Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Telecomunicaciones

Diseño e Implementación de un Prototipo Basado en IoT para el Monitoreo de los Parámetros Básicos de la Calidad del Agua en la Planta Potabilizadora del Sector de Pucará

Trabajo de Integración Curricular previo a la obtención del título de Ingeniero en Telecomunicaciones.

AUTOR:

Edwin Abel Jimbo Sarmiento

DIRECTORA:

Ing. Marianela del Cisne Carrión Gonzales. Mg. Sc

Loja – Ecuador

2023

Certificación

Loja, 18 de julio del 2023

Ing. Marianela del Cisne Carrión Gonzales Mg. Sc.
DIRECTORA DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICO:

Que he revisado y orientado todo el proceso de la elaboración del Trabajo de Integración Curricular denominado: **Diseño e Implementación de un Prototipo Basado en IoT para el Monitoreo de los Parámetros Básicos de la Calidad del Agua en la Planta Potabilizadora del Sector de Pucará**, previo a la obtención del título de **Ingeniero en Telecomunicaciones**, de la autoría del estudiante **Edwin Abel Jimbo Sarmiento,** con **cédula de identidad** Nro. **1105824294**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación para la respectiva sustentación y defensa.


Firmado electrónicamente por:
MARIANELA DEL CISNE CARRION GONZALEZ

Ing. Marianela del Cisne Carrión Gonzales Mg. Sc.
DIRECTORA DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Edwin Abel Jimbo Sarmiento**, declaro ser autor del presente Trabajo de Integración Curricular denominado: **Diseño e Implementación de un Prototipo Basado en IoT para el Monitoreo de los Parámetros Básicos de la Calidad del Agua en la Planta Potabilizadora del Sector de Pucará**, y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma:

Cédula de Identidad:1105824294

Fecha: 27 de septiembre de 2023

Correo electrónico: edwin.jimbo@unl.edu.ec

Celular: 0962544629

Carta de autorización por parte del autor, para la consulta, reproducción parcial o total, y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular.

Yo, **Edwin Abel Jimbo Sarmiento,** declaro ser autor del Trabajo de Integración Curricular denominado: **Diseño e Implementación de un Prototipo Basado en IoT para el Monitoreo de los Parámetros Básicos de la Calidad del Agua en la Planta Potabilizadora del Sector de Pucará,** como requisito para optar el título de **Ingeniero en Telecomunicaciones**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los veintisiete días del mes de septiembre del año dos mil veintitrés.

Firma: 

Autor: Edwin Abel Jimbo Sarmiento

Cédula: 1105824294

Dirección: El Sagrario

Correo electrónico: edwin.jimbo@unl.edu.ec

Teléfono: 0962544629

DATOS COPLEMENTARIOS:

Directora del Trabajo de Integración Curricular:

Ing. Marianela del Cisne Carrión Gonzales **Mg. Sc.**

Dedicatoria

Dedico este trabajo a mi padre Vicente Jimbo y a mi madre Rosalía Sarmiento quienes me han apoyado incondicionalmente en mis estudios universitarios y en la vida diaria.

A mis hermanos, por estar siempre brindándome su apoyo para conseguir los objetivos académicos con honestidad y ética.

Edwin Abel Jimbo Sarmiento

Agradecimiento

Agradezco al Dios de Abraham, Isaac y Jacob por darme la vida y guiar mis pasos a lo largo de mi carrera universitaria.

A mi familia por su comprensión y apoyo incondicional.

A la Ing. Marianela Carrión, mi directora del Trabajo de Integración Curricular, por su paciencia y constancia. Sus consejos y observaciones fueron siempre útiles.

A mis maestros por haberme brindado sus conocimientos para mi desarrollo profesional y personal.

A la Unidad Municipal de Agua Potable y Alcantarillado, especialmente al personal administrativo y de servicio de la planta potabilizadora del sector de Pucara, por su colaboración para la implementación del sistema de monitoreo IoT.

Edwin Abel Jimbo Sarmiento

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Diseño e Implementación de un Prototipo Basado en IoT para el Monitoreo de los Parámetros Básicos de la Calidad del Agua en la Planta Potabilizadora del Sector de Pucará

2. Resumen

La calidad del agua es uno de los temas más importantes en la actualidad porque es el líquido vital para el ser humano, y además porque sus cualidades están siendo afectadas por la contaminación. Organizaciones nacionales e internacionales buscan mejorar la calidad mediante diferentes métodos, por ejemplo, reduciendo la contaminación, protegiendo áreas verdes o mejorando el control del saneamiento del agua en las plantas potabilizadoras. El presente trabajo está relacionado con el control de la calidad del agua en las plantas de tratamiento, a través del uso de métodos actualizados. Con lo anterior, se implementó en la planta potabilizadora de Pucará del cantón de Loja un prototipo basado en IoT para el monitoreo de los 4 parámetros básicos de la calidad del agua utilizando la red inalámbrica de baja potencia denominada LoRaWAN. Para conseguir este resultado, se realizó un proceso de diseño en donde se destaca el uso del software RadioPlanner para la simulación del diagrama de cobertura. Así mismo, se utilizaron 4 sensores para recolectar la información en una placa Arduino Uno, la misma que transmite la información por comunicación serial UART a la placa Tx TTGO V1. También, se empleó el servidor de red "TTN", el cual se comunica con la placa Tx TTGO V1 mediante la red de baja potencia LoRaWAN, y para ello se utilizó la puerta de enlace HT-M00. Finalmente, en el servidor de aplicación "TagoIO" se desarrolló un panel de monitoreo con la data del servidor de red "TTN". Como resultados generales, con el sistema de monitoreo IoT se logró recaudar una gran cantidad de información de una manera sencilla y cómoda.

Palabras clave: *calidad de agua, LoRaWAN, Arduino IDE, servidor de red, servidor de aplicación, sensores.*

2.1 Abstract

The quality of water is one of the most important issues today because it is the vital liquid for human beings, and also because its qualities are being affected by pollution. National and international organizations seek to improve quality through different methods, for example, reducing pollution, protecting green areas or improving control of water sanitation in drinking water treatment plants. The present work is related to the control of water quality in treatment plants, through the use of updated methods. With the above, an IoT-based prototype was implemented at the Pucará water treatment plant in the canton of Loja for monitoring the 4 basic parameters of water quality using the low-power wireless network called LoRaWAN. To achieve this result, a design process was carried out where the use of the RadioPlanner software for the simulation of the coverage diagram stands out. Likewise, 4 sensors were used to collect the information on an Arduino Uno board, the same one that transmits the information by UART serial communication to the Tx TTGO V1 board. Also, the "TTN" network server was used, which communicates with the Tx TTGO V1 board through the low-power LoRaWAN network, and for this the HT-M00 gateway was used. Finally, in the "TagoIO" application server, a monitoring panel was developed with the data from the "TTN" network server. As general results, with the IoT monitoring system it was possible to collect a large amount of information in a simple and comfortable way.

Keywords: *water quality, LoRaWAN, Arduino IDE, network server, application server, sensors.*

3. Introducción

La calidad del agua en las plantas de tratamiento es un tema amplio que se lo puede analizar desde los ámbitos de control y saneamiento. Esta investigación se enfoca en el control, específicamente en el aspecto del monitoreo de los parámetros de la calidad del agua. Existen varios parámetros de la calidad del agua, pero en este trabajo se enfoca en el monitoreo de los 4 parámetros básicos: temperatura, turbidez, potencial de hidrógeno (pH) y conductividad eléctrica (EC). Para el monitoreo se implementó un prototipo que utilizó la red inalámbrica de baja potencia "LoRaWAN" basada en IoT. Se ha implementado el prototipo en la planta potabilizadora del sector de Pucará del cantón de Loja, la cual no cuenta por el momento con un sistema de monitoreo actualizado. A continuación, se detalla el problema que motivo a realizar el proyecto, la justificación del mismo y por último se redactan los objetivos que se cumplieron.

3.1 Problemática

Más del 80% de las aguas residuales resultantes de la actividad humana se vierten en los ríos o en el mar sin ningún tipo de tratamiento (Samboni & Carvajal, 2007). En la Tabla 1 se presentan las enfermedades producidas por el consumo de agua contaminada en el Ecuador en los últimos años. El agua contaminada y el saneamiento deficiente están relacionados con la transmisión de enfermedades como el cólera, diarreas, disentería, hepatitis A, fiebre tifoidea y la poliomielitis. Si no hay tratamiento y saneamiento del agua, o si estos son insuficientes o están gestionados de forma inapropiada, la población estará expuesta a riesgos de salud.

Tabla 1. Casos de Etas reportados a nivel Nacional en Ecuador 2017 – 2021

Evento	2017	2018	2019	2020	2021
Hepatitis A	3499	4126	4314	1057	73
Infecciones debidas a Salmonella	2063	2680	1614	1099	117
Fiebre tifoidea y paratifoidea	1659	1476	1106	766	79
Shigelosis	560	386	248	112	12
Cólera	1	0	2	0	0
Total	**7782**	**8668**	**7284**	**3034**	**281**

Nota. Las enfermedades producidas por consumir alimentos o agua contaminada se le denomina ETAS. Recuperado del Ministerio de Salud Pública del Ecuador (2021).

En los enunciados anteriores se ha presentado de manera general las consecuencias del consumo de agua contaminada. A continuación, nos centraremos en la planta de Pucará, que es una las 3 plantas de tratamiento con las que cuenta el cantón Loja.

En la planta potabilizadora ubicada en el sector de Pucará, el monitoreo de los parámetros de calidad del agua se lo realiza de manera manual utilizando el método de "Jarras" el cual consiste en sacar muestras de cada una de las unidades para luego ser analizadas en el laboratorio. Existen 4 unidades, en la primera unidad se trata el agua cruda, en la segunda el agua decantada, en la tercera el agua filtrada y en la cuarta unidad es donde el agua está lista para ser distribuida. Los parámetros que monitorean en esta planta potabilizadora son: temperatura, color, turbidez, pH, alcalinidad, conductividad. Existen otros parámetros que también se monitorean: metales y actividad microbiológica. Los 6 parámetros antes mencionados son los principales y se los usa para el control del proceso de potabilización. El monitoreo se lo ejecuta 9 veces al día, en intervalos de 1 hora, el primer monitoreo se lo efectúa a las 8 am y el último a las 5 pm. El monitoreo manual representa un problema debido a que se lo realiza desde las 8 am hasta las 5 pm, es decir que en horas de la noche no existe monitoreo. Además, el monitoreo manual es pesado debido a que es repetitivo. El actual monitoreo es evidente que solo se lo puede hacer en la planta de potabilización, lo cual representa un problema de movilidad para el personal.

3.2 Justificación

Este proyecto colabora con el cumplimiento de los artículos 11, 12, 15, 18, 32, 58, 66, 326, 398, 411, 412, 419, 423, 426 de la Constitución de la República del Ecuador, en donde se menciona que el agua es un derecho humano constitucional, y en consecuencia todos los ciudadanos tenemos derecho a disponer de agua en cantidad y calidad. Además, el artículo 415 plantea que "los gobiernos autónomos descentralizados desarrollarán programas de uso racional del agua, y de reducción, reciclaje y tratamiento adecuado de desechos sólidos y líquidos" (Constitución de la República del Ecuador, 2008).

En este trabajo se monitorean 2 parámetros físicos que son la temperatura y la turbidez, y dos parámetros químicos que son el potencial de hidrógeno (pH) y conductividad eléctrica (EC). La cantidad de iones en el agua definen la conductividad eléctrica, este parámetro por sí solo podría medir la contaminación del agua, según el estudio realizado por la universidad de Malasia (Universiti Putra Malaysia, 2014). El pH es un parámetro del agua que es alterado por la lluvia ácida, minas e industria, el cual influye en los procesos de coagulación química,

desinfección y el control de la corrosión. Este parámetro es significativo porque puede identificar si el agua carece de nutrientes o presenta niveles de toxicidad. Un pH 6,5 es recomendable, pero un pH bajo puede incrementar la corrosión de los tubos de acero (Pérez, 2016). Por otro lado, el parámetro de turbidez está afectado por la suspensión de partículas de aguas servidas (Pradillo, 2016). La temperatura influye en la viscosidad, la solubilidad de los gases y sales, y es muy relevante para la solubilidad del oxígeno (Water Science School, 2018). Mediante estos 4 parámetros se determina la calidad básica del agua para el consumo humano. Se optaron por estos 4 parámetros debido a que son meritorios para la calidad del agua y representan 4 de los 6 parámetros monitoreados en la planta de Pucará.

Con la red IoT se logró monitorear en tiempo real los parámetros básicos de la calidad del agua, además, este sistema es más cómodo y actualizado. Mediante un dispositivo inteligente que tenga acceso a internet se consiguió observar la data de los sensores. El sistema de monitoreo se lo ha implementado en la unidad de distribución donde el agua está lista para el consumo humano.

3.3 Objetivos

Objetivo General

- Diseñar e implementar una red basada en IoT para el monitoreo en tiempo real de los parámetros básicos de la calidad del agua en la unidad de distribución de la planta potabilizadora ubicada en el sector de Pucará.

Objetivos Específicos

- Diseñar la red inalámbrica de baja potencia basada en IoT con tecnología de acceso radio “LoRa” para el monitoreo en tiempo real de la temperatura, el potencial de hidrógeno (pH), conductividad y turbidez del agua.
- Implementar y evaluar la red IoT en la planta potabilizadora del sector de Pucará.
- Presentar y analizar en tiempo real los datos obtenidos por la red mediante una API de una plataforma IoT.

4. Marco teórico

En esta sección se proporciona el marco teórico para esta investigación. Se inicia analizando antecedentes de proyectos relacionados con el presente trabajo. Luego se investiga los parámetros que determinan la calidad del agua. A continuación, se analiza temas relacionados con internet de las cosas (IoT). Finalmente, se analizan las características de LoRaWAN, las cuales fueron fundamentales en el desarrollo del proyecto.

4.1 Antecedentes

Existe gran cantidad de trabajos en los cuales se menciona la importancia que tienen los parámetros de temperatura, conductividad eléctrica (EC), potencial de hidrógeno (pH) y turbidez. Según Garcia & Perez (2022), en su trabajo titulado "The importance of water temperature in water supply systems", menciona que la temperatura afecta a los procesos físicos, químicos y biológicos involucrados en el transporte del agua potable, por lo que condiciona la calidad del agua suministrada. Por otro lado, el trabajo de investigación "Drinking water turbidity and gastrointestinal illness in the elderly of Philadelphia" realizado por Schwartz et al. (2009), se establece la influencia que tiene la turbidez en las enfermedades gastrointestinales en la población general, pero concretamente en los niños y ancianos. Asimismo, el pH es crucial para la calidad del agua porque afecta sus procesos químicos y biológicos, alterando el estado químico de muchos contaminantes como el cobre y amoniaco, modifica la solubilidad, transporte y biodisponibilidad. Las alteraciones de pH aumentan la exposición y la toxicidad de algunos metales, estos resultados se obtuvieron de la investigación realizada por la Agencia de Protección Ambiental de los Estados Unidos (EPA, 2022). Por otra parte, en el trabajo realizado por Rusydi (2018), se menciona la correlación de la conductividad eléctrica (EC) y los sólidos disueltos totales (TDS). Además, la conductividad del agua indica la cantidad de sustancias disueltas, productos químicos y minerales presentes en el agua. Se puede identificar el contaminante que ha ingresado al agua con base en los cambios significativos de la conductividad. Estos 4 parámetros son significativos en la calidad del agua, y son los que se monitorean en este proyecto.

El monitoreo de la calidad del agua es un trabajo de investigación que se ejecuta alrededor del mundo debido a su gran importancia. Se han realizado varios estudios sobre en este ámbito, por ejemplo, Vikesland (2018), realizo un estudio denominado "Nanosensors for water quality monitoring", en el cual se señala que los sensores habilitados con nanomateriales se están diseñando para aplicaciones de detección de alta eficiencia, funcionalidad múltiplex y

alta flexibilidad. Por otro lado, varios trabajos recomiendan utilizar IoT para el monitoreo, según Garrido & Peris (2022), utilizar la computación en la nube y el Internet de las Cosas (IoT) mejora la calidad de monitoreo porque se utilizan sensores de bajo consumo que facilitan las soluciones a los problemas debido a que son duraderos y fáciles de instalar. Trabajos como el de Sendra et al. (2022), incentivan al uso de IoT, en este trabajo se desarrolló una red IoT con tecnología de acceso a radio "LoRa" para el monitoreo de la calidad de agua en las zonas costeras, se crearon varios nodos y se probaron varios dispositivos para evaluar su rendimiento. Se realizaron 2 pruebas, en la prueba número uno se evaluó el correcto funcionamiento de los sensores y la arquitectura de red, y en la prueba número dos se evaluó el rango de cobertura. Por otro lado, Khatri et al. (2022) uso en su red de monitoreo de calidad de agua la tecnología de acceso a radio "Zigbee" y placas de desarrollo Raspberry Pi y Arduino. La tecnología de acceso en los diferentes proyectos varía de acuerdo a las condiciones que presenta en lugar de implementación, hasta el momento observamos que se han utilizado 2 tecnologías de acceso a radio: LoRa y Zigbee. Hay que mencionar que también se puede usar Sigfox como en el trabajo de Gennaro et al. (2018), el cual también exhibió buenos resultados en la obtención de los datos de calidad de agua. Como una posible mejor solución a los métodos tradicionales Wang & Lv (2022), propone emplear NB-IoT como tecnología de acceso a radio para mejorar los ciclos de recopilación de datos, tiempo de rendimiento, y el valor de innovación y aplicación.

En esta sección se han analizado varios estudios en los cuales se demuestra la importancia que tiene el monitoreo de los 4 parámetros básicos de la calidad del agua, y lo factible que es utilizar IoT, se pueden utilizar varias redes de baja potencia, pero en esta investigación se utilizó "LoRaWAN".

4.2 Calidad de agua

En esta sección se redacta información relevante sobre los 4 parámetros que se monitorearan mediante el sistema IoT. En ella se mencionan datos interesantes los cuales sirven para la programación y para la configuración del aplicativo IoT en TagoIO.

Según la Organización Mundial de la Salud y otros organismos internacionales mencionan que la calidad del agua representa las condiciones en que se encuentra el agua respecto a características físicas, químicas y biológicas. La calidad del agua es asociada principalmente al uso del agua para consumo humano, entendiéndose que el agua es de calidad cuando es apta para el consumo humano. Según OMS (2014), menciona que la calidad del agua se altera por la presencia de sustancias químicas o de otra naturaleza en concentraciones

superiores a las condiciones naturales. Hay tres tipos de parámetros de calidad del agua físicos, químicos y biológicos (Spellman, 2017).

La calidad del agua se puede clasificar en cuatro tipos: agua potable, agua apetecible, agua contaminada y agua infectada (Chatterjee, 1998):

- Agua potable: es seguro para beber, agradable al gusto y utilizable para fines domésticos.
- Agua palatable: es estéticamente agradable; considera la presencia de productos químicos que no representan una amenaza para la salud humana.
- Agua contaminada: es aquella agua que contiene sustancias físicas, químicas, biológicas o radiológicas no deseadas y que no es apta para beber ni para uso doméstico.
- Agua infectada: está contaminada con organismos patógenos.

4.2.1 Parámetros de monitoreo

Turbidez: según Alley (2007), la turbidez es una medida de la capacidad de la luz para atravesar el agua. La turbidez es la nubosidad del agua, la cual es causada por el material en suspensión como arcilla, limo, material orgánico, plancton y otras partículas en el agua. Es una medida de la capacidad de la luz para atravesar el agua.

La turbidez en el agua potable es estéticamente inaceptable. El impacto de la turbidez se puede resumir en los siguientes puntos:

- Aumentar el costo del tratamiento del agua (Mackenzie, 2010).
- Los microorganismos dañinos se esconden en las partículas para protegerse del proceso de desinfección (Edzwald, 2011).
- Las partículas suspendidas absorben metales pesados como mercurio, cromo, plomo, cadmio y muchos contaminantes orgánicos peligrosos como los bifenilos policlorados (PCB), los hidrocarburos aromáticos policíclicos (HAP) y muchos pesticidas (Cole, 1999).
- La turbidez se mide con un instrumento llamado turbidímetro nefelométrico, que expresa la turbidez en términos de Unidad de Turbidez Nefelométrica (NTU) o TU. Una UT equivale a 1 mg/L de sílice en suspensión. La turbidez de más de 5 NTU puede ser visible para la persona promedio, mientras que la turbidez en agua fangosa supera las 100 NTU (Apha, 2017).

Temperatura: la palatabilidad, la viscosidad, la solubilidad, los olores y las reacciones químicas están influenciadas por la temperatura (Apha, 2017). Por lo tanto, los procesos de sedimentación y cloración y la demanda biológica de oxígeno (DBO) dependen de la temperatura (Mackenzie, 2010). También afecta el proceso de biosorción de los metales pesados disueltos en agua (Sayer, 1997). La mayoría de las personas encuentran que el agua a temperaturas de 10 a 15 °C es más apetecible (Rowe, 1985).

Conductividad eléctrica: la conductividad eléctrica (CE) del agua es una medida de la capacidad de una solución para transportar o conducir una corriente eléctrica. La conductividad aumenta a medida que aumenta la concentración de iones. Las unidades de su medida son las siguientes: unidades estadounidenses (microohms/cm) y unidades del sistema internacional: milliSiemens/m (mS/m) o dS/m (deciSiemens/m).

Potencial de hidrógeno: el potencial de hidrógeno (pH) es el parámetro más importante y se define como el logaritmo negativo de la concentración de iones de hidrógeno. Es un número adimensional que indica la fuerza de una solución ácida o básica. El pH determina cuán ácida o básica es el agua. El agua ácida contiene iones de hidrógeno adicionales ($\boldsymbol{H^+}$) y el agua básica contiene iones de hidroxilo adicionales ($\boldsymbol{OH^-}$) (Alley, 2007). Como se muestra en la figura 1, el pH oscila entre 0 y 14, siendo 7 neutro. Un pH inferior a 7 indica acidez, mientras que un pH superior a 7 indica una solución básica. El agua pura es neutra, con un pH cercano a 7,0 a 25 °C. La lluvia normal tiene un pH de aproximadamente 5,6 debido al gas de dióxido de carbono atmosférico. Los rangos seguros de pH para el agua potable son de 6,5 a 8,5 para uso doméstico y las necesidades de los organismos vivos (Apha, 2017).

Un cambio de 1 unidad en una escala de pH representa un cambio de 10 veces, de modo que el agua con un pH de 7 es 10 veces más ácida que el agua con un pH de 8. Hay dos métodos disponibles para la determinación del pH: métodos electrométricos y colorimétricos. Los valores de pH excesivamente altos y bajos son perjudiciales. Un pH alto hace que el sabor sea amargo y disminuye la eficacia de la desinfección con cloro. La cantidad de oxígeno en el agua aumenta a medida que aumenta el pH. El agua con pH bajo corroerá o disolverá los metales y otras sustancias (Apha, 2017).

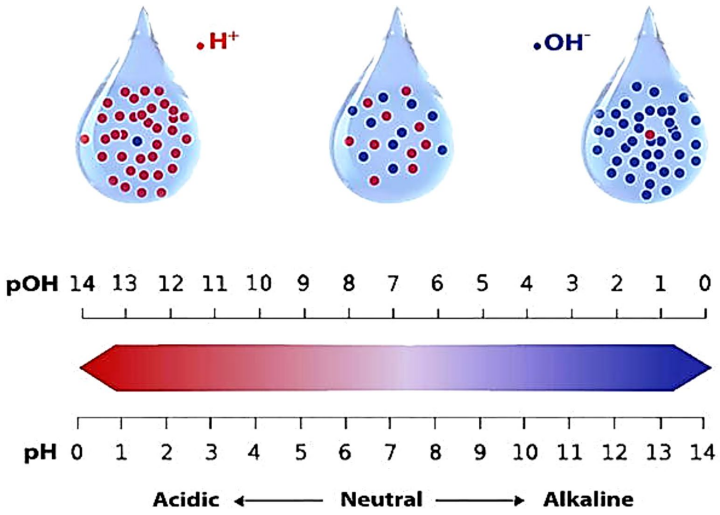


Figura 1. Potencial de hidrógeno del agua. Fuente Hassan (2019).

Nota. En la imagen podemos observar cuando el agua es ácida (exceso de iones de hidrógeno (H^+)), cuando es básica (exceso de iones hidroxilo (OH^-)) y cuando es neutra (pH de 7).

El agua con pH muy bajo o alto es fatal. Un pH inferior a 4 o superior a 10 es tóxico para el consumo. Los metales pesados como el cadmio, el plomo y el cromo se disuelven fácilmente en agua con pH muy bajo. Esto es importante porque muchos metales pesados se vuelven mucho más tóxicos cuando se disuelven en agua. Un cambio en el pH puede cambiar las formas de algunos químicos en el agua. Por ejemplo, el amoniaco en aguas neutras es inofensivo, pero a medida que aumenta el pH se vuelve peligroso (Dezuane, 1997).

4.3 Comunicación Serial

En un inicio se realizaron pruebas conectando de manera directa los sensores al módulo Tx TTGO LoraWAN pero la sensibilidad de los sensores provocaron que exista una variación significativa en la data. El Arduino uno recolecto mejor la data de los sensores, pero esta data debe ser enviada al módulo Tx para luego ser envida al servidor IoT. Es por esta razón que se optó por utilizar una comunicación serial entre una tarjeta Arduino Uno y el Módulo Tx TTGO LoraWAN. A continuación, se redacta información sobre este tipo de comunicación serial específicamente sobre la comunicación UART.

Para que los circuitos individuales intercambien información comparten un protocolo de comunicación. Los protocolos de comunicación se clasifican en dos categorías: paralelo o en serie. Ejemplos de protocolos de comunicación paralelos son ISA, ATA, SCSI, PCI e IEEE-

488. De manera similar, hay varios ejemplos de protocolos de comunicación en serie como CAN, ETHERNET, I2C, UART, SPI, RS232, USB, 1-Wire y SATA (Herrera, 2006).

Las interfaces paralelas transfieren múltiples bits al mismo tiempo. Los datos se transfieren en grandes cantidades de 1 y 0.

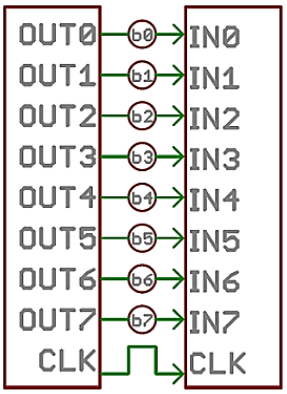


Figura 2. Ejemplo gráfico de una comunicación en paralelo. Fuente: Robots Argentina (2020).

Nota. Un bus de datos de 8 bits, controlado por un reloj, que transmite un byte cada pulso de reloj. Se utilizan 9 hilos.

Las interfaces seriales transmiten sus datos, un solo bit a la vez. Estas interfaces pueden operar con tan solo un cable y máximo con cuatro.

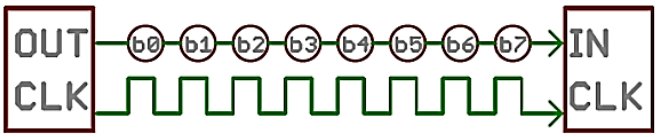


Figura 3. Ejemplo gráfico de una comunicación en serie. Fuente: (Robots Argentina, 2020).

Nota. Ejemplo de una interfaz serial con 2 cables, transmitiendo un bit cada pulso de reloj.

4.3.1 Protocolo Transmisor-Receptor Asíncrono Universal (UART)

El objetivo principal de un UART es transmitir y recibir datos en serie. Los UART transmiten datos de forma asincrónica, lo que significa que no hay una señal de reloj para sincronizar la salida de bits del UART transmisor con el muestreo de bits del UART receptor. Los datos transmitidos por UART se organizan en paquetes. Cada paquete contiene 1 bit de inicio, de 5 a 9 bits de datos (según el UART), un bit de paridad opcional y 1 o 2 bits de parada (Campbell, 2016).

A continuación, se presentan los pasos para la transmisión UART (Peña & Legaspi, 2020):

1. El UART transmisor recibe datos en paralelo desde el bus de datos.
2. El UART transmisor agrega el bit de inicio, el bit de paridad y los bits de parada a la trama de datos.
3. El paquete completo se envía en serie desde el bit de inicio hasta el bit de parada desde el UART transmisor hasta el UART receptor. El UART receptor muestrea la línea de datos a la velocidad en baudios preconfigurada.
4. El UART receptor descarta el bit de inicio, el bit de paridad y el bit de parada del marco de datos.
5. El UART receptor convierte los datos en serie nuevamente en paralelo y los transfiere al bus de datos en el extremo receptor.

Depende de que ambos lados del bus estén configurados para un reloj o una velocidad de transmisión específicos. La velocidad de transmisión es una medida del número de símbolos transferidos por segundo. Las velocidades de transmisión comunes son 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800 y 921600 baudios. Cuanto mayor sea la velocidad de reloj, más limitada será la longitud del cable. Por ejemplo, a 9600 baudios, se puede usar la longitud máxima de 15 m de cable. A velocidades de transmisión más altas, la longitud máxima del cable se reducirá (Pini, 2019).

4.4 Internet de las cosas

En esta sección se analizan temas relacionados a IoT. Se inicia analizando de manera general a IoT, luego se analizan las redes de baja potencia, específicamente “LoRaWAN”, a continuación, se indaga en sobre la comunicación serial, de manera puntual la comunicación UART, y finalmente se redacta información sobre TTN y TagoIO que son los servidores de red y de aplicación, que se utilizaron en este proyecto.

Internet de las cosas (IoT) significa una red mundial interconectada basada en sensores, comunicación, redes y tecnologías de procesamiento de la información. Hasta la fecha, hay una serie de tecnologías implicadas en la IO (internet de los objetos), como las redes de sensores inalámbricos (WSN), los códigos de barras, la detección inteligente, la RFID, la NFC, las comunicaciones inalámbricas de baja energía, la computación en la nube, etc.

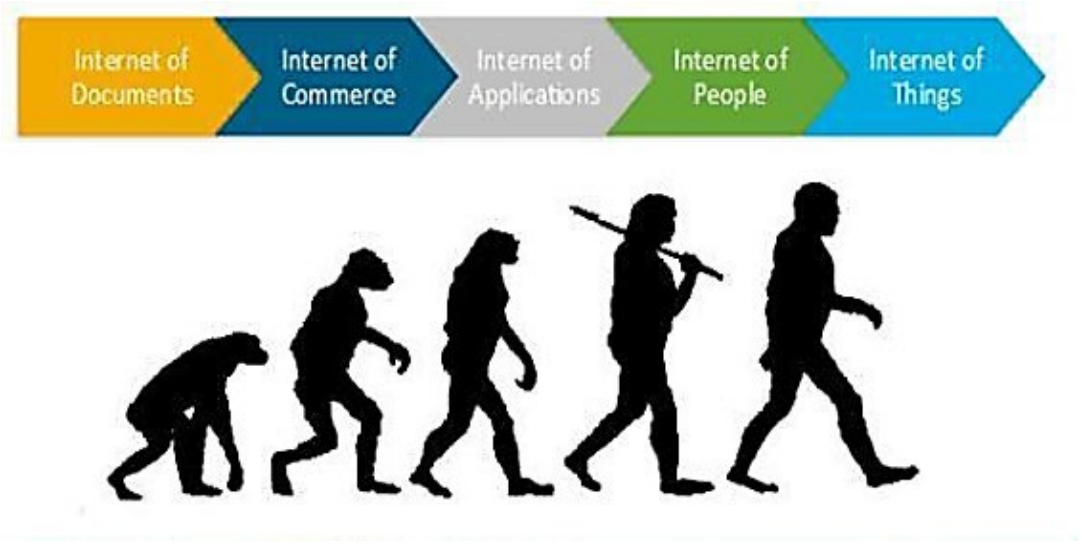


Figura 4. Evolución de la internet. Fuente: Shancang et al. (2015).

Nota. En la gráfica podemos observar la evolución de internet que abarca desde internet de documentos hasta internet de las cosas.

Un artículo publicado en Network World ha revelado las estrategias de IoT de los principales proveedores de tecnologías de la información (TI). Hewlett-Packard (HP), profesa un mundo en donde las personas están conectadas de todas las formas posibles. Cisco cree en la automatización industrial y la convergencia de la tecnología operativa. Intel se centra en dotar de inteligencia a miles de millones de dispositivos existentes. Por su parte, IBM tiene la visión de un planeta más inteligente mediante el control remoto de los dispositivos a través de servidores seguros (Violino, 2014). A pesar de tener visiones diferentes, todos coinciden en una red de dispositivos interconectados.

El diseño de la arquitectura de IoT implica muchos factores, como la red, la comunicación, los procesos, etc. En el diseño de la arquitectura de IoT, se debe tener en cuenta la extensibilidad, la escalabilidad y la operatividad entre dispositivos.

La arquitectura orientada a los servicios (SoA) trata un sistema complejo como un conjunto de objetos simples o subsistemas bien definidos. Esos objetos o subsistemas pueden reutilizarse y se mantienen individualmente; por lo tanto, los componentes de software y hardware de un IoT pueden reutilizarse y actualizarse de forma eficiente. Debido a estas ventajas, la SoA se ha aplicado ampliamente como arquitectura principal.

SoA consta de cuatro capas con funcionalidades diferenciadas:

- La capa de detección se integra con todos los objetos disponibles para percibir su estado.

- La capa de red es la infraestructura que soporta las conexiones inalámbricas o por cable, entre otras cosas.
- La capa de servicios consiste en crear y gestionar los servicios que necesitan los usuarios o las aplicaciones.
- La capa de interfaces consiste en los métodos de interacción con los usuarios o las aplicaciones

Los desafíos de seguridad de IoT se dividen en dos clases; desafíos tecnológicos y de seguridad (Mahalle et al., 2012). Los desafíos tecnológicos surgen debido a la naturaleza heterogénea y ubicua de los dispositivos IoT, mientras que los desafíos de seguridad están relacionados con los principios y funcionalidades que deben aplicarse para lograr una red segura (Leo et al., 2014).

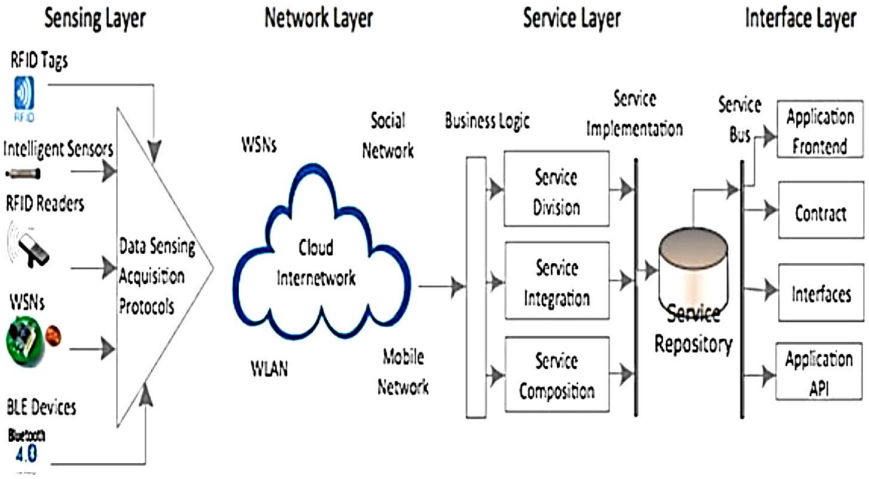


Figura 5. Arquitectura del internet de las cosas (IoT). Fuente: Hakiri (2015).

