
**UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN**

**TEMA:** "SISTEMA DE MONITOREO DE FERTIRRIEGO EN PRODUCCIÓN DE ARÁNDANOS EN 'LA DELICIA' – SAN JOSÉ DE CHALTURA UTILIZANDO TECNOLOGÍA IEEE 802.15.4G Y SUN-FSK CAPA FÍSICA"

**TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN**

**AUTOR:** ADRIANA HAYDEÉ HERDOÍZA DÍAZ

**DIRECTOR:** MSC. EDGAR ALBERTO MAYA OLALLA

**Ibarra – Ecuador**
**2023**

**UNIVERSIDAD TÉCNICA DEL NORTE**

**BIBLIOTECA UNIVERSITARIA**

# AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

## IDENTIFICACIÓN DE LA OBRA

En cumplimiento con el Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información.

| **DATOS DEL CONTACTO** | |
|---|---|
| **CÉDULA DE IDENTIDAD** | 070348452-7 |
| **APELLIDOS Y NOMBRES** | Herdoíza Díaz Adriana Haydeé |
| **DIRECCIÓN** | Calle Grijalva y Sánchez y Cifuentes |
| **EMAIL** | ahherdoizad@utn.edu.ec |
| **TELÉFONO MÓVIL** | 0996482363 |
| **DATOS DE LA OBRA** | |
| **TEMA** | "Sistema de monitoreo de Fertirriego en producción de arándanos en "La Delicia"-San José de Chaltura, utilizando Tecnología IEEE 802.15.4g y SUN-FSK capa física. |
| **AUTOR** | Herdoíza Díaz Adriana Haydeé |
| **FECHA** | 02/02/2023 |
| **PROGRAMA** | PREGRADO_**X**_ POSTGRADO___ |
| **TÍTULO** | Ingeniera en Electrónica y Redes de Comunicación |
| **DIRECTOR** | MSc. Edgar Alberto Maya Olalla |

**UNIVERSIDAD TÉCNICA DEL NORTE**

**BIBLIOTECA UNIVERSITARIA**

## CONSTANCIA

El autor manifiesta que la obra objeto de la presente autorización es original y se desarrolló sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 24 días del mes de febrero del 2023

EL AUTOR:

Adriana Haydeé Herdoíza Díaz

C.I.: 070348452-7

**UNIVERSIDAD TÉCNICA DEL NORTE**

**FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS**

## CERTIFICACIÓN

MAGÍSTER EDGAR MAYA, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que, el presente Trabajo de Titulación "SISTEMA DE MONITOREO DE FERTIRRIEGO EN PRODUCCIÓN DE ARÁNDANOS EN 'LA DELICIA' – SAN JOSÉ DE CHALTURA UTILIZANDO TECNOLOGÍA IEEE 802.15.4G Y SUN-FSK CAPA FÍSICA". Ha sido desarrollado por la señorita Herdoíza Díaz Adriana Haydeé, bajo mi supervisión.

Es todo en cuanto puedo certificar en honor de la verdad.

MSc. Edgar Maya

DIRECTOR

**DEDICATORIA**

*Para la mejor familia del mundo, la familia que Dios me regaló;*

*a mi mamá Carmita, a mis hermanos Mafer, Mauri, Diego y a mi Kyrita;*

*por haber sido mis pilares y mi impulso diario para tener fuerzas cuando éstas se terminaban; con mucho cariño les dedico este logro, que no es sólo mío, es su logro también.*

*Los amo infinitamente.*

***Adriana Herdoíza Díaz***

**AGRADECIMIENTOS**

*Primeramente, agradezco a Dios por ser mi pilar y mi fortaleza en todo este proceso.*

*Agradezco a mi madre por el cariño, la paciencia y el apoyo incondicional; gracias por ser padre y madre para nosotros, es por su ejemplo que hoy soy la mujer que soy.*

*A mis hermanos, por haber dejado de lado sus sueños para que yo cumpla los míos, sin su entrega y sus consejos, nada de esto sería posible.*

*Agradezco a Marcelo por su paciencia a lo largo de todo este tiempo, gracias por ser mi compañero incondicional.*

*La carrera me ha regalado grandes amigos con quienes compartí muchos momentos alegres, gracias Mateo y Joha por siempre estar.*

*Finalmente, agradezco a mi tutor del proyecto, por dedicar su tiempo desde el inicio hasta cumplir con la meta, brindarme su conocimiento ha sido un regalo invaluable.*

***Adriana Herdoíza Díaz***

**Tabla de Contenido**

## Tabla de Figuras

## Contenido de Tablas

## RESUMEN

El presente trabajo describe el desarrollo de un sistema de monitoreo de fertirriego en una producción de arándanos basándose en el estándar IEEE 802.15.4g y SUN-FSK capa física, se utilizan módulos OpenMote-B con sus antenas de SubGHz, cada uno de ellos toman distintas acciones, dos nodos recolectan información de sensores y la transmiten hacia otro nodo que actúa como Router de Borde que se encargará de recibir datos y enrutarlos.

Se desarrolla el proyecto bajo la metodología del PMBOK en el que intervienen cinco macroprocesos, el primero de ellos es el Inicio del Proyecto, en esta etapa se realiza el levantamiento de información sobre las necesidades en el fertirriego de arándanos, estas se basan en cómo debe ser el proceso de fertirriego, las cantidades de nutrientes y los valores que deben marcar cada uno de los sensores; se investiga además sobre el estándar IEEE 802.15.4g y protocolo MQTT-SN. El segundo macroproceso es la Planificación donde se tiene el diseño completo de la red con cada uno de los diagramas correspondientes. El siguiente macroproceso es el de Implementación del Sistema, aquí se instalarán los dispositivos en el terreno denominado "La Delicia" en Chaltura y se pondrá en funcionamiento el sistema tomando en cuenta las condiciones de diseño. El cuarto macroproceso son las Pruebas en las cuales se analizará las métricas que intervienen para verificar la calidad de la red y el funcionamiento adecuado de la misma. Finalmente, se tiene el Cierre del Proyecto dónde se realizará la entrega oficial y final del mismo tanto la parte práctica como teórica que permita sustentar los objetivos iniciales.

Los resultados y anexos se presentan al final del documento, mismos que comprueban que el Sistema de Monitoreo de Fertirriego funciona de manera correcta bajo los requerimientos de diseño ya que permiten verificar que las dosificaciones de las sustancias necesarias para el cultivo sean las correctas, la toma de los datos se muestra en tiempo real en Grafana y esto, a su

vez, ayuda para que no se pierda el cultivo de arándanos por falta o exceso de algún tipo de nutriente, esto se verifica con los valores de lectura de sensores que estén dentro del rango necesario por la planta en el fertirriego.

## ABSTRACT

This work describes the development of a fertigation monitoring system in a blueberry production base don the standard IEEE 802.15.4g and SUN-FSK in physical layer, OpenMote-B modules are used with their SubGHz antenas, each of them take different actions, two nodes collect information from sensors and transmit it to another node that acts as Border Router that will which is responsible of receiving data and routing it.

The project is developed under the PMBOK methodology in which five macro-processes are involved, the first of which is the Start of the Project, at this stage information is collected on the needs of blueberry fertigation, these are based on how the fertigation process should be, the amounts of nutrients and the values that each of the sensors should mark; In addition, the IEEE 802.15.4g standard and the MQTT SN protocol are investigated. The second macroprocess is Planning, where you have the complete design of the network with each of the corresponding diagrams. The next macro process is System Implementation, here the devices will be installed on the land called "La Delicia" in Chaltura and the system will be put into operation taking into account the design conditions. The fourth macroprocess is the Tests in which the metrics that intervene to verify the quality of the network and its proper functioning will be analyzed. Finally, there is the Closing of the Project where the official and final delivery of the same will be made, both the practical and theoretical part that allows to support the initial objectives.

The results and annexes are presented at the end of the document, which verify that the Fertigation Monitoring System works correctly under the design requirements, since they allow verifying that the dosages of the substances necessary for the crop are correct, the intake of the data is shown in real time in Grafana and this, in turn, helps so that the blueberry crop is not lost

due to a lack or excess of some type of nutrient, this is verified with the reading values of sensors that are inside the range needed by the plant in fertigation.

# CAPÍTULO I

## ANTECEDENTES

En este capítulo se detalla el trabajo de titulación a realizarse, es decir, se muestra el tema, problema, objetivos, alcance y justificación con el objetivo de reconocer los problemas del fertirriego en el cultivo de Arándanos y así lograr implementar un sistema de monitoreo del mismo.

### 1.1. Tema

"SISTEMA DE MONITOREO DE FERTIRRIEGO EN PRODUCCIÓN DE ARÁNDANOS EN 'LA DELICIA' - SAN JOSÉ DE CHALTURA UTILIZANDO TECNOLOGÍA IEEE 802.15.4G Y SUN-FSK CAPA FÍSICA".

### 1.2. Planteamiento del Problema

De acuerdo con los datos que presenta el GAD de Antonio Ante (Ante, 2019) la Parroquia de San José de Chaltura, es una de las parroquias rurales de Ecuador perteneciente al cantón Antonio Ante de la provincia de Imbabura. La palabra Chaltura significa "Plano Bajo", con esto se hace referencia a que su altura es de 2340 metros sobre el nivel del mar con un clima templado de 16° promedio que favorece a la producción agrícola, frutícola, pecuaria, ganadera y actualmente agroindustrial y turística. Chaltura se encuentra ubicada a 3.5 Km de la ciudad de Atuntaqui. En el barrio "El Carmen" se encuentra ubicado un lote de terreno denominado "La Delicia", este posee aproximadamente 8000 metros, dicho terreno cuenta con una plantación nueva de arándanos con alrededor de 600 plantas con vista a una expansión; dado que las condiciones climáticas en Chaltura son en el día más caliente y por la noche más frío, se adapta perfectamente para la producción de arándanos en cantidades considerables originando así productos con excelente calidad.

Al ser San José de Chaltura una zona en la cual la mayoría de su gente se dedica a la agricultura y, además, una zona rural, el agua con la que se produce el riego viaja mediante acequias hasta llegar hacia cada uno de los terrenos de los moradores, este riego puede producirse ya sea en el día o en la noche, así como en las madrugadas en algunas ocasiones, sin embargo, existen personas que han contaminado el agua de acequia originando así la pérdida de productos. Según un artículo proporcionado por Intagri se menciona que "La fertirrigación es la técnica que ha permitido proporcionar la cantidad de agua y nutrimentos en el momento y lugar que las plantas lo necesitan, esta técnica se asocia con un sistema de riego localizado", en base a todo lo expuesto, no existe un monitoreo que permita identificar la calidad del agua ni la cantidad de sustrato, ya que este no puede ser ni muy alcalino ni muy ácido, por otro lado, la planta debe controlar su estrés en la cantidad de riego que ha absorbido, por eso, su principal problema es buscar la mejor manera de monitorear todo el proceso de fertirriego realizado, desde que llega el agua hasta que las plantas han sido regadas.

Con el fin de solucionar estos problemas que acarrean deficiencias en los cultivos de arándanos, se implementa un sistema monitoreo en el proceso de fertirriego en arándanos, esto consiste en diseñar una red de sensores colocados en lugares estratégicos para monitorear los factores pertinentes en las diferentes etapas, en caso de existir alteraciones en los valores del PH del agua, deberá ser cortado el proceso antes de ingresar a la parte de mezclas, en esta sección se monitorea las cantidades de mezclas de nutrientes para nuevamente monitorear sobre los valores de fertirriego que llegó a la planta después del proceso de riego por goteo; para lograr esto, se utiliza módulos que trabajan con tecnología de comunicación inalámbrica IEEE 802.15.4g (Yen, 2015), esto permite que los datos puedan obtenerse en tiempo real, por lo tanto, se podrá actuar de manera inmediata en las acciones pertinentes para el caso, además, puede cubrir el área

deseada, todo ello con el objetivo de que la información se reciba de manera confiable y a tiempo, logrando así que no mueran los cultivos de arándanos debido a infecciones de agua, exceso de mezclas o estrés de las plantas (Mishra & Kertesz, 2020).

## 1.3. Objetivos de la Investigación

### 1.3.1. Objetivo General

Diseñar un Sistema de Monitoreo de Fertirriego utilizando redes de sensores y tecnología IEEE 802.15.4g como medio de comunicación inalámbrico y SUN-FSK Capa Física para evitar la pérdida del cultivo de arándanos en "La Delicia" – Chaltura.

### 1.3.2. Objetivos Específicos

- Investigar acerca de las necesidades de medición en el proceso de fertirriego, sobre la tecnología IEEE 802.15.4g como medio de comunicación inalámbrico y protocolo MQTT para transmisión de datos.
- Construir una red de sensores de acuerdo a las necesidades para el monitoreo de fertirriego en el cultivo de arándanos.
- Implementar la solución del sistema de monitoreo para control de fertirriego en la producción de arándanos ubicado en "La Delicia" perteneciente a San José de Chaltura.
- Analizar parámetros de calidad de la red para verificar que la eficiencia de la misma cubra las necesidades en el proceso de fertirriego para el cultivo de arándanos.

### 1.4. Alcance

El proyecto de titulación que se presenta requiere un orden detallado del proceso de desarrollo del mismo, por lo tanto, trabajará bajo la metodología de PMBOK, se trata de una metodología que brinda un conjunto de procesos, modelos de administración y criterios que permitirán la dirección correcta en el desarrollo del proyecto. Dentro del PMBOK se definen cuáles son los requisitos que se necesitan para el diseño de hardware y software del sistema a implementar, estos pueden definirse en cinco macroprocesos que permitirán la culminación del mismo; dentro de la guía se encuentra el primer macroproceso es el inicio del proyecto, para luego seguir con la planificación del mismo, la ejecución será el tercer macroproceso, el control y monitorización será el cuarto macroproceso para finalizar con el cierre del mismo verificando la satisfacción de los resultados obtenidos, cada uno de estos procesos se detallan a continuación en base al proyecto indicado (Institute, 2017).

El proyecto se centra en diseñar un sistema de monitoreo de fertirriego que utiliza aguas provenientes de acequia para, de esta manera, evitar la pérdida del cultivo por falta o exceso de nutrientes en el proceso de fertirriego en arándanos.

El primer macroproceso del Pmbok es el Inicio del mismo, en este se tiene la etapa del levantamiento de información sobre las necesidades en el fertirriego en el cultivo de arándanos, estas se basan en cómo debe ser el proceso correcto de fertirriego, es decir, las mezclas de nutrientes y agua que deben ser medidos, en que parte del proceso y los valores adecuados que deberían marcar; una vez que se obtenga toda esta información, se deberá investigar sobre la tecnología que se utilizará como medio de comunicación inalámbrica entre nodos en este caso, se trata de la tecnología IEEE 802.15.4g, se continúa con el estudio del protocolo MQTT-SN, se trata de un protocolo que permite la comunicación máquina a máquina entre dispositivos de IoT

mediante servicios de mensajería basados en publicador/suscriptor (Molinero, 2018) necesario para la transmisión y envío de datos entre los sensores encargados de las mediciones.

Siguiendo con la metodología, en la fase de planificación, se tiene el diseño de la red, para esto, se tendrá dos nodos de sensores, el primer nodo contendrá sensores que toman medidas en el agua que llega de la acequia, es decir, un sensor de PH de agua que tomará sus valores en el reservorio o tanque de almacenamiento, aquí, se podrá observar la alteración de pH para identificar si existe algún tipo de contaminación del agua, si el agua está infectada el proceso de absorción de la misma no puede continuar, todos estos datos pasarán al nodo uno mediante comunicación serial. En el siguiente nodo se establecerán dos sensores más, aquí se toman medidas cuando se haya realizado la mezcla de los sustratos necesarios para el fertirriego, las medidas a tomar por los sensores, será nuevamente de PH de la mezcla que deberá marcar valores que estén dentro de los rangos que puede recibir la planta y el otro sensor es de conductividad eléctrica, aquí también se determina que los valores de la mezcla no sean ni tan alcalinos ni tan ácidos, esto evita un daño en el cultivo de arándanos o en la muerte del mismo, si estos valores se encuentran dentro de los rangos, se procede con el riego al producto, nuevamente los datos pasarán a otro nodo; cada uno de estos nodos utilizará tecnología IEEE 802.15.4g para comunicarse con el Gateway central, y SUN-FSK Capa Física.

Además, se hará uso del protocolo MQTT-SN para transmisión y recepción de datos entre nodos, cuando todos los datos hayan sido obtenidos se procede a utilizar el protocolo MQTT para que los datos puedan estar disponibles en servicios en la nube que, a través de AWS y Grafana esto es posible; al hablar de AWS se trata de Amazon Web Services que es una plataforma en la nube que contiene más de 200 servicios integrales de centros de datos a nivel global (Amazon, 2020) con ello, los datos podrán llegar hacia la persona encargada del cultivo

de arándanos en un dispositivo final mediante una interfaz gráfica en la nube que se identifica como Grafana, este monitoreo lo podrán recibir en su oficina o zona de monitoreo, finalmente, se hará uso de una herramienta Sniffer para verificar el intercambio de paquetes.

Una vez concluido el proceso de planificación o diseño de la red, se inicia la fase de implementación el sistema, este proceso se desarrollará en el terreno denominado “La Delicia” en la Parroquia de Chaltura, Provincia de Imbabura, lo que quiere decir que se pondrá en marcha el funcionamiento del sistema en dicho lugar tomando en cuenta todas las condiciones de diseño y los sitios estratégicos para el monitoreo para que el sistema sea eficiente.

Una vez finalizado el proceso de implementación, se debe realizar pruebas que permitan analizar métricas que intervengan en la calidad de la red, mismas que son, total de paquetes enviados, total de paquetes recibidos, total de paquetes perdidos, potencia de la señal en cada nodo, esto permitirá conocer si el proyecto se ha desarrollado de manera adecuada, si las necesidades han sido cubiertas o si se debe mejorar en algún aspecto, de tal manera, se determinará la eficiencia de la red, además, dentro de las pruebas a realizarse se encuentra la verificación del monitoreo, que estas se muestren en tiempo real, que los valores obtenidos sean correctos y que la plataforma en la nube se encuentre disponible para el agricultor en el momento que lo necesite. El último proceso es el cierre del proyecto, en este se realiza la entrega oficial del mismo, tanto la parte práctica como teórica para sustentar los objetivos iniciales.

### 1.5. Justificación

El sector agrícola de Ecuador, específicamente en la zona Norte, provincia de Imbabura, es muy variado; gracias a las grandes extensiones de tierra, la población busca sacar provecho de este sector, por tal motivo, incursionan en la producción de diferentes tipos de productos con el fin de sustentar a sus familias y en ocasiones sobresalir con productos con calidad de

exportación. Existen organizaciones que apoyan al progreso del cultivo de un producto en específico, sin embargo, también hay quienes empiezan a trabajar de forma individual y, por ende, la inversión recae sobre una sola persona, misma que apuesta todo al desarrollo de su proyecto.

Con la búsqueda de innovar siempre, en Chaltura, algunas personas han comenzado con la producción de Arándanos (Strik & Yarborough, 2005), sin embargo, al ser un producto costoso en el inicio de su desarrollo, al cual no solamente se le debe invertir mucho dinero, sino también tiempo, algunos pueden dar un paso hacia atrás, el miedo a que el agua de riego que se utiliza para la producción esté infectada y pueda dañar una producción de esa magnitud puede hacer que las cosas se paralicen, por otro lado, el producto es muy delicado (Telégrafo, 2021),por dicha razón, la atención en todo el proceso de fertirriego del producto puede resultar un poco tedioso, algunos productores comentan que deberían utilizar sensores para hacer pruebas a mano en cada etapa, otros, simplemente hacerlo al ojo pero, quizá no obtengan los resultados esperados, ya que, si los componentes del fertirriego son alterados podrían morir las plantas y con ello perder una gran inversión.

La calidad de un producto de exportación o de venta local, siempre debe ser el mejor, dicho esto, me ha interesado ayudar a quienes invierten grandes cantidades de dinero en la producción de Arándanos en Chaltura y, este es el caso del terreno denominado “La Delicia”; implementar una red de sensores que permita mantener el monitoreo constante del proceso de fertirriego ayudará de manera exponencial al desarrollo del producto final y el tiempo del productor en el sembrío sería un ahorro total.

Mantener un monitoreo estratégico está vinculado con utilizar la tecnología correcta que se pueda adaptar a las condiciones del lugar de producción, este es el caso de la tecnología IEEE

802.15.4g, con el uso de esta tecnología se abre un proceso investigativo para quien realiza el proyecto, debido a que se debe comprender la manera exacta de cómo funciona para que su implementación en la red pueda brindar los resultados deseados y el proceso de fertirriego permita obtener productos sanos sin peligro de ser infectados o sobrecargados de compuestos que puedan dañar la producción (Pritts, Hancock, Strik , Eames, & Calentano, 2000).

Se espera que, a partir del desarrollo de este proyecto, los datos obtenidos en el análisis de calidad de la red en un escenario como el indicado y con la implementación de la tecnología investigada, la eficiencia y disponibilidad de la red sea tan exitosa que permita su implementación en nuevos escenarios, además, será posible proponer medidas que faciliten desarrollar mejoras continuas en el área de la agricultura en Ecuador y en sectores que precisan controlar de mejor manera sus cultivos.

# CAPÍTULO II

## FUNDAMENTACIÓN TEÓRICA

En el capítulo mencionado anteriormente, se obtiene la recopilación de toda la información bibliográfica que se necesita en el desarrollo del proyecto presente. Dentro de los componentes teóricos se habla sobre el cultivo de arándano, cómo es su nutrición y manejo, además, el punto importante es el proceso de fertirriego utilizado en la producción de arándanos, así como las métricas necesarias en dicho proceso.

Lo siguiente es aprender acerca del Internet de las cosas y su vinculación con las Redes de Sensores Inalámbricas, además, se encuentra información detallada sobre la tecnología a utilizar, sus características y los protocolos que intervienen dentro del proyecto. De esta manera, se puede encontrar toda la información con más relevancia para establecer un criterio óptimo y el estudio esté perfectamente orientado.

### 2.1. Arándano

Los Arándanos son una fruta pequeña popular originaria de América del Norte, estos crecen en arbustos, pueden ser altos o bajos. Este tipo de arándano son un tipo de fruto pequeño y redondo de 0,2 a 0,6 pulgadas de ancho, el color de estos puede variar de azul a morado, además, este fruto no requiere preparación para consumirlas, sin embargo, el cuidado de la planta que los contiene es bastante riguroso, son varios detalles que deben cuidarse para una correcta producción de los mismos.

Desde hace poco tiempo, la producción de arándanos en Ecuador ya es una realidad, este fruto se puede cultivar tanto en la Costa como en la Sierra y en zonas que tengan temperatura más alta en el día y en las noches temperatura más baja, por esta razón, Chaltura es el clima perfecto para ello.

### 2.1.1. Nutrición y manejo hidropónico de arándano

El manejo integral de la nutrición de arándano se basa en 5 fases que se indican a continuación:

- Acondicionamiento del suelo, en esta fase, para el cultivo de arándano es muy importante que su suelo se encuentre en buenas condiciones, es decir, deberá estar tratado con el tipo de sustrato necesario, ya sea en el suelo como tal o en bolsas con implementos específicos para su cultivo.

*Figura 1. Acondicionamiento del suelo.*
*Fuente: (García, 2020).*

- Correcta selección del sustrato, en este punto, es necesario colocar el tipo de sustrato necesario para la cultivación de arándano.

*Figura 2. Selección del sustrato.*
*Fuente: (García, 2020).*

- Estimulación de raíces es el punto tres que se deberá tomar en cuenta.

*Figura 3. Estimulación de raíces.*
*Fuente: (García, 2020).*

- El cuarto punto es la protección de raíces.

*Figura 4. Protección de raíces.*
*Fuente: (García, 2020).*

- Finalmente, se tendrá la nutrición y fertirriego que será la parte más importante en el proyecto planteado.

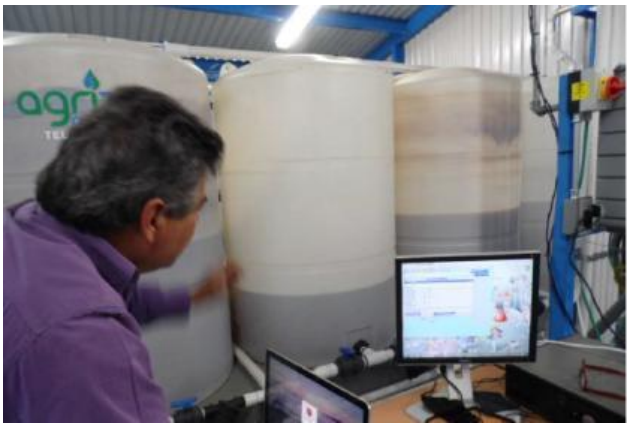


*Figura 5. Nutrición y fertirriego.*
*Fuente: (García, 2020).*

### 2.1.2. Interpretación del análisis químico del agua de riego para Arándano y su manejo

Para el muestreo del agua de riego para arándanos, es muy importante varios detalles que se analizarán, a continuación, se indica una tabla con los parámetros, símbolos y unidades.

*Tabla 1. Parámetros a analizar en el agua de riego.*

| Parámetros | Símbolo | Unidades |
|---|---|---|
| pH | | |
| Conductividad Eléctrica | CE | $dS\ m^{-1}$ |
| Calcio y Magnesio | $Ca^2 + Mg^{2+}$ | $me\ L^{-1}$ |
| Sodio | $Na^+$ | $me\ L^{-1}$ |
| Carbonatos y Bicarbonatos | $CO_3^{2-} + HCO_3 -$ | $me\ L^{-1}$ |
| Cloruros | CI- | $me\ L^{-1}$ |
| Sulfatos | $SO_4^{2-}$ | $me\ L^{-1}$ |
| Relación ajustada de absorción de sodio | $RAS_{aj}$ | |
| Nitratos y Amonio | $N - NO_3 + N - NH_4 mgL^{-1}$ | |
| Boro | B | $mg\ L^{-1}$ |
| Hierro y Manganeso | Fe+Mn | $mg\ L^{-1}$ |
| Partículas sólidas en | | $mg\ L^{-1}$ |

| |
|---|
| suspensión |

*Fuente: Autor*.

Como ya es sabido, el tipo de arándano que se manejará en la producción se encuentra sembrado utilizando sustrato, es por eso que, se analiza la importancia de las mediciones para ph y conductividad eléctrica en las plantas que utilizan sustrato.

En nuestro caso de estudio, los principales parámetros a analizar son el pH y la Conductividad Eléctrica:

- **pH**

Es una medida que indica la acidez del agua (pH bajo = ácido) o alcalino (pH alto = básico o alcalino) del medio. El rango varía de 0 a 14, siendo 7 el rango promedio o también conocido como el valor neutro, si se tiene un valor de pH menor a 7 quiere decir que el agua está ácida, pero, si se tiene un valor superior a 7 se habla de un rango básico, por lo tanto, el pH indica la cantidad relativa de iones de hidrógeno e hidróxido en el agua, a mayor cantidad de iones de hidrógeno mayor será su acidez.

El pH del medio de cultivo controla las reacciones químicas que determinan si los nutrientes van a estar o no disponibles (solubles o insolubles) para lograr su absorción, por esta razón, los problemas nutritivos más comunes se dan cuando el pH se encuentra fuera del rango óptimo.

El rango de pH = 5.5 – 6.5 (indican mayor disponibilidad de nutrientes, usados en sustratos no orgánicos), sin embargo, el rango óptimo de pH en arándano es de pH = 4.5 – 5.5 (sustratos orgánicos). Si el pH del sustrato se encuentra dentro del rango óptimo quiere decir que la mayoría de los nutrientes se encuentran en su máximo nivel de solubilidad, si se encuentran debajo de ese rango se podrían presentar varias deficiencias como nitrógeno, potasio, calcio y

magnesio, por el contrario, si están por encima del rango significaría que disminuye la solubilidad del hierro, fósforo, magnesio, zinc y cobre (Bárbaro, Karlanian, & Mata, 2018).

- **Conductividad Eléctrica (CE)**

La Conductividad Eléctrica son los Sólidos disueltos totales (TDS) que se deben medir en miligramos por unidad de volumen de agua (mg/l) o también se conocen como partes por millón (ppm). Los TDS, por lo tanto, son compuestos inorgánicos que se encuentran en el agua como por ejemplo las sales, metales pesados y algunas trazas de compuestos orgánicos que se disuelven en el agua, existen compuestos que pueden ser beneficiosos para la vida, pero, también existen los que pueden ocasionar daños si se tiene una cantidad mayor de la requerida (CircuitSchools, 2020).

Cabe mencionar que el agua pura (H2O) no conduce electricidad, por lo tanto, si se le coloca un lector de TDS en el agua pura, este debe marcar "0" o un número muy bajo en caso de que exista algo de minerales, entonces, en agua que contenga más minerales, se conducirá más electricidad que en el agua sin minerales. En este caso, el rango óptimo de conductividad eléctrica en el arándano es CE = 1.25 – 1.6 ppm en condiciones ideales, sin embargo, podrían tener valores de hasta 3.5 ppm.

Dicho de otra forma, la concentración de sales solubles presentes en la solución del sustrato se mide mediante la Conductividad Eléctrica, por lo tanto, es la medida de la capacidad de un material para conducir la corriente eléctrica, mientras más alto sea el valor más fácil se mueve la corriente, es decir, a mayor CE será mayor la concentración de sales; si se tiene una CE baja en el sustrato, se facilita el manejo de la fertilización (Bárbaro, Karlanian, & Mata, 2018).

### 2.1.3. Características de un sustrato ideal para Arándano

Una de las interrogantes más escuchadas es el por qué se debe producir arándano en sustrato y la respuesta es la siguiente:

Primeramente, es un cultivo rentable que puede ser producido en zonas con mala calidad del suelo, agua y clima. Su producción se da desde el primer año ddt (producción forzada).

Por otro lado, permite reducir costos en mano de obra, también, facilita la cosecha y aplicaciones. Con este tipo de producción es posible automatizar en su totalidad el cultivo y, finalmente, se reduce el impacto ambiental.

### 2.1.4. Elaboración y manejo de soluciones nutritivas para Arándano

Una solución nutritiva es una mezcla homogénea de uno o más fertilizantes en el agua de riego con una concentración, pH y balance iónico definidos, si tan solo existiese una mala mezcla y/o aplicación de fertilizantes puede generar obstrucción de goteros.

#### 2.1.4.1. Fertirriego en Arándano

El fertirriego es conocido como la combinación de agua y fertilizantes o nutrientes que sirve para nutrir los cultivos, el realizar este tipo de riego trae varias ventajas, una de ellas es la reducción de costos en la aplicación de fertilizantes, por otro lado, ayuda también a la reducción del daño en los cultivos ya que se podrá aplicar todos los nutrientes en cantidades exactas y de manera uniforme en cada una de las plantas.

Para comprender un poco mejor este proceso de fertirriego se presentan las siguientes secciones:

Primeramente, se tendrá la bomba con filtro, esta bomba es la encargada de succionar el agua del reservorio y separarlo de las impurezas y, a su vez, absorber los nutrientes o fertilizantes para ser mezclados con el agua.

*Figura 6. Bomba con filtro.*
*Fuente: Autor.*

La siguiente parte importante dentro del fertirriego son los tanques que contienen los fertilizantes, de estos tanques también son succionados los nutrientes mediante la bomba, mismos que serán mezclados con el agua.

*Figura 7. Tanques contenedores de fertilizantes (nutrientes).*
*Fuente: Autor.*

Finalmente, se tiene el riego de las plantas con todo el proceso realizado anteriormente, de recolección de agua, filtrado de la misma y agregación de nutrientes.

*Figura 8. Riego por goteo en plantación de Arándanos.*
*Fuente: Autor.*

## 2.2. Internet de las cosas

Las siglas que lo representan son IoT, y esto quiere decir que es una red completa que puede conectar hasta millones de dispositivos y personas mediante la tecnología, de esta manera, se crea un entorno inteligente que permita mejoras en cuanto a la salud, energía, transporte, industria, edificios, casas, entre otros (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).

Para lograr todo lo mencionado anteriormente, es necesario contar con sensores que sean capaces de detectar, identificar o ubicar posicionamiento para que de esta manera puedan comunicarse con otros objetos inteligentes y también con seres humanos, alcanzando así áreas que normalmente serían de difícil acceso sin las redes de sensores inteligentes.


*Figura 9. Dispositivos conectados por Internet y la evolución futura.*
*Fuente: (Cisco, 2011).*

La arquitectura de IoT también se compone por varias capas que son sugerencias de la ITU, estará será la arquitectura que deben utilizar los objetos inteligentes para cumplir con sus funciones, las capas que contienen son:

- **Capa del dispositivo:** se encuentran todos los dispositivos tales como sensores, actuadores, etc y también los gateways usados para la recolección de lecturas de los sensores.
- **Capa de red:** esta capa se encarga del transporte y enrutamiento de datos IoT hacia donde estos vayan a ser procesados.
- **Capa soporte:** se trata de una capa intermedia en la cual se esconden las capas inferiores a la capa aplicación para de esta manera poder dar servicios específicos como almacenamiento.
- **Capa aplicación:** son todos los servicios de aplicación que ofrece IoT que utilizan varios tipos de tecnologías y, mediante ellas se pueda brindar el servicio de inteligencia en varios aspectos, como ciudades inteligentes, transporte, energía, entre otros (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).

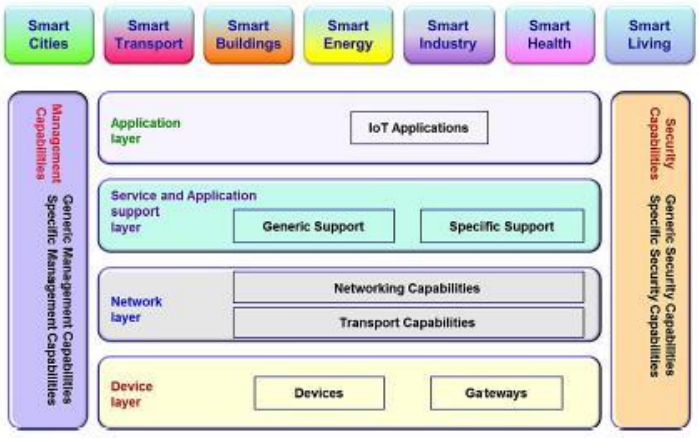



*Figura 10. Arquitectura de Capas de IoT.*
*Fuente: (ITU-T, 2015).*

### 2.3. Redes de Sensores Inalámbricos (WSN)

Por sus siglas en inglés (Wireless Sensor Network), una red de sensores inalámbricos tiene un enfoque de costos bajos que pueden ser utilizados en un medio ambiente determinado; por lo tanto, un WSN es un grupo de dispositivos de sensores que se encuentran distribuidos en un campo y pueden ser utilizados para realizar procesamiento, detección y, además, son capaces de comunicarse entre sí. Por lo tanto, una de las acciones en las que más se utilizan las WSN es en la recopilación de datos, para lo cual existe un nodo central o sumidero, este será el encargado de recibir la información del resto de sensores tal como se indica en la figura siguiente (IbrahiemEmary, 2017).

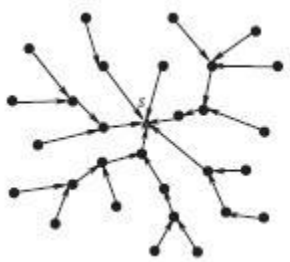



*Figura 11. Recolección de datos en un WSN con un solo sumidero s.*
*Fuente: (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).*

Como ya se ha mencionado anteriormente, los nodos sensores son la característica principal de las WSN y estos consisten básicamente en computadores de pequeños tamaños que manejan funciones básicas, con memoria limitada, entre otras características (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).

#### 2.3.1. Componentes de los nodos sensores

- **Procesador:** se encarga del procesamiento de la información que ha recolectado, tanto de forma local como la que proviene de otros dispositivos. Por lo general, estos procesos tienen dos modos de operación, uno de ellos es el “dormido” que hace referencia al ahorro de energía, también es conocido como “inactivo o idle”,

esto quiere decir que mientras espera datos de otros nodos sensores, este se encuentra en estado inactivo; por otro lado, se tiene al modo “activo” cuando se da el proceso de detectar o enviar datos entre sensores se usará este estado.

- **Fuente de alimentación:** ya que estas motas cumplen con la función de obtener datos hasta en lugares no muy accesibles, se requiere que usen poca energía, las fuentes más comunes de energía son los paneles solares, baterías recargables y condensadores.
- **Memoria:** como ya es sabido, se utiliza para almacenar programas y datos necesarios para el sistema de red.
- **Radio:** un radio inalámbrico de velocidad baja y poca cobertura, va incluido en los dispositivos de la WSN, la velocidad es de hasta 100 kbps y su cobertura de 100 metros.
- **Sensores:** existen diferentes tipos de sensores de acuerdo a las necesidades, por ello, una red de sensores se compone por varios de estos, las tres principales categorías de los sensores de acuerdo con su desarrollo y expansión son físico, químico y biológico (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).

### 2.3.2. Características técnicas de la WSN

- **Conexión inalámbrica**

La comunicación o intercambio de datos entre las motas o nodos sensores se realiza a través de radio, gracias a este tipo de comunicación, la interacción o la obtención de datos se da en tiempo real, logrando así monitorear de manera correcta y tomar acciones de acuerdo a los requerimientos.

Existen dos tipos de red en cuanto a la comunicación inalámbrica:

- Multi-hop: cuando para la comunicación se utilizan otros nodos como relés.
- Single-hop: cuando los nodos establecen una comunicación directa entre ellos o si es que utilizan además un Gateway.

- **Auto-organización**

Los nodos sensores se organizan de manera arbitraria como una red de tipo ad-hoc, dado a su fácil capacidad de organización es más sencillo expandir la red, colocarla en los puntos necesarios, mantenerlos en su sitio y poseen una buena resistencia a fallos de forma individual.

- **Bajo consumo:**

Dado a que se requiere la instalación de las motas en lugares que pueden ser remotos, no existirán fuentes de alimentación, por tal motivo, se necesitan baterías independientes u otro tipo de fuentes de energía como por ejemplo paneles solares, además, cabe mencionar que para que se pueda mantener en funcionamiento las motas durante más tiempo se usan radios y procesadores de bajo consumo (Liñán, Vives, Bagula, Zennaro, & Pietrosemoli, 2015).

### 2.3.3. Aplicaciones WSN

Las aplicaciones de las WSN son bastante extensas, entre ellas se encuentran la agricultura, el monitoreo ambiental, la salud, seguridad, entre otras, algunas de las aplicaciones comunes son:

- Rastreo de movimiento de animales
- Detección de incendios forestales
- Detección de inundaciones
- Investigación geofísica
- Monitoreo detallado de agricultura
- Mantenimiento preventivo en seguridad

- Aplicaciones en el campo de salud

### 2.3.4. Roles en una WSN

De acuerdo a las necesidades que se tengan para el despliegue de una WSN, estas pueden cumplir diferentes roles, entre ellos se tienen los siguientes:

- Nodos sensores: son usados para la detección, recolección y transmisión de lecturas de datos hacia un nodo recolector (sink) o también es conocido como "estación base".
- Nodos recolectores (sink): como se ha mencionado anteriormente, también son conocidos como estación base, estos se encargan de recolectar todos los datos leídos por los sensores de otros nodos para luego ser procesados y analizados.
- Actuadores: estos son utilizados para controlar el ambiente, esto quiere decir que mantienen capacidades de control, como por ejemplo encender una luz o encender un ventilador en caso de temperaturas altas.

## 2.4. Tecnología IEEE 802.15.4

Se trata de un estándar de IEEE para Redes inalámbricas de tasas bajas, es decir, al hablar de IEEE 802.15.4g se habla de un consumo de energía ultra bajo. El IEEE 802.15.4g ha traído el interés de la investigación como un nuevo estándar inalámbrico para bandas de frecuencia de sub-GHz. A este estándar también se lo conoce como Redes Inalámbricas de baja velocidad o LR-WPAN.

Las redes inalámbricas se utilizan para transmitir información a distancias relativamente cortas, las conexiones efectuadas a través de WPAN implican poca o ninguna infraestructura. Esta característica puede implementar soluciones pequeñas, económicas y energéticamente eficientes para una amplia gama de dispositivos como se menciona en (IEEE, 2020).

### 2.4.1. Alcance de IEEE 802.15.4

Este estándar define la capa física (PHY) y las especificaciones de la subcapa de control de acceso al medio o MAC para conectividad inalámbrica de baja velocidad de datos con dispositivos fijos, portátiles y móviles sin batería o requisitos de consumo de batería limitado. Como se menciona en (IEEE, 2020), este estándar proporciona complejidad ultra baja, costo ultra bajo, consumo de energía ultra bajo; la velocidad de datos que este estándar requiere ya satisface a un conjunto de aplicaciones, sin embargo, es posible escalar a las necesidades de los sensores y las necesidades de automatización para comunicaciones inalámbricas.

Las capas físicas se definen para:

- Dispositivos que funcionan en bandas sin licencia de 868-868,6 MHz, 902-928 MHz y 2400-2483,5 MHz.
- Dispositivos con rango de precisión, rango extendido y robustez y movilidad mejoradas.
- Dispositivos que operan de acuerdo con las regulaciones chinas para 314-316 MHz, 430 MHz y las bandas de frecuencia de 779-787 MHz.
- Dispositivos que funcionan para 950 a 956 MHz.

### 2.4.2. Descripciones generales de IEEE 802.15.4

Un LR-WPAN es una red de comunicación simple y bajo costo para conectividad inalámbrica en aplicaciones de potencia limitada y requisitos de rendimiento bajos.

Objetivo principal: facilidad de instalación, transferencia de datos confiable, costo bajo, duración razonable de batería, protocolo de mantenimiento sencillo y flexible.

En IEEE 802.15.4 participan dos tipos de dispositivos diferentes, estos se mencionan a continuación:

- **Dispositivo de función completa (FFD):** estos dispositivos sirven como coordinador o coordinador PAN (Personal Area Network).
- **Dispositivo de función reducida (RFD):** estos no pueden servir ni como coordinador ni como coordinador PAN, sino que, están diseñados para aplicaciones simples como interruptor de luz o sensor de infrarrojos pasivo, no tiene la necesidad de enviar grandes cantidades de datos; cada RFD solamente se asocia con un FFD a la vez.

### 2.4.3. Espacios de aplicaciones especiales

Según (IEEE, 2020), dentro de las principales aplicaciones se tiene las Redes de Servicios Inteligentes (SUN), estos SUN permiten que varias aplicaciones operen sobre recursos de red compartidos para proporcionar monitoreo y control de un sistema.

Los SUN están diseñados para funcionar en sistemas inalámbricos de baja potencia y gran escala. Por otro lado, estos deben cubrir áreas geográficamente extendidas que contienen gran cantidad de dispositivos al aire libre, para esto, los SUN emplean técnicas de múltiples saltos en malla o peer-to-peer para comunicarse con el punto de acceso.

Otras de las aplicaciones son:

- Control y comunicaciones ferroviarias (RCC)
- Espacio en blanco de televisión (TVWS)
- Identificación por radiofrecuencia (RFID)
- Monitoreo de infraestructura crítica de baja energía (LECIM)
- Servicios de red de área corporal médica (MBAN)
- Banda médica de China (CMB)

### 2.4.4. Componentes de IEEE 802.15.4

Dos o más dispositivos que se comunican en el mismo canal físico constituyen una WPAN según lo menciona el estándar de IEEE 802.15.4 (IEEE, 2020). Esta red debe incluir al menos un FFD o dispositivo de función completa que actúa como coordinador de la red.

Para medios inalámbricos no existe un área de cobertura bien definida debido a que las características de propagación son dinámicas e inciertas.

#### 2.4.4.1. Topologías de Red

Dependiendo de los requisitos de la aplicación, un IEEE 802.15.4 opera en dos topologías, una en estrella y otra en peer-to-peer.

- **Topología en Estrella**

La comunicación dentro de esta topología se establece entre los dispositivos y un único controlador central llamado coordinador PAN. Un coordinador puede tener una aplicación específica y, además, ser utilizado para iniciar, finalizar o enrutar la comunicación en la red. El coordinador PAN es el controlador principal del PAN. Existen algunas aplicaciones que se benefician de una topología en estrella como, por ejemplo, la domótica, periféricos de computadoras personales, juegos y cuidados de la salud personal (IEEE, 2020).

Como se muestra en la Figura 12, el nodo de color negro que se encuentra en el centro es el Coordinador PAN, es decir, todos aquellos que sea color negro son Dispositivos de Funciones Completas, mientras que, aquellos que se encuentran de color blanco son los Dispositivos con Funciones Reducidas, finalmente, las flechas indican la comunicación entre el coordinador y el resto de dispositivos.

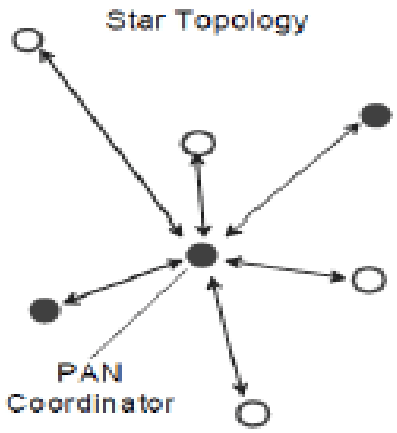



*Figura 12. Topología en Estrella de IEEE 802.15.4.*
*Fuente: (IEEE, 2020).*

**Formación de la Red en Estrella**

Como se indicó en la Figura 12, cuando un FFD está activo, es decir, un Dispositivo con Funciones completas se encuentra activo, puede establecer su red propia y, de la misma manera, se convierte en el coordinador de la PAN; cuando se tienen redes en estrella, cada una de ellas operan de forma independiente de todas las demás redes de estrella que estén funcionando en ese mismo momento.

Para lograr que funcionen por separado, es necesario elegir un ID para dicha PAN que no haya sido utilizada por otra red que esté dentro del rango de comunicaciones, cuando ya se haya elegido este ID, el coordinador de PAN permitirá que otros dispositivos ya sean FFD o RFD se puedan unir a la red, cabe mencionar que se puede utilizar como terminación del coordinador PAN a un dispositivo de función reducida solamente con recepción (RFD-RX) (IEEE, 2020).

- **Topología peer-to-peer**

Este tipo de topología también posee un coordinador PAN, sin embargo, cualquier dispositivo puede comunicarse con cualquier otro dispositivo siempre que estén dentro del alcance del otro. Esta topología permite formar redes más complejas como por ejemplo la topología de redes de malla.

Algunas aplicaciones que se benefician de este tipo de topología son: control y monitoreo industrial, redes de sensores inalámbricos, agricultura inteligente, seguridad.

En la Figura número 13 se puede observar lo mencionado anteriormente, es decir, un coordinador de color negro que son los Dispositivos con Funciones completas que se va a comunicar con todos los dispositivos de funciones completas, se encuentra un dispositivo de funciones reducidas y las flechas indican el sentido de la comunicación que, en este caso se trata de una comunicación en los dos sentidos.

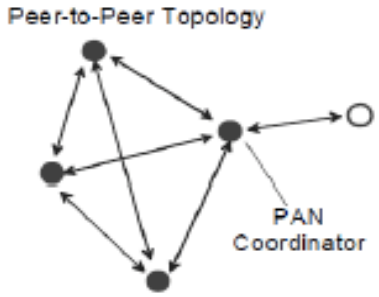



*Figura 13. Topología peer-to-peer de IEEE 802.15.4.*
*Fuente: (IEEE, 2020).*

De manera general, cada PAN independiente selecciona una ID única, misma que permite la comunicación entre dispositivos dentro de una red utilizando direcciones cortas y también puede permitir transmisiones entre dispositivos a través de redes independientes (IEEE, 2020).

**Formación de la Red Peer-to-peer**

Principalmente, se debe tomar en cuenta que en esta topología cada dispositivo puede comunicarse con otro que se encuentre dentro del rango de comunicaciones, luego, se designa como coordinador PAN a un dispositivo, la elección de este puede ser por ser el primero que emita comunicación con el canal, estas topologías son más flexibles y se pueden constituir más estructuras basadas en estas (IEEE, 2020).

Una de las topologías más comunes en cuanto a peer-to-peer es la de árbol de clúster, en esta red la mayoría de los dispositivos son FFD y un RFD se conecta a un clúster de la red de árbol tal cual como si fuera una hoja de una rama ya que los RFD no permiten que otros dispositivos se asocien a la red.

Como se puede observar, en la Figura número 14 se indica que el coordinador del PAN formará parte del primer clúster y elige a un ID de PAN que no se esté utilizando y transmite tramas beacon hacia los dispositivos vecinos y, el dispositivo que reciba la trama beacon realiza una solicitud para lograr unirse a la red mediante el coordinador PAN, si este permite que el dispositivo se una, será agregado como dispositivo secundario en la lista de vecinos.

Posteriormente, el dispositivo que recién se ha unido va a agregar o a reconocer al coordinador como su padre en su lista de vecinos y, entonces, comenzará a transmitir balizas periódicas; si el dispositivo que acaba de unirse no hubiese podido hacerlo, el coordinador PAN busca a otro dispositivo. Las líneas en la figura que se muestra a continuación, indican la relación entre padre e hijo mas no el flujo de comunicación, la principal ventaja del multiclúster es la gran área de cobertura, pero aumenta la latencia tal como lo menciona el estándar IEEE 802.15.4 en su versión actualizada de 2020 (IEEE, 2020).

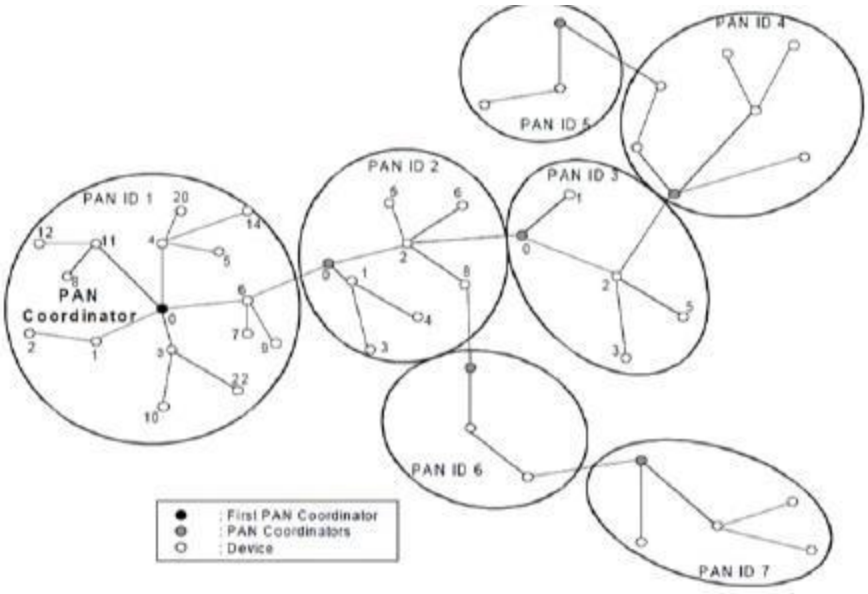



*Figura 14. Red de árbol de clúster.*
*Fuente: (IEEE, 2020).*

#### 2.4.4.2. Arquitectura de Red

La arquitectura de IEEE 802.15.4 se define en términos de varios bloques para simplificar el estándar. A estos bloques se los conoce como capas; cada una de las capas es responsable de una parte del estándar y ofrece servicios a las capas superiores. Las interfaces que se encuentran colocadas entre capas sirven para definir los enlaces lógicos.

