosas: un gateway LoRaWAN que reciba los datos de los dispositivos y un módulo LoRA que se conecta al dispositivo y que se encarga de enviar los datos al gateway.

El gateway LoRaWAN es un dispositivo que recibe los datos de cualquier dispositivo que se conecta a este. El gateway de la UCA es proveído por The Things Network (TTN), una compañía especializada en conexiones inalámbricas utilizando la red LoRaWAN. El gateway se adquiere de TTN e incluye una antena que se instala en el lugar ideal para que todos los dispositivos de la zona se puedan conectar con este. El gateway ya estaba configurado previamente por la UCA. Para mayor información de la configuración de LoRaWAN en la UCA, ver [1] y [2].

Como dispositivo de conexión al gateway se hizo uso de una placa de desarrollo Arduino UNO con un módulo LoRa RF96, conectados a través de una breadboard. El Arduino UNO es una placa de desarrollo programable de código libre que se puede utilizar para diferentes aplicaciones, como detección de sensores, activación de mecanismos de forma remota, etc. Para los usos de este proyecto se enfocó en la detección de datos de sensores; es decir, los sensores detectan fenómenos específicos en el ambiente y el Arduino UNO recibe los datos de los sensores a través de un código que se sube a la placa. El Arduino UNO luego envía, a través de dicho código, los datos del sensor al módulo LoRa RF96. El módulo LoRa RF96 es el que se encarga de conectar y transmitir los datos del dispositivo hacia el gateway LoRaWAN a través del protocolo homónimo. En la figura 3 se muestra una representación gráfica de cómo se conecta el Arduino UNO al módulo LoRa RF96.

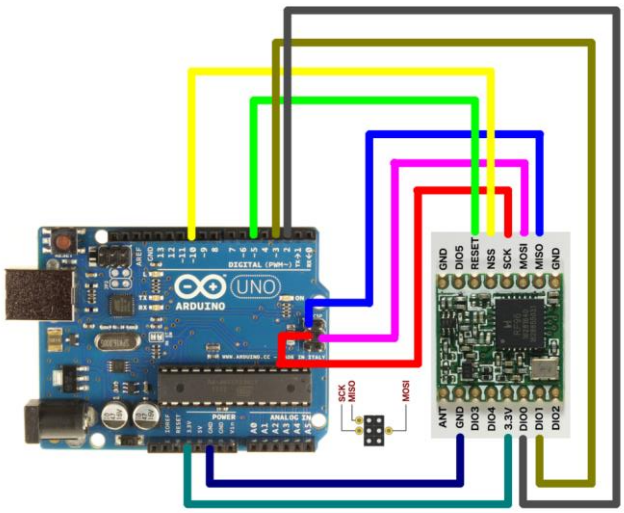

Fig. 3 Diseño de circuito para conexión base de módulo lora RF96 con placa de desarrollo Arduino UNO.

B. *Conexión al gateway LoRaWAN*

El sistema LoRa, utilizado actualmente en la UCA, fue establecido gracias a los esfuerzos de trabajos de graduación anteriores. La primera incursión en este campo para la UCA se llevó a cabo mediante el trabajo de Garcilazo García y Santos Salgado en 2022. Su proyecto consistió en la prueba de transmisión de información entre dos dispositivos LoRa.

Posteriormente, se implementó un sistema completo de gateway LoRaWAN. Para esta instalación, se adquirió equipo de The Things Network (TTN), una compañía especializada en hardware que utiliza la tecnología LoRa. El servidor LoRa de la UCA, proporcionado por TTN, ofrece un portal donde se puede configurar todo lo necesario para su funcionamiento. En la figura 4 se muestra una captura de pantalla de un ejemplo de gateway LoRaWAN.