Nota. En esta imagen se representa las capas funcionales de SoA que es la arquitectura más común en IoT.

Las Redes Inalámbricas de Sensores o RIS (WSN por sus siglas en inglés) están compuestos por diversos sensores electrónicos que operan con baterías, llamados nodos sensores (motes) y que son distribuidos a lo largo de un ambiente (Chio et al., 2020). Se analizan 3 topologías de red básicas y esenciales. En una topología en estrella, cada nodo está conectado directamente y únicamente con la puerta de enlace. Una topología de árbol organiza los nodos por jerarquía. Esto significa que solo los nodos de la capa superior están conectados directamente a la puerta de enlace y los mensajes desde nodos bajos tienen que pasar por pares jerárquicamente superiores para llegar a la puerta de enlace. En una topología de malla, los

nodos se pueden configurar para tener múltiples enlaces a sus vecinos. Este enfoque da libertad a los nodos para elegir la mejor ruta para que cada paquete llegue al destino.

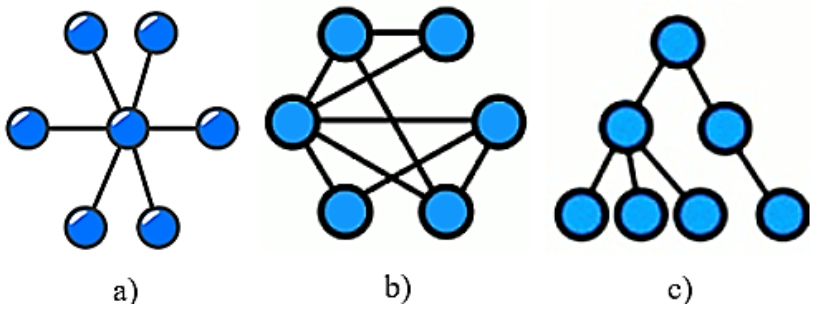


Figura 6. Topologías de red básicas. Fuente: *Ortiz & Leidy (2011).*

Nota. Topologías de red básicas: a) topología estrella, b) tipología malla, c) topología árbol.

4.4.1 Tecnologías de redes de área amplia de bajo consumo

Las tecnologías LPWAN son particularmente adecuadas para aplicaciones IoT que requieren la transmisión de mensajes pequeños un par de veces al día a largo alcance, ya que permite un rango de comunicación de hasta 40 km en áreas rurales y 10 km en áreas urbanas. Han surgido varias tecnologías LPWAN, entre ellas Sigfox y LoRaWAN en espectro de frecuencia sin licencia y NB-IoT en espectro con licencia.

La empresa Sigfox fue fundada por L. Le Moan y C. Fourtet en 2010 (Sigfox, 2022). La tecnología Sigfox se basa en la modulación inalámbrica patentada de banda ultra estrecha (UNB) en la capa física. Narrow Band-IoT (NB-IoT) es una tecnología de área amplia de baja potencia basada en los estándares desarrollados por la Asociación de Tercera Generación (3GPP) para proporcionar comunicación entre dispositivos finales y servicios mediante comunicación celular. El objetivo de NB-IoT es mejorar el consumo de energía de los dispositivos de los usuarios, la capacidad del sistema y la eficiencia del espectro. LoRaWAN es una red de área amplia de baja potencia, sus especificaciones son desarrolladas y mantenidas por la asociación abierta LoRa Alliance. Su objetivo es estandarizar la red LPWAN (LoRa Alliance, 2022). LoRaWAN basa en la técnica de modulación LoRa (LongRange).

A continuación, en la Tabla 2 se realiza una comparación entre las características principales de las tecnologías LPWAN.

Tabla 2. Tabla comparativa entre las tecnologías de baja potencia (LPWAN)

Red LPWAN Característica	Sigfox	LoRaWAN	NB-IoT
Tipo de modulación	GFSK/DBPSK	CSS	BPSK/QPSK
Frecuencia	Bandas ISM sin licencia: 868 MHz en EU 915 MHz en USA 433 MHz en Asia	Bandas ISM sin licencia: 868 MHz en EU 915 MHz en USA 433 MHz en Asia	Bandas de frecuencias LTE con licencias
Ancho de banda por canal	100 Hz en UE 600 Hz en EE. UU	125 KHz, 250 KHz, 500 KHz	200 KHz, 180 KHz
Bidireccional	Limitado/Semidúplex	Semidúplex	Semidúplex
Presupuesto de enlace	156 dB	164 dB	164 dB
Potencia de Tx	14 dBm en la UE 21,5 dBm en EE. UU	13 dBm en la UE 20 dBm en EE. UU	23 dBm
Velocidad máxima de datos	100 bps en UE 600 bps en EE. UU	50 kbps	200 kbps
Longitud máxima de carga útil	Up: 12 bytes Down: 8 bytes	243 bytes	1600 bytes
Cobertura	En urbano aprox. 10 km En rural aprox. 40 km.	En urbano aprox. 5 km En rural aprox. 20 km	En urbano aprox. 1 km En rural aprox. 10 km
Inmunidad a interferencias	Alto	Muy alto	Bajo
Autenticación y cifrado	Generación de claves, cifrado de mensajes, verificación MAC, secuencia.	AES CCM 128	NSA AES 256
Movilidad	No	Si	Si
Duración de batería	10 años	10 años	10 años
Permitir la red privada	No	Si	No

Nota. En esta tabla se realizado la comparación entre LoRaWAN, Sigfox, NB-IoT. Recuperado de: Elaboración propia basada en ETSI (2014), IMTS LoRaWAN (2018), Wang et al. (2017).

4.4.2 Beneficios de LoRaWAN

LoRaWAN tiene ventajas sobre tecnologías propietarias como Sigfox, que también carecen de capacidades críticas como movilidad y comunicaciones bidireccionales. LoRaWAN es mejor para la mayor cantidad de aplicaciones de IoT en función de su estándar abierto, su ecosistema abierto y sus características técnicas. Además, el ecosistema LoRaWAN ha visto reducciones dramáticas año tras año en costos y tiene disponibilidad global, incluida la red de Senet en 80 países.

LoRaWAN tiene el ecosistema más grande, mientras que los operadores celulares tienen problemas para encontrar dispositivos NB-IoT en volumen. Los sensores LoRa, por el contrario, están disponibles a escala y bajo costo. La duración de la batería de los dispositivos

LoRa dura más de 10 años, mientras que NB-IoT es un protocolo "hablador" que agota la batería rápidamente. Además, LoRa tiene un modelo de seguridad enriquecido basado en el cifrado de extremo a extremo, donde NB-IoT tiene un cifrado de salto por salto que es menos seguro.

NB-IoT y LTE-M ofrece velocidades de datos más altas, lo que es importante para los casos de uso con gran cantidad de datos, pero a un costo más alto y con impactos críticos, como una duración más corta de la batería del dispositivo final.

4.5 LoRaWAN

LoRa Alliance desarrollo LoRaWAN como un estándar abierto. Sin embargo, la capa física (LoRa) fue desarrollada por Semtech, que es el único productor de circuitos integrados LoRa. La tecnología LoRa es una de las nuevas tecnologías de largo alcance y baja potencia de la banda ISM (Semtech, 2022). La tecnología inalámbrica LoRa utiliza una modulación chirp-spread-spectrum (CSS) con diferentes factores de dispersión (SF) y ancho de banda. LoRa emplea las bandas ISM de 433 MHz, 868 MHz o 915 MHz, según la jurisdicción, con la banda dividida en canales. La combinación de SF y ancho de banda compensa la velocidad por el alcance (Semtech, 2020).

LoRaWAN implica una pila de protocolos con la tecnología inalámbrica LoRa como capa física. La mota LoRaWAN se comunica por aire con una pasarela que incorpora un concentrador de receptores capaz de descodificar 10 transmisiones simultáneas. Si la transmisión de una mota LoRaWAN es detectada por varias pasarelas, el servidor de red decide qué pasarela usar para enviar un acuse de recibo (si es necesario). El servidor de red pasa el paquete de datos a un "Servidor de Aplicaciones" y el servidor de aplicaciones pasa los datos al "Servidor de Clientes".

4.5.1 Arquitectura LoRaWAN

La arquitectura LoRaWAN se ha propuesto para proporcionar interoperabilidad entre las cosas inteligentes sin necesidad de complejas instalaciones locales. La arquitectura LoRaWan se compone de nodos finales, pasarelas, un servidor de red y un servidor de aplicaciones, como se presenta en la Figura 7.

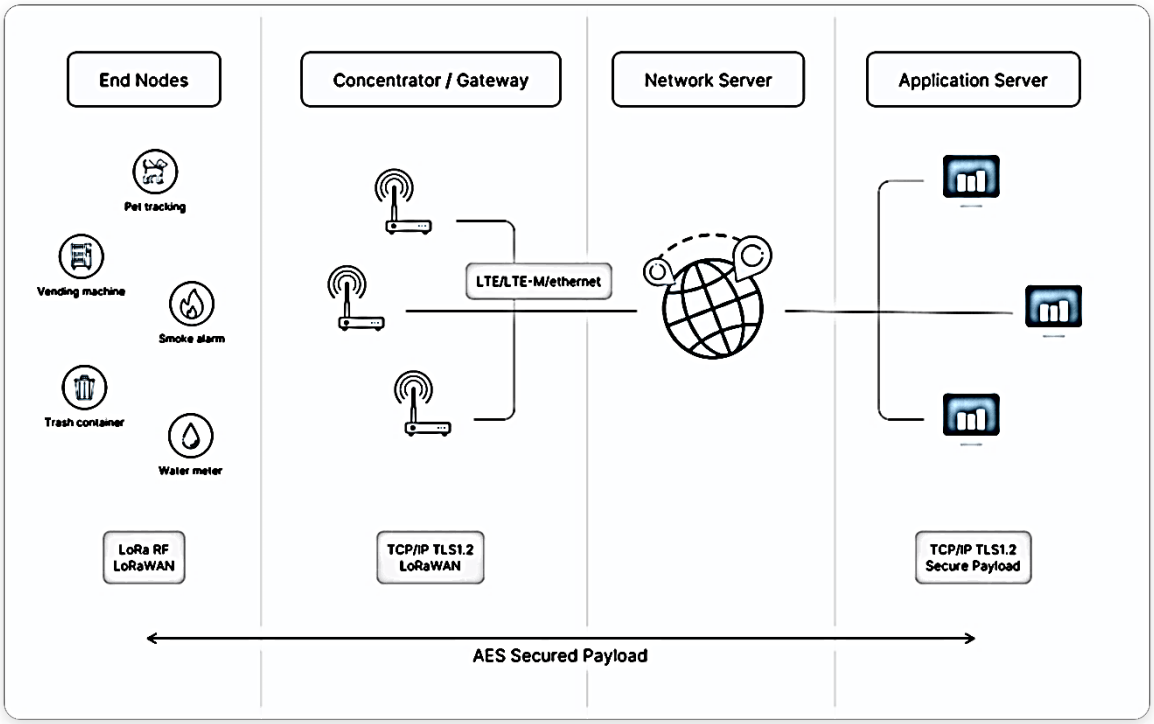


Figura 7. Arquitectura de una red LoRaWAN. Fuente: The Things Network (2021)

Nota. Las redes LoRaWAN se implementan en una topología de estrella de estrellas. Los dispositivos finales se comunican con puertas de enlace cercanas y cada puerta de enlace está conectada al servidor de red. Las redes LoRaWAN utilizan un protocolo basado en ALOHA, por lo que los dispositivos finales no necesitan emparejarse con puertas de enlace específicas.

En primer lugar, el nodo final envía los datos recogidos a una o varias pasarelas utilizando la capa física de Lora. Las puertas de enlace reciben los mensajes de dispositivos finales y los reenvía al servidor de red usando algún backhaul (wifi, celular, Ethernet o satélite). El servidor de red es la entidad inteligente que gestionará la red, realizará las comprobaciones de seguridad, llevará a cabo tasas de datos adaptativas, filtrará los paquetes recibidos redundantes, etc. Los servidores de aplicaciones son una pieza de software que se ejecuta en un servidor que es responsable de procesar de forma segura los datos de la aplicación (LoRa Alliance, 2015).

4.5.2 Parámetros regionales

Cada uno de los planes de canales regionales puede caracterizarse a grandes rasgos en uno de los dos tipos de planes: planes de canales dinámicos (modelados según el plan original EU868) y planes de canales fijos (modelados según el plan original US915).

Figura 8. Bandas de frecuencia regionales de LoRa. Fuente: TechDesing (2021).

Nota. La banda de frecuencia de funcionamiento baja hace que los módulos LoRa sean capaces de inmunizar contra el ruido.

Los planes de canales dinámicos se caracterizan por el hecho de que un pequeño número de canales por defecto y de unión están rígidamente especificados, mientras que todos los demás canales (hasta el máximo actual de 16 canales totales, incluyendo los canales por defecto) son definidos dinámicamente por la red en frecuencias arbitrarias dentro de la banda. Los planes de canales fijos suelen definirse en regiones con gran cantidad de espectro disponible para uso sin licencia. En estos planes, se define un número relativamente grande de canales en frecuencias fijas en toda la banda.

LoRaWAN opera en las bandas ISM (industriales, científicas y médicas) sin licencia. La siguiente tabla enumera los últimos planes de frecuencia y sus nombres comunes.

Tabla 3. Planes de frecuencia

Plan de canales	Nombre común
UE863-870	UE868
US902-928	US915
CN779-787	CN779
UE433	UE433
AU915-928	AU915

CN470-510	CN470
AS923	AS923
KR920-923	KR920
IN865-867	IN865
RU864-870	RU864

Nota. LoRaWAN opera en espectro de radio sin licencia utilizando frecuencias de radio más bajas con un alcance más largo. Recuperado de The Things Network (2021).

Banda ISM US902-928: a continuación, se presenta la clasificación de la velocidad de datos: DR0 - DR4 y DR8 - DR13 se utilizan para la modulación LoRa, DR4 es idéntico a DR12, y DR8 - DR13 solo se usan para mensajes de enlace descendente.

Todos los dispositivos finales US902-928 admitirán una de las siguientes opciones de velocidad de datos: DR0 – DR4 y DR8 – DR13 son conjunto de velocidad de datos mínimo requerido para obtener la certificación LoRaWAN, y DR0 – DR13 son todas las velocidades de datos se implementan en el dispositivo final.

Cuando se emplea la activación por aire (OTAA), el dispositivo final debe transmitir el mensaje de solicitud de unión en un canal seleccionado al azar de la siguiente manera: 64 canales (cada uno con un ancho de banda de 125 kHz) definidos mediante DR0 y 8 canales (cada uno con un ancho de banda de 500 kHz) definidos mediante DR4.

Tabla 4. Resumen de los parámetros relevantes de la banda ISM US902-928

Banda de frecuencia predeterminada	902-928 MHz
Frecuencias de canal obligatorias para solicitud de incorporación	Upstream: 64 canales - 902,3 - 914,9 MHz en incrementos de 200 kHz) Upstream: 8 canales - 903,0 - 914,2 MHz en incrementos de 1,6 MHz Descendente: 8 canales - 923,3 - 927,5 MHz en incrementos de 600 kHz

Tarifas de datos obligatorias para solicitud de incorporación	64 (canales de 125kHz) usando DR0 y 8 (canales de 500kHz) usando DR4
Tarifas de datos opcionales	5-6
Número de canales	Aguas arriba: 64 (125kHz) + 8 (500kHz) Descendente: 8 (500 kHz)
Canales predeterminados	Canal 0 - Canal 71
Ciclo de trabajo	Sin límite
Limitación del tiempo de permanencia	Ch0-Ch63: 400ms Ch64-Ch71: No
PIRE máx. (predeterminado) - TXPower	+30dBm
Tasa de datos RX2 predeterminada	DR8
Frecuencia RX2 predeterminada	923,3 MHz

Nota. En la tabla se han redactado los parámetros más relevantes para la configuración de los equipos. Fuente: elaboración propia.

Debe tener un conocimiento básico sobre algunos parámetros importantes que se incluyen en otros planes de frecuencia.

- CN779-787: se aplica a China. El ciclo de trabajo es <1% y no hay limitación de tiempo de permanencia. La EIRP máxima predeterminada permitida es +12,15 dBm.
- AU915-928: se aplica a Australia y todos los demás países cuya banda se extiende desde 915 a 928 MHz. No hay limitación de ciclo de trabajo aplicable y la limitación de tiempo de permanencia es de 400 ms. El EIRP máximo predeterminado permitido es +30 dBm.AS923: Aplicado para varias regiones (algunos países de Asia y Oceanía). Todos los dispositivos finales operados en Japón deben realizar Escuchar antes de hablar (LBT) según las normas ARIB STD-T108.
- KR920: las regulaciones en Corea del Sur permiten la opción de utilizar una limitación del ciclo de trabajo o la gestión de transmisión Escuchar antes de hablar Agilidad de frecuencia adaptativa (LBT AFA).
- IN865: se aplica a la India. El EIRP máximo predeterminado permitido es +30 dBm.

Hay algunas configuraciones predeterminadas recomendadas disponibles que se pueden aplicar a todas las regiones.

- Retraso RX1: 1s.
- Retardo RX2: 2s (Retardo RX1 + 1s).
- Unirse Aceptar 1 Retraso: 5s.
- Unirse Aceptar 2 Demora: 6s.

4.5.3 Seguridad LoRaWAN

LoRaWAN utiliza 2 capas de seguridad: uno para la red y otro para la aplicación (LoRa Alliance, 2017):

- Network Session Key (NwkSKey): asegura la autenticidad del nodo.
- Application Session Key (AppSKey): se emplea la encriptación AES-128 junto con el intercambio de claves utilizando un identificador IEEE EU164.

En la Figura 9 se muestra el proceso de la encriptación de los datos que inicia en el nodo Tx y culmina en el servidor de aplicación.

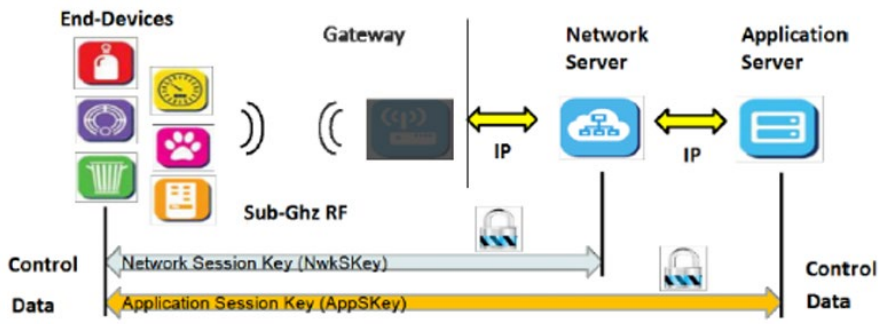


Figura 9. Encriptación de datos en LoRaWAN. Fuente: LoRa Alliance (2015).

Nota. LoRaWAN 1.0 especifica una serie de claves de seguridad: NwkSKey, AppSKeyy AppKey. Todas las claves tienen una longitud de 128 bits. El algoritmo utilizado para esto es AES-128, similar al algoritmo utilizado en el estándar 802.15.4

Cuando un dispositivo se une a la red (esto se denomina unión o activación), se generan una clave de sesión de aplicación AppSKeyy una clave de sesión de red NwkSKey. El NwkSKeyse comparte con la red, mientras que el AppSKeyse mantiene privado. Estas claves de sesión se utilizarán durante la duración de la sesión. Estas dos claves de sesión (NwkSKeyy

AppSKey) son únicas por dispositivo, por sesión. Si activa dinámicamente su dispositivo (OTAA), estas claves se vuelven a generar en cada activación. Si activa estáticamente su dispositivo (ABP), estas claves permanecerán igual hasta que las cambie.

4.5.4 Clases de dispositivos

La especificación LoRaWAN define tres tipos de dispositivos: Clase A, Clase B y Clase C. Todos los dispositivos LoRaWAN deben implementar la Clase A, mientras que la Clase B y la Clase C son extensiones de la especificación de los dispositivos de Clase A. Todas las clases de dispositivos admiten comunicación bidireccional. La especificación LoRaWAN define tres tipos de dispositivos: Clase A, Clase B y Clase C. Todos los dispositivos LoRaWAN deben implementar la Clase A, mientras que la Clase B y la Clase C son extensiones de la especificación de los dispositivos de Clase A. Todas las clases de dispositivos admiten comunicación bidireccional (enlace ascendente y descendente) (Neumann et al., 2016).

Clase A: los dispositivos de Clase A transmiten según sea necesario y crean dos ventanas de enlace descendente poco después de la transmisión. Los casos de uso común de estos dispositivos son para el monitoreo, detección y seguimiento (Mahmoud et al., 2016).

En la Figura 10 se muestran las ventanas que crean los dispositivos de clase A en el momento de la comunicación.

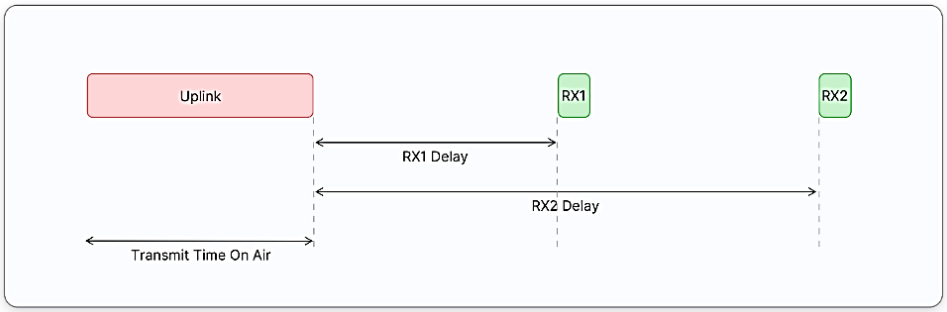


Figura 10. Ventanas de recepción Clase A. Fuente: The Things Network (2022).

Nota. Los dispositivos de Clase A consumen menos energía y tienen la latencia de enlace descendente más alta. El retraso entre el final de la transmisión de enlace ascendente y el inicio de la ventana de recepción RX1.

Clase B: los dispositivos de Clase B son similares a los de Clase A, pero añaden ranuras de recepción programadas (la programación se activa mediante la emisión de balizas desde las

pasarelas). Algunos casos de uso común de estos dispositivos son para contadores de servicios públicos e informes de temperatura (Neumann et al., 2016).

En la Figura 11 se muestra las ventanas de recepción que crean los dispositivos de clase B en el momento de la transmisión.

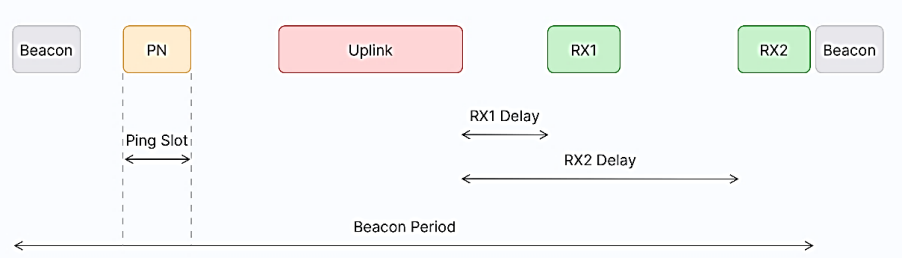


Figura 11. Ventanas de recepción Clase B. Fuente: The Things Network (2022).

Nota. Los dispositivos de Clase B se sincronizan con la red mediante balizas periódicas y tienen una latencia más baja que los dispositivos de Clase A.

Clase C: los dispositivos de Clase C escuchan continuamente cuando no están transmitiendo. La elección de la clase óptima es vital para las aplicaciones con requisitos de tiempo de respuesta o consumo mínimo de energía. Estos dispositivos comúnmente se los utiliza para contadores de servicios públicos con válvulas de corte/interruptores y luces de la calle (Mahmoud et al., 2016).

En la Figura 12 observamos las ventanas que se crean en la comunicación de los dispositivos de clase C.

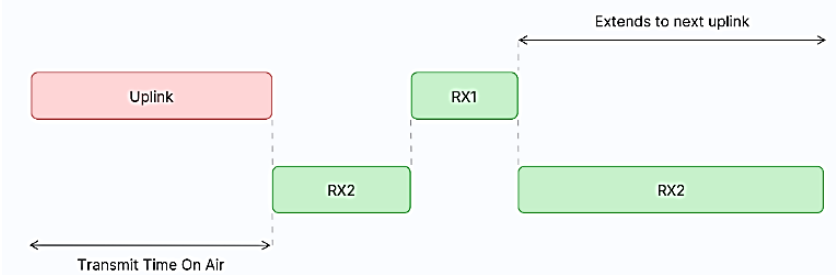


Figura 12. Ventanas de recepción de Clase C. Fuente: The Things Network (2022).

Nota. Los dispositivos de Clase C amplían la Clase A al mantener abiertas las ventanas de recepción a menos que estén transmitiendo. Esto permite una comunicación de baja latencia, pero consume muchas veces más energía que los dispositivos de Clase A.

4.5.5 Activación de dispositivo final

Se requiere un procedimiento de activación para que los dispositivos finales participen en las actividades de la red. LoRaWAN define dos procedimientos de activación: la activación por personalización (ABP) y la activación en el aire (OTAA). En el ABP, los dispositivos finales poseen en su memoria la información requerida: por lo tanto, no se requiere comunicación para unirse a la red (Milind & Prasad, 2011). En la OTAA, el dispositivo final envía una solicitud de incorporación a la puerta de enlace, que reenvía la trama al servidor de red. El servidor de red responde con un join accept, que es reenviado por la puerta de aceptación enlace (Fafoutis et al., 2014).

Activación por aire en LoRaWAN 1.0.x: en LoRaWAN 1.0.x, el procedimiento de unión requiere que se intercambien dos mensajes MAC entre el dispositivo final y el servidor de red: solicitud de unión: del dispositivo final al servidor de red y unirse-aceptar: desde el servidor de red hasta el dispositivo final

Antes de la activación, AppEUI, DevEUI y AppKey deben almacenarse en el dispositivo final. La AppKey es una clave secreta AES de 128 bits conocida como clave raíz. La misma AppKey se debe aprovisionar en la red donde se va a registrar el dispositivo final. AppEUI y DevEUI no son secretos y son visibles para todos.

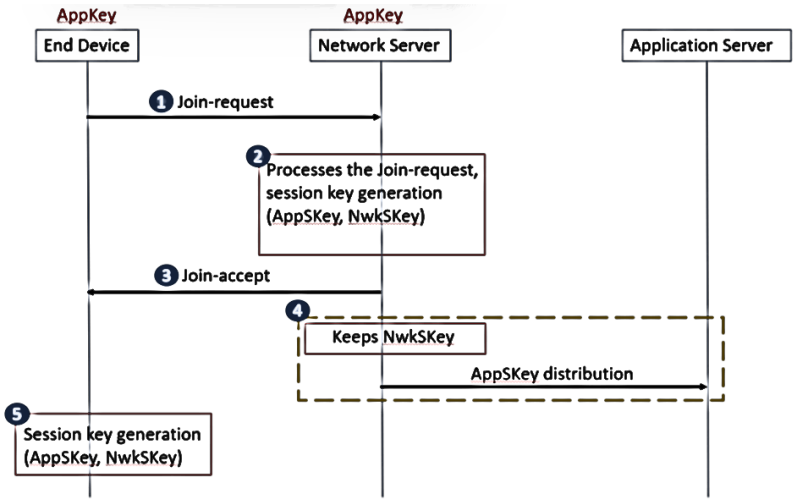


Figura 13. Flujo de mensajes OTAA en LoRaWAN 1.0. Fuente: LoRa Alliance (2017).

Nota. La AppKey nunca se envía a través de la red.

Activación por aire en LoRaWAN 1.1: en LoRaWAN 1.0.x, el procedimiento de unión requiere que se intercambien dos mensajes MAC entre el dispositivo final y el servidor

de unión: solicitud de unión: del dispositivo final al servidor de unión y unirse-aceptar: desde el servidor de unión hasta el dispositivo final.

Antes de la activación, JoinEUI, DevEUI, AppKey y NwkKey deben almacenarse en el dispositivo final. AppKey y NwkKey son claves secretas AES de 128 bits conocidas como claves raíz. La AppKey, NwkKey y DevEUI correspondientes deben proporcionarse en el servidor de unión que ayudará en el procesamiento del procedimiento de unión y la derivación de la clave de sesión. JoinEUI y DevEUI no son secretos y visibles para todos.

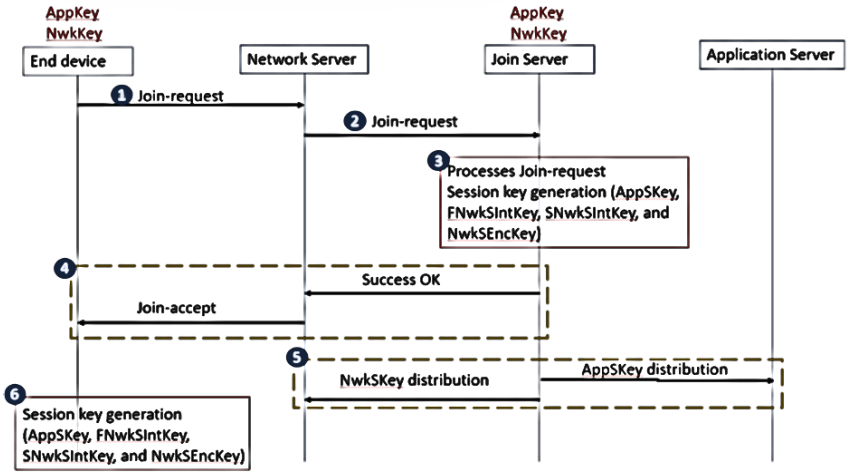


Figura 14. Flujo de mensajes OTAA en LoRaWAN 1.1. Fuente: LoRa Alliance (2017).

Nota. AppKey y NwkKey nunca se envían a través de la red. En LoRaWAN 1.1, AppEUI se reemplaza con JoinEUI. La NwkKey no se envía con el mensaje de solicitud de ingreso, y el mensaje de solicitud de ingreso no está encriptado, sino que se envía como texto sin formato.

Activación Por Personalización en LoRaWAN 1.0.x: DevAddr y las dos claves de sesión NwkSKey y AppSKey se almacenan directamente en el dispositivo final en lugar de DevEUI, AppEUI y AppKey. Cada dispositivo final debe tener un conjunto único de NwkSKey y AppSkey. El mismo DevAddr y NwkSKey deben almacenarse en el servidor de red y AppSKey debe almacenarse en el servidor de aplicaciones.

Figura 15. DevAddr y claves de sesión en LoRaWAN 1.0 (ABP). Fuente: LoRa Alliance (2017).

Activación Por Personalización en LoRaWAN 1.1: DevAddr y las claves de cuatro sesiones FNwkSIntKey, SNwkSIntKey, NwkSEncKey y AppSKey se almacenan directamente en el dispositivo final en lugar de DevEUI, JoinEUI, AppKey y NwkKey. Las mismas DevAddr, FNwkSIntKey, SNwkSIntKey y NwkSEncKey deben almacenarse en el servidor de red y AppSKey debe almacenarse en el servidor de aplicaciones.

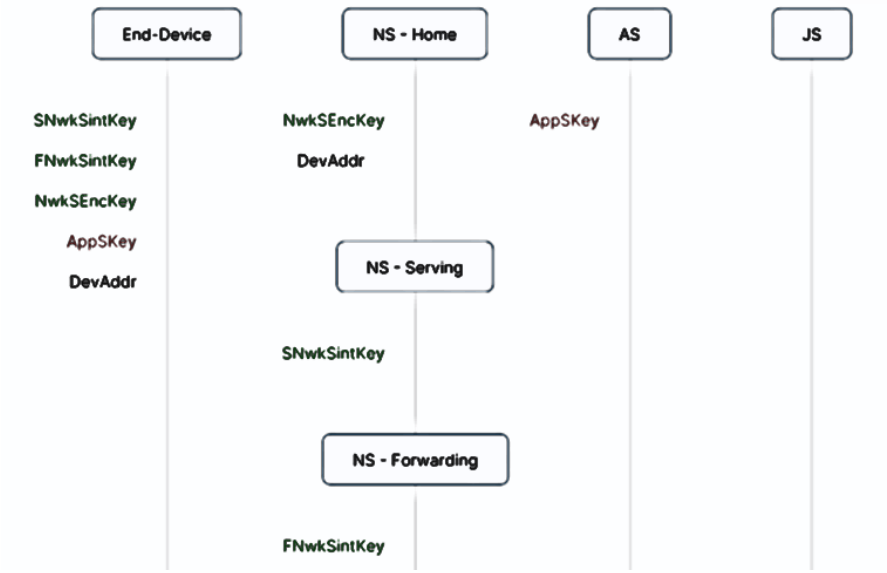


Figura 16. DevAddr y claves de sesión en LoRaWAN 1.1 (ABP). Fuente: LoRa Alliance (2017).

4.5.6 Factores de propagación

La modulación LoRa tiene un total de 6 factores de dispersión de SF7 a SF12. Los factores de propagación influyen en la velocidad de datos, el tiempo en el aire, la duración de la batería y la sensibilidad del receptor.

Tabla 5. Sensibilidad del Rx de acuerdo al factor de dispersión

Factor de dispersión	Sensibilidad del receptor para el ancho de banda fijado en 125 kHz
SF7	-123dBm
SF8	-126dBm
SF9	-129dBm

SF10	-132dBm
SF11	-134,5dBm
SF12	-137dBm

Nota. Los factores de dispersión más altos proporcionan una mayor sensibilidad del receptor. Por lo general, LoRa usa factores de dispersión más altos cuando la señal es débil. Recuperado de Petäjäjärvi et al. (2017).

4.5.7 Servidores LoRaWAN

El servidor de red conecta sensores, puertas de enlace y aplicaciones de usuario final y garantiza un enrutamiento de datos confiable y seguro a lo largo de la red LoRaWAN. Junto con el Operation Support System, son el cerebro que controla la red LoRaWAN completa. Sus principales objetivos son garantizar la seguridad, escalabilidad y confiabilidad del enrutamiento de datos a través de la red.

A continuación, se redactan los servidores más populares:

- The Things Network.
- The Things Industries.
- Everynet.
- ThingPark.
- Multitech.
- ChirpStack.
- LORIOT.