En la Figura 15 se puede observar la arquitectura a manera de bloques en la cual se demuestra que, un dispositivo LR-WPAN tiene por lo menos una capa física PHY que contiene al transceptor de radiofrecuencia (RF) y un mecanismo de control de nivel bajo, la capa MAC que gracias a ella se puede obtener el acceso hacia el canal físico; además, se tiene el bloque de capas superiores que hacen referencia a la capa de red que se encarga de configuración, manipulación y enrutamiento de mensajes y una capa de aplicación que realiza ya la función de cada dispositivo.

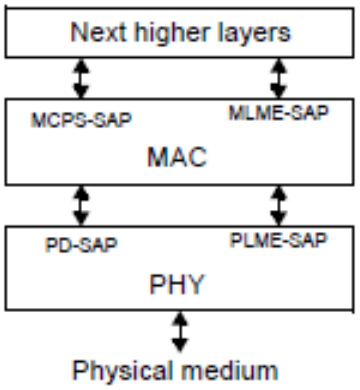



*Figura 15. Arquitectura de dispositivos LR-WPAN.*
*Fuente: (IEEE, 2020).*

**Capa Física (PHY):** las características de PHY son:

- Activación y desactivación de radio
- Detección de energía (ED)
- Enlace de indicación de calidad (LQI)
- Selección del canal
- CCA (Evaluación clara de canal)
- Alcance
- Transmisión y recepción de paquetes a través del medio físico

**Subcapa MAC:** la subcapa MAC proporciona dos servicios:

- Servicio de datos MAC: permite la transmisión y recepción de unidades de datos del protocolo MAC (MPDU) en el servicio de datos PHY
- Gestión de subcapa MAC

Las características de esta subcapa MAC son las siguientes:

- Administración de beacons
- Acceso al canal
- Gestión de intervalo de tiempo garantizado (GTS)

- Validación de tramas
- Asociación y disociación
- Implementación de mecanismos de seguridad mediante enlaces

### 2.4.5. Requerimientos Generales de PHY

Todos los PHY utilizan FCS (control de secuencia de trama) de dos octetos. Las tareas que debe cumplir una PHY son las siguientes según el estándar (IEEE, 2020):

- Activación y desactivación de transceivers de radio
- ED (Detección de energía) dentro del canal actual
- LQI (indicación de calidad del enlace) para paquetes recibidos
- CCA (evaluación clara del canal) para CSMA-CA (detección de portadores de acceso múltiple con prevención de colisiones)
- Selección de frecuencia de canal
- Transmisión y recepción de datos
- Rango de precisión para PHY de UWB (banda ultra ancha)

Cada dispositivo debe iniciar en el modo de PHY que se le indique.

A continuación, se indicará la numeración de canales para las bandas de 868 MHz, 915 MHz y 2450 MHz haciendo referencia a PHY de SUN y TVWS.

La frecuencia central del canal ChanCenterFreq que se utiliza para todos los PHY de SUN y TVWS, excepto los SUN O-QPSK que están operando en las bandas de 676-870 MHz, se derivan de la siguiente manera:

$$ChanCenterFreq = ChanCenterFreq_0 + NumChan * ChanSpacing$$

Donde:

- ChanCenterFreq0 es la frecuencia central del primer canal

- ChanSpacing es la separación entre canales adyacentes
- NumChan es el número de canal de 0 a TotalNumChan - 1
- TotalNumChan es el número total de canales para la banda de frecuencia disponible

Los parámetros ChanSpacing, TotalNumChan y ChanCenterFreq0 para diferentes bandas de frecuencia y los esquemas de modulación se especifican en la Figura que se muestra a continuación; se realizará una tabla para MR-FSK de acuerdo a la frecuencia central y número de canales para cada uno de los modos de operación de FSK.

| Frequency band (MHz) | Modulation | *ChanSpacing* (MHz) | *TotalNumChan* | *ChanCenterFreq*$_0$ (MHz) |
|---|---|---|---|---|
| 169.400–169.475 | MR-FSK operating mode #1 & #2 & #3 | 0.0125 | 6 | 169.40625 |
| 450–470 | MR-FSK operating mode #1 & #2 | 0.0125 | 1599 | 450.00625 |
| 470–510 | MR-FSK operating mode #1 | 0.2 | 199 | 470.2 |
| | MR-FSK operating mode #2 & #3 | 0.4 | 99 | 470.4 |
| | OFDM Option4 | 0.2 | 199 | 470.2 |
| | O-QPSK | 0.4 | 99 | 470.4 |
| 779–787 | MR-FSK operating mode #1 | 0.2 | 39 | 779.2 |
| | MR-FSK operating mode #2 & #3 | 0.4 | 19 | 779.4 |
| | OFDM Option4 | 0.2 | 39 | 779.2 |
| | OFDM Option3 | 0.4 | 19 | 779.4 |
| | OFDM Option2 | 0.8 | 9 | 779.8 |
| | OFDM Option1 | 1.2 | 6 | 780.2 |
| | O-QPSK | 2 | 4 | 780 |
| 863–870 | MR-FSK operating mode #1 | 0.2 | 34 | 863.125 |
| | MR-FSK operating mode #2 & #3 | 0.4 | 17 | 863.225 |

*Figura 16. Número total de canales y frecuencias centrales del primer canal para SUN PHY. Fuente: (Association, 2012).*

### 2.4.6. IEEE 802.15.4g y SUN FSK-PHY

A lo que hace referencia IEEE 802.15.4g es a una mejora del estándar IEEE 802.15.4, esta se basa principalmente a redes inalámbricas de baja velocidad de datos y especializada en

exteriores para cumplir con requisitos de la red de brindar servicios de mediciones inteligentes, esto según (Harada, Mizutani, & Fujiwara, 2017).

Por otro lado, define un PHY alternativo y solo las modificaciones MAC necesarias para apoyar su implementación. IEEE 802,15.4g integra tres clases de PHY: multi-tasa y multi-regional (MR-) FSK, MR-desplazamiento QPSK y MR-OFDM. Las PHY MR-FSK son las más conocidas y comercializadas.

De acuerdo con (IEEE, 2020), un dispositivo SUN debe admitir SUN FSK PHY para permitirle señalización requerida MPM (gestión de PHY múltiple) utilizando CSM (modo de señalización común.

En la imagen siguiente para IEEE 802.15.4g se tienen los parámetros que se aplicarán de acuerdo a la modulación utilizada que será FSK, el modo de operación es el #1 y se utilizará el canal 0, la tabla realizada de frecuencias de acuerdo a los modos de operación y canales, se podrá verificar en los Anexos.

| Frequency band (MHz) | Parameter | Operating mode #1 | Operating mode #2 | Operating mode #3 |
|---|---|---|---|---|
| 863–870 (Europe) | Data rate (kb/s) | 50 | 100 | 200 |
| | Modulation | Filtered 2FSK | Filtered 2FSK | Filtered 4FSK |
| | Modulation index | 1.0 | 1.0 | 0.33 |
| | Channel spacing (kHz) | 200 | 400 | 400 |

*Figura 17. Parámetros relacionados con la frecuencia y la modulación de MR-FSK en IEEE 802.15.4g. Fuente: (Association, 2012).*

### 2.5. Método de Acceso al Medio CSMA-CA

Principalmente, se tiene los siguientes métodos de acceso al medio:

- CSMA-CA sin ranuras utilizando en PAN sin beacon
- CSMA-CA ranurado utilizado en PAN habilitados para beacon
- TSCH CCA utilizado en ranuras no compartidas en un TSCH PAN

- TSCH CSMA-CA utilizado para ranuras compartidas en un TSCH PAN
- CSMA-CA con acceso al canal prioritario (PCA) para eventos críticos
- LECIM ALOHA con PCA

### 2.5.1. Algoritmo CSMA-CA

Acceso múltiple por detección de portadora/evitar colisiones (CSMA-CA) es un protocolo para la transmisión de portadora en redes, fue desarrollado para minimizar el potencial de que ocurra una colisión cuando dos o más estaciones envían sus señales a través de una capa de enlace de datos. Este algoritmo requiere que cada estación verifique primero el estado del medio antes de iniciar una transmisión para evitar así colisiones al escuchar los nodos de transmisión y luego informar a los dispositivos para que transmitan cuando el canal esté libre

Este algoritmo será utilizado antes de la transmisión de datos o comandos MAC transmitidos dentro del CAP, a menos que la trama pueda transmitirse rápidamente después del reconocimiento de un comando de solicitud de dato, el algoritmo CSMA-CA no se utiliza en transmisiones de beacons ni de ACKs.

En CSMA-CA ranurado, los límites del período de retroceso de cada dispositivo PAN, se alinearán con el límite de ranura de la super trama del coordinador PAN; es decir, el inicio del primer período de backoff de cada dispositivo es alineado con el inicio de transmisión del beacon. En CSMA-CA ranurado, la subcapa MAC debe garantizas que el PHY comienza todas sus transmisiones en el límite de un período de interrupción. En CSMA-CA sin ranura, los períodos de interrupción de un dispositivo no están relacionados en el tiempo con los períodos de interrupción de cualquier otro dispositivo PAN (IEEE, 2020).

### 2.6. Protocolo MQTT-SN y MQTT

Existen Redes de Sensores Inalámbricas o también conocidas como WSN, estas son muy conocidas dentro del campo del Internet de las Cosas y son redes formadas por dispositivos con capacidad de procesamiento limitada, limitada energía y memoria (Soni & Makwana, 2017).

Por sus siglas en inglés que son Message Queing Telemetry Transport, y para redes de sensores se conoce como MQTT-SN, es un protocolo de comunicación muy delgado y de alto rendimiento, es distribuido en un entorno industrial de una manera muy amplia, por tal motivo, se está convirtiendo en uno de los estándares más conocidos y utilizados dentro de la industria de IoT.

Se trata de un protocolo bidireccional que permite leer los datos en el entorno y enviar comandos a dispositivos habilitados para MQTT. Un punto importante y muy nombrado dentro de este estándar es el llamado “Broker”, este se encarga de recibir los datos de todos los dispositivos operativos y filtra la información para enviarla solamente a quienes la hayan solicitado (Rilheva, 2019).

De forma general el funcionamiento de MQTT se basa en que, los dispositivos se conectan mediante MQTT y se tendrá quien Publica y quien se Suscribe para enviar y recibir la información que se requiera de acuerdo a los datos que se recibirán.

Message Queue Telemetry Transport (MQTT) cuenta con una arquitectura de publicación/suscripción y se ejecuta sobre TCP, como HTTP; como se ha mencionado anteriormente, MQTT-SN es la versión ligera de MQTT utilizado para redes de sensores, a diferencia de MQTT, MQTT-SN se ejecuta sobre UDP.

Los radioenlaces inalámbricos tuenen una mayor tasa de fallas que los alámbricos debido a su susceptibilidad a perturbaciones de interferencia, además, tiene una tasa de transmisión más

baja, por ejemplo, las WSN basadas en IEEE 802.15.4g tienen un ancho de banda máximo de 250 kbit/s en las bandas de 2,4 GH; además, para que los paquetes sean más resistentes a los errores en la transmisión, los paquetes deben ser muy cortos, así en IEEE 802.15.4g la longitud del paquete en la capa física está limitada a 128 bytes.

### 2.6.1. Message Queue Telemetry Transport for Sensor Networks (MQTT-SN)

El protocolo MQTT es un protocolo asíncrono mediante publish/suscribe que se ejecuta sobre TCP. En todos los mensajes que se manejan en este protocolo interviene un publicador, un suscriptor y un bróker que controla y administra los paquetes intercambiados entre publishers y suscribers, esto se puede observar en la figura a continuación (Martí, García, & Campo, 2019).

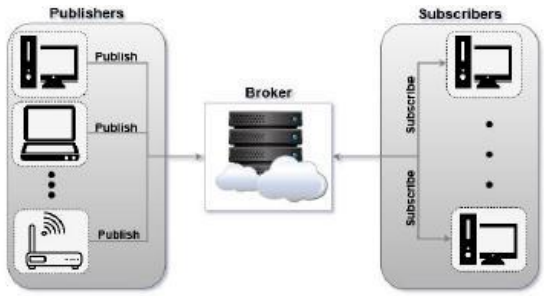



*Figura 18. Patrón de Publish/Suscribe del protocolo de Transporte de Telemetría de cola de Mensajes (MQTT). Fuente: (Martí, García, & Campo, 2019).*

En la imagen siguiente se muestra como un cliente (suscriber) se puede registrar en diferentes tópicos que le interesen en el Broker. Luego, los sensores (publishers) envían al bróker todos los datos que éstos recopilen, luego el bróker los reenvía a los clientes que se han suscrito a los temas relacionados con ese mensaje de publicación.

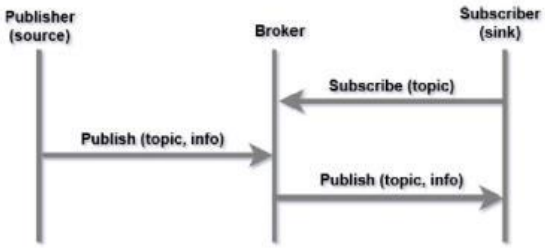



*Figura 19. Mensajes intercambiados por el protocolo MQTT.*

*Fuente: (Martí, García, & Campo, 2019).*

- **Publish/Suscribe**

El Publisher es quien publica mensajes y los usuarios se suscriben a temas, a esta relación se la conoce como modelo de Publicación/Suscripción. Por otro lado, el Suscribe se inscribe a temas de su interés, por lo tanto, todos los mensajes que tengan que ver con dichos temas serán recibidos. Además, los clientes también pueden publicar mensajes a temas para que todos los suscriptores accedan a dichos mensajes.

- **Temas y suscripciones**

Como ya se ha mencionado, el Publisher publica mensajes a temas que serían considerados como los asuntos del mensaje, por lo tanto, un suscriptor se suscribirá a temas específicos para recibir dichos mensajes.

- **Mensajes retenidos**

En este caso se hace referencia a la existencia de un Broker, aquí se retienen los mensajes una vez que estos hayan sido distribuidos a todos los clientes presentes, posteriormente, los mensajes que se retuvieron vuelven a ser enviados al nuevo cliente siempre y cuando estén dentro del tema (Soni & Makwana, 2017).

### 2.6.2. Calidad de Servicio (QoS) de MQTT

Este protocolo de comunicación ofrece tres niveles de calidad de servicio que permiten garantizar la entrega de mensajes publish, estos se detallan a continuación:

- **QoS 0 – Máximo una vez:** este nivel proporciona el menor esfuerzo en cuanto a la entrega de datos, es decir, no espera confirmación de la recepción, por tal motivo, si existe un error en la entrega, este mensaje no será retransmitido. El

receptor no envía ACK, este nivel también es conocido como “activar y olvidar”, es el más simple y contiene gastos generales más bajos.

- **QoS 1 – Al menos una vez:** en este nivel si se retransmite el mensaje hasta que el receptor confirme que ha recibido el mensaje, por lo tanto, garantiza que el mensaje se entregará al destinatario por lo menos una vez; el destinatario podría recibir más de una vez el mensaje.
- **QoS 2 – Exactamente una vez:** este es el nivel de calidad de servicio más alto, garantiza que el mensaje será recibido exactamente una vez por el destinatario, sin embargo, aumenta el consumo de ancho de banda y el tiempo de comunicación; este nivel puede ser utilizado en sistemas de facturación en los que no se permita mensajes duplicados (Rilheva, 2019).

### 2.6.3. Arquitectura de MQTT-SN

MQTT-SN es una versión de MQTT para redes de sensores, por lo tanto, fue diseñado específicamente para trabajar en Redes de Sensores Inalámbricos (WSN). MQTT-SN funciona sobre UDP mientras que MQTT en su estado original funciona sobre TCP y TLS.

Este protocolo fue diseñado para parecerse lo más posible a MQTT, por tal motivo, trabaja con la misma infraestructura que MQTT, la única diferencia en la arquitectura es que MQTT-SN necesita una entidad en su sistema que se conoce como puerta de enlace o Gateway, este se encargará de traducir todos los mensajes MQTT-SN sobre UDP a mensajes MQTT sobre TCP como se muestra en la Figura que se muestra a continuación (Martí, García, & Campo, 2019).

En la actualidad existen varios tipos de Broker que ya cuentan con la funcionalidad del intercambio de mensajes de forma integrada, esto quiere decir que toda la complejidad estará en el lado del Broker/Gateway, el lado del cliente siempre se debe mantener lo más simple posible.

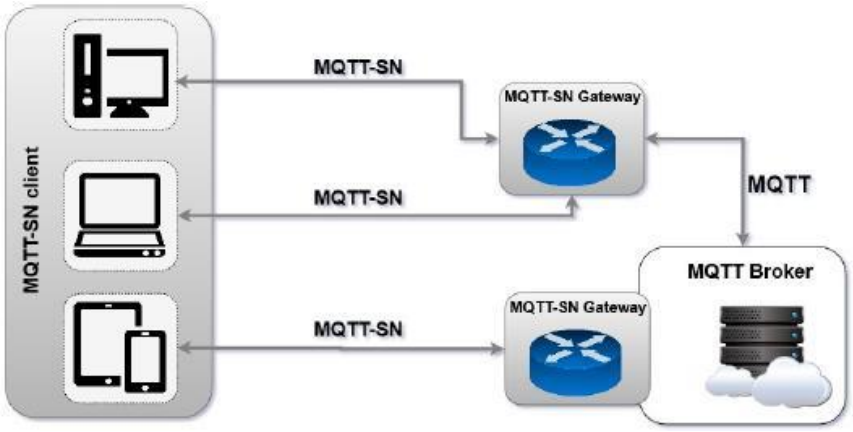



*Figura 20. Arquitectura MQTT-SN.*
*Fuente: (Martí, García, & Campo, 2019).*

En cuanto a funcionalidades, MQTT-SN tiene una ventaja sobre MQTT en un procedimiento de conexión fuera de línea que se define para admitir "clientes durmiendo"; gracias a este procedimiento, los equipos que funcionen con baterías pueden estar en estado de suspensión y, en ese momento, todos los mensajes designados para los mismos, se almacenarán en un server/Gateway para ser entregados cuando los nodos se despierten.

### 2.6.4. Funcionamiento de MQTT-SN e Intercambio de Paquetes

Identifica al protocolo de comunicación basado en el servicio de "publicar (Publish) / Suscribir (Suscribe)". Los publishers o editores, vienen a ser "proveedores, y los suscribers o suscriptores son los conocidos "usuarios".

Para lograr el intercambio de datos se debe agrupar los mensajes de manera lógica, es decir, en categorías de acuerdo a los temas que se manejen. Este intercambio de mensajes es gestionado o manejado por un intermediario, esto quiere decir, un servidor que recibe mensajes

de los publishers y los entrega a los suscribers, cabe indicar que solamente serán entregados los mensajes que hagan referencia a los temas que hayan seleccionado previamente.

Para ser más específicos, sí, por ejemplo, se tiene un sensor de CE, este ocupa el lugar de un Publisher que brindará los datos hacia el bróker sobre el tema específico seleccionado; a continuación, el suscriber se conecta con el bróker y filtra los temas que requiera incluyendo en este caso las de una sonda de temperatura, esto, haciendo referencia a la continuación del ejemplo anterior. Cuando el bróker haya recibido una publicación sobre un tema cualquiera, enviará el mensaje relacionado a todos aquellos suscriptores que solicitaron el tema de temperatura (Rilheva, 2019).

El Broker, o corredor es quien controla la distribución de la información y, además, responsable de recibir todos los mensajes del Publisher, filtrarlos, decidir a quien le interesa para luego enviar los mensajes a todos los clientes suscritos, por lo tanto, sus trabajos son los siguientes: aceptar solicitudes de clientes, recibir los mensaje publicados por los usuarios, procesar diferentes solicitudes como Suscribirse y Desuscribirse, después de recibir mensajes del Publisher los envía a los usuarios a los que les interesa los datos.

MQTT-SN tiene varios tipos de mensajes, en la Figura que se indica a continuación se muestra el intercambio típico de mensajes MQTT-SN. Es importante mencionar que MQTT-SN no admite DTLS (Protocolo de Seguridad de Capa Transporte) debido a la limitación del tamaño de la carga útil de los mensajes (Martí, García, & Campo, 2019).

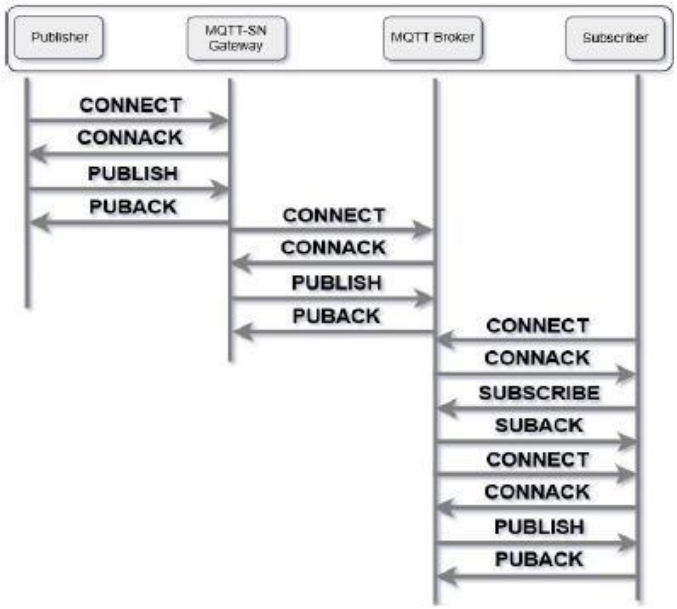



*Figura 21. Intercambio de mensajes con el Protocolo MQTT-SN.*

*Fuente: (Martí, García, & Campo, 2019).*

MQTT-SN está presente en todo el proceso hasta llegar al Broker Mosquitto, sin embargo, la comunicación no termina en el bróker ya que el puente actúa como otro cliente suscrito al mismo tema que se comunica con otro bróker llamado AWS, en el caso del proyecto, el Gateway representa al Router de Borde.

#### 2.6.4.1. Formato del Mensaje

Primeramente, se tiene el formato general del Mensaje MQTT-SN como se muestra en la tabla siguiente. Un mensaje MQTT-SN consta de dos partes: un encabezado largo de 2 o 4 octetos y una variable opcional. El encabezado siempre está presente con el mismo campo, el contenido de la parte variable depende del tipo de mensaje.

*Tabla 2. Formato del Mensaje General.*

| Message Header (2 o 4 octetos) | Message Variable Part (n octetos) |
|---|---|

*Fuente: Autor.*

- **Cabecera del Mensaje**

El formato de la cabecera del mensaje se muestra en la siguiente tabla y está compuesto de dos partes, la Longitud y el Tipo de Mensaje.

*Tabla 3. Cabecera del mensaje.*

| Length (1 o 3 octetos) | MsgType (1 octeto) |
|---|---|

*Fuente: Autor*.

- *Length*

Este campo tiene una longitud de 1 a 3 octetos y especifica el número total de octetos contenidos en el mensaje, incluido el propio campo. El primer octeto del campo Length se codifica como “0x01”, los otros dos indican el número total de octetos del mensaje. Los tres octetos permiten la codificación de longitudes de hasta 65535 octetos. Debido a que MQTT-SN no admite la fragmentación ni el reensamblaje de mensajes, la longitud máxima del mensaje en una red se rige por el tamaño máximo de paquete que admite la red y no por la longitud máxima que podría codificar MQTT-SN.

- *MsgType*

La longitud de este campo es de 1 octeto y especifica el tipo de mensaje, se establecerá en uno de los valores que se muestran en la Tabla 4.

*Tabla 4. Valores del campo MsgType.*

| Valor campo MsgType | MsgType | Valor campo MsgType | MsgType |
|---|---|---|---|
| 0x00 | ADVERTISE | 0x01 | SEARCHGW |
| 0x02 | GWINFO | 0x03 | Reserved |
| 0x04 | CONNECT | 0x05 | CONNACK |
| 0x06 | WILLTOPICREQ | 0x07 | WILLTOPIC |
| 0x08 | WILLMSGREQ | 0x09 | WILLMSG |
| 0x0A | REGISTER | 0x0B | REGACK |
| 0x0C | PUBLISH | 0x0D | PUBACK |
| 0x0F | PUBCOMP | 0x0F | PUBREC |
| 0x10 | PUBREL | 0x11 | Reserved |
| 0x12 | SUSCRIBE | 0x13 | SUBACK |
| 0x14 | UNSUSCRIBE | 0x15 | UNSUBACK |

| | | | |
|---|---|---|---|
| 0x16 | PINGREQ | 0x17 | PINGRESP |
| 0x18 | DISCONNECT | 0x19 | Reserved |
| 0x1A | WILLTOPICUDP | 0x1B | WILLTOPICRESP |
| 0x1C | WILLMSGUDP | 0x1D | WILLMSGRESP |
| 0x1E-0Xfd | Reserved | 0xFE | Encapsulated |
| 0xFF | reserved | | message |

*Fuente: Autor*.

- **Parte Variable del Mensaje**

Este campo va a depender del tipo de mensaje; los campos siguientes están definidos para la parte variable del mensaje.

- *ClientID*

Como en MQTT, este campo tiene una longitud variable y contiene una cadena de 1 a 23 caracteres de longitud que solamente va a identificar al cliente ante el servidor.

- *Data*

El campo Data corresponde a la carga útil de un mensaje MQTT PUBLISH, tiene longitud variable y contiene a los datos de la aplicación que se están publicando.

- *Duration*

Tiene una longitud de 2 octetos y especifica la duración de un período de tiempo en segundos; el valor máximo que puede codificar es de aproximadamente 18 horas.

- Flags

O banderas, consta de 1 octeto y contiene las banderas que se muestran en la Tabla 5.

*Tabla 5. Campo de Banderas.*

| DUP | QoS | Retain | Will | CleanSession | TopicIdType |
|---|---|---|---|---|---|
| (bit 7) | (6, 5) | (4) | (3) | (2) | (1, 0) |

*Fuente: Autor*.

DUP: se establece en “0” si el mensaje se envía por primera vez, en “1” si es retransmitido (dolo dentro de mensajes PUBLISH).

QoS: contiene tres niveles de QoS, para nivel de QoS 0 se establece en “0b01”, para nivel de QoS 1 se establece en “0b10” y para QoS nivel 2 se establece en “0b11”, esto se da solo en mensajes PUBLISH enviados por un cliente.

Retain: solo relevante en mensajes PUBLISH.

Will: si se establece, indica que el cliente está solicitando el tema Will (en mensajes CONNECT).

CleanSession: igual que en MQTT, pero extendido para el tema Will, el mensaje Will solo es relevante dentro de mensajes de CONNECT.

TopicIdType: indica si el campo TopicId o TopicName que se incluye en este mensaje, contiene un ID de tema (0b00), un ID de tema predefinido (0b01) o un nombre de tema corto (0b10).

- *Gw Add*

Contiene una longitud variable y la dirección de un GW. Depende de la red a través de la cual MQTT-SN opera y se indica en el primer octeto de este campo.

- *GwId*

Este campo tiene longitud de 1 octeto e identifica la Puerta de Enlace o Gateway.

- *MsgId*

Longitud de 2 octetos y corresponde al parámetro MQTT “Message ID”. Permite al remitente hacer coincidir un mensaje con su ACK correspondiente.

- *ProtocolId*

Tiene 1 octeto de longitud, está presente solo en el mensaje CONNECT y corresponde al protocolo MQTT “nombre” y “versión”. Está codificado en 0x01.

- *Radius*

Este campo tiene una longitud de 1 octeto e indica el valor del radio de transmisión. El valor 0x00 significa “transmitir a todos los nodos de la red”.

- *ReturnCode*

Un octeto de longitud y su valor y significado se muestran en la siguiente tabla.

*Tabla 6. Valores del ReturnCode.*

| Valor de ReturnCode | Significado |
|---|---|
| 0x00 | Aceptar |
| 0x01 | Rechazado: congestión |
| 0x02 | Rechazado: ID de tema no válido |
| 0x03 | Rechazado: no soportado |
| 0x04 – 0xFF | Reservado |

*Fuente: Autor*.

- *TopicId*

Longitud de 2 octetos y contiene el valor de la identificación del tema. Los valores “0x0000” y “0xFFFF” son reservados y no deben utilizarse.

- *TopicName*

Este campo tiene una longitud variable y contiene una cadena codificada en UTF8 que especifica el nombre del tema.

- *WillMsg*

Longitud variable y contiene el mensaje Will.

- *WillTopic*

Longitud variable y contiene el nombre del tema Will.

- **Formato Individual del Mensaje**

Aquí se indica el formato de los mensajes MQTT-SN de forma individual. Todos los mensajes se describen con el Campo de Longitud de 1 octeto.

- ***CONNECT***

Este mensaje es enviado por un cliente para establecer la conexión y su formato es el que se indica en la tabla siguiente:

*Tabla 7. Mensaje CONNECT.*

| Length (octeto 0) | MsgType (1) | Flags (2) | ProtocolId (3) | Duration (4, 5) | ClientId (6: n) |
|---|---|---|---|---|---|

*Fuente: Autor*.

- Length y MsgType: se encuentran especificados en la Tabla 4.
- Flags:

  DUP, QoS, Retain, TopicIdType: no utilizado.

  Will: si se establece, indica que el cliente solicita un tema y un aviso de mensaje del tema.

  CleanSession: limpiar sesión.
- ProtocolId: Nombre y Versión del protocolo del mensaje MQTT CONNECT.
- Duration: contiene el valor del temporizador Keep Alive.
- ClientId: identificación del cliente, es una cadena de 1 a 23 caracteres de longitud que identifica al cliente ante el servidor.
- ***CONNACK***

El servidor o Broker envía el mensaje de CONNACK en respuesta a una solicitud de conexión de un cliente, su formato se indica a continuación.

*Tabla 8. Mensaje CONNACK.*

| Length (octet 0) | MsgType (1) | ReturnCode (2) |
|---|---|---|

*Fuente: Autor*.

- Length and MsgType: se indican en la Tabla 4.
- ReturnCode: codificada de acuerdo con la Tabla 6.

- ***REGISTER***

Este mensaje es enviado por un cliente a un GW para solicitar el valor del topic id y el topic name. También lo envía un GW para informar a un cliente sobre el valor de identificación del tema que ha asignado al nombre del tema incluido. Su formato es el de la Tabla 9.

*Tabla 9. Mensaje REGISTER.*

| Length (octet 0) | MsgType (1) | TopicId (2, 3) | MsgId (4:5) | TopicName (6:n) |
|---|---|---|---|---|

*Fuente: Autor*.

- Length y MsgType: se encuentran definidos en la Tabla 4.
- TopicId: si es enviado por un cliente, se codifica como 0x0000 y no es relevante; si lo envía un GW, contiene la identificación del valor del topic id asignado al nombre del tema incluido el campo TopicName.
- MsgId: se codifica de tal manera que pueda utilizarse para identificar el mensaje REGACK correspondiente.
- TopicName: contiene el nombre del topic.

- ***REGACK***

Este mensaje es enviado por un cliente o por un GW como acuse de recibo (ACK) y procesamiento de un mensaje de REGISTRO. Su formato se indica a continuación.

*Tabla 10. Mensaje REGACK.*

| Length (octet 0) | MsgType (1) | TopicId (2, 3) | MsgId (4,5) | ReturnCode (6) |
|---|---|---|---|---|

*Fuente: Autor*.

- Length y MsgType: se encuentra especificado en la Tabla 4.
- TopicId: es el valor que utilizará como ID del tema en los mensajes de PUBLISH.
- MsgId: es el mismo valor que el contenido en el mensaje de REGISTER correspondiente.

- ReturnCode: aceptar o rechazar.

- ***PUBLISH***

Este mensaje lo utilizan tanto los clientes como los GW para publicar datos sobre un tema determinado. Su formato se indica a continuación.

*Tabla 11. Mensaje PUBLISH.*

| Length (octet 0) | MsgType (1) | Flags (2) | TopicId (2, 3) | MsgId (5-6) | Data (7:n) |
|---|---|---|---|---|---|

*Fuente: Autor*.

- Length y MsgType: se muestran en la Tabla 4.
- Flags:

  DUP: indica si el mensaje se envía por primera vez o no.

  QoS: contiene el nivel de QoS para este mensaje PUBLISH.

  Retain: contiene la bandera de Retener.

  Will: no se usa.

  CleanSession: no se usa.

  TopicIdType: indica el tipo de ID del tema contenido en el campo TopicId.
- TopicId: contiene el valor de identificación del tema o el nombre corto del tema para el que se publican los datos.
- MsgId: mismo significado que el MQTT "ID" de mensaje; solo relevante en el caso de los niveles QoS 1 y 2, caso contrario se codifica en 0x0000.
- Data: datos publicados.

- ***DISCONNECT***

Un cliente envía un mensaje DISCONNECT para indicar que desea cerrar la conexión. La puerta de enlace acusará recibo de mensaje devolviendo una desconexión (DISCONNECT) al

cliente. Un servidor o un GW, también puede enviar DISCONNECT a un cliente, en caso de que un GW por algún error no pueda asignar un mensaje a un cliente. Al recibir un mensaje DISCONNECT de este tipo, un cliente debe intentar configurar la conexión de nuevo enviando un mensaje de Disconnect al GW. En todos estos casos el mensaje de DISCONNECT no contiene el campo Duración.

Un mensaje de DISCONNECT con un campo de duración es enviado por un cliente cuando quiere pasar al estado "dormido". El GW también envía un ACK a este mensaje mediante un mensaje de DISCONNECT sin un campo de duración. El formato es el siguiente.

*Tabla 12. Mensaje DISCONNECT.*

| Length (octet 0) | MsgType (1) | Duration (opcional) (2-3) |
|---|---|---|

*Fuente: Autor*.

- Length y MsgType: se describen en la Tabla 4.
- Duration: contiene el valor del temporizador para dormir; este campo es opcional y lo incluye un cliente "dormido" que quiere pasar de estado durmiendo a estado dormido.

### 2.7. Componentes requeridos

Dentro del proyecto a realizar es necesario tener en cuenta algunos componentes requeridos, para conocer cuáles son sus funciones principales dentro del proyecto de titulación.

#### 2.7.1. Open Mote B

Open Mote B es un producto nuevo que proviene de su fabricante Industrial Shields quienes cuentan con más de 10 años de experiencia en nodos con protocolos abiertos para IoT.

Se trata de un hardware IoT que puede ser programado por plataformas con código abierto y cumple con el estándar IEEE 802.15.4g, este Open Mote se puede configurar de

acuerdo a las necesidades del usuario, su consumo es bastante bajo y servirá como módulos centrales para recolección de datos de sensores (Shields, 2019).

### 2.7.2. Sensor de pH

Un sensor de pH permite medir la acidez o alcalinidad del agua con un valor entre 0 y 14, si un valor está por encima de siete se vuelve más alcalino mientras que, si está por debajo de siete el agua o el líquido a medir será más ácido, en otras palabras, este sensor ayuda a determinar la calidad del agua, para el proyecto se utilizarán dos sensores de pH, uno antes de agregarle nutrientes y otro después, es decir, ya en el proceso de fertirriego.

### 2.7.3. Sensor Conductividad Eléctrica

Estos sensores miden la Conductividad Eléctrica (EC) en una solución, en este caso se requiere medirlo en el fertirriego, es decir, cuando ya se hayan agregado los nutrimentos al agua de riego, de esta manera se medirán los sólidos disueltos totales (TDS), estos se miden en miligramos por unidad de volumen de agua (mg/l) o también son conocidos como partes por millón (ppm).

### 2.7.4. Arduino UNO

Un Arduino UNO hace referencia a un microcontrolador que es la unidad de procesamiento y tiene código abierto; contiene 14 pines digitales, 6 entradas analógicas, una cabeza de ICSP, botón de reinicio, conexión de alimentación y una entrada USB.

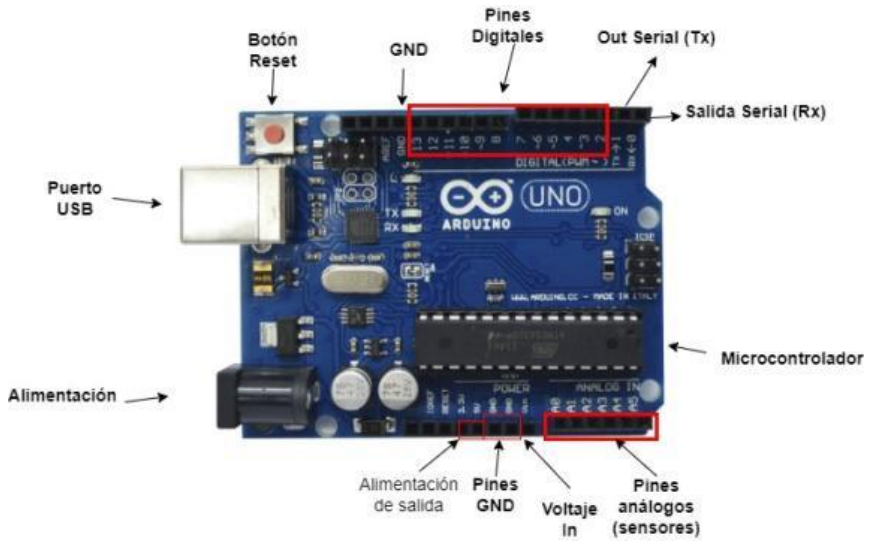



*Figura 22. Pines de Arduino UNO.*

*Fuente: (Arduino, 2023).*

Cada uno de los elementos antes mencionados son los que intervienen en el desarrollo del Proyecto de Titulación para el Monitoreo de Fertirriego en una producción de Arándanos.

# CAPÍTULO III

## DISEÑO E IMPLEMENTACIÓN

El capítulo que se detalla a continuación tiene un propósito específico y es la elección tanto de Hardware como de Software a utilizarse en el desarrollo del proyecto, esto se lo realiza tomando en cuenta el estándar a utilizar que, en este caso es IEEE 802.15.4g y los diferentes protocolos que intervienen, para todo el desarrollo se hace uso de la metodología de PMBOK.

### 3.1. Metodología

Para el correcto desarrollo del proyecto se necesita la guía de una metodología que permita, mediante una serie de procedimientos la construcción del mismo de una manera ordenada, por lo tanto, el modelo a utilizarse es el de PMBOK, ya que se adapta perfectamente a los requerimientos del mismo.

#### 3.1.1. PMBOK

(Project Management Institute, 2017) en su libro denominado “La Guía de PMBOK” hace referencia a que esta metodología brinda un conjunto de procesos, modelos de administración y criterios que permitirán la dirección correcta de un proyecto en general. Dentro del PMBOK se definen cuáles son los requisitos que se necesitan para el diseño de Hardware y Software del sistema a implementar, estos pueden definirse mediante cinco macroprocesos que permitirán la culminación del mismo.

Si bien es cierto, la guía de PMBOK empieza con el inicio del proyecto, en este punto los encargados de dar inicio a nuevos proyectos es en base a respuestas de factores que van a intervenir en el mismo, por ello, existen cuatro categorías sobre estos factores, tales como, cumplir requisitos de carácter regulatorio, legales o sociales: satisfacer las solicitudes o necesidades de las personas interesadas; implementar o cambiar las estrategias de negocio o

tecnológicas; crear, mejorar o reparar productos, procesos o servicios; estas categorías se indican en la Figura 26 (Project Management Institute, 2017).

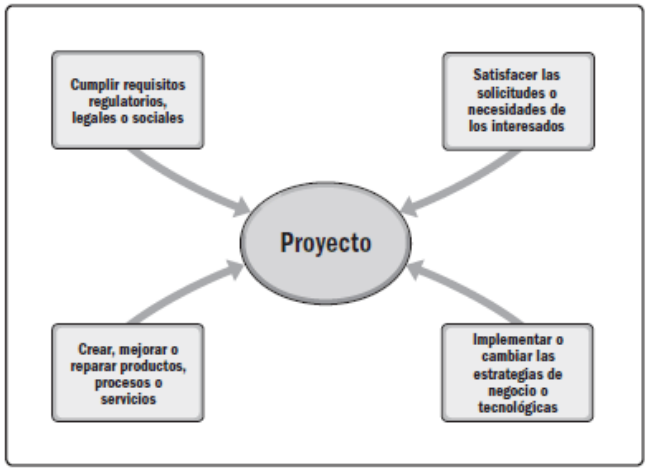



*Figura 23. Contexto de iniciación de un Proyecto.*

*Fuente: (Project Management Institute, 2017).*

#### 3.1.1.1. Etapas de la Gestión de PMBOK

La metodología de PMBOK se basa en cuatro etapas o macroprocesos para la administración del proyecto, en la guía de PMBOK (Project Management Institute, 2017) se indica que dentro de cada fase existen procesos más pequeños importantes en el desarrollo de un proyecto que se basa en PMBOK.

En la Figura 24 se muestra el ciclo de vida del proyecto, misma que tendrá su Inicio, Organización y Preparación, Ejecución del Trabajo y Finalización del Proyecto; de estas cuatro etapas se derivan los Grupos de Procesos necesarios para la correcta ejecución del mismo.

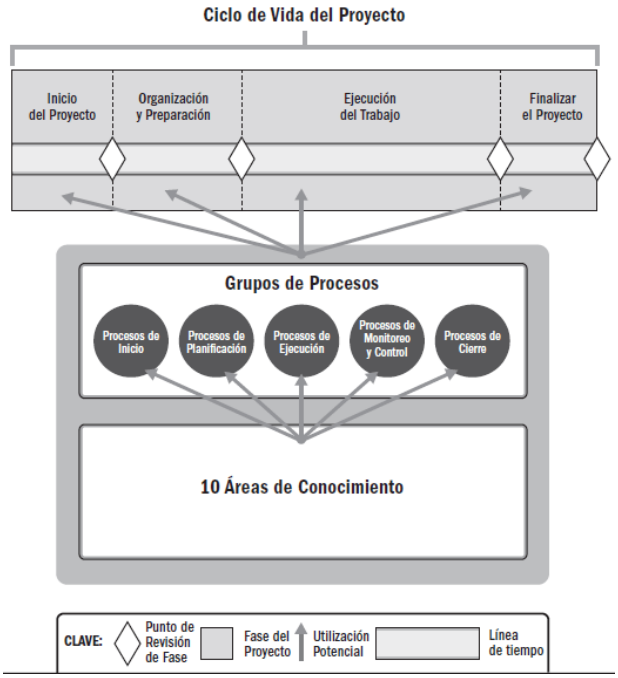



*Figura 24. Interrelación entre los Componentes Clave de los Proyectos de la Guía del PMBOK.*

*Fuente: (Project Management Institute, 2017).*

A continuación, se detallan los cinco macroprocesos en los que se encuentran incluidos cuarenta y siete procesos estándares que intervienen en cualquier proyecto:

- **Inicio**

Se conforma por dos procesos de carácter menor, aquí lo que se busca es la definición de un proyecto nuevo o una nueva fase de ejecución, esto, por lo tanto, conlleva a que posteriormente se reciba una autorización para llevar a cabo el mismo.

- **Planificación**

Dentro de este macroproceso se tienen veinticuatro procesos menores que están orientados a concretar los objetivos del mismo, además, se procederá con el diseño de estrategias para lograr cumplir el proyecto, en este caso, se trata del diseño de la red.

- **Ejecución**

En este proceso se incluirán ocho más que estarán involucrados directamente con el correcto desempeño y desarrollo del proyecto, es decir, de acuerdo a las estrategias planteadas, serán cada una de las actividades que permitan el logro del mismo.

- **Control y Monitorización**

En este macroproceso se incluyen once procesos menores que se involucran a la sección de supervisión y evaluación del desempeño del proyecto.

- **Cierre**

Finalmente, se tiene el macroproceso de cierre, este también cuenta con dos procesos más que se encargan de cerrar totalmente el proyecto haciendo referencia a la satisfacción que brindó los resultados obtenidos del desarrollo del mismo.

### 3.2. Análisis

A continuación, se realizará un análisis que permite el establecimiento de requerimientos necesarios para el desarrollo e implementación del sistema de monitoreo, en este caso, se logrará definir los elementos de Hardware y el Software necesario para este proyecto.

#### 3.2.1. Situación Actual

Actualmente, la agricultura en el Ecuador es uno de los sectores que más ayuda en el comercio y la generación de fuentes de trabajo, la alta demanda de los productos de diferentes zonas ha hecho que los cultivos requieran diferentes factores que prevengan el daño de los mismos.

Según (Gómez, 2020) fruticultora, el arándano es de la familia de los blueberries y es considerada una fruta extraordinaria por sus beneficios en cuanto a la salud humana, es un antioxidante que puede reducir riesgos de contraer cáncer, aumenta defensas, es antiinflamatorio,

regulador de presión arterial, entre otros; por todo lo expuesto, en épocas de pandemia el consumo de arándanos aumentó considerablemente y su demanda ha sido cada vez mayor.

Por otro lado, la revista Inhaus (Ochoa, 2021) conversó con Patricio Ñacato Gerente General de Ecuarándano, esta es una empresa pionera en la producción de dicha fruta en Ecuador, quien cuenta que desde hace cinco años comenzó la producción del arándano por ser un producto con muchos beneficios para la salud y, de esta manera, decidieron convertirse en pioneros, además, comenta que tiene alianza con otros proveedores que le representan el 30% del volumen total de sus ventas, mientras que, el resto proviene directamente de sus campos.

Además, (Ochoa, 2021) comenta que el producto se puede cultivar en Ecuador tanto en la Costa como en la Sierra, sus cultivos se encuentran a 2000 metros sobre el nivel del mar con una temperatura máxima de 35°C y con temperaturas mínimas de 8 grados centígrados, sin embargo, comentan que el primer año fue el más difícil debido al complejo manejo del cultivo, tuvieron que realizar diversas parcelas pilotos en distintos suelos y pisos climáticos para entender el tratamiento del arándano.

En base a los datos indicados anteriormente, se encuentran las necesidades de realizar la implementación de un sistema de monitoreo en los cultivos de arándano, principalmente en aquellos sectores en los que el agua proviene de acequias como lo es Chaltura, gracias a este sistema de monitoreo se puede obtener datos reales en el proceso de fertirriego. Como ya es conocido, las tecnologías hoy en día son una ayuda bastante importante en los cultivos, utilizando ingeniería se puede solucionar varios problemas que afectan a los productores de diversos elementos en el sector agrícola.

Para la realización e implementación del sistema se utilizará un cultivo de arándanos ubicado en la parroquia de Chaltura en un terreno denominado “La Delicia”, los miembros

encargados de llevar la producción han dado las facilidades para poder realizar la implementación del sistema en el tiempo que se requiera, en este caso, se trata de un cultivo de Arándano de aproximadamente un año desde su plantación, actualmente sus productos se entregan a nivel Nacional bajo el nombramiento de productos 100% orgánicos, el bloque de sembrío en el terreno es de aproximadamente 200 metros en los cuales se sitúan 600 plantas de Arándano en dos variedades.

### 3.2.2. Stakeholders

Al hablar de Stakeholders se hace referencia a todas las personas o grupos de personas que puedan influir en el desarrollo del proyecto o se puedan beneficiar del mismo ya sea de manera directa o indirecta, para que un proyecto pueda desarrollarse de la mejor manera siempre es importante tener en cuenta a todos estos individuos involucrados, mismos que se encuentran especificados en la Tabla 13.

*Tabla 13. Funciones de Stakeholders.*

| **Stakeholders** | |
|---|---|
| **Usuarios directos** | UTN |
| **Usuarios indirectos** | Sr. Daniel Rosero |
| | Sra. Claudia López |
| **Tutor** | Msc. Edgar Maya |
| **Asesor** | Msc. Jaime Michilena |
| **Desarrollador** | Srta. Adriana Herdoíza |

*Fuente: Autor*.

Todos los Stakeholders indicados en la Tabla 13 tienen un grado de importancia debido a su intervención y aporte en el desarrollo del sistema.

### 3.3. Requerimientos para el diseño de la WSN

Para el establecimiento de requerimientos se debe manejar el estándar IEEE 29148, este permitirá determinar las necesidades de los stakeholders y así establecer los requisitos que servirán para el diseño y desarrollo del sistema basados en el ciclo de vida del proyecto (ISO/IEC/IEEE29148/2018, 2018).

#### 3.3.1. Nomenclatura de requerimientos

Se procede a indicar las abreviaturas utilizadas en la descripción de los requerimientos que se mostrarán a continuación en la Tabla 14.

*Tabla 14. Abreviatura de Requerimientos.*

| Abreviatura | Requerimiento |
|---|---|
| RSHS | Requerimiento de Stakeholders |
| RS | Requerimientos del Sistema |
| RAS | Requerimientos de Arquitectura |

*Fuente: Autor*.

#### 3.3.2. Requerimientos de Stakeholders

A continuación, se muestra la Tabla 15 que incluye los requerimientos de los Stakeholders identificados anteriormente.

*Tabla 15. Requerimientos Operacionales y de Usuarios.*

| Requerimientos de Stakeholders (RSHS) | | Prioridad | | |
|---|---|---|---|---|
| | | **Alta** | **Media** | **Baja** |
| Requerimientos Operacionales | | | | |
| RSHS 1 | Baterías disponibles para alimentación de OpenMote B | X | | |
| RSHS 2 | Ubicación estratégica de nodos centrales y sensores | X | | |
| RSHS 3 | Los equipos utilizados deben tener accesibilidad a Internet | X | | |

| | | |
|---|---|---|
| RSHS 4 | El envío de datos debe ser inmediato | X |
| RSHS 5 | Nodos sensores que soporten direccionamiento IPv6 | X |
| Requerimientos de Usuarios | | |
| RSHS 6 | Toda la información recolectada debe ser entendible para el usuario | X |
| RSHS 7 | Los datos deben visualizarse en una plataforma | X |

*Fuente: Autor*.

### 3.3.3. Requerimientos del Sistema

Los requerimientos del sistema son derivados de la observación directa de las necesidades del usuario.

*Tabla 16. Requerimientos del Sistema.*

| Requerimientos del Sistema (RS) | | Prioridad | | |
|---|---|---|---|---|
| | | **Alta** | **Media** | **Baja** |
| Requerimientos de Uso | | | | |
| RS 1 | Instalación segura del sistema | | X | |
| RS 2 | Los nodos deben encenderse para poder monitorear | X | | |
| RS 3 | Los nodos sensores deben medir el pH del agua en dos secciones y la CE. | X | | |
| RS 4 | Los códigos de lectura deben estar bien flasheados en los nodos para dar inicio al monitoreo | X | | |
| Requerimientos de Interfaces | | | | |
| RS 5 | La plataforma de visualización debe ser amigable con el usuario | X | | |
| RS 6 | Alimentación del sistema con 3V o USB | X | | |

| | | |
|---|---|---|
| RS 7 | Tanto los OpenMote-B como los sensores deben tener puertos compatibles entre ellos para su conexión | X |
| Requerimientos de Estados | | |
| RS 8 | El dispositivo encargado de la visualización debe estar encendido | X |
| RS 9 | El sistema debe permanecer en estado de ejecución para realizar el monitoreo | X |
| RS 10 | La antena de Sub-GHz debe permanecer siempre conectada | X |
| Requerimientos Físicos | | |
| RS 11 | Estructura adaptable a todo tipo de terrenos | X |
| RS 12 | Ubicación estratégica de sensores y nodos | X |
| RS 13 | Seguridad de elementos para evitar robos o daños | X |
| RS 14 | Tener disponible el cultivo para la implementación del sistema | X |
| RS 15 | Realización de pruebas antes de implementación | X |

*Fuente: Autor.*

### 3.3.4. Requerimientos de Arquitectura

En la Tabla 17 se observan los requerimientos que se deben tomar en cuenta en toda la sección de Arquitectura del Sistema.