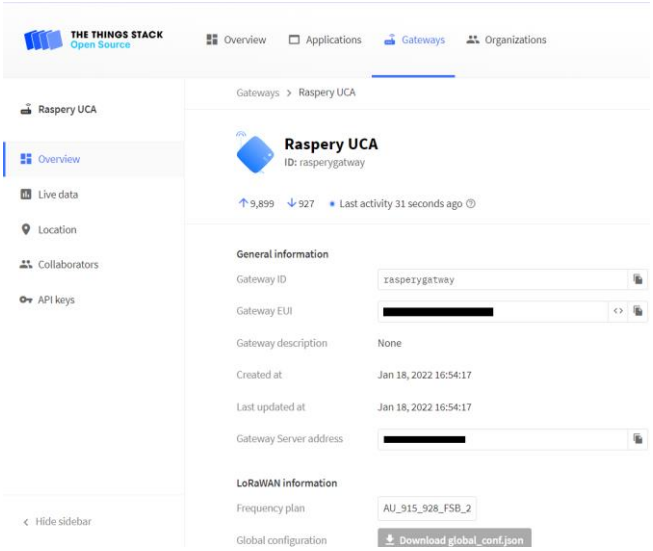

Fig. 4 Captura de pantalla del gateway LoRaWAN de la UCA.

El gateway de TTN permite crear “aplicaciones”, que son proyectos dentro del servidor. Estas aplicaciones permiten agregar y guardar varios dispositivos finales para su uso posterior. Esta funcionalidad es útil porque mantiene los dispositivos relacionados en un mismo lugar.

Una vez creada la aplicación, se procede a la adición de un dispositivo final que se conectará al gateway LoRaWAN, como se muestra en la figura 5.

Fig. 5 Agregar un dispositivo al gateway LoRaWAN

Para configurar correctamente el dispositivo, se deben llenar manualmente los siguientes parámetros:

- Plan de frecuencia: Se elige la banda de frecuencia en la que se transmitirá el dispositivo final. En este caso, se optó por “Australia 915-928 MHz, FSB (usado por TTN)” debido a que el gateway LoRaWAN de la UCA también opera en esta frecuencia.
- Versión LoRaWAN: Se selecciona la versión “MAC V1.0.2” para asegurar que tanto el gateway como el dispositivo final trabajen con la misma versión.
- Región: Los parámetros regionales especifican la frecuencia y otras configuraciones de comunicación para diferentes áreas geográficas. Se eligió la versión “PHY V1.0.2 REV B” para mantener la coherencia con la configuración del gateway.
- Método de activación: Se optó por el método OTAA (Over-The-Air Activation) para la autenticación y enlace entre el dispositivo y el gateway LoRaWAN.
- Opciones de red y clúster: Se seleccionaron las opciones para utilizar la configuración MAC por defecto de la red local y usar servidores backend de LoRaWAN externos de TTN. La dirección del servidor de red y de acceso se estableció como “dei1.uca.edu.sv”.

Además, se asignaron valores específicos para DevEUI, AppEUI y AppKey, que son esenciales para garantizar el acceso del dispositivo final al gateway LoRaWAN. El DevEUI y el AppEUI son creados por el usuario, mientras que la AppKey es creada por el gateway. Estos valores luego se insertan en el código del dispositivo como se ve en la figura 6.

Fig. 6 Porción de código del dispositivo donde se asignan los valores de autenticación del gateway.

La función “do_send” del código de programación del dispositivo se encarga de establecer la conexión con el gateway y enviar los datos. Se realizan lecturas del sensor (por ejemplo, un medidor de humo MQ-2) y se calculan los valores de concentración de partículas en el aire. Estos valores se envían a través de LoRa utilizando la función “LMIC_setTxData2” (ver figura 7).

```
void do_send(osjob_t* j){
    // Comprobar si no hay un trabajo TX/RX actual en ejecución
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Preparar la transmisión de datos ascendentes en el próximo momento posible.
        // El valor debe de ser descompuesto para ser enviado byte  por byte.
        float R0 = 1100;
        int sensorValue = analogRead(MQ2);
        sensor_volt = ((float)sensorValue / 1024) * 5.0;
        RS_gas = ((5.0-sensor_volt)/sensor_volt)*RL; // Depende de RL en tu módulo
        ratio = RS_gas / R0; // ratio = RS/R0
        PPM = 585.19*(pow(ratio,-2.042));
        // Convertir a String el valor que se va a mandar
        // para descomponerlo por bytes.
        payload = String(PPM);
        // Función de LMIC para descomponer el valor
        // a enviar en formato de bytes.
        LMIC_setTxData2(1, payload.c_str(), payload.length(), 0);
        Serial.println(payload);
        Serial.println(F("Packet queued"));
    }
    // El próximo TX está programado después del evento TX_COMPLETE.
}
```

Fig. 7 Código de función do_send que se encarga de leer el valor del sensor y enviarlo al gateway LoRaWAN.