4.5.8 Plataformas IoT

Con las plataformas de IoT, los desarrolladores pueden crear aplicaciones específicamente para propósitos de IoT. Estas plataformas brindan a los usuarios la capacidad de crear, probar, implementar e iterar rápidamente en aplicaciones específicas de IoT. Una vez construidas, las empresas pueden conectar estas aplicaciones y mejorar continuamente las soluciones. Las plataformas de IoT a menudo ofrecen una funcionalidad similar a las plataformas de desarrollo de código bajo o sin código, como elementos de arrastrar y soltar. Sin embargo, la mayoría requiere algún nivel de conocimiento de codificación y las plataformas

más sofisticadas pueden requerir desarrolladores altamente calificados. Además de su funcionalidad estándar, algunos productos de plataforma en la nube como servicio pueden ofrecer la capacidad de crear aplicaciones habilitadas para IoT.

A continuación, se redactan las plataformas IoT más populares:

- Ubidots.
- AWS IoT Core.
- IBM Watson IoT Platform.
- TagoIO.
- Google Cloud Platform.
- OpenRemote.
- IRI Voracity.
- Particle.
- ThingWorx

TagoIO: ofrece la plataforma en la nube que permite a los desarrolladores crear soluciones para cualquier tipo de sensores y conectividad, incluidas redes como LoRaWAN y Sigfox.

- Visión general
  - Mejorar y crear nuevas fuentes de información.
  - Toma acciones en tiempo real integrándote con los mejores servicios.
  - Enriquezca el compromiso del cliente con su solución y marca.
- Gestión de dispositivos

  Administra cada dispositivo en función de las reglas que definirás para la operación, como el número máximo de solicitudes por hora, el tiempo de espera de entrada o el consumo de datos.
  - Gestión de tokens: cada dispositivo se autentica con un token único generado por el back-end de TagoIO. Toda la comunicación entre los dispositivos y TagoIO se realiza a través de una capa segura HTTPS.
  - La gestión del inventario: agregue etiquetas para organizar los dispositivos en grupos o simplemente para que sea más fácil encontrarlos. Esa es una gran característica ya sea que tenga 10 o 10,000 dispositivos.
  - Conectividad de modo múltiple: conéctese usando RESTful, MQTT o su protocolo propietario. TagoIO ya está conectado con varios tipos de puertas

de enlace, sensores, fuentes de datos y proveedores de red. Ya sea que su producto esté conectado a través de WiFi, Bluetooth, Sigfox, LoRa, RPMA, CDMA, 3G, satélite o cualquier otro medio, Tago se puede configurar fácilmente para admitir su aplicación.

- Gestión de datos

  TagoIO organiza todos los datos provenientes de los dispositivos en cubos. Todos los datos están disponibles en estos cubos para ser visualizados y procesados en tiempo real.

  - Compartir datos: comparta datos individuales o un conjunto de ellos en función de cómo sus dispositivos están vinculados a los cubos.
  - Copia de seguridad y retención automáticas: descargue sus datos a su controlador local u otros servidores.
  - Manipulación de datos: obtenga una imagen clara sobre la cantidad de registros y de dónde provienen. Además, puede administrar vaciar todo el depósito o simplemente eliminar variables específicas.

5. Metodología

5.1 Procesos del proyecto

A. Diseño de la red IoT: en esta sección se redactan las características esenciales de la red IoT. Se mencionan los materiales, las características de la tecnología de acceso a radio, diagrama de cobertura, servidor IoT, servidor de aplicaciones, etc.
B. Implementación de la red IoT: en este apartado se redacta la puesta en marcha de la red IoT. Se menciona la programación de los sensores, configuración del: Gateway, servidor red y del servidor de aplicación.
C. Análisis de la red IoT: en este punto se analiza la estabilidad de la red IoT y los datos que llegan al servidor de aplicación. Este apartado estará redactado en la sección de resultados.

5.2 Diseño

5.2.1 Materiales

En esta sección de materiales se redactan las características de fábrica de cada dispositivo que sirvieron para programar, conectar y configurar.

Como se lo ha mencionado en el presente trabajo, se utilizaron 4 sensores, los cuales podemos observar en la Figura 17.

Figura 17. Imágenes de los sensores que se utilizaron. Fuente: Elaboración propia.

Nota. En la imagen el literal a) representa el sensor de temperatura, el b) simboliza el sensor de turbidez, el c) es la figura del sensor de pH y el d) simboliza el sensor de conductividad eléctrica. Elaboración propia.

Sensor de Temperatura DS18B20 (ver Figura 18): detalles.

- Fuente de alimentación: 3,0 V ~ 5,5 V.
- Resolución: 9 a 12 (bits ajustables).
- Rango de visualización de temperatura: 14.0 °F a +185.0 °F (error ± 32.9 °F).
- Rango de temperatura de funcionamiento: - 67.0 °F a 257.0 °F.
- Cableado: amarillo (DATA); rojo (VCC); negro (GND).

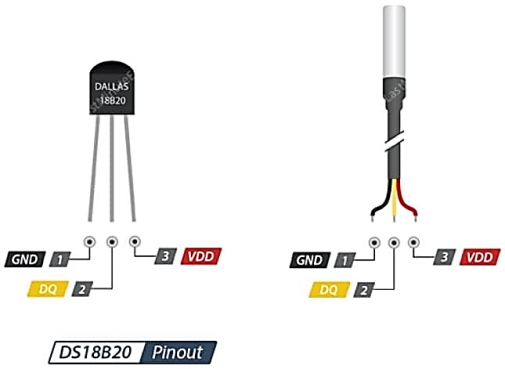


Figura 18. Pinout del sensor DS18B20. Fuente: Elaboración propia.

Sensor de pH (ver Figura 19): detalles.

- Potencia del módulo: 5,00 V.
- Tamaño del módulo: 1.693 x 1.260 in.
- Rango de medición: 0 – 14 PH.
- Temperatura de medición: 0 a 140.0 °F.
- Precisión: ± 0,1 pH (77.0 °F).
- Tiempo de respuesta: ≤ 1 min.
- Sensor de pH con conector BNC.
- Interfaz pH2.0 (parche de 3 pies).
- Potenciómetro de ajuste de ganancias.
- LED indicador de potencia.

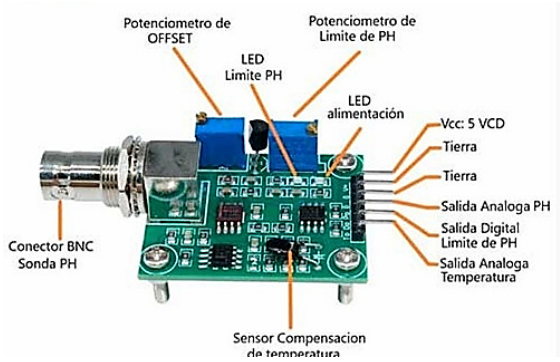


Figura 19. Pinuot de la tarjeta acondicionadora del sensor de pH. Fuente: Robocraze (2022).

Sensor de turbidez (ver Figura 20): detalles.

- Modelo: TSW-20M.
- Voltaje de trabajo: DC 5 V.
- Corriente de trabajo: 11 mA.
- Rango de detección: 0 – 4550 NTU.
- Método de cableado: VCC (G), OUT (A), GND (D).
- Temperatura de funcionamiento: 30 ~ 158.0 °F, 35 – 95 % RH.
- Peso: aprox. 0.56 oz.

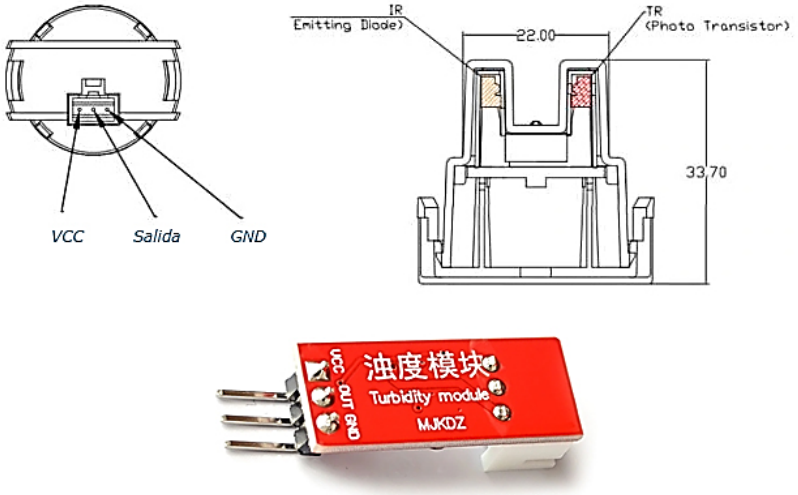


Figura 20. Pinout del sensor de turbidez. Fuente: DFRobot (2022c).

Nota. Las medidas presentadas en la imagen están en milímetros (mm).

Sensor de conductividad eléctrica (ver Figura 21): detalles.

- Voltaje de funcionamiento: 5.00 V.
- Rango de medición: 1 ms / cm - 20 ms / cm.
- Temperatura de funcionamiento: 5 – 40 °C.
- Precisión: <± 5% F.S.
- PH 2,0 Interface (3 pines SMD).
- Indicador de encendido.

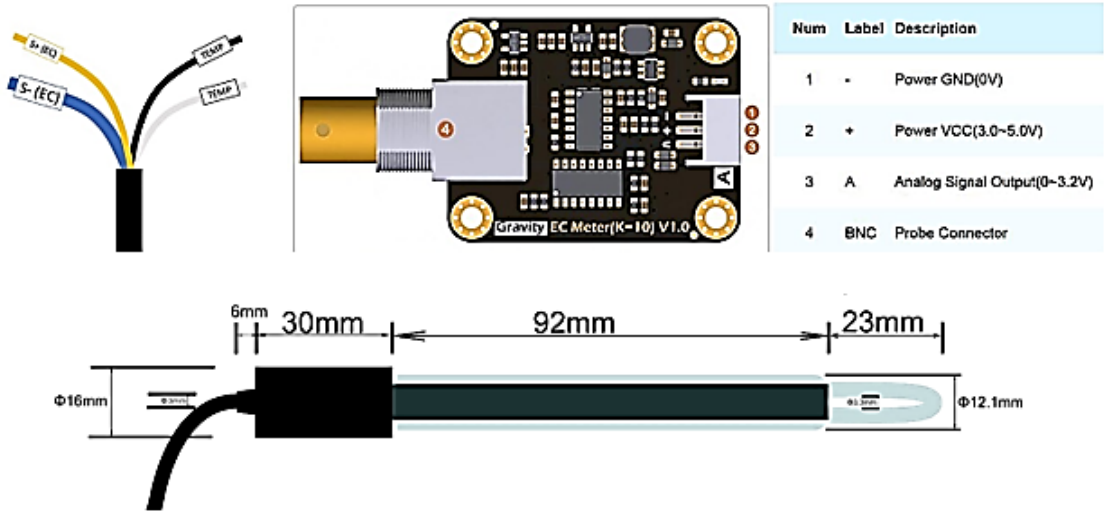


Figura 21. Pinout del sensor de conductividad eléctrica (EC). Fuente: DFRobot (2023).

Nota. En la imagen podemos observar el pinout de la tarjeta acondicionadora y algunas características físicas del sensor de conductividad eléctrica.

La placa Arduino Uno (ver Figura 23 literal a) se la utilizo para hacer un programa para recolectar la información de los sensores y mediante la comunicación serial UART enviarla al módulo transmisor (Tx). Esta placa tiene el microcontrolador ATMega328P, trabaja con 5 V, contiene 14 pines digitales y 6 pines analógicos, posee una memoria de 2KB de RAM. En esta placa se recolectó la información de cada sensor mediante funciones para crear un paquete que luego se envía al módulo Tx. En el Anexo 7 se encuentra más información sobre esta placa.

El módulo transmisor que se utilizó en el presente proyecto es el TX TTGO LoRa32 V1.0 (ver Figura 23) que se lo puede conseguir en Ecuador. A continuación, se redactan algunas características de este módulo.

Este producto es un chip SX1276 basado en ESP32 WIFI aumentado OLED, a saber, módem remoto LoRa, frecuencia de 868-915 MHz, sensibilidad superior a 148 dBm, potencia de salida de + 20 dBm, alta confiabilidad, larga distancia de transmisión. En la Figura 22 podemos observar la distribución de los pines, los cuales fueron importantes en el momento de la instalación y programación.

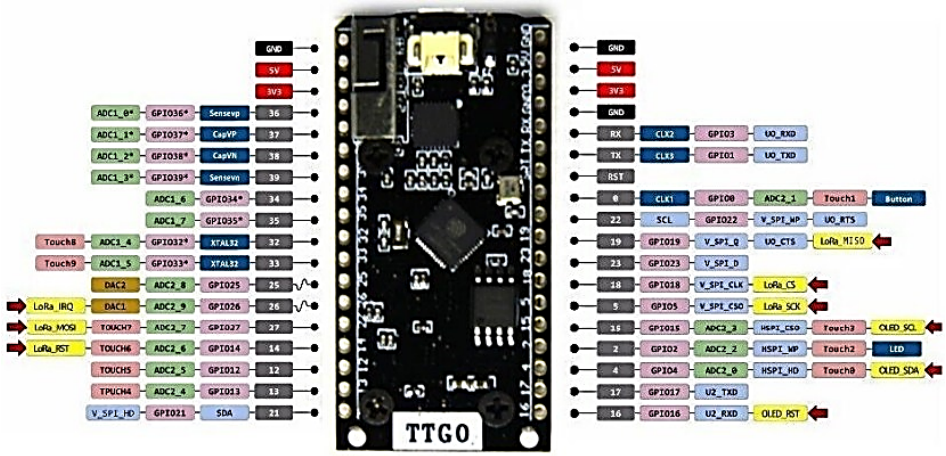


Figura 22. Pinout de la placa OLED TTGO LoRa32. Fuente: LILYGO (2022).

Nota. En la imagen podemos observar los diferentes pines con los que cuenta la placa Tx.

Detalles:

- Voltaje de funcionamiento: 3,3 – 7 V
- Rango de temperatura de funcionamiento: -40 a 90 °C (-40 a 194 °F)
- Compatibilidad con los modos de análisis de protocolo de software Sniffer, Station, SoftAP y Wi-Fi Direct
- Velocidades de datos: 150 Mbps @ 11n HT40, 72 Mbps @ 11n HT20, 54 Mbps @ 11g, 11 Mbps @ 11b

- Potencia de transmisión: 19,5 dBm @ 11b, 16,5 dBm @ 11g, 15,5 dBm @ 11n
- Sensibilidad del receptor hasta -98 dBm.
- Rendimiento sostenido UDP: 135 Mbps.

La puerta de enlace (en inglés Gateway) que se implementó en el presente trabajo es la HT-M00 (ver Figura 23). A continuación, se presentan algunas características relevantes.

HT-M00 es una puerta de enlace de doble canal. Funcionan con menos de 30 nodos LoRa. Esta puerta de enlace se basa en dos chips SX1276 impulsados por ESP32. Posee un factor de dispersión de 125 kHz SF7~SF12. Detalles:

- Certificado CE.
- Tamaño: 75x30x13mm.
- Emula demoduladores LoRa.
- Salida máxima: 18 ± 1 dBm.
- Interfaz de comunicación Tipo-C USB.
- El voltaje de la fuente de alimentación: 5 V.
- Soporte para protocolos LoRaWAN: Clase A y Clase C.

La Figura 23 representa al módulo Tx y al Gateway que se utilizaron en este trabajo. Hay que mencionar que el módulo Tx es la versión 1, y que el Gateway tiene características para uso interior.

Figura 23. Módulo Tx y Gateway. Fuente: Elaboración propia.

Nota. En la imagen el literal a) es la placa Arduino Uno, b) representa el módulo transmisor (Tx) y el c) es la figura del Gateway.

El servidor IoT que se utilizó en el presente proyecto lleva por nombre en inglés The Things Network (TTN). Este servidor es el más popular porque admite la mayoría de los Gateways, y en este caso admitió al Gateway HT-M00.

El servidor de aplicación que se usó es TagoIO, la cual presento buenos resultados en la vinculación con TTN y en el panel de monitoreo. A continuación, se presenta algunas características generales.

TagoIO es una plataforma web, 100% en la nube y de alto nivel, para monitoreo de ambientes a través de dispositivos IoT conectados a su red. Con el panel configurado y los equipos instalados, tendré acceso a varias funciones de TagoIO que me sirvieron para planear y personalizar su solución de IoT. Algunas de ellas son:

- Configuración de los dispositivos IoT.
- Configuración de un banco de datos.
- Funciones de análisis.
- Sincronizar eventos y ejecutar acciones.

5.2.1 Presupuesto

En la Tabla 6 se han redactado los materiales primordiales que se utilizaron en la implementación del prototipo en la planta de potabilizadora de Pucará. Basándome en estos materiales se ha elaborado el presupuesto económico que representa el coste del proyecto.

Tabla 6. Presupuesto sobre los recursos que se necesitó para el proyecto

Material	Costo
Sensor de pH	46 USD
Sensor de temperatura	4 USD
Sensor de turbidez	40 USD
Sensor de conductividad eléctrica	120 USD
Gateway	150 USD
Modulo Tx	35 USD
Arduino Uno	20 USD
Estructura	70 USD
Gastos adicionales	56 USD
Total	541 USD

Nota. Para este presupuesto se ha tomado en cuenta los materiales principales. Fuente: Elaboración propia.

5.2.2 Diagrama de cobertura

Para el diagrama de cobertura se utilizó el software RadioPlanner, el cual es de pago, pero tiene una versión prueba de 10 días. La versión que se utilizó es RadioPlanner 2.1. En la

Figura 24 se observa la configuración de los parámetros de red, tanto del módulo Tx y del Gateway. En el Anexo 11 se observa el reporte generado por RadioPlanner.

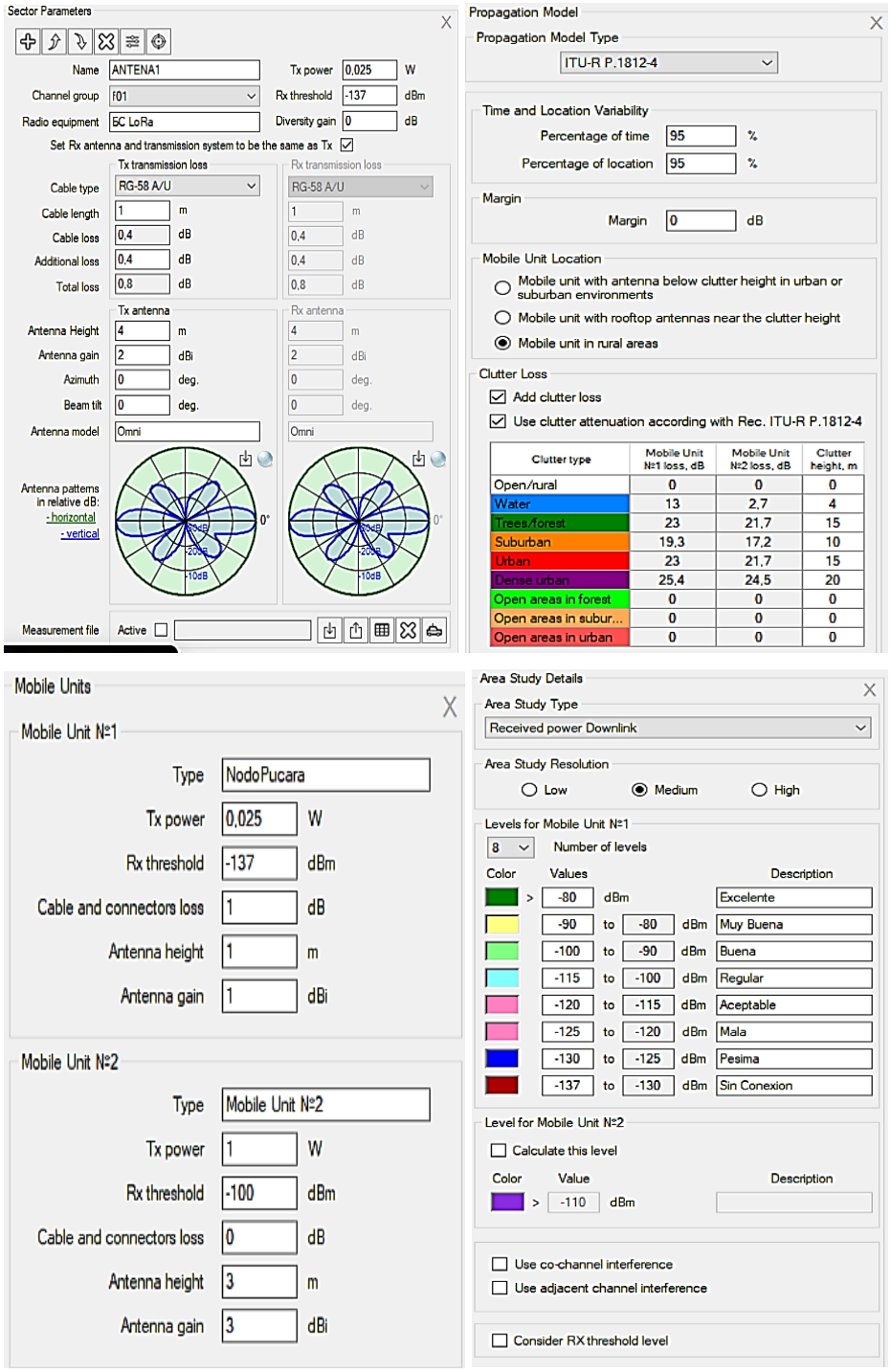


Figura 24. Parámetros configurados para el diagrama de cobertura. Fuente: Elaboración propia.

Nota. En la imagen podemos observar parámetros de configuración como la potencia de transmisión de 0.025 W, ganancia de la antena 2 dBi en estación base y en el nodo 1 dBi, la sensibilidad de -137 dBm (estándar) y en el nodo -100 dBm. Para el cálculo de las perdidas se utiliza el modelo de propagación definido en la ITU-R P1812-4. Se han creado 8 niveles, los cuales representa el nivel de potencia de la señal.

Esta herramienta sirve para planificación de radiofrecuencia (RF) en:

- Redes GSM / WCDMA / CDMA / UMTS / LTE / 5G
- Redes de radiomóvil terrestre P25 / TETRA / DMR / dPMR / NXDN / GSM-R /McWiLL.
- Redes basadas en tecnologías IoT inalámbricas: LoRa, SigFox y otras.
- Redes de transmisión de radio y televisión terrestre DVB-H / DVB-T / DVB-T2 / ISDB-T / ATSC / DAB / DAB+.
- Sistemas de comunicación aire-tierra y radionavegación operando en las frecuencias VHF, UHF y microondas (Control UAV (Drone), Radio aire-tierra, ADS-B, VOR, DME).

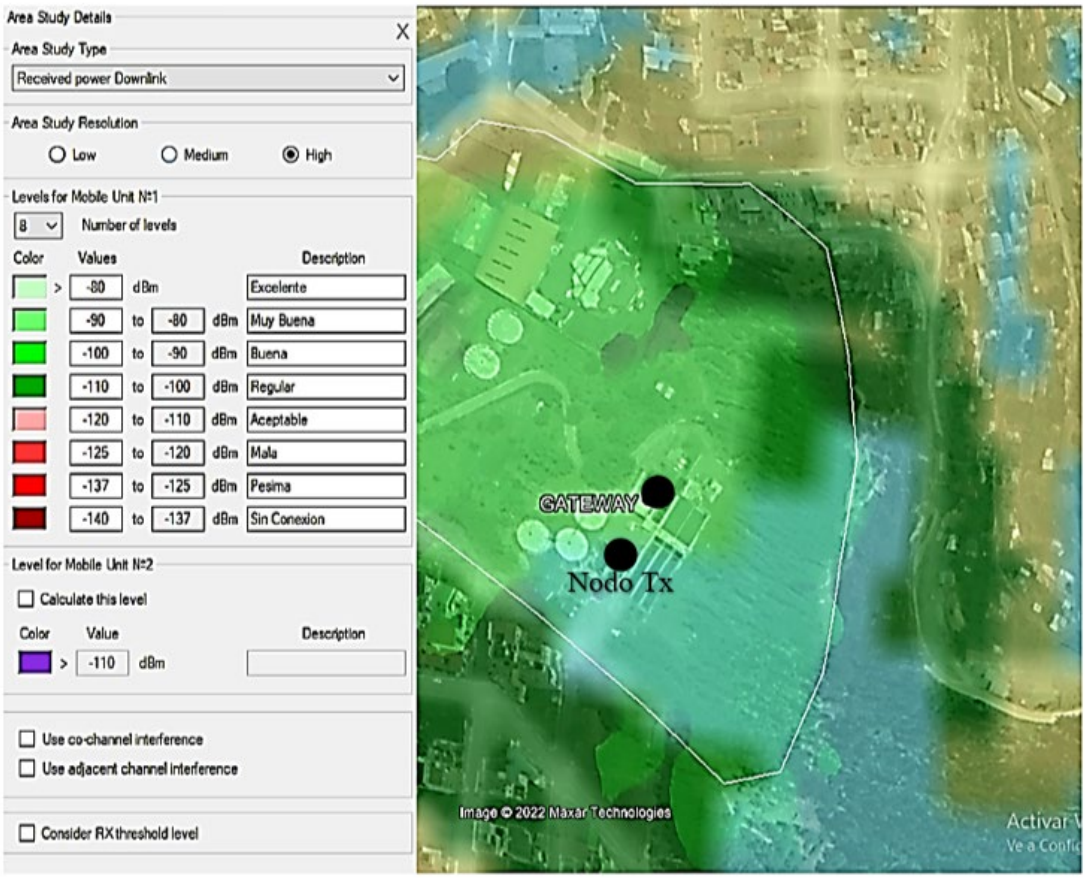


Figura 25. Diagrama de cobertura en RadioPlanner. Fuente: Elaboración propia.

Nota. En la imagen podemos observar que el nivel de potencia que irradia el Gateway en la zona donde se implementó la red IoT. Al módulo Tx que se ubicó aproximadamente a 40 m, recepta un nivel de potencia entre - 90 dBm y - 110 dBm, lo cual representa un buen nivel de potencia debido a que el Rx y el Tx emiten una potencia alrededor de los 8 dBm y tienen una sensibilidad máxima de –137 dBm. Elaboración propia.

5.2.3 Esquema de la red IoT

A continuación, se presenta un esquema gráfico de la red IoT (ver Figura 26). En este esquema podemos ver los sensores, el módulo Tx, el Gateway, el servidor de red y el servidor de aplicación.

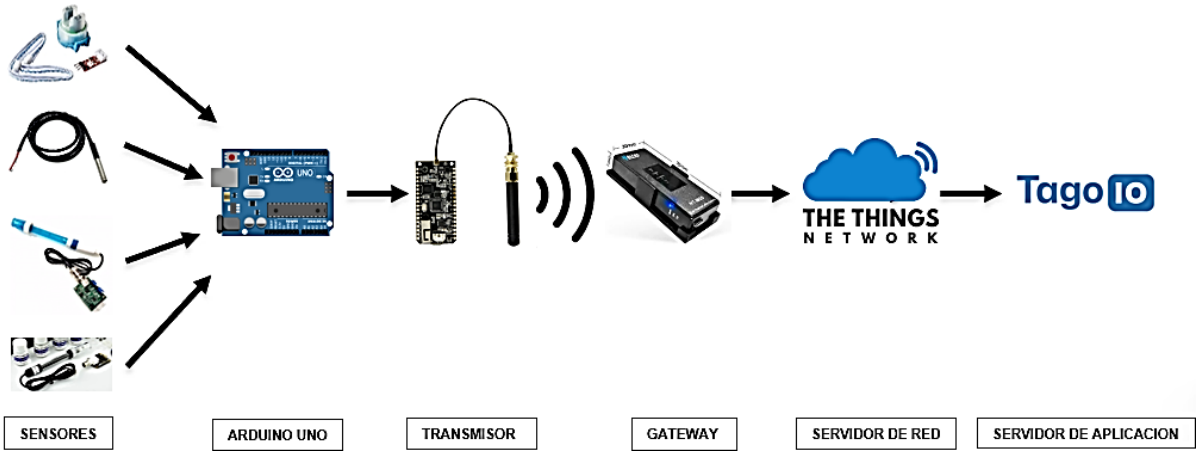


Figura 26. Esquema gráfico de la red IoT. Fuente: Elaboración propia.

Nota. Este esquema está elaborado en base a los materiales y servidores que se utilizaron en el presente proyecto.

5.3 Implementación

5.3.1 Configuración del Gateway

La configuración del Gateway HT-M00 se realizó de la siguiente manera:

1) Se encendió la puerta de enlace a través del cable de datos tipo C, luego se presionó el botón "CFG", y el botón "RST", a continuación, se soltó el botón "RST" y después, el botón "CFG".
2) Luego me conecte al wifi ("M00_X") e ingrese "192.168.4.1" a través del navegador, después inicie sesión en la página de configuración de la puerta de enlace con el nombre de usuario "HT-M00" y la contraseña "heltec.org".
3) Configuré todos los parámetros necesarios y guardé la configuración.

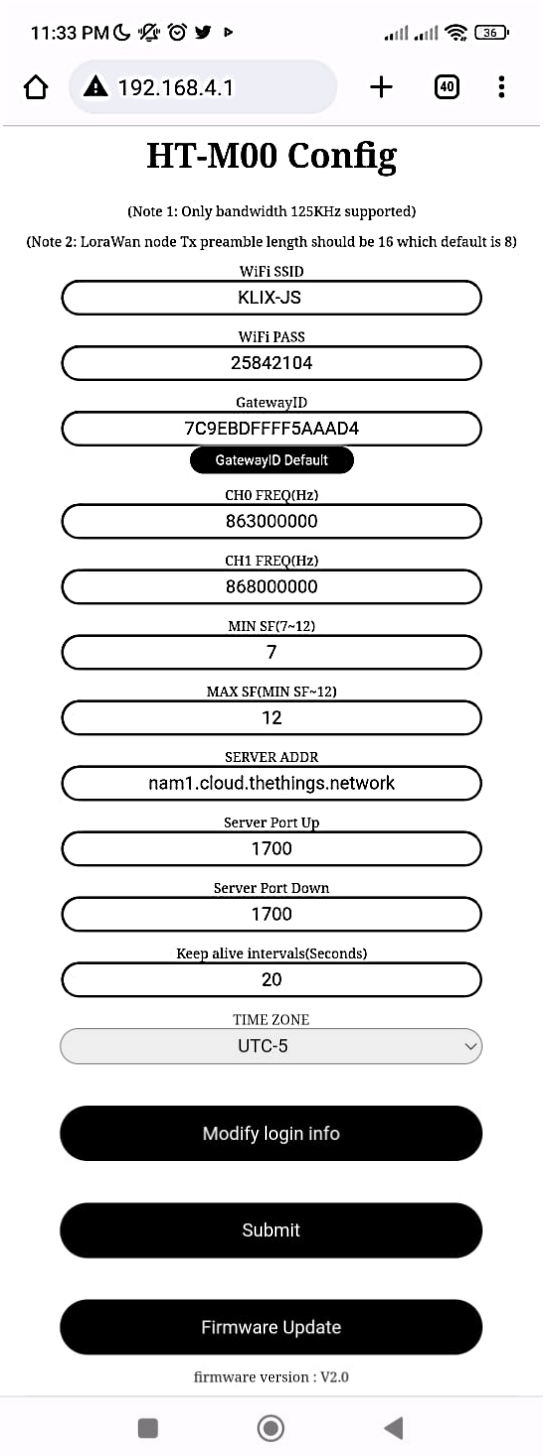


Figura 27. Parámetros que se configuraron en el Gateway HT-M00. Fuente: Elaboración propia.

Nota. En la imagen podemos observar la configuración de los canales de frecuencia, el factor de dispersión, la dirección del servidor, el puerto y la red wifi.

5.3.2 Configuración del servidor TTN

TTN es uno de los mejores servidores IoT debido a que acepta a la mayoría de Gateways. La configuración del Gateway y del servidor TTN están relacionadas en algunos parámetros que sirven de vinculación. La Figura 28 representa la habilitación del Gateway HT-

M00 en el servidor de red TTN, en esta gráfica podemos observar los parámetros del Gateway que se configuró en TTN y viceversa.

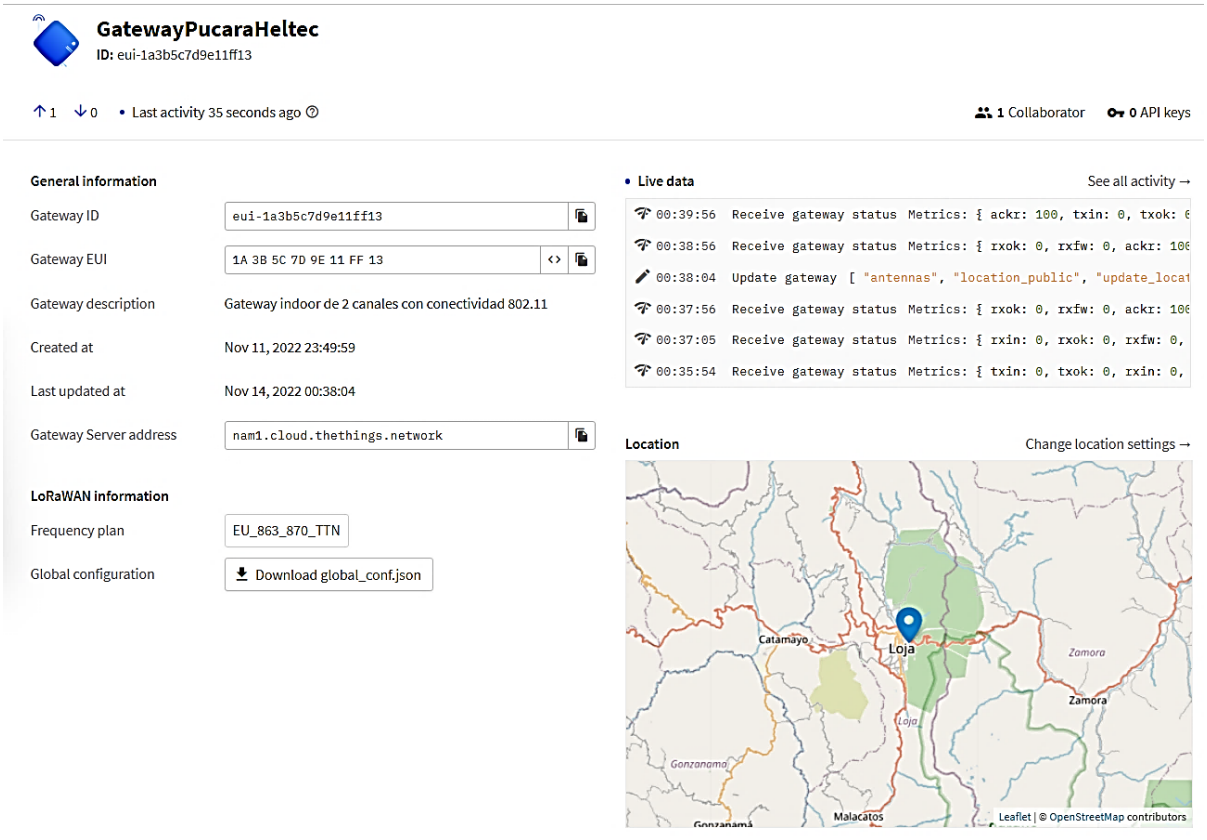


Figura 28. Parámetros del Gateway configurados en TTN. Fuente: Elaboración propia.