*Tabla 17. Requerimientos de Arquitectura.*

| **Requerimientos de Arquitectura (RAS)** | | **Prioridad** | | |
|---|---|---|---|---|
| | | **Alta** | **Media** | **Baja** |
| Requerimientos Lógicos | | | | |
| RAS 1 | Costo del sistema moderado | | X | |

| | | |
|---|---|---|
| RAS 2 | Compatibilidad de todo el Hardware para el funcionamiento del sistema | X |
| RAS 3 | Protocolo MQTT/MQTT-SN | X |
| RAS 4 | Protocolo RPL/UDP | X |
| RAS 5 | IEEE 802.15.4g | X |
| RAS 6 | Soporte IPv6 | X |
| Requerimientos de Hardware | | |
| RAS 7 | El consumo de batería de los nodos debe ser extremadamente pequeño | X |
| RAS 8 | Se debe contar con antena de sub GHz | X |
| RAS 9 | Se debe contar con los sensores para monitoreo y que éstos sean compatibles con los nodos OpenMote-B | X |
| RAS 10 | La capacidad de procesamiento de datos debe ser extremadamente rápida | X |
| RAS 11 | Contar con baterías de repuesto para OpenMote-B | X |
| RAS 12 | Conexión eléctrica para Cliente que va a visualizar los datos obtenidos | X |
| RAS 13 | Armado de todo el circuito para sensores | X |
| RAS 14 | Disponibilidad de puertos USB en nodos para grabar los equipos | X |
| Requerimientos de Software | | |
| RAS 15 | Sistema compatible con Linux | X |
| RAS 16 | Lenguaje de programación Python | X |
| RAS 17 | Soporte de obtención de datos en tiempo real | X |
| RAS 18 | Base de programación gratuito | X |

| | | | |
|---|---|---|---|
| RAS 19 | Plataforma funcional para visualización de datos | X | |
| RAS 20 | Capacidad de memoria extensa para almacenamiento de datos recopilados | X | |
| RAS 21 | Instalación de Sniffer gratuito para análisis de red | | X |
| RAS 22 | Instalación de SO sencilla | X | |

*Fuente: Autor*.

### 3.3.5. Propuesta del sistema

Se ha propuesto crear un sistema de monitoreo de fertirriego mediante la utilización del estándar IEEE802.15.4, el escenario en el cual se desarrollará es un cultivo de arándanos en la parroquia de Chaltura, esta súper fruta cuenta con características que deben ser tomadas en cuenta como primordiales para que la planta pueda vivir, son exactas las mediciones en cuanto al fertirriego, que si existe un exceso en algún elemento, esta planta se estresaría y se pierde, por tal razón, el sistema permitirá este monitoreo constante al obtener los datos de mediciones utilizando sensores y nodos para sensores.

### 3.3.6. Descripción General del sistema

El Sistema de Monitoreo de Fertirriego con aguas provenientes de acequia para la producción de Arándanos utilizará tecnología IEEE 802.15.4g para establecer la comunicación entre nodos con la finalidad de obtener datos en tiempo real, esto permitirá tomar acciones inmediatas para evitar daños en la plantación, gracias al monitoreo en el proceso de fertirriego se podrá regular manualmente los nutrimentos que se involucran en este proceso, por lo tanto, se recibirá los diferentes datos de acuerdo a los sensores utilizados en dicho proceso.

Por otro lado, el monitoreo de la producción de arándano no se basará solamente en la adquisición de datos, sino que, también se realizará el análisis de resultados en cuanto a los datos

totales transmitidos, datos perdidos en el camino, entre otros; la información obtenida brindará datos adecuados de parámetros que permitan el fertirriego de manera correcta en arándanos en Chaltura, dando como resultado la confiabilidad del sistema al usuario.

### 3.4. Elección de Software y Hardware

Esta sección se basa en determinar mediante valoraciones cuáles serían los componentes tanto de Software como en Hardware que mejor se adapten de acuerdo con los requerimientos planteados anteriormente. Los valores serán tomados entre 0 y 1, dónde 0 = no cumple y 1 = si cumple, con dichos valores se realizará una suma total para determinar los dispositivos que serán seleccionados.

#### 3.4.1. Elección de Software

De acuerdo a los requisitos de software que se describieron con anterioridad en los requerimientos de Stakeholders se procede a realizar la elección del mismo, este será el que mejor se adapte a las necesidades del sistema.

- **Sistema Operativo**

Para trabajar con Internet de las Cosas se requiere un Sistema Operativo que sea compatible con los dispositivos o nodos que serán utilizados, para esta elección se tomará en cuenta los requerimientos que se muestran en la Tabla 18, además, la ponderación será de 0 cuando no cumple con el requerimiento y 1 cuando si lo cumple.

*Tabla 18. Comparativa de Sistemas Operativos basados en requerimientos.*

| | **Sistema Operativo** | |
|---|---|---|
| **Requerimientos** | **RIOT-OS** | **Contiki-OS** |
| RAS 4 | 1 | 1 |
| RAS 6 | 1 | 1 |
| RAS 15 | 1 | 1 |

| | | |
|---|---|---|
| RAS 17 | 1 | 0 |
| RAS 17 | 1 | 1 |
| RAS 18 | 1 | 1 |
| RAS 22 | 1 | 0 |
| Valoración | 7 | 5 |

Cumplimiento: 0 – No cumple

1 – Si cumple

*Fuente: Autor*.

Realizando un análisis de la Tabla 18 de acuerdo con los requerimientos planteados con anterioridad, se puede observar que el Sistema Operativo RIOT-OS cuenta con una ponderación mayor a Contiki-OS, esto se debe a que Contiki que basa en un sistema de programación C++, mientras que, RIOT se debe a una programación basada en Python, se había escogido esta programación debido a que es más orientada a estos procesos tecnológicos por su simplicidad y sintaxis, además de ser un lenguaje mucho más amigable. Por otro lado, RIOT-OS es mucho más sencillo de instalar y utilizarlo ya que, simplemente se debe descargar la carpeta de Riot y utilizarla como un directorio común, esto facilita mucho más el proceso de creación y desarrollo del proyecto, en conclusión, RIOT-OS es un sistema Operativo de código abierto que se lo utiliza en dispositivos de Internet de las Cosas ya que cuenta con soporte para una gran cantidad de hardware de bajo consumo energético, redes inalámbricas y cableadas (RIOT-OS, 2018).

### 3.4.2. Elección de Hardware

Haciendo referencia a los requerimientos de Stakeholders en cuanto a la arquitectura del sistema, se determinará la selección de los elementos de hardware que cumplan con las características indicadas.

- **Nodos para Comunicación Inalámbrica**

  En este punto se analizará los dispositivos que cuenten con las características necesarias para establecer los nodos tanto como Gateway del sistema, así como nodos recolectores de datos de sensores, los dispositivos a analizarse son OpenMote-B, Texas Instruments TIIDA-010003 y LaunchXL-CC1352P1.

*Tabla 19. Tabla de características de nodos para comunicación inalámbrica.*

| | **Placa** | | |
|---|---|---|---|
| **Características** | **OpenMote-B** | **TIDA 010003** | **LaunchXL-CC1352P1** |
| Costo | $ 100,75 | $ 40,00 | $ 66,50 |
| Procesador | ARM Cortex-M3 | ARM Cortex-M3 | ARM Cortex-M4F |
| Transceivers | TI CC2538 ATMEL (AT86RF215) | CC1310 | CC1352P1 |
| Antenas/frecuencia | SubGHz (868/915MHz), 2.4GHz | SubGHz 863 MHz, 902MHz, 433 MHz | 868MHz, 915MHz, 2.4 GHz |
| IEEE 802.15.4g | Si posee | Si posee | Si posee |
| Software Libre | Si: Contiki-OS, Linux (RIOT-OS) | Plataforma propia de programación propia, basada en Contiki-NG | Launchpad propio, más complejo de programar |
| RPL | Si acepta | Si acepta | No cuenta con la información |
| 6LoWPAN | Si acepta | Si acepta | Si acepta |
| Memoria | RAM: 32 Kbytes FLASH: 512 Kbytes | RAM: 22 Kbytes FLASH: 128 Kbytes | RAM: 80 Kbytes FLASH: 352 Kbytes |
| Seguridad | Encriptación AES-128/256, SHA2, intercambio seguro de claves ECC-128/256, algoritmos RSA | ECC, AES256, SHA256 | ECC, AES256, SHA256 |
| Puertos | USB, JTAG (cargar y depurar código) | JTAG | USB, JTAG |
| Dimensiones | 45mm x 32mm x 10mm | 70mm x 45mm | 70mm x 45mm |

| | | | |
|---|---|---|---|
| Alimentación | Portabatería 2xAA (3V)<br>USB (5V) | 3.3V | 3.3 V<br>No incluye portabatería |

*Fuente: Autor*.

Como se pudo observar, se realizó la tabla de características principales de los nodos mencionados para la realización del proyecto, se hace notar que el OpenMote-B tiene una mayor capacidad de memoria, tanto la FLASH como la RAM, esto permitirá que mejore el rendimiento en comparación con TIDA y con el CC1352P1; el procesador es el mismo en el OpenMote-B y en TIDA-010003, las características el procesador de CC1352P1 son levemente mejoradas, se observa además que, OpenMote-B contiene dos transceivers de acuerdo a las frecuencias que maneja, tanto para las subGHz como para la de 2.4 GHz, además, estas podrían funcionar simultáneamente si se lo requiriera, un punto muy importante y a favor de OpenMote es l software que maneja, es libre en este caso y tiene varias opciones para su implementación, tomando en cuenta que Riot-OS es basado en Python y mucho más sencillo de programar, a diferencia de los otros dos que tienen su propio Lunchpad para programación, se basan simplemente en Contiki y este SO es un poco más complejo para su programación. Por otro lado, el tamaño de OpenMote es mucho más pequeño, con esto se logra que la implementación en el sembrío sea mucho más compacta, además, el OpenMote cuenta con un portabaterías integrado, esto hace posible el funcionamiento de los nodos sin necesidad de mantenerse conectado a una fuente externa o se puede obviar el adaptar complementos para ello. Dadas las características, se procede a realizar una comparativa de acuerdo a los Stakeholders en la tabla 8, para esta puntuación se dará un "1" si cumple con el requerimiento y un "0" si no lo cumple.

*Tabla 20. Tabla para elección de Nodo.*

| **Módulo inalámbrico** | **RAS6** | **RAS7** | **RAS9** | **RAS10** | **RAS13** | **Total** |
|---|---|---|---|---|---|---|
| OpenMote-B | 1 | 1 | 1 | 1 | 1 | 5 |

| | | | | | | |
|---|---|---|---|---|---|---|
| TIDA 010003 | 1 | 1 | 0 | 0 | 1 | 3 |
| LPCC1352P1 | 1 | 1 | 0 | 0 | 1 | 3 |

*Fuente: Autor*.

De acuerdo con los requisitos planteados y la puntuación con los módulos, se ha escogido el OpenMote-B, este cuenta con características más precisas para la realización del proyecto, a pesar de que su costo es más elevado que el resto con $100 dólares, cumple con funciones más específicas, el almacenamiento de datos es mayor, por lo tanto, su operabilidad será más alta y rápida para procesar los datos pertinentes. En la Figura 25 se puede observar el nodo OpenMote-B con sus respectivas antenas, tanto para Sub-GHz como para 2.4 GHz.

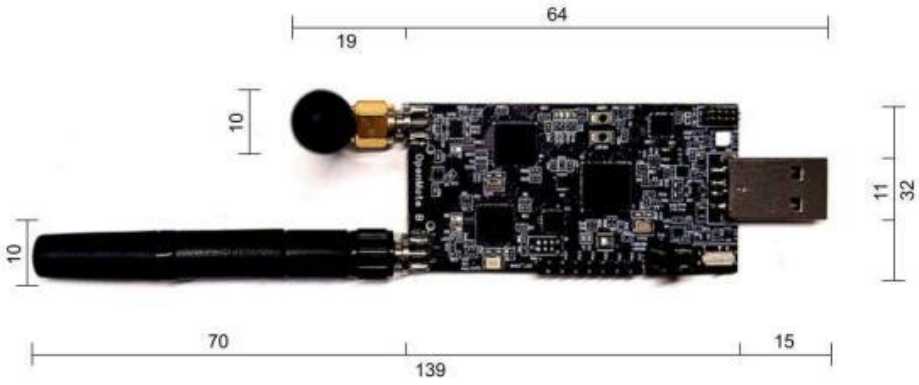



*Figura 25. Nodo OpenMote-B.*

*Fuente: (Shields, 2019).*

Es importante tomar en cuenta que, a pesar de que el nodo OpenMote-B cuenta con todas las características físicas y lógicas para ser seleccionado como placa para la comunicación inalámbrica, se realizó una primera prueba de obtención de datos directamente desde el OpenMote-B, sin embargo, este no permitió la calibración correcta de los sensores debido a los voltajes que maneja, por tal motivo, fue necesario realizar la lectura de datos mediante Arduino UNO con el fin de utilizar sus pines de GND y positivos, así como los puertos de TX y RX.

- **Elección del Sensor de pH**

A continuación, se determinará los sensores de pH que pueden ser utilizados para el presente proyecto, para esto, primeramente, en la tabla 21, se indicarán las características comparativas para los sensores de pH.

*Tabla 21. Características de sensores de pH.*

| | **Sensores** | |
|---|---|---|
| **Características** | **Sensor Precisión 0.1pH + KIT sonda BNC** | **BOOTOP – Sensor pH-4502C** |
| Precio | $ 45,00 | $ 42,00 |
| Voltaje | 5 V | 5 V |
| Rango de lectura | 0-14 | 0-14 |
| Temperatura | 32-140 °F | 32-176 °F |
| Resistencia interna | 250 MΩ | 250 MΩ |
| Tamaño | 42 mm x 32 mm | 42 mm x 32mm x 20mm |
| Salida | Voltaje analógico | Señal voltaje analógico |

*Fuente: Autor*.

Como se ha podido observar en la tabla presentada anteriormente, básicamente tienes las mismas características en el caso de los dos sensores, lo que va a variar es el precio, por tal razón, se escogerá el sensor de pH BOOTOP Ph0-14, mismo que se indicará en la Figura 26, como se mencionó en las características, este viene con su respectiva placa, el sensor y la sonda.

*Figura 26. Sensor de pH-4502C.*

*Fuente: (CircuitSchools, 2020).*

- **Elección del Sensor de Conductividad Eléctrica**

El sensor de Conductividad Eléctrica cumple un papel muy importante ya que determinará los sólidos disueltos totales (TDS) se miden en miligramos por unidad de volumen de agua (mg/l) o también se conocen como partes por millón (ppm) (CircuitSchools, 2020). Se presentan las características de estos sensores en la tabla que se muestra a continuación.

*Tabla 22. Características de Sensores de Conductividad Eléctrica.*

| | **Sensores** | |
|---|---|---|
| **Características** | **TDS V1.0** | **Gravity EC Meter (K=1) V2.0** |
| Precio | $ 30,00 | $ 120,00 |
| Voltaje entrada | 3.3 - 5 V | 3 - 5 V |
| Voltaje salida | 0 – 2.3 V | 0 – 3.4 V |
| Corriente | 3 – 6 mA | 3 – 6 mA |
| Rango medición TDS | 0 – 1000 ppm | 0 – 20 ms/cm |
| Interfaz módulo | PH2.0-3P | PH2.0-3P |

| | | |
|---|---|---|
| Interfaz electrodo | XH2.54-2P | XH2.54-2P |

*Fuente: Autor.*

Dentro de los sensores de Conductividad Eléctrica mencionados en la Tabla 22, se destacan dos características que serán claves para la elección del mismo, la primera es el precio, el TDS V1.0 genérico tiene un precio de $30 dólares ya en el mercado, mientras que, Gravity Meter V2.0 solamente existe en dicha versión de fabricante y tiene un precio de $120 dólares, además, el TDS V1.0 ya cuenta con el rango de medición que se necesita, es decir en ppm que fue el requerimiento de los dueños de la producción de arándanos en Chaltura; por tal motivo, el sensor elegido será el TDS V1.0 en su versión genérica y este se indica en la Figura 27.

*Figura 27. Sensor Conductividad Eléctrica.*

*Fuente: (CircuitSchools, 2020).*

### 3.5. Diseño del Sistema

Luego de haber definido cada uno de los requerimientos de Hardware y Software se continúa con el diseño completo del Sistema, al realizar este diseño se tomará en cuenta todo lo

mencionado en apartados anteriores, éstos permitirán que el diseño sea factible para el Sistema de Monitoreo del Fertirriego en Arándanos.

Esta es la etapa en la cual, mediante distintos diagramas se permite conocer la funcionalidad del sistema de acuerdo con sus diseños por partes que se realizarán.

### 3.5.1. Diagrama de bloques del sistema

A continuación, se presenta el diagrama de bloques del funcionamiento del sistema en sus diferentes etapas en el proceso; primeramente, se cuenta con dos dispositivos OpenMote-B que se los denomina Nodo A y Nodo B, el primero es quien recibe los datos del sensor de PH, mientras que el Nodo B recibirá los datos de los sensores de PH y conductividad Eléctrica del Agua, estos dos, mediante tecnología IEEE 802.15.4g se encargan de enviar las lecturas obtenidas hacia un nuevo OpenMote-B que será el Router de Borde; en este punto, los datos se encuentran con el protocolo MQTT-SN que trabajan sobre UDP, por lo tanto, para que puedan pasar al protocolo MQTT, se necesita un puente transparente que se encargará de realizar este cambio, todos estos se especifican y se realizan dentro de la PC a la cual se conecta el router de borde, el último bloque es cuando se pasan los datos a la nube, para esto, se debe almacenar los datos en la base de datos denominada “Tesis” en DynamoDB en AWS para posteriormente llegar hasta el visualizador de datos que será Grafana, sin embargo, estos no pueden tener conexión directa por lo que se conecta AWS con Panoply y Panoply con Grafana para tener una correcta comunicación y visualización de datos en la Nube.

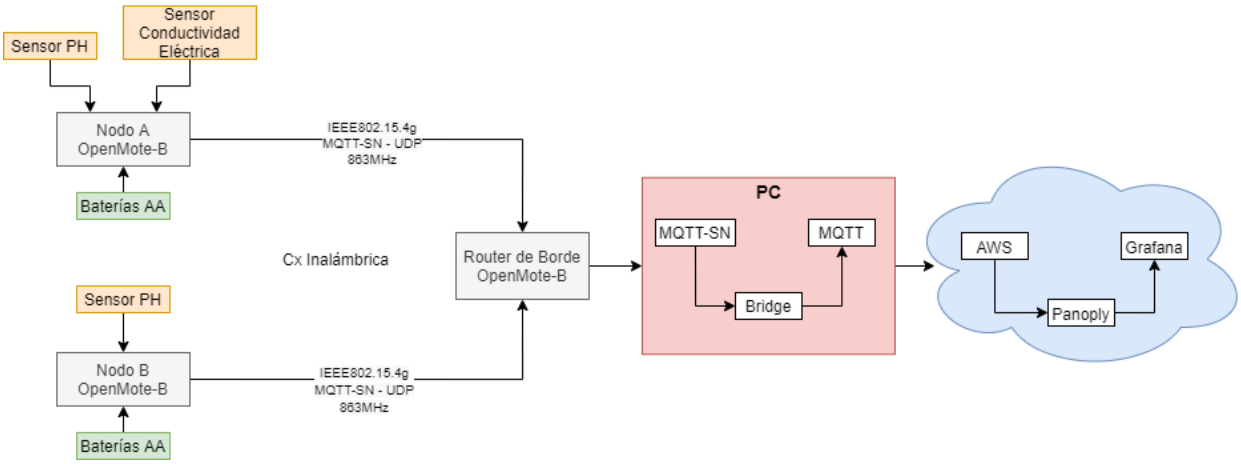



*Figura 28. Diagrama de Bloques del Sistema.*

*Fuente: Autor.*

### 3.5.2. Diagramas de flujo del sistema

En los siguientes diagramas de flujo del sistema para el monitoreo del fertirriego en una plantación de arándanos, se explicará la secuencia del código desde la obtención de los datos mediante los sensores, hasta la visualización de datos en Grafana.

- **Diagrama de Flujo Arduino de Nodo 1**

El primer Diagrama de Flujo hará referencia al código en Arduino que va conectado al sensor de pH, este código se encarga de realizar la calibración del sensor y de armar el mensaje con formato JSON para ser enviado por serial hacia el OpenMote 1 o Nodo 1 para luego seguir con el proceso. Una vez que los datos se envían por serial hacia el Nodo, existe un delay para que el proceso vuelva a repetirse desde la lectura del pin analógico que es quien recibe los datos.

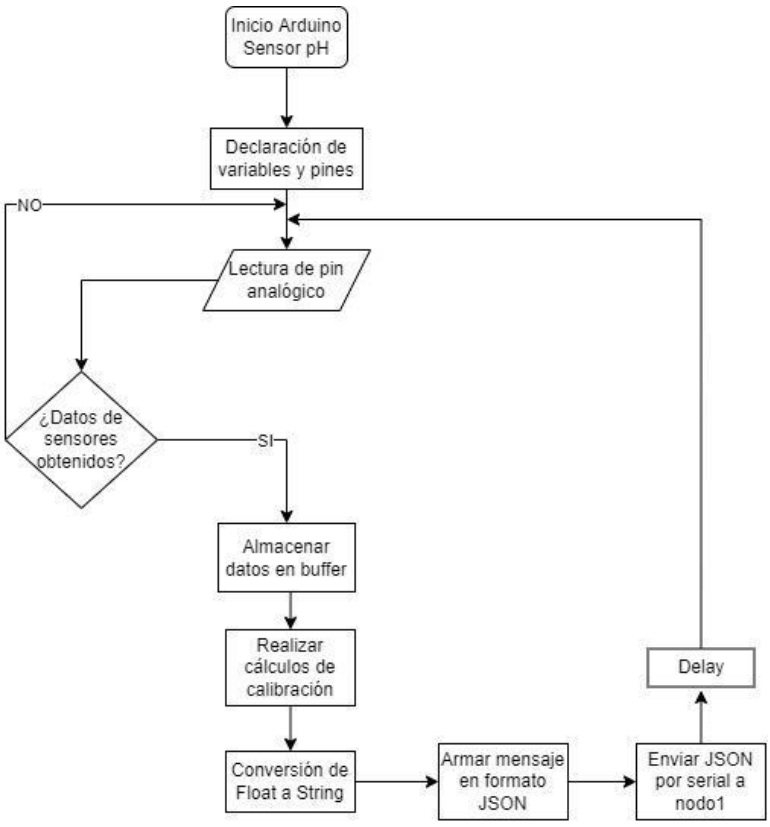



*Figura 29. Diagrama de Flujo Arduino de Nodo 1.*

*Fuente: Autor.*

- **Diagrama de Flujo Arduino de Nodo 2**

En la siguiente figura se muestra el diagrama de flujo del Arduino 2, este Arduino se conecta directamente hacia dos sensores, uno es de pH y el otro es de Conductividad eléctrica, además, también se conecta hacia el Nodo 2 u OpenMote-B. Este Arduino toma los datos por separado de los sensores y realiza un proceso de calibración de cada sensor, además, una conversión de float a String para luego tomar los datos finales de los dos sensores y formar el mensaje con formato JSON, mismo que por comunicación serial será enviado hacia el Nodo 2; finalmente, se tiene un delay y los procesos vuelven a repetirse desde la lectura de datos de los pines analógicos A0 para leer el sensor de pH y A1 para leer el sensor de CE.

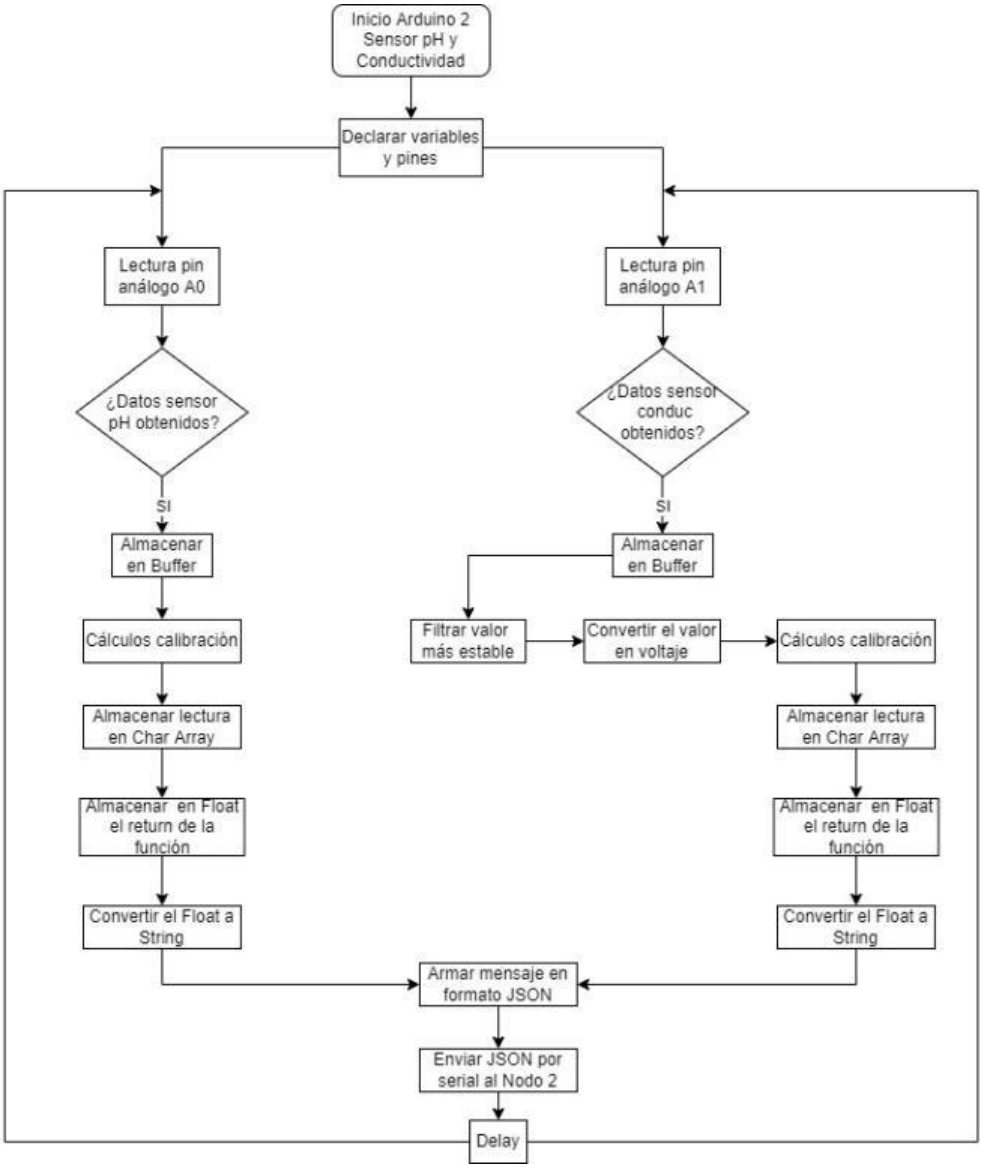



*Figura 30. Diagrama de Flujo Arduino de Nodo 2.*

*Fuente: Autor.*

- **Diagrama de Flujo Router de Borde**

El diagrama de flujo del Router de Borde se muestra a continuación, en este se observa el inicio del mismo, luego se dan las Configuraciones iniciales automáticas como lo son canal, separación de canal, data rate, modulación, direccionamiento, entre otros, cabe mencionar que el Router de Borde tiene la capacidad de direccionar los paquetes; enseguida entrará en un estado de espera para recibir paquetes, se realiza la pregunta de si ha recibido paquetes MQTT-SN y si

es así, reenvía dichos paquetes hacia el Broker mosquitto, caso contrario nuevamente espera más paquetes y el proceso continúa.

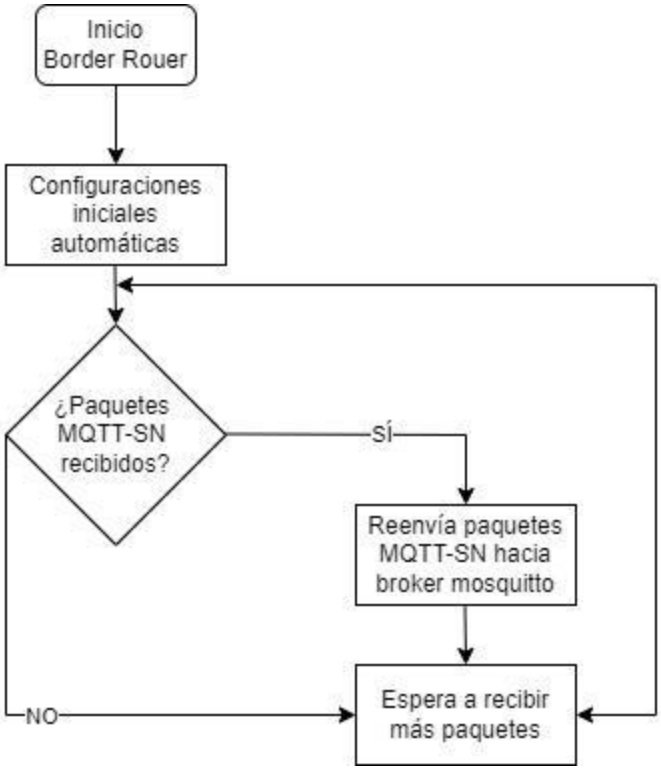



*Figura 31. Diagrama de Flujo Router de Borde.*

*Fuente: Autor.*

- **Diagrama de Flujo OpenMote-B Nodo 1**

Como ya se ha mencionado anteriormente, el Nodo 1 es aquel que se conecta directamente con el Arduino de Nodo 1 que contiene datos de un solo sensor de pH, el proceso se detalla a continuación.

Se da inicio al Nodo sensor, se incluyen todas las librerías necesarias para cada uno de los procesos que desarrollará este Nodo 1, además, se debe iniciar la comunicación por los puertos UART que son los de Transmisión y Recepción, en este caso se habla de los puertos PB2 y PB3 del OpenMote-B, se deben definir otras variables y habilitar el puerto 1883 para el envío de paquetes MQTT-SN; se tiene además varias configuraciones automáticas como lo es el inicio del

nodo, el protocolo RPL, el tipo de modulación, canal y más detalles de los mismos, cabe mencionar que todos los OpenMote deben contar con las mismas configuraciones para que se puedan comunicar correctamente.

Simultáneamente se inician dos hilos, uno de ellos es el Serial, aquí se encuentra una función que permite la lectura por Serial, luego se hace la pregunta de si existen datos por serial, si la respuesta es NO, entonces nuevamente vuelve a escuchar por serial, si la respuesta es SI, entonces almacena en un búfer los datos que lee por serial, se sigue almacenando los datos hasta que el búfer se llena y el contador termina, luego de eso se copian los datos a un json global y el proceso se repite desde la lectura por serial.

Por otro lado, se da inicio a una función de envío automático, este contendrá todos los pasos para conectarse hacia la puerta de enlace mediante la dirección IP que es la 2001:db8::1 y con el puerto 1885 que es el puerto por el cual el Broker Mosquitto está escuchando para que le lleguen los paquetes MQTT-SN, luego se toman los datos almacenados en el json global y se procede con la publicación de los datos en el tema, lo siguiente es registrarse en el topic, publicar los datos y prender leds para indicar el proceso de envío y publicación, luego se da el paso de Desconexión del broker, se espera con un delay de 15 segundos y el proceso vuelve a repetirse desde la toma de datos del json global.

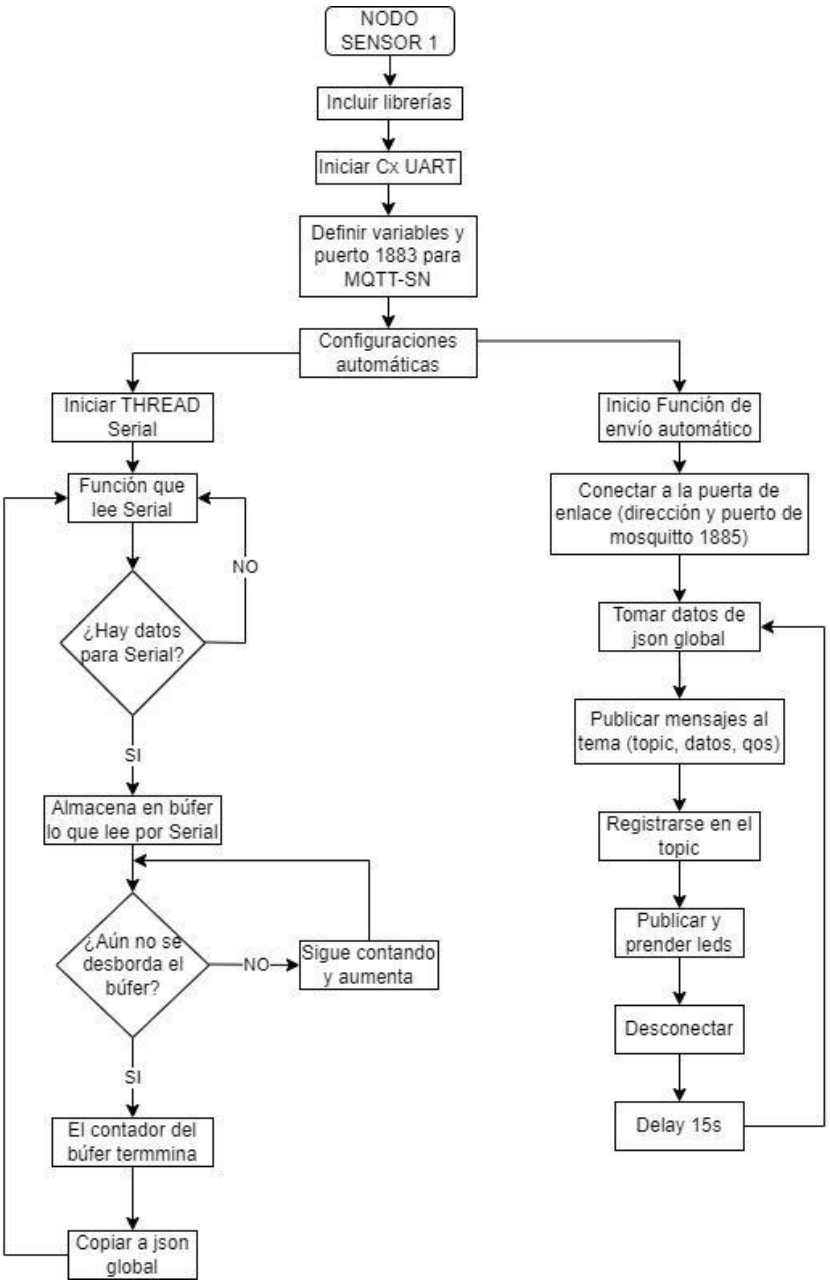



*Figura 32. Diagrama de flujo OpenMote-B Nodo 1*

*Fuente: Autor*.

- **Diagrama de Flujo OpenMote-B Nodo 2**

Finalmente, se tiene el Diagrama de Flujo del Nodo 2, es decir, del OpenMote-B 2, este nodo se conecta con el Arduino del Nodo 2 que contiene a dos sensores, un sensor de pH y luego un sensor de Conductividad Eléctrica; este proceso se exactamente igual al del Nodo 1,

solamente cambiará en los identificadores del ID, este es ID 2 y su nombre de identificación es SENSOR 2, el resto de código procede de manera igual que el anterior.

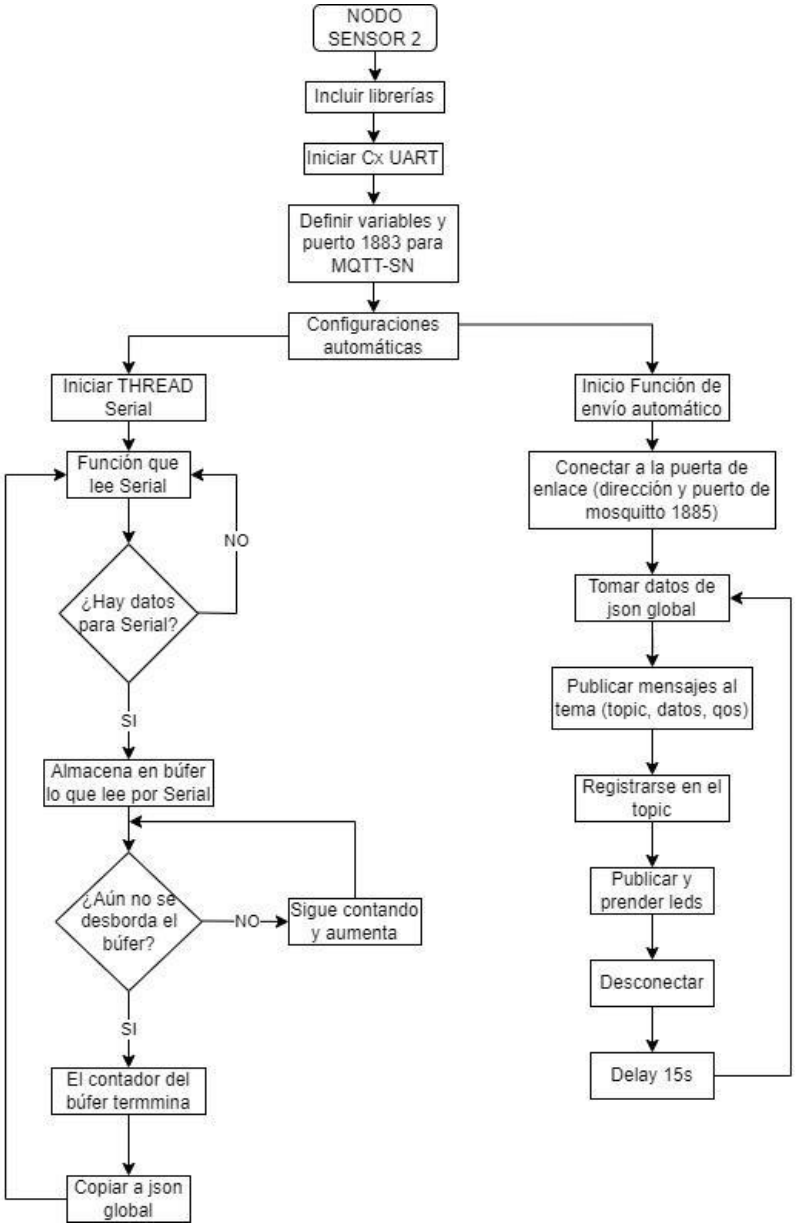



*Figura 33. Diagrama de Flujo OpenMote-B Nodo 2.*

*Fuente: Autor.*

### 3.5.3. Diagrama de Arquitectura del sistema

A continuación, se procede con la explicación de la arquitectura planteada para el proyecto, esta arquitectura es analizada de izquierda a derecha, por lo tanto, se tiene el primer

Stack de sensores que consta de un sensor de PH del agua, esto sirve para verificar la calidad con la que llega el agua de riego a través de la acequia en un pozo contenedor de la misma; el segundo stack de sensores contiene un nuevo sensor de PH que en este caso medirá la mezcla de agua con los nutrientes y un sensor de Conductividad eléctrica que medirá los sólidos disueltos en el agua. El primer stack de sensores nombrado anteriormente enviará sus datos mediante comunicación serial hacia un nodo, en este caso se utiliza OpenMote-B (nodo 1), estos equipos hacen uso de Puertos USB o baterías de 3V, además, cuentan con varios pines configurables necesarios para la comunicación con los sensores, también contienen un microcontrolador y se utilizará la antena de Sub-GHz que será de 863MHz, lo mismo sucede con el segundo stack de sensores que se comunican con un segundo OpenMote-B (nodo 2).

Una vez armada y configurada esta sección del Hardware, se procede a utilizar la Tecnología IEEE802.15.4g para enviar los datos que obtienen el OpenMote-B 1 y OpenMote-B 2 hacia un Gateway o Router de Borde sobre el cual trabajará Mosquitto MQTT-SN Broker, el protocolo utilizado para esta comunicación es MQTT-SN sobre UDP, los datos que han sido enviados cumplen una función en la cual se deben enviar con un ID y hacia un puerto específico que será el 1885, cuando los datos ya se encuentran en el Gateway o nodo central se necesita realizar una comunicación con el MQTT Broker, para esto se hace uso del protocolo MQTT, sin embargo, al haber utilizado MQTT-SN para la recolección de los datos, se debe utilizar un puente transparente para convertirlo a MQTT, este puente transparente tendrá la función de suscribirse a un tema (sensor/station) y se colocará el número de ID con el que se enviaron los datos, de esta manera los recolectará de forma correcta.

A partir de esta sección en la que el puente tiene los datos, estos serán enviados a los servicios de AWS, para esto se tendrá un Cliente MQTT como Suscriptor al bróker, los datos

obtenidos se enviarán a una base de datos dentro de una tabla denominada “Tesis” de DynamoDB que trabaja en AWS y, posteriormente se enviará la visualización de los mismos hacia Grafana pero, al no tener comunicación directa DynamoDB con Grafana, habrá un intermediario que es Panoply, teniendo todos los datos completos se colocará un sniffer para verificar y analizar los parámetros de la calidad de la red.

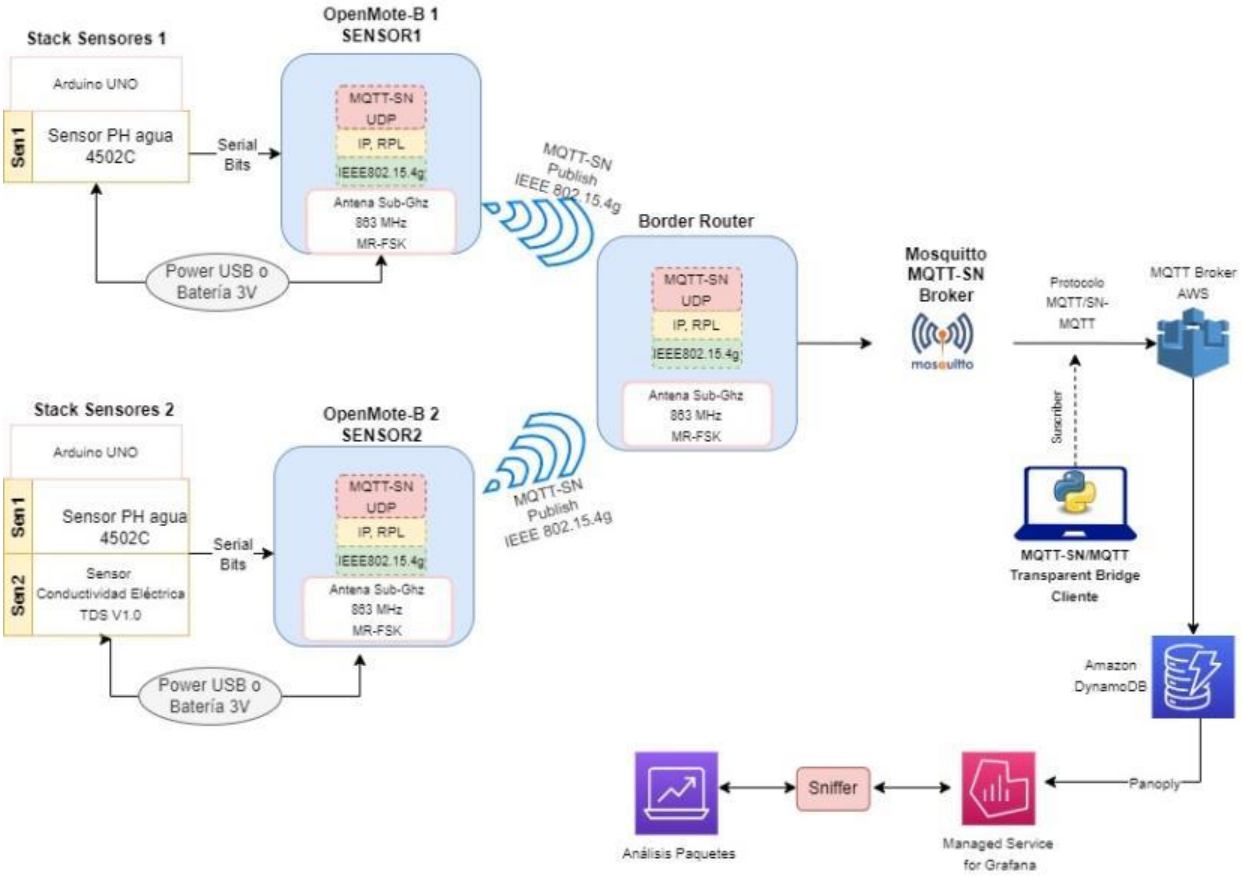



*Figura 34. Arquitectura del sistema.*

*Fuente: Autor*.

### 3.5.4. Diseño de la Red en Radio Mobile

Radio Mobile es un Software gratuito que permite la simulación de la propagación de Radio Frecuencia que puede operar en los rangos de frecuencia de 20 MHz a 20 GHz, su modelo

de propagación es basado en ITS (Longley-Rice). Es permitido dibujar mapas con elevaciones de áreas utilizando datos SRTM descargados para mapear el terreno (Remko, 2022).

Otro de los diseños creados para la Red ha sido mediante la utilización de la herramienta de Radio Mobile, para esto lo que se realizó fue la ubicación de los puntos de los Nodos en el mapa de Google Earth, es decir, el Router de Borde, el Nodo 1 y el Nodo 2, cada uno con sus respectivas coordenadas como se muestra a continuación.

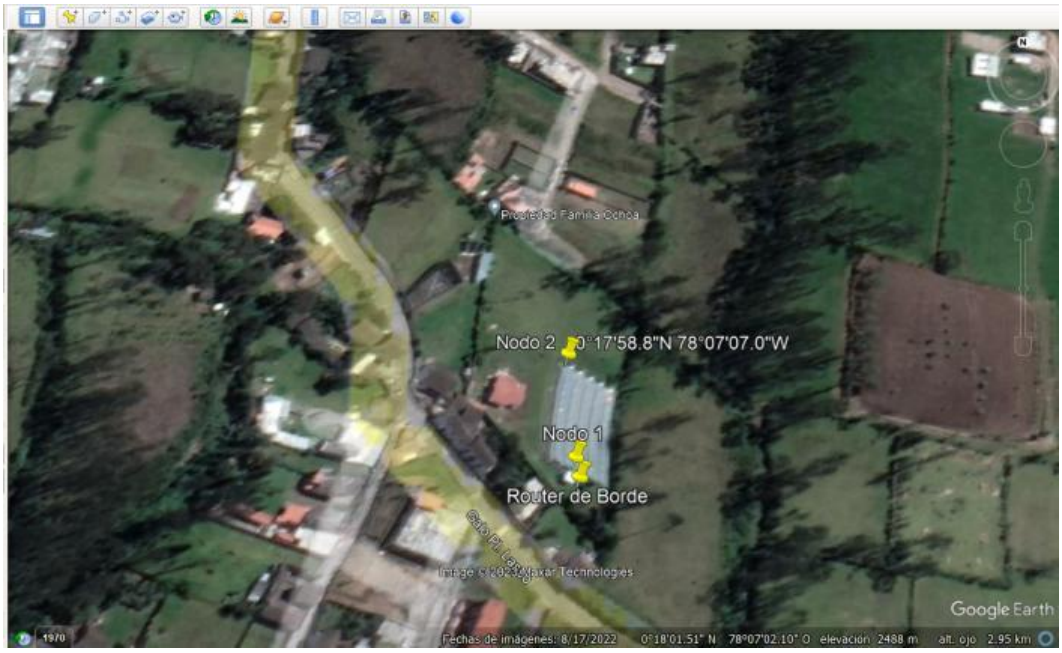



*Figura 35. Ubicaciones de Nodos y Router de Borde en el mapa.*

*Fuente: Autor.*

Partiendo de ello, se procedió a realizar la ubicación de los mismos puntos en el mapa de Radio Mobile, además, se toma en cuenta los datos necesarios para generar el Radio enlace de acuerdo con las distancias, con la potencia y la frecuencia con la que se trabajará, lo mencionado se indica a continuación.

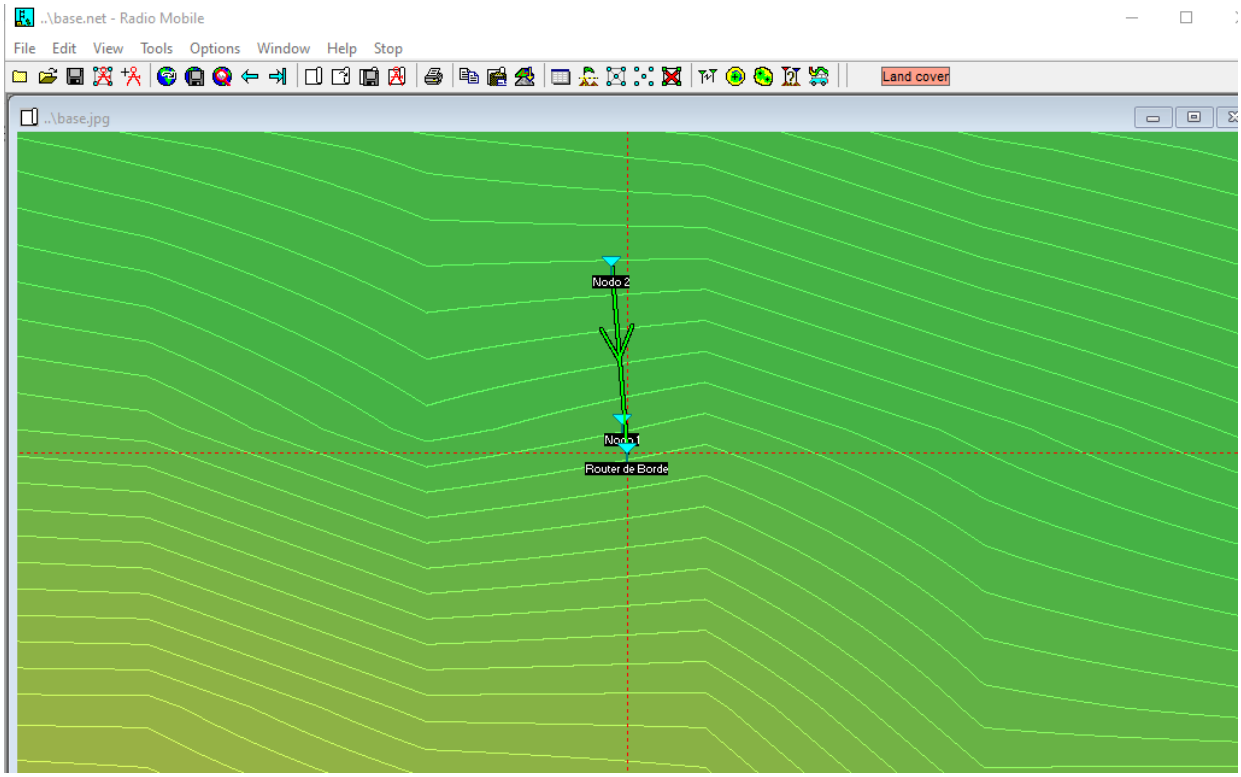



*Figura 36. Radioenlace en Radio Mobile.*

*Fuente: Autor.*

Se analiza cada uno de los nodos que realizan el envío de los datos con respecto al Router de Borde que será quien los reciba. El primer punto a analizar en el Nodo 1 es que se tiene un rango de frecuencia máxima y mínima, la transmisión de los datos que será de 3dB, se tiene la ganancia de la antena entre otros aspectos.

Se toma en cuenta que dentro de la imagen siguiente se podrá observar quien es el que envía y quien es el Receptor, en este caso como Transmitter se tiene al Nodo 1 y como Receiver se tiene al Router de Borde.