Una vez enviados los datos, el gateway LoRaWAN los recibe y los muestra en su consola. El Payload formatter permite personalizar el formato en el que se presentan los datos. En este caso, se utiliza JavaScript para transformar los valores en un formato adecuado para su posterior procesamiento en un backend web. El payload formatter se puede apreciar en la figura 8.

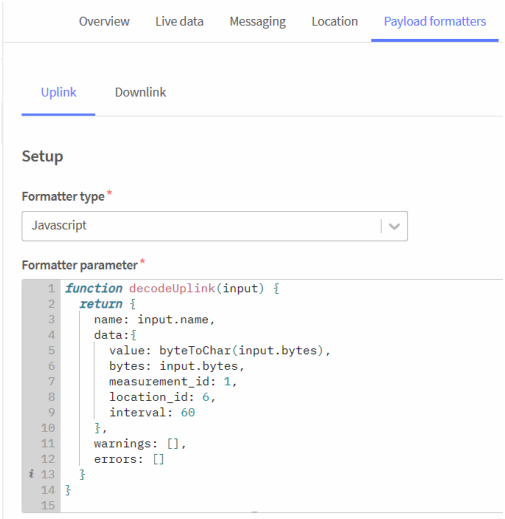

Fig. 8 Payload formatter de tipo JavaScript con función para decodificar el paquete de datos recibido.

Este proceso se puede replicar tantas veces como sea necesario para crear una red de dispositivos, que en conjunto pueden ser una herramienta de gran utilidad para la mejora continua del entorno en el campus. Sin embargo, los datos de los dispositivos son difíciles de manejar e interpretar cuando se leen directamente desde el gateway LoRaWAN. Este último no fue diseñado con una experiencia de usuario intuitiva, sino más bien para usuarios avanzados. Además, los datos almacenados en el gateway LoRaWAN son volátiles, lo que significa que se pierden al recargar la página donde está alojado el gateway.

Por estas razones, depender únicamente del gateway LoRaWAN para recolectar, almacenar y mostrar los datos para un posterior análisis no es una opción viable. Es necesario implementar un sistema más robusto y amigable para gestionar y visualizar los datos de los dispositivos.

Para abordar esta necesidad, se ha desarrollado una aplicación web que recibe, almacena y presenta los datos de los dispositivos de manera ordenada y accesible. A continuación, se detalla el proceso de almacenamiento y visualización de datos en la aplicación.

C. Backend web

Recepción de Datos: El proceso comienza cuando un dispositivo captura un dato, lo procesa y lo transmite hacia el gateway LoRaWAN. Una vez en el gateway, un webhook se encarga de realizar una petición POST al servidor de la aplicación, a un endpoint previamente especificado. La figura 9 ilustra una versión simplificada de este proceso.

Figura 9. Esquema del proceso de captura y almacenamiento de datos.

Manejo de Rutas y Middleware: Utilizando Express (un framework para Node.js), se configuran las rutas y middleware necesarios para manejar las peticiones POST y GET. Por ejemplo, cuando se recibe una petición POST en la ruta "/uplinks", se ejecuta el middleware "postUplink" (ver figura 10).

```
router.post('/uplinks', postUplink);
router.get('/dashboard', obtenerDatos);
router.get('/:id', obtenerDatosDispositivo);
router.get('/:id/tabla', lecturasEnTabla);
```

Fig. 10 Endpoints a utilizar en esta aplicación.

El middleware "postUplink" es una función asíncrona que recibe dos parámetros: un objeto petición (request) y un objeto respuesta (response). En caso de éxito, se crean registros en la tabla "Dispositivos" y "Lecturas". Si el dispositivo ya está registrado, solo se crea un nuevo registro en la tabla "Lecturas" (ver figura 11).

```
const postUplink = async (req, res) => {
    try {
        const lectura = req.body.uplink_message.decoded_payload
        const dispositivoId = req.body.end_device_ids.device_id
        const dispositivo = await Dispositivo.findByPk(dispositivoId)

        if(!dispositivo){
            await Dispositivo.create({
                dispositivo_id: dispositivoId,
                medicion_id: lectura.measurement_id,
                ubicacion_id: lectura.location_id,
                dispositivo_lectura_intervalo: lectura.interval
            });

            await Lectura.create({
                lectura_valor: lectura.value,
                dispositivo_id: dispositivoId
            });
        }
        else{
            await Lectura.create({
                lectura_valor: lectura.value,
                dispositivo_id: dispositivoId
            });
        }
        return res.end()
    } catch (error) {
        console.log(error)
    }
}
```

Fig. 11 Middleware para registrar dispositivos y lecturas.