Nota. En la imagen podemos observar el estado, la identificación, el plan de frecuencias y la dirección del servidor al cual está conectado el Gateway.

5.3.3 Calibración de los sensores

Para la calibración de los sensores se utilizó Arduino Uno que es una placa estándar de Arduino. Arduino UNO está basado en un microcontrolador ATmega328P. El Arduino UNO incluye 6 pines de entrada analógicos, 14 pines digitales, un conector USB, un conector de alimentación y un encabezado ICSP (programación en serie en circuito). Está programado basándome en el IDE, que significa entorno de desarrollo integrado. Puede ejecutarse tanto en plataformas en línea como fuera de línea.

Sensor de temperatura DS18B20: a continuación, se presenta la Figura 29, la cual representa el diagrama de conexión del sensor de temperatura con la placa Arduino Uno.

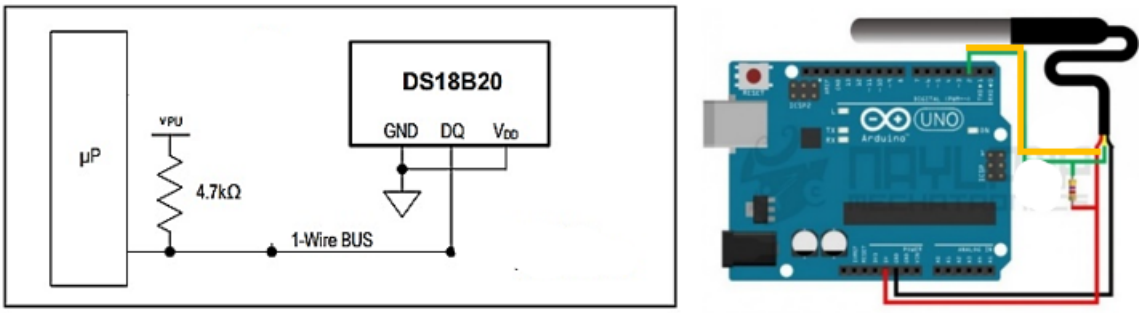
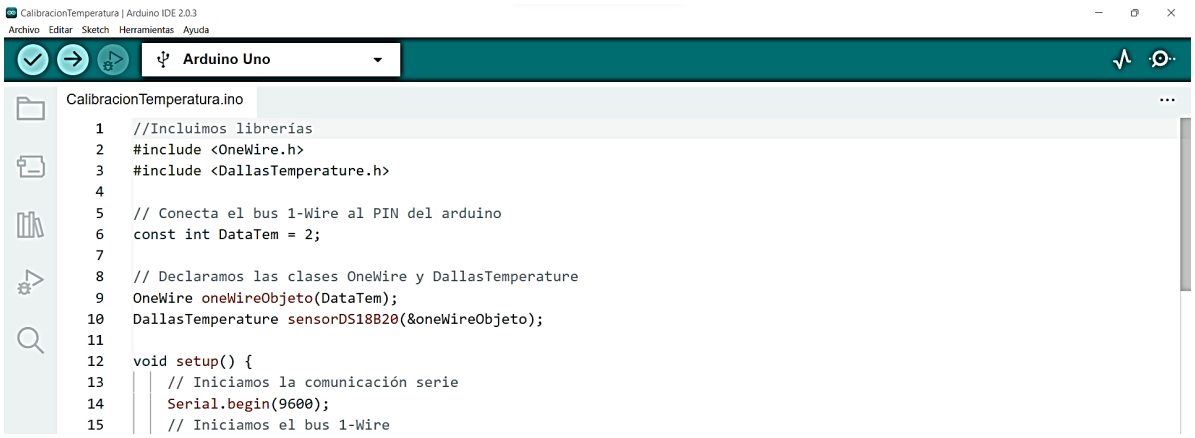


Figura 29. Conexión del sensor de temperatura al Arduino Uno. Fuente: Elaboración propia.

Nota. En la imagen observamos los 3 pines GND (negro), DQ (amarillo), VDD (rojo), que tiene el sensor de temperatura. Utilizamos una resistencia pull-up la cual nos permite establecer voltajes de reposo para asegurar una correcta lectura. La resistencia varía de acuerdo con la longitud de la sonda, en este caso como la sonda es corta (0 m – 5 m) se emplea una resistencia de 4.7 kΩ.

La codificación del sensor de temperatura se la realizo en el IDE de Arduino. Para la codificación se necesitó las librerías de OneWire y DallasTemperature. Utilice el pin número 2 para recolectar la información. La recolección de la data del sensor de temperatura se la realizo en la placa Arduino Uno y en el módulo Tx, en ambos casos el sensor recolecto información sin ningún problema. En el Anexo 1 se ha redactado información sobre el sensor de temperatura, en él se encuentran los enlaces de los sitios web que sirvieron para realizar la respectiva conexión y programación.

En la Figura 30 se observa el código de programación en el IDE de Arduino. En la imagen observamos que el código se ejecuta sin ningún problema, en este caso no fue necesario utilizar condicionales para su calibración.



```
    sensorDS18B20.begin();
}

void loop() {

    sensorDS18B20.requestTemperatures();
    // Toma de Data del sensor de Temperatura.
    Serial.println("Sensor de Temperatura_Pucara");
    // Mostrando datos en el monitor serie.
    Serial.print("Temperatura: ");
    Serial.print(sensorDS18B20.getTempCByIndex(0));
    Serial.println(" C");
    delay(2000);
}
```

```
Salida
Usando librería OneWire con versión 2.3.5 en la carpeta: C:\Users\DETPC\OneDrive\Documentos\Arduino\libraries\OneWire
Usando librería DallasTemperature con versión 3.9.0 en la carpeta: C:\Users\DETPC\OneDrive\Documentos\Arduino\libraries\DallasTemp
"C:\\Users\\DETPC\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.1-arduino7/bin/avr-size" -A "C:\\U
El Sketch usa 5564 bytes (17%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 289 bytes (14%) de la memoria dinámica, dejando 1759 bytes para las variables locales. El máximo es 20
Lín. 1, col. 1  UTF-8  Arduino Uno on 1COM7  1
```

Figura 30. Código de calibración del sensor de temperatura. Fuente: Elaboración propia.

La calibración se la realizo con base en la hoja de datos del sensor y a la información del censado de la planta potabilizadora de Pucará. A continuación, se presenta la Figura 31 en la cual podemos observar la programación del sensor de temperatura que está conectado a la placa Arduino Uno. Para el proceso de calibración se sumergió la parte pasiva del sensor en recipientes con agua con diferentes temperaturas. Para este sensor no se necesitó placa acondicionadora.

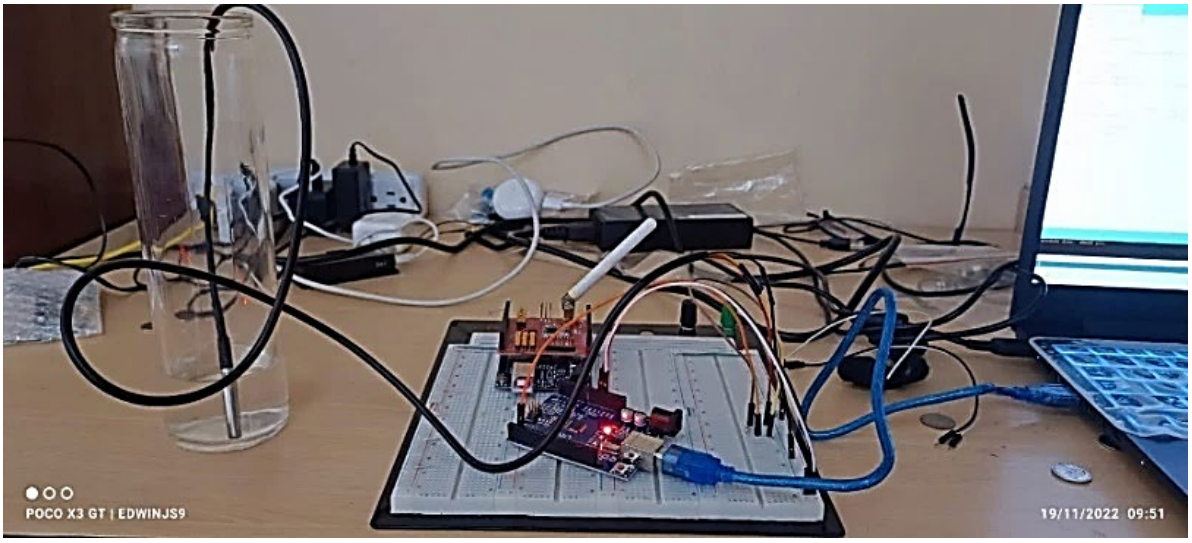


Figura 31. Calibración del sensor de temperatura. Fuente: Elaboración propia.

Sensor de turbidez: a continuación, se presenta el diagrama de conexión que se utilizó entre el sensor de turbidez y el Arduino Uno (ver Figura 32). Para este sensor de uso una placa acondicionadora, la cual sirvió para modelar la señal. La tarjeta acondicionadora es pequeña con un tamaño aproximado de 2 cm x 3 cm.

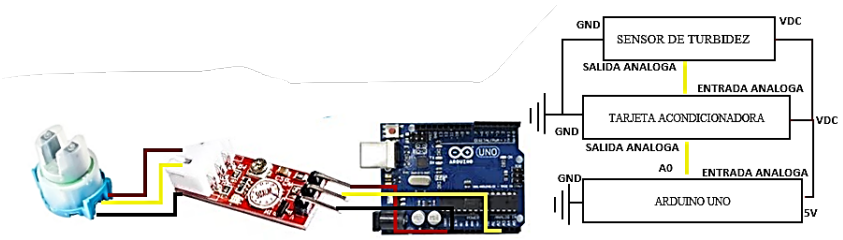


Figura 32. Conexión del sensor de turbidez y el Arduino Uno. Fuente: Elaboración propia.

Nota. El color rojo simboliza el voltaje, el color amarillo simboliza la salida analógica y el color negro simboliza tierra. El sensor trabaja con 5 V y arroja una salida analógica.

La programación se la realizo en el IDE de Arduino. Para este sensor no se necesitó una librería específica. El pin A0 se lo utilizo para leer la información enviada por el sensor, la cual es inestable y su rango de medición es limitado. Se efectuó una recolección de datos para luego obtener un promedio el cual es un valor más estable. Se han restringido los valores mediante condicionales para que estos sean más realistas.

turbidez | Arduino IDE 2.0.3

Archivo Editar Sketch Herramientas Ayuda

Arduino Uno

turbidez.ino

```
//Iniciamos las variables para leer los datos del sensor
int PinTurArd = A0;
float VolTurArd;
//Definimos el numero de muetras
int samples = 600;
float ntu; // Nephelometric Turbidity Units
void setup() {
//Inicialisamos el monitor serie
 Serial.begin(9600); // for debugging purposes
//Declaramos al PIN de entrada
 pinMode(PinTurArd, INPUT);
}

void loop() {
    VolTurArd = 0;
    //Utilizamos un ciclo for para sumar 600 muestras del voltaje arrojado por el sensor.
    for(int i=0; i<samples; i++)
    {
// Creamos una variable acumulativa
        VolTurArd += ((float)analogRead(PinTurArd)/1024)*4.5;
    }
    //Sacamos el promedio en base al muestreo
    VolTurArd = VolTurArd/samples;

     //Creamos condicionales para identificar el grado de turbidez.
    VolTurArd = round_to_dp(VolTurArd,2);
    //Para el agua potable.
    if(VolTurArd > 3){
      ntu = (((-1120.4*square(VolTurArd)+ 5742.3*VolTurArd - 4352.9))/1000);
    //Para agua limpia.
    }else if(VolTurArd > 2.45 && VolTurArd < 3.1){
      ntu = (((-1120.4*square(VolTurArd)+ 5742.3*VolTurArd - 4352.9))/1000);
   //Variacion agua limpia.
    }else if (VolTurArd > 2.35 && VolTurArd < 2.46){
      ntu = (((-1120.4*square(VolTurArd)+ 5742.3*VolTurArd - 4352.9))/1000);
  //Agua turbia
    }else if (VolTurArd > 2.29 && VolTurArd < 2.36){
      ntu = (((-1120.4*square(VolTurArd)+ 5742.3*VolTurArd - 4352.9))/1000) + 10;
    }else if(VolTurArd > 2.29 && VolTurArd < 2.36){
      ntu = (((-1120.4*square(VolTurArd)+ 5742.3*VolTurArd - 4352.9))/1000) + 50;
    }
    Serial.println("Sensor de turbidez PUCARA");
    Serial.print ("Voltaje: ");
    Serial.print ("Turbidez: ");
    Serial.print(ntu);
    Serial.println(" NTU");
    delay(2000);
}

float round_to_dp( float in_value, int decimal_place )
{
  float multiplier = powf( 10.0f, decimal_place );
  in_value = roundf( in_value * multiplier ) / multiplier;
  return in_value;
}
```

Salida

```
"C:\\Users\\DETPC\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.1-arduino7/bin/avr-size" -A "C:\\Users\\DETPC\\AppData\\Local
El Sketch usa 4298 bytes (13%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 262 bytes (12%) de la memoria dinámica, dejando 1786 bytes para las variables locales. El máximo es 2048 bytes.
```

Lín. 44, col. 33 UTF-8 Arduino Uno on 1COM7

Figura 33. Código de calibración del sensor de turbidez. Fuente: Elaboración propia.

La calibración la realicé a base de la hoja de datos del sensor y a la información del censado de la planta potabilizadora de Pucará. En la Figura 34 podemos observar la programación del sensor de turbidez que está conectado a la placa Arduino Uno. Para la programación se sumergió la parte pasiva del sensor en agua limpia y luego en agua turbia.

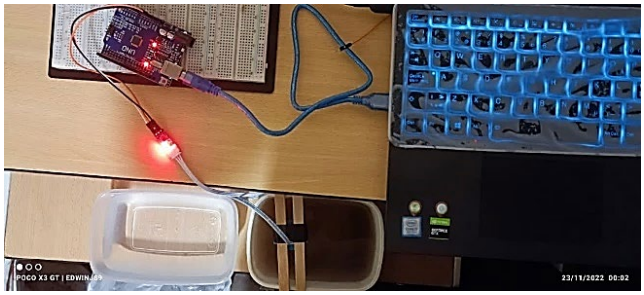


Figura 34. Calibración del sensor de turbidez. Fuente: Elaboración propia.

El sensor de turbidez presento inconvenientes al momento de trabajar en conjunto con los otros sensores. Fue necesario utilizar un módulo Tx aparte para poder estabilizar mejor la información.

Sensor de conductividad eléctrica: empleé el sensor de conductividad eléctrica (EC) de la marca DFRobot el cual consta con una placa acondicionadora. Se usó la placa Arduino Uno para recolectar la información. En la Figura 35 se observa la forma en que se conectó el sensor de EC al Arduino Uno. Como podemos observar, la tarjeta acondicionadora contiene un conector BNC en el cual se conecta la sonda, y también contiene 3, pines los cuales se conectan al Arduino Uno. Un pin se lo conecto a GND, otro a VCC y el último a la entrada analógica.

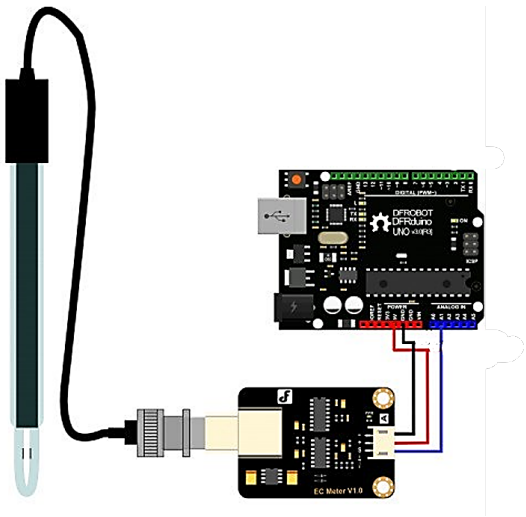


Figura 35. Esquema de conexión del sensor de conductividad eléctrica. Fuente: DFRobot (2023).

Nota. En la imagen podemos observar las conexiones entre el sensor de conductividad, la tarjeta acondicionadora y el microcontrolador. El cable de color rojo representa la conexión al voltaje de entrada, el negro tierra y el azul representa los datos enviados por el sensor.

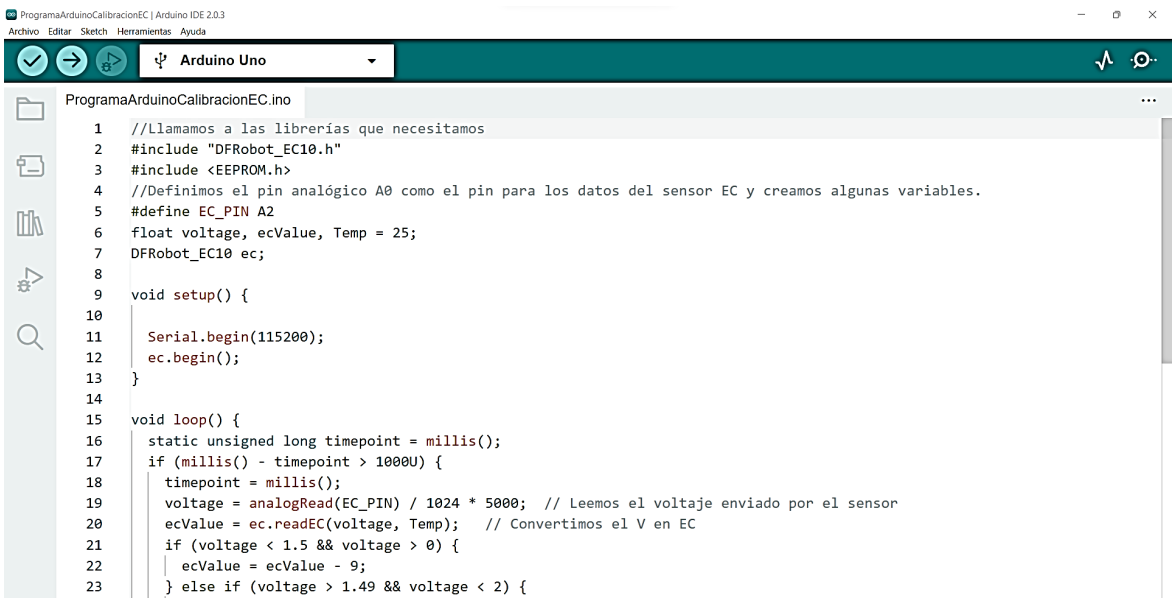
Para él condigo necesitamos la librería DFRobot_EC10. Este sensor, según su fabricante, se puede calibrar de manera automática. Seguí los siguientes pasos:

1) Subí el código de muestra en la placa Arduino.
2) Lavé la sonda con agua destilada. Inserté la sonda en la solución tampón estándar de 12,88 ms/cm, revolví suavemente hasta que los valores sean estables.
3) Cuando los valores fueron estables, pude calibrar el punto único. Los pasos específicos son los siguientes:
    a) Primero ingresé al modo de calibración mediante el comando ENTEREC en el monitor serial.
    b) Lugo ingresé los comandos CALEC para iniciar la calibración.

Cuando termine de realizar los pasos el sensor presentaba error en la calibración automática, intente varias veces y en cada una de ellas surgía el error. Es por esta razón que se implementó una calibración manual con base en restricciones.

Para la programación de este sensor se necesitó instalar las librerías DFRobot. Estas librerías ayudaron moldear y estabilizar la información enviada por el censor. Para recolectar la información se usó el pin analógico A2. Para la calibración su emplearon condicionales, se usaron la hoja de datos y la información de la planta potabilizadora de Pucará. En esta calibración se usó la variación del voltaje del sensor.

En la Figura 36 se pude observar el código de calibración en base a condicionales, se pueden observar las librerías y algunas otras características. En el Anexo 4 se ha redactado información sobre este sensor: imágenes y publicaciones. Esta información sirvió para la conexión y programación.



```
      ecValue = ecValue - 5;
    } else if (voltage > 1.99 && voltage < 2.5) {
      ecValue = ecValue;
    } else if (voltage > 2.49 && voltage < 3) {
      ecValue = ecValue + 5;
    } else if (voltage > 2.99 && voltage < 3.5) {
      ecValue = ecValue + 10;
    } else if (voltage > 3.45 && voltage < 4) {
      ecValue = ecValue + 15;
    } else if (voltage > 3.99 && voltage < 4.5) {
      ecValue = ecValue + 20;
    } else if (voltage > 4.99 && voltage < 5) {
      ecValue = ecValue + 25;
    } else if (voltage > 4.99) {
      ecValue = ecValue + 50;
    }
    Serial.print("^C  EC:");
    Serial.print(ecValue, 2);
    Serial.println("ms/cm");
  }
  ec.calibration(voltage, Temp);  // Proceso de calibración
}
```

Salida

```
Usando librería DFRobot_EC10 con versión 1.0.0 en la carpeta: C:\Users\DETPC\OneDrive\Documentos\Arduino\libraries\DFRobot_EC10
Usando librería EEPROM con versión 2.0 en la carpeta: C:\Users\DETPC\AppData\Local\Arduino15\packages\arduino\hardware\avr\1.8.6\l
"C:\\Users\\DETPC\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.1-arduino7/bin/avr-size" -A "C:\\U
El Sketch usa 5776 bytes (17%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 313 bytes (15%) de la memoria dinámica, dejando 1735 bytes para las variables locales. El máximo es 20
```

Lín. 1, col. 1 UTF-8 Arduino Uno on 1COM7 1

Figura 36. Código de calibración del sensor de conductividad. Fuente: Elaboración propia

En la Figura 37 se observa cómo se realizó la calibración del sensor de EC. Para esta calibración se introdujo la parte pasiva del sensor en recipientes con agua con distintos niveles de conductividad. Estos diferentes niveles de conductividad se los obtuvo mezclando agua con las sustancias de calibración que traía el sensor en su empaque.

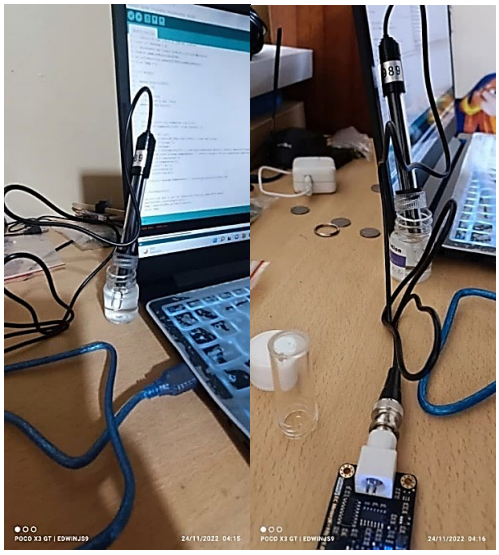


Figura 37. Calibración del sensor de conductividad eléctrica. Fuente: Elaboración propia.

Sensor de pH: para el sensor de pH se utilizó una tarjeta acondicionadora para preparar la señal. En la Figura 38 se muestra el diagrama de conexión entre el sensor de pH y el Arduino Uno. La parte pasiva del sensor se conectó a la tarjeta acondicionadora a través de la sonda que

contiene un conector BNC. El Arduino Uno se conectó a la placa transmisora mediante tres pines: VCC, GND y señal analógica. En el Anexo 3 se ha redactado información que se utilizó para la conexión y programación del sensor de pH.

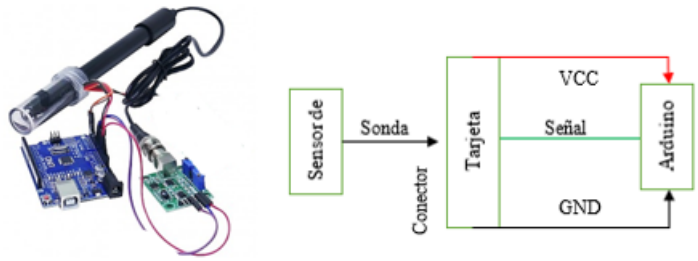
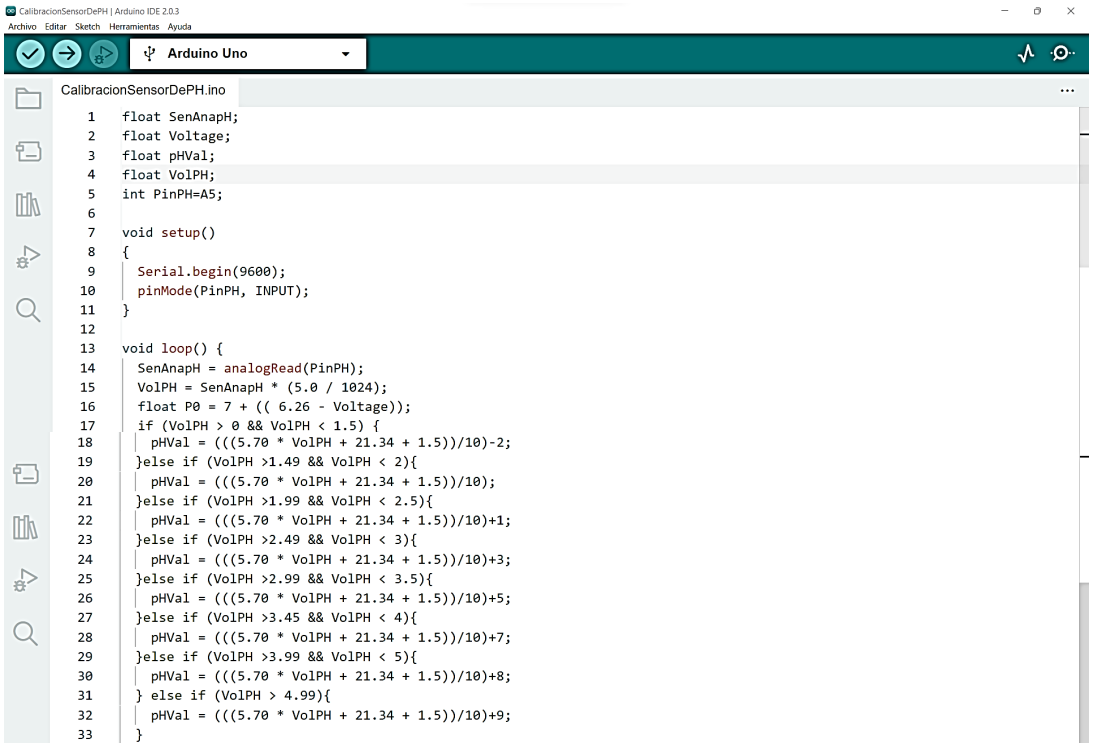


Figura 38. Conexión entre el sensor de pH, tarjeta acondicionadora y el Arduino uno. Fuente: Elaboración propia.

Nota. En la imagen podemos observar cómo conectar los pines de la tarjeta acondicionadora al módulo Arduino. La tarjeta acondicionadora se conecta con el sensor de pH a través de un conector BNC.

Para el código en el ID de Arduino no es necesario ninguna librería, lo que se realiza el leer la señal analógica enviada por el sensor, y luego con base en la señal determinar el pH. Lo que se realizó en primer lugar es recaudar varias muestras de voltaje para luego sacar el promedio, obteniendo un valor más estable. Se utilizaron condicionales para el proceso de calibración.



```
34   Serial.println("Valor del pH: ");
35   Serial.println(pHVal);
36   delay(500);
37   }
38
```

Salida

```
"C:\\Users\\DETPC\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.1-arduino7/bin/avr-size" -A "C:\\U
El Sketch usa 4344 bytes (13%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 218 bytes (10%) de la memoria dinámica, dejando 1830 bytes para las variables locales. El máximo es 20
```

Lín. 4, col. 13 UTF-8 Arduino Uno on 1COM7 1

Figura 39. Código de calibración del sensor de pH. Fuente: Elaboración propia.

En la Figura 40 se observa la calibración del sensor de pH. Para este proceso se introdujo el sensor en recipientes que contenían agua con distintos valores de pH para luego en base a la programación de los condicionales estabilizar el sensor. La variación de los niveles de pH se la realizo mezclando agua con sustancias como: limón (pH=2), leche (pH=6), bicarbonato (pH=9) y lejía (pH=13).

Figura 40. Calibración del sensor de pH. Fuente: Elaboración propia.

5.3.4 Comunicación serial UART entre el Arduino Uno y el módulo Transmisor

Se realizó la comunicación serial UART (Transmisor-Receptor Asíncrono Universal) para enviar la data del Arduino Uno al módulo Tx. El diagrama de conexión lo podemos observar en la Figura 41. Se conectó el pin Tx del Arduino Uno con el pin Rx del módulo, y el pin Tx del módulo con el pin Rx del Arduino Uno.

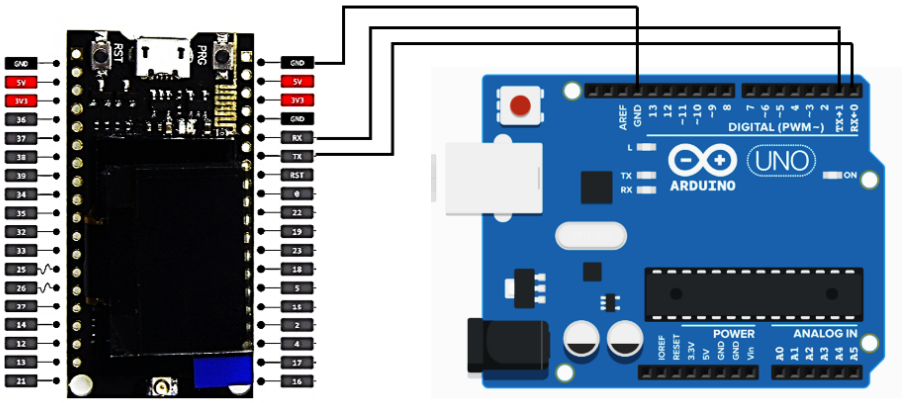


Figura 41. Conexión entre el Arduino Uno y el módulo Tx. Fuente: Elaboración propia.

Nota. En la figura se puede observar el diagrama para la comunicación UART entre el módulo Tx y el Arduino Uno.

Programación en el Arduino Uno: en la placa Arduino Uno se realizó el programa para recoger la información de los sensores y mediante la comunicación UART estos datos son enviados al módulo Tx.

En la sección de calibración de los sensores están los códigos de programación de cada uno de los sensores, cada uno de estos códigos fueron ingresados en una función para facilitar el uso de sus datos. A continuación, se presenta el proceso que se realizó para el enviar los datos desde el Arduino Uno al módulo Tx. El código completo lo podemos encontrar en el Anexo 8.

```
void loop() {
  int Control = 10;
  datos [0]=Control;

  int Tra = ((int)(Temra)*100);
  datos [1] = Tra;

  //Turbidez
  float UNTU= readTur();
  int Tudez = ((int)(UNTU)*100);
  datos [2] = Tudez;

  //ConductividadElectrica.
  float UCON = readEC();
  int Condad = ((int)(UCON)*100);
  datos [3] = Condad;

  //PotencialDeHidrogeno
  float UPH = readPH();
  int PoHi = ((int)(UPH)*100);
  datos [4] = PoHi;

  String txt = "";

  for (int x = 0; x<5; x++){
    txt += datos [x];
    txt += Ide [x];
  }

Serial.println(txt);
delay(1500);

}
```

Figura 42. Programa para enviar los datos de los sensores del Arduino Uno al módulo Tx.

Fuente: Elaboración propia.

Nota. En el código se agregaron valores de control y letras para poder identificar los datos de los sensores en el módulo Tx.

La programación del módulo Tx contiene varios parámetros que engloban algunas funciones, librerías, etc. Los parámetros de vinculación están relacionados con la información de sesión y de activación, la cual es la misma tanto en TTN y en el programa del módulo Tx. Para este trabajo se realizó una activación denominada OTTA (activación por aire) que es más segura que la activación ABP. Para proceder con la vinculación OTTA entré en módulo Tx y el aplicativo en TTN se necesitó la siguiente información: un identificador de dispositivo, un identificador de aplicación y una clave de aplicación. En la Figura 43 observamos la programación de la vinculación OTAA en el módulo Tx en el IDE de Arduino.

```
static const u1_t PROGMEM APPEUI[8]={ 0xE3, 0xD5, 0xB1, 0xC9, 0xD8, 0xA7, 0xC6, 0xE5 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={ 0x11, 0x77, 0x05, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
static const u1_t PROGMEM APPKEY[16] = { 0x47, 0xB3, 0x3E, 0x19, 0x47, 0x92, 0x83, 0x55, 0x49, 0xB3, 0x04, 0x00, 0xAE, 0xDE, 0x23, 0x1E };
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Figura 43. Información de vinculación OTAA programada en el IDE Arduino. Fuente: Elaboración propia.

Nota. Es muy importante tener en cuenta que la información de APPEUI y DEVEUI tiene que estar en formato LSB (bit menos significativo), mientras que la información de APPKEY tiene que estar en formato MSB (bit más significativo).

La asignación de pines es otro parámetro que hay que tener en cuenta en la programación del módulo Tx. El mapeo para estos modelos varía dependiendo de la versión del módulo. En la Figura 44 se observa el mapeo que se utilizó para el módulo ESP32 TTGO V1 LoRaWAN.

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 33, 32},
};
```

Figura 44. Programación de los pines del módulo Tx. Fuente: Elaboración propia.

Luego de programar estos parámetros, se necesita receptar los datos enviados por el Arduino Uno a través de la comunicación serial UART. En primer lugar, se inicializó la comunicación serial de acuerdo a los pines asignados para Tx y Rx, luego se receptó, decodifico

y separo los datos de los sensores. Para mejorar la manipulación de los datos para armar el payload se realizó el uso de funciones para cada uno de los sensores.