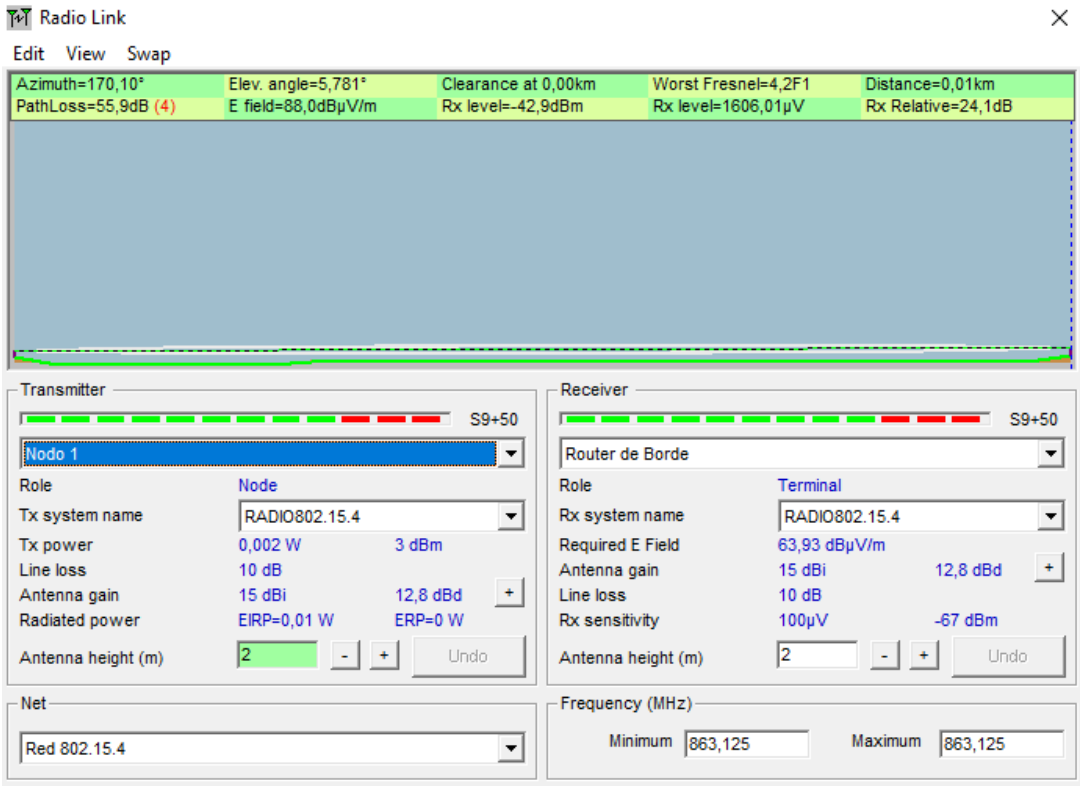



*Figura 37. Radioenlace Radio Mobile Nodo 1.*

*Fuente: Autor.*

Para este enlace se identifica que en Rx level se muestra un -44,0 dBm que es la señal que va a llegar al Receptor; Worst Fresnel=4,2F1 muestra que no hay ningún tipo de problema porque debería estar en 0,6 para que exista; lo siguiente es el Rx Relative=23,0dB, mientras mayor sea ese número mayor será la señal ya que este indica cuanto se pasa del mínimo permitido que para este caso fue de 100 y se muestra que está 23 veces por encima, ese valor nunca debe marcar en rojo ya que indicaría que estaría llegando menos señal de la necesaria.

La línea que indica el tramo también está completamente verde, eso significa que todo el enlace está correcto, si se mostrara de color rojo sería que el nivel no está correcto por lo que la señal no sería buena para poder recibirla.

El valor de E field=87,1 indica el campo eléctrico en un punto, dicho campo eléctrico también se demuestra en la imagen siguiente, aquí se observa cómo evoluciona el nivel de la

señal de acuerdo a los altos y bajos que esta tenga, se representa por la línea verde, por otro lado, la línea amarilla indica cual sería lo mínimo y como se observa, toda la señal del campo eléctrico está por arriba del mínimo.

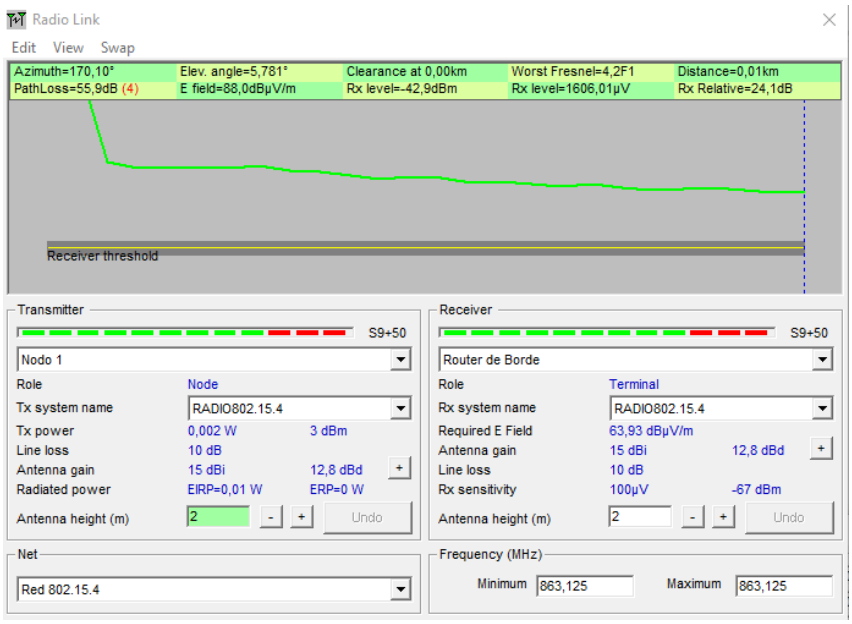



*Figura 38. Campo eléctrico Radio Mobile Nodo 1.*

*Fuente: Autor.*

Así mismo, se tiene el radioenlace desde el Nodo 2 como transmisor hacia el Router de Borde como receptor, en este punto se ve que la zona de Fresnel es de 1,6F1, lo que significa que está en perfecto estado, el Rx leven se encuentra en -59,0 dBm para indicar la señal que va a llegar al receptor; el Rx Relative en este caso es de 8,0 dB que indica que está ocho veces por encima del valor mínimo permitido de 100; otro punto importante es que todo el enlace o ruta se encuentra de color verde indicado que está bien.

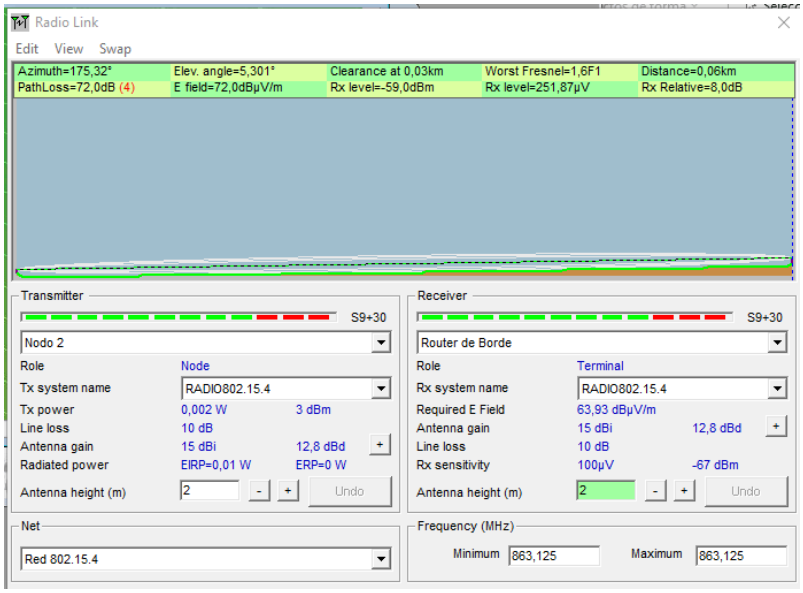



*Figura 39. Radioenlace Radio Mobile Nodo 2.*
*Fuente: Autor.*

El valor de E field=72,1 indica el campo eléctrico en un punto, dicho campo eléctrico también se demuestra en la imagen siguiente, aquí se observa cómo evoluciona el nivel de la señal de acuerdo a los altos y bajos que esta tenga, se representa por la línea verde, por otro lado, la línea amarilla indica cual sería lo mínimo y como se observa, toda la señal del campo eléctrico está por arriba del mínimo.

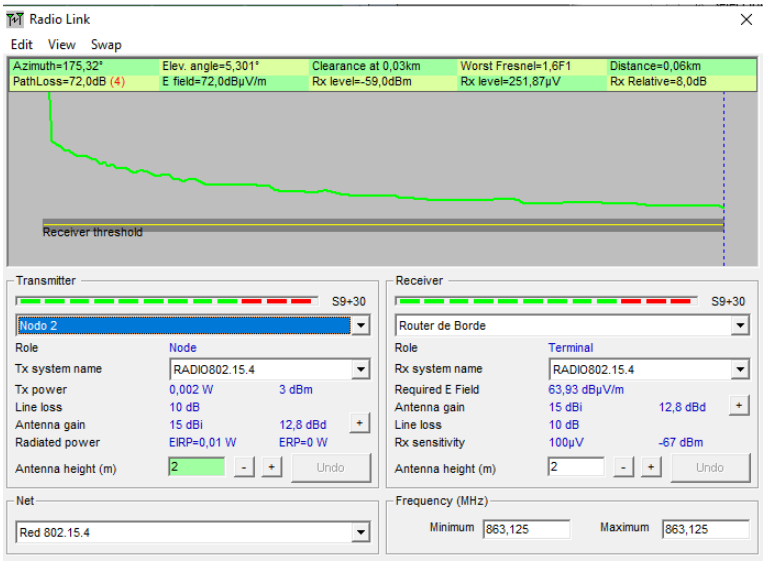



*Figura 40. Campo eléctrico Radio Mobile Nodo 2.*
*Fuente: Autor.*

Si se quiere ver detalles más cercanos se tiene la siguiente imagen, en ella se observa la altura del nodo que es de 2 metros, la altitud del lugar que es de 2499,6 metros sobre el nivel del mar, la altura del Router de borde también está a 2 metros, pero la altura del terreno en la parte del Router de Borde es de 2505,4 metros sobre el nivel del mar; la distancia entre los dos puntos es de 63 metros con una frecuencia de trabajo de 863.1 MHz. El Fresnel es de 1,5F1, por lo que está correcto, en ningún sitio marca 0,6.

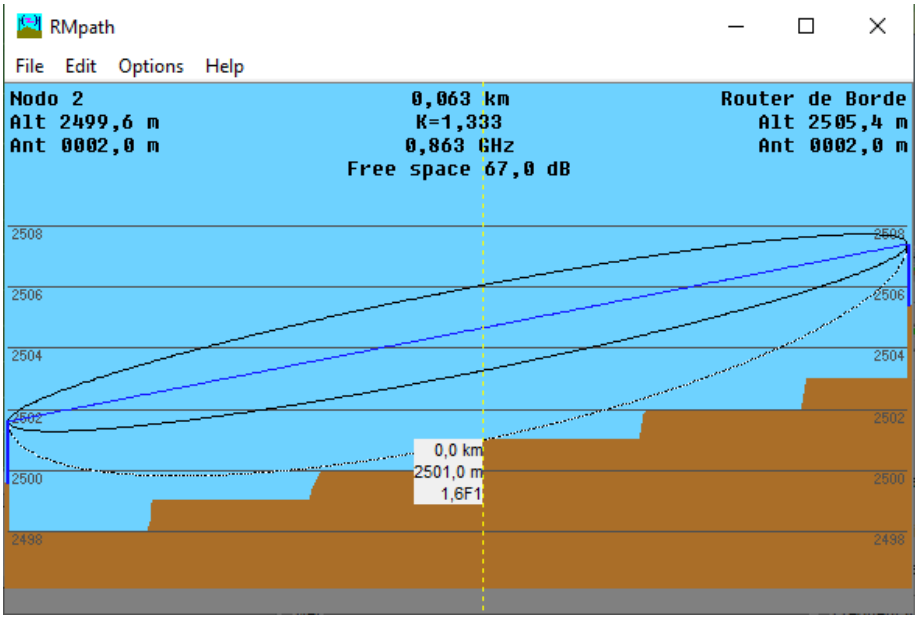



*Figura 41. Zona de Fresnel detallada en Radio Mobile.*

*Fuente: Autor.*

Finalmente, se localizarás las zonas de Fresnel para cada uno de los nodos junto con el Router de Borde, pero esta vez localizadas de manera general en el mapa de Google Earth, esta imagen lo demuestra a continuación.


*Figura 42. Zonas de Fresnel Generales Nodo 1 y Nodo 1.*

*Fuente: Autor.*

Las mismas zonas se pueden identificar acercando el mapa para observar por separado el gráfico década una de las zonas de Fresnel, la primera en indicarse será entre el nodo 1 y el Router de Borde en la posición correcta en el mapa del sembrío.

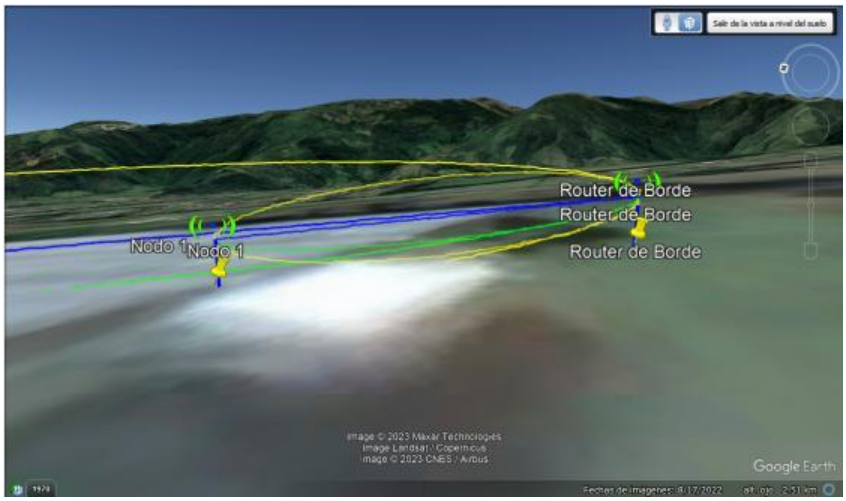



*Figura 43. Zona de Fresnel Nodo 1 Google Earth.*

*Fuente: Autor.*

La zona de Fresnel del Nodo 2 en el mapa de Google Eart también se identifica desde cerca dentro de la plantación de Arándanos estudiada.

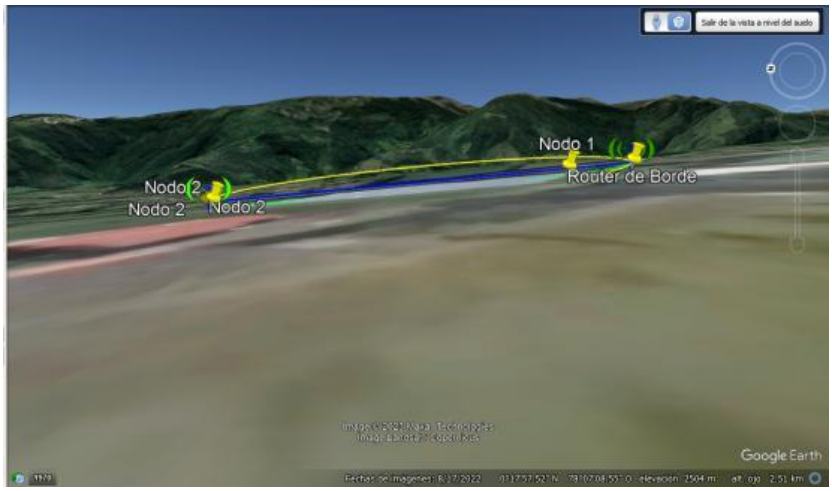



*Figura 44. Zona de Fresnel Nodo 2 Google Earth.*

*Fuente: Autor.*

### 3.5.5. Diagramas por cada capa de IoT

En los apartados que se tienen a continuación, se detallan las conexiones por cada una de las capas de IoT, es decir, la Capa de Dispositivos o Capa Física, Capa de Red o puntos de acceso, Capa de Servicio y Capa de Aplicación.

#### 3.5.5.1. Capa Física

En la capa Física se muestran todas las conexiones realizadas con los equipos y sus diferentes elementos, aquí intervienen los sensores de pH y Conductividad Eléctrica en las dos secciones, así como los arduinos que ayudarán a la recolección de los datos de sensores y los OpenMote-B que son los nodos sensores y otro como Router de Borde; en resumen, esta capa también es conocida como Capa de dispositivo ya que cuenta con varios elementos como sensores, cámaras entre otros que recopilan datos y realizan tareas variadas.

- **Diagrama de Conexión de Nodo 1**

A continuación, se presenta el diagrama de conexión del Nodo 1, en este nodo existirá un sensor que es el de pH, este sensor estará colocado en el agua que llega para el riego, pero antes de estar con los nutrientes para el fertirriego. En la imagen siguiente se pueden observar los

diferentes elementos que se utilizan, en este caso son, un Arduino UNO con su respectivo banco de baterías AA, un OpenMote-B y el sensor de pH. Del sensor de pH se derivan conexiones directas hacia el Arduino, desde la alimentación hacia los 5 voltios del Arduino, los dos pines de GND del sensor van conectados hacia los dos del Arduino, esto se debe a que un pin alimenta a la sonda de ph y el otro pin alimenta a que placa, además, el pin de P0 del sensor es quien enviará los datos recolectados hacia el pin A0 del Arduino. posteriormente, se conecta el otro pin GND del Arduino hacia el GND del OpenMote-B y el pin PB3 se conecta hacia el pin TX0 del Arduino y el PB2 hacia RX de Arduino, respectivamente; finalmente, el OpenMote-B se encuentra alimentado por dos pilas en su parte inferior.

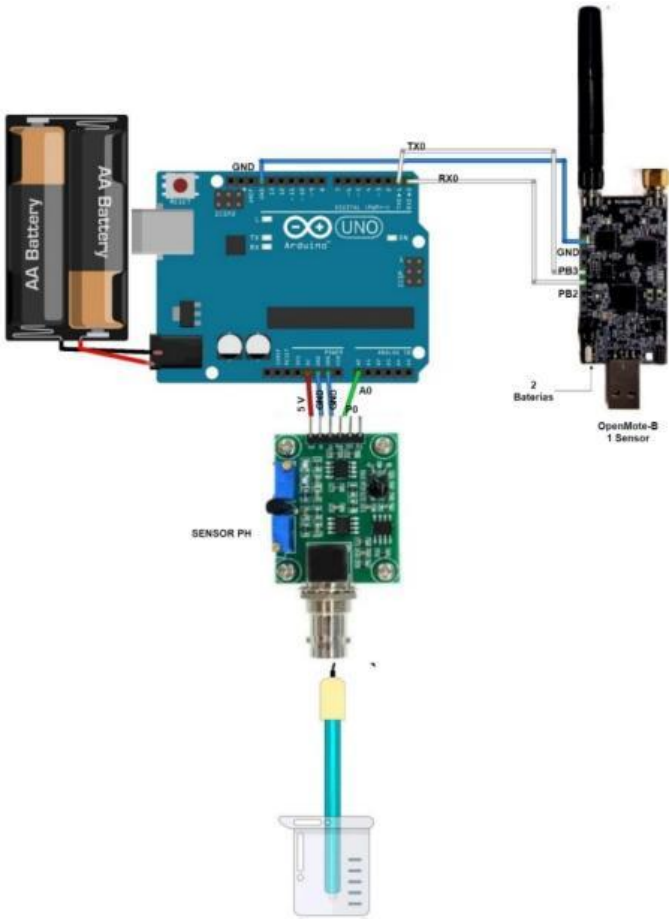



*Figura 45. Diagrama de Conexión Capa Física - Nodo 1.*

*Fuente: Autor.*

- **Diagrama de Conexión de Nodo 2**

El siguiente Diagrama de Conexión se trata del Nodo 2 que cuenta con dos sensores los cuales son el de Ph y el sensor de Conductividad Eléctrica; en este caso, están ubicados en la sección del fertirriego, es decir, después de haber colocado los nutrientes necesarios para dicho proceso.

En cuanto a la fuente de alimentación del Arduino y el OpenMote-B sigue siendo mediante dos pilas AA respectivamente, además, las conexiones entre ellos para GND, PB3 y PB2 siguen siendo las mismas que en el nodo 1 haciendo hincapié en la conexión con los pines de TX y RX de Arduino. Ahora, entre el sensor de pH y Conductividad Eléctrica utilizarán puntos comunes para GND y positivo, distribuidos de la siguiente manera: el sensor de pH esta vez se conecta a una placa o protoboard, del cual, uno de sus pines GND se conecta directamente hacia un pin GND del Arduino, así mismo, el pin P0 de del sensor de ph se conecta hacia el pin A0 del Arduino que es quien recibe los datos. El otro pin GND y el positivo, irán hacia el punto común de la protoboard para conectarse con el sensor de CE y dirigirse hacia el GND y 5V del Arduino.

El sensor de Conductividad Eléctrica cuenta con 3 pines, el GND de este se conecta hacia el punto común con el sensor de pH y luego hacia el GND de Arduino, su positivo se conecta al común con el sensor de pH y luego hacia el Arduino; finalmente, su pin de recolección de datos (A) se conecta de forma directa con el pin A1 de Arduino, este recibirá los datos de esta manera.

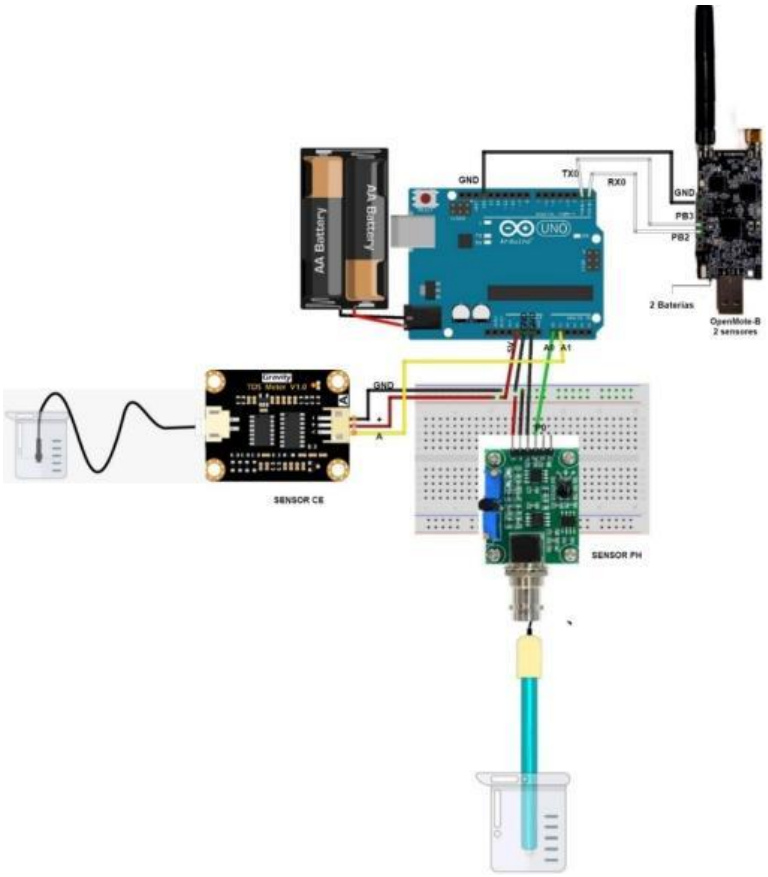



*Figura 46. Diagrama de conexión Capa Física - Nodo 2.*

*Fuente: Autor*.

- **Diagrama de Conexión Router de Borde**

El Router de Borde tiene una conexión bastante sencilla, este simplemente debe conectarse mediante USB hacia la computadora que contendrá la máquina virtual y Sistema Operativo para todo el proceso que se desarrollará, por lo tanto, su conexión se verá solamente de la siguiente manera.

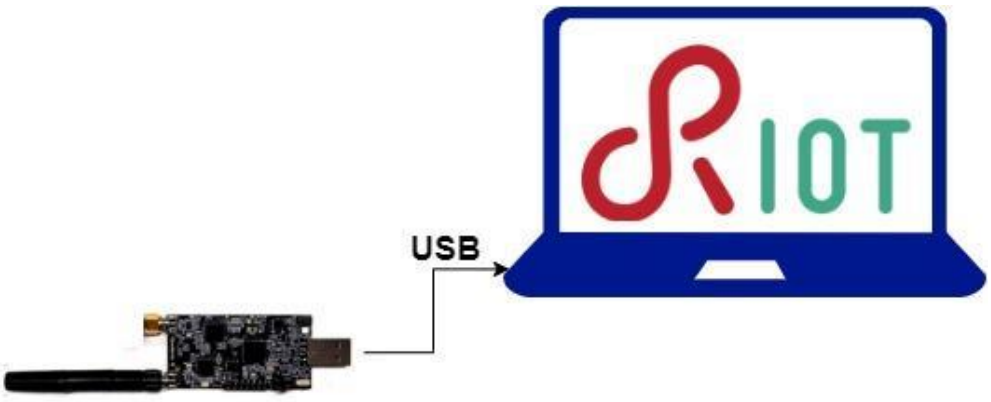



*Figura 47. Diagrama de conexión Capa Física - Router de Borde.*

*Fuente: Autor.*

#### 3.5.5.2. Capa de Red

En la Capa de Red se mostrarán los protocolos para el transporte y enrutamiento que intervienen dentro del desarrollo del proyecto, así también las redes alámbricas e inalámbricas. Es decir, esta capa de IoT transmite los datos desde diversos dispositivos como por ejemplo lo sensores, a centros de datos en las instalaciones o en la nube. Se involucran aquí varios protocolos de transferencia de datos, en este caso se habla de MQTT y MQTT-SN (para redes de sensores) que es para transporte de telemetría de cola de mensajes, para enrutamiento se hará uso del protocolo RPL, para la comunicación entre nodos sensores y Router de Borde se hará uso de IEEE 802.15.4g, además, la máquina virtual utilizada deberá estar conectada previamente a la red WiFi (Simmons, 2022).

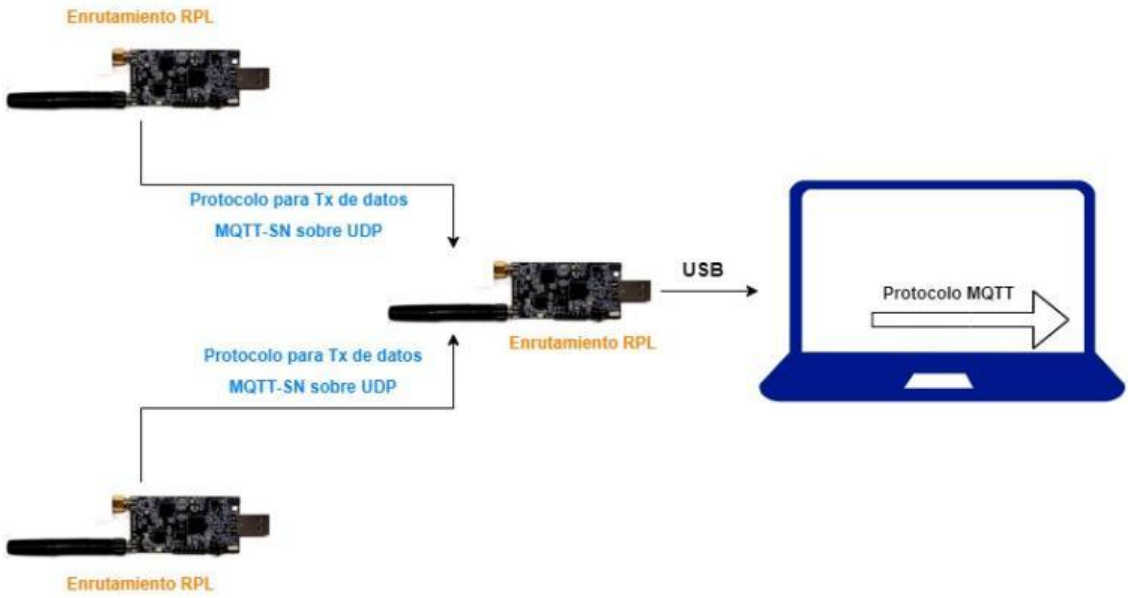



*Figura 48. Diagrama de conexión Capa de Red.*

*Fuente: Autor*.

### 3.5.5.3. Capa de Servicios

La Capa de Servicio o también conocida como Introducción en la Nube, podría encontrarse en la puerta de enlace o en la Nube según la aplicación y la implementación; en este caso de proyecto, se encuentra en la plataforma en la nube de AWS, aquí las aplicaciones pueden acceder a los datos recopilados para luego darle los usos necesarios, como por ejemplo cuando se necesitan datos en tiempo real. Además, los proveedores de plataformas en la nube como AWS, tienen diferentes servicios para IoT que permiten el enrutamiento de datos hacia la nube brindando así la facilidad para el almacenamiento de los mismos y ampliar la infraestructura si existe el caso de que la implementación vaya creciendo; por todo lo mencionado anteriormente, se tiene a la base de datos de DynamoDB dentro de AWS donde se ha creado una tabla denominada “Tesis” que permite el almacenamiento de los datos obtenidos mediante los sensores, su estructura se indica a continuación.


*Figura 49. Diagrama Capa de Servicios.*
*Fuente: Autor.*

#### 3.5.5.4. Capa de Aplicación

Finalmente, se tiene la Capa de Aplicación, esta es la capa que actúa directamente con el usuario, se encuentra en la nube y los usuarios finales utilizan estos datos para diferentes aplicaciones como, por ejemplo, ciudades inteligentes, medicina o agricultura inteligente como es el caso, estos datos se utilizan posteriormente para monitorear el rendimiento del sistema o identificar problemas. Para el proyecto presente, se utiliza Grafana para la visualización de datos finales, sin embargo, desde DynamoDB no se puede pasar los datos fácilmente, para ello, se utiliza un intermediario que es Panoply, está en la nube y es un puente entre DynamoDB y Grafana, se utilizan las credencias de AWS para poder recopilar los datos desde la base de datos hacia Panoply para utilizarlos posteriormente en Grafana.


*Figura 50. Diagrama Capa Aplicación.*
*Fuente: Autor.*

### 3.5.6. Direccionamiento y Topología de Red

A continuación, se presenta la Topología y Direccionamiento de la Red WSN con IEEE 802.15.4g implementada en la plantación de Arándanos para monitorear el Fertirriego de la misma, si indica las direcciones de cada una de las interfaces de los equipos y su ubicación.

*Tabla 23. Direccionamiento de Red.*

| Equipo | Interfaz | Dirección |
|---|---|---|
| Nodo 1 | 6 | 2001:db8::4 /48 |
| Nodo 2 | 6 | 2001:db8::5 /48 |
| Border Router | 4 | 2001:db8::3 /48 |
| | 5 | 2001:db8::2 /64 |
| PC | Tap5 | 2001:db8::1 /64 |

*Fuente: Autor*.

Con el direccionamiento ya señalado, será más sencillo entender la topología de Red con sus respectivas conexiones. Como se indica en la Figura siguiente, se observa que el Nodo 1 y Nodo 2 cuentan con sus direcciones globales en su interfaz 6 Wireless, dichas direcciones son con las que se trabajará precisamente. El Router de Borde tiene su Interfaz 5 que es la Wired que se conecta directamente con la Tap5 creada como puente, mientras que, su interfaz 4 es la Wireless y es la que se conectará con las los Nodos 1 y 2. Además, el Router de Borde se conecta con la Tap 5 por medio de su interfaz Wired.

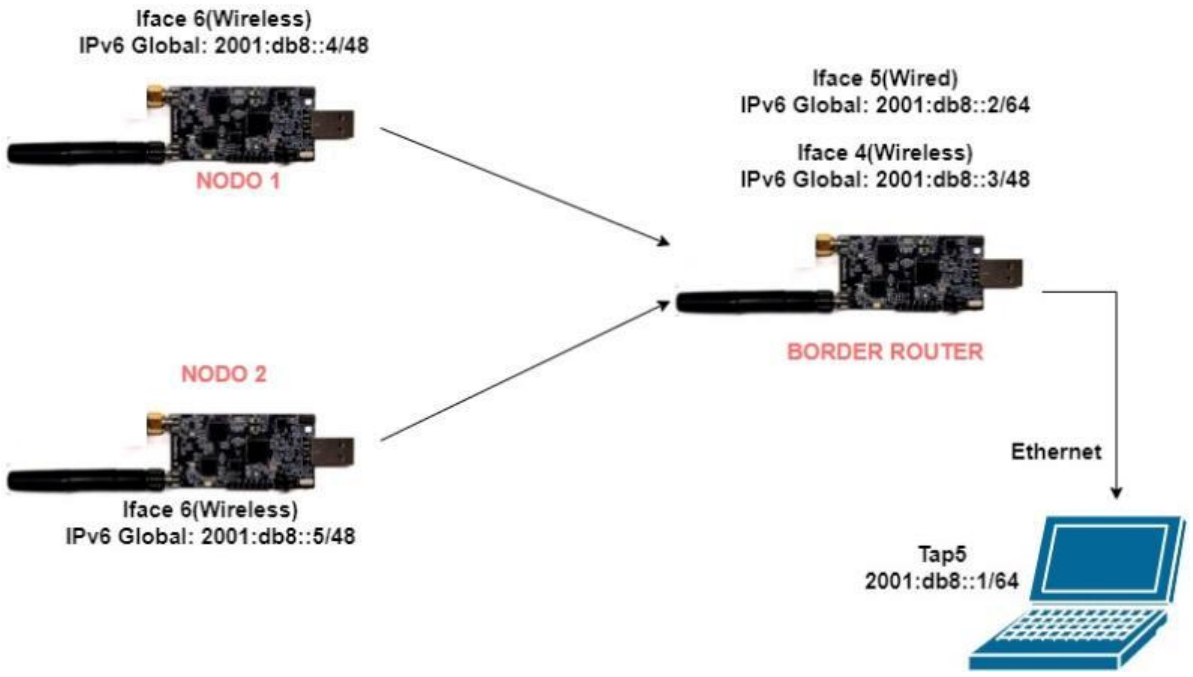



*Figura 51. Topología de Red.*

*Fuente: Autor.*

### 3.5.7. SaaS (Software as a Services)

En el presente Trabajo de Grado se pretende llegar hacia la visualización de los datos en el Monitor de Grafana, para lograr esto, se deben resolver algunos pasos anteriores, estos son que, los datos recopilados en el Bridge serán enviados primeramente hacia DynamoDB dentro de AWS, esta base de datos tendrá una conexión con Panoply que es el puente para traspaso de datos hacia el destino final que es Grafana; antes de mencionar los pasos a seguir, se conocerá de que se trata cada uno de estos Software de Servicio.

- **AWS (Amazon Web Services)**

AWS es una plataforma en la nube que brinda una gran cantidad de servicios integrales de centros de datos a nivel global, cuenta con algunas tecnologías como cómputo, almacenamiento, bases de datos, aprendizaje automático, inteligencia artificial e internet de las cosas, de esta manera es más sencillo llevar aplicaciones a la nube. Dentro de AWS se hará uso

de DynamoDB como base de datos para almacenamiento de las lecturas para monitoreo de fertirriego en la producción de arándanos (AWS, 2022).

Amazon DynamoDB es un servicio de base de datos NoSQL que permite un rápido rendimiento y posee escalabilidad, DynamoDB cuenta con cifrado en reposo, esto significa que brinda una seguridad mejorada ya que permite cifrar los datos en reposo gracias a las claves de cifrado que se almacenan en AWS Key Management Services, de esta forma la carpa de operación se verá reducida y también habrá menos riesgo de involucrar datos confidenciales (AWS, 2022).

Gracias a las utilidades de DynamoDB se podrán crear tablas de bases de datos que se encargan de recuperar cantidades grandes de datos, además, para reducir el espacio de almacenamiento se puede eliminar elementos antiguos de las tablas de manera automática; los principales componentes de DynamoDB son las tablas (colección de elementos), elementos (colección de atributos) y atributos.

- **Panoply**

Panoply es una plataforma que permite de manera más sencilla integrar DynamoDB con Grafana, en este punto, el proyecto indica que los datos almacenados en instancias de DynamoDB serán visualizados al finalizar en el monitor de Grafana, sin embargo, DynamoDB no tiene una conexión directa con la misma, por esta razón, se debe utilizar un intermediario que será Panoply, este permite una prueba gratis de 14 días antes de cobrar por la cuenta.

Esta integración se logra gracias a que Panoply cuenta con conectores nativos para DynamoDB y, además, un soporte integrado para Grafana, de esta forma Panoply actuará como un intermediario de canalización (Panoply, 2021).

- **Grafana**

Se habla de una plataforma que permite la visualización de datos de código abierto y ha sido desarrollada por Grafana Labs, esta plataforma permite que los usuarios que hayan creado una cuenta en Grafana puedan ver sus datos a través de tablas y gráficos que puedan ser unificados en uno o más paneles; también se pueden consultar las métricas en cualquier lugar donde la información se encuentre almacenada; así, será más fácil analizar los datos, identificar anomalías o hacer más eficientes a los procesos (Grafana, 2022).

Gracias a los tres elementos mencionados anteriormente, como lo son DynamoDB de AWS, Panoply y Grafana, se ha logrado la obtención de datos en tablas de DynamoDB, y el traspaso de los mismos hacia Grafana mediante un canal denominado Panoply, los pasos a seguir se muestran detallados en el ANEXO 7 del presente documento.

### 3.6. Implementación y Pruebas

En esta fase de Implementación y Pruebas se podrá observar todo el proceso de conexión de cada uno de los nodos con sus respectivos sensores, además, se realizan las primeras pruebas caseras para verificar el funcionamiento correcto, por lo tanto, las distancias de las primeras pruebas serán cortas dado al espacio en el que se han aplicado. Las segundas pruebas ya son en la plantación real de arándanos, por ello, los nodos tendrán las ubicaciones correctas para la toma de datos de acuerdo con los requerimientos, las distancias serán distintas y los resultados ya arrojarán los valores reales.

#### 3.6.1. Implementación y Configuración

En la siguiente fase se pretende arrancar con el funcionamiento del sistema, para lo cual, se necesita realizar las configuraciones necesarias en ambientes caseros principalmente; de la misma manera, se podrá ir verificando las primeras soluciones de pruebas para comprobar la

conexión de los nodos, las configuraciones de cada uno, la obtención de datos de los sensores, configuraciones automáticas al inicio de los mismos, paso de datos a la Tabla de DynamoDB en AWS y el paso final hacia Grafana.

Los dispositivos SUN están diseñados para funcionar en entornos inalámbricos de bajo consumo y gran escala proporcionando largo alcance, la modulación que se utiliza para el presente proyecto es MR-FSK (Association, 2012).

Como ya es conocido, IEEE 802.15.4g adopta tres PHY: multitasa y multirregional (MR-FSK), MR-desplazamiento QPSK y MR-OFDM. Dentro de los PHY, MR-FSK es el PHY más comercializado debido a que admite operaciones multirregionales, el cambio de frecuencia multitasa multirregional de MR-FSK proporciona una buena eficiencia de potencia de transmisión debido a la envolvente constante de la señal de transmisión, por tal motivo, el mecanismo genérico de los PHY es MR-FSK (Harada, Mizutani, & Fujiwara, 2017).

MR-FSK PHY proporciona tasas de datos de 50 a 400 kbps, por lo tanto, logra una alta eficiencia energética con baja complejidad de implementación porque transmite una señal constante. Por lo tanto, puede cumplir su rendimiento esperado a un bajo costo y con una estructura relativamente simple cuando la tasa de datos es un poco más lenta.

De acuerdo con lo planteado para la realización del proyecto y buscando satisfacer las necesidades de la plantación y de los dueños del cultivo, es beneficioso usar FSK cuando se requiere una estructura económica y de baja complejidad como lo fue en este caso; se demuestra en la siguiente imagen que si es competente FSK gracias a la relación de SNR normalizado y el BER de acuerdo con las otras modulaciones (Oh, Lee, & Hwang, 2015).

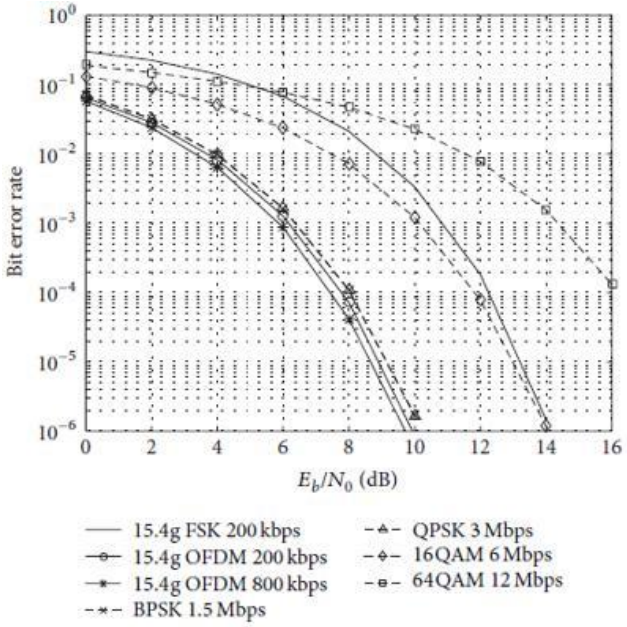



*Figura 52. BER vs Eb/No en canal AWGN.*

*Fuente: (Oh, Lee, & Hwang, 2015)*

Las pruebas se mostrarán de la siguiente manera:

**Configuraciones en Router de Borde**

En primera instancia, se realizan las configuraciones necesarias para lograr una conexión exitosa entre el router de Borde, la tap5 creada en una instancia Linux y cada uno de los Nodos. Al inicio, todas las configuraciones se realizaban de forma manual hasta comprobar que la conexión sea correcta y estable, luego se procedió a introducir en el código los parámetros necesarios para su inicio automático.

Por lo tanto, al iniciar nuestro Router de Borde se encontrará que se agregó las direcciones respectivas a cada una de las interfaces, en este caso, la interfaz 5 es la Wired mientras que la interfaz 4 es la Wireless. Lo siguiente es activar la modulación que utilizará IEEE 802.15.4g que en este caso es MR-FSK o, modulación por desplazamiento de frecuencia multitasa y multiregional, esta se encarga de proporcionar una buena eficiencia de potencia de transmisión debido a la envolvente constante de la señal de transmisión. A continuación, se

establece la potencia en 13 en dBm, el canal que se utilizará es el 0, el índice de modulación es el 1, la velocidad de símbolo es de 50 kbps y el espacio de canal es de 200 kHz; todo esto debido a la tabla de acuerdo a la frecuencia que utiliza que es de 863,125 MHz. Finalmente, se activa el protocolo RPL en la interfaz Wireless tal como se indica en la figura siguiente.

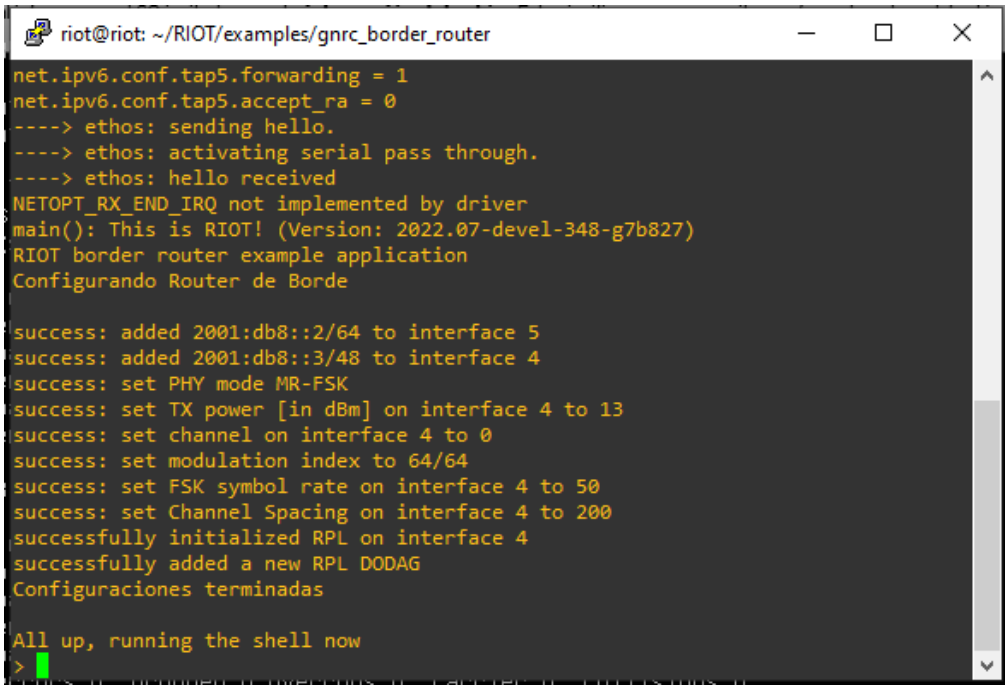



*Figura 53. Configuraciones en Router de Borde.*

*Fuente: Autor.*

### Configuraciones de Tap5

A continuación, se realiza las configuraciones en la tap5 creada al iniciar el Router de Borde, en un nuevo terminal se utiliza el comando → ifconfig para verificar que la tap se haya creado, esta Tap5 cumple la función de ser un puente entre el Router de Borde y los equipos que se encuentran fuera de él, hacia toda la parte interna de la red. Los comandos que se utilizan a continuación son:

- Agregar dirección a la Tap5: *sudo ip a a 2001:db8::1/64 dev tap5*
- Eliminar rutas por defecto existentes en la red: *sudo ip r d 2001:db8::/64 dev tap5*

- Agregar una ruta por defecto hacia la interfaz Wired del Router de Borde: *sudo ip r a 2001:db8::2 dev tap5*
- Agregar una nueva ruta por defecto desde la tap hacia todo lo que se encuentra fuera del Router de Borde mediante la interfaz Wired del mismo*: sudo ip r d 2001:db8::/64 via 2001:db8::2 dev tap5*

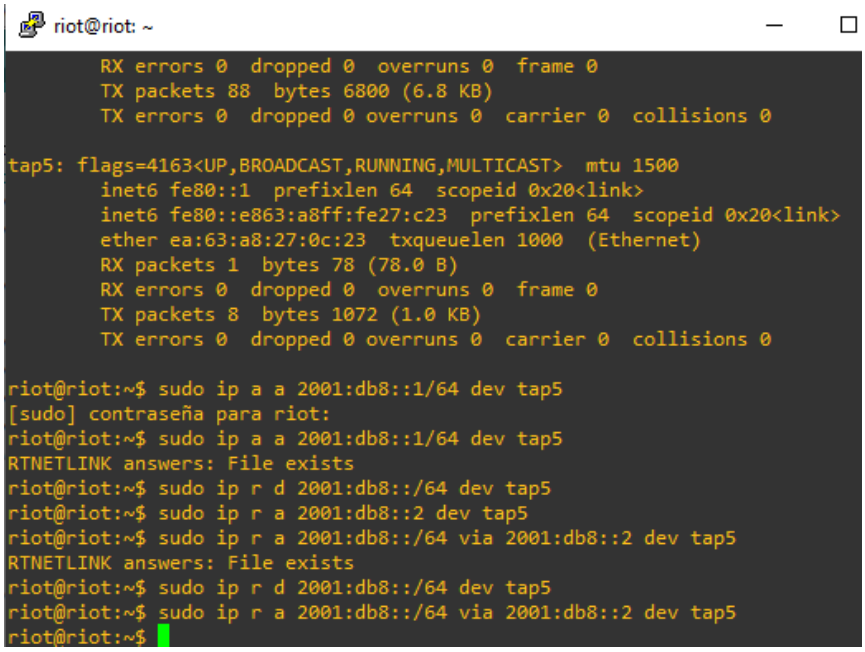



*Figura 54. Configuraciones Tap5.*

*Fuente: Autor.*

**Broker Mosquitto**

El Broker Mosquitto será el encargado de abrir los puertos para establecer las conexiones y recibir los datos enviados, es decir, este tendrá una comunicación con el cliente que enviará los datos para luego reenviar los mismos hacia el Bridge o Puente; por otra parte, en el Broker se observará el intercambio de mensajes MQTT-SN en el proceso de publish/suscribe, estos mensajes serán analizados posteriormente.

En el archivo que se ejecuta mediante la siguiente ruta → ***cd mosquitto.rsmb-master/rsmb/src***, dentro de la ruta se ejecuta el comando → ***./bróker_mqtts config.conf*** y así se

da inicio a los puertos de escucha. Se puede verificar que el puerto 1885 está utilizado por el protocolo MQTT-SN sobre UDP para recibir dichos mensajes, el puerto 1886 se utiliza para MQTT sobre TCP, sin embargo, en este caso se utilizará el puerto 1885.

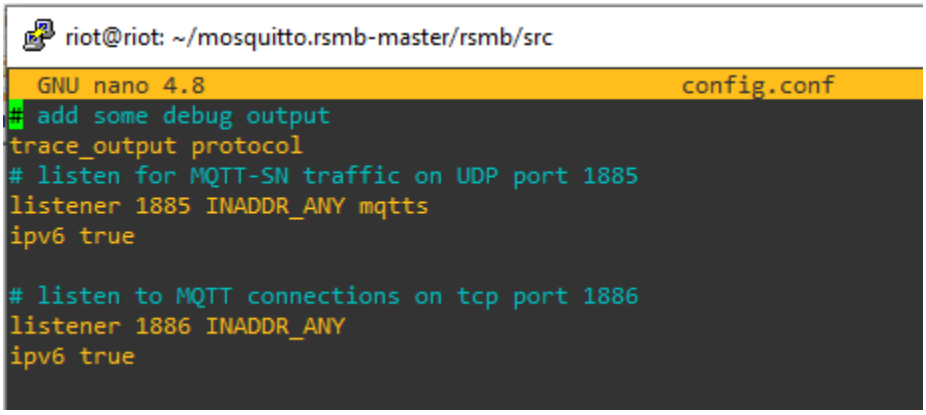



*Figura 55. Archivo de Broker para habilitar puertos.*

*Fuente: Autor.*

**Bridge**

Este puente Transparente hará el papel de recibir los datos desde el Broker para transformar los mensajes MQTT-SN en mensajes MQTT con la finalidad de que luego puedan llegar hasta la Tabla de DynamoDB en la plataforma de AWS.

Se describen algunas partes principales dentro del Bridge o puente, una de ellas es definir algunos aspectos como el recurso en el cual se van a compartir los datos que será en AWS, para esto se define “dynamodb”, la región en la que se encuentra creada la Tabla, el nombre de la Tabla que en este caso es → Tesis, además, se define el puerto a utilizarse, como se puede observar, el puerto es el 1885 dedicado para mensajes MQTT-SN sobre UDP.

```
MQTTSNbridge.py
1  from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
2  from datetime import datetime
3  import pytz
4  import MQTTSNclient
5  import json
6  import boto3
7
8  # conexión a DynamoDB y acceso a la tabla Tesis
9  dynamodb = boto3.resource('dynamodb', region_name='us-east-2')
10 dynamoTable = dynamodb.Table('Tesis')
11 jsonP = ''
12
13 # clientes para MQTT y MQTTS
14 MQTTClient = AWSIoTMQTTClient("MQTTSNbridge")
15 MQTTSNClient = MQTTSNclient.Client("bridge", port=1885)
16
```

*Figura 56. Definición de puerto en Bridge.*

*Fuente: Autor.*

Otro de los aspectos principales es que se debe colocar las claves de AWS para configurar el acceso con el bróker AWS MQTT, además, se indica el tópico que se utilizará para suscribirse, por ejemplo, se hará uso de → ***sensor/station*** y se colocará el **id**, este puede ser 1 o 2 de acuerdo al nodo sensor a referirse.

```
MQTTSNbridge.py
41 # configurar el acceso con el broker AWS MQTT
42 MQTTClient.configureEndpoint("aazppadns27yu-ats.iot.us-east-2.amazonaws.com", 8883) #aazppadns27yu-ats.iot.us-east-2.amazonaws.com
43 MQTTClient.configureCredentials(path+"AmazonRootCA1.pem",
44                                  path+"a694b69cdd0034fd87318f2e79ab39dd2e9de9bfce8a9555bfd3c74d9912c2c3-private.pem.key",
45                                  path+"a694b69cdd0034fd87318f2e79ab39dd2e9de9bfce8a9555bfd3c74d9912c2c3-certificate.pem.crt")
46
```

*Figura 57. Ingreso de claves en el Bridge.*

*Fuente: Autor.*

### 3.6.2. Pruebas

En la siguiente Fase se pretende dar inicio a las pruebas en dos escenarios diferentes, el primero será en un ambiente casero con la finalidad de comprobar la conexión y los datos obtenidos de las mediciones, el segundo escenario será ya en la producción de Arándanos para verificar diferentes aspectos necesarios a destacar en el proyecto.