Consultas a la Base de Datos: Se han definido tres endpoints para solicitudes GET: "obtenerDatos", "obtenerDatosDispositivo" y "lecturasEnTabla". Estos endpoints consultan la información necesaria en la base de datos y la retornan al cliente (ver figuras 12, 13, 14 y 15).

```
const obtenerDatos = async (req, res) => {
    try {
        const dispositivos = await Dispositivo.findAll({
            include: [{
                model: Ubicacion,
                as: "ubicacion",
                attributes: ["ubicacion_nombre", "ubicacion_tipo"]
            }, {
                model: Medicion,
                as: "medicion",
                attributes: {
                    exclude: ["medicion_id", "createdAt", "updatedAt"]
                }
            }],
            order: ["dispositivo_id"],
        });

        if (dispositivos.length <= 0) {
            throw new ErrorConCodigo("No se encontraron dispositivos", 404);
        }

        for (const dispositivo of dispositivos) {
            let lecturasRecientes = await Lectura.findAll({
                where: {
                    dispositivo_id: dispositivo.dispositivo_id
                },
                attributes: {
                    exclude: ["updatedAt"]
                },
                order: ["createdAt"],
                limit: 10,
                raw: true
            });
            dispositivo.dataValues.lecturasRecientes = lecturasRecientes;
        }

        return res.json(dispositivos);
    } catch (error) {
        if (error.status) {
            res.statusMessage = error.message;
            return res.status(error.status).send(error.message);
        }
        console.log(error)
        return res.status(500).end();
    }
}
```

Fig. 12. Middleware para consultar las lecturas de todos los dispositivos registrados.

```
const obtenerDatosDispositivo = async (req, res, next) => {
    try {
        const dispositivo_id = req.params.id;

        const dispositivo = await Dispositivo.findByPk(dispositivo_id, {
            include: [{
                model: Ubicacion,
                as: "ubicacion",
                attributes: ["ubicacion_nombre", "ubicacion_tipo"]
            }, {
                model: Medicion,
                as: "medicion",
                attributes: {
                    exclude: ["medicion_id", "createdAt", "updatedAt"]
                }
            }],
        });

        if (!dispositivo) {
            throw new ErrorConCodigo("Dispositivo no encontrado", 404);
        }

        const lecturasRecientes = await Lectura.findAll({
            where: {
                dispositivo_id
            },
            attributes: {
                exclude: ["updatedAt"]
            },
            order: [
                ["createdAt", "DESC"]
            ],
            limit: 10,
            raw: true
        });

        if (!lecturasRecientes.length) {
            throw new ErrorConCodigo("No hay lecturas disponibles", 404);
        }
```

Fig. 13 Middleware para consultar las lecturas recientes y de hace 24 horas de un dispositivo específico (primera mitad).

```
        lecturasRecientes.reverse();

        let fecha = new Date(lecturasRecientes[0].createdAt);

        fecha.setDate(fecha.getDate() - 1);
        fecha.setMinutes(fecha.getMinutes() - 1);
        fecha = fecha.toISOString();

        const lecturasAnteriores = await Lectura.findAll({
            where: {
                dispositivo_id,
                createdAt: {
                    [Op.gte]: fecha
                }
            },
            attributes: {
                exclude: ["updatedAt"]
            },
            order: [
                ["createdAt", "ASC"]
            ],
            limit: 10,
            raw: true
        })

        dispositivo.dataValues.lecturasRecientes = lecturasRecientes;
        dispositivo.dataValues.lecturasAnteriores = lecturasAnteriores;

        return res.json([dispositivo]);
    } catch (error) {
        if (error.status) {
            res.statusMessage = error.message;
            return res.status(error.status).send(error.message);
        }
        console.log(error)
        return res.status(500).end();
    }
}
```