En la Figura 45 se observa el código de programación en el IDE de Arduino que se usó para decodificar la información del sensor de temperatura. Lo primero que se realizó es determinar el tamaño de la información y sus límites, luego se creó una variable tipo vector "String" con la información. Finalmente, se transforma esa información en una sola cadena de texto para luego convertirla en número. Este proceso se lo realizo para todos los sensores, y el código de programación se encuentra en el Anexo 9.

```
int readTem() {

//leemos la informacion enviada por el Arduino Uno
data = Serial2.readString();

int ConTem = 0;
int ConControl = 0;

//Determinaños el tamaño de la iformacion del sensor de temperatura
for(size_t i = 0; i<data.length(); i++){
   if (data.charAt(i) == 'z'){
     ConControl = i;
   }else if (data.charAt(i) == 'a'){
     ConTem = i-ConControl-1;
   }
 }

//Crear un string del tamaño de la infomacion
char Tem [ConTem];

//Llenamos el arreglo con la informacion
int a=2;
for (int i = 0; i<ConTem;i++){
  a++;
  Tem[i]=data.charAt(a);
}

//StringSensores
String Temx = "";

//Unir arreglos
for (int x = 0; x<ConTem; x++){
  Temx += Tem [x];
}

//Transformamos el texto a numero
numTem = Temx.toInt();

//Retornamos el numero
return numTem;

}
```

Figura 45. Función para receptar en el módulo Tx los datos del sensor de temperatura. Fuente: Elaboración propia.

Una vez que se terminó de programar las funciones para receptar los datos de los sensores, se procedió a armar el payload para enviar al servidor de red TTN. Se llamó a la función de cada sensor para guardar sus datos en un arreglo para ser enviado. En la Figura 46 se observa el programa desarrollado en el IDE de Arduino para el módulo Tx. El código completo lo encontramos en el Anexo 9.

```
//Llamamos a la funcion de cada sensor
int ValTem = readTem();
payload [0]= ValTem >> 8;
payload [1];
int ValTur = readTur();
payload [2]= ValTur >> 8;
payload [3];
int ValPo = readPH();
payload [4]= ValPo >> 8;
payload [5];
int ValCon = readCon();
payload [6] = ValCon >> 8;
payload [7];
LMIC_setTxData2(1, payload, sizeof(payload), 0);
Serial.println(F("Packet queued"));
```

Figura 46. Carga útil a enviar por el módulo Tx. Fuente: Elaboración propia.

5.3.5 Servidor de red

Una vez configurados el Gateway en TTN se creó una aplicación para el nodo Tx. A continuación, en la Figura 47 se observa la conexión entre el módulo Tx y el servidor TTN. En este apartado también se configuró la activación OTTA para que exista una correcta vinculación con el nodo Tx.

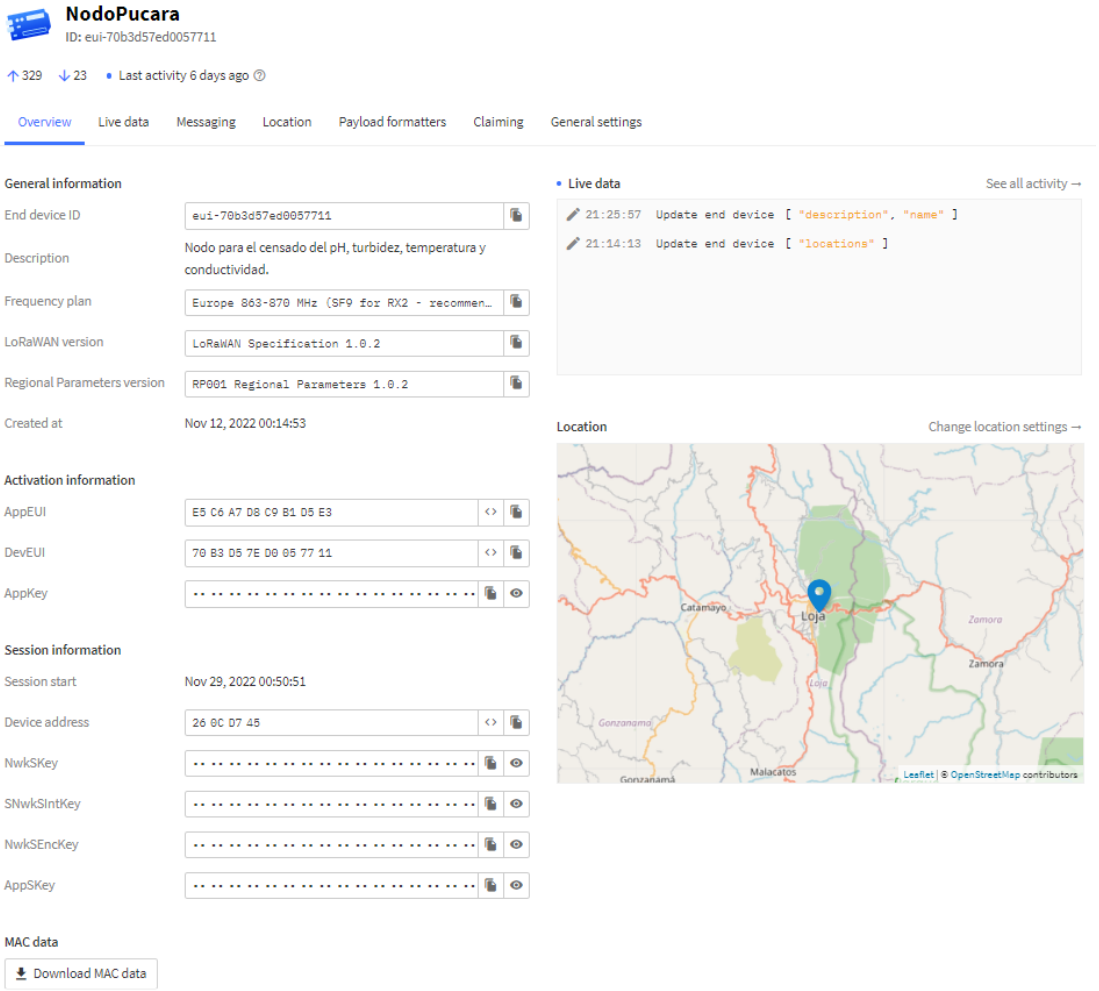
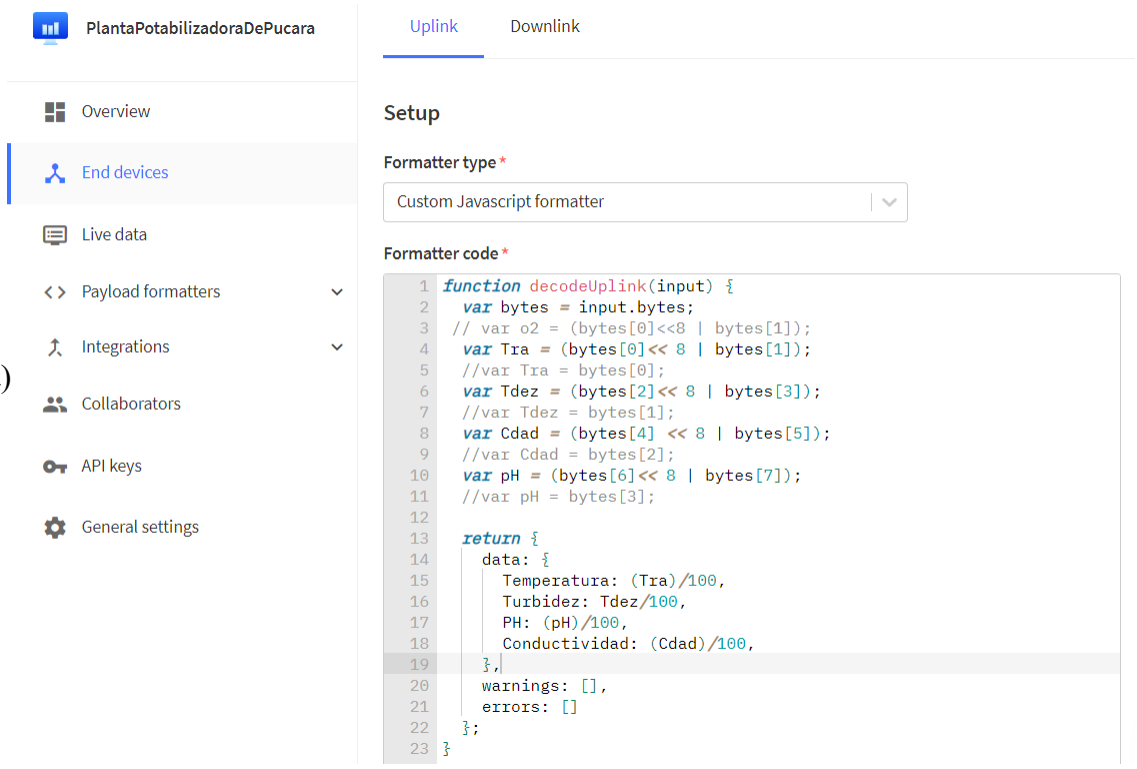


Figura 47. Registro de los parámetros del nodo Tx en TTN. Fuente: Elaboración propia.

Nota. En la imagen podemos observar la configuración del nodo Tx en TTN. Se ha utilizado el método de activación más seguro (OTAA).

Cuando la información llega al servidor TTN es necesario decodificarla, existen varias formas de decodificarla, en este caso se usó la decodificación empleando el formato JavaScript. En la Figura 48 se observa el programa desarrollado en TTN en formato JavaScript y la forma en cómo se presenta la data de cada sensor. Lo primero que se realiza en el programa es guardar la data en un arreglo para luego guardar por separado la información de cada sensor en diferentes variables. Finalmente, a cada variable numérica se la divide por 100, que es el valor que se multiplicó en el módulo Tx.

a)



b)

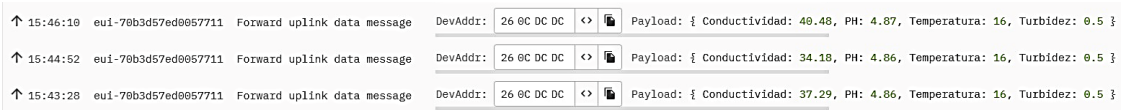


Figura 48. Programa en TTN en formato JavaScript para decodificar el payload. Fuente: Elaboración propia.

Nota. En la imagen en el literal a) podemos observar la decodificación del payload y en el b) observamos el payload decodificado.

5.3.6 Servidor de aplicación

Lo primero que se realizó es vincular TTN a TagoIO. Esto se lo efectuó basándome en algunos parámetros, entre ellos el más importante es la clave de autorización. En la Figura 49 se observa la vinculación con los diferentes parámetros.

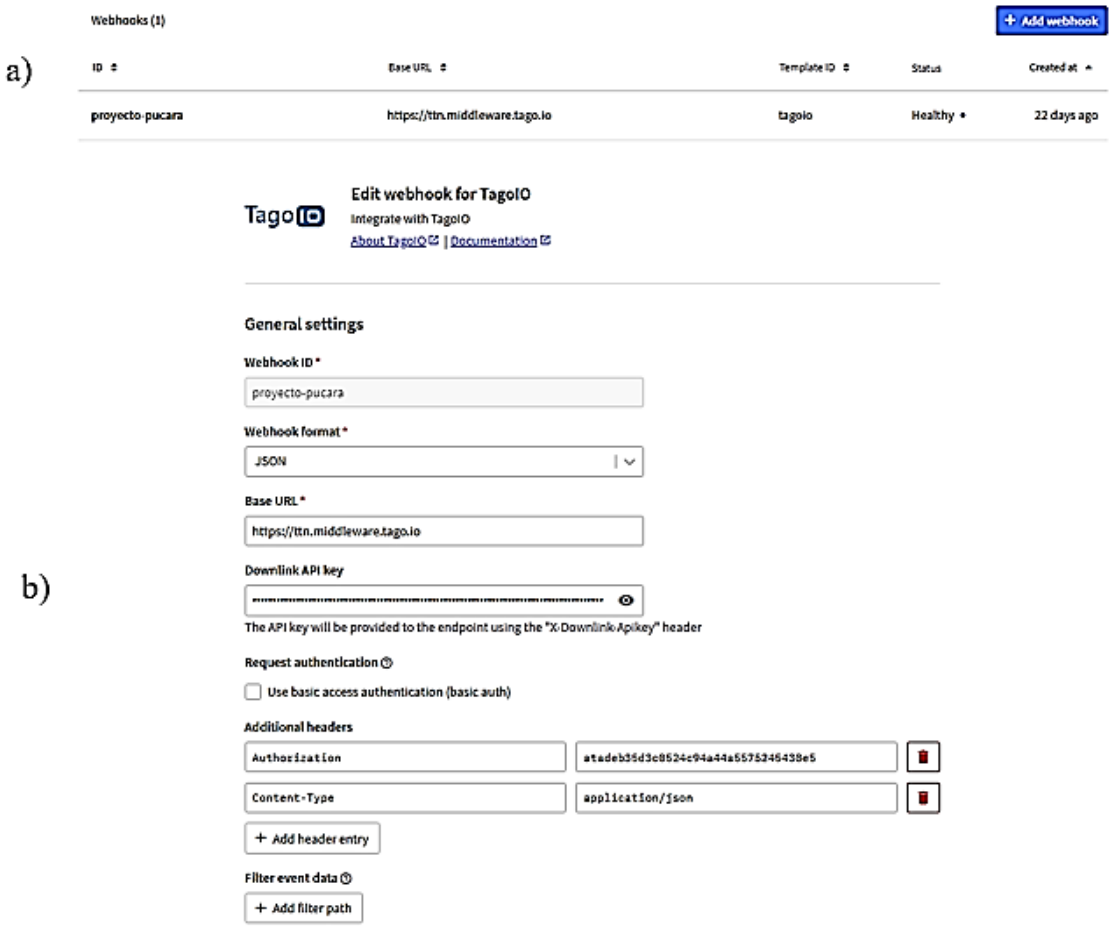


Figura 49. Vinculación de TagoIO a TTN. Fuente: Elaboración propia.

Nota. En la imagen podemos observar en el literal a) la vinculación de TagoIO a TTN mediante la agregación de un Webhook. Mientras que el literal b) representa la configuración de los parámetros de vinculación.

La vinculación de TTN a TagoIO se lo realizó al momento de agregar el dispositivo. Una vez creado el dispositivo para completar el vínculo, TagoIO intercambia información de sesión con TTN. En la Figura 50 se observa el dispositivo creado en TagoIO vinculado a TTN.

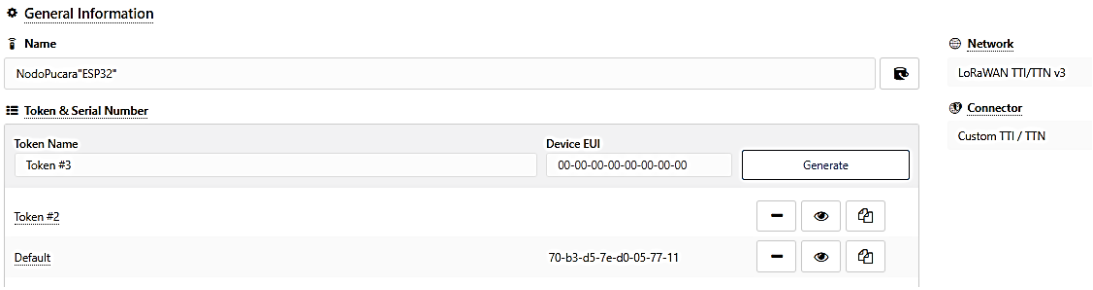


Figura 50. Información general del dispositivo en TagoIO vinculado a TTN. Fuente: Elaboración propia

Se realizó un programa en JavaScript en TagoIO para analizar el paquete enviado por TTN. En resumen, el programa lo que hace es extraer solo la información útil, luego se separa la información de los sensores. Una vez separado los valores de cada sensor es necesario restar o dividir valores agregados en la programación en el IDE de Arduino en el módulo Tx. Para finalizar se guarda los valores de los sensores en variables para luego visualizar en el panel de control. Todo el proceso antes mencionado lo podemos observar en la Figura 51.

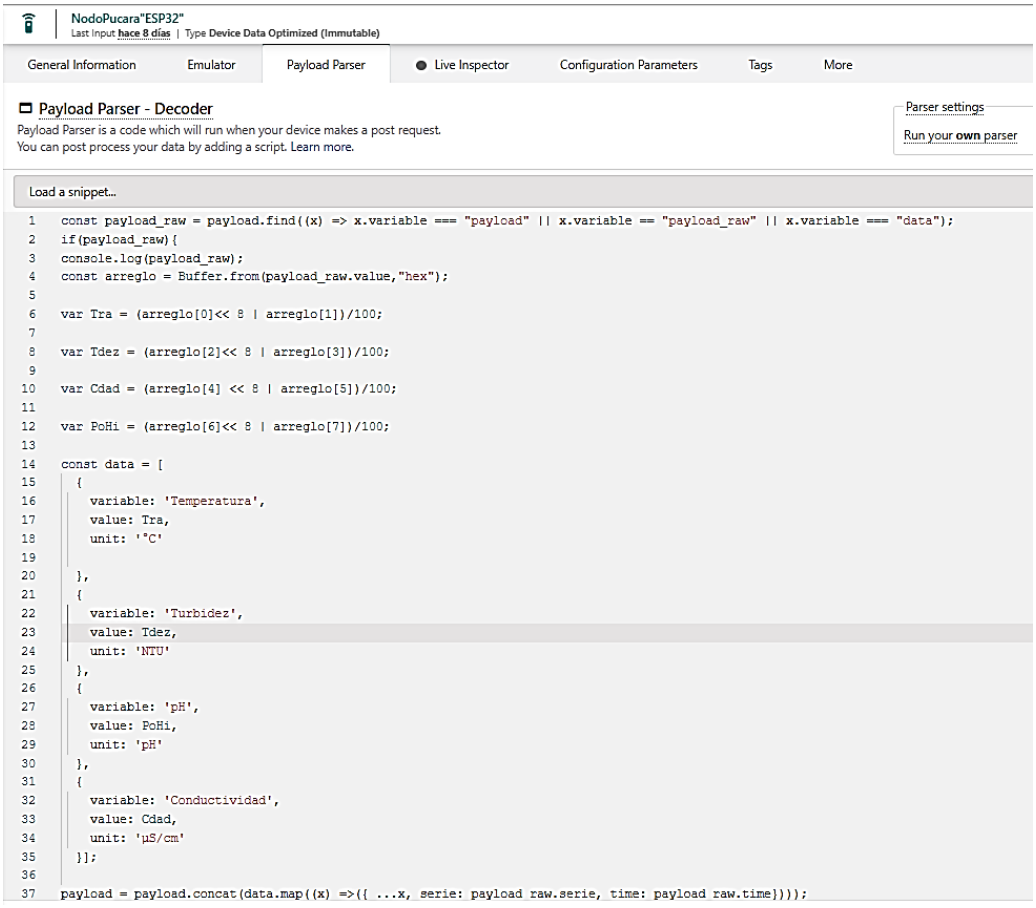


Figura 51. Programa para decodificar en TagoIO el payload enviado por TTN. Fuente: Elaboración propia.

Una vez que guardé las variables en TagoIO elaboré un tablero en donde podemos observar los datos de los sensores. En la Figura 52 se observa el panel de monitoreo en TagoIO con los datos de cada sensor con su respectiva gráfica.

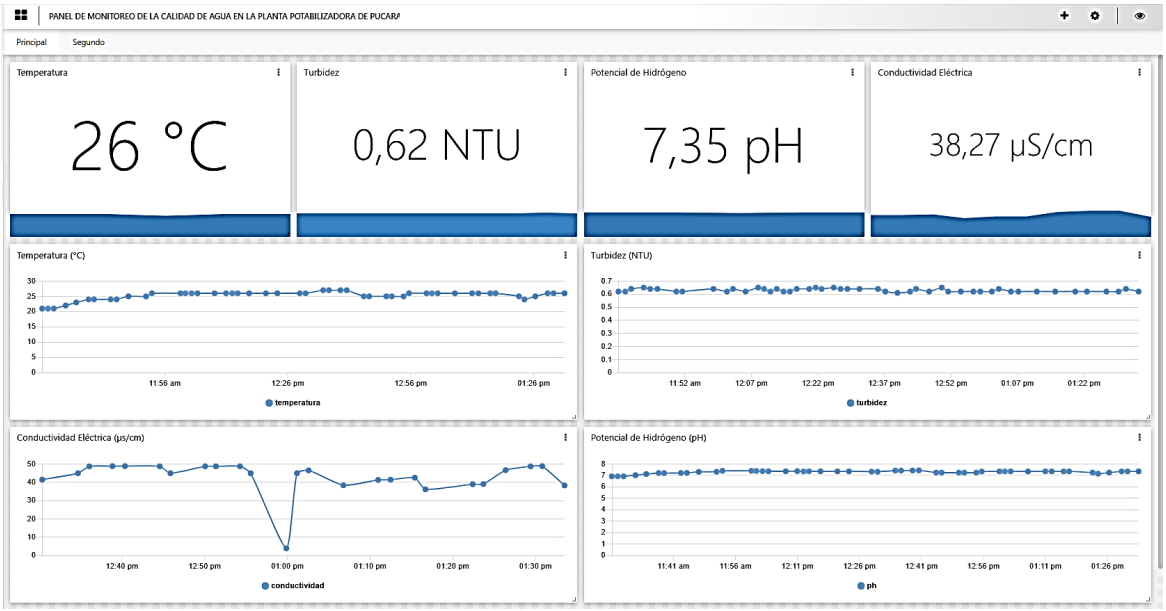
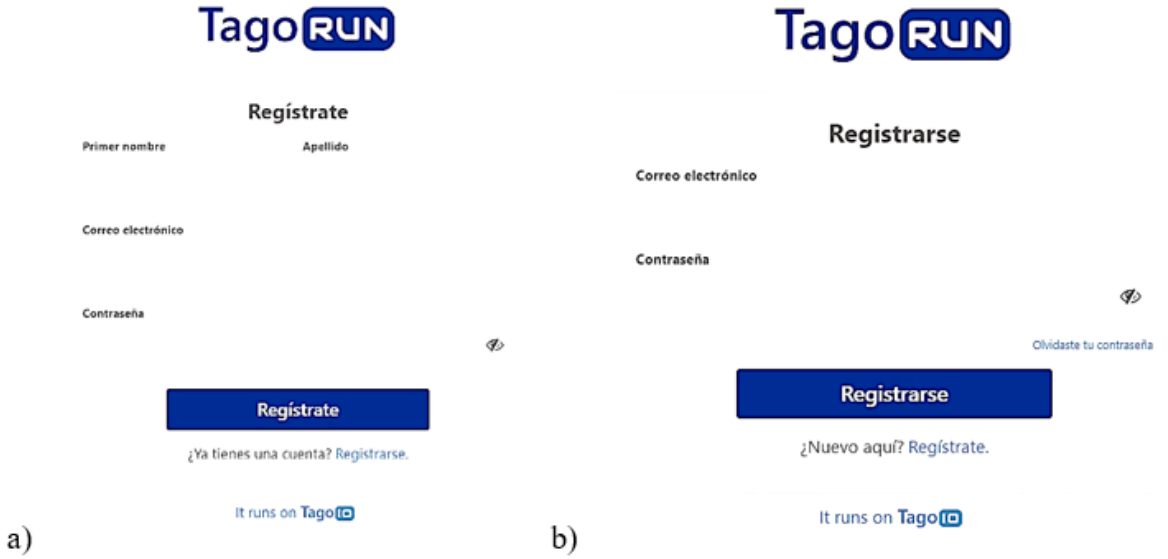


Figura 52. Panel de monitoreo en TagoIO. Fuente: Elaboración propia.

Después que realicé el panel de monitoreo, procedí a crear el control de acceso a los usuarios. Para esto se utilizó la opción "RUN" la cual permite el acceso al panel de control mediante un correo electrónico y una contraseña. En la Figura 53 se observa cómo crear una cuenta y como ingresar al panel de monitoreo de TagoRUN. Además, en la Figura 53 también se muestra el panel de monitoreo en TagoRUN que es similar al panel mostrado en TagoIO.



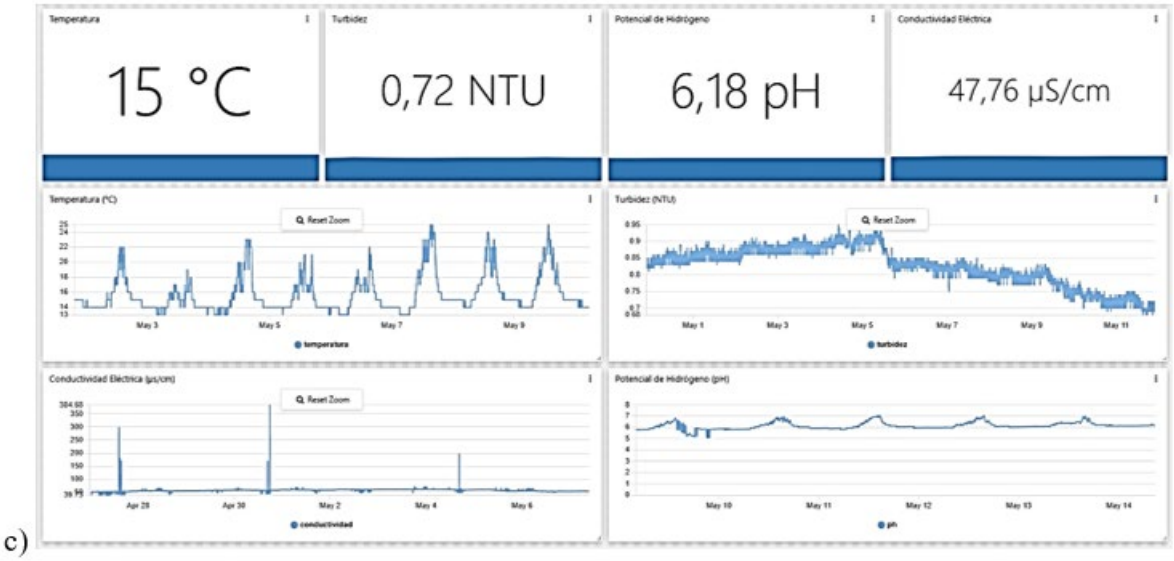


Figura 53. TagoRUN. Fuente: Elaboración propia.

Nota. En la imagen el literal a) hace referencia a los datos que se necesitan para crear una cuenta, mientras que el literal b) representa las credenciales para ingresar al panel de monitoreo y el literal c) es lo que los usuarios lograron observar cuando se les concedió el permiso.

En TagoIO en la opción de "ACCESS" se configuró las políticas de acceso que se les otorga a los usuarios. También es posible ingresar desde el aplicativo celular que podemos descargar desde la Play Store o App Store, la aplicación está con el nombre "TagoRUN", el dominio que hay que ingresar es el siguiente https://63425c5d765e7200186f7746.tago.run/, y luego se continúa con los pasos antes mencionados.

5.3.7 Construcción

Se ha requerido a la Unidad Municipal de agua potable y alcantarillado de Loja (UMAPAL) mediante una solicitud (ver Anexo 12) que se me conceda el permiso para realizar la implementación. Esta fue aceptada (ver Anexo 13) favorablemente, por lo cual pude realizar la implementación (ver Anexo 14). En la planta potabilizadora de Pucará se me asigno un lugar en donde pude implementar el sistema de monitoreo. Para realizar el monitoreo se necesitó 2 cajas de un tamaño de 13 cm x 15 cm. Una caja se utilizó para ubicar el Arduino Uno, el módulo Tx y la parte activa de los sensores, mientras que en la otra caja se receptó el agua y se colocó la parte pasiva de los sensores. A continuación, en la Figura 54 se observa la construcción del nodo Tx.

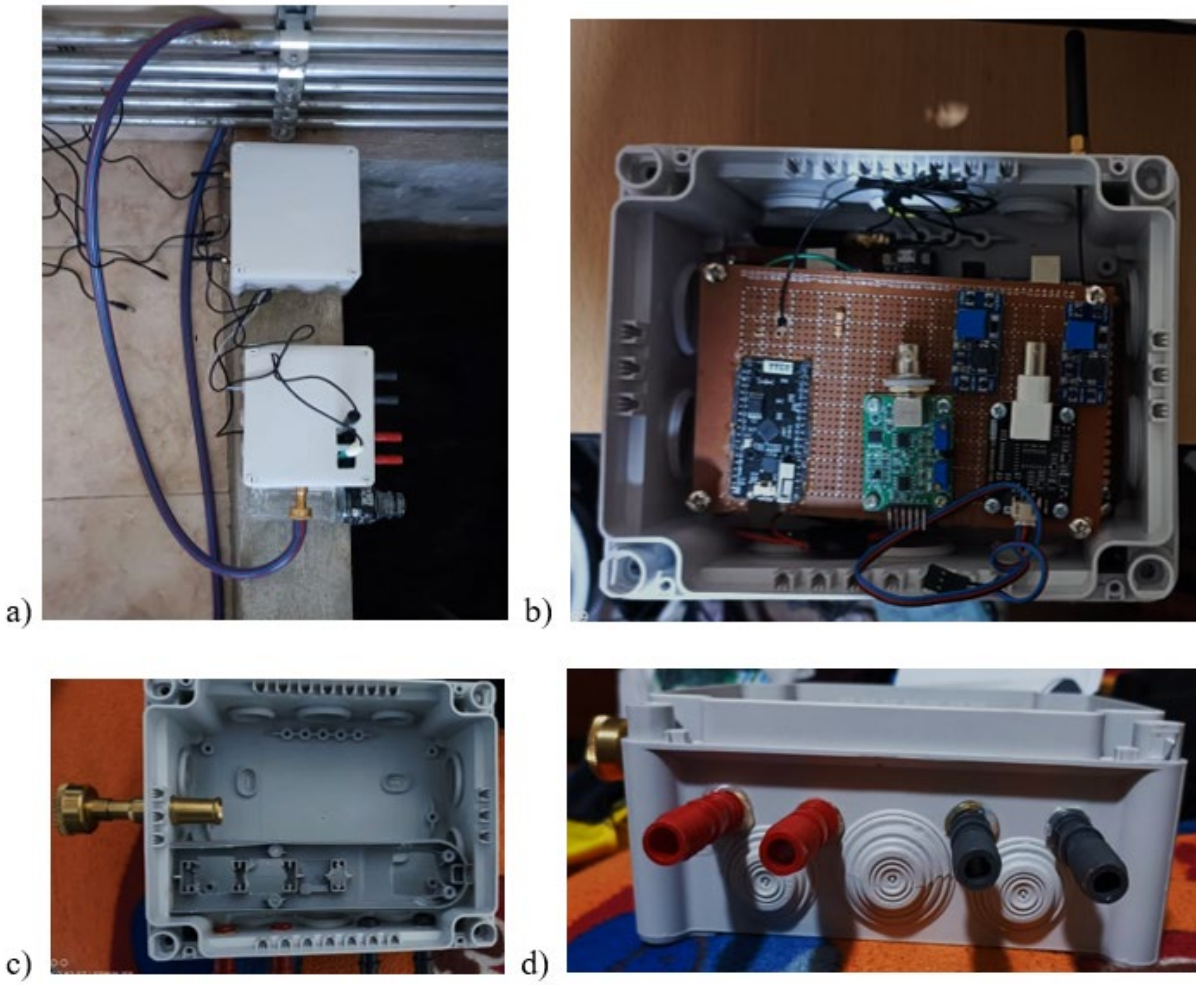


Figura 54. Construcción del sistema de monitoreo IoT. Fuente: Elaboración propia.

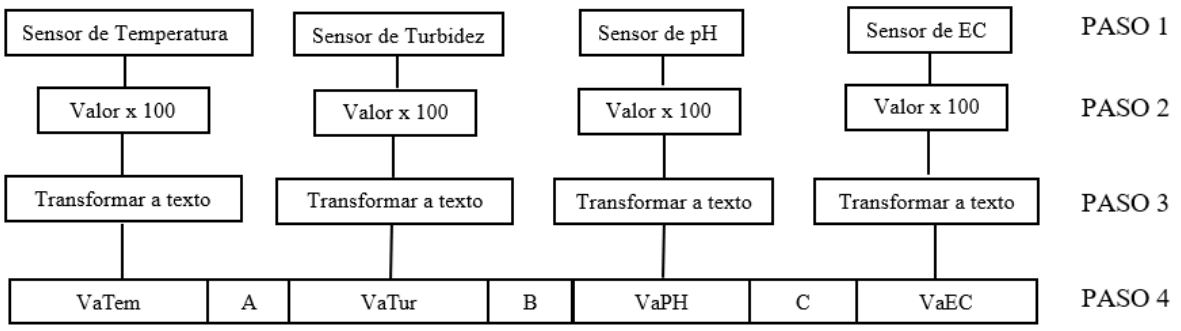
Nota. En la imagen el literal a) representa el lugar en donde se implementó el sistema de monitoreo, el literal b) es la caja en donde está la parte activa de los sensores, las placas de control y transmisión, y los literales c) y d) representan el lugar en donde se ubican la parte pasiva de los sensores.

6. Resultados

La propuesta del presente proyecto consistió en desarrollar una red IoT para monitorear los parámetros de turbidez, temperatura, pH y conductividad del agua. El prototipo se desarrolló en diferentes fases: diseño, construcción e implementación. Como resultado final se obtuvo comunicación; entre los sensores y el Arduino uno; entre el Arduino Uno y el módulo Tx; entre el módulo Tx y el servidor de red TTN; y entre servidor TTN y servidor de aplicación TagoIO. TagoIO sirvió para generar el panel de control de monitoreo, el cual está compuesto por las variables de los sensores y los widgets.

6.1 Comunicación entre los sensores y el Arduino Uno

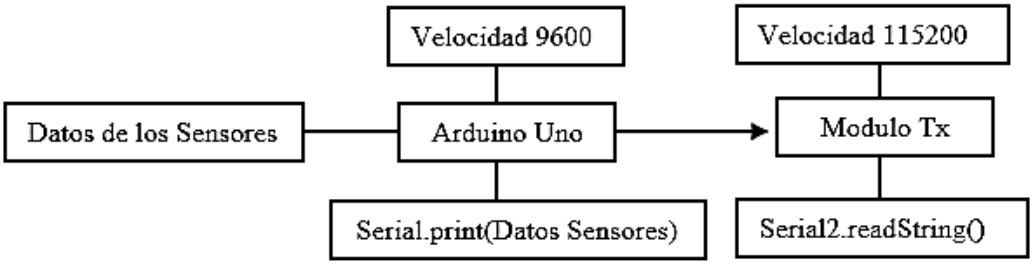
En este punto se pudo recolectar la información de los sensores. En la programación del Arduino Uno para algunos sensores necesitamos instalar librerías y para otros no, un ejemplo es el sensor de conductividad eléctrica (EC) que para utilizarlo instale las librerías de DFRobot. Una vez que recolecté los datos, los codifique y los transforme en texto para ser enviados al módulo Tx por medio de una comunicación UART. A continuación, se presenta un diagrama de bloques del código de programación.



En el diagrama podemos observar 4 pasos: en el paso 1 se realiza la adquisición de la información por parte de los sensores, la cual son cantidades con cifras decimales, para trabajar con cifras enteras, en el paso 2 se multiplicó por una centena. Luego en el paso 3 a las cantidades enteras se las transformo en cadenas de texto, y finalmente en el paso 4 se une la información enviada por cada sensor agregando una letra para poder lograr la decodificación en el módulo Tx. Como resultado, en esta sección logré obtener en la placa Arduino Uno una cadena de texto con la información de los sensores, la cual se la envió al módulo Tx.