#### 3.6.2.1. Primer Escenario

En este primer escenario, las pruebas mostradas serán dentro de casa, un punto a analizar importante son las distancias tomadas para realizar los pings y verificar el RSSI con el que se

está transmitiendo, además, se verifica la conexión de cada uno de los nodos utilizados en el proceso.

### Pruebas de Conexión y RSSI

El RSSI es el indicador de la intensidad o fuerza de la señal recibida en redes inalámbricas, cabe mencionar que esto no define la calidad. Para comprobar esto, se realizó la conexión normal de los equipos y, desde el Router de Borde se realizó un ping hacia los nodos sensores ubicados primeramente a una distancia de 1 metro, el ping se realiza hacia la 2001:db8::5 que es el Nodo 2 con dos sensores y el ping hacia la 2001:db8::4 que es la dirección del Nodo 1 con un sensor.

Se puede observar que el rssi hacia el Nodo1 es de un promedio de -46 dBm, esto hace referencia a que es un valor de señal de transmisión estable e idóneo, esto se debe a la cantidad de sensores y elementos ocupados en este escenario; el tiempo de transmisión es el mismo para los dos casos de nodos sensores.

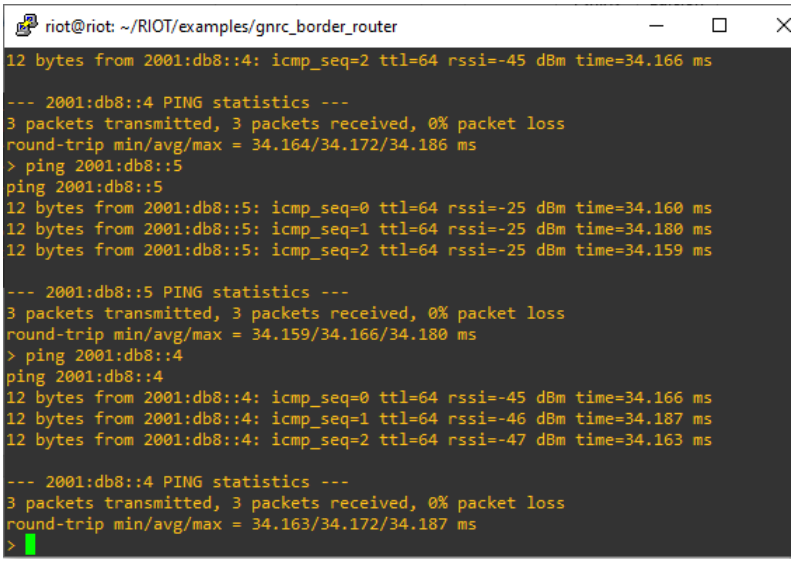



*Figura 58. Prueba de RSSI en 1 metro.*

*Fuente: Autor.*

Se realizó una nueva prueba a los 10 metros de distancia, esta vez con el Nodo2 que contiene dos sensores, esta vez que observa que el RSSI es mucho mejor ya que está en un promedio de -60 dBm, es decir, la señal de transmisión está estable en la comunicación.

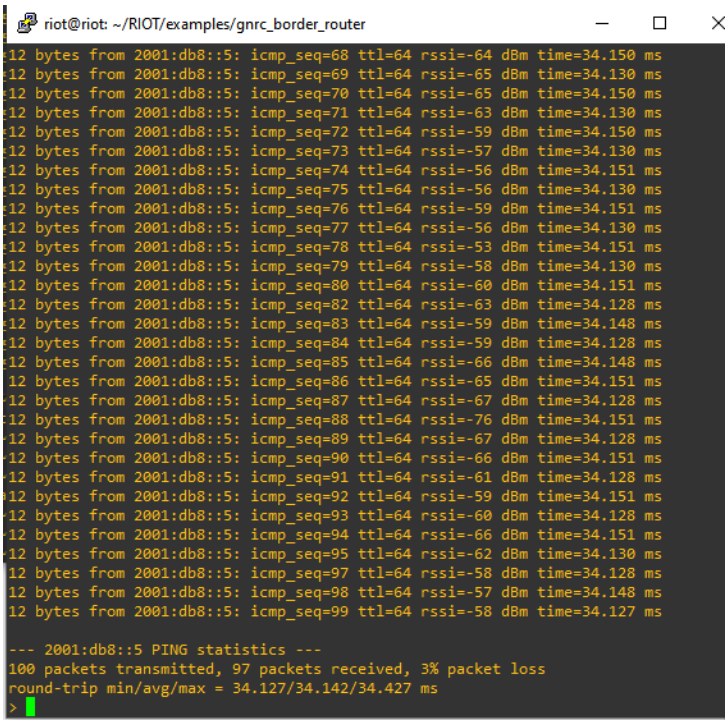



*Figura 59. Prueba de RSSI a los 10 metros.*
*Fuente: Autor.*

Cabe mencionar que todas estas pruebas de Conexión y de toma de datos para el Monitoreo, se realizaron primeramente en ambientes caseros, en los cuales se tomó medidas en espacios como agua simplemente para verificar la toma correcta de los datos y su transmisión final hasta Grafana, estas pruebas con todos los componentes conectados como baterías a los Arduinos y pilas a los OpenMote se muestran a continuación.


*Figura 60. Prueba de Conexión Nodo 1.*
*Fuente: Autor.*

En la Figura 60 se muestra la conexión completa del Nodo 1 con su respectivo sensor de pH, el banco de baterías para el Arduino y el OpenMote con sus pilas para enviar ya los datos de forma inalámbrica.

En la Figura siguiente se tiene la Prueba de Conexión del Nodo2 con su respectivo sensor de CE y su sensor de pH, así mismo el Banco de baterías para el Arduino y sus respectivas pilas para el OpenMote-B, de esta manera, todos los datos ya podrán ser enviados de forma inalámbrica hacia el Router de Borde para luego seguir con el proceso respectivo hasta obtener los datos en Grafana.

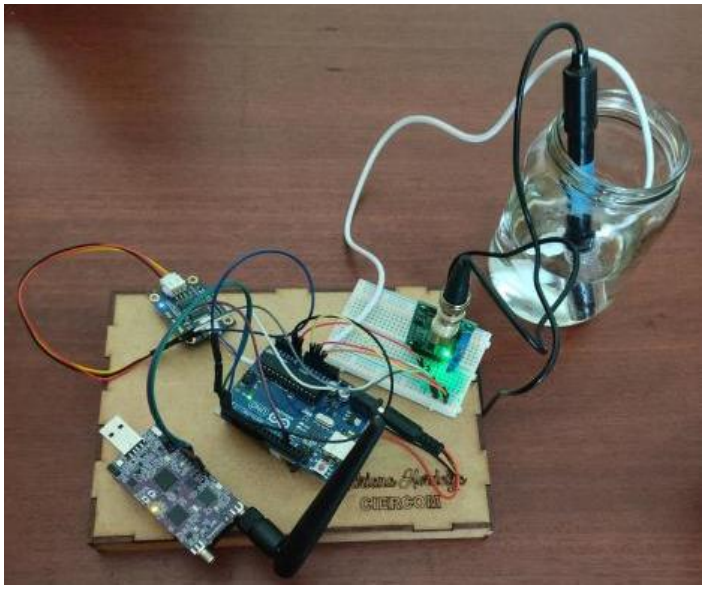


*Figura 61. Prueba de Conexión Nodo 2.*
*Fuente: Autor.*

Lo siguiente fue verificar la conexión del Router de Borde directamente a la PC, esto con el fin de iniciar el mismo y que se de paso al proceso respectivo, esta conexión se observa a continuación para lograr obtener los datos correctos desde los Nodos inalámbricos.

*Figura 62. Conexión de Rputer de Borde a PC para obtención de datos.*
*Fuente: Autor.*

### Modo Escucha y conexión en Mosquitto

En la ruta que se observa en la Figura siguiente, se inicia el bróker mosquitto, en este caso será Mosquitto Really Small Message Broker (RSMB) este será el punto en el cual se verifica la conexión y los mensajes que llegarán del establecimiento de conexión de MQTT-SN, por tal motivo, el puerto 1885 esperará el tráfico UDP MQTT-SN y el puerto 1886 el tráfico TCP MQTT.

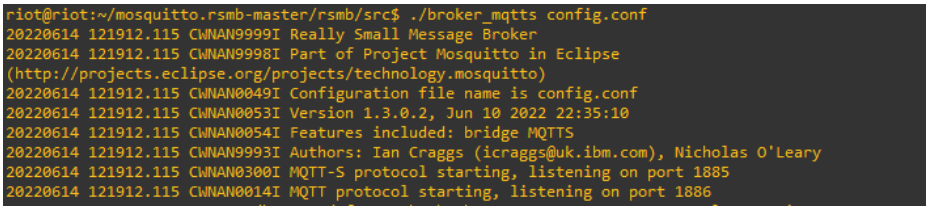



*Figura 63. Puertos de escucha Mosquitto.*
*Fuente: Autor.*

Mientras se mantiene en escucha, se puede iniciar los equipos de Nodos sensores y de la misma manera iniciar el Bridge. Cuando un Nodo sensor se inicia y comienza con la conexión hacia el mosquito, aparecen los siguientes mensajes, junto con los mensajes de publicación y finalmente los de Desconexión.

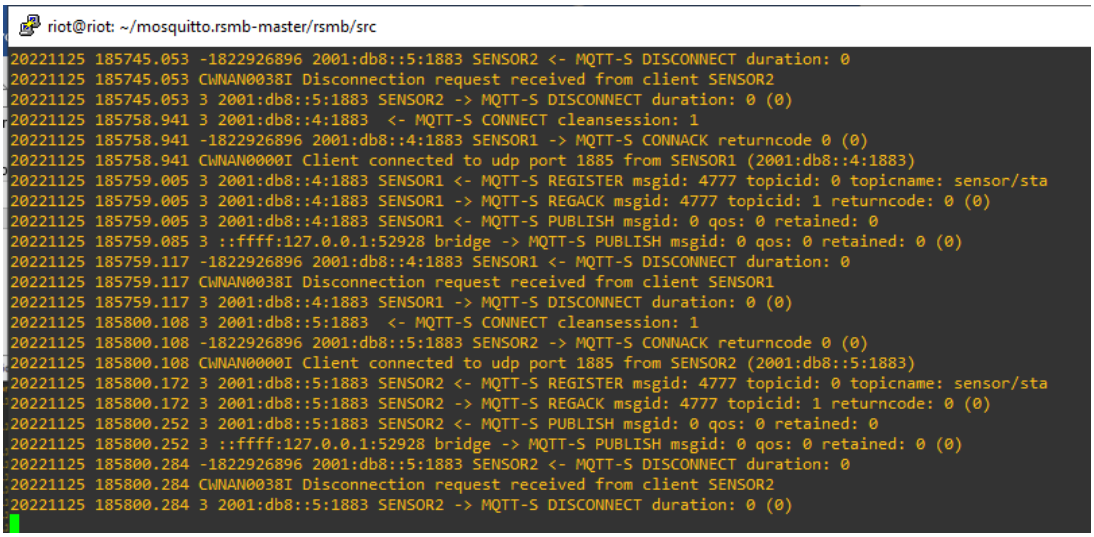



*Figura 64. Paquetes recibidos en Mosquitto de Nodo Sensor1 y Nodo Sensor2.*
*Fuente: Autor.*

### Inicio del Bridge

En la ruta siguiente, se debe dar inicio al Bridge, este es un puente transparente entre paquetes MQTT-SN y MQTT entre el RSMB y AWS, esto se debe a que RSMB no es compatible directamente con el servicio de AWS, por tal motivo se crea un Script de Python que permitirá enviar los datos hacia las tablas de DynamoDB de AWS. Dentro de la carpeta se ejecuta el Bridge y se colocarán los ID de los dispositivos que están transmitiendo datos y así se suscribirán al mismo.

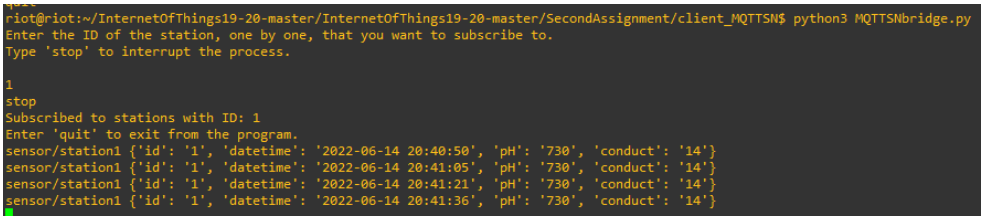



*Figura 65. ID de nodos a los que se escuchará.*
*Fuente: Autor.*

Los datos recibidos se muestran a continuación con sus respectivos ID, tanto el Nodo 1 que contiene un sensor de pH, como el Nodo 2 que contiene otro sensor de pH y un sensor de Conductividad.

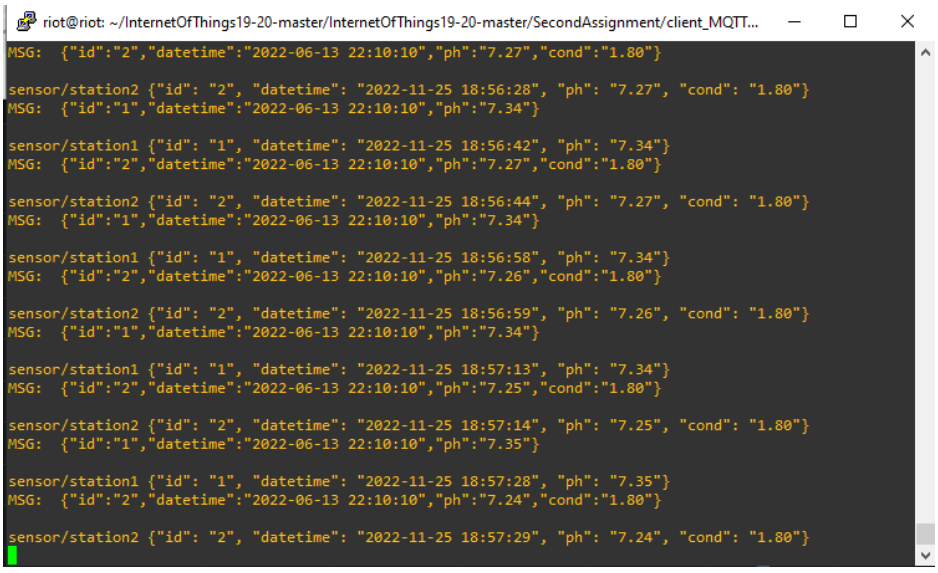



*Figura 66. Recepción de datos en Bridge mediante suscripción de ID.*
*Fuente: Autor.*

**Visualización de datos en AWS**

Como se había mencionado anteriormente, el Bridge será encargado de enviar los datos obtenidos hacia el AWS gracias a las credenciales indicadas en el bridge y los accesos a las tablas, esos se muestran a continuación.

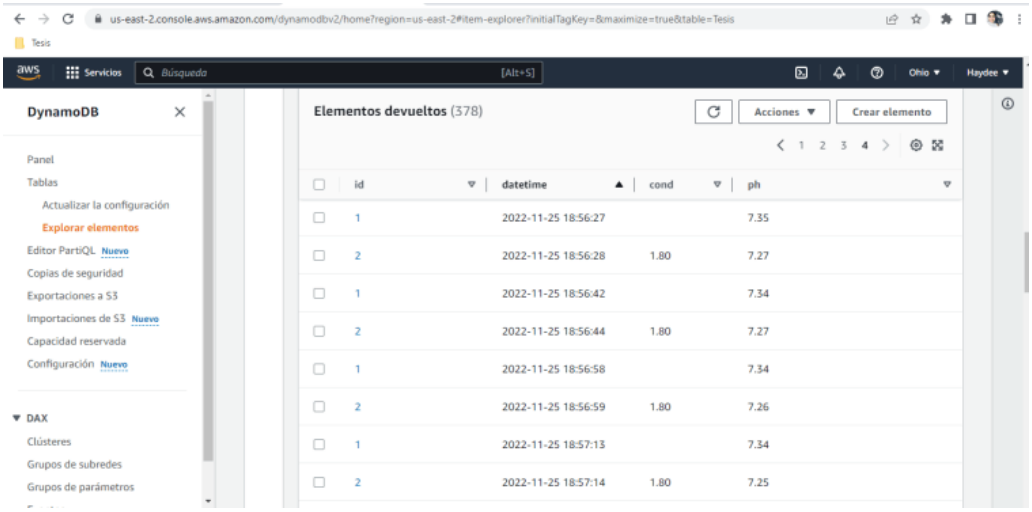



*Figura 67. Visualización de datos en Tabla "Tesis" de DynamoDB.*
*Fuente: Autor.*

### 3.6.2.2. Segundo Escenario – Producción de Arándanos

En este segundo escenario se toma ya en cuenta la Producción de Arándanos, se visitó el sitio y se procedió a realizar las respectivas instalaciones de los equipos para la toma de Pruebas, además, mediante la realización de un ping desde el Router de Borde, se pudo comprobar la distancia a la cual se perdía la comunicación entre el Router de Borde y los Nodos con sensores.

Lo primero que realizamos fue la llegada al lugar de la plantación y verificar cada uno de los puntos que van a ser tomados en cuenta para la toma de medidas; la Figura que se indica a continuación, es un paneo general de la plantación, en este punto cuentan con 600 plantas en un estado de producción continua y 1000 plantas más que están en su proceso de crecimiento.

*Figura 68. Paneo general de la plantación de arándanos.*
*Fuente: Autor.*

Lo siguiente fue verificar cual sería el Centro de Control que, al mismo tiempo es el lugar donde se tomarán las medidas para el Nodo Sensor 1 y también se observarán los productos que serán tomados en cuenta para la toma de medidas del Nodo 2. La caseta de color verde será el Centro de Control de datos y, a su vez lo explicado anteriormente.

*Figura 69. Centro de Control de Datos.*
*Fuente: Autor.*

Se procedió con un recorrido para visualizar las plantas y como es su riego por goteo en el cual interviene el Fertirriego en análisis en este proyecto de titulación; como se puede ver en

las Figuras siguientes, cada una de las plantas cuenta con un tipo de tubería por la que circula el Fertirriego que no es más que la mezcla de sustancias para lograr la nutrición correcta de la planta, cada manguera delgada de color blanca cuenta con una punta, esta contiene algunos agujeros por los cuales saldrá la mezcla de Fertirriego.

*Figura 70. Fertirriego en las plantas de Arándano.*
*Fuente: Autor.*

El dispositivo de color negro es el que se inyecta en cada una de las plantas, de esta se desprenden las sustancias previamente mezcladas para el Fertirriego de las plantas, todo el elemento es de plástico y este va sujeto a una manguera delgada de color blanco, éstas a su vez, tienen una conexión con una manguera negra que serán las que van conectadas de forma directa hacia las bombas que filtrarán el agua y extraen las sustancias que se necesitan para el riego de las plantas de arándanos.

*Figura 71. Dispositivo y manguera para Fertirriego de Arándano.*
*Fuente: Autor.*

A continuación, se procede con la conexión y configuración del Ruter de Borde, por lo tanto, se dirige hacia el Centro de Control y se procede a conectar el Router de Borde hacia la PC como ya es conocido, se da inicio al Router de Borde con sus respectivas configuraciones.

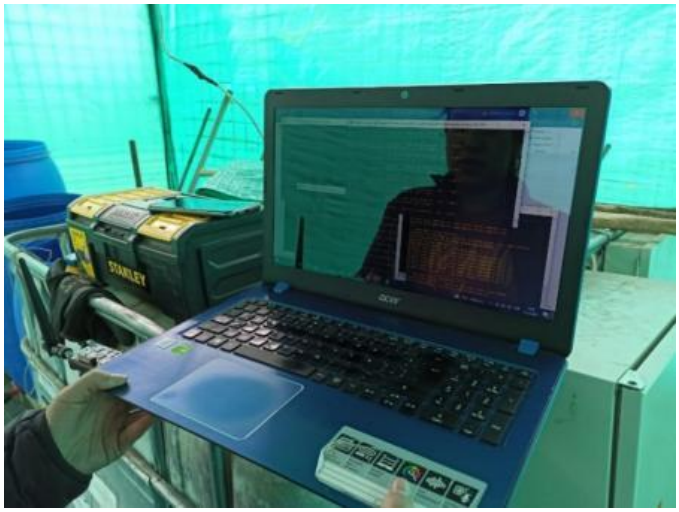



*Figura 72. Colocación de Router de Borde en Centro de Control.*
*Fuente: Autor.*

Posteriormente, se da inicio al Router de Borde y se observan las características y configuraciones que están previamente creadas en el mismo código de inicio, esto se da con el fin de no introducir manualmente las direcciones, sino que ya estén creadas automáticamente como se muestra a continuación.

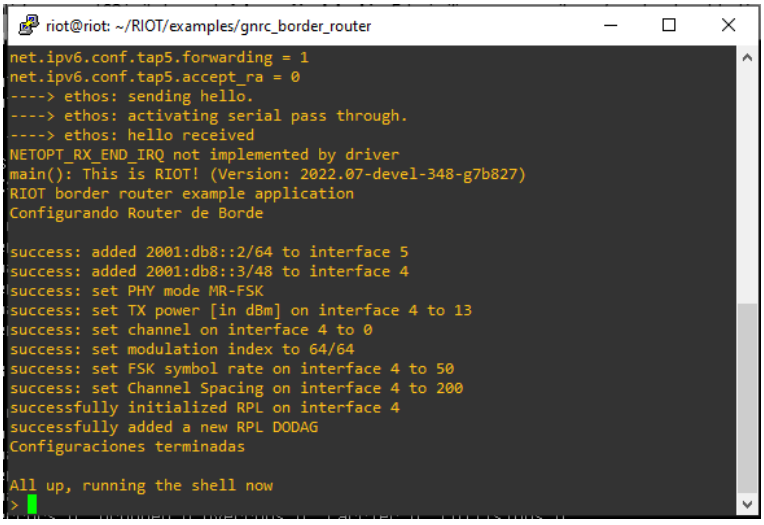



*Figura 73. Inicio configuraciones automáticas Router de Borde.*
*Fuente: Autor.*

Una vez iniciado el Router de Borde, se necesita establecer las configuraciones en la interfaz Tap5 que se crea al iniciar Border Router, estas configuraciones son iguales a las mencionadas en el Primer Escenario.

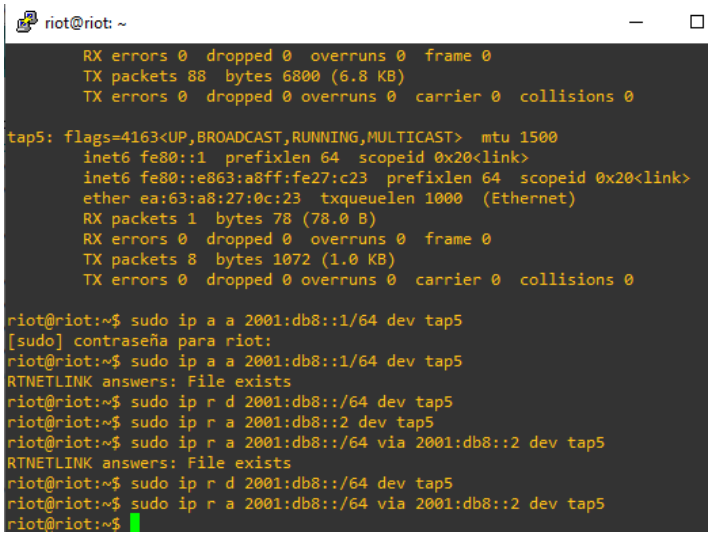



*Figura 74. Configuraciones de Tap5.*
*Fuente: Autor.*

Antes de colocar los Nodos, se debe poner en modo de escucha al Broker Mosquitto por los puertos que utilizará MQTT-SN sobre UDP para el intercambio de mensajes y el traspaso de datos obtenidos, para esto dentro de la carpeta de Mosquitto se utiliza el comando →

***./bróker_mqtts config.conf***, inmediatamente los puertos se ponen en espera y escucha para empezar con el intercambio de mensajes mqtt-sn.

```
riot@riot:~/mosquitto.rsmb-master/rsmb/src$ ./broker_mqtts config.conf
20220614 121912.115 CWNAN9999I Really Small Message Broker
20220614 121912.115 CWNAN9998I Part of Project Mosquitto in Eclipse
(http://projects.eclipse.org/projects/technology.mosquitto)
20220614 121912.115 CWNAN0049I Configuration file name is config.conf
20220614 121912.115 CWNAN0053I Version 1.3.0.2, Jun 10 2022 22:35:10
20220614 121912.115 CWNAN0054I Features included: bridge MQTTS
20220614 121912.115 CWNAN9993I Authors: Ian Craggs (icraggs@uk.ibm.com), Nicholas O'Leary
20220614 121912.115 CWNAN0300I MQTT-S protocol starting, listening on port 1885
20220614 121912.115 CWNAN0014I MQTT protocol starting, listening on port 1886
```

*Figura 75. Puertos de escucha para MQTT-SN.*
*Fuente: Autor.*

Se procede a ir hacia la primera instancia para colocar el Nodo1 con su respectivo sensor de pH, como es sabido, este primer sensor está ubicado en la zona de recolección del agua que será utilizada posteriormente en el fertirriego, esta se encuentra recolectada en el Centro de Control en un pozo, tal como se muestra a continuación.

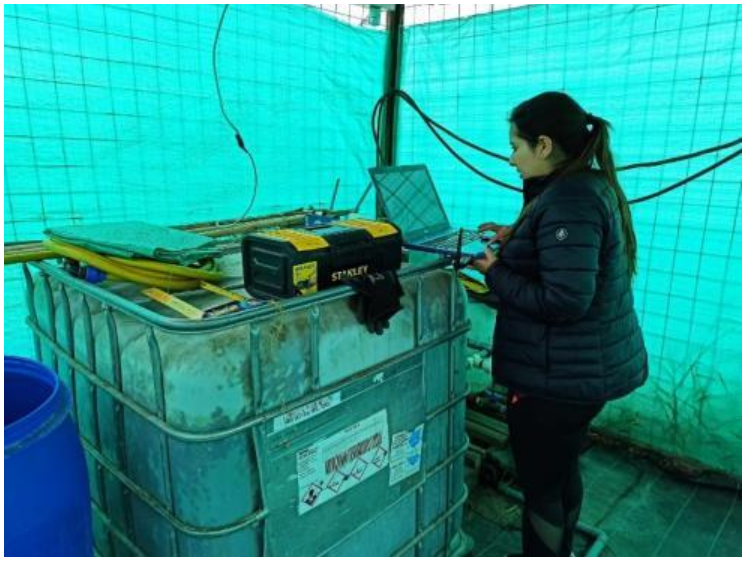



*Figura 76. Pozo para recolección de agua para Fertirriego.*
*Fuente: Autor.*

Se realiza toda la conexión del Nodo 1, esto quiere decir que, el OpenMote-B, el Arduino y el sensor de pH, cada equipo con su respectiva fuente de energía. Como se visualiza en la Figura, el sensor, el OpenMote-B y el Arduino para el primer Nodo, se han colocado dentro de

una caja con el respectivo nombre, ha sido colocado con los dispositivos encendidos para inmediatamente proceder a tomar los datos que se recolecten.

*Figura 77. Armado de Nodo 1 en plantación.*
*Fuente: Autor.*

Se tapa la caja y, por lo orificios se procede a sacar tanto la Antena del equipo OpenMote-B como el sensor de pH, el sensor es colocado en el agua que se encuentra en el pozo de recolección, esta agua es directamente tomada de las fuentes, es decir, no cuenta con ningún tipo de aditivo por el momento, más tarde, esta será absorbida por la bomba para luego ser mezclada con los nutrientes para el Fertirriego.

*Figura 78. Colocación de Sensor de pH en zona de medición.*
*Fuente: Autor.*

Lo siguiente es verificar la zona para colocar el Nodo 2, en este caso, este nodo debe ser colocado ya en la parte final del riego, es decir, cuando tenga ya la mezcla de los nutrientes para el mismo; por esto, se debe dirigir de igual manera hacia el Centro de Control, en este espacio están los elementos necesarios para la mezcla, como se observa en la Figura siguiente, se encuentra en los tanques azules, de izquierda a derecha con: Ácidos, Nitratos y Sulfatos, necesarios para este proceso, además, está el contenedor de agua que se necesita para toda la mezcla general; de cada tanque se puede ver que existen conductos que van directamente hacia la bomba que se encargará de absorber y mezclar.


*Figura 79. Sustancias necesarias para el Fertirriego.*
*Fuente: Autor.*

Como ya se mencionó con anterioridad, el primer tanque azul cuenta con los ácidos que se necesitan para la mezcla, este líquido se presenta de un color transparente tal como se indica a continuación en la Figura, nombrándolos de izquierda a derecha, el líquido de color oscuro son los Nitratos y Finalmente se encuentran los Sulfatos, este también tiene un color claro, casi transparente, pero con algunas burbujas dentro.

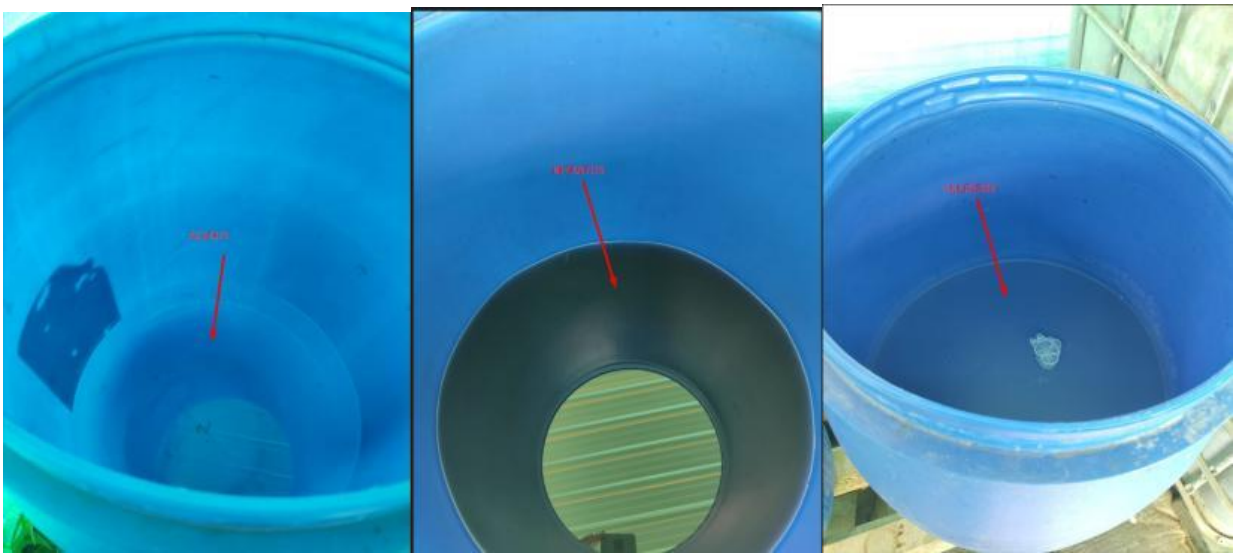


*Figura 80. Ácidos, Nitratos y Sulfatos para Fertirriego en Arándanos.*
*Fuente: Autor.*

La bomba que se utiliza para absorber cada una de las sustancias necesarias para el Fertirriego se muestra a continuación, esta bomba cuenta con varios compartimientos que luego podrán enviar ya la mezcla completa para el Fertirriego, se distribuirá por medio de las mangueras y luego por cada punto de riego por goteo en cada una de las plantas.

*Figura 81. Bomba de succión y mezcla para Fertirriego.*
*Fuente: Autor.*

El armado del Nodo 2 debe estar listo, este de igual manera consta de un OpenMote-B, un Arduino 1, cada uno con sus fuentes de energía, un sensor de pH y un sensor de Conductividad Eléctrica; dado a las baterías y pilas que utilizan estarán completamente de manera inalámbrica, previamente los códigos han sido cargados para cada uno de ellos.

Tanto el sensor de CE como de pH contienen puntos en común de GND y positivo, estas conexiones se mostraron en el diseño del sistema.

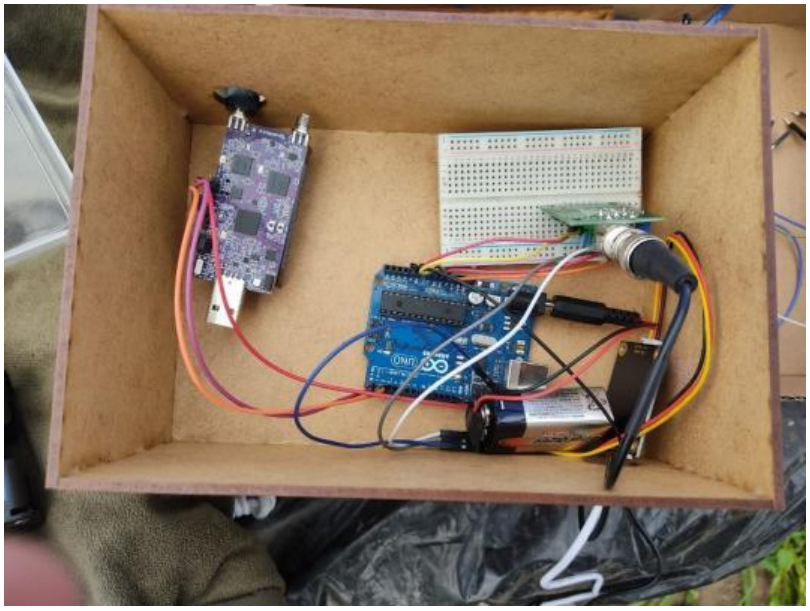


*Figura 82. Armado de Nodo 2 en plantación de Arándano.*
*Fuente: Autor.*

Una vez que se haya realizado la conexión completa del Nodo 2 se procede a su colocación, para tener el concentrado de la mezcla de sustancias para el Fertirriego se introduce en un vaso una de las mangueras pequeñas que van directamente hacia la planta, de esta manera obtendremos el Fertirriego que va hacia la planta y será más fácil realizar la lectura de los datos monitoreados por estos sensores; el sensor de CE es aquel que tiene la sonda blanca y el de pH contiene la sonda negra; todos los dispositivos integrados se colocaron dentro de una caja para mayor facilidad de portabilidad.

*Figura 83. Colocación de sensores del Nodo 2 en la zona de medición.*
*Fuente: Autor.*

Se puede observar los sensores ya conectados y colocados en la zona de toma de datos, estos están en su respectiva caja y se puede indicar que la antena del OpenMote-B sale por uno de los agujeros de la caja realizada.

*Figura 84. Sensores colocados en la zona de toma de datos mediante el Nodo 2.*
*Fuente: Autor.*

El sensor de CE tiene un poco de retraso en su lectura inicial de datos debido a los delay que contiene, por tal motivo, se esperó un momento hasta verificar que los datos se estén tomando de manera correcta, mientras esto sucedía, se trasladó el Router de Borde que estaba en el Centro de Control hacia la zona de medición como tal, como se ve en este caso se está tomando los datos de la recolección de Fertirriego que va a una planta directamente, esto se recoge en un vaso para saber exactamente lo que toma la planta.

*Figura 85. Toma de datos de sensores de pH y CE en el Nodo 2.*
*Fuente: Autor*.

Cuando los nodos están listos para transmitir sus datos, se ejecuta el puente transparente o Bridge y se colocan las ID de los nodos que se desea escuchar, en este caso es el ID=1 para el Nodo 1 con el sensor de pH y el ID=2 es para el Nodo 2 con el nuevo sensor de pH y sensor de Conductividad Eléctrica. El Bridge se encuentra dentro de las carpetas indicadas anteriormente en InternetOfThings19-20-master. Los datos que se muestran a continuación son los datos reales del Fertirriego en la plantación; se observa además que, los datos están intercalados en el Bridge, tanto del Nodo 1 con su id=1 o id=2 para el nodo 2, también el identificador es el → *sensor/station1* o → *sensor/station2*.


*Figura 86. Datos reales de Nodo 1 y Nodo 2 en el Fertirriego.*
*Fuente: Autor.*

Cuando se están enviando los datos se procede a ingresar a la Tabla de DynamoDB en AWS denominada “Tesis” y se pueden observar los datos obtenidos del Nodo 1 con id=1 y el Nodo 2 con el id=2, para el Nodo 1 solo se tiene el valor de pH y para el Nodo 2 se tiene un nuevo valor de pH y el valor de Conductividad Eléctrica.

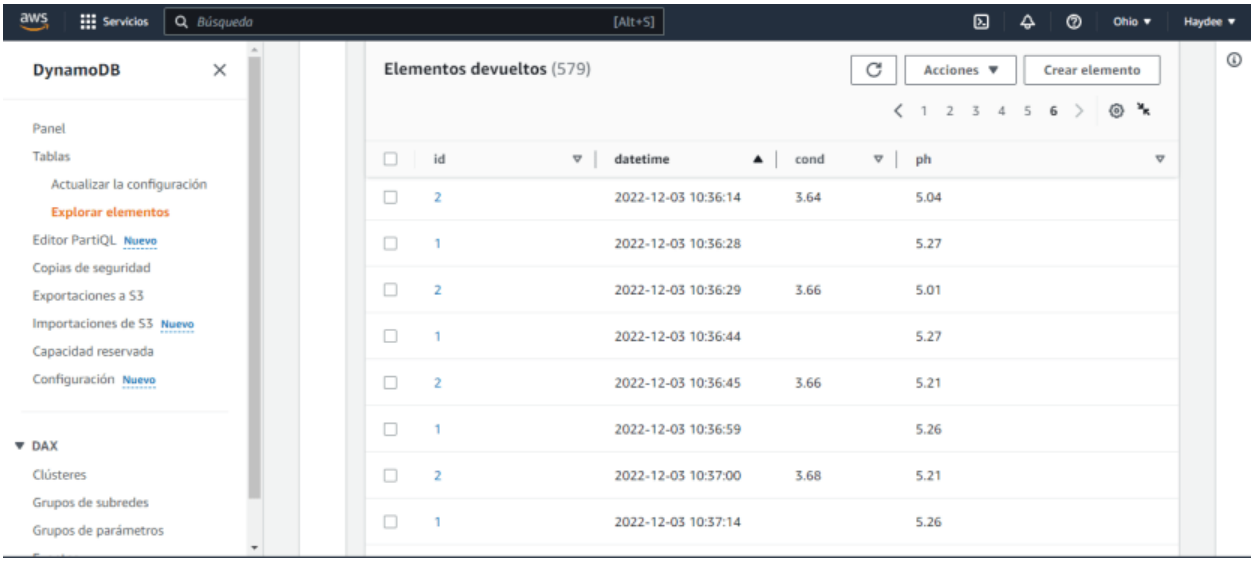
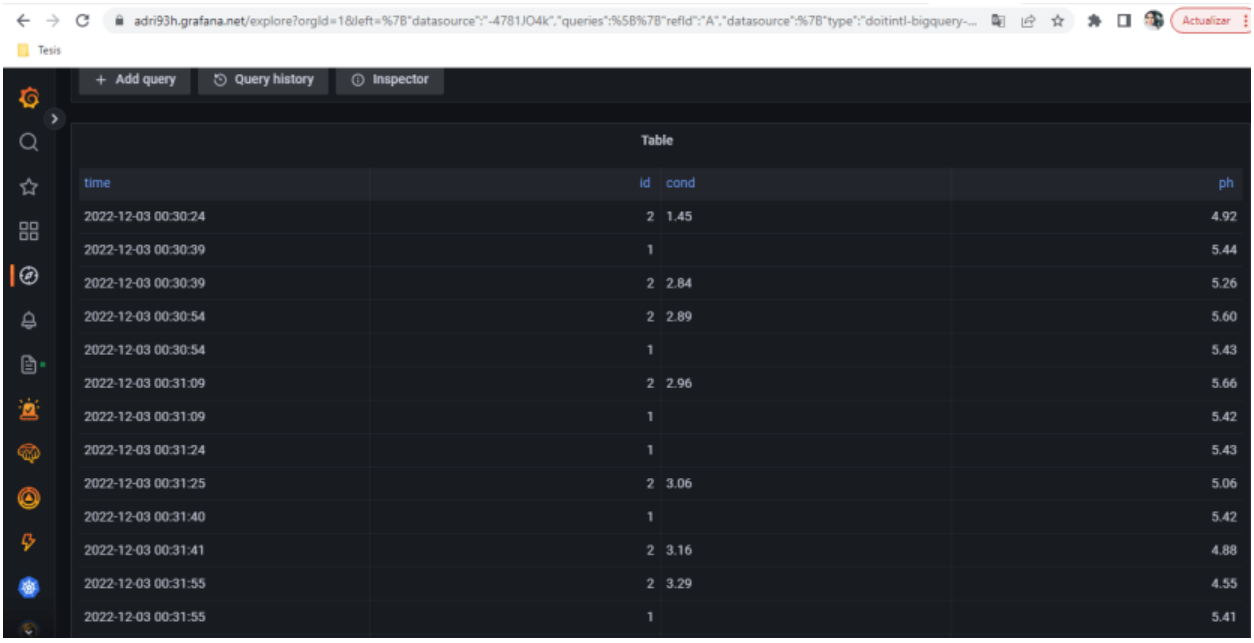



*Figura 87. Datos de Fertirriego obtenidos en la Tabla de DynamoDB en AWS.*
*Fuente: Autor.*

En Panoply es necesario ingresar hacia las tablas y volver a cargarlas para que se hayan tomado los datos desde DynamoDB hacia Panoply para nuevamente pasar hacia Grafana.

Finalmente, se puede ingresar a Grafana donde se encuentran reflejados los datos de id, tanto el 1 como el 2, y los valores de sensores para ph y conductividad, los valores son los tomados de la plantación en las zonas antes descritas para la colocación de cada Nodo.




| time | id | cond | ph |
|---|---|---|---|
| 2022-12-03 00:30:24 | 2 | 1.45 | 4.92 |
| 2022-12-03 00:30:39 | 1 | | 5.44 |
| 2022-12-03 00:30:39 | 2 | 2.84 | 5.26 |
| 2022-12-03 00:30:54 | 2 | 2.89 | 5.60 |
| 2022-12-03 00:30:54 | 1 | | 5.43 |
| 2022-12-03 00:31:09 | 2 | 2.96 | 5.66 |
| 2022-12-03 00:31:09 | 1 | | 5.42 |
| 2022-12-03 00:31:24 | 1 | | 5.43 |
| 2022-12-03 00:31:25 | 2 | 3.06 | 5.06 |
| 2022-12-03 00:31:40 | 1 | | 5.42 |
| 2022-12-03 00:31:41 | 2 | 3.16 | 4.88 |
| 2022-12-03 00:31:55 | 2 | 3.29 | 4.55 |
| 2022-12-03 00:31:55 | 1 | | 5.41 |

*Figura 88. Datos de sensores reflejados en Grafana.*
*Fuente: Autor.*

Se puede tomar algunos gráficos desde Grafana en los cuales se indiquen las estadísticas de los datos desde que se empezaron a tomar dichos datos con pruebas caseras hasta el día de las pruebas en el lugar de la plantación de Arándanos con los datos reales obtenidos, por ejemplo, la figura siguiente a manera de líneas apiladas indica la relación entre las fechas (mes y día) en la parte inferior y los valores en la parte lateral izquierda, con color celeste se indican los valores de ph y con color amarillo los valores de conductividad eléctrica, el id del nodo se indica con color verde como se indica a continuación.

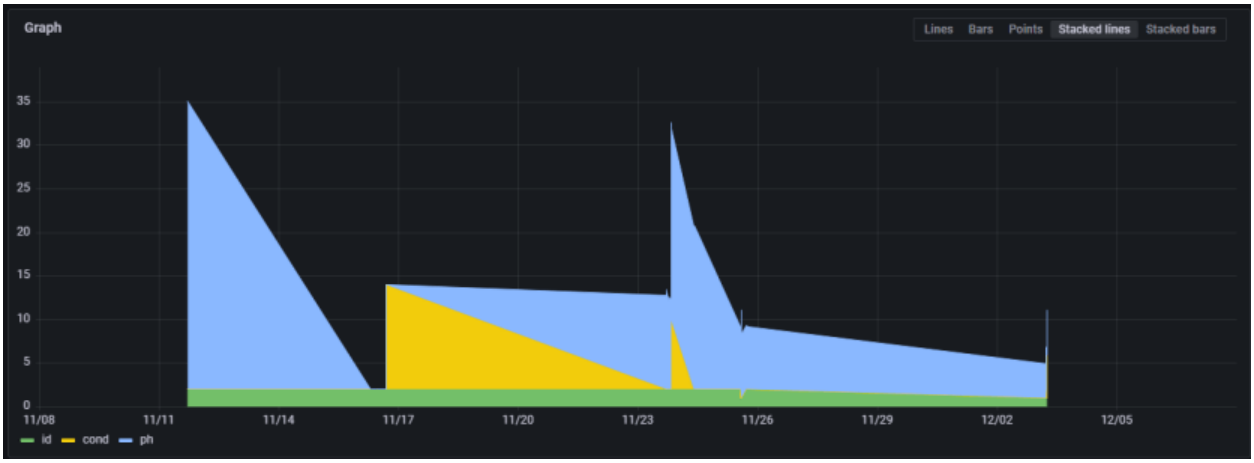



*Figura 89. Líneas apiladas de estadísticas de datos en Grafana.*
*Fuente: Autor.*

Las barras también demuestran un tipo de estadísticas para los datos tomados en AWS y reflejados en Grafana como se puede observar en la Figura que se muestra a continuación, en la parte inferior se indican las fechas con mes y día de la toma de datos, además, si se acerca el cursor, mostrará la fecha completa, la hora de toma de toma de datos y el valor, en el ejemplo se visualiza un valor de ph de 3.98; en la parte lateral izquierda se muestran los valores de los datos de medición de acuerdo con los sensores, con color celeste se identifica a los sensores de ph mientras que, con color amarillo al sensor de conductividad eléctrica.


*Figura 90. Barras de estadísticas de datos en Grafana.*
*Fuente: Autor.*

Gracias a todas las pruebas realizadas, se logró verificar que el sistema implementado para el monitoreo de fertirriego funciona perfectamente y en tiempo real, las distancias a las que se colocaron los nodos funcionan perfectamente, por lo tanto, se puede verificar en el momento exacto los valores de cada uno de los factores que intervienen en el proceso de fertirriego, en este caso la medida de pH en el primer nodo que si se encuentra en un rango de 5 aproximadamente, en el segundo nodo después de la mezcla de agua, nitratos, sulfatos y ácidos, el pH marca los valores adecuados que son entre 4,5 y 5,5; por otro lado, el valor de conductividad eléctrica indica los valores adecuados entre 1,25ppm hasta un máximo de 3ppm, con esto se verifica que el sistema funciona adecuadamente y al poder controlar mediante el monitoreo los valores correctos en el proceso de fertirriego, de esta manera se evita la pérdida del cultivo por exceso de nutrientes o por valores que no deben ser los adecuados para la planta.

Para comprobar esto, la dueña de la plantación “Berry Cute – Arándano Fresco” la Ingeniera Claudia López, proporcionó una carta indicando que el sistema implementado funciona de la manera adecuada, esto se muestra en el ANEXO 10.

# CAPÍTULO IV

## RESULTADOS

Este capítulo indicará los Resultados obtenidos de las pruebas que se aplicaron con anterioridad ya en la plantación, entre ellas están los análisis de Paquetes de MQTT-SN, las tomas de datos de RSSI a diferentes distancias en la plantación, así como los paquetes transmitidos, recibidos y perdidos de acuerdo a las distancias probadas, de esta manera se podrá comprobar la eficiencia de la Red creada de acuerdo con sus características, parámetros de red y modulación.

### 4.1.1. RSSI (Indicador de Fuerza de la Señal Recibida), Paquetes enviados, recibidos y perdidos

Otra prueba muy importante que se realizó fue la del RSSI, la toma de este dato se realizó mediante un ping establecido desde el Router de Borde hacia los Nodos que previamente habían sido ubicados. Las pruebas se realizaron moviendo el Router de Borde a varias distancias para identificar cuando se podía perder la comunicación y con qué RSSI se estaban enviando los datos con la modulación escogida MR-FSK y con la aplicación del estándar IEEE 802.15.4g.

El Nodo 2 hizo su primera prueba de RSSI a un metro de distancia entre el Router de Borde y el Nodo, en este punto el tiempo promedio es de 34 ms que está en referencia con el Nodo 1, el RSSI aquí es de -46, esta es conocida como una seña idónea que cuenta con tasas de transferencia estables sin pérdida de paquetes; además, se puede observar que los paquetes recibidos fueron los mismos que los enviados, por tal motivo no existe pérdida alguna de datos.

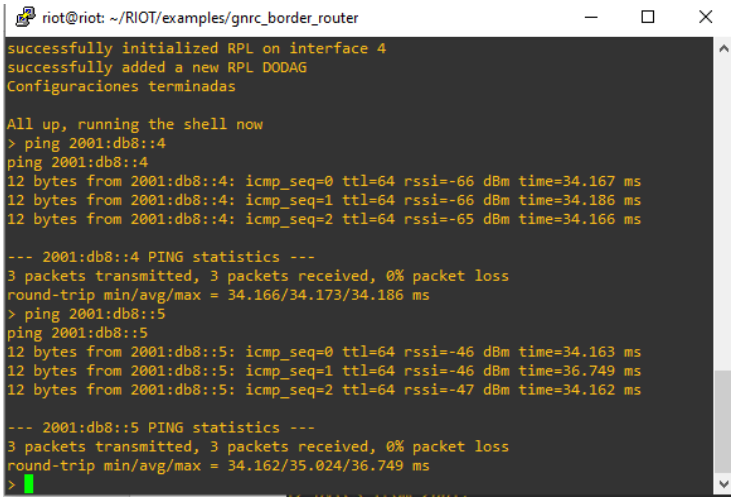



*Figura 91. RSSI hacia el Nodo 2 a un metro del Router de Borde.*
*Fuente: Autor.*

El primer ping de conexión del Nodo 1 se realizó a los 10 metros del Router de Borde, recordemos que este nodo estuvo dentro de la caseta del Centro de Control donde se encontraba el pozo o tanque que conservaba el agua antes de ser mezclada con los aditivos para el Fertirriego; el nodo 1 es la dirección 2001:db8::4 y se realiza el respectivo ping desde el Router de Borde, se puede ver que no existe ningún paquete perdido en relación con los paquetes enviados y que el promedio del rssi es de -65, este enlace es bastante bueno, la transmisión puede estar todavía al 90% pero no se descarta la posibilidad de que esté casi al 100%.