Fig. 14 Middleware para consultar las lecturas recientes y de hace 24 horas de un dispositivo específico (segunda mitad).

```
const lecturasEnTabla = async (req, res) => {
    try {

        const dispositivo_id = req.params.id;
        const { fechaInicio, fechaFin } = req.query;

        let fechaFinModificada = new Date(fechaFin);

        fechaFinModificada.setDate(fechaFinModificada.getDate() + 1);
        fechaFinModificada = fechaFinModificada.toISOString().slice(0, 10);

        const lecturas = await Lectura.findAll({
            where: {
                dispositivo_id,
                createdAt: {
                    [Op.and]: {
                        [Op.gte]: fechaInicio,
                        [Op.lt]: fechaFinModificada
                    }
                }
            },
            attributes: {
                exclude: ["lectura_id", "updatedAt", "dispositivo_id"]
            },
            order: [
                ["createdAt", "ASC"]
            ],
            raw: true
        });

        if (lecturas.length <= 0) {
            throw new ErrorConCodigo("No hay lecturas en ese rango", 404);
        }

        return res.json(lecturas);
    } catch (error) {
        if (error.status) {
            res.statusMessage = error.message;
            return res.status(error.status).send(error.message);
        }
        console.log(error)
        return res.status(500).end();
    }
}
```

Fig. 15 Middleware para consultar las lecturas de un dispositivo específico con rango de fechas.

III. RESULTADOS Y DISCUSIÓN

A. Frontend Web

La interfaz de usuario en el sistema web consta de tres vistas principales. La página de inicio proporcionará una visión general, mostrando resúmenes de los dispositivos y sus últimas lecturas. Cada gráfico incluirá información relevante como:

- Título: el nombre de la medición del fenómeno.
- Subtítulo: la ubicación y la fecha de la lectura más reciente.
- Cuerpo principal: representará las 10 lecturas más recientes y su promedio, con ejes que indican el valor y la hora (ver figura 16).

Fig. 16 Página de inicio.

La página dedicada a la información de un dispositivo individual permitirá explorar en detalle los datos de un dispositivo específico, incluyendo gráficos interactivos para visualizar las lecturas.

Para presentar los valores de las lecturas, se utilizarán distintos tipos de gráficos:

- Gráficos de líneas: ideales para seguir tendencias a lo largo del tiempo.
- Gráficos de columnas: útiles para comparar valores entre diferentes dispositivos o fechas.
- Gráficos de área: permiten visualizar la distribución de los datos.
- Gráficos de barras: adecuados para mostrar valores individuales.

Además, se podrá cambiar el tipo de gráfico y comparar lecturas de 24 horas antes (ver figura 17).

Fig. 17 Información detallada de dispositivo que mide variación de luz.

Por último, es posible visualizar la información en formato de tabla, lo cual permite ver más de las 10 lecturas mostradas en los gráficos. Para ello, se debe ingresar una fecha para consultar un solo día, o dos fechas para consultar un rango. La tabla incluye fecha, hora y valor de la lectura. Además, se pueden filtrar los valores según rangos específicos y descargarlos, como se puede ver en la figura 18.

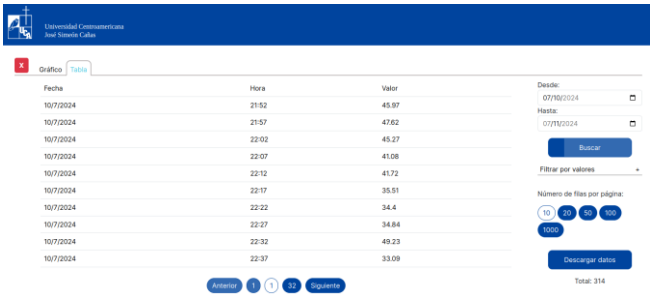

Fig. 18 Tabla con registro histórico para un dispositivo específico.

B. Aplicación Móvil

La aplicación móvil para dispositivos Android complementará la experiencia web. En la pantalla principal se mostrarán cartas expandibles con resúmenes de las lecturas. Cada carta incluye:

- Tipo de medición: por ejemplo, temperatura o humedad.
- Ubicación: dónde se realizó la lectura.
- Valor: el dato específico.
- Unidades: las unidades de medida en las que se realiza la medición.
- Descripción: contexto adicional sobre la medición.
- Intervalo de lectura: intervalo de tiempo entre cada medición.
- Fecha y hora: cuándo se realizó la lectura.
- Lecturas: se utilizará un gráfico de líneas para visualizar las tendencias a lo largo del tiempo (ver figura 19).