6.2 Comunicación entre el Arduino Uno y el módulo Tx

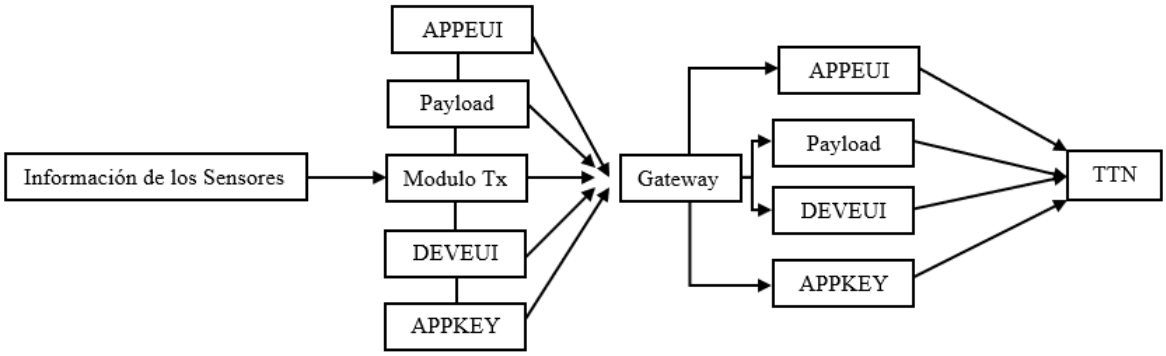
Esta comunicación es en serie UART (Universal Asynchronous Receiver-Transmitter). Iniciamos la comunicación en serie sincronizando la velocidad del emisor a 9600 baudios y el receptor a 115200, luego envié desde el Arduino Uno la información mediante el comando de Serial.print(), y la recibí en el módulo Tx mediante el comando Serial2.readString(). A continuación, se presenta el diagrama del proceso de esta comunicación.



Es muy importante tener en cuenta la velocidad de la comunicación serial para evitar errores de comunicación. La decodificación de la información se la realizo de la siguiente forma: el primer paso es recorrer la cadena de texto e identificar las letras para poder determinar el tamaño de la información de cada sensor, en este paso se utilizó un ciclo "for" y un condicional "if". Luego de determinar la información de cada sensor, transformamos la cadena de texto a número con la función "toInt". Como resultado de esta fase se logró obtener la información de los sensores en el módulo Tx mediante una comunicación serial UART con el Arduino Uno.

6.3 Comunicación entre el módulo Tx y el servidor TTN

Para esta comunicación se utilizó la tecnología de acceso a radio LoRa específicamente el protocolo LoRaWAN y la tecnología inalámbrica 802.11. El proceso de la comunicación es el siguiente: lo primero que se realizó es registra en el servidor TTN el Gateway HT-M00 que sirvió como puerta de enlace para el módulo Tx. En el módulo Tx se programó la vinculación OTTA mediante la información de sesión y activación, que son el identificativo de la aplicación (APPEUI) y del dispositivo (DEVEUI), y la clave de aplicación (APPKEY). Una vez que terminé la vinculación procedí a armar el payload en un arreglo con los valores de los sensores, es importante mencionar que esta información enviada por el módulo Tx es en formato hexadecimal. A continuación, se presenta un diagrama del proceso de esta comunicación.



Existen varios puntos importantes que hay que señalar en el programa del módulo Tx al momento de enviar la información: uno de ellos es que para este trabajo se utilizó la biblioteca MCCI LoRaWAN LMIC, también hay que señalar la programación del "Pin Mapping" debido a que esta varía de acuerdo a la versión de la placa Tx (observar el Anexo 8), para enviar la información se utiliza el siguiente comando "LMIC_setTxData2". Existen otros parámetros de programación, he redactado los anteriores porque he considerado que con los más relevantes. Como resultado de esta etapa se logró enviar la información de los sensores en formato hexadecimal a través de una comunicación LoRaWAN al servidor de red TTN por medio del protocolo de internet inalámbrico 802.11.

En la Figura 55 se puede observar la data que es receptada y decodificada en el servidor de red TTN. En la imagen se observa la data de los 4 sensores con su respectiva variación en un lapso de 30 minutos.

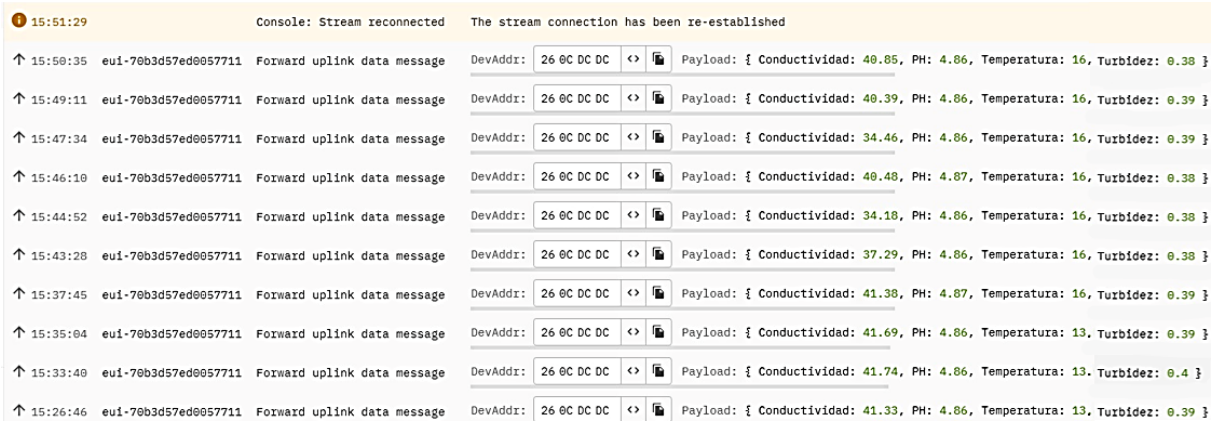
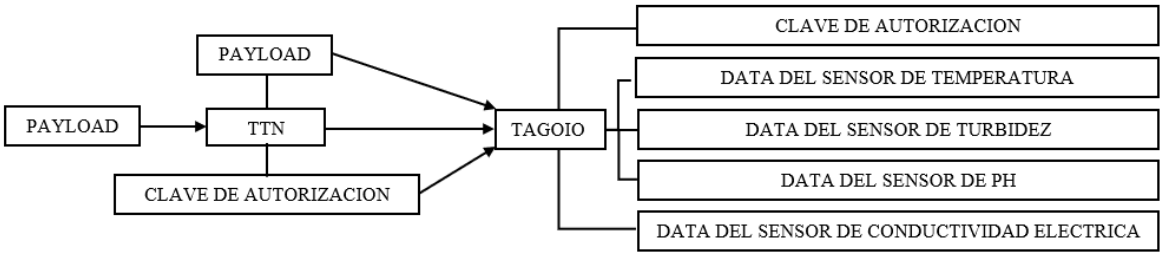


Figura 55. Visualización de los datos en el servidor de red TTN. Fuente: Elaboración propia.

Nota. En la imagen se puede observar la data enviada en una media hora por el módulo Tx al servidor de red TTN.

6.4 Comunicación Servidor de red “TTN” - Servidor de aplicación “TagoIO”

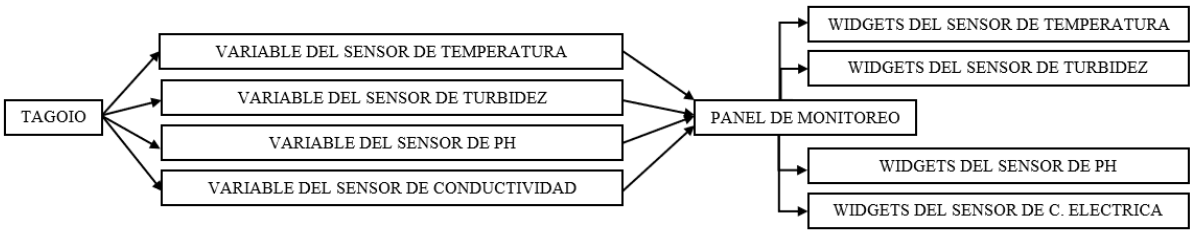
Una vez que se recibió la información en TTN se la adecuo mediante un programa en formato JavaScript, luego se procede a vincular al servidor TTN el servidor TagoIO. En TTN hay que crear un Webhook, una característica importante en el Webhook es la clave de autorización la cual la podemos obtener en TagoIO. Lo que se realizó en TagoIO es que al momento de crear el dispositivo se le indicó que este procedería del servidor TTN y posteriormente se creó una clave de autorización. Para la manipulación de la información se creó un programa en formato JavaScript del cual obtuvimos 4 variables, una para cada sensor, las cuales fueron utilizadas en el panel de monitoreo. A continuación, se presenta el diagrama de la comunicación.



Existen varios parámetros de programación y configuración, pero los que he mencionado son los más notables en este proceso de comunicación. Como resultado de esta comunicación obtuve un programa (ver Figura 48) en TTN el cual acondicionó los datos para su uso en TagoIO, por otro lado, en TagoIO conseguí realizar un programa (ver Figura 51) el cual permitió almacenar la data de cada sensor en variables diferentes para su posterior uso en el panel de control.

6.5 Panel de monitoreo

El panel de monitoreo se lo realizo en el servidor de aplicación TagoIO. Para el proceso de creación del panel se necesitó la data de cada sensor almacenada en variables diferentes (ver Figura 51), con esta información se procedió a crear los widgets los cuales permitieron visualizar los datos y su variación. A continuación, se presenta un diagrama del proceso de creación del panel de control.



El panel de monitoreo es pasivo haciendo referencia a que sirve para visualización. Para la visualización del panel de monitoreo se utilizó “TagoRUN”, el cual permitió crear acceso al panel mediante un link y una aplicación móvil que se la puede descargar de la Play Store o App Store. Para ingresar se necesita estar registrado y con permiso del administrador. Como resultado, en esta sección se logró crear un panel de monitoreo con las variables de los sensores, y mediante “TagoRUN” se creó el acceso a los usuarios al panel de control. A continuación, se presenta el panel de monitoreo del administrador (ver Figura 56) y del usuario (ver Figura 57). Además, en TagoRUN el usuario puede analizar data de cada sensor extrayéndola en formato Excel (XLSX) o en un archivo separado por comas (CSV) (ver Figura 58), esta opción se utilizó para analizar la variación del número de paquetes.

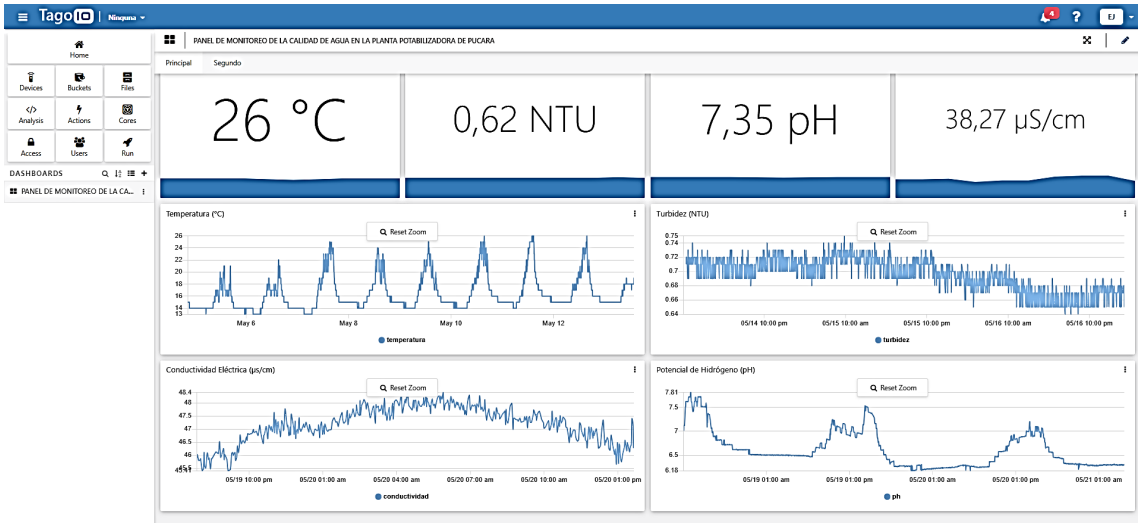


Figura 56. Panel de monitoreo del administrador. Fuente: Elaboración propia.

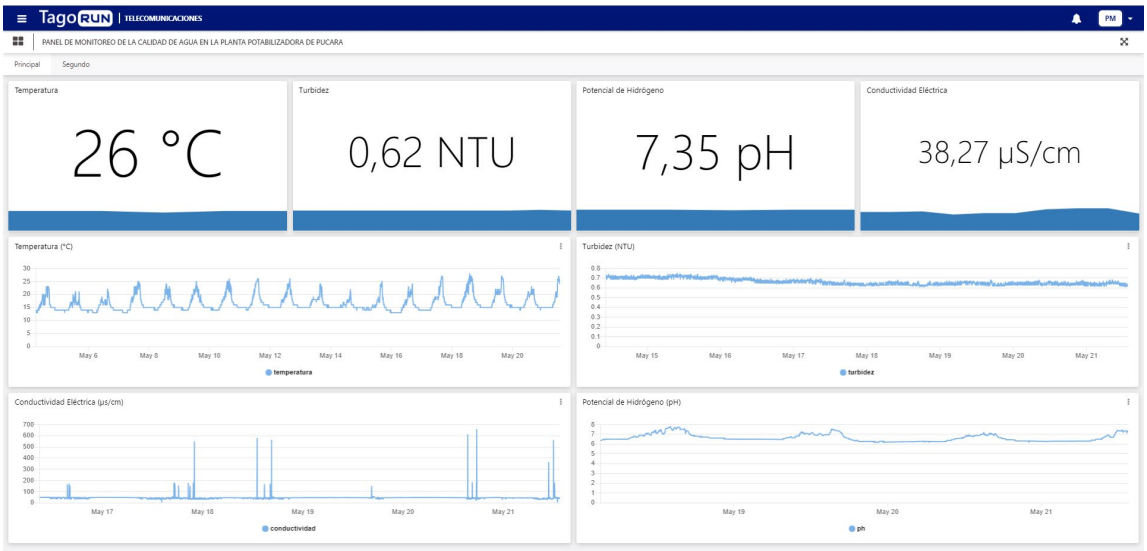


Figura 57. Panel de monitoreo del usuario. Fuente: Elaboración propia.

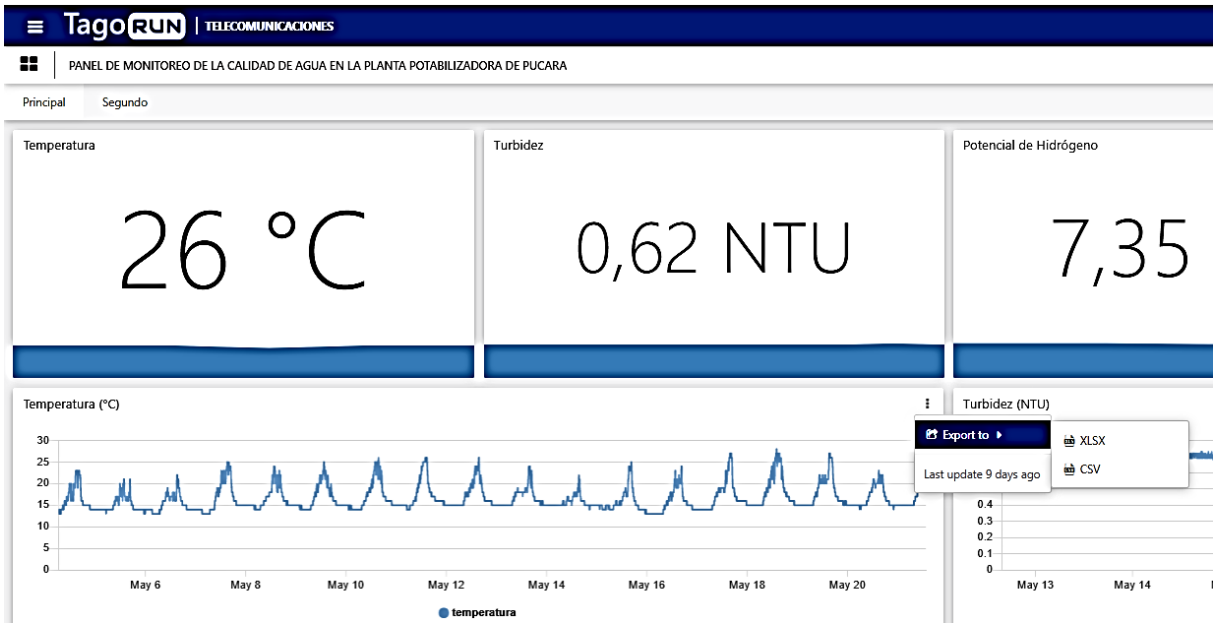
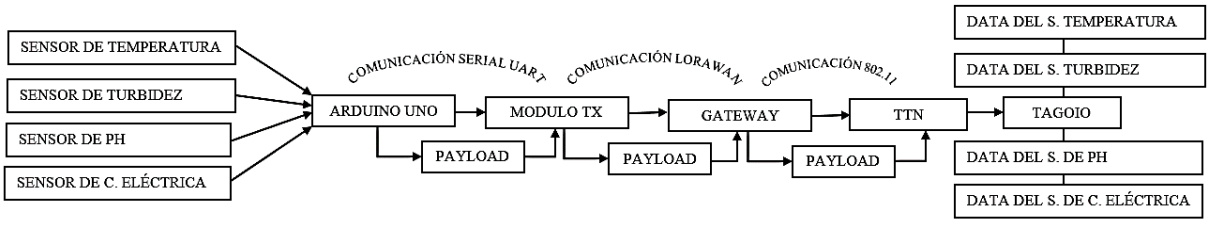


Figura 58. Formas de exportar la data de los sensores en TagoRUN. Fuente: Elaboración propia.

6.6 Construcción final de la red IoT

En esta fase se realizó la construcción del sistema de monitoreo. Se utilizaron 2 cajas, en una de ellas está la parte activa de la red y en la otra la parte pasiva. Como resultado se obtuvo la red IoT para el monitoreo de la temperatura, turbidez, pH y conductividad implementada en la planta potabilizadora de Pucará. A continuación, se presenta un diagrama general de la red IoT.



Mediante una manguera se llevó el agua de un grifo hacia el lugar en donde están ubicadas las cajas. Hay que mencionar que se controló el flujo del agua debido a la sensibilidad de los sensores. Un flujo fuerte provoca variación de la información, para controlar esta variación se utilizó un regulador de flujo mecánico en la caja. En la Figura 59 se puede observar con más detalle el lugar en donde se ubicaron los equipos de la red IoT.

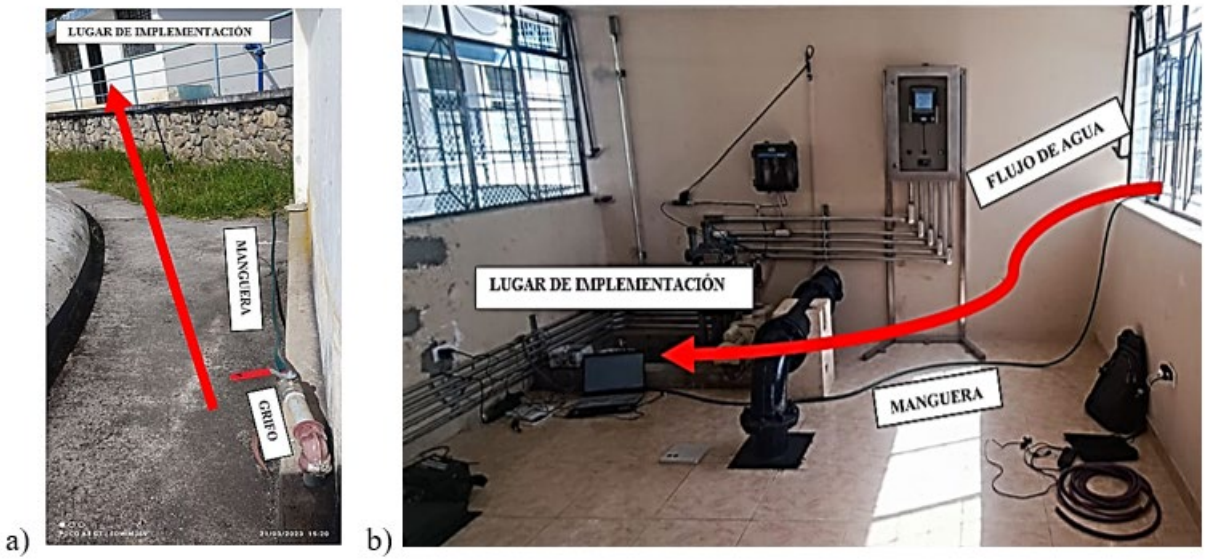


Figura 59. Lugar de implementación de la red IoT. Fuente: Elaboración propia.

Nota. En el literal a) se observa el lugar de donde se trasladó el agua a través de una manguera de aproximadamente 20 m al lugar de implementación (ver literal b).

Como se lo ha estado mencionando se utilizaron 2 cajas (ver Figura 60), en una de ellas está la parte activa como son el módulo Tx y las tarjetas controladoras de los sensores de pH, conductividad eléctrica y turbidez. En la otra caja está la parte pasiva, hay que mencionar que en esta caja llega el agua del grifo y luego pasa hacia la zona de distribución.

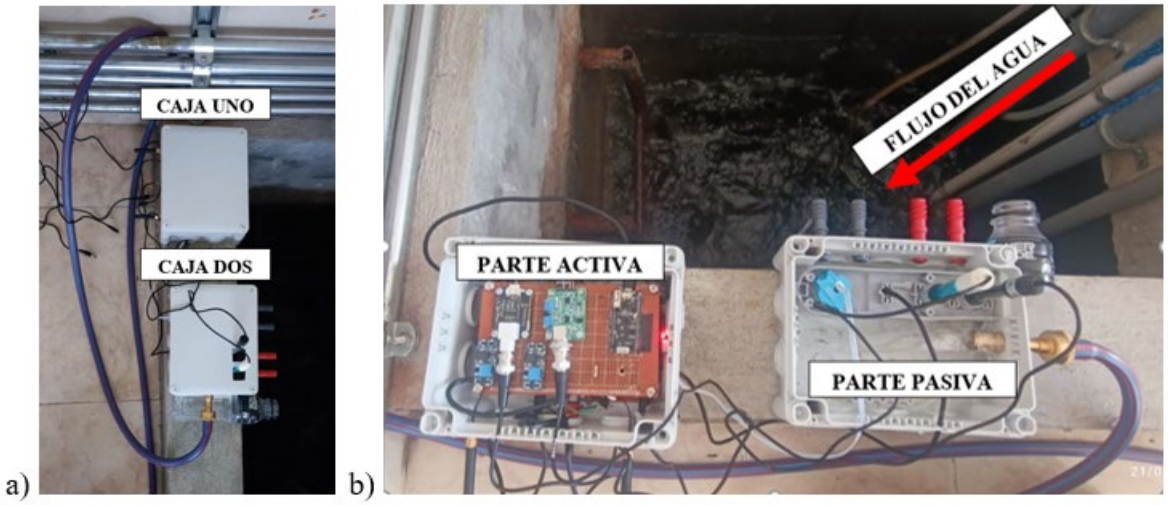


Figura 60. Nodo de transmisión y caja de monitoreo. Fuente: Elaboración propia.

Nota. El nodo de transmisión es la caja uno en donde está la parte activa de los sensores y la tarjeta Tx. La caja dos es en donde se realiza el monitoreo con la parte pasiva de los sensores.

6.7 Análisis de la red IoT

Para este análisis de la red IoT se obtuvieron alrededor de 3 mil paquetes, pero la red IoT estuvo en funcionamiento aproximadamente un mes, y recolecto un aproximado de 32 mil paquetes. En la Figura 61 se muestran la cantidad de paquetes que registro la red IoT.

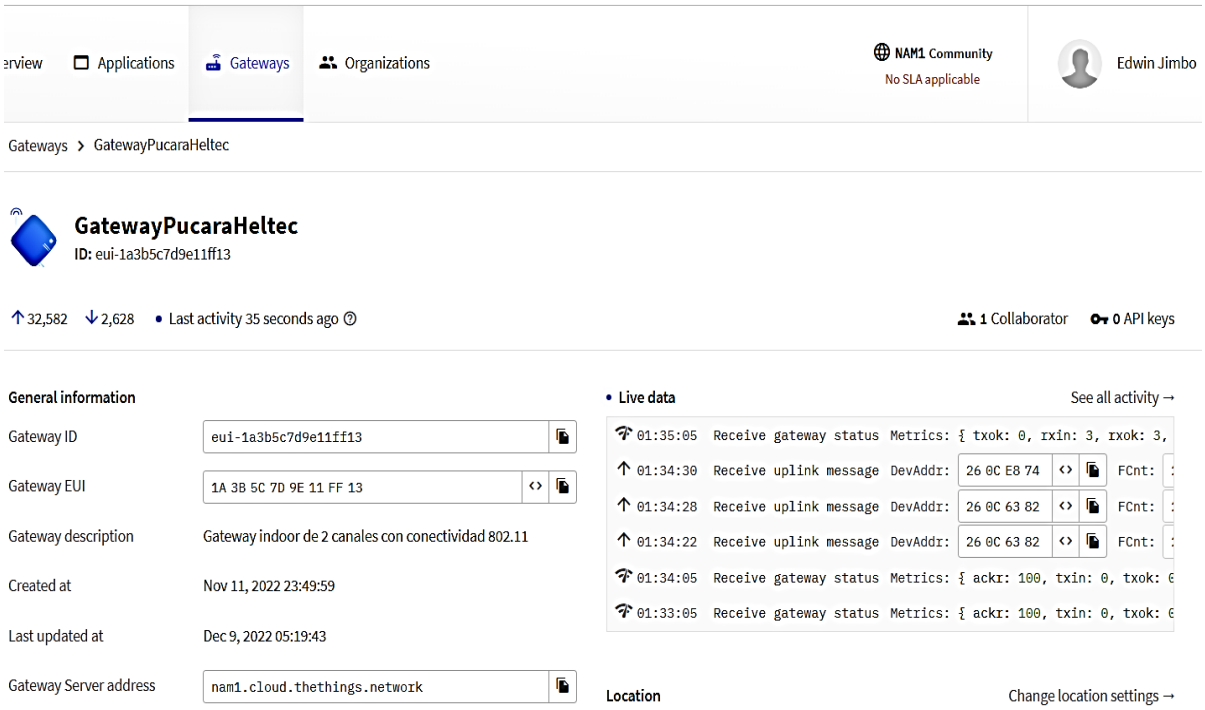


Figura 61. Número de paquetes recolectados por el sistema IoT. Fuente: Elaboración propia.

Con respecto a los sensores, estos padecían de estabilidad al momento de presentarse cambios en el movimiento del agua o el movimiento del mismo sensor. La información recolectada mediante la red IoT se la comparo con el registro que realiza la planta de Pucará mediante el método de "JARRAS". En la comparación de los datos se pudo evidenciar algunas diferencias entre los datos, esto debido a la calidad de los sensores y al agua que se está analizando. Por un lado, la planta de Pucará utiliza otra clase de sensores, y recolecta muestras de agua para analizarlas en el laboratorio. El sistema IoT empleo un monitoreo en tiempo real en donde el agua estaba fluyendo.

Se fijó en un lugar seguro las 2 cajas, además también se reguló el flujo del agua para minimizar esta variación. En la Figura 62 se observa el registro de los parámetros de calidad del agua en la planta potabilizadora de Pucará.

Figura 62. Control del proceso de potabilización en la planta de Pucara. Fuente: Elaboración propia.

Nota: Estos valores son registrados mediante el método de "Jarras".

El método de Jarras consiste en sacar muestras del agua en las diferentes fases del proceso de potabilización. Luego esas muestras se las analizan en el laboratorio. Y finalmente se realiza un registro mediante el cual se tiene un mejor control en el proceso de potabilización.

En la Figura 63 se muestran los paquetes de cada sensor decodificados en el servidor de red TTN. De esta manera se recolectó la información para poder realizar la comparación con la información recolectada en la planta de Pucara.

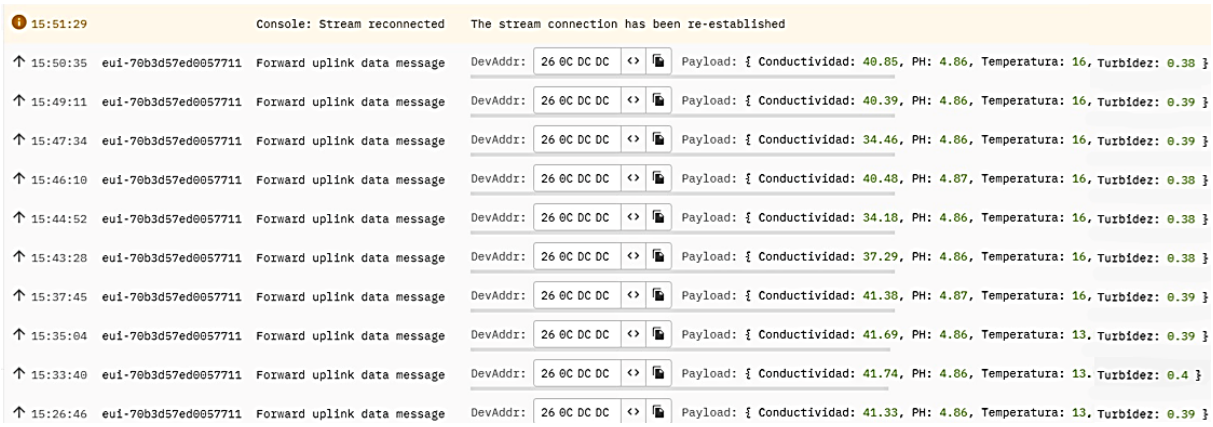


Figura 63. Análisis de la data de los sensores en TTN. Fuente: Elaboración propia.

Nota. El servidor TTN ha registrado 10 paquetes. Cada paquete contiene la data de los 4 sensores.

A continuación, se presenta la Tabla 7 que la realice basándome en el registro de la planta de Pucará presentado en la Figura 62. Del Anexo 10 se ha redactado la Tabla 8 que es la data que se ha recolectado con el sistema IoT para realizar la comparación. Esta comparación se la realizo a inicios del mes de marzo del 2023.

Tabla 7. Monitoreo en la Plata de Pucara

Hora	Tem (°C)	Tur (NTU)	EC (µS/cm)	pH
8:00	12.80	0.41	36.90	6.20
9:00	12.90	0.61	30.30	5.11
10:00	13.00	0.41	35.60	4.71
11:00	13.00	0.54	32.30	5.27
12:00	15.40	0.72	35.50	5.50
13:00				
14:00				
15:00	14.10	0.56	32.10	5.18
16:00	14.00	0.46	32.20	5.08
17:00	13.90	0.94	35.10	4.82

Nota: Valores registrados en la planta de Pucará mediante el método de Jarras. Fuente: Elaboración propia.

Tabla 8. Monitoreo en base al sistema IoT

Hora	Tem (°C)	Tur (NTU)	EC (µS/cm)	pH
8:00	14.00	0.60	50.50	5.92
9:00	15.00	0.65	47.58	6.18
10:00	15.00	0.61	48.93	6.02
11:00	17.00	0.63	48.85	6.21
12:00	17.00	0.67	45.32	6.21
13:00				
14:00				
15:00	18.00	0.60	34.87	6.47
16:00	18.00	0.58	37.68	6.47
17:00	16.00	0.65	44.25	6.28

Nota: Estos valores fueron registrados mediante la red IoT. Fuente: Elaboración propia.

La variación que existe entre los valores registrados en la planta de Pucará y los registrados por el sistema de monitoreo IoT es por diversas razones, podemos observar esta variación en las Tablas 7 y 8. La primera son los equipos: el método de “Jarras” es un método de laboratorio y por consiguiente se utilizan equipos profesionales. La segunda es el agua: el método de “Jarras” usa muestras de agua, mientras que el agua está en movimiento en el método IoT, también hay que mencionar que en el método IoT el agua también ha pasado por otros procesos, por lo cual es diferente a la utilizada en el método de “Jarras”.

Con respecto a la comunicación UART esta fue estable y robusta porque los valores que se enviaban desde el Arduino Uno llegaban de manera correcta al módulo Tx. Lo importante en esta comunicación es la velocidad de transmisión.

La comunicación entre el módulo Tx y el servidor TTN presenta algunas perdidas de paquetes, esto se producen por la inestabilidad de la red wifi y también se debe a los equipos que se están utilizando: el Gateway es de solo 2 canales y el módulo Tx es una versión 1. Pese a que existen perdidas de paquetes, la red IoT presento buenos resultados en la comunicación entre el nodo y el servidor TTN.

Figura 64. Visualización de la pérdida de paquetes en el servidor TTN. Fuente: Elaboración propia.

En la Figura 64 podemos observar la comunicación que se estableció entre el módulo Tx y el servidor TTN. La comunicación debe realizarse como mínimo cada minuto, pero como podemos observar en los literales de la Figura 64, esto no ocurre. En el literal a, se envían un correcto número de paquetes en el minuto 19 y 20, pero existe perdida de paquetes en los minutos 20 al 22. En el literal b, existe una gran pérdida de paquetes, observamos que se envía un paquete en el minuto 37 y luego en el minuto 43, han pasado 6 minutos y no se han registrado paquetes, este lapso de tiempo puede variar, pueden ser 2, 4, 5, 10 e incluso en ocasiones 20 minutos. Esta pérdida de paquetes se debe principalmente a que el Gateway es de 2 canales y se conecta hacia la Internet mediante el protocolo 802.11. Un Gateway robusto tiene mínimo 8 canales y se conecta hacia la Internet mediante el protocolo 802.3. Finalmente, en el literal c, podemos observar que en ocasiones el quipo en donde estamos observando la data se desconecta de la red de internet, esto provoca que los paquetes no se puedan visualizar, pero, aunque no se puedan visualizar en TTN, si se pueden visualizar en TagoIO.

En la Tabla 9 se registra la variación del número de paquetes que se receptó en 5 días en TagoIO. Esta información se la adquirió de TagoRUN en los días 15 hasta 19 de mayo del 2023 (ver Figura 58) en formato XLSX.

Tabla 9. Variación del número de paquetes recibidos en TagoRUN

Día	Lunes	Martes	Miércoles	Jueves	Viernes
Paquetes	585	582	564	585	601

Nota. En la tabla podemos observar la variación del número de paquetes que se receptaron a diario por 5 días. Fuente: Elaboración propia.