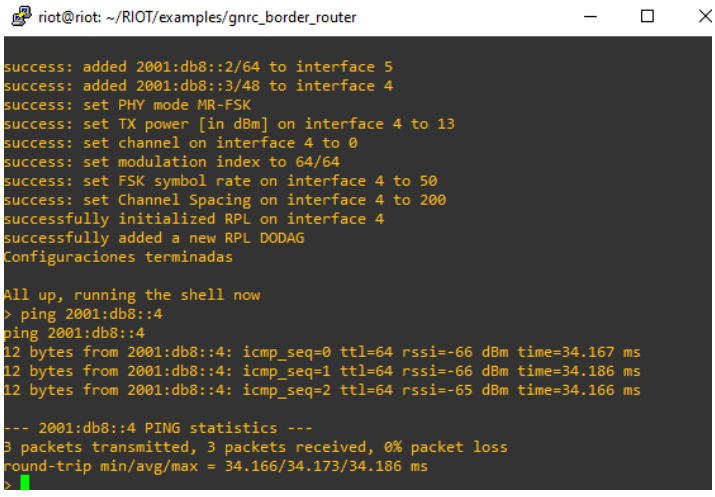



*Figura 92. RSSI hacia nodo 1 a los 10 metros del Router de Borde.*
*Fuente: Autor.*

Esta primera etapa de toma de RSSI del Nodo 1 a los 10 metros del Router de Borde y el Nodo 2 a un metro del Router de Borde se puede evidenciar mediante la siguiente imagen tomada en la plantación.

*Figura 93. Toma de datos de RSSI a un metro del Nodo 2 desde el Router de Borde y a 10 metros del Nodo 1.*
*Fuente: Autor.*

La segunda toma de RSSI fue a los 50 metros aproximadamente del Nodo 1, en este punto el promedio es de -71, esto quiere decir que está en el inicio de un enlace medio-bajo, puede ser una señal bastante buena o medianamente buena que, con la existencia de lluvia o polvo podría tener problemas; sin embargo, se tiene que no existen pérdidas de ningún paquete, es decir, lo que se envía es lo mismo que se recibe, no existe ninguna alteración de paquetes.

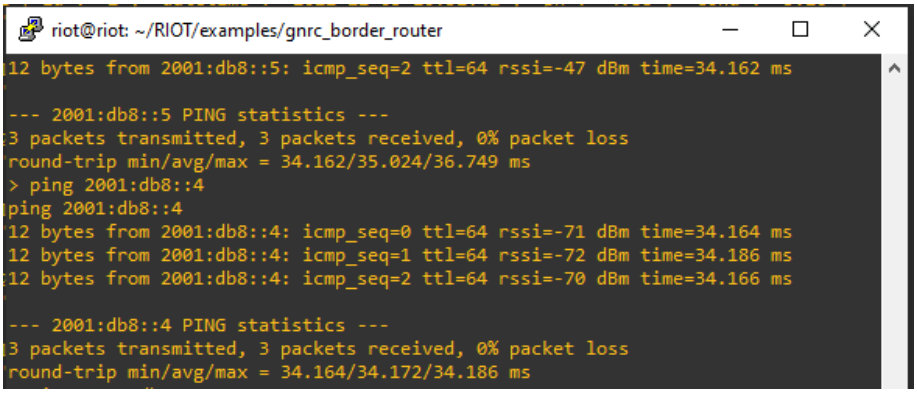



*Figura 94. RSSI hacia nodo 1 a los 50 metros del Router de Borde.*
*Fuente: Autor.*

A los 50 metros del Router de Borde y tomando el RSSI hacia el Nodo 2, se observa que este no tiene paquetes perdidos pero los tiempos varían considerablemente, tanto así que el tiempo más bajo es de 34 ms y el más alto de 205 ms, el rssi está en un promedio de -84, esta nuevamente es una señal mínima aceptable, puede encontrarse estable, pero podría sufrir alteraciones en su comunicación.

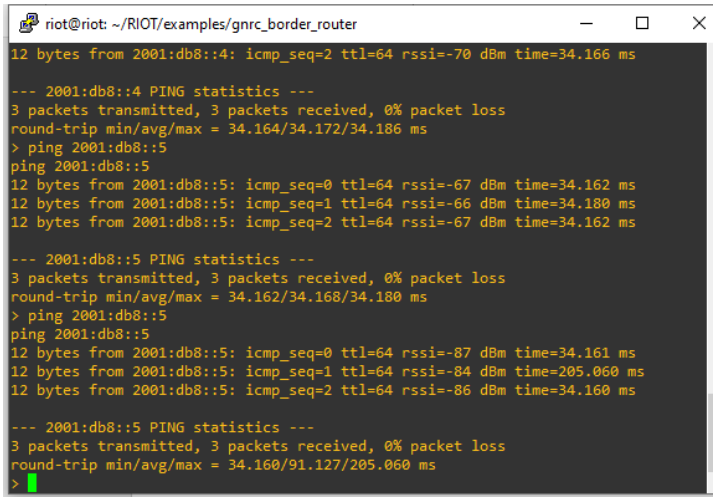



*Figura 95. RSSI hacia el nodo 2 a los 50 metros del Router de Borde.*
*Fuente: Autor.*

La toma de los datos se evidencia en la imagen siguiente.

*Figura 96. Toma de datos de RSSI desde Router de Borde hacia nodos 1 y 2 a los 50 metros.*
*Fuente: Autor.*

El siguiente es un RSSI a los 100 metros, es decir, aún dentro de la zona de la plantación, en este punto ya se observa que pueden haber ciertas pérdidas de paquetes pero el enlace una vez establecido se puede estabilizar y no existen pérdidas, el tiempo se sigue manteniendo muy similar a los anteriores pero el rssi es de -85, este tipo de señal está conocido como mínimo aceptable para una transmisión ya que podrían haber cortes de transmisión o pérdida de datos, sin embargo, en la experiencia que se tuvo en el caso, solamente una vez se perdieron paquetes pero el resto llegaron completos.

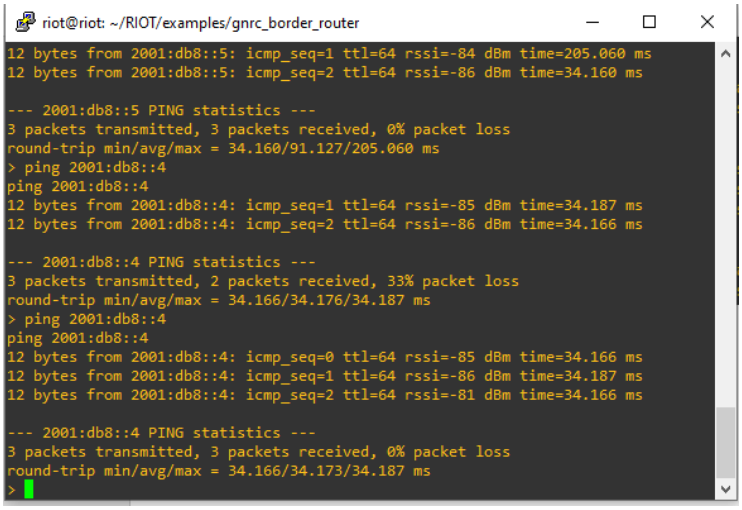



*Figura 97. RSSI hacia el Nodo 1 a los 100 metros del Router de Borde.*
*Fuente: Autor.*

A los mismos 100 metros se realiza una comprobación de red con el nodo 2 mediante un ping, en el que el rssi marca que tiene una potencia de -84 dBm con un tiempo similar al que se veía trabajando de 34 ms, esto prácticamente no varía, además, no existen pérdidas de paquetes, los mismos que se reciben son los que se enviaron.

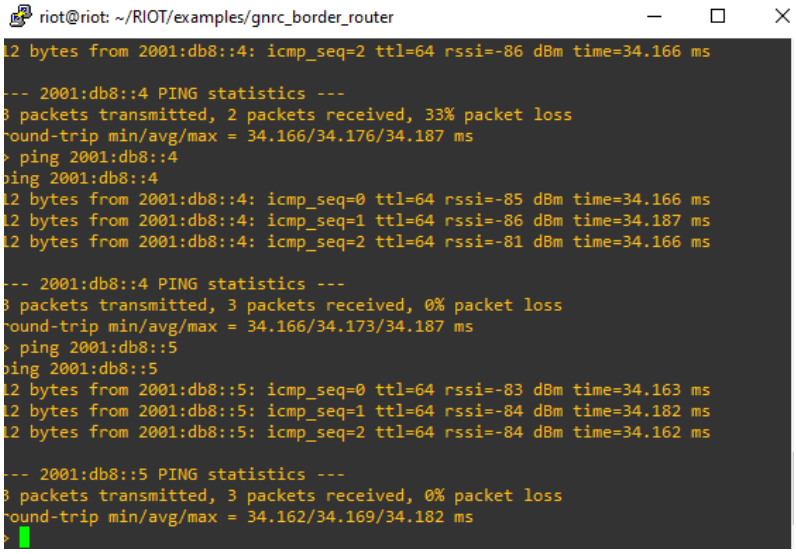



*Figura 98. RSSI hacia nodo 2 a los 100 metros del Router de Borde.*
*Fuente: Autor.*

La toma de datos en ese punto fue al final de la plantación para una mejor experiencia de red, desde este punto se encontraba ubicado el Router de Borde, se procedió a tomar una foto, misma que se encuentra a continuación.

*Figura 99. Toma de datos de RSSI a los 100 metros de los nodos desde Router de Borde.*
*Fuente: Autor.*

Una toma más del RSSI fue fuera de la producción de arándanos, en este punto se cuenta con aproximadamente 150 metros desde el Nodo 1 hacia el Router de Borde, aquí el RSSI cambia a un promedio de -78 dBm lo que significa que la señal es mucho mejor que la anterior,

pero, en este caso el tiempo de envío de datos varía entre 50, 30 y 90 ms, en un primer intento los datos se perdieron, pero luego el enlace se estabilizó y no existió pérdida alguna de paquetes.

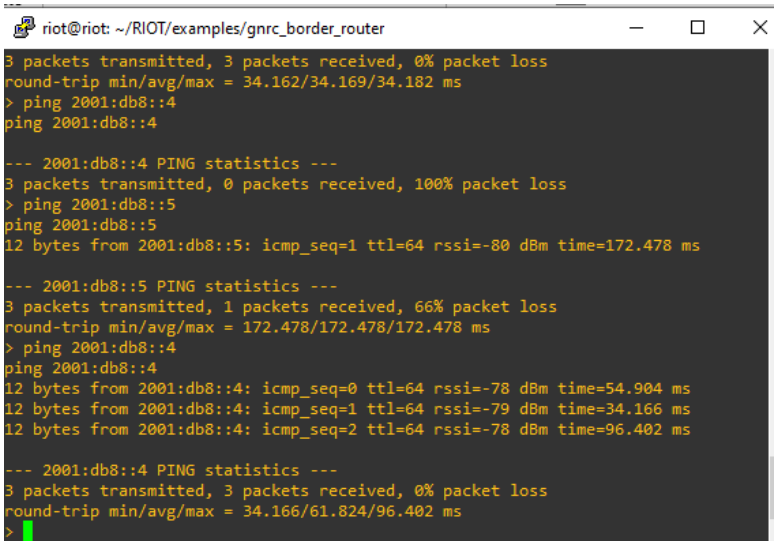



*Figura 100. RSSI hacia el Nodo 1 a los 150 metros del Router de Borde.*
*Fuente: Autor.*

A los 150 metros del Router de Borde, el RSSI para el nodo 2 estaría sobre los -81 dBm que como ya se sabe, es una señal muy poco aceptable, se perdieron algunos paquetes, pero el tiempo sigue siendo el mismo de puntos anteriores.

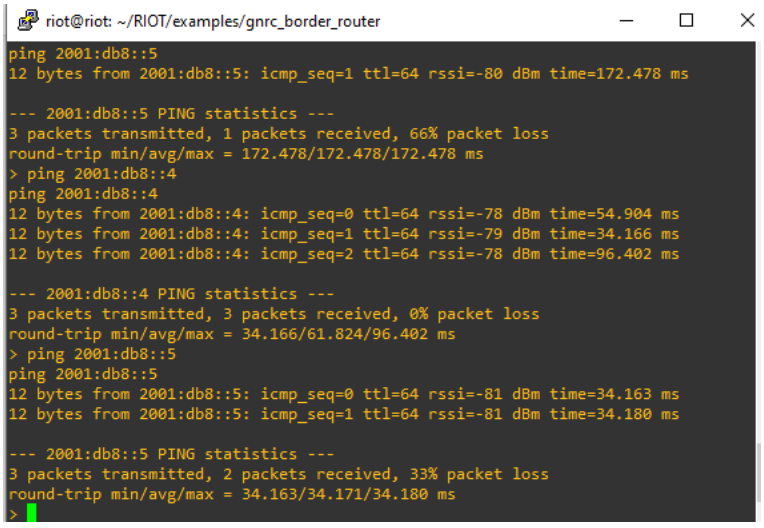



*Figura 101. RSSI de nodo 2 hacia Router de Borde a los 150 metros.*

*Fuente: Autor.*

Finalmente, los datos con más distancia se tomaron desde la parte de afuera de la zona cerrada del invernadero de arándanos, al final se observa a mi persona con el Router de Borde, enviando desde ahí un ping para verificar el RSSI a los aproximadamente 150 metros. La conexión se iba por momentos, pero luego se puso estable la comunicación y no hubo pérdida de paquetes.

*Figura 102. Toma de dato de RSSI desde los 150 metros de distancia entre los nodos 1 y 2 y el Router de Borde.*

*Fuente: Autor*.

Con todo lo mencionado anteriormente, la red está funcionando de la manera requerida, aproximadamente a los 150 o 180 metros podrían empezar algunas pérdidas de datos, sin embargo, el enlace se estabiliza y no existen pérdidas, los tiempos de envío son relativamente bajos, por ello se tiene la transmisión y recepción en tiempo real; el RSSI es bastante bueno para las condiciones establecidas, para el tamaño de producción es una red que proporciona una calidad excelente para el caso.

### 4.1.2. Análisis de Paquetes MQTT-SN en Broker Mosquitto

De acuerdo con el Protocolo MQTT-SN, los paquetes serán analizados de acuerdo con lo que dice el Protocolo y con los paquetes recibidos en el Broker Mosquitto que será el encargado de realizar dicho intercambio de paquetes.

Los clientes inician una comunicación MQTT-SN sobre UDP con el Broker mosquitto y el primer mensaje es el → CONNECT, este mensaje contiene información sobre el usuario, contraseña, id.

El bróker mosquitto responde con un mensaje → CONNACK hacia el cliente con el resultado de la conexión, es decir, rechazada o aceptada.

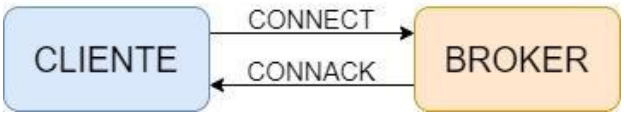



*Figura 103. Intercambio de mensajes MQTT-SN Connect y Connack.*
*Fuente: Autor.*

En el Broker Mosquitto estos mensajes se visualizan de la siguiente manera, la dirección 2001:db8::4 es la dirección del cliente o nodo1, es el nodo que tiene solamente 1 sensor, utiliza el puerto 1883 para transferir su mensaje Connect hacia el Broker mosquitto, el bróker mosquitto responde al cliente con un Connack para aceptar la conexión.

```
20221125 185758.941 3 2001:db8::4:1883  <- MQTT-S CONNECT cleansession: 1
20221125 185758.941 -1822926896 2001:db8::4:1883 SENSOR1 -> MQTT-S CONNACK returncode 0 (0)
20221125 185758.941 CWNAN0000I Client connected to udp port 1885 from SENSOR1 (2001:db8::4:1883)
```

*Figura 104. Mensajes CONNECT y CONNACK visualizados en Broker Mosquitto.*
*Fuente: Autor.*

Los mensajes siguientes son → *REGISTER* y → *REGACK*, el Register es el mensaje que envía el Cliente hacia el Broker Mosquitto para ya registrar ya la conexión y posteriormente el Regack es enviado desde el Broker como respuesta al cliente de la aceptación de la comunicación.

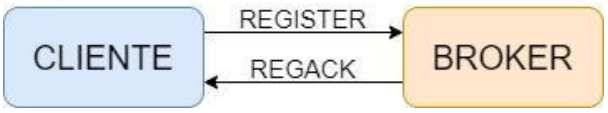



*Figura 105. Intercambio de mensajes MQTT-SN Register y Regack.*
*Fuente: Autor.*

De igual manera, se tiene la dirección 2001:db8::4 con el puerto 1883, estos son los datos del cliente que tiene conectado el Nodo 1 que se llama → SENSOR1, ahí se envía en esa dirección el Register, luego el Regack se envía desde el Broker hacia el Mosquitto como se muestra en la figura siguiente.


*Figura 106. Mensajes REGISTER y REGACK visualizados en Broker Mosquitto.*
*Fuente: Autor.*

A continuación, se tiene el mensaje de PUBLISH, este mensaje va desde el cliente hacia el Broker Mosquitto y es el que lleva los datos tomados por los sensores para posteriormente ser enviados al bróker.


*Figura 107. Intercambio de mensaje MQTT-SN Publish.*
*Fuente: Autor.*

En el Broker Mosquitto este mensaje se visualiza como se observa a continuación, la dirección es la misma 2001:db8::4 con el puerto 1883 y va desde el cliente hacia el Broker para poder publicar los datos.

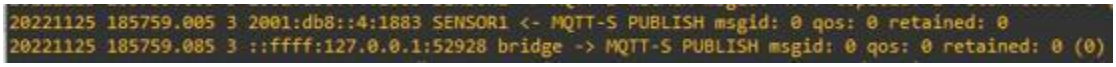



*Figura 108. Mensaje PUBLISH visualizado en Broker Mosquitto.*
*Fuente: Autor.*

Finalmente, se tiene un par más de mensajes, en este caso son los de DISCONNECT, este mensaje existe en los dos lados, después de que el cliente haya publicado los datos leídos,

procede a enviar su mensaje de desconexión del Broker, éste le responde con otro mensaje de DISCONNECT, el intercambio se da de la siguiente manera.

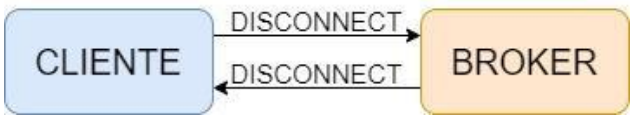



*Figura 109. Intercambio de mensajes MQTT-SN Disconnect.*
*Fuente: Autor.*

El Broker Mosquitto visualizará los mensajes DISCONNECT como se muestra en la figura, el primer mensaje va desde el cliente hacia el broker y el segundo va desde el broker hacia el cliente para completar la transmisión de datos y volver al inicio del ciclo de conexión.


*Figura 110. Mensajes DISCONNECT visualizados en Broker Mosquitto.*
*Fuente: Autor.*

Además, cabe mencionar que el Puente Transparente también se suscribe al Broker mediante los paquetes de SUBSCRIBE que va desde el puente hacia el Broker para indicar que desea establecer comunicación para suscribirse a los topic y recibir los datos que a este llegan, luego, el broker le responde con un paquete de SUBACK para aceptar su petición y poder enviarle los datos que reciba, esto se puede comprobar mediante la siguiente captura tomada en el Broker Mosquitto.

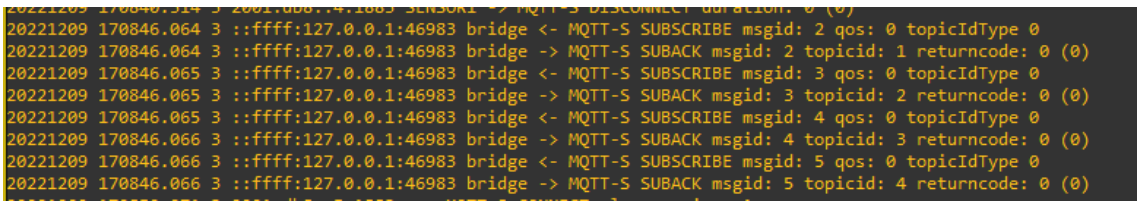



*Figura 111. Suscripción del Puente Transparente hacia el Broker.*
*Fuente: Autor.*

### 4.1.3. Análisis Wireshark de paquetes MQTT-SN

Anteriormente, se ha indicado el intercambio de mensajes visualizados desde el Broker Mosquitto, para seguir con la explicación, se realizó una captura de Wireshark de la interfaz Tap5 que es el puente creado con el Router de Borde. Una vez capturados los paquetes se procede a realizar el análisis teniendo en cuenta la captura del broker y el sentido de los mensajes capturados por el Sniffer.

Primeramente, cabe mencionar que los mensajes MQTT-SN capturados son los encargados de enviar los datos recolectados de los sensores, estos serán enviados hacia el Router de Borde siguiendo el proceso de envío de datos.

A continuación, se coloca una captura de los mensajes que han sido visualizados en el Broker mosquitto, aquí, cada uno tendrá un número para identificarlos más adelante con cada una de las capturas de Wireshark, todos los paquetes de MQTT-SN son transmitidos sobre el protocolo UDP como se visualiza.

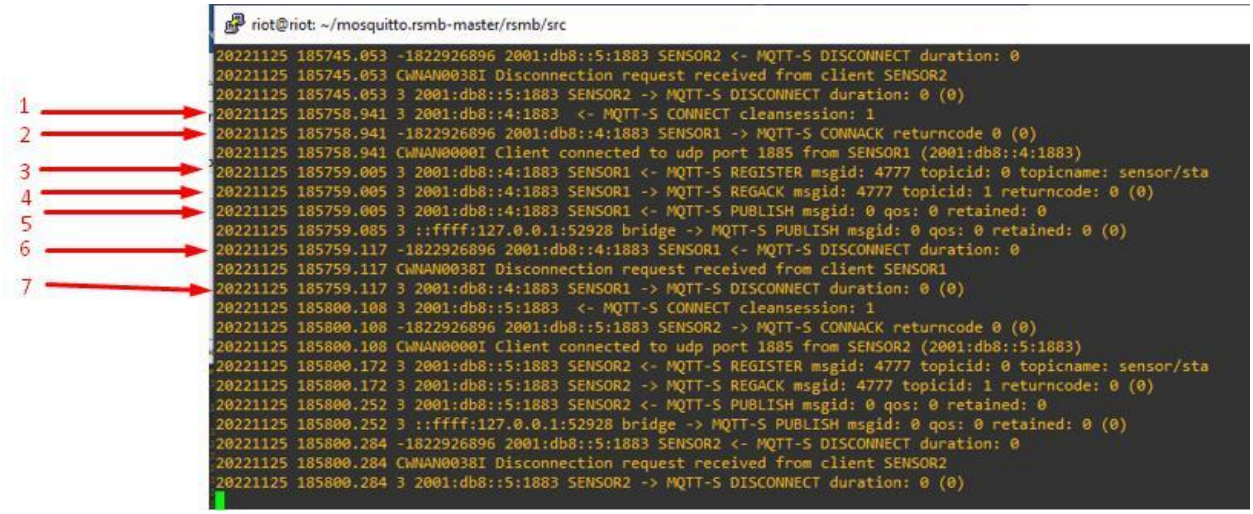



*Figura 112. Señalización en broker de mensajes MQTT-SN recibidos.*
*Fuente: Autor.*

Una vez identificados los mensajes que se van a analizar, es necesario dirigirse hacia las capturas de Wireshark para proseguir, la primera captura hace referencia al número 1 de la Figura anterior, es decir al CONNECT, mismo que se da desde el cliente hacia el mosquitto para indicar que se quiere conectar, en el Data en este caso, se indica que es el SENSOR1 quien se quiere conectar, de igual manera se observan las direcciones entre las cuales se establece, es decir, desde la 2001:db8::4 que es el cliente hacia la 2001:db8::1 que es el broker.

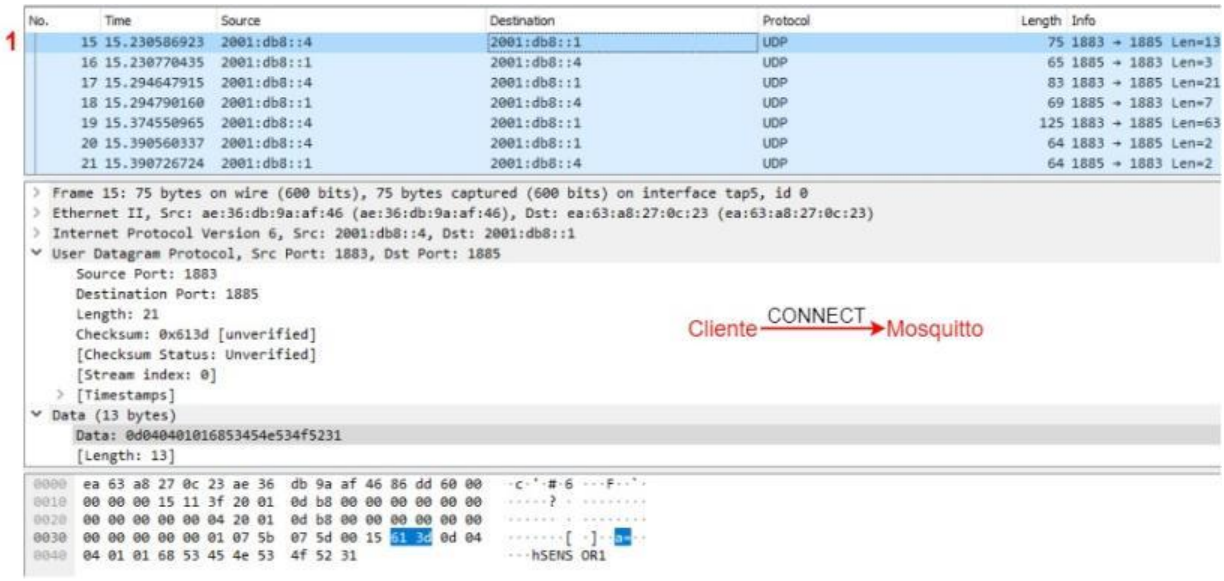



*Figura 113. Paquete CONNECT en Wireshark.*
*Fuente: Autor.*

El segundo paquete es el CONNACK que va en respuesta al CONNECT, en este caso este mensaje va desde el broker con IP 2001:db8::1 hacia el cliente que es la 2001:db8::4 para decirle que si puede establecer conexión con él. El puerto 1885 es el encargado de escuchar los mensajes MQTT-SN sobre UDP y el puerto 1883 recibirá en este caso el paquete; este paquete tiene una longitud de 3 bytes.

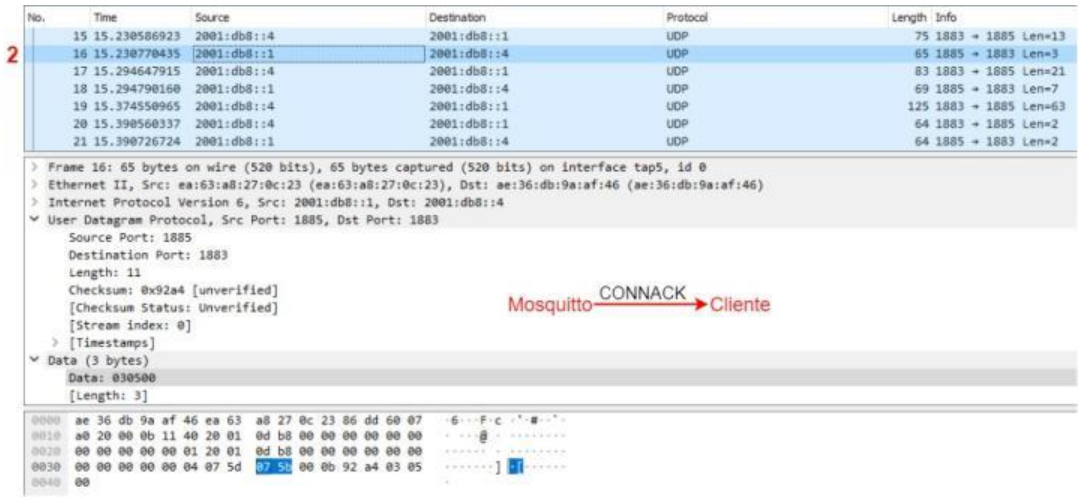



*Figura 114. Paquete CONNACK en Wireshark.*
*Fuente: Autor.*

El paquete siguiente tiene una longitud de 21 bytes y es el REGISTER, en este paquete se envía el nombre de quien va a publicar, en este caso se trata de → sensor/station1 para hacer referencia al nodo 1, es decir, se manifiesta el registro mediante la interfaz tap5, el sentido de este mensaje es desde el cliente hacia el broker mosquitto y el protocolo utilizado para enviar este mensaje MQTT-SN es mediante UDP como se ha indicado en casos anteriores, el intercambio de puertos es el mismo en todos los casos.

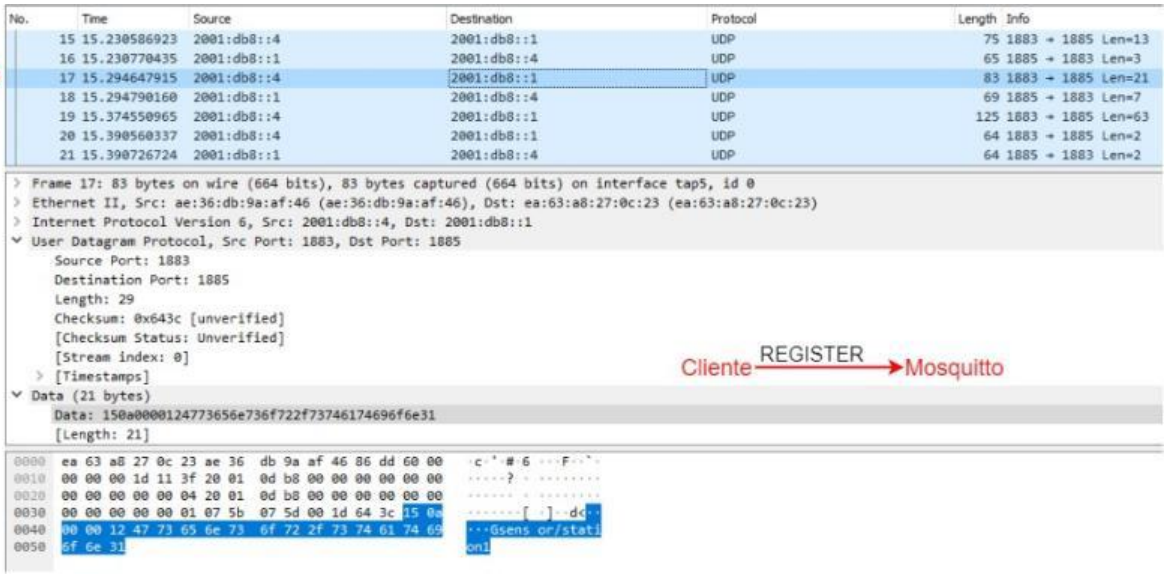



*Figura 115. Paquete REGISTER en Wireshark.*
*Fuente: Autor.*

Como respuesta al paquete REGISTER, se tiene un REGACK, este hace referencia a que acepta el registro del nodo que quiere conectarse para transmitir, por lo tanto, envía este mensaje como un ACK, este paquete va de la dirección del broker mosquitto hacia el cliente que tiene la dirección 2001:db8::4 por el protocolo UDP para enviar un paquete MQT-SN, la longitud es de 7 bytes y el puerto de origen es el 1883 y el de destino es el 1885.

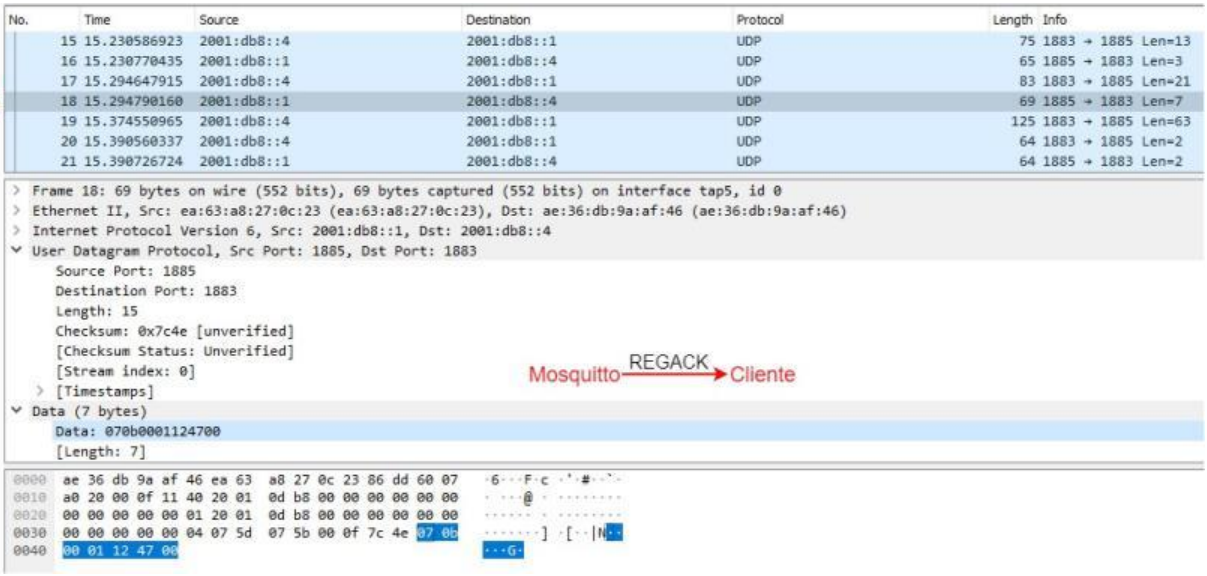



*Figura 116. Paquete REGACK en Wireshark.*
*Fuente: Autor.*

El quinto paquete es el que lleva ya los datos a publicar, se trata del PUBLISH, este contiene todos los datos que van a ser enviados hacia el broker, por ello, el sentido es desde el Cliente hacia el broker mosquitto para enviar los datos obtenidos de los sensores junto con el id que es el identificador, en este caso es el 1, también envía el datetime que contiene la fecha y ahora, además, enviará el valor de ph para el caso del nodo que contiene un sensor. Cabe mencionar que envía todos estos datos ya que está formado directamente el json para ser enviado una vez que haya obtenido el dato del sensor en cuestión. La longitud de los datos es de 63 bytes y de igual manera se manejan los puertos 1883 y 1885 para destino y origen respectivamente.

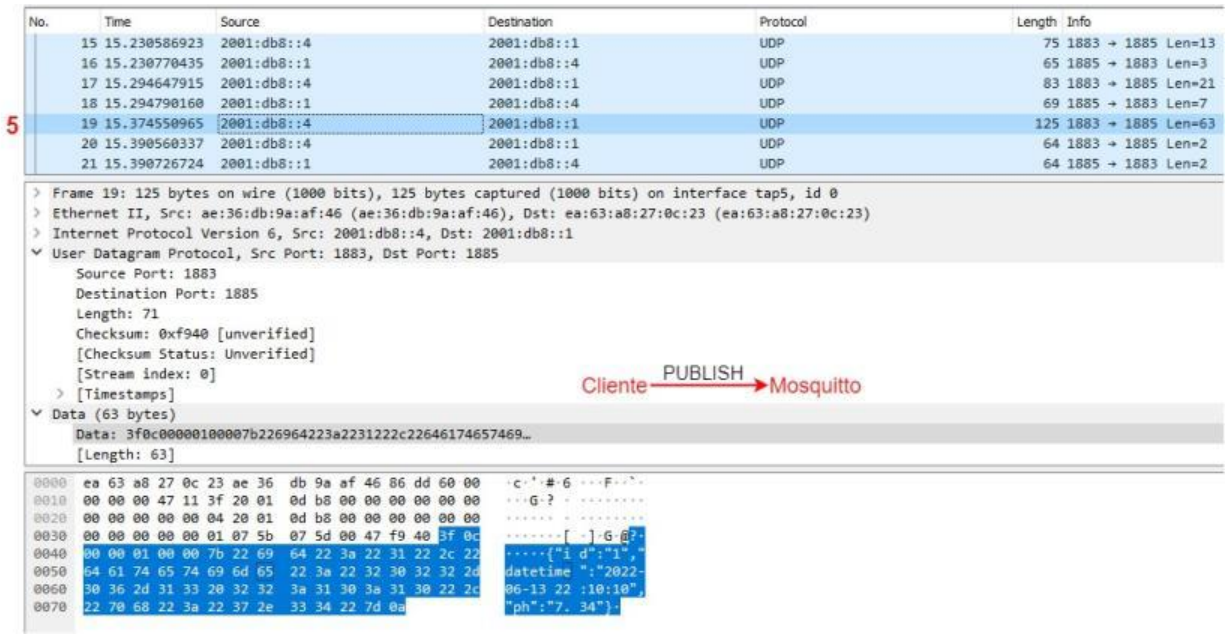



*Figura 117. Paquete PUBLISH en Wireshark.*
*Fuente: Autor.*

Una vez que el cliente haya enviado los datos, inmediatamente envía un paquete de DISCONNECT, de esta forma pide que se finalice la transmisión de los datos para comenzar un nuevo ciclo, la longitud de los datos enviados ahora será de 2 bytes, la dirección es desde el cliente hacia el broker, el puerto de origen es el 1883 y el de destino es el 1885.

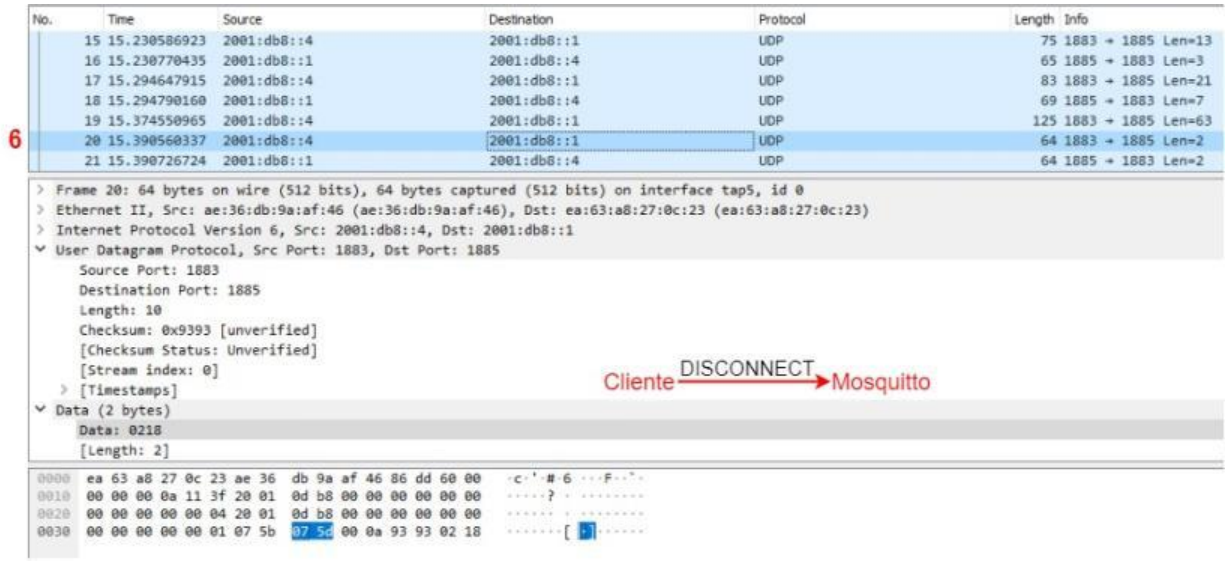



*Figura 118. Paquete DISCONNECT desde el cliente hacia el Broker en Wireshark.*
*Fuente: Autor.*

Finalmente, el Broker Mosquitto responde con otro DISCONNECT al cliente para aceptar que se desconecte y dar por finalizada dicha transmisión, lo siguiente será que todo empiece de nuevo con un mensaje de Connect. De la misma forma el tamaño es de 2 bytes para este paquete.

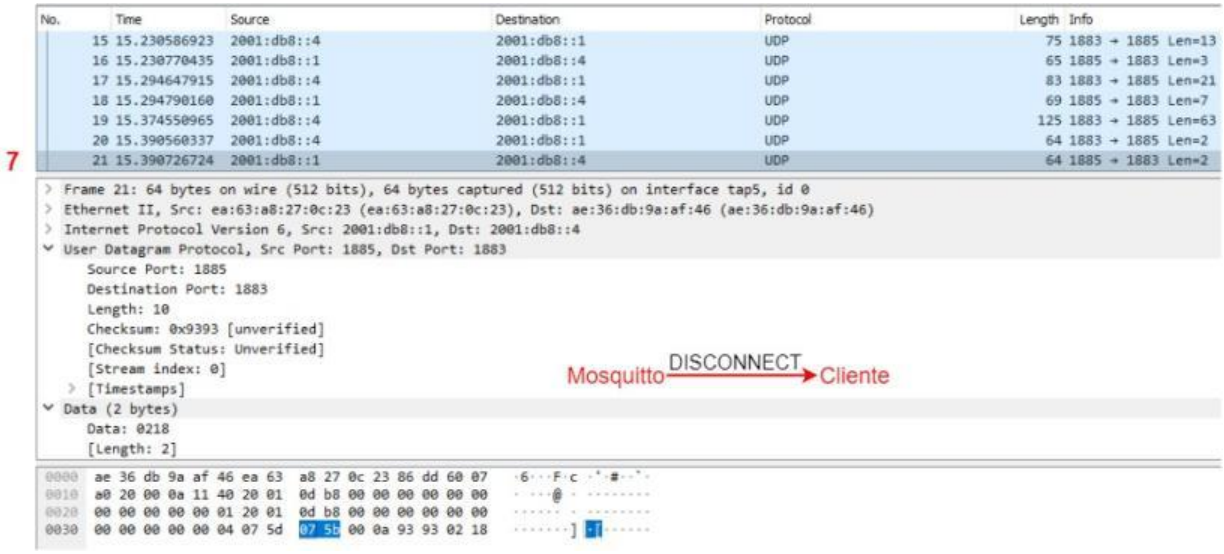



*Figura 119. Paquete DISCONNECT desde el Broker hacia el Cliente en Wireshark.*
*Fuente: Autor.*

### 4.1.4. Análisis Señales

En la imagen que se muestra a continuación, se tiene un muestreo IQ o también conocido como muestreo complejo o muestreo de cuadratura, se puede observar todo el espectro dentro de la modulación FSK, esta tiene un ancho de banda de 200 kHz, se puede también observar el pico más alto de la señal transmitida en el intercambio de paquetes con IEEE 802.15.4g.

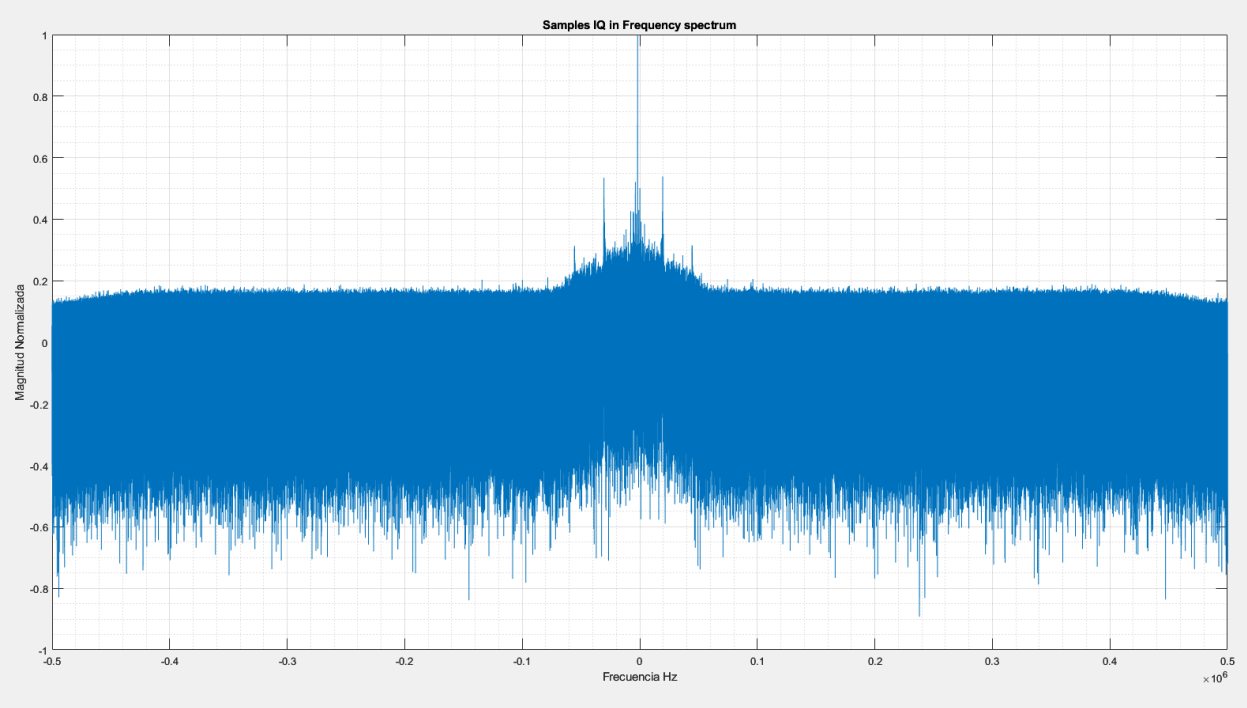



*Figura 120. Espectro completo de modulación FSK.*

*Fuente: Autor.*

En la figura 121, se tiene una muestra específica de la señal, la primera parte de la imagen se trata de la señal en el dominio del tiempo, es decir, se indica la duración del pulso de la señal para el envío de datos; en la segunda parte de la imagen es la señal pero en el dominio de la frecuencia, en este caso se tiene presente el ancho de banda que es de 200 KHz así como la frecuencia central a la que trabaja que es 863,1 MHz; en la parte final de la imagen se tiene el espectrograma, este se encarga de dejar ver la señal en tiempo y frecuencia.

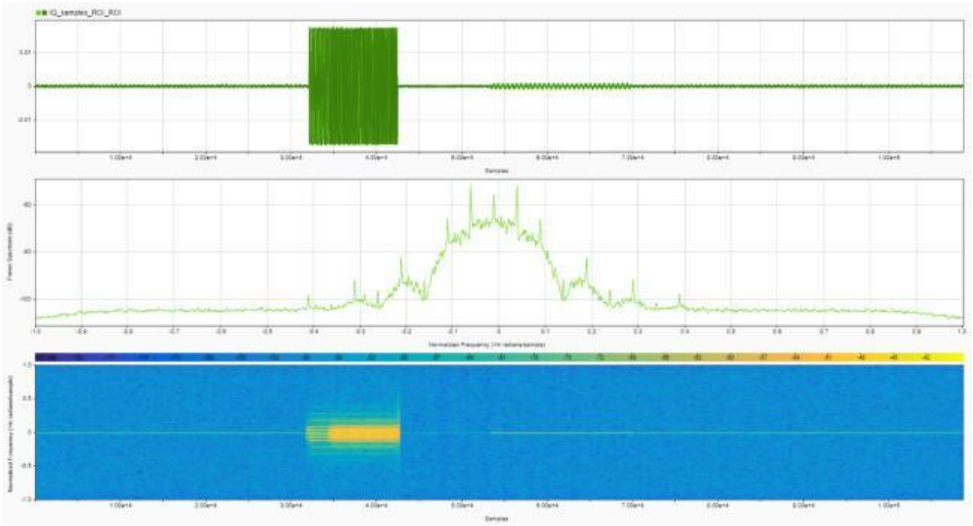


*Figura 121. Señal con zoom en diferentes dominios.*
*Fuente: Autor.*

En la figura 122 se repiten los aspectos tratados en la Figura anterior, con la diferencia que aquí se visualiza, pero de forma más espaciada o más lejana, por ejemplo, los pulsos se muestran en esta ocasión dos en la primera parte de la figura, en la segunda parte nuevamente está la señal en el dominio de la frecuencia indicando su AB y, finalmente, el espectro completo alejado de la señal de modulación y datos enviados.

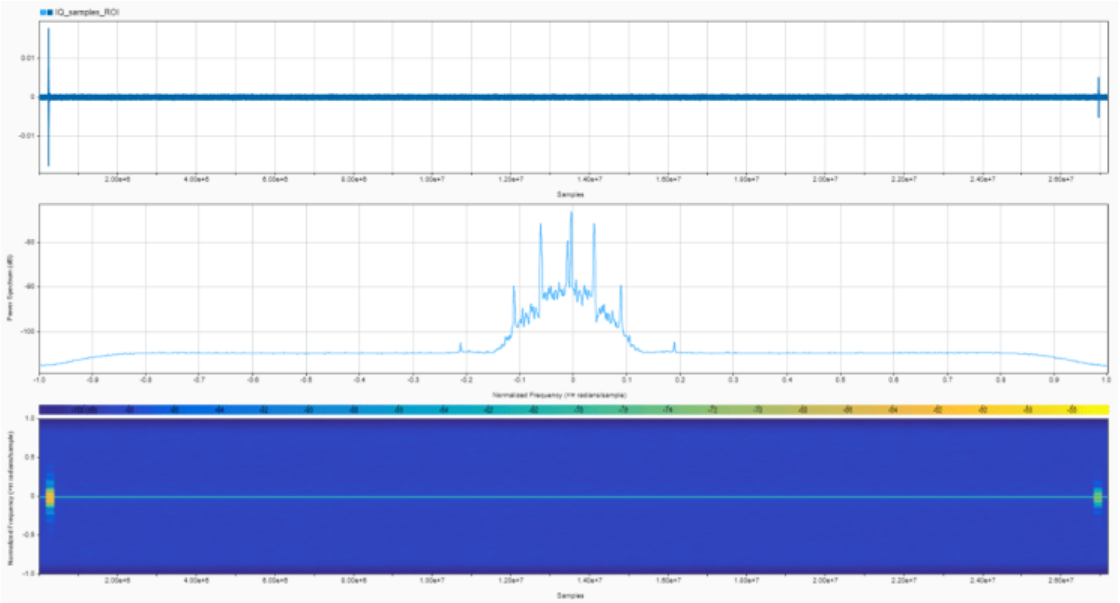


*Figura 122. Señal sin zoom en diferentes dominios.*
*Fuente: Autor.*

# CONCLUSIONES Y RECOMENDACIONES

En este apartado se darán a conocer las conclusiones y recomendaciones que se pueden brindar de acuerdo al tema de Titulación desarrollado en este documento, estas se expondrán en el listado a continuación.

## Conclusiones

Gracias a la implementación del sistema en “Berry Cute” utilizando IEEE 802.15.4g y a todas las pruebas realizadas, se pudo verificar que el monitoreo de Fertirriego funciona perfectamente ya que, mediante las medidas tomadas por los sensores sobre pH y CE en tiempo real se demostró que la dosificación de las sustancias para el fertirriego son las correctas, por tal motivo, se evita la quema de las plantas y, por ende, la pérdida del cultivo.

Basándose en el estándar al que hace referencia el proyecto, la modulación FSK fue la idónea para el desarrollo ya que el sistema funciona perfectamente para las condiciones de la plantación por su distancia e implementación más sencilla.

Las pruebas realizadas fueron en varias distancias, la más lejana fue de 150 metros aproximadamente, donde se comprobó que el enlace nunca se perdió, sin embargo, la distancia de operación entre los diferentes nodos fue de menos de 100 metros, en los que se verificó que los paquetes en el monitoreo de fertirriego nunca se perdieron y la señal era la idónea para la transmisión de datos.