Fig. 19 Pantalla de inicio de aplicación móvil.

IV. CONCLUSIONES

El presente estudio ha demostrado el potencial de la red LoRaWAN para la conexión de dispositivos de pequeñas dimensiones a largas distancias y la transmisión de paquetes de datos de pequeñas dimensiones. Este sistema es ideal para la recepción de información de diversos sensores instalados en una área amplia.

Se desarrolló con éxito un sistema para la recolección de datos de estos dispositivos, que permite recopilar todos los datos en un solo lugar, almacenar toda esa información y presentarla al público de una manera que facilite la reportería y la interpretación de los datos detectados por el dispositivo en el lugar donde se instaló. Este sistema requiere conocimientos interdisciplinarios en las áreas de redes inalámbricas, transmisión de datos, electrónica de circuitos y desarrollo web y móvil.

El sistema de recolección de datos se desarrolló con una estructura Rest API + Frontend. La API se encarga de recibir los datos del gateway y almacenarlos en una base de datos relacional, mientras que el frontend se encarga de consumir los datos desde la API y presentarlos en forma de gráficas con datos históricos y tarjetas que muestran la lectura más reciente de cada dispositivo.

Este sistema tiene el potencial de convertirse en un referente para futuros trabajos universitarios que busquen la evolución y explotación de la red LoRa. Además, puede convertirse en una herramienta útil para la lectura, interpretación y mejora continua del estado ambiental de la UCA.

Durante el desarrollo del proyecto se realizaron pruebas con distintas placas de programación. Las placas Arduino Mega y Arduino Mega + ESP8266 generaron problemas de compatibilidad con la librería MCCI LoRaWAN LMIC Arduino. Por lo tanto, se concluyó que la librería no es apta para estas placas y la solución fue utilizar la placa de programación Arduino UNO, que sí es compatible con la librería.

El sistema de recolección de datos desarrollado en el proyecto es muy versátil porque permite la integración de una aplicación móvil con la API para actuar como una versión móvil del frontend que está desplegado en un formato web. Esta aplicación móvil es un prototipo que permite la lectura del valor más reciente de cada sensor en un formato de tarjetas. Este prototipo tiene el potencial de ser mejorado a gran escala para convertirse en una versión móvil del frontend web que contiene gráficas históricas, además de tarjetas, y que permitiría un mayor alcance de la población universitaria.

REFERENCIAS

[1]. SemTech. (s.f.). LoRa PHY. Recuperado el 12 de febrero de 2024, de https://www.semtech.com/lora/what-is-lora

[2]. The Things Network. (2021, 12 diciembre). What are LoRa and LoRaWAN? Recuperado el 17 de marzo de 2024, de https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/

[3]. The Things Network. (s.f.). LoRaWAN Architecture. https://www.thethingsnetwork.org/docs/lorawan/architecture/

[4]. Becolve Digital. (2013, 27 abril). Conceptos técnicos básicos que te ayudarán a entender LoRa y LoRaWAN (Low Power Wide Area Network) en pocos minutos. Recuperado el 27 de febrero de 2024, de https://becolve.com/blog/conceptos-tecnicos-basicos-que-te-ayudaran-a-entender-lora-y-lorawan-low-power-wide-area-network-en-pocos-minutos/

[5]. Garcilazo García, D. A. y Santos Salgado, K. A. (2022, mayo). *Medición y análisis espectral de señales de transceptores LoRa [Trabajo de graduación, Universidad Centroamericana José Simeón Cañas]*. Biblioteca institucional, Universidad Centroamericana José Simeón Cañas.

[6]. Araya Flores, A. J., Escobar Cruz, J. F., García Castillo, E. M. y Quezada Serpas, G. L. (2022, noviembre). *Red LoRaWAN. Modelado y caracterización de los sistemas de transmisión, procesamiento y del entorno de propagación [Trabajo de graduación, Universidad Centroamericana José Simeón Cañas]*. Biblioteca institucional, Universidad Centroamericana José Simeón Cañas.