Como podemos observar en la Figura 62 mediante el método de Jarras utilizado en la planta de Pucará se obtienen en el día 8 registros diarios, con el sistema IoT se obtuvieron un mínimo 582 y un máximo de 601 registros.

Los servidores TTN y TagoIO mostraron una robustez en su uso. TTN, por su parte, admitió al Gateway HT-M00, el cual por ser de 2 canales algunos servidores IoT no permitían su registro. Por otra parte, TagoIO permitió la vinculación de TTN, lo cual facilito el uso de los datos de los sensores en el panel de monitoreo.

Como se lo menciono en la sección de metodología. El sensor de turbidez utilizó un módulo aparte para poder enviar la información. En la Figura 65 se observa el módulo que se creó para el sensor de turbidez, y también se observa el módulo creado los 3 sensores restantes en las plataformas de TTN y TagoIO.

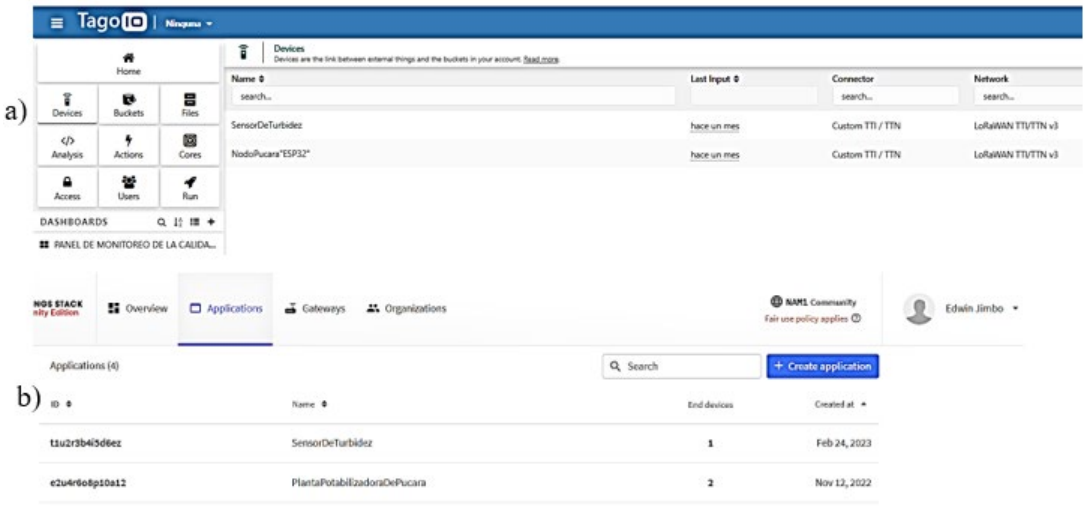


Figura 65. Dispositivos y aplicaciones creados en TTN y TagoIO. Fuente: Elaboración propia.

Nota. El literal a) representa la plataforma TagoIO y el b) la plataforma TTN.

7. Discusión

Como se lo mencionó en un inicio, el presente trabajo de titulación tiene como propósito crear un prototipo de una red IoT para el monitoreo de la temperatura, turbidez, conductividad eléctrica (EC) y potencial de hidrógeno (pH) del agua en la planta potabilizadora del sector de Pucará de la provincia de Loja. Para lograr este objetivo lo primero en realizar fue un diseño de red, en el cual se analizó parámetros de la red como cobertura, potencias, materiales, etc. Luego, en la implementación se realizó la configuración, programación y construcción de la red IoT. En los resultados se pudieron analizar 4 tipos de comunicaciones: Sensores - Arduino Uno, Arduino Uno – Módulo Tx, Módulo Tx – TTN y TTN – TagoIO.

En los resultados del trabajo, en la comunicación entre los sensores y el Arduino Uno se obtuvo una regular recolección de información. Algunos de ellos como el sensor de pH y el sensor de turbidez presentaron mala sensibilidad en el censado, lo cual provoco una variación poco satisfactoria. El sensor de conductividad eléctrica utilizado en este trabajo coincide con el sensor utilizado en el trabajo "Monitoreo de calidad del agua en sistema de agua potable rural" realizado por Conejeros et al. (2021), este sensor cuenta con una tarjeta acondicionadora realizada por DRobot la cual mostró el mejor resultado en la variación con respecto a los sensores de pH y turbidez, los cuales también utilizaron una tarjeta acondicionadora. La buena sensibilidad del sensor de EC favoreció la programación, pero también provoco que se generen altos y bajos en el registro de la data. En la investigación efectuada por Conejeros et al. (2021) la mayoría de los sensores usados son de la marca DFRobot, en el trabajo que efectué los sensores de pH y turbidez emplearon una tarjeta acondicionadora sencilla, lo cual provoco una inestabilidad en el censado. Por otro lado, en el presente trabajo y en el trabajo de Conejeros et al. (2021) se usaron el Arduino Uno para la recolección de la información, en otros trabajos como en el de Jiménez et al. (2020) la recolección de la data de los sensores se lleva a cabo en el módulo Tx. El Arduino Uno exhibió buenos resultados en la recolección de la información, para la presente investigación se realizaron pruebas de recolección de información en el módulo Tx las cuales no presentaron buenos resultados para los sensores analógicos, pero sí para el sensor de temperatura. En la investigación de Conejeros et al. (2021) todos los sensores se conectan al Arduino Uno, en el trabajo que realice se puede conectar todos los sensores al Arduino Uno, o también al sensor de temperatura se lo puede conectar directamente al módulo Tx, de ambas formas funciona bien la recolección de datos. También se realizó la recolección información del sensor de turbidez en un nodo aparte con el fin de mejorar la calidad de la data

debido a que los pines analógicos de la placa Arduino Uno provocaban interferencia entre ellos, la cual era más significativa en el sensor de turbidez.

En los resultados de la comunicación serial UART entre el Arduino Uno y el módulo Tx se pudo observar un excelente resultado en la transferencia de información. Otros trabajos como el de Silva & Coello (2020) utilizan una comunicación serial diferente para él envió de la información entre el Arduino Uno y el Heltec esp32 LoRa, mientras que en el estudio de Conejeros et al. (2021) se realiza una comunicación USB para el envío de información. La comunicación serial que se realizó en el presente proyecto y en el proyecto de Silva & Coello, (2020) exhibieron buenos resultados, porque se pudo obtener la información recolectada en el Arduino Uno en el Módulo Tx.

En resultados de la comunicación entre el módulo Tx y el servidor de red TTN se pudo apreciar excelente aceptación del Gateway HT-M00 por parte del servidor de red TTN. El Gateway y el servidor que fueron utilizados en el presente proyecto coinciden con la investigación realizada por Cacuango (2022), en ambos trabajos tanto el Gateway como el servidor ostentaron buenos resultados. En el presente estudio se emplea el módulo Tx TTGO esp32 v1 Chip Sx1278 mientras que en el estudio de Cacuango (2022) se utiliza el Heltec Automation LoRa Chip Sx1276. El trabajo de Litardo & Mora (2021) usa un Gateway Dragino LG308 con mejores características con respecto al HT-M00, puesto que este Gateway cuenta con 10 canales y posee conectividad Wifi, Ethernet e incluso celular. El servidor de red que utiliza Litardo & Mora (2021) es TTN, el cual es empleado en la mayoría de proyectos, pero según las indagaciones que efectué sobre el Gateway es que también se lo puede utilizar en otros servidores como por ejemplo LORIOT. El servidor de red LORIOT tiene una lista de Gateways permitidos en la cual no consta el Gateway HT-M00.

En los resultados de la comunicación entre el servidor de red TTN y el servidor de aplicación TagoIO obtuve un panel de monitoreo con la data de cada sensor. TTN es el servidor de red más utilizado en los proyectos de IoT, pero al momento de realizar el panel de monitoreo para la visualización de la información existen diversas plataformas, en el trabajo realizado por Crespin (2020) utiliza a TTN como servidor, pero para crear el panel de visualización utiliza la plataforma IoT “Cayenne”. Por otro lado, Idrovo (2021) en su estudio sobre una agricultura inteligente usa TTN y TagoIO, en su trabajo realza a los 2 servidores por sus excelentes resultados. En el trabajo que lleve a cabo, TagoIO facilito la creación de widgets para cada uno de los sensores, los cuales permitieron que los usuarios visualicen la información de los sensores. Por otra parte, TTN me ayudo a acondicionar la el payload para luego ser enviado a TagoIO.

8. Conclusiones

Se han obtenido buenos resultados en el cumplimiento de los objetivos planteados al comienzo del presente trabajo, obteniendo una red IoT para el monitoreo de la turbidez, temperatura, pH y conductividad eléctrica en la planta potabilizadora del sector de Pucará de la provincia de Loja. A continuación, se presentan las siguientes conclusiones:

- La calidad del agua es importante para la salud de los seres humanos, porque con base en ella se determina si el agua es apta para el consumo humano. Existe un gran número de parámetros que determinan la calidad del agua, pero en este proyecto se definieron 4 parámetros que son meritorios en la calidad del agua, por lo cual se los ha considerado básicos.
- Respecto al diseño de la red IoT, en el optaron por materiales capacitados para la implementación en la planta de Pucará. Además, la versión gratuita del software RadioPlanner permitió crear un diagrama de cobertura de la red IoT. Con respecto a los equipos, el Gateway se conecta al servidor IoT a través del protocolo 802.11 y al módulo Tx atreves del protocolo LoRaWAN, por otra parte, el módulo Tx se conecta al Arduino mediante una comunicación serial UART.
- En relación con la implementación es relevante mencionar al entorno de desarrollo integrado (IDE) de Arduino que permitió programar el módulo Tx y placa Arduino Uno. En este IDE se programó en el módulo Tx los parámetros de vinculación (APPEUI, DEVEUI, APPKEY) a TTN, también se programó la decodificación de la información enviada por comunicación serial y posteriormente se programó el payload. En el Arduino Uno se realizó el programa para recolectar la información de los sensores y enviarla por comunicación serial (UART) al módulo Tx.
- Acerca del análisis, con el sistema de monitoreo IoT se logra un monitoreo mejor en comparación con el método de Jarras utilizado en la planta de Pucará, en los ámbitos de: frecuencia y comodidad.

9. Recomendaciones

En esta sección se presentan algunas sugerencias que se deben considerar para el desarrollo del presente proyecto y para futuros proyectos.

- El Gateway que se está utilizando es de 2 canales con conectividad Wifi. Para este prototipo el Gateway es el adecuado para el lugar de implementación. En una red IoT el Gateway tiene que ser fundamental para soportar características del terreno y la red. Si se desea tener una red más robusta utilizar un Gateway mínimo de 8 canales con conectividad ethernet.
- Los sensores que se implementaron son económicos, por ende, su utilización es compleja en el enfoque de la calidad de precisión y programación. Sugiero emplear sensores gama alta los cuales recolectan mejor los datos, como consecuencia obtendremos un monitoreo más preciso. Además, estos sensores contienen gran cantidad de información en sus hojas de datos, lo cual colabora a su manipulación y programación.
- Si en un trabajo futuro se desea utilizar actuadores para automatizar los procesos de la planta potabilizadora los cuales se hace en base a los datos que se monitorean, recomiendo utilizar otro módulo Tx. El módulo que se usó para el presente proyecto es la versión 1.0 de la serie TTGO ESP32 LoRaWAN, ha ostentado buenos resultados dado que en este caso me he centrado en enviar información, pero en el caso de la automatización también tiene que enfocarse en recibir la información. Un módulo Tx actualizado brindará robustez a la comunicación uplink y downlink.

10. Bibliografía

Alley, R. (2007, April 16). *Water Quality Control Handbook, Second Edition*. https://www.accessengineeringlibrary.com/content/book/9780071467605

Apha. (2017, May 23). *Standard Methods for the Examination of Water and Wastewater, 23rd Edition*. https://engage.awwa.org/PersonifyEbusiness/Store/Product-Details/productId/65266295

BricoGeek. (2022). *TTGO LORA32 ESP32 con OLED - 900Mhz (868Mhz y 915Mhz)*. https://tienda.bricogeek.com/lora/1122-ttgo-lora32-esp32-con-oled-900-mhz.html

Cacuango, R. (2022). *SISTEMA DE MONITOREO DE VARIABLES DE MACRO MEDICIÓN PARA LA GESTIÓN DEL RECURSO HÍDRICO Y EL CONTROL DE CALIDAD DE LA EP-EMASA-PM.* https://repositorio.uta.edu.ec/bitstream/123456789/36145/1/t2053ec.pdf

Campbell, S. (2016, March 6). *Basics of UART Communication*. https://www.circuitbasics.com/basics-uart-communication/

Chatterjee, A. (1998, June 15). *Water Supply, Waste Disposal and Environmental Pollution Engineering: Including Odour, Noise, Air Pollution and Its Control*. https://librarysearch.nirmauni.ac.in/cgi-bin/koha/opac-detail.pl?biblionumber=49304

Chio, N., Tibaduiza, D., Aparicio, L., & Caro, L. (2020, May 23). *Redes de Sensores Inalámbricos*. https://www.academia.edu/5366011/Redes_de_Sensores_Inal%C3%A1mbricos

Cole, S. (1999, October 16). *Guidelines for managing water quality impacts within UK European marine sites*. http://ukmpa.marinebiodiversity.org/uk_sacs/pdfs/water_quality.pdf

Conejeros, A., Hueichaqueo, C., Placeres, A., & Martinez, B. (2021). Water quality monitoring in rural drinking water system. *H2Open Journal*, *1*(2), 160–168. https://doi.org/10.2166/H2OJ.2018.014

Constitución de la Republica del Ecuador. (2008). CONSTITUCIÓN DE LA REPÚBLICA DEL ECUADOR. In *Registro Oficial* (Vol. 449, Issue 20). www.lexis.com.ec

Crespin, J. (2020). *ANÁLISIS Y DISEÑO DE UN PROTOTIPO PARA UN SISTEMA DE CONTROL DE RIEGO AUTOMATIZADO CON MONITOREO Y ALERTAS A DISPOSITIVOS MÓVILES UTILIZANDO ARDUINO, CONECTIVIDAD BLE Y SOFTWARE OPEN SOURCE PARA LOS DIFERENTES CULTIVOS EN EL CANTÓN DAULE PROVINCIA DE GUAYAS*. http://repositorio.ug.edu.ec/bitstream/redug/48784/1/B-CINT-PTG-N.%20486%20Camino%20Crespìn%20Jaime%20Fernando.pdf

Dezuane, J. (1997, January 16). *Handbook of Drinking Water Quality, 2nd Edition*. https://www.wiley.com/en-us/Handbook+of+Drinking+Water+Quality%2C+2nd+Edition-p-9780471287896

DFRobot. (2022a). *Conductivity Electrode User Manual*. https://www.dfrobot.com/product-1123.html

DFRobot. (2022b). *DFR0300_v1.0_schematic*. https://pdfcoffee.com/dfr0300-v10-schematic-pdf-free.html

DFRobot. (2022c). *Turbidity sensor*. https://dfimg.dfrobot.com/nobody/wiki/8e585d98aafe2bab22be39c5b68165c5.pdf

DFRobot. (2023, February 10). *Analog EC Meter SKU DFR0300 DFRobot*. https://wiki.dfrobot.com/Analog_EC_Meter_SKU_DFR0300

DPV TECHNOLOGY. (2019). *Arduino DS18B20 temperature sensor tutorial*. https://www.youtube.com/watch?v=lIpgGru2Wv0

Edison. (2023, January 2). *How to make a Turbidity meter using an Arduino Uno | Calibration - YouTube*. https://www.youtube.com/watch?v=XCOskRNX46I

Edzwald, J. (2011, July 18). *Water Quality & Treatment: A Handbook on Drinking Water, Sixth Edition | McGraw-Hill Education - Access Engineering*. https://www.accessengineeringlibrary.com/content/book/9780071630115

EPA. (2022, June 21). *pH*. https://www.epa.gov/caddis-vol2/ph

ETSI. (2014). *Low Throughput Networks (LTN); Use Cases for Low Throughput Networks*. https://standards.globalspec.com/std/9867413/GS%20LTN%20001

Fafoutis, X., Sørensen, T., & Madsen, J. (2014). Energy Harvesting - Wireless Sensor Networks for Indoors Applications Using IEEE 802.11. *Procedia Computer Science*, *32*, 991–996. https://doi.org/10.1016/J.PROCS.2014.05.523

Garcia, S., & Perez, J. (2022, May 10). *La importancia de la temperatura del agua en las redes de abastecimientoThe importance of water temperature in water supply systems*. https://iwaponline.com/IA/article/26/2/107/88686/La-importancia-de-la-temperatura-del-agua-en-las

Garrido, V., & Peris, M. (2022, July 18). *Smart sensors in environmental/water quality monitoring using IoT and cloud services*. https://www.sciencedirect.com/science/article/abs/pii/S2214158822000204

Gennaro, P., Lofu, D., Vitanio, D., Tedeschi, P., & Boccadoro, P. (2018, September 19). *WaterS: A Sigfox-compliant prototype for water monitoring*. https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.74

GreenPonik. (2019). *EC Meter with ESP32 and DFRobot EC module (DFR0300) - YouTube*. https://www.youtube.com/watch?v=n1EBzMDPI74

Guatemala Digital. (2023, January 12). *Robot PH Value Data Detection and Acquisition Sensor*. https://guatemaladigital.com/Teyleten-Robot-PH-Value-Data-Detection-and-Acquisition-Sensor-Module-Acidity-and-Alkalinity-Sensor-Monitoring-and-Control-ph0-14-for-Arduino/ProductoP/B09H1MJS4S

Hakiri, A. (2015, October 5). *High-level IoT architecture*. https://www.researchgate.net/figure/High-level-IoT-architecture_fig1_281896657

Hassan, N. (2019, October 16). *Water Quality Parameters*. https://www.intechopen.com/chapters/69568

Heltec Automation. (2022, February 16). *HT-M00 Dual Channel LoRa Gateway*. https://heltec.org/project/ht-m00/

Herrera, J. (2006). *UnidadBásicadeComunicaciónserialenunMicrocontrolador*. https://www.redalyc.org/pdf/4026/402640446001.pdf

Idrovo, R. (2021). *Monitorización en la nube de "livestock" para aplicaciones IoT en el ámbito de "Smart Agriculture."* https://www.researchgate.net/profile/Roger-Idrovo/publication/352465870_Monitorizacion_en_la_nube_de_'livestock'_para_aplicaciones_IoT_en_el_ambito_de_'Smart_Agriculture'/links/60ca5861a6fdcc01d47a8f0e/Monitorizacion-en-la-nube-de-livestock-para-aplicaciones-IoT-en-el-ambito-de-Smart-Agriculture.pdf

IMTS LoRaWAN. (2018, September 11). *New 2.4 GHz LoRa module iM282A-L*. https://imst.com/imst/en/news/iM282A-L.php

Jiménez, G., Carolina, L., Quishpe, S., & Mauricio, H. (2020). *Sistema WaterAlert para el monitoreo del agua basado en el paradigma de internet de las cosas (IoT) y la tecnología de comunicación LoRa*. http://repositorio.espe.edu.ec/bitstream/21000/23414/1/T-ESPE-044191.pdf

Khatri, P., Kumar, K., & Gupta, K. (2022, April 8). *Real-time water quality monitoring for distribution networks in IoT environment*. https://www.inderscienceonline.com/doi/abs/10.1504/IJESD.2022.123939

Leo, M., Carli, M., Neri, A., & Battisti, F. (2014, November 5). *A federated architecture approach for Internet of Things security*. https://www.researchgate.net/publication/268147167_A_federated_architecture_approach_for_Internet_of_Things_security

LILYGO. (2022, August 9). *Ttgo Lora Sx1278 Esp32*. https://www.aliexpress.us/item/2251832638444203.html?gatewayAdapt=esp2usa4itemAdapt&_randl_shipto=US

Litardo, R., & Mora, E. (2021). *DISEÑO E IMPLEMENTACIÓN DE UNA RED IOT BASADO EN LORAWAN PARA EL USO DE PROYECTOS DESARROLLADOS POR ESTUDIANTES DE LA UNIVERSIDAD DE GUAYAQUIL, CON UN SISTEMA DE CONTROL DE ACCESO A LA RED IOT*. http://repositorio.ug.edu.ec/bitstream/redug/56463/1/B-CINT-PTG-N.%20698%20Del%20Rosario%20Litardo%20Ra%c3%bal%20Fernando%20%20.%20%20Meza%20Mora%20Eder%20Javier%20.pdf

LoRa Alliance. (2015). *A technical overview of LoRa ® and LoRaWAN ™ What is it?* https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf

LoRa Alliance. (2017, February 13). *Documento técnico de seguridad de LoRaWAN*. https://resources.lora-alliance.org/security/lorawan-security-whitepaper

LoRa Alliance. (2022, April 12). *Homepage - LoRa Alliance*®. https://lora-alliance.org/

Mackenzie, L. (2010, June 18). *Water and Wastewater Engineering: Design Principles and Practice*. https://www.accessengineeringlibrary.com/content/book/9780071713849

Mahalle, P., Anggorojati, B., Prasad, N., & Prasad, R. (2012, October 1). *Identity authentication and capability based access control (IACAC) for the internet of things — Universitas Indonesia*. https://scholar.ui.ac.id/en/publications/identity-authentication-and-capability-based-access-control-iacac

Mahmoud, S., Auday, A., & Mohamad, H. (2016, April 22). *A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT)*

Applications. https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/referencespapers.aspx?referenceid=1739358

Maxim Integrated. (2022). *Sensor de Temperatura (ds18b20)*. https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf

Milind, V., & Prasad, N. (2011, October 5). *A cooperative Internet of Things (IoT) for rural healthcare monitoring and control*. https://www.researchgate.net/publication/224244925_A_cooperative_Internet_of_Things_IoT_for_rural_healthcare_monitoring_and_control

Neumann, P., Montavont, J., & Noel, T. (2016, October 1). *Indoor deployment of low-power wide area networks (LPWAN): A LoRaWAN case study*. https://www.semanticscholar.org/paper/Indoor-deployment-of-low-power-wide-area-networks-A-Neumann-Montavont/569a6e4e91e7abe03879ab1c2e37c077da520ae9

OMS. (2014, October 22). *Calidad del agua*. https://www.un.org/spanish/waterforlifedecade/quality.shtml

Ortiz, A., & Leidy, J. (2011, November 30). *Topología de Red*. https://repositorio.konradlorenz.edu.co/handle/001/3859

Peña, E., & Legaspi, M. (2020, June 8). *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices*. https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

Pérez, E. (2016, September 3). *Control de calidad en aguas para consumo humano en la región occidental de Costa Rica*. https://www.scielo.sa.cr/scielo.php script=sci_arttext&pid=S0379 39822016000300003#B14

Petäjäjärvi, J., Mikhaylov, K., Pettissalo, M., Janhunen, J., & Iinatti, J. (2017). Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage. *Undefined*, *13*(3). https://doi.org/10.1177/1550147717699412

Pini, A. (2019, May 22). *Los UART proporcionan una comunicación serial confiable | DigiKey*. https://www.digikey.com/es/articles/uarts-ensure-reliable-long-haul-industrial-communications

Pradillo, B. (2016, September 12). *Parámetros de control del agua potable*. https://www.iagua.es/blogs/beatriz-pradillo/parametros-control-agua-potable

Robocraze. (2022, February 5). *pH Sensor Kit*. https://robocraze.com/products/ph-sensor-kit

Robots Argentina. (2020, July 20). *¿Qué es la comunicación serie?* https://robots-argentina.com.ar/didactica/que-es-la-comunicacion-serie/

Rowe, R. (1985, May 24). *Environmental engineering (Libro, 1985)*. https://www.worldcat.org/title/environmental-engineering/oclc/10430912

Rusydi, A. (2018, October 11). *Correlation between conductivity and total dissolved solid in various type of water*. https://iopscience.iop.org/article/10.1088/1755-1315/118/1/012019/meta

Samboni, N., & Carvajal, Y. (2007, October 22). *Revisión de parámetros fisicoquímicos como indicadores de calidad y contaminación del agua*. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=s0120-56092007000300019

Santos, S. (2021, April 26). *ESP32 with Built-in SX1276 LoRa and SSD1306 OLED Display*. https://makeradvisor.com/esp32-sx1276-lora-ssd1306-oled/

Sayer, A. (1997, June 1). *Microbial solubilization and immobilization of toxic metals: key biogeochemical processes for treatment of contamination*. https://academic.oup.com/femsre/article/20/3-4/503/516842

Schwartz, J., Levin, R., & Goldstein, R. (2009, June 11). *Drinking water turbidity and gastrointestinal illness in the elderly of Philadelphia*. https://jech.bmj.com/content/54/1/45.short

Semtech. (2020). *LoRa and LoRaWAN: A Technical Overview LoRa® and LoRaWAN®: A Technical Overview*. https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf

Semtech. (2022). *RF Wireless | LoRa Wireless Products | Semtech | Semtech*. https://www.semtech.com/products/wireless-rf

Sendra, S., Parra, L., Jimenez, J., Garcia, L., & Lloret, J. (2022, July 12). *LoRa-based Network for Water Quality Monitoring in Coastal Areas*. https://link.springer.com/article/10.1007/s11036-022-01994-8

Shancang, L., Li Da, X., & Shanshan, Z. (2015, April 11). *The internet of things: a survey*. https://uwe-repository.worktribe.com/output/836601/the-internet-of-things-a-survey

Sigfox. (2022). *Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)*. https://www.sigfox.com/en

Silva, D., & Coello, S. (2020). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO EN TIEMPO REAL DE SENSORES DE TEMPERATURA, TURBIDEZ, TDS Y PH PARA LA CALIDAD DEL AGUA UTILIZANDO LA TECNOLOGÍA LORAWAN*. https://dspace.ups.edu.ec/bitstream/123456789/19627/1/UPS-GT003087.pdf

Spellman, F. (2017, October 12). *The Drinking Water Handbook*. https://www.taylorfrancis.com/books/mono/10.1201/9781315159126/drinking-water-handbook-frank-spellman

Sun Robotronics. (2020). *PH sensor working and calibration / water quality monitoring using Arduino (attach wifi for IOT) - YouTube*. https://www.youtube.com/watch?v=5Vu10e5qSss

TechDesing. (2021, May 17). *Quick Guide to Understand LoRa and LoRa Modules*. https://blog.techdesign.com/quick-guide-to-understand-lora-and-lora-modules/

The Things Network. (2021, April 15). *LoRaWAN Architecture*. https://www.thethingsnetwork.org/docs/lorawan/architecture/

The Things Network. (2022, March 14). *Device Classes*. https://www.thethingsnetwork.org/docs/lorawan/classes/

Tienda de Robótica. (2022). *Una guía práctica sobre el mundo de Arduino*. https://www.maristashuelva.es/webinfo/tecnologia/arduino/Libro_kit_Basico.pdf

Universiti Putra Malaysia. (2014, April 6). *Current test' for water pollution*. https://www.sciencedaily.com/releases/2014/03/140306132813.htm

Vikesland, P. (2018, November 2). *Nanosensors for water quality monitoring*. https://www.nature.com/articles/s41565-018-0209-9

Violino, B. (2014, September 15). *Top IT vendors reveal their IoT strategies*. https://www.networkworld.com/article/2604766/top-it-vendors-reveal-their-iot-strategies.html

Wang, Y., Lin, X., Adhikary, A., Grovlen, A., Sui, Y., Blankenship, Y., & Razaghi, H. (2017, May 11). *The Adaptive Random Access Carrier Allocation Scheme in NB-IoT Networks*. https://acortar.link/enPdHm

Wang, Y., & Lv, F. (2022, July 18). *Design of water quality monitoring system based on NB-IoT technolog*. https://ieeexplore.ieee.org/abstract/document/9824243

Water Science School. (2018, June 6). *Temperature and Water*. https://www.usgs.gov/special-topics/water-science-school/science/temperature-and-water

11. Anexos

En esta sección se presenta información relevante respecto a los equipos utilizados y la programación de las diferentes placas.

Anexo 1. Especificaciones del sensor de Temperatura (DS18B20)

A continuación, se presenta el Anexo 1 que hace referencia al sensor de temperatura DS18B20. Toda la información la podemos encontrar en la publicación de Maxim Integrated (2022). La publicación de DPV TECHNOLOGY (2019) en You Tube sirvió como referencia para la programación. A continuación, se presenta información sobre este sensor:

DS18B20 **Programmable Resolution 1-Wire Digital Thermometer**

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

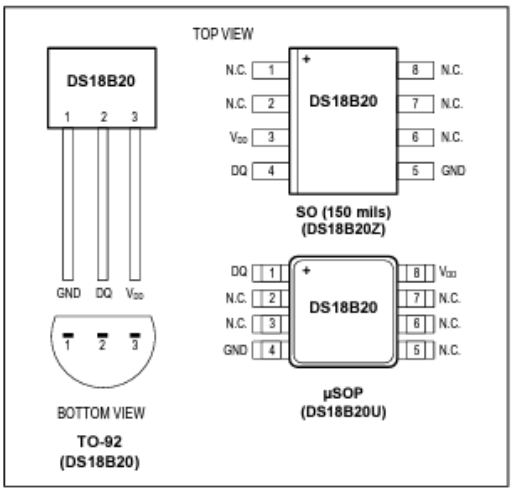
Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



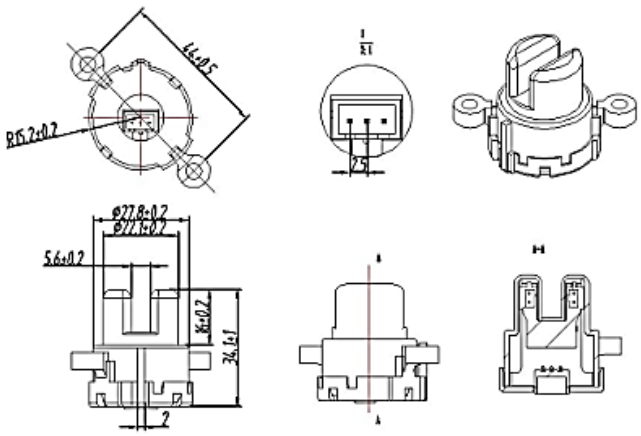
Ordering Information ***appears at end of data sheet.***

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

Anexo 2. Especificaciones del sensor de Turbidez

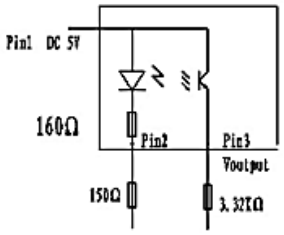
El datasheet específico del sensor de Turbidez que se utilizó es difícil de encontrar, pero a continuación presento una hoja de datos de un sensor similar. Toda la información la pueden encontrar la publicación realizada por DFRobot, (2022c). En la publicación de Edison (2023) se muestra un video que sirvió para la programación. A continuación, se muestran algunas imágenes sobre este sensor:

4. Outline Dimensions:

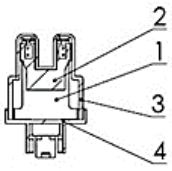


5. Internal Structure:

The sensor and its test circuit



6. The Brand, Parameters And Composition of the Key Components:



No.	Name	Material Composition	Environmental Standards
1	PCB Components	CM-3, photosensitive element	RoHS
2	Support	PA6+15%	RoHS
3	Shell	PP	RoHS
4	Back Cover	PA6+15%	RoHS

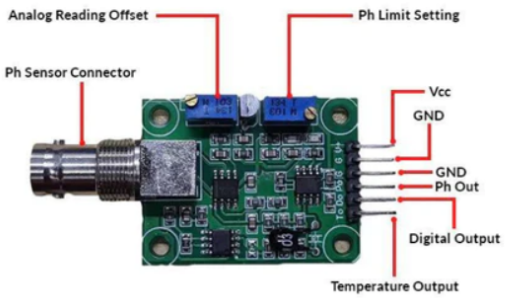
7. Electric Performance Parameters

- Rated Voltage: DC 5 V
- Rated Current: 30mA
- Withstand Voltage: No flash-over or breakdown when applying AC 1500V, 50Hz voltage between the energized terminal and shell for 1S.
- Leakage Current: Leakage between coil and shell < 0.25mA
- Insulation Resistance: The insulation resistance should be greater than 100MΩ when the voltage of DC 500V is applied between the charged spot and the exposed non-charged metal and non-metal.
- Turbidity Reference Curve:

Anexo 3. Especificaciones del sensor de Potencial de Hidrógeno

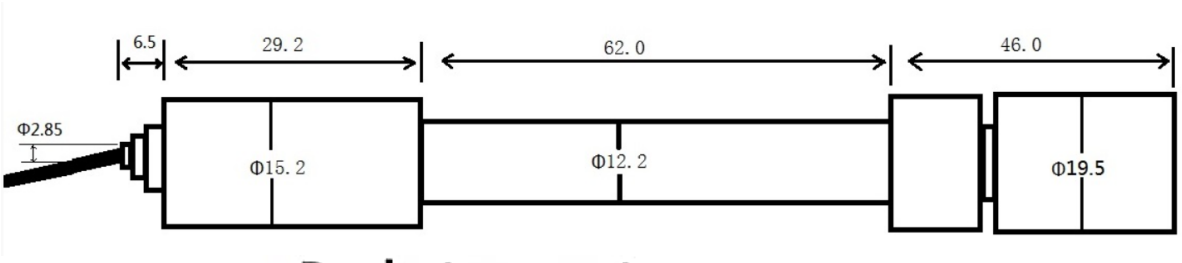
El Anexo 3 especifica la información del sensor de pH. No se ha encontrado un datasheet especifico, pero en algunas páginas en línea encontré información relevante que sirvió para el proyecto. La información completa la encontramos en la publicación realizada por Guatemala Digital (2023). En la publicación realizada por Sun Robotronics (2020) se presenta un video el cual sirvió como referencia para la programación.