La plataforma en la nube utilizada para este proyecto es acertada gracias a que AWS permite una gran cantidad de almacenamiento en su base de datos, a su vez, AWS permitió la integración con Grafana que fue la aplicación con la que el usuario estuvo al tanto del monitoreo de su cultivo de arándanos.

**Recomendaciones**

Se recomienda la implementación del Sistema de Monitoreo de Fertirriego creado con IEEE 802.15.4g y utilizando la modulación FSK para los diferentes tipos de cultivos que necesiten un monitoreo constante, en tiempo real y con un costo relativamente bajo, esto ayudará para que no exista pérdida de cultivos por falta de monitoreo en los momentos adecuados.

Para que la calidad de la red y rendimiento de la misma sean idóneas para la plantación, es recomendable realizar varias pruebas de conexión y envío de datos en diferentes distancias y medir el RSSI, esto permitirá identificar el mejor lugar para colocar los nodos y el Router de Borde para un monitoreo correcto.

Para que la integración de DynamoDB de AWS con Grafana sea más sencilla, se recomienda utilizar un intermediario como lo es Panoply que actuará como un puente entre los softwares para el traspaso de datos desde la base de datos hacia la plataforma con la que actuará el usuario final.

Es aconsejable utilizar un dispositivo extra para la obtención de los datos, en este caso un Arduino UNO, este permitirá que los datos sean leídos de manera correcta y enviados a OpenMote-B, si se toma los datos directamente desde el nodo podrían existir alteraciones en sus valores de lectura.

Se recomienda que los nodos con sensores estén en cajas herméticas que puedan cubrir las placas de sensores de lluvias y vientos, solamente las sondas para medición y la antena de los OpenMote-B deberán estar fuera de las cajas.

## BIBLIOGRAFÍA


Ante, G. A. (2019). *GAD Municipal Antonio Ante*. Obtenido de https://www.antonioante.gob.ec/AntonioAnte/index.php/canton/informacion-general/17-canton/98-chaltura

Arduino. (12 de 01 de 2023). *Docs Arduino.* Obtenido de https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf

Association, I. S. (2012). *IEEE 802.15.4 Low Rate-Wireless Personal Area Networks (LR-WPANs).* New York: The Institute of Electrical and Electronics.

AWS. (2022). *Amazon Web Services*. Obtenido de https://aws.amazon.com/es/what-is-aws/

Bárbaro, L., Karlanian, M., & Mata, D. (2018). *Importancia del pH y la Conductividad Eléctrica (CE) en los sustratos para plantas.* INTA.

Callahan, C. (2019). *Tecnología de poscosecha de productos hortícolas perecederos.*

CircuitSchools. (13 de 6 de 2020). *CircuitSchools Staff*. Obtenido de https://www.circuitschools.com/tds-meter-with-tds-sensor-and-arduino-for-water-quality-monitoring/

Cisco. (2011). *Cisco.*

García, P. S. (2020). Nutrición y Manejo Hidroónico de Arándano. México: Colegio de Postgraduados.

Gómez, F. (27 de 06 de 2020). Talleres para conocer la producción de arándanos. *El Heraldo*.

Grafana. (2022). *Grafana.* Obtenido de https://grafana.com/

Harada, H., Mizutani, K., & Fujiwara, J. (2017). *IEEE 802.15.4g Based Wi-SUN Communication Systems.* Tokio: Copyright.

IbrahiemEmary. (2017). *Wireless Sensor Network From Theory to Applications.* London: CRC Press.

IEEE. (2020). *IEEE Standar for Low-Rate Wireless Networks. IEEE 802.15.4.* New York: IEEE Computer Society.

ISO/IEC/IEEE29148/2018. (2018). *Systems and software engineering-Life cycle processes-Requirements engineering.* ISO.

ITU-T. (2015). *ITU-T.*

Liñán, A., Vives, Á., Bagula, A., Zennaro, M., & Pietrosemoli, E. (2015). *Internet de las cosas.*

Martí, M., García, C., & Campo, C. (2019). *Performance Evaluation of CoAP and MQTT-SN in an IoT Environment.* Madrid: Proceedings.

Mishra, B., & Kertesz, A. (2020). The use of MQTT in M2M and IoT Systems: A Survey. Szeged: IEEE Access.

Ochoa, C. (2021). Ecuarándano. *Inhaus.*

Oh, E.-S., Lee, S. H., & Hwang, S. H. (2015). *Comparison of SUN and Wi-Fi P2P WSN in M2M Environments.* Korea: Chunming Qiao.

Panoply. (22 de abril de 2021). *Panoply.* Obtenido de https://blog.panoply.io/how-to-analyze-dynamodb-data-in-grafana

Project Management Institute. (2017). Guía del PMBOK. Pennsylvania: Project Management Institute, Inc.

Remko. (23 de 11 de 2022). *Radio Mobile*. Obtenido de http://radiomobile.pe1mew.nl/

rest, A.-D. e. (s.f.). *AWS-DynamoDB encription at rest*. Obtenido de https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/EncryptionAtRest.html

Rilheva. (2019). *How to integrate the MQTT protocol into your Industrial IoT project.* Rilheva.

RIOT-OS. (Diciembre de 2018). *RIOT*. Obtenido de https://doc.riot-os.org/

Shields, I. (2019). *Industrial Shields*. Obtenido de https://www.industrialshields.com/es_ES/shop/product/is-omb-001-openmote-b-721#attr=

Simmons, A. (13 de 11 de 2022). *dgtlinfra*. Obtenido de https://dgtlinfra.com/internet-of-things-iot-architecture/

Soni, D., & Makwana, A. (2017). *A Survey on MQTT: A Protocol Of Internet of Things (IoT).* India.

STMicroelectronics. (2002). *HCF4051B Single 8-channel analog multiplexer/demultiplexer.* Italy: STMicroelectronics Group of Companies.

Yen, L. (2015). The room shortage problem of tree-based ZigBee/IEEE 802.15.4 wireless network. Taiwan.

Yuni, J., & Urbano, C. (2016). *Técnicas para Investigar.* Argentina: Editorial Brujas.

# ANEXOS

## ANEXO 1: Código Arduino para Nodo con 1 sensor para recopilar datos de sensores y enviar archivo tipo JSON a Router de Borde

```
/* Adriana Herdoíza
 * Trabajo de Grado
 * Nodo1 - Sensor pH
*/

//Código para calibración de sensor de pH
float calibration = 23.80; //valor de calibración
const int analogInPin = A0; //Pin análogo para lectura
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10], temp;
String valorpH;
char nodo1[256];
char lectura[6];

void setup() {
  Serial.begin(115200); //iniciar comunicación serial
}

float lecturaPH() {
  for (int i = 0; i < 10; i++) {
    buf[i] = analogRead(analogInPin);
    delay(30);
  }
  for (int i = 0; i < 9; i++) {
    for (int j = i + 1; j < 10; j++) {
      if (buf[i] > buf[j]) {
        temp = buf[i];
        buf[i] = buf[j];
        buf[j] = temp;
      }
    }
  }
  avgValue = 0;
  for (int i = 2; i < 8; i++) {
    avgValue += buf[i];
  }

  float pHVol = (float)avgValue * 5.0 / 1024 / 6;
  float phValue = -5.70 * pHVol + calibration;
  delay(200);
  return (phValue);
}

void loop() {
  //Pasar de float a string
  char phChar[32]; //Almacena la lecutra en un char array para conversion
  float phFloat = lecturaPH(); //Almacena en un float el return de la funcion
```

```
  dtostrf(phFloat, 4, 2, phChar); //Convierte un float(phFloat) de minimo 4
de longitud con 2 decimales hacia un char(phChar)
  String phString(phChar); //Se crea una variable String con el contenido del
char(phChar)

  sprintf(nodo1, "{\"id\":\"1\",\"datetime\":\"2022-06-13
22:10:10\",\"ph\":\"%s\"}\n", phChar);
  Serial.write(nodo1);
  delay(12000);
}
```

## ANEXO 2: Código Arduino para Nodo con 2 sensores para recopilar datos de sensores y enviar archivo tipo JSON a Router de Borde

```
//Adriana Herdoíza
//Nodo con dos sensores - ph y conductividad eléctrica

//PH
float calibration = 20.80; //valor de calibración
const int analogInPinPH = A0;
const int analogInPinCOND = A1;
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10], temp;
String valorpH;
char nodo2[256];
char lectura[6];

//COND
//#define TdsSensorPin A1
#define VREF 5.0 // voltaje de referencia analógico (voltios) deel ADC
#define SCOUNT 30 // suma de punto de muestra
float tdsFin;
int analogBuffer[SCOUNT]; // almacenar el valor analógico en la matriz, leer
desde ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;

void setup() {
  Serial.begin(115200);
}

float lecturaPH() {
  for (int i = 0; i < 10; i++) {
    buf[i] = analogRead(analogInPinPH);
    delay(30);
  }
  for (int i = 0; i < 9; i++) {
    for (int j = i + 1; j < 10; j++) {
      if (buf[i] > buf[j]) {
```

```
        temp = buf[i];
        buf[i] = buf[j];
        buf[j] = temp;
      }
    }
  }
  avgValue = 0;
  for (int i = 2; i < 8; i++) {
    avgValue += buf[i];
  }

  float pHVol = (float)avgValue * 5.0 / 1024 / 6;
  float phValue = -5.70 * pHVol + calibration;
  delay(200);
  return (phValue);
}

float lecturaCOND() {
  //Lectura PIN
  analogBuffer[analogBufferIndex] = analogRead(analogInPinCOND); //leer el
valor analógico y guardarlo en el buffer
  analogBufferIndex++;
  if (analogBufferIndex == SCOUNT) {
    analogBufferIndex = 0;
  }
  delay(40);

  //Filtro
  for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++) {
    analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
  }
  averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF /
1024.0; // leer el valor analógico más estable por el algoritmo de filtrado y
convertirlo en valor de voltaje
  float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0);
//temperature compensation formula: fFinalResult(25^C) =
fFinalResult(current)/(1.0+0.02*(fTP-25.0));
  float compensationVolatge = averageVoltage / compensationCoefficient;
//temperature compensation
  tdsValue = (133.42 * compensationVolatge * compensationVolatge *
compensationVolatge - 255.86 * compensationVolatge * compensationVolatge +
857.39 * compensationVolatge) * 0.5; //convert voltage value to tds value
  //Serial.print("voltage:");
  //Serial.print(averageVoltage,2);
  //Serial.print("V ");
  //Serial.print("TDS Value:");
  tdsFin = tdsValue / 100;
  //Serial.println(tdsFin);
  //Serial.println("ppm");

  return (tdsFin);
}

int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++)
```

```
    bTab[i] = bArray[i];
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
    for (i = 0; i < iFilterLen - j - 1; i++)
    {
      if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

void loop() {

  char phChar[32]; //Almacena la lecutra en un char array para conversion
  char condChar[32]; //Almacena la lecutra en un char array para conversion

  float phFloat = lecturaPH(); //Almacena en un float el return de la funcion
  float condFloat = lecturaCOND(); //Almacena en un float el return de la
funcion


  dtostrf(phFloat, 4, 2, phChar); //Convierte un float(phFloat) de minimo 4
de longitud con 2 decimales hacia un char(phChar)
  //String phString(phChar); //Se crea una variable String con el contenido
del char(phChar)
  dtostrf(condFloat, 4, 2, condChar); //Convierte un float(phFloat) de minimo
4 de longitud con 2 decimales hacia un char(phChar)
  //String phString(condChar); //Se crea una variable String con el contenido
del char(phChar)


  sprintf(nodo2, "{\"id\":\"2\",\"datetime\":\"2022-06-13
22:10:10\",\"ph\":\"%s\",\"cond\":\"%s\"}\n", phChar, condChar);
  Serial.write(nodo2);
  delay(12000);

}
```

**ANEXO 3: Código OpenMote-B para Router de Borde para recibir paquetes MQTT-SN encapsulados en UDP y enrutar.**

```
/*
 * Adriana Herdoíza
 *
 *      Universidad Técnica del Norte
 *          Trabajo de Grado
 */

// La parte principal del Border Router es saber enrutar: Recibe los paquetes
MQTT-SN encapsulados en UDP
//para luego solamente enviarlos al mosquitto
#include <stdio.h>      //Maneja todas las entradas y salidas de datos

#include "shell.h"      //Para poder utilizar la función con los comandos
automáticos, se fuerza para q asi sea
#include "msg.h"        //Necesario en el gneric netif, ipv6, es necesario en
todo el conjunto de librerias
                        //para poder tener enrutamiento, direccionamiento y
formato de msjs que se van a enviar
#include <string.h>     //Maneja todas las cadenas de texto como chars

#define MAIN_QUEUE_SIZE     (8)         //Se utiliza para mandar comandos
shell
static msg_t _main_msg_queue[MAIN_QUEUE_SIZE];  //Se utiliza para mandar
comandos shell

    //De la 21 a la 26 es la función forzada para forzar comandos y directo
al main
void sh(char *c)
{
  char lbuf[100];
  strcpy(lbuf, c);
  handle_input_line(NULL, lbuf);
}

int main(void)
{
    /* necesitamos una cola de mensajes para el subproceso que ejecuta el
shell con el fin de
     * recibir paquetes de red entrantes potencialmente rápidos */
    msg_init_queue(_main_msg_queue, MAIN_QUEUE_SIZE);   //Thread para usar el
shell
    //De la 34 a la 46 son los comandos forzados
    puts("RIOT border router example application");
    puts("Configurando Router de Borde\n");
    sh("ifconfig 5 add 2001:db8::2/64");    //Agregar dirección a la
interface wired
    sh("ifconfig 4 add 2001:db8::3/48");    //Agregar dirección a la interfaz
wireless
    sh("ifconfig 4 set phy mr-fsk");        //Establece la modulación
    sh("ifconfig 4 set power 13");          //Potencia en dbm
    sh("ifconfig 4 set channel 0");         //Canal 0
    sh("ifconfig 4 set modulation_index 1");    //Índice de modulación
```

```
    sh("ifconfig 4 set symbol_rate 50");         //Simbol rate
    sh("ifconfig 4 set channel_spacing 200");    //Separación del canal
    sh("rpl init 4");                            //Activar RPL en iface
wireless
    sh("rpl root 1 2001:db8::3");                //Ruta forzada de rpl a la
inalambrica
    puts("Configuraciones terminadas\n");

    /* shell de inicio */
    puts("All up, running the shell now");
    char line_buf[SHELL_DEFAULT_BUFSIZE];        //Línea 50 y 51 permiten que
se inicie el shell
    shell_run(NULL, line_buf, SHELL_DEFAULT_BUFSIZE);



    /* no debe ser alcanzado */
    return 0;
}
```

## ANEXO 4: Código OpenMote-B Nodo 1 para recolectar datos de Arduino, activación de comandos automáticos y envío de datos al Router de Borde mediante MQTT-SN.

```
// Adriana Herdoíza
//  Trabajo de Titulación - Código para recolectar datos de arduino uno
//Tiene thread para recibir por serial y actualizar una variable char array
global_json
//Tiene cmd_auto funcionando, que esta haciendo el proceso completo de CON,
// REGISTER, PUB y DISCON cada 10s con los datos de global_json (ver lineas
39 y 222)
//Las 3 primeras librerias son propias de C++ incluidas en RIOT, para q C++
haga algo debe contar
//con estas librerías
#include <stdio.h>      //Maneja todas las entradas y salidas de datos
#include <string.h>     //Maneja todas las cadenas de texto como chars
#include <stdlib.h>     //Conjunto de librerías generales
// Librerias agregadas: todas las agregadas sirven para ejecutar procesos
necesarios en el main
#include <time.h>       //Si se utilizara native, para pedir el tiempo del
equipo y generar los datos
#include <xtimer.h>     //Utilizada para OpenMote para poner Delay. Formas de
como usar delays
#include <ztimer.h>     //Utilizada para OpenMote para poner Delay
#include <float.h>      //El RIOT por defecto no maneja float, por esto se lo
incluye
#include "shell.h"      //Para poder utilizar la función con los comandos
automáticos, se fuerza para q asi sea
#include "msg.h"        //Necesario en el gneric netif, ipv6, es necesario en
todo el conjunto de librerias de abajo
                        //para poder tener enrutamiento, direccionamiento y
formato de msjs que se van a enviar
#include "net/emcute.h"     //El emcute tiene todos los métodos del MQTT
como: conectar, desconectar, publicar, etc
```

```
#include "net/ipv6/addr.h"  //Para que tenga el IPv6
#include "net/gnrc.h"       //Necesarios para comunicación UART, es decir,
con la compu
#include "net/gnrc/netif.h" //Necesarios para comunicación UART, es decir,
con la compu
#include "net/netif.h"      //Necesarios para comunicación UART, es decir,
con la compu
#include "periph/adc.h"     //Periph ADC para utilizar pines de OpenMote como
out o in. No se utiliza ningún conversor análogo digital
#include "periph/gpio.h"    //General purpose input/output. Esta y la
anterior libreria peermitía leer los sensores. Se utiliza para la definición
de los 4 leds del openmote
#include <math.h>           //Permite utilizar funciones matemáticas q no
vienen por defecto

//Comunicación UART
#include "board.h"          //Se utilizan para la comunicación uart
#include "periph_conf.h"    //Se utilizan para la comunicación uart

#include "stdio_uart.h"     //Se utilizan para la comunicación uart
#define TEST_BUF_SIZE 256   //Definir variable estática: nombre de variable y
valor. Es la longitud del
                            //búfer a la hora de la lectura serial. Esto
ayuda en tiempos de procesamiento
                            //la lectura del búfer es un proceso repetitivo
por eso se define así

//De la línea 39-43 se utiliza para la comunicación MQTT-SN, por esto se
llama EMCUTE porq RIOT
//ha definido así su libreria para esta funcionalidad
#ifndef EMCUTE_ID               //¿Voy a tener ID?     EMCUTE_ID actúa como
bandera
#define EMCUTE_ID ("SENSOR1")   //Si voy a tener ID definela como SENSOR1.
#endif
#define EMCUTE_PORT (1883U)
#define EMCUTE_PRIO (THREAD_PRIORITY_MAIN - 1)

//SERIAL_THREAD - Forma o estructura en la que se definen los hilos
static char serial_stack[THREAD_STACKSIZE_MAIN]; //Este char es un proceso
para guardar datos en otro char
                                                 //La estructura de los hilos
es: nombre y tamaño (THREAD_STACKSIZE_MAIN)
                                                 //Estos tamaños son variables
definidas que ya están dentro de librerias del RIOT

//SENDMQTT_THREAD - Forma en la que se definen los hilos
static char sendMqtt_stack[THREAD_STACKSIZE_MAIN];

//Configuraciones automaticas (54-55)
#define MAIN_QUEUE_SIZE     (8)                  //Para enviar comandos al
open son las dos lineas (comandos shell)
static msg_t _main_msg_queue[MAIN_QUEUE_SIZE];

static char stack[THREAD_STACKSIZE_DEFAULT];     //Es otro hilo pero para
escuchar comandos por teclado
static msg_t queue[8];                           //Cola o búfer que se define
para los comandos
```

```
char test_buffer[TEST_BUF_SIZE] = "POR DEFECTO";; //Variable para UART. Se
guarda por defecto lo que se lee por serial

char json_global[256] = "DEFECTO";  //Otro búfer que va a guardar
globalmente, este es el q se muestra


//De la 69 a la 74 es una función llamada sh sobrecargada con un char
//Significa que cada que llame a la función sh, lo que yo ponga lo ejecuta la
consola como si yo lo escribiera
//Los datos no deben ser más de 100. Hace un strcopy y copia en el lbuf y
utiliza la funcion handle_input_line con un parámetro NULO y el comando
//y se ejecuta el comando de una manera forzada
void sh(char *c)        //(un char con apuntador significa que le puedo pasar
un string)
{
  char lbuf[100];
  strcpy(lbuf, c);
  handle_input_line(NULL, lbuf);
}

//De la 81 a la 86 es la Siguiente parte de como se define el thread de mqtt
//Le pone el nombre con el apuntador*. Le ponen en el void los argumentos q
se van a repetir
//por eso, lo que se va a repetir siempre es emcute_run.
//proceso que siempre va a estar activo para los sockets para abrir el mqtt
porq agrupa un puerto y un id
//el puerto es el del OpenMote (1883) y el ID que es SENSOR1. 1885 es el q
siempre escucha el mosquitto
static void *emcute_thread(void *arg)
{
    (void)arg;
    emcute_run(EMCUTE_PORT, EMCUTE_ID);
    return NULL; /* nunca debe ser alcanzado */
}

//SERIAL_THREAD - Siguiente parte de como se define el hilo
//el thread del emcute esta arriba incluso, aqui va bien este
//tiene la misma estructura de static void con el nombre para asignarle al
hilo con un apuntador y sobrecargado
//significa q es un proceso q siempre se va a ejecutar, a pesar de q hayan
más procesos, este siempre se ejecuta
static void *readSerial_thread(void *arg)
{
    (void)arg;
    //SERIAL RX
    int received = 0;

    while (1) {             //Con while significa q ni bien entra siempre se
ejecuta
        char c;             //Se quiere esto para definir primero un char o
caracter
        stdio_read(&c, 1);  //le metemos lo que haya leido la función
stdio_read (es la funcion q lee por serial)
                            //Se le dice q lea el char q acabamos de definir
con tamaño 1 y va a poner ahí el caracter que acabó de leer
```

```
                            //le guardamos en el mismo caracter de tamaño 1

        test_buffer[received] = c;  //Se define un nuevo búfer y dice q en la
posición received (posición 0) meta el caracter q acabó de leer
                                    //el proceso sigue
        if (received < TEST_BUF_SIZE - 1) {     //si todavía no se desborda
el buffer sigue contando, el dato q sigue aumenta una posicion y continuamos
            received++;                         //y continuamos
            //LED0_TOGGLE;
            //LED1_TOGGLE;
        }

        if (c == '\n' || c == '\r') {       //Si es q también el caracter q
acabamos de recibir usa un salto de linea o enter, significa que hasta ahí
llegó
            strncpy(json_global, test_buffer, received);    //copiame a un
string (json global q es otro bufer definido) lo que hay en test_bufer, es
decir,
                                                            //lo q acabas de
leer copialo en el json
            json_global[received] = '\0';       //la última posicion sea
igual a /0. Es para que como tiene 256 lea solo hasta el caracter de
terminacion

            //Reenvio a arduino
            stdio_write(test_buffer, received);
            stdio_write("\n", 1);

            //LEDS
            LED2_TOGGLE;    //leds
            received = 0;   //se resetea el contador
        }
    }
    return NULL; /* nunca debe ser alcanzado */ //cuando termine el proceso
se repite el proceso

    //hasta aquí se acaba el hilo del serial
}

        //MQTT-SN
        //Función de Desconexión
// Función que se desconecta de la puerta de enlace mqttsn (todo el bloque
133-147)
static int discon(void){        //si llamo al discon ya me desconecta. //en
todo este bloque se llama a funciones propias de la libreria q portamos int
res=emcute
    int res = emcute_discon();  // emcute_discon es el método q se llama para
desconectar, pero el resultado de eso se guarda en un entero

    if (res == EMCUTE_NOGW) {
        puts("error: not connected to any broker");
        return 1;
    }
    else if (res != EMCUTE_OK) {
        puts("error: unable to disconnect");
        return 1;
    }
```

```
    puts("Disconnect successful");
    return 0;
}

// Esta es la función que publica mensajes al topic
// toma como entrada el topic, el mensaje a enviar y
// el valor de qos
static int pub(char* topic, char* data, int qos){   //para publicar se pasa
el topic, los datos y el qos q se maneja
  emcute_topic_t t;     //Define un objeto emcute_topic_t para guardar en una
determinada estructura los datos del topic
  unsigned flags = EMCUTE_QOS_0;

  switch (qos) {
      case 1:
        flags |= EMCUTE_QOS_1;      //banderas para determinar la calidad del
servicio
        break;
      case 2:
        flags |= EMCUTE_QOS_2;
        break;
      default:
        flags |= EMCUTE_QOS_0;
        break;
  }

  /* step 1: obtener el id del tema */
  t.name = topic;       //Lo q les mando de topic lo guardan en el objeto t
definido en 154 xq tiene una estructura
                        //en t.name está el nombre del topic al q se quieren
publicar

  if (emcute_reg(&t) != EMCUTE_OK) {    //funcion para registrarnos en el
topic
      puts("error: unable to obtain topic ID"); //de acuerdo a lo q envien en
el topic podemos decir q hubo error o no decir nada
      return 1;     //para decir hasta aqui llega la funcion si hay error
  }

  /* step 2: publicar datos */
  //emcute_pub(&t, data, strlen(data), flags);

        //Función para publicar
  /* Adriana*/ //Chequear si es q se publicó 183-187
  if (emcute_pub(&t, data, strlen(data), flags) != EMCUTE_OK) { //le pasamos
el objeto t (topic), la data, longitud de data y las banderas de qos
      printf("error: unable to publish data to topic '%s [%i]'\n",  //si da
error regresa 1 e imprime el error, si no hay error ya pasa el dato
              t.name, (int)t.id);
      return 1;
  }
  //el proceso anterior sigue si no hay error y va a la 186 para publicar y
se prenden los leds
  printf("Publicado %s en el topic %s\n", data, topic);
  LED3_TOGGLE;
```

```
  return 0;
}

        // Función que se conecta a la puerta de enlace mqtt
// toma como entrada la dirección ip y el puerto
static int con(char* addr, int port){
  sock_udp_ep_t gw = { .family = AF_INET6, .port = EMCUTE_PORT };   //crea un
objeto gw de familia INET6 (maneja ipv6) y se le pone el puerto definido
antes como EMCUTE_PORT = 1883
  gw.port = port;   //pone el puerto

  /* analizar dirección */ //De la línea 201-205 la lógica de chequear, que
quiero chequear? ipv6_addr_from_str. Solo para decir si lo q se ingresó si es
una dirección ipv6 válida o no
  if (ipv6_addr_from_str((ipv6_addr_t *)&gw.addr.ipv6, addr) == NULL) {
      printf("error parsing IPv6 address\n");
      return 1;
  }
    //emcute_con(&gw, false, NULL, NULL, 0, 0);
  //Línea 208, intenta conectar y chequea el mensaje para ver si hay error o
no
  if (emcute_con(&gw, true, NULL, NULL, 0, 0) != EMCUTE_OK) {
      printf("error: unable to connect to [%s]:%i\n", addr, port);
      return 1;
  }
  //como lo anterior estuvo bien, entonces Conectado exitosamente
  printf("Conectado exitosamente al Gateway [%s]:%i\n", addr, port);
  return 0;
}

        //Aquí se terminó la función de conectar

// nuevo comando de shell: auto envio
// la función toma la dirección IP de entrada y el puerto de la puerta de
enlace,
// y el id de la estación especificada
// cada cinco segundos genera nuevos valores de sensor y los publica en
// sensor/station + station id
static int cmd_auto(){
  puts("Inicio cmd_auto");

  // nombre del topic
  char topic[32];
  sprintf(topic,"sensor/station%d", 1);



  while(1){ //Me da a que da problemas un loop junto con el serial
    // intenta conectarse a la puerta de enlace
    if (con("2001:db8::1", 1885)) {
      continue;
    }

    // json que se publicará
    //char json[256] = "0";

    // actualiza los valores del sensor
```

```
    //sprintf(json, "%s", test_buffer);

    // publicar en el topic
    pub(topic, json_global, 0);

    // se desconecta de la puerta de enlace
    discon();

    // duerme durante diez segundos
    xtimer_sleep(10);
  }

  return 0;
}

//MQTT THREAD - Hilo que va a estar ejecutándose
static void *sendMqtt_thread(void *arg)
{
    (void)arg;

    puts("Inicio cmd_auto");  //desde aqui es lo mismo de la función cmd
topic. Inicio de hilo

    // nombre del topic
    char topic[32];  //el topic va a tener máximo 32 bits xq es para
sensor/station
    sprintf(topic,"sensor/station%d", 1); //sensor/station es el tema, forma
en la q mqtt-sn maneja sus mensajes. Todo lo q vaya a publicar se va al tema
sensor/station

    // json que se publicará
    //char json[256];

    while(1){
        // intenta conectarse a la puerta de enlace mediante dirección y
puerto
        if (con("2001:db8::1", 1885)) {
            LED0_TOGGLE;        //si da error se prende un led y continua
          continue;     //cuidado, en nuestro ciclo while pasó algo raro,
olvida el código de abajo y vuelve a empezar
        }               //si todo va bien se va a la línea 283 para publicar

        // actualiza los valores del sensor
        //sprintf(json, "%s", test_buffer);

        // publicar en el topic
        pub(topic, json_global, 0);     //se le pasa el topic, los datos o
sea el json global con qos 0

        // después de enviar se desconecta de la puerta de enlace
        discon();

        LED1_TOGGLE;        //se enciende un led para indicar

        // duerme durante quince segundos para q se vuelva a ejecutar el
while
        xtimer_sleep(15);
```

```
    }

    return NULL; /* nunca debe ser alcanzado */
}



        //Desde aquí hasta el final, se definen los comandos personalizados
para el shell
static const shell_command_t shell_commands[] = {
    {"sauto", "Start the station auto", cmd_auto},
    {NULL, NULL, NULL}};

int main(void)
{

    //printf("Puertos PB2 y PB3 inicializados!\n");
    puts("Aplicacion para enviar datos de sensores a traves de MQTT-SN\n");
    puts("Escriba 'help' para ver todos los comandos, envie los datos con el
formato: "
         "<comando> <Direccion IP> <Puerto> <ID>\n");
    puts("Por ejemplo: send 2001:db8::1 1885 1\n");

    //Tercera parte para definir el hilo de los comandos
/* el hilo principal necesita una cola de mensajes para poder ejecutar
`ping6`*/
    msg_init_queue(queue, ARRAY_SIZE(queue));   //esta es la tercera parte

    /* Se necesita una cola de mensajes para el subproceso que ejecuta el
shell
     * para recibir paquetes de red entrantes potencialmente rápidos */
    msg_init_queue(_main_msg_queue, MAIN_QUEUE_SIZE);
    puts("RIOT network stack example application");

    //Tercera parte del hilo del emcute
    /* empieza el hilo emcute */
    thread_create(stack, sizeof(stack), EMCUTE_PRIO, 0,
                  emcute_thread, NULL, "emcute");
    //Adriana*/

    //SERIAL_THREAD
    thread_create(serial_stack,             /* terminar de definir el hilo.
*/
                  sizeof(serial_stack),     /* tamaño */
                  THREAD_PRIORITY_MAIN - 1, /* prioridad del hilo (todos
estamos manejando iguales) */
                  0,                        /* bandera de configuración
extra, se le pone 0 si no se usa */
                  readSerial_thread,        /* nombre del hilo (así se llama
donde está todo el proceso del mqtt)*/
                  NULL,                     /* argumento adicional nulo */
                  "readSerial_thread");     /* nombre adicional para
consultas */

    //SERIAL_THREAD
    thread_create(sendMqtt_stack,           /* en sendMqtt está todo el
proceso del while de enviar mqtt, desconectar y dormir */
```

```
                sizeof(sendMqtt_stack),     /* tamaño */
                THREAD_PRIORITY_MAIN - 1,   /* prioridad del hilo (todos
estamos manejando iguales) */
                0,                          /* bandera de configuración
extra, se le pone 0 si no se usa */
                sendMqtt_thread,            /* nombre del hilo (así se llama
donde está todo el proceso del serial) */
                NULL,                       /* argumento adicional nulo */
                "sendMqtt_thread");         /* nombre adicional para
consultas */

    //Configuraciones automaticas
    puts("Configurando Nodo sensor 1\n");
    sh("ifconfig");
    sh("ifconfig 6 add 2001:db8::4/48");
    sh("ifconfig 6 set phy mr-fsk");
    sh("ifconfig 6 set power 13");
    sh("ifconfig 6 set channel 0");
    sh("ifconfig 6 set modulation_index 1");
    sh("ifconfig 6 set symbol_rate 50");
    sh("ifconfig 6 set channel_spacing 200");
    sh("rpl init 6");
    sh("ifconfig");
    puts("Configuraciones terminadas\n");
    puts("Empezando envio de datos...\n");

    //INICIA cmd_auto y hace el proceso MQTT cada 10s
    //a la par el thread esta recibiendo datos y almancenando en global_json
    //cmd_auto funcionara independientemente de si llegan o no datos por
serial
    //estara enviando peridocamente cada 10s los datos que hayan llegado (o
no)
    //cmd_auto(); SU PROPIO THREAD

    /* inicio del shell */
    char line_buf[SHELL_DEFAULT_BUFSIZE];
    shell_run(shell_commands, line_buf, SHELL_DEFAULT_BUFSIZE);



    /* Nunca debe ser alcanzado */
    return 0;

}
```

**ANEXO 5: Código OpenMote-B Nodo 2 para recolectar datos de Arduino, activación de comandos automáticos y envío de datos al Router de Borde mediante MQTT-SN.**

```
// Adriana Herdoíza
//  Trabajo de Titulación - Código para recolectar datos de arduino uno con
dos sensores
//Tiene thread para recibir por serial y actualizar una variable char array
global_json
//Tiene cmd_auto funcionando, que esta haciendo el proceso completo de CON,
// REGISTER, PUB y DISCON cada 10s con los datos de global_json (ver lineas
39 y 222)
//Las 3 primeras librerias son propias de C++ incluidas en RIOT, para q C++
haga algo debe contar
//con estas librerías
#include <stdio.h>      //Maneja todas las entradas y salidas de datos
#include <string.h>     //Maneja todas las cadenas de texto como chars
#include <stdlib.h>     //Conjunto de librerías generales
// Librerias agregadas: todas las agregadas sirven para ejecutar procesos
necesarios en el main
#include <time.h>       //Si se utilizara native, para pedir el tiempo del
equipo y generar los datos
#include <xtimer.h>     //Utilizada para OpenMote para poner Delay. Formas de
como usar delays
#include <ztimer.h>     //Utilizada para OpenMote para poner Delay
#include <float.h>      //El RIOT por defecto no maneja float, por esto se lo
incluye
#include "shell.h"      //Para poder utilizar la función con los comandos
automáticos, se fuerza para q asi sea
#include "msg.h"        //Necesario en el gneric netif, ipv6, es necesario en
todo el conjunto de librerias de abajo
                        //para poder tener enrutamiento, direccionamiento y
formato de msjs que se van a enviar
#include "net/emcute.h"     //El emcute tiene todos los métodos del MQTT
como: conectar, desconectar, publicar, etc
#include "net/ipv6/addr.h"  //Para que tenga el IPv6
#include "net/gnrc.h"       //Necesarios para comunicación UART, es decir,
con la compu
#include "net/gnrc/netif.h" //Necesarios para comunicación UART, es decir,
con la compu
#include "net/netif.h"      //Necesarios para comunicación UART, es decir,
con la compu
#include "periph/adc.h"     //Periph ADC para utilizar pines de OpenMote como
out o in. No se utiliza ningún conversor análogo digital
#include "periph/gpio.h"    //General purpose input/output. Esta y la
anterior libreria peermitía leer los sensores. Se utiliza para la definición
de los 4 leds del openmote
#include <math.h>           //Permite utilizar funciones matemáticas q no
vienen por defecto

//Comunicación UART
#include "board.h"          //Se utilizan para la comunicación uart
#include "periph_conf.h"    //Se utilizan para la comunicación uart

#include "stdio_uart.h"     //Se utilizan para la comunicación uart
```

```
#define TEST_BUF_SIZE 256   //Definir variable estática: nombre de variable y
valor. Es la longitud del
                            //búfer a la hora de la lectura serial. Esto
ayuda en tiempos de procesamiento
                            //la lectura del búfer es un proceso repetitivo
por eso se define así

//De la línea 39-43 se utiliza para la mcomunicación MQTT-SN, por esto se
llama EMCUTE porq RIOT
//ha definido así su libreria para esta funcionalidad
#ifndef EMCUTE_ID              //¿Voy a tener ID?     EMCUTE_ID actúa como
bandera
#define EMCUTE_ID ("SENSOR2")   //Si voy a tener ID definela como SENSOR2.
#endif
#define EMCUTE_PORT (1883U)
#define EMCUTE_PRIO (THREAD_PRIORITY_MAIN - 1)

//SERIAL_THREAD - Forma o estructura en la que se definen los hilos
static char serial_stack[THREAD_STACKSIZE_MAIN];    //Este char es un proceso
para guardar datos en otro char
                                                    //La estructura de los hilos
es: nombre y tamaño (THREAD_STACKSIZE_MAIN)
                                                    //Estos tamaños son variables
definidas que ya están dentro de librerias del RIOT

//SENDMQTT_THREAD
static char sendMqtt_stack[THREAD_STACKSIZE_MAIN];

//Config automaticas (54-55)
#define MAIN_QUEUE_SIZE     (8) //Para enviar comandos al open son las dos
lineas (comandos shell)
static msg_t _main_msg_queue[MAIN_QUEUE_SIZE];

static char stack[THREAD_STACKSIZE_DEFAULT];        //Es otro hilo pero para
escuchar comandos por teclado
static msg_t queue[8];                              //Cola o búfer que se
define para los comandos

char test_buffer[TEST_BUF_SIZE] = "POR DEFECTO";; //Variable para UART. Se
guarda por defecto lo que se lee por serial

char json_global[256] = "DEFECTO";  //Otro búfer que va a guardar
globalmente, este es el q se muestra


//De la 69 a la 74 es una función llamada sh sobrecargada con un char
//Significa que cada que llame a la función sh, lo que yo ponga lo ejecuta la
consola como si yo lo escribiera
//Los datos no deben ser más de 100. Hace un strcopy y copia en el lbuf y
utiliza la funcion handle_input_line con un parámetro NULO y el comando
//y se ejecuta el comando de una manera forzada
void sh(char *c)        //(un char con apuntador significa que le puedo pasar
un string)
{
  char lbuf[100];
  strcpy(lbuf, c);
  handle_input_line(NULL, lbuf);
```

```
}

//De la 81 a la 86 es la Siguiente parte de como se define el thread de mqtt
//Le pone el nombre con el apuntador*. Le ponen en el void los argumentos q
se van a repetir
//por eso, lo que se va a repetir siempre es emcute_run.
//proceso que siempre va a estar activo para los sockets para abrir el mqtt
porq agrupa un puerto y un id
//el puerto es el del OpenMote (1883) y el ID que es SENSOR2. 1885 es el q
siempre escucha el mosquitto
static void *emcute_thread(void *arg)
{
    (void)arg;
    emcute_run(EMCUTE_PORT, EMCUTE_ID);
    return NULL; /* nunca debe ser alcanzado */
}

//SERIAL_THREAD - Siguiente parte de como se define el hilo
//el thread del emcute esta arriba incluso, aqui va bien este
//tiene la misma estructura de static void con el nombre para asignarle al
hilo con un apuntador y sobrecargado
//significa q es un proceso q siempre se va a ejecutar, a pesar de q hayan
más procesos, este siempre se ejecuta
static void *readSerial_thread(void *arg)
{
    (void)arg;
    //SERIAL RX
    int received = 0;

    while (1) {             //Con while significa q ni bien entra siempre se
ejecuta
        char c;             //Se quiere esto para definir primero un char o
caracter
        stdio_read(&c, 1);  //le metemos lo que haya leido la función
stdio_read (es la funcion q lee por serial)
                            //Se le dice q lea el char q acabamos de definir
con tamaño 1 y va a poner ahí el caracter que acabó de leer
                            //le guardamos en el mismo caracter de tamaño 1

        test_buffer[received] = c;  //Se define un nuevo búfer y dice q en la
posición received (posición 0) meta el caracter q acabó de leer
                                    //el proceso sigue
        if (received < TEST_BUF_SIZE - 1) {     //si todavía no se desborda
el buffer sigue contando, el dato q sigue aumenta una posicion y continuamos
            received++;                         //y continuamos
            //LED0_TOGGLE;
            //LED1_TOGGLE;
        }

        if (c == '\n' || c == '\r') {   //Si es q también el caracter q
acabamos de recibir usa un salto de linea o enter, significa que hasta ahí
llegó
            strncpy(json_global, test_buffer, received);    //copiame a un
string (json global q es otro bufer definido) lo que hay en test_bufer, es
decir,
                                                            //lo q acabas de
leer copialo en el json
```

```
            json_global[received] = '\0';       //la última posicion sea
igual a /0. Es para que como tiene 256 lea solo hasta el caracter de
terminacion

            //Reenvio a arduino
            stdio_write(test_buffer, received);
            stdio_write("\n", 1);

            //LEDS
            LED2_TOGGLE;    //leds
            received = 0;   //se resetea el contador
        }
    }
    return NULL; /* nunca debe ser alcanzado */  //cuando termine el proceso
se repite el proceso

    //hasta aquí se acaba el hilo del serial
}

        //MQTT-SN
        //Función de Desconexión
// Función que se desconecta de la puerta de enlace mqttsn (todo el bloque
134-148)
static int discon(void){        //si llamo al discon ya me desconecta. //en
todo este bloque se llama a funciones propias de la libreria q portamos int
res=emcute
    int res = emcute_discon();  // emcute_discon es el método q se llama para
desconectar, pero el resultado de eso se guarda en un entero

    if (res == EMCUTE_NOGW) {
        puts("error: not connected to any broker");
        return 1;
    }
    else if (res != EMCUTE_OK) {
        puts("error: unable to disconnect");
        return 1;
    }

    puts("Disconnect successful");
    return 0;
}

// Esta es la función que publica mensajes al topic
// toma como entrada el topic, el mensaje a enviar y
// el valor de qos
static int pub(char* topic, char* data, int qos){   //para publicar se pasa
el topic, los datos y el qos q se maneja
  emcute_topic_t t;         //Define un objeto emcute_topic_t para guardar en
una determinada estructura los datos del topic
  unsigned flags = EMCUTE_QOS_0;

  switch (qos) {
      case 1:
        flags |= EMCUTE_QOS_1;      //banderas para determinar la calidad del
servicio
        break;
      case 2:
```

```
        flags |= EMCUTE_QOS_2;
        break;
      default:
        flags |= EMCUTE_QOS_0;
        break;
  }

  /* step 1: obtener el id del tema */
  t.name = topic;       //Lo q les mando de topic lo guardan en el objeto t
definido en 153 xq tiene una estructura
                        //en t.name está el nombre del topic al q se quieren
publicar

  if (emcute_reg(&t) != EMCUTE_OK) {    //funcion para registrarnos en el
topic
      puts("error: unable to obtain topic ID");     //de acuerdo a lo q
envien en el topic podemos decir q hubo error o no decir nada
      return 1;         //para decir hasta aqui llega la funcion si hay error
  }

  /* step 2: publicar datos */
  //emcute_pub(&t, data, strlen(data), flags);

        //Función para publicar
  /* Adriana*/  //Chequear que si se publicó 183 - 187
  if (emcute_pub(&t, data, strlen(data), flags) != EMCUTE_OK) {     //le
pasamos el objeto t (topic), la data, longitud de data y las banderas de qos
      printf("error: unable to publish data to topic '%s [%i]'\n",  //si da
error regresa 1 e imprime el error, si no hay error ya pasa el dato
             t.name, (int)t.id);
      return 1;
  }
  //el proceso anterior sigue si no hay error y va a la 186 para publicar y
se prenden los leds
  printf("Publicado %s en el topic %s\n", data, topic);
  LED3_TOGGLE;

  return 0;
}

        // Función que se Conecta a la puerta de enlace mqtt
// toma como entrada la dirección ip y el puerto
static int con(char* addr, int port){
  sock_udp_ep_t gw = { .family = AF_INET6, .port = EMCUTE_PORT };
  gw.port = port;

  /* analizar dirección */  //De la línea 201-205 la lógica de chequear, que
quiero chequear? ipv6_addr_from_str. Solo para decir si lo q se ingresó si es
una dirección ipv6 válida o no
  if (ipv6_addr_from_str((ipv6_addr_t *)&gw.addr.ipv6, addr) == NULL) {
      printf("error parsing IPv6 address\n");
      return 1;
  }
    //emcute_con(&gw, false, NULL, NULL, 0, 0);
    //Línea 208, intenta conectar y chequea el mensaje para ver si hay error
o no
  if (emcute_con(&gw, true, NULL, NULL, 0, 0) != EMCUTE_OK) {
```

```
      printf("error: unable to connect to [%s]:%i\n", addr, port);
      return 1;
  }
  //como lo anterior estuvo bien, entonces Conectado exitosamente
  printf("Conectado exitosamente al Gateway [%s]:%i\n", addr, port);
  return 0;
}

        //Aquí se terminó la función de conectar

// nuevo comando de shell: auto envio
// la función toma la dirección IP de entrada y el puerto de la puerta de
enlace,
// y el id de la estación especificada
// cada cinco segundos genera nuevos valores de sensor y los publica en
// sensor/station + station id
static int cmd_auto(){
  puts("Inicio cmd_auto");

  // nombre del topic
  char topic[32];
  sprintf(topic,"sensor/station%d", 2);



  while(1){ //Me da a que da problemas un loop junto con el serial
    // intenta conectarse a la puerta de enlace
    if (con("2001:db8::1", 1885)) {
      continue;
    }

    // json que se publicará
    //char json[256] = "0";

    // actualiza los valores del sensor
    //sprintf(json, "%s", test_buffer);

    // publicar en el topic
    pub(topic, json_global, 0);

    // se desconecta de la puerta de enlace
    discon();

    // duerme durante diez segundos
    xtimer_sleep(10);
  }

  return 0;
}

//MQTT THREAD - Hilo que va a estar ejecutándose
static void *sendMqtt_thread(void *arg)
{
    (void)arg;

    puts("Inicio cmd_auto");    //desde aqui es lo mismo de la función cmd
topic. Inicio de hilo
```

```
    // nombre del topic
    char topic[32];     //el topic va a tener máximo 32 bits xq es para
sensor/station
    sprintf(topic,"sensor/station%d", 2);   //sensor/station es el tema,
forma en la q mqtt-sn maneja sus mensajes. Todo lo q vaya a publicar se va al
tema sensor/station

    // json que se publicará
    //char json[256];

    while(1){
        // intenta conectarse a la puerta de enlace mediante dirección y
puerto
        if (con("2001:db8::1", 1885)) {
            LED0_TOGGLE;        //si da error se prende un led y continua
          continue;             //cuidado, en nuestro ciclo while pasó algo
raro, olvida el código de abajo y vuelve a empezar
        }                       //si todo va bien se va a la línea 283 para
publicar

        // actualiza los valores del sensor
        //sprintf(json, "%s", test_buffer);

        // publicar en el topic
        pub(topic, json_global, 0);     //se le pasa el topic, los datos o
sea el json global con qos 0

        // Después de enviar se desconecta de la puerta de enlace
        discon();

        LED1_TOGGLE;        //se enciende un led para indicar

        // duerme durante quince segundos para q se vuelva a ejecutar el
while
        xtimer_sleep(15);
    }

    return NULL; /* nunca debe ser alcanzado */
}



        //Desde aquí hasta el final, se definen los comandos personalizados
para el shell
static const shell_command_t shell_commands[] = {
    {"sauto", "Start the station auto", cmd_auto},
    {NULL, NULL, NULL}};

int main(void)
{

    //printf("Puertos PB2 y PB3 inicializados!\n");
    puts("Aplicacion para enviar datos de sensores a traves de MQTT-SN\n");
    puts("Escriba 'help' para ver todos los comandos, envie los datos con el
formato: "
         "<comando> <Direccion IP> <Puerto> <ID>\n");
```

```
    puts("Por ejemplo: send 2001:db8::1 1885 1\n");

    //Tercera parte para definir el hilo de los comandos
/* el hilo principal necesita una cola de mensajes para poder ejecutar
`ping6`*/
    msg_init_queue(queue, ARRAY_SIZE(queue));

    /* Se necesita una cola de mensajes para el subproceso que ejecuta el
shell
     * para recibir paquetes de red entrantes potencialmente rápidos */
    msg_init_queue(_main_msg_queue, MAIN_QUEUE_SIZE);
    puts("RIOT network stack example application");

    //Tercera parte del hilo del emcute
    /* empieza el hilo emcute */
    thread_create(stack, sizeof(stack), EMCUTE_PRIO, 0,
                  emcute_thread, NULL, "emcute");
    //Adriana*/

    //SERIAL_THREAD
    thread_create(serial_stack,             /* terminar de definir el hilo.
*/
                  sizeof(serial_stack),     /* tamaño */
                  THREAD_PRIORITY_MAIN - 1, /* prioridad del hilo (todos
estamos manejando iguales) */
                  0,                        /* bandera de configuración
extra, se le pone 0 si no se usa */
                  readSerial_thread,        /* nombre del hilo (así se llama
donde está todo el proceso del mqtt)*/
                  NULL,                     /* argumento adicional nulo */
                  "readSerial_thread");     /* nombre adicional para
consultas */

    //SERIAL_THREAD
    thread_create(sendMqtt_stack,           /* en sendMqtt está todo el
proceso del while de enviar mqtt, desconectar y dormir */
                  sizeof(sendMqtt_stack),   /* tamaño */
                  THREAD_PRIORITY_MAIN - 1, /* prioridad del hilo (todos
estamos manejando iguales) */
                  0,                        /* bandera de configuración
extra, se le pone 0 si no se usa */
                  sendMqtt_thread,          /* nombre del hilo (así se llama
donde está todo el proceso del serial) */
                  NULL,                     /* argumento adicional nulo */
                  "sendMqtt_thread");       /* nombre adicional para
consultas */

    //Configuraciones automaticas
    puts("Configurando Nodo sensor 2\n");
    sh("ifconfig");
    sh("ifconfig 6 add 2001:db8::5/48");
    sh("ifconfig 6 set phy mr-fsk");
    sh("ifconfig 6 set power 13");
    sh("ifconfig 6 set channel 0");
    sh("ifconfig 6 set modulation_index 1");
    sh("ifconfig 6 set symbol_rate 50");
    sh("ifconfig 6 set channel_spacing 200");
```

```
    sh("rpl init 6");
    sh("ifconfig");
    puts("Configuraciones terminadas\n");
    puts("Empezando envio de datos...\n");

    //INICIA cmd_auto y hace el proceso MQTT cada 10s
    //a la par el thread esta recibiendo datos y almancenando en global_json
    //cmd_auto funcionara independientemente de si llegan o no datos por
serial
    //estara enviando peridocamente cada 10s los datos que hayan llegado (o
no)
    //cmd_auto(); SU PROPIO THREAD

    /* start shell */
    char line_buf[SHELL_DEFAULT_BUFSIZE];
    shell_run(shell_commands, line_buf, SHELL_DEFAULT_BUFSIZE);



    /* Nunca debe ser alcanzado */
    return 0;

}
```

## ANEXO 6: Código Bridge para cambiar paquetes de MQTT-SN a MQTT y establecer comunicación con AWS para enviar los datos.

```
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient
from datetime import datetime
import pytz
import MQTTSNclient
import json
import boto3

# conexión a DynamoDB y acceso a la tabla Tesis
dynamodb = boto3.resource('dynamodb', region_name='us-east-2')
dynamoTable = dynamodb.Table('Tesis')
jsonP = ''

# clientes para MQTT y MQTTS
MQTTClient = AWSIoTMQTTClient("MQTTSNbridge")
MQTTSNClient = MQTTSNclient.Client("bridge", port=1885)


class Callback:
  # función que responde un mensaje del intermediario MQTTSN al intermediario
MQTT
  # e inserta en la base de datos el mensaje recién llegado
  def messageArrived(self, topicName, payload, qos, retained, msgid):
      message = payload.decode("utf-8")
      print("MSG: ", message)
      # datetime
      now = datetime.now(pytz.timezone('America/Guayaquil'))
```

```
        dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
        jsonP = json.loads(message)
        jsonP["datetime"] = dt_string
        #jsonP["ph"] = "99"
        #json_mylist = json.dumps(jsonP, separators=(',', ':'))
        jsonDoble = json.dumps(jsonP)
        print(topicName, jsonDoble)
        #MQTTClient.publish(topicName, json_mylist, qos) #message
        MQTTClient.publish(topicName, jsonDoble, qos)
        dynamoTable.put_item(Item=jsonP) #jsonP
        return True

# ruta que indica la posición de los certificados
path = "/home/riot/InternetOfThings19-20-master/InternetOfThings19-20-
master/SecondAssignment/certs/"

# configurar el acceso con el broker AWS MQTT
MQTTClient.configureEndpoint("aazppadns27yu-ats.iot.us-east-2.amazonaws.com",
8883) #aazppadns27yu-ats.iot.us-east-2.amazonaws.com
MQTTClient.configureCredentials(path+"AmazonRootCA1.pem",

path+"a694b69cdd0034fd87318f2e79ab39dd2e9de9bfce8a9555bfd3c74d9912c2c3-
private.pem.key",

path+"a694b69cdd0034fd87318f2e79ab39dd2e9de9bfce8a9555bfd3c74d9912c2c3-
certificate.pem.crt")

# configure el broker MQTT
MQTTClient.configureOfflinePublishQueueing(-1)  # Infinite offline Publish
queueing
MQTTClient.configureDrainingFrequency(2)  # Draining: 2 Hz
MQTTClient.configureConnectDisconnectTimeout(10)  # 10 sec
MQTTClient.configureMQTTOperationTimeout(5)  # 5 sec

# registrar la devolución de llamada
MQTTSNClient.registerCallback(Callback())

# Hacer conexiones con los clientes
MQTTClient.connect()
MQTTSNClient.connect()

# preguntar al usuario a que station quiere suscribirse
# Los clientes RiotOS MQTTSN publican en el topic sensor/station + id
# el usuario define solo el id
station_ids = ""
print("Enter the ID of the station, one by one, that you want to subscribe
to.")
print("Type 'stop' to interrupt the process.\n")
while True:
    current_id = input("")
    if current_id == 'stop':
        break
    else:
        station_ids += current_id + " "

# Suscríbete a los temas elegidos por el usuario.
for id in station_ids:
```

```
    MQTTSNClient.subscribe("sensor/station" + id)
print("Subscribed to stations with ID: " + station_ids)

# ciclo que espera un comando para cerrar el programa
while True:
    if input("Enter 'quit' to exit from the program.\n")=="quit":
        break

# Desconectarse de los clientes
MQTTSNClient.disconnect()
MQTTClient.disconnect()
```

**ANEXO 7: Paso de datos recolectados desde DynamoDB en AWS hasta Grafana por medio de Panoply.**

Cuando se haya concluido con la programación de todo el proceso de la red de sensores, es importante verificar que los datos se estén tomando en la base de datos DynamoDB de AWS, específicamente en la Tabla denominada "Tesis", a partir de ello, se podrá hacer uso de las demás herramientas como lo es el intermediario Panoply para finalizar con la visualización de los datos recibidos en Grafana.

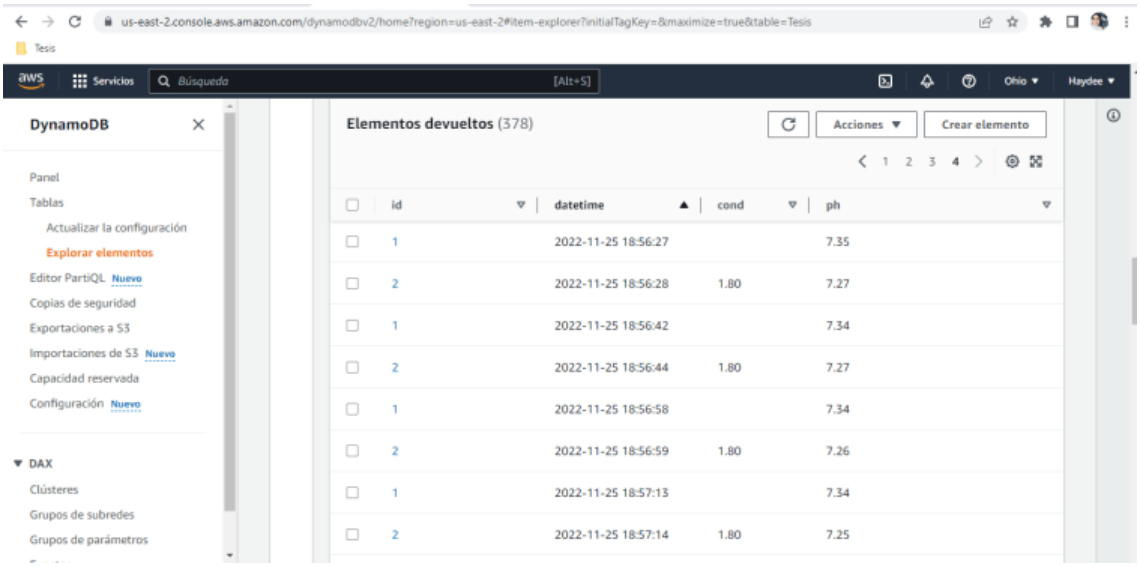



*Ilustración 1. Datos obtenidos en DynamoDB en AWS.*
*Fuente: Autor.*

Una vez que los datos ya se están visualizando en la Base de Datos DynamoDB de AWS, se procede a conectar con el puente que será Panoply, para ello, primeramente, se creará un rol de IAM para Panoply.

Dirigirse hacia la consola de AWS IAM y dar clic en → ***Usuarios*** y luego en → ***Agregar usuario*** en la parte superior derecha de la ventana.

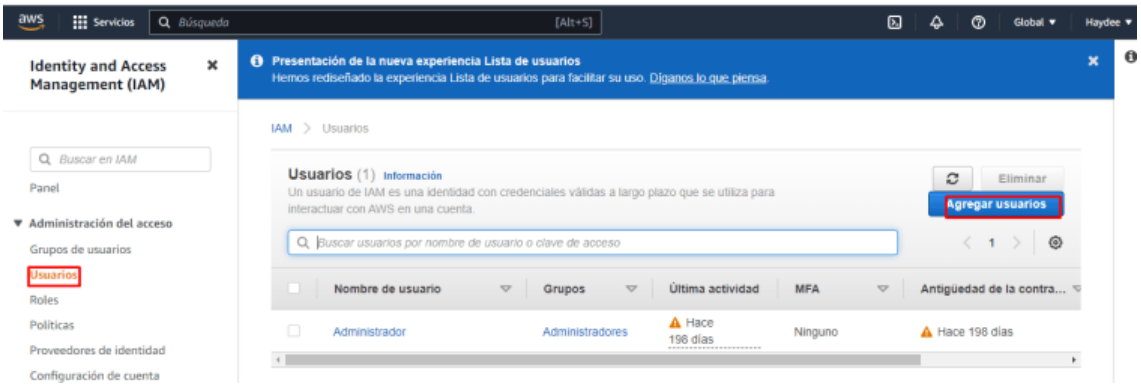



*Ilustración 2. Ventana para agregar usuarios de IAM.*
*Fuente: Autor.*

En la ventana que aparece a continuación, se debe colocar un nombre de Usuario como se muestra en la Figura, y se debe marcar la casilla de → ***Acceso mediante programación***, esto permite generar un *ID de clave de acceso* y una *clave de acceso secreta*. Cuando se haya ingresado el nombre se da clic en → ***Siguiente: Permisos***.

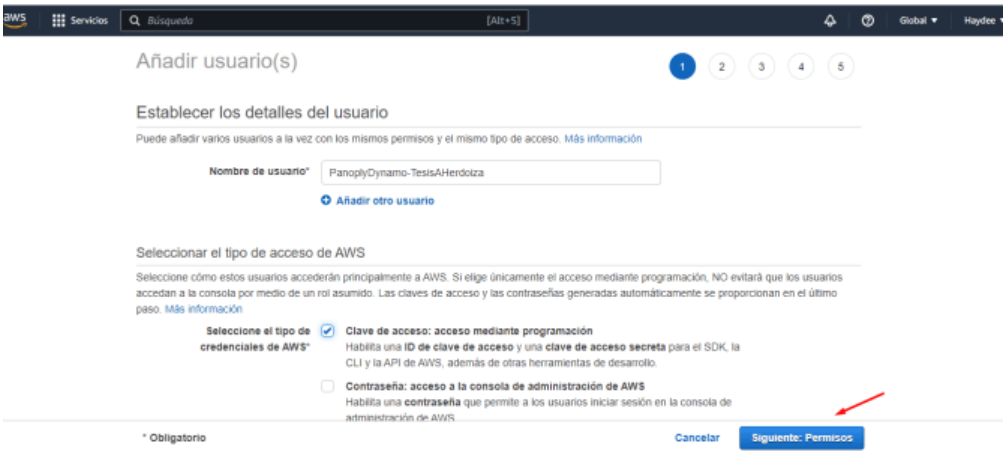



*Ilustración 3. Nombre de nuevo usuario de IAM*
*Fuente: Autor.*

Lo siguiente es agregar este usuario a un grupo existente, para esto se necesita un nuevo grupo para la cuenta de Panoply o agregar políticas de IAM directamente a la cuenta que se creó, para esto, en la nueva ventana se debe elegir en → ***Adjuntar políticas existentes directamente*** desde las opciones de arriba. Luego filtrar las políticas escribiendo en el buscador → ***DynamoDB*** y seleccionar la política → ***Acceso de sólo lectura a DynamoDB*** tal como se muestra en la Figura siguiente, además, se debe dar clic en → ***Siguiente: Etiquetas*** y este paso omitirse.

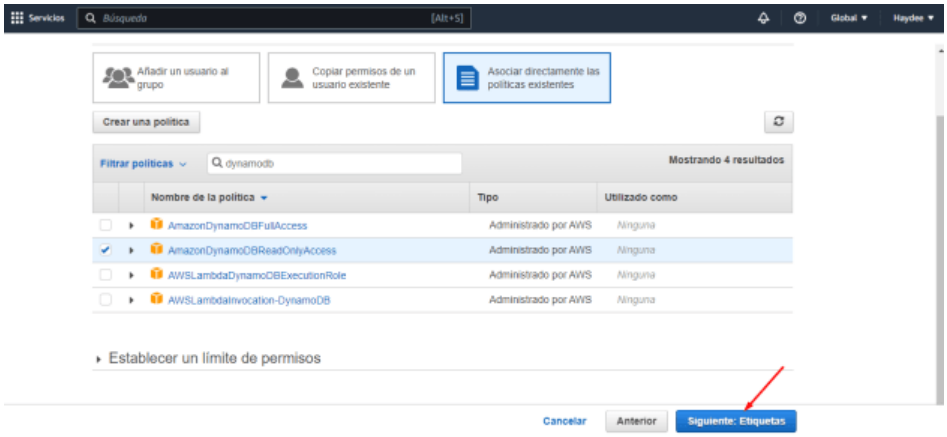



*Ilustración 4. Selección de Política de Acceso.*
*Fuente: Autor.*

Al dar clic en → ***Siguiente*** se da paso a la → *Revisión*, si todo está correcto dar clic en → ***Crear Usuario*** en la parte inferior derecha como se muestra.

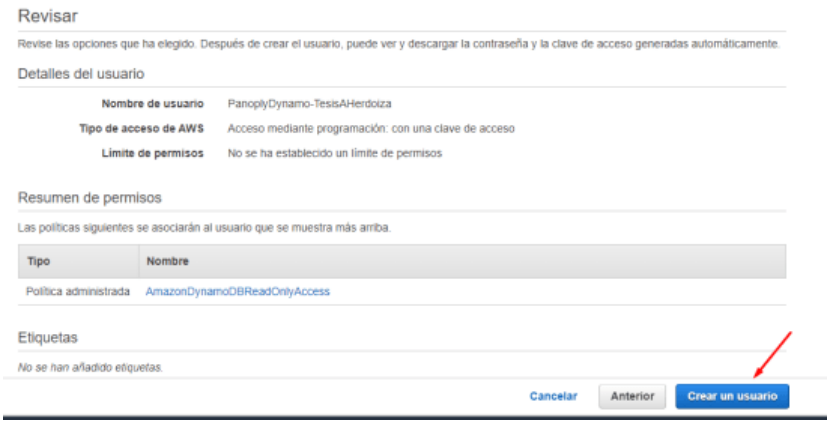



*Ilustración 5. Crear usuario IAM.*
*Fuente: Autor.*

La página siguiente muestra el → ***ID de clave y acceso*** y la → ***clave de acceso secreta***, es aconsejable descargarlos como archivos → *csv* en el apartado indicado.

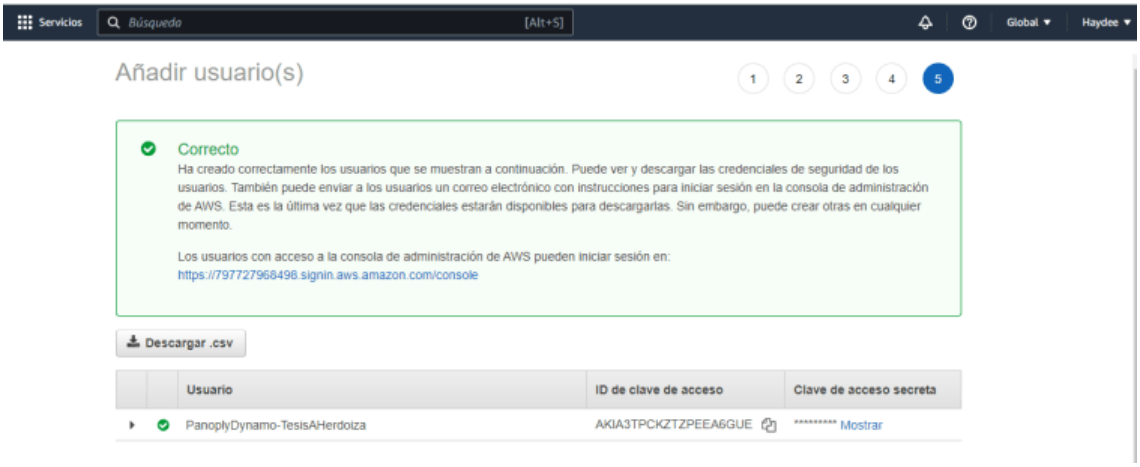



*Ilustración 6. Ventana para descargo de claves de usuario IAM para Panoply.*
*Fuente: Autor.*

A continuación, se deben extraer los datos de DynamoDB y llevarlos a Panoply, es necesario crear una cuenta con un correo empresarial en Panoply, la prueba durará 14 días antes de ser suspendida. Desde el panel de Panoply dar clic en → ***Fuentes de datos*** en el lado izquierdo de la ventana y luego presionar en → ***Agregar fuente de datos*** en la esquina superior derecha y se busca a DynamoDB.

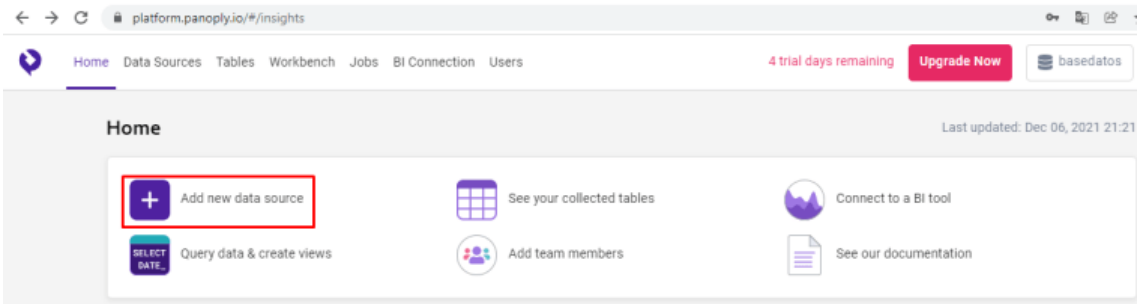



*Ilustración 7. Agregar fuentes de datos.*
*Fuente: Autor.*

Luego de haber seleccionado → *DynamoDB*, se abrirá un nuevo panel que requiere que se ingresen las credenciales antes descargadas para el usuario de IAM y la región a la que pertenecen las tablas a utilizarse de acuerdo con AWS; enseguida se conectará y se cargarán las

Tablas creadas en dicha región, la que se utilizará es la denominada → ***Tesis***, y luego se debe dar clic en → ***Save Changes***.

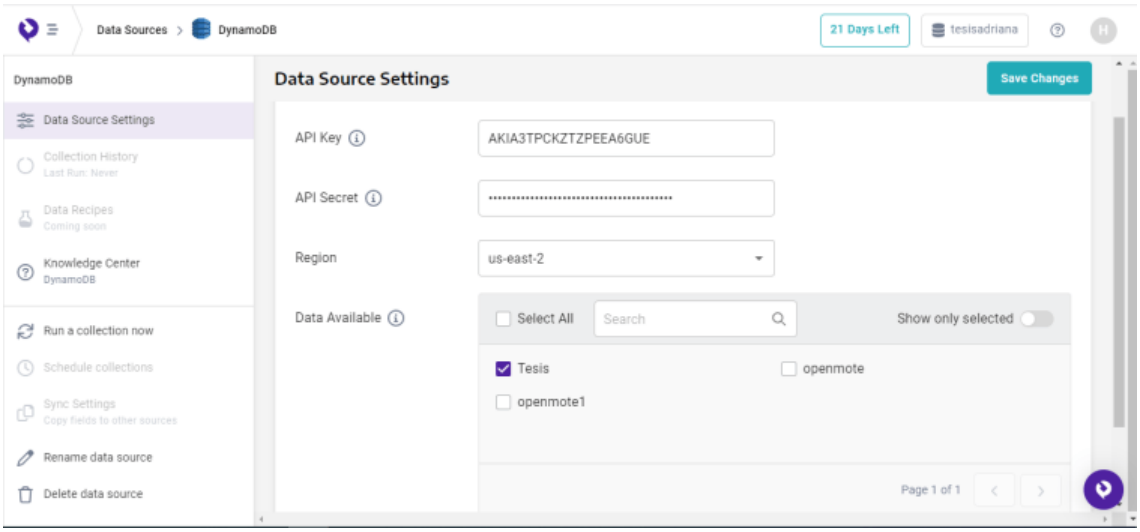



*Ilustración 8. Ingreso de claves, región y selección de la Tabla de almacenamiento de datos de DynamoDB.*
*Fuente: Autor.*

Cuando ya se han guardado los cambios, aparecerá un botón de color azul que dice → ***Collect Data***, se da clic en dicho botón y se espera que carguen los datos que está recuperando.

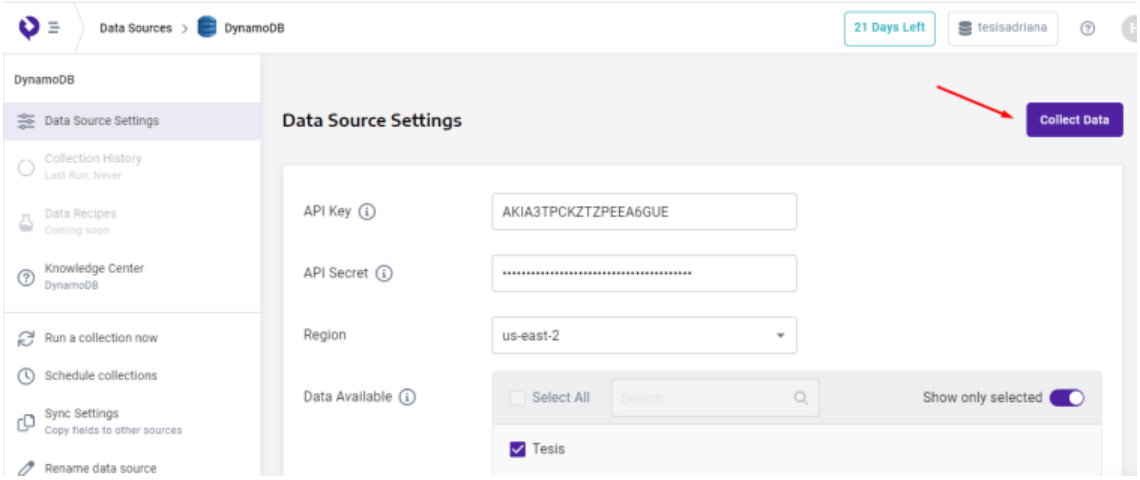



*Ilustración 9. Recolección de datos de la Tabla de DynamoDB.*
*Fuente: Autor.*

El proceso de recolección de datos puede tardar algunos minutos, pero eso se mostrará en la pantalla que se carga a continuación donde indica que se está estableciendo la recolección.

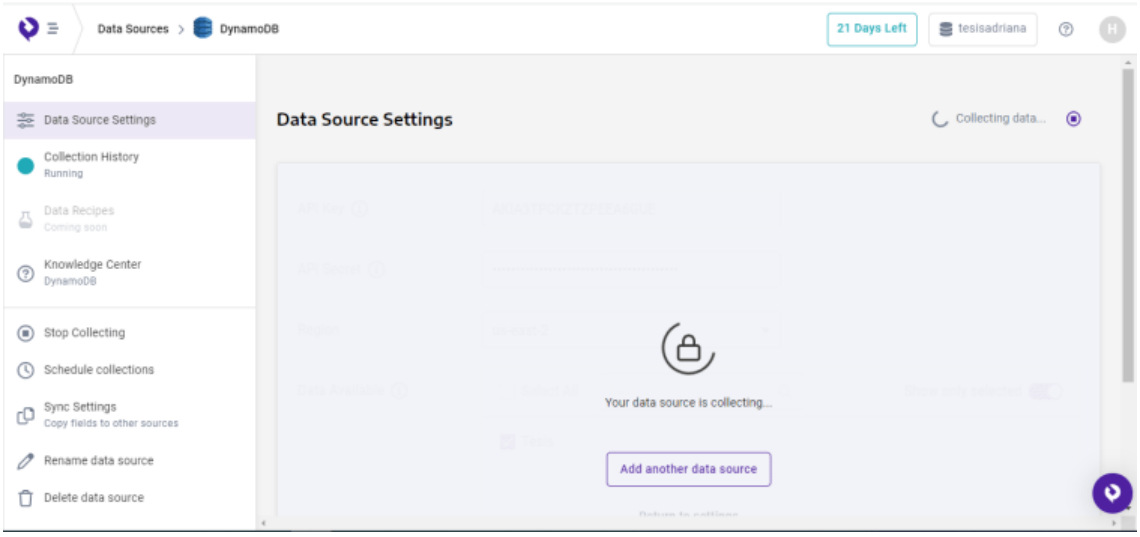



*Ilustración 10. Proceso de recolección de datos de la tabla de DynamoDB en Panoply.*
*Fuente: Autor.*

A continuación, se observa que los datos ya han sido recuperados y se puede revisar que son los mismos datos de la Tabla Tesis de DynamoDB la que ahora se visualiza en Panoply.

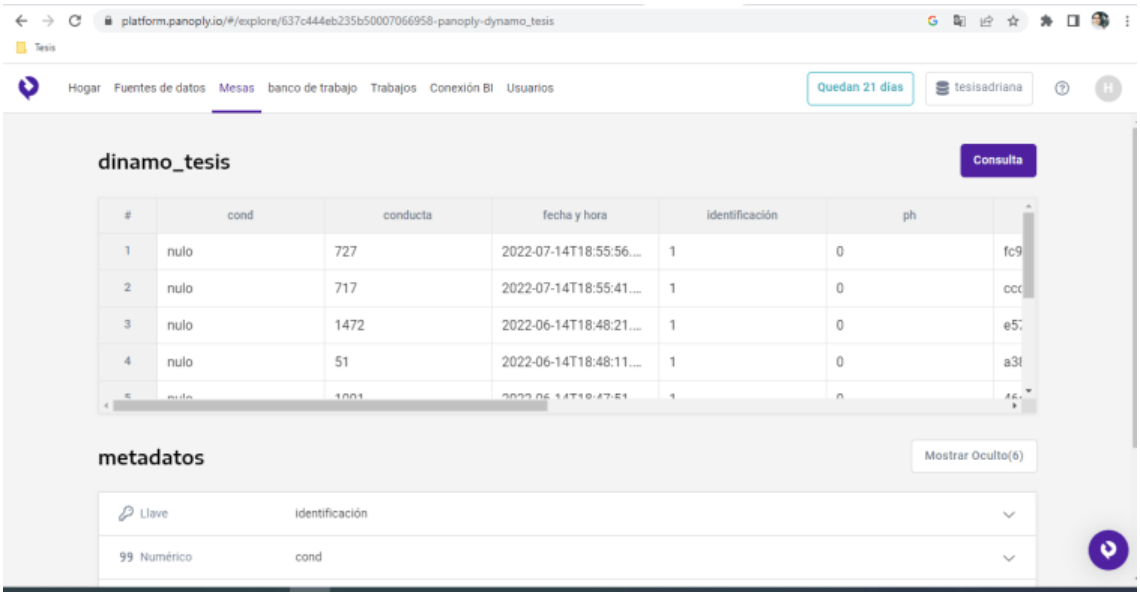



*Ilustración 11. Datos recolectados en Panoply de la Tabla de DynamoDB.*
*Fuente: Autor.*

Para lograr que Grafana se conecte con Panoply, se necesita utilizar la herramienta BigQuery, en Panoply se debe dar clic en → ***BI Connection*** y se obtendrán los datos del Panoply junto con el usuario, nombre de proyecto, entre otros, se debe dar clic en → ***Send your service***

***account JSON file to your email***. Llegará un archivo en el nuevo correo y es necesario descargarlo.

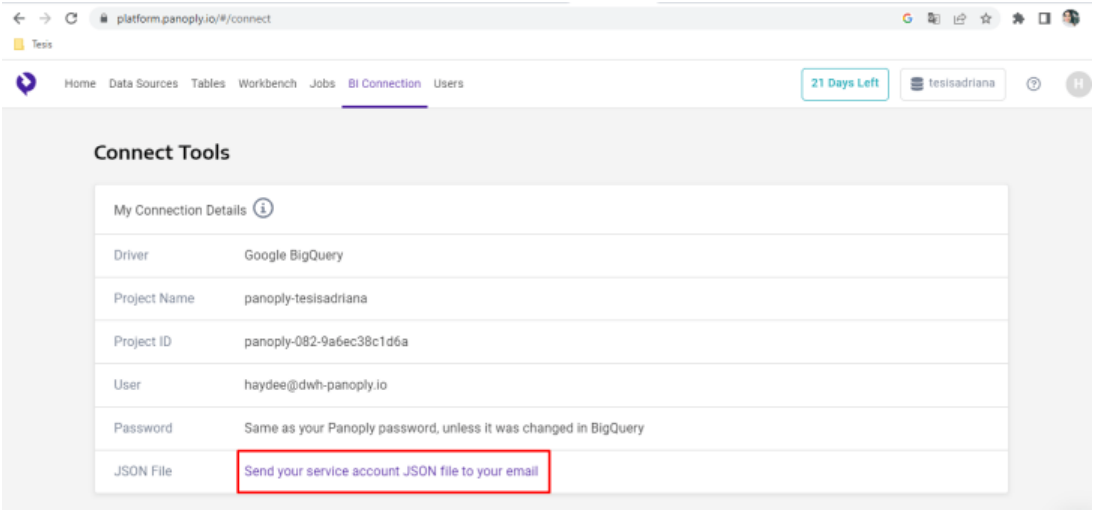



*Ilustración 12. Enviar archivo JSON al correo adjunto en Panoply.*
*Fuente: Autor.*

Lo siguiente es iniciar sesión en Grafana, dentro de esta se da clic en → **Configuración** y luego en → **Complementos** para instalar un plugin.

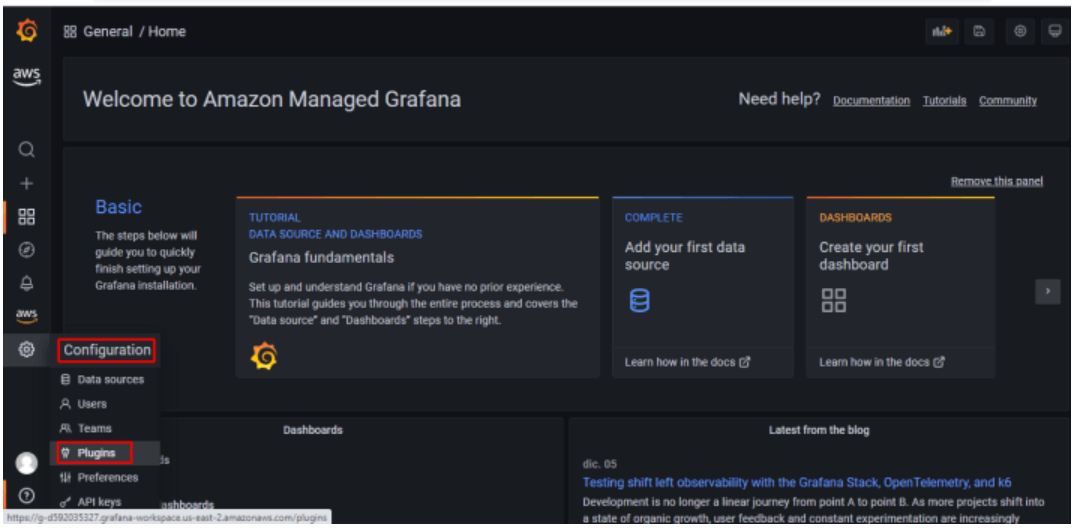



*Ilustración 13. Ingreso a los complementos disponibles.*
*Fuente: Autor.*

En el buscador se debe escribir → ***Google BigQuery*** que es el complemento que se necesita y luego se da clic en el botón de → ***Instalar vía Grafana.com*** como se observa en la Figura siguiente.


*Ilustración 14. Instalar BigQuery mediante Grafana.*
*Fuente: Autor.*

Se procede a realizar la descarga de BigQuery dando clic en el botón verde → ***Install Plugin***, además, en la ventana se puede observar la cuenta de la cual se está realizando la descarga.

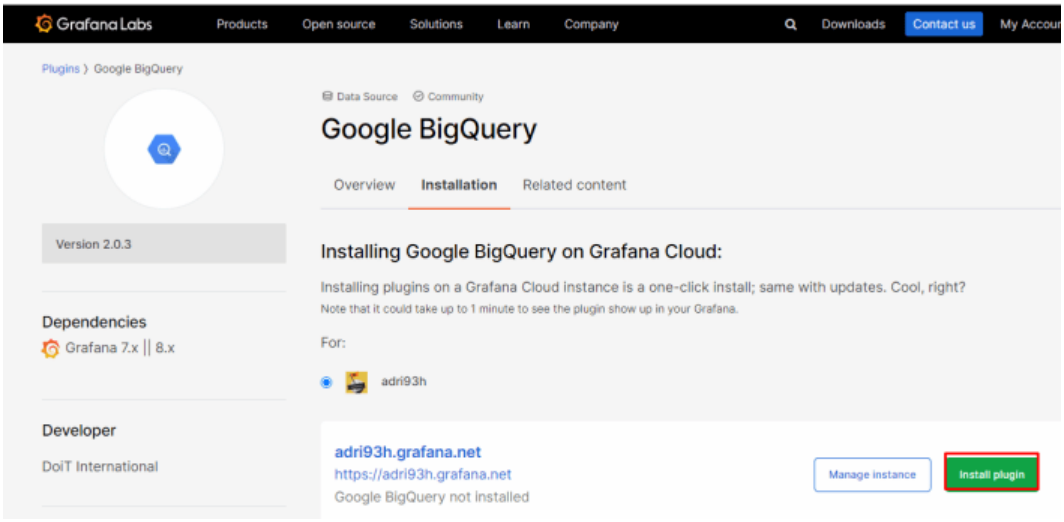



*Ilustración 15. Instalar plugin de BigQuery.*
*Fuente: Autor.*

Se puede observar que el plugin se instaló correctamente en la ventana que aparece a continuación y se observa un Botón rojo donde se indica que puede → *Eliminar* el plugin.

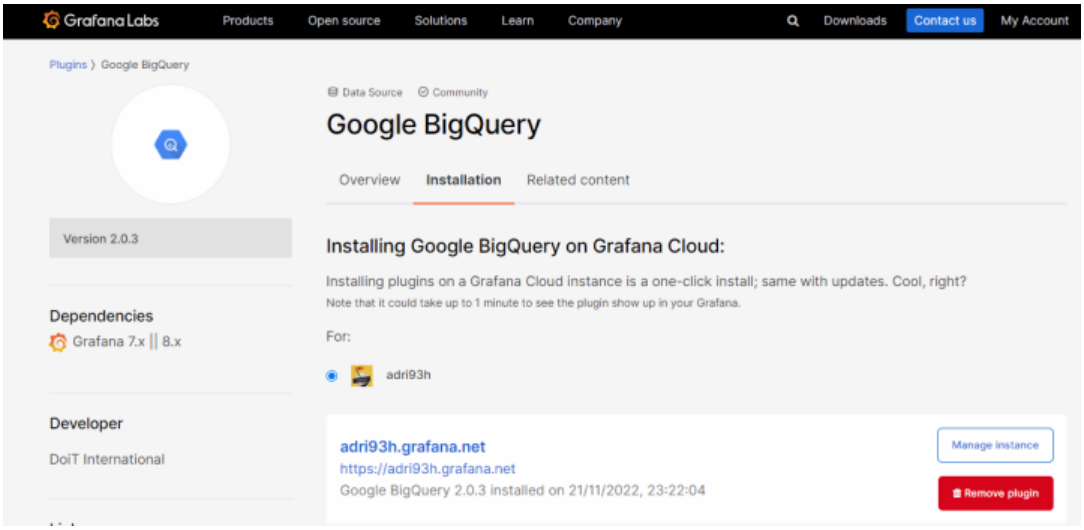



*Ilustración 16. Instalación correcta del plugin.*
*Fuente: Autor.*

Se debe regresar a Grafana.net y luego dar en → ***Configuración*** y luego en → ***Fuentes de datos***, en esta sección se debe escribir → ***Google BigQuery*** en la barra de búsqueda y luego en → ***Agregar fuente de datos***.

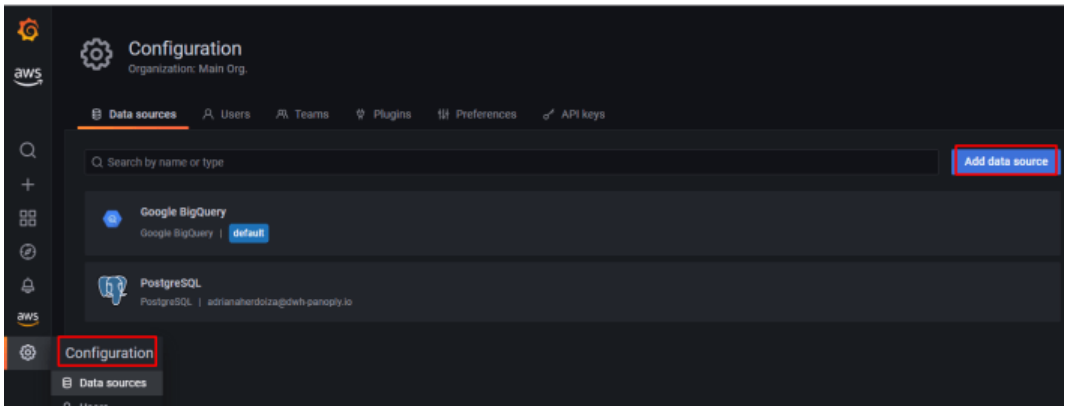



*Ilustración 17. Agregar fuente de datos de BigQuery.*
*Fuente: Autor.*

La pantalla que aparece seguidamente, aparece automáticamente los campos necesarios ya llenos, además, hay un campo vacío que permite ingresar los datos que se encontraban en el JSON descargado con anterioridad, se da clic en → ***Save and test***.

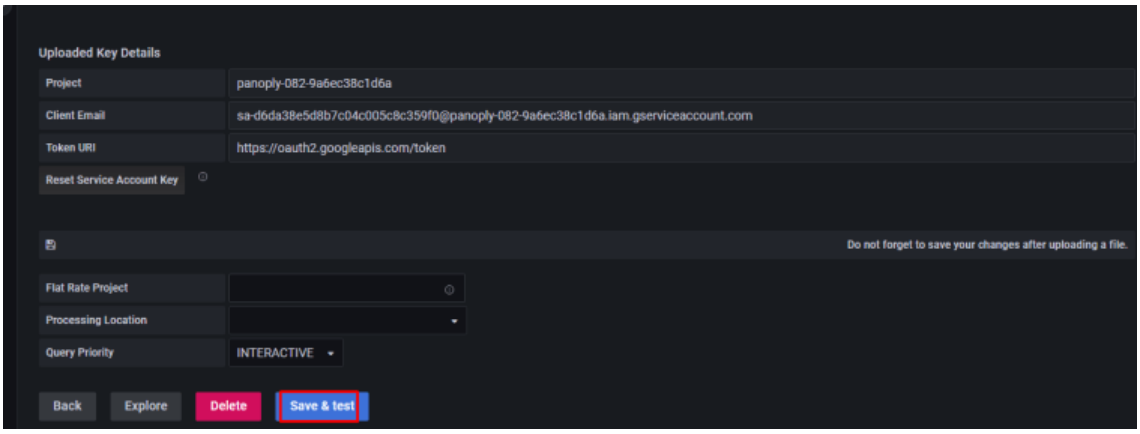



*Ilustración 18. Llenar campos y subir JSON.*
*Fuente: Autor.*

Aparecerá un visto de color verde que indica que los campos requeridos por BigQuery han sido guardados exitosamente.

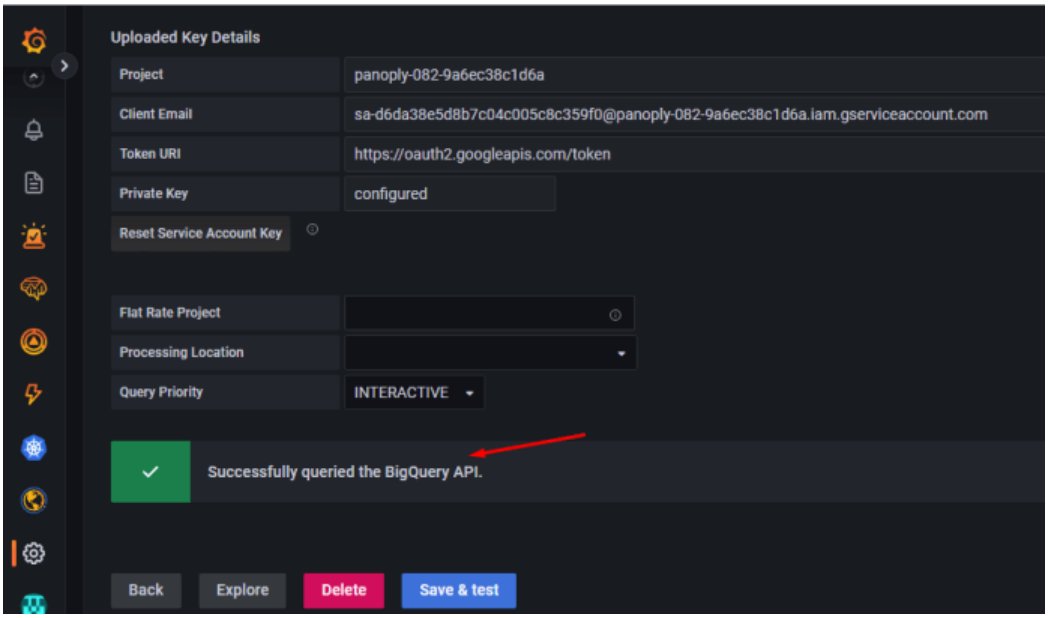



*Ilustración 19. Campos de BigQuert guardados exitosamente.*
*Fuente: Autor.*

A continuación, hay que dirigirse hacia la sección de → ***Explorar*** y en el buscador que aparece escribir BigQuery y dar clic sobre el mismo.

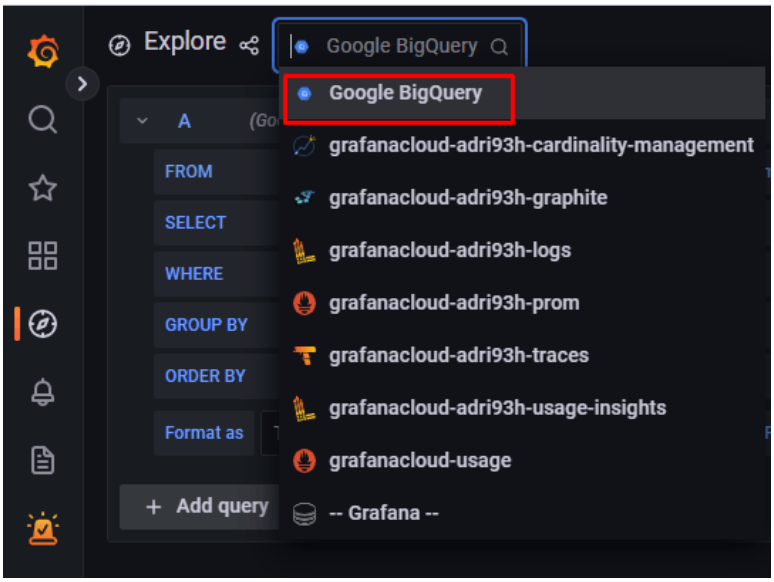



*Ilustración 20. Escoger BigQuery en el Explorador.*
*Fuente: Autor.*

Finalmente, aparecerán las métricas necesarias en las cuales se colocará en este caso la conductividad y el ph que serán las medidas que se tomarán, se observa en la parte inferior que los datos se han obtenido correctamente en Grafana desde DynamoDB.

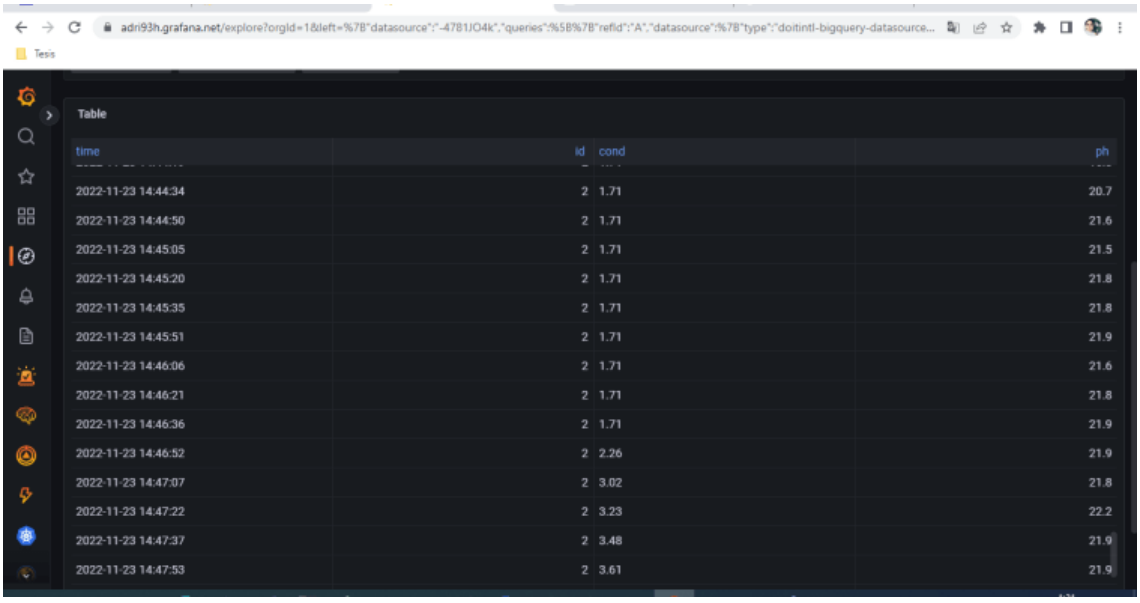



*Ilustración 21. Datos obtenidos en Grafana.*
*Fuente: Autor.*

**ANEXO 8: Tabla de Canales, Modos y Frecuencias**

De acuerdo con la información recopilada del estándar IEEE 802.15.4g, se tiene para la modulación de FSK tres modos diferentes, cada uno de ellos con frecuencias centrales iniciales y número de canales distintos, se procede a realizar una tabla con las Frecuencias para cada uno de los canales utilizados en FSK, donde se especifica que para el modo de operación 1, FSK trabaja con 35 canales, para el modo de operación 2 y 3 trabaja con 17 canales, gracias a los datos de ancho de ancho de banda, se podrá calcular cada una de las frecuencias por canal, en el caso del Modo 1, el ancho de banda es de 200, mientras que para los otros dos es de 400 el ancho de banda.

*Tabla 24. Tabla de Canales, Modos y Frecuencias para FSK.*

| FSK | | | |
|---|---|---|---|
| Canales | Modos (MHz) | | |
| | Modo 1 | Modo 2 | Modo 3 |
| 0 | 863,125 | 863,225 | 863,225 |
| 1 | 863,325 | 863,625 | 863,625 |
| 2 | 863,525 | 864,025 | 864,025 |
| 3 | 863,725 | 864,425 | 864,425 |
| 4 | 863,925 | 864,825 | 864,825 |
| 5 | 864,125 | 865,225 | 865,225 |
| 6 | 864,325 | 865,625 | 865,625 |
| 7 | 864,525 | 866,025 | 866,025 |
| 8 | 864,725 | 866,425 | 866,425 |
| 9 | 864,925 | 866,825 | 866,825 |
| 10 | 865,125 | 867,225 | 867,225 |
| 11 | 865,325 | 867,625 | 867,625 |
| 12 | 865,525 | 868,025 | 868,025 |
| 13 | 865,725 | 868,425 | 868,425 |
| 14 | 865,925 | 868,825 | 868,825 |
| 15 | 866,125 | 869,225 | 869,225 |
| 16 | 866,325 | 869,625 | 869,625 |
| 17 | 866,525 | | |
| 18 | 866,725 | | |
| 19 | 866,925 | | |
| 20 | 867,125 | | |
| 21 | 867,325 | | |

| | |
|---|---|
| 22 | 867,525 |
| 23 | 867,725 |
| 24 | 867,925 |
| 25 | 868,125 |
| 26 | 868,325 |
| 27 | 868,525 |
| 28 | 868,725 |
| 29 | 868,925 |
| 30 | 869,125 |
| 31 | 869,325 |
| 32 | 869,525 |
| 33 | 869,725 |
| 34 | 869,925 |

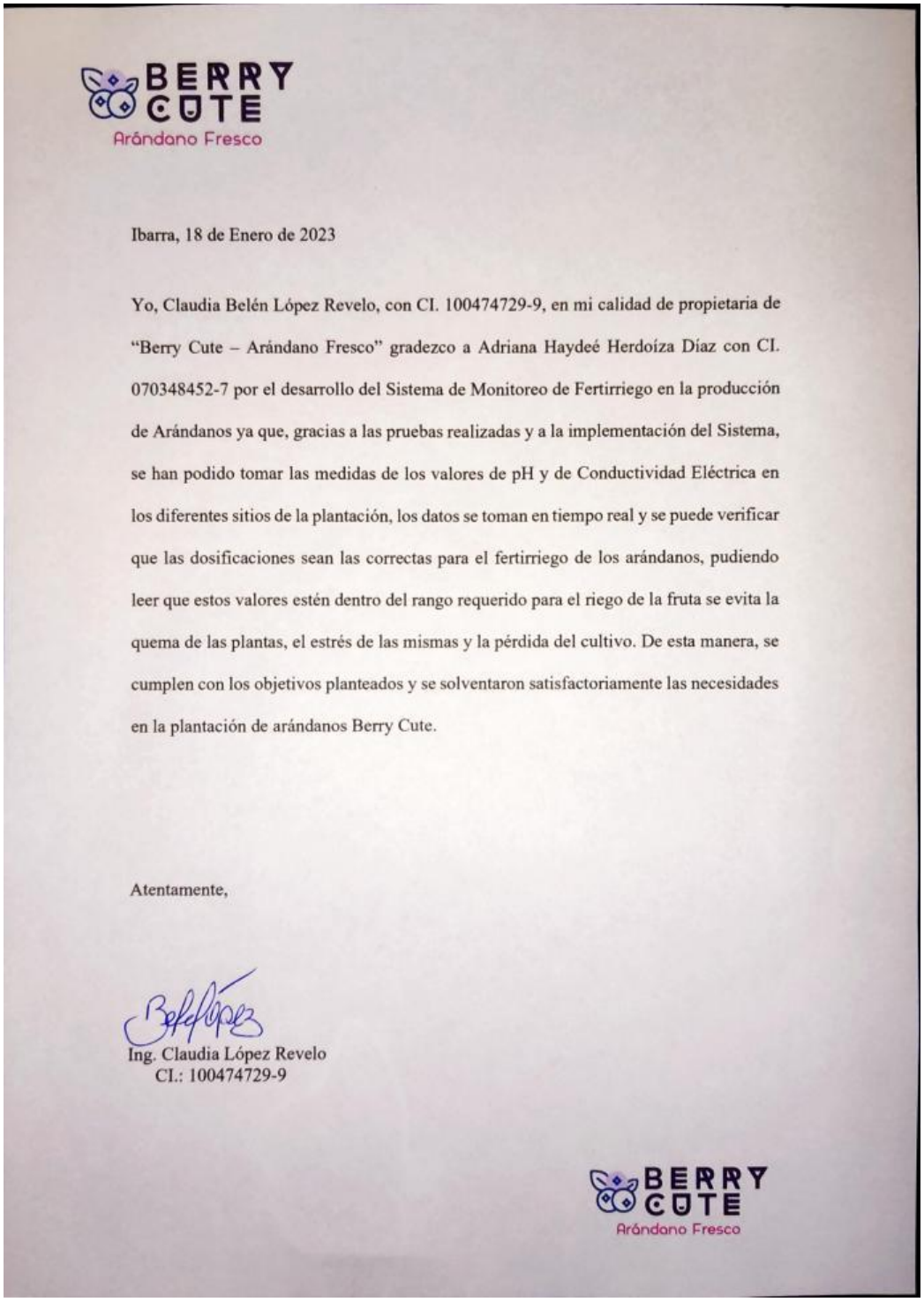
*Fuente: Autor*.

El Modo con el que se trabaja en la realización del Proyecto de Titulación es el Modo 1 en el Canal 0, por lo tanto, la Frecuencia con la que se está operando es de 863,125 MHz, mismos que se comprobaron mediante el Analizador de espectros.

**ANEXO 9: Video explicación Funcionamiento del Sistema**

Link de acceso: https://youtu.be/qRyxKvAxXJ4

**ANEXO 10: Oficio Berry Cute**



BERRY CUTE
Arándano Fresco

Ibarra, 18 de Enero de 2023

Yo, Claudia Belén López Revelo, con CI. 100474729-9, en mi calidad de propietaria de "Berry Cute – Arándano Fresco" gradezco a Adriana Haydeé Herdoíza Díaz con CI. 070348452-7 por el desarrollo del Sistema de Monitoreo de Fertirriego en la producción de Arándanos ya que, gracias a las pruebas realizadas y a la implementación del Sistema, se han podido tomar las medidas de los valores de pH y de Conductividad Eléctrica en los diferentes sitios de la plantación, los datos se toman en tiempo real y se puede verificar que las dosificaciones sean las correctas para el fertirriego de los arándanos, pudiendo leer que estos valores estén dentro del rango requerido para el riego de la fruta se evita la quema de las plantas, el estrés de las mismas y la pérdida del cultivo. De esta manera, se cumplen con los objetivos planteados y se solventaron satisfactoriamente las necesidades en la plantación de arándanos Berry Cute.

Atentamente,

Ing. Claudia López Revelo
CI.: 100474729-9

BERRY CUTE
Arándano Fresco