PINOUT DEL SENSOR DE SONDA DE PH:



Asignación de pines del sensor de PH

- TO – Salida de temperatura
- DO - Salida de 3,3 V (del potenciómetro de límite de ph)
- Salida analógica PO – PH ==> Arduino A0
- Gnd - Gnd para la sonda PH (puede provenir del pin Arduino GND) ==> Arduino GND
- Gnd – Gnd para la placa (también puede provenir del pin Arduino GND) ==> Arduino GND
- VCC - 5V DC (puede provenir del pin Arduino 5V) ==> pin Arduino 5V
- POT 1: desplazamiento de lectura analógica (más cercano al conector BNC)
- POT 2 – Configuración del límite de PH



· Product parameter

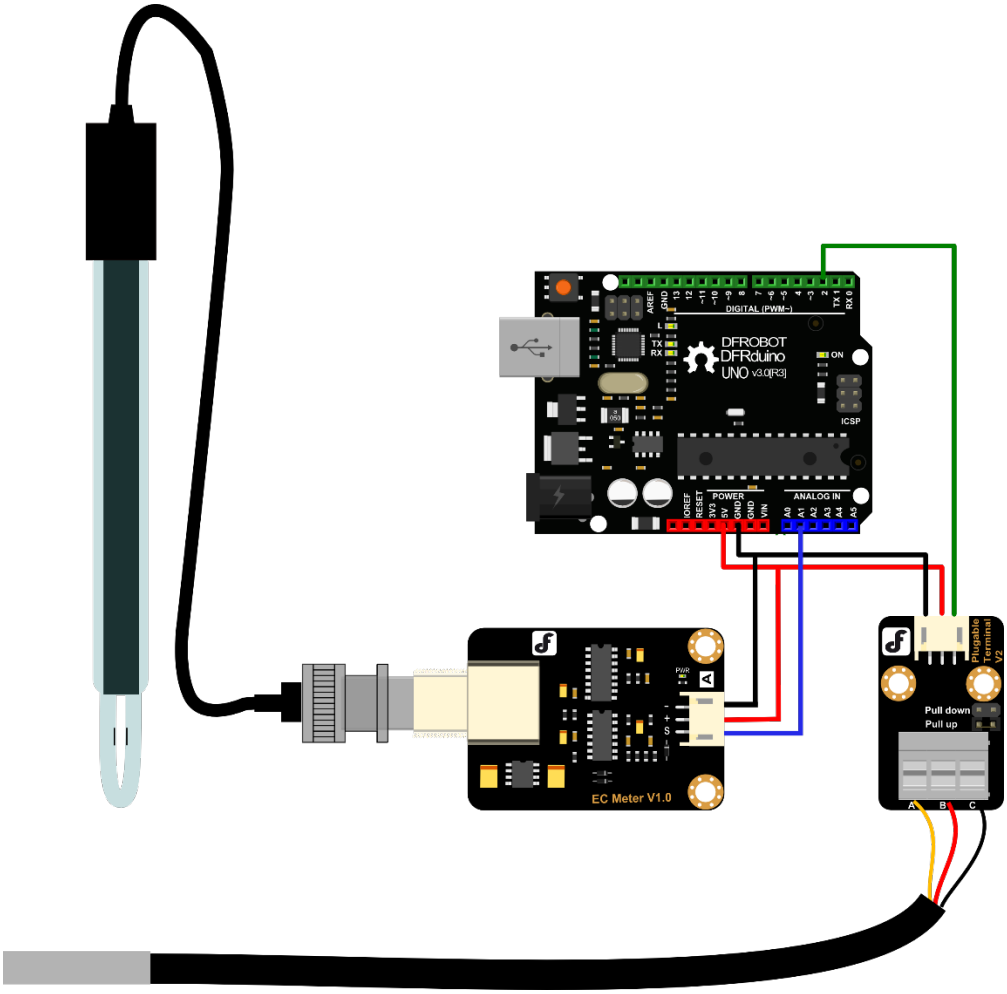
1. Heating voltage: 5 ± 0.2V (DC)
2. Working current: 5-10mA
3. Detection concentration range: PH0-14
4. Detection temperature range: 0-80°C
5. Response time: ⩽5S
6. Stability time: ⩽60S
7. Component power consumption: ⩽0.5W
8. Working temperature: -10~50°C (nominal temperature 20°C)
9. Working humidity: 95%RH (nominal humidity 65%RH)
10. Output mode: analog voltage signal output
11. With temperature compensation output

Anexo 4. Especificaciones del sensor de Conductividad Eléctrica

En este Anexo se presenta la hoja de datos del sensor de conductividad eléctrica. Específicamente este sensor es de marca DFRobot. Su hoja de datos se encuentra en la publiacion realizada por DFRobot (2023), el manual en el siguiente enlace (DFRobot, 2022a) y un esquema (DFRobot, 2022b). La publicación realizada por GreenPonik (2019) sirvió como referencia para la programación del sensor. A continuación, se presentan algunas imágenes referentes a este sensor:

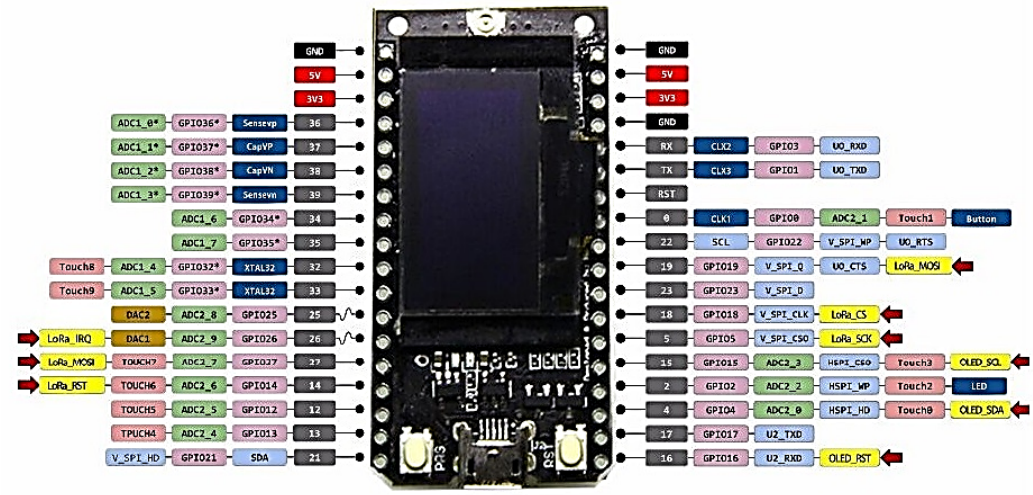
Specification

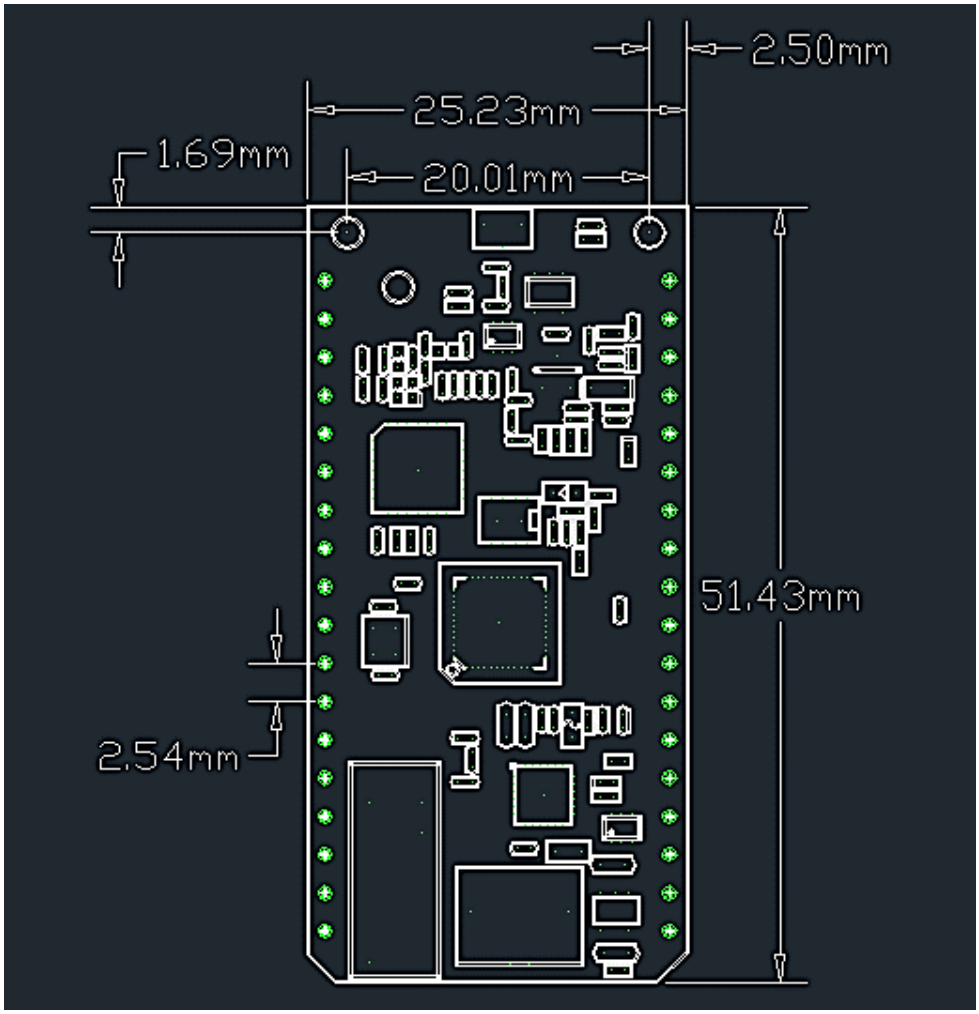
- Operating Voltage: +5.00 V
- PCB Size: 45mm × 32mm
- Measuring Range: 1ms/cm~20ms/cm
- Operating Temperature: 5-40 ℃
- Accuracy: <±10% F.S (specific accuracy depends on the accuracy of your calibration soli
- PH2.0 Interface (3-pin SMD)
- Conductivity Electrode (Electrode Constant K = 1, BNC connector)
- Cable Length of the Electrode: about 60cm
- DS18B20 Temperature Sensor (Waterproof)
- Power Indicator



Anexo 5. Especificaciones del Tx TTGO SX1276 SX1278 LoRa ESP32 868 / 915MHz

La información del módulo Tx se la puede encontrar en varias partes, he colocado la publicación realizada por BricoGeek (2022) en la cual se encuentra la información completa. Existe una guía la cual es muy explicativa y la podemos encontrar en la publicación de Santos (2021). A continuación, se presentan algunas imágenes relacionadas al módulo Tx:





Anexo 6. Especificaciones del LoRa Gateway HT-M00

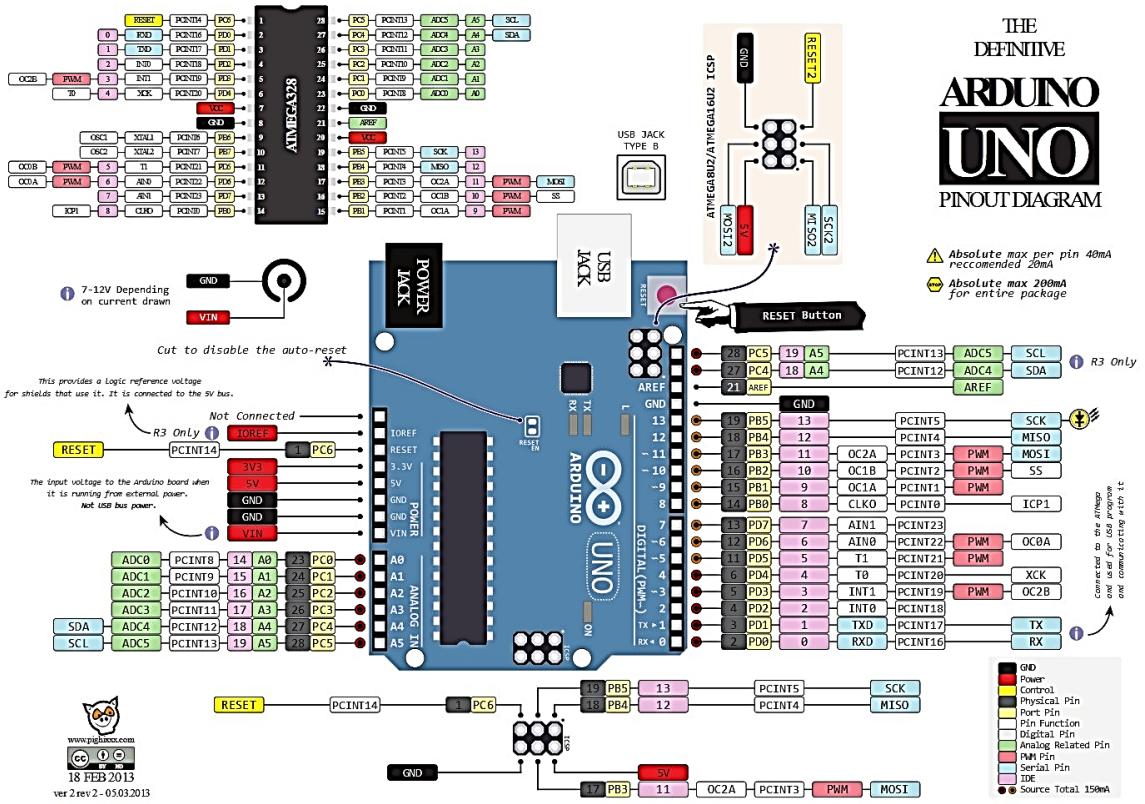
Este anexo representa la información relevante del Gateway HT-M00. La información completa la podemos encontrar en la página de su fabricante (Heltec Automation, 2022). A continuación, se presenta información relevante sobre este equipo:

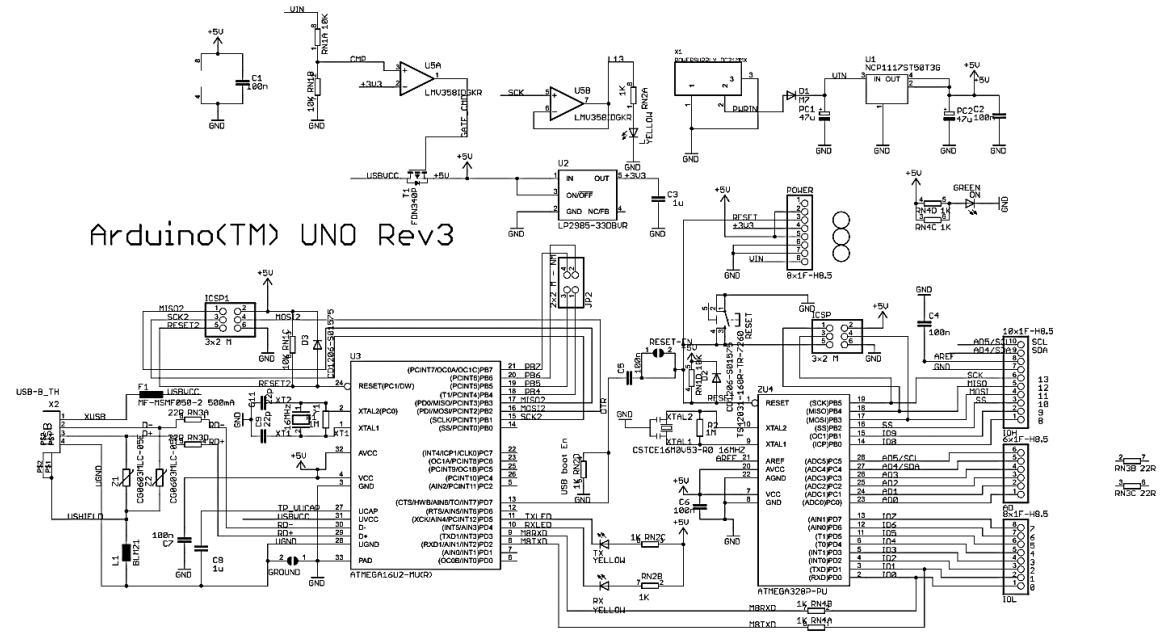
No.	Model	Description
1	HT-M00-470T510	470~510MHz working LoRa frequency, used for China mainland (CN470) LPW band.
2	HT-M00 -863T870	863~870MHz working LoRa frequency, used for EU868, IN865 LPW bands.
3	HT-M00-902T923	902~923MHz working frequency, used for AS923, US915, AU915, KR920 LPW bands.

Parameters	Description
MCU	ESP32-D0WDQ6
LoRa Chipset	SX1276
Receiving Sensitivity	-110dBm @ 300bps
Frequency	863～870MHz, 902～928MHz, 470～510MHz
Interface	Type-C USB x 1
Max. TX Power	17dB ± 1dB
Operating temperature	-20 ~ 70 ℃
Dimensions	30 x 76 x14 mm

Anexo 7. Especificaciones del Arduino Uno

En este Anexo se presenta la disposición de los pines y el esquema del Arduino Uno. También se presenta información sobre el regulador de voltaje que utiliza esta placa. La información completa la podemos encontrar en la publicación realizada por Tienda de Robótica (2022).





Anexo 8. Programa en Arduino Uno

```
//Libreria Turbidez
#include <Wire.h>

//Librerias Temperatura
#include <OneWire.h>
#include <DallasTemperature.h>

//Librerias Conductividad electrica
#include "DFRobot_EC10.h"
#include <EEPROM.h>
#define EC_PIN A4

//Variables Temperatura
const int DataTem = 8;
OneWire oneWireObjeto(DataTem);
DallasTemperature sensorDS18B20(&oneWireObjeto);

//Variables Turbidez
int PinTur = A0;
float VolTur;
int samples = 600;
float ntu; // Nephelometric Turbidity Units

//Variables EC
float ecValue,temp = 25;
DFRobot_EC10 ec;

//PayLoad
int datos [5] = {0,0,0,0,0};
char Ide [5] = {'z','a', 'b', 'c', 'd'};
String txt = "";

float readTem(){
  sensorDS18B20.begin();
  sensorDS18B20.requestTemperatures();
  float Tem = sensorDS18B20.getTempCByIndex(0);
  float Temperatura = Tem * 100;
  return Temperatura;
}

float readTur(){
  float VolTur = 0;
  float ntu = 0;
  int samples = 600;
  for(int i=0; i<samples; i++) {
    VolTur += ((float)analogRead(PinTur)/1024)*4.5;
    }

  VolTur = VolTur/samples;
  VolTur = round_to_dp(VolTur,2);

  if(VolTur > 3){
      ntu = (((-1120.4*square(VolTur) + 5742.3*VolTur - 4352.9))/10000);
    }else if(VolTur > 2.45 && VolTur < 3.1){
      ntu = (((-1120.4*square(VolTur) + 5742.3*VolTur - 4352.9))/10000);
```

```
    }else if (VolTur > 2.35 && VolTur < 2.46){
      ntu = (((-1120.4*square(VolTur) + 5742.3*VolTur - 4352.9))/10000);
    }else if (VolTur > 2.29 && VolTur <2.36 ){
      ntu = (((-1120.4*square(VolTur) + 5742.3*VolTur - 4352.9))/10000) + 10;
    }else if (VolTur > 2.25 && VolTur < 2.30 ){
      ntu = (((-1120.4*square(VolTur) + 5742.3*VolTur - 4352.9))/10000) + 50;
    }
    ntu = ntu*100;
    return ntu;
}

float round_to_dp( float in_value, int decimal_place )
{
  float multiplier = powf( 10.0f, decimal_place );
  in_value = roundf( in_value * multiplier ) / multiplier;
  return in_value;
}

 float readPH(){
   float MuesPH = 0;
   float VolPH = 0;
   int SamPH = 600;

   for(int i=0; i<SamPH; i++) {
    int val = analogRead(A2);
    MuesPH += float (val)/1024*5;
    //VolPH += ((float)analogRead(A1)/1024)*4.5;
  }
  VolPH = MuesPH/SamPH;
  float pHValue = 0;
  if (VolPH > 0 && VolPH < 1.5){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)-2;
  }else if (VolPH >1.49 && VolPH < 2){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10);
  }else if (VolPH >1.99 && VolPH < 2.5){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+1;
  }else if (VolPH >2.49 && VolPH < 3){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+3;
  }else if (VolPH >2.99 && VolPH < 3.5){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+5;
  }else if (VolPH >3.45 && VolPH < 4){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+7;
  }else if (VolPH >3.99 && VolPH < 5){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+8;
  }else if (VolPH > 4.99){
    pHValue = (((5.70 * VolPH + 21.34 + 1.5))/10)+9;
  }
  /*Serial.println("Voltaje: ");
  Serial.println(VolPH);
  Serial.println("pH: ");
  Serial.println(pHValue);*/
  float VPH = pHValue * 100;
  //Serial.println(VolPH);
  return VPH;
 }

 float readEC(){
   static unsigned long timepoint = millis();
  if(millis()-timepoint>1000U)  //time interval: 1s
```

```
  {
    int SamCon = 800;
    float ProCon = 0;
    float voltage = 0;
    for (int i = 0; i<SamCon ; i++){

      timepoint = millis();
      float voltageCon = analogRead(EC_PIN)/1024.0*5000;   // read the
voltage        // read your temperature sensor to execute temperature compensation
      ProCon = ProCon + voltageCon;

    }
    voltage = ProCon/800;
    if (voltage > 0 && voltage < 1.5 ){
      ecValue =  ec.readEC(voltage,temp) - 9;
    } else if (voltage > 1.49 && voltage < 2){
      ecValue =  (ec.readEC(voltage,temp)) - 5;
    }else if(voltage > 1.99 && voltage < 2.5){
      ecValue =  (ec.readEC(voltage,temp));
    }else if(voltage > 2.49 && voltage < 3){
      ecValue =  (ec.readEC(voltage,temp)) + 5;
    } else if (voltage > 2.99 && voltage < 3.5){
      ecValue =  (ec.readEC(voltage,temp)) + 10;
    }else if (voltage > 3.45 && voltage < 4){
      ecValue =  (ec.readEC(voltage,temp)) + 15;
    }else if (voltage > 3.99 && voltage < 4.5){
      ecValue =  (ec.readEC(voltage,temp)) + 20;
    }else if (voltage > 4.99 && voltage < 5){
      ecValue =  (ec.readEC(voltage,temp)) + 25;
    }else if (voltage > 4.99){
      ecValue =  (ec.readEC(voltage,temp)) + 50;
    }
    //ecValue = voltage;
    }
    ecValue = ecValue*100;
    return ecValue;
 }


void setup() {
  Serial.begin(9600);

  //Turbidez
  pinMode(A0, INPUT);

  //PH
  pinMode (A1,INPUT);

  //EC
  ec.begin();
}
void loop() {
  int Control = 10;
  datos [0]=Control;

  int Tra = ((int)(Temra));
  datos [1] = Tra;

  //Turbidez
```

```
  float UNTU= readTur();
  int Tudez = ((int)(UNTU));
  datos [2] = Tudez;
  //datos [4];

  //ConductividadElectrica.
  float UCON = readEC();
  int Condad = ((int)(UCON));
  datos [3] = Condad;
  //datos [6];

  //PotencialDeHidrogeno
  float UPH = readPH();
  int PoHi = ((int)(UPH));
  datos [4] = PoHi;

  String txt = "";

  for (int x = 0; x<5; x++){
    txt += datos [x];
    txt += Ide [x];
  }

Serial.println(txt);
delay(1500);

}
```

Anexo 9. Programa en el módulo Tx.

```
//LORAWAN
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

//ComunicacionUart
#define RXD2 16
#define TXD2 17

//PayLoad
String data = "";

//Contador de variables
int ConTem;
int ConTur;
int ConPo;
int ConCon;

//Guardar el valor entero.
int numTem;
int numTur;
int numPo;
int numCon;

//UnirArrayenString
String Temx = "";
String Tur = "";
String Po = "";
String Con = ""

// Formato lsb TTN.
static const u1_t PROGMEM APPEUI[8]={ 0xE3, 0xD5, 0xB1, 0xC9, 0xD8, 0xA7, 0xC6,
0xE5 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

//Formato lsb TTN.
static const u1_t PROGMEM DEVEUI[8]={ 0x11, 0x77, 0x05, 0xD0, 0x7E, 0xD5, 0xB3,
0x70 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// Formato msb TTN.
static const u1_t PROGMEM APPKEY[16] = { 0x47, 0xB3, 0x3E, 0x19, 0x47, 0x92, 0x83,
0x55, 0x49, 0xB3, 0x04, 0x00, 0xAE, 0xDE, 0x23, 0x1E };
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static uint8_t payload[8];
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
```

```
    .dio = {26, 33, 32},
};

//Funcion para el sensor de Temperatura.
int readTem() {

data = Serial2.readString();

int ConTem = 0;
int ConControl = 0;

  for(size_t i = 0; i<data.length(); i++){
    if (data.charAt(i) == 'z'){
      ConControl = i;
     }else if (data.charAt(i) == 'a'){
       ConTem = i-ConControl-1;
     }
   }

  //Crear un string para cada una de las variables
  char Tem [ConTem];

  //Contador
  int a=2;
  for (int i = 0; i<ConTem;i++){
    a++;
    Tem[i]=data.charAt(a);
  }

  //StringSensores
  String Temx = "";

  //Unir arreglos
  for (int x = 0; x<ConTem; x++){
    Temx += Tem [x];
  }
  numTem = Temx.toInt();
  return numTem;
}

//Funcion para el sensor de Turbidez
int readTur(){
data = Serial2.readString();
int ConTem = 0;
int ConTur = 0;
int ConControl = 0;
  for(size_t i = 0; i<data.length(); i++){
    if (data.charAt(i) == 'z'){
      ConControl = i;
     }else if (data.charAt(i) == 'a'){
       ConTem = i-ConControl-1;
     }else if (data.charAt(i) == 'b'){
       ConTur = i-ConTem-ConControl-2;
     }
   }

  //Crear un string para cada una de las variables
  char Tem [ConTem];
  char Tur [ConTur];
```

```
  //Contador
  int a=2;
  for (int i = 0; i<ConTem;i++){
    a++;
    Tem[i]=data.charAt(a);
  }
  int b=ConTem+3;
  for (int i = 0; i<ConTur;i++){
    b++;
    Tur[i]=data.charAt(b);
  }

  //StringSensores
  String Turx = "";

  //Unir arreglos
    for (int x = 0; x<ConTur; x++){
    Turx += Tur [x];
  }

  numTur = Turx.toInt();

  return numTur;
}

//Funcion Potencial de Hidrogeno
int readPH(){
data = Serial2.readString();
int ConTem = 0;
int ConTur = 0;
int ConPo = 0;
int ConControl = 0;
  for(size_t i = 0; i<data.length(); i++){
    if (data.charAt(i) == 'z'){
      ConControl = i;
     }else if (data.charAt(i) == 'a'){
       ConTem = i-ConControl-1;
     }else if (data.charAt(i) == 'b'){
       ConTur = i-ConTem-ConControl-2;
     }else if (data.charAt(i) == 'c'){
       ConPo = i-ConTur-ConTem-ConControl-3;
     }
   }

  //Crear un string para cada una de las variables
  char Po [ConPo];

  //Contador
  int c = ConTur+ConTem+4;
  for (int i = 0; i<ConPo;i++){
    c++;
    Po[i]=data.charAt(c);
  }

  //StringSensores
  String Pox = "";

  //Unir arreglos
```

```
    for (int x = 0; x<ConPo; x++){
    Pox += Po [x];
  }

  numPo = Pox.toInt();

  return numPo;
}


//FUNCION Conductividad Electrica
int readCon(){
data = Serial2.readString();
int ConTem = 0;
int ConTur = 0;
int ConPo = 0;
int ConCon = 0;
int ConControl = 0;
  for(size_t i = 0; i<data.length(); i++){
    if (data.charAt(i) == 'z'){
      ConControl = i;
     }else if (data.charAt(i) == 'a'){
       ConTem = i-ConControl-1;
     }else if (data.charAt(i) == 'b'){
       ConTur = i-ConTem-ConControl-2;
     }else if (data.charAt(i) == 'c'){
       ConPo = i-ConTur-ConTem-ConControl-3;
     }else if (data.charAt(i) == 'd'){
       ConCon = i-ConPo-ConTur-ConTem-ConControl-4;
     }
   }

  //Crear un string para cada una de las variables
  char Con [ConCon];

  int d = ConTur+ConTem+ConPo+5;
  for (int i = 0; i<ConCon;i++){
    d++;
    Con[i]=data.charAt(d);
  }

  //StringSensores
  String Conx = "";

  //Unir arreglos
    for (int x = 0; x<ConCon; x++){
    Conx += Con [x];
  }

  numCon = Conx.toInt();
  return numCon;
}


void printHex2(unsigned v) {
    v &= 0xff;
    if (v < 16)
        Serial.print('0');
    Serial.print(v, HEX);
```

```
}

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_SCAN_TIMEOUT:
            Serial.println(F("EV_SCAN_TIMEOUT"));
            break;
        case EV_BEACON_FOUND:
            Serial.println(F("EV_BEACON_FOUND"));
            break;
        case EV_BEACON_MISSED:
            Serial.println(F("EV_BEACON_MISSED"));
            break;
        case EV_BEACON_TRACKED:
            Serial.println(F("EV_BEACON_TRACKED"));
            break;
        case EV_JOINING:
            Serial.println(F("EV_JOINING"));
            break;
        case EV_JOINED:
            Serial.println(F("EV_JOINED"));
            {
              u4_t netid = 0;
              devaddr_t devaddr = 0;
              u1_t nwkKey[16];
              u1_t artKey[16];
              LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);
              Serial.print("netid: ");
              Serial.println(netid, DEC);
              Serial.print("devaddr: ");
              Serial.println(devaddr, HEX);
              Serial.print("AppSKey: ");
              for (size_t i=0; i<sizeof(artKey); ++i) {
                if (i != 0)
                  Serial.print("-");
                printHex2(artKey[i]);
              }
              Serial.println("");
              Serial.print("NwkSKey: ");
              for (size_t i=0; i<sizeof(nwkKey); ++i) {
                      if (i != 0)
                              Serial.print("-");
                      printHex2(nwkKey[i]);
              }
              Serial.println();
            }
            // Disable link check validation (automatically enabled
            // during join, but because slow data rates change max TX
      // size, we don't use it in this example.
            LMIC_setLinkCheckMode(0);
            break;
        /*
        || This event is defined but not used in the code. No
        || point in wasting codespace on it.
        ||
        || case EV_RFU1:
        ||     Serial.println(F("EV_RFU1"));
```

```
        ||     break;
        */
        case EV_JOIN_FAILED:
            Serial.println(F("EV_JOIN_FAILED"));
            break;
        case EV_REJOIN_FAILED:
            Serial.println(F("EV_REJOIN_FAILED"));
            break;
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.print(F("Received "));
              Serial.print(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            // Schedule next transmission
            os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL),
do_send);
            break;
        case EV_LOST_TSYNC:
            Serial.println(F("EV_LOST_TSYNC"));
            break;
        case EV_RESET:
            Serial.println(F("EV_RESET"));
            break;
        case EV_RXCOMPLETE:
            // data received in ping slot
            Serial.println(F("EV_RXCOMPLETE"));
            break;
        case EV_LINK_DEAD:
            Serial.println(F("EV_LINK_DEAD"));
            break;
        case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
        case EV_TXSTART:
            Serial.println(F("EV_TXSTART"));
            break;
        case EV_TXCANCELED:
            Serial.println(F("EV_TXCANCELED"));
            break;
        case EV_RXSTART:
            /* do not print anything -- it wrecks timing */
            break;
        case EV_JOIN_TXCOMPLETE:
            Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));
            break;

        default:
            Serial.print(F("Unknown event: "));
            Serial.println((unsigned) ev);
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
```

```
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
      // Prepare upstream data transmission at the next possible time.
     // uint8_t payload [8];
      int ValTem = readTem();
      payload [0] = ValTem >> 8;
      payload [1] = ValTem;
      int ValTur = readTur();
      payload [2] = ValTur >> 8;
      payload [3] = ValTur;
      int ValPo = readPH();
      payload [4] = ValPo >> 8;
      payload [5] = ValPo;
      int ValCon = readCon();
      payload [6] = ValCon >> 8;
      payload [7] = ValCon;

      Serial.println(ValTem);
      Serial.println(ValTur);
      Serial.println(ValPo);
      Serial.println(ValCon);
//hasta 65 535
      LMIC_setTxData2(1, payload, sizeof(payload), 0);
      Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
  //ComunicacionUart
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  Serial.println(F("Starting"));
  #ifdef VCC_ENABLE
  // For Pinoccio Scout boards
  pinMode(VCC_ENABLE, OUTPUT);
  digitalWrite(VCC_ENABLE, HIGH);
  delay(1000);
  #endif
  // LMIC init
  os_init();
  // Reset the MAC state. Session and pending data transfers will be discarded.
  LMIC_reset();
  // Start job (sending automatically starts OTAA too)
  do_send(&sendjob);

}

void loop() {

    os_runloop_once();
}
```

Anexo 10. Data recolectada en el sistema IoT implementado

Anexo 11. Reporte del diagrama de cobertura

Del diagrama de cobertura presentado en la sección de diseño RadioPlanner 2.1 genero un reporte el cual consta podemos observar algunas características de la red IoT.

Project name:	Red IoT para el monitoreo de los parametros basicos de la calidad del agua
Customer:	Edwin Abel Jimbo Sarmiento
Data:	2022/11/13 21:39
Radio System Type:	Mobile Radio Communication
Frequency:	923 MHz
Propagation Model Type:	ITU-R P.1812-4
Percentage of time:	95%
Percentage of location:	95%
Margin:	0 dB
Mobile unit location:	Mobile unit in rural areas
Clutter loss:	Yes
Area Study Type:	Received power Downlink
Co-channel interference:	No
Adjacent channel interference:	No

	Mobile Unit №1		Coverage area
	> -80 dBm	Excelente	0,1 km²
	> -90 dBm	Muy Buena	0,4 km²
	> -100 dBm	Buena	4,9 km²
	> -115 dBm	Regular	18,9 km²
	> -120 dBm	Aceptable	21,8 km²
	> -125 dBm	Mala	25 km²
	> -130 dBm	Pesima	30,1 km²
	> -137 dBm	Sin Conexion	37,9 km²

Base Stations Parameters

№	Name	Latitude Longitude	Sector azimuth	Antenna model	Antenna height	Antenna beam tilt	Antenna gain, dBi	Tx power, W	Loss, dB
1	GATEWAY	S04,006843°	0°	Omni	4 m	0°	2	0,025	0,8
		W79,196441°							

Mobile Units Parameters

Name	Tx power, W	Rx threshold, dBm	Cable and connectors loss, db	Antenna height	Antenna gain, dBi
NodoPucara	0,025	-137	1	1 m	1
Mobile Unit №2	1	-100	0	3 m	3

Clutter loss

Clutter type	Mobile Unit №1 loss, dB	Mobile Unit №2 loss, dB	Clutter height, m
Open/rural:	0	0	0
Water:	13	2,7	4
Trees:	23	21,7	15
Suburban:	19,3	17,2	10
Urban:	23	21,7	15
Dense urban:	25,4	24,5	20

Anexo 12. Solicitud presentada a la UMAPAL de Loja.

Loja, 10 diciembre del 2022

Ing. Jacqueline Jaramillo Jaramillo.
Directora de UMAPAL

Presente -

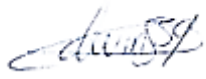
De mis consideraciones:

Yo, Edwin Abel Jimbo Sarmiento, con cédula de ciudadanía 1105824294, estudiante del **Noveno Ciclo** la **carrera de Telecomunicaciones** de la **Universidad Nacional de Loja**. Actualmente estoy desarrollando el proyecto titulado **"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO BASADO EN IOT PARA EL MONITOREO DE LOS PARÁMETROS BÁSICOS DE LA CALIDAD DEL AGUA EN LA PLANTA POTABILIZADORA DEL SECTOR DE PUCARA".**
Solicito muy comedidamente si es posible su autorización para evaluar el sistema de monitoreo en la planta potabilizadora ubicada en el sector de Pucara.

Los objetivos principales del proyecto son los siguientes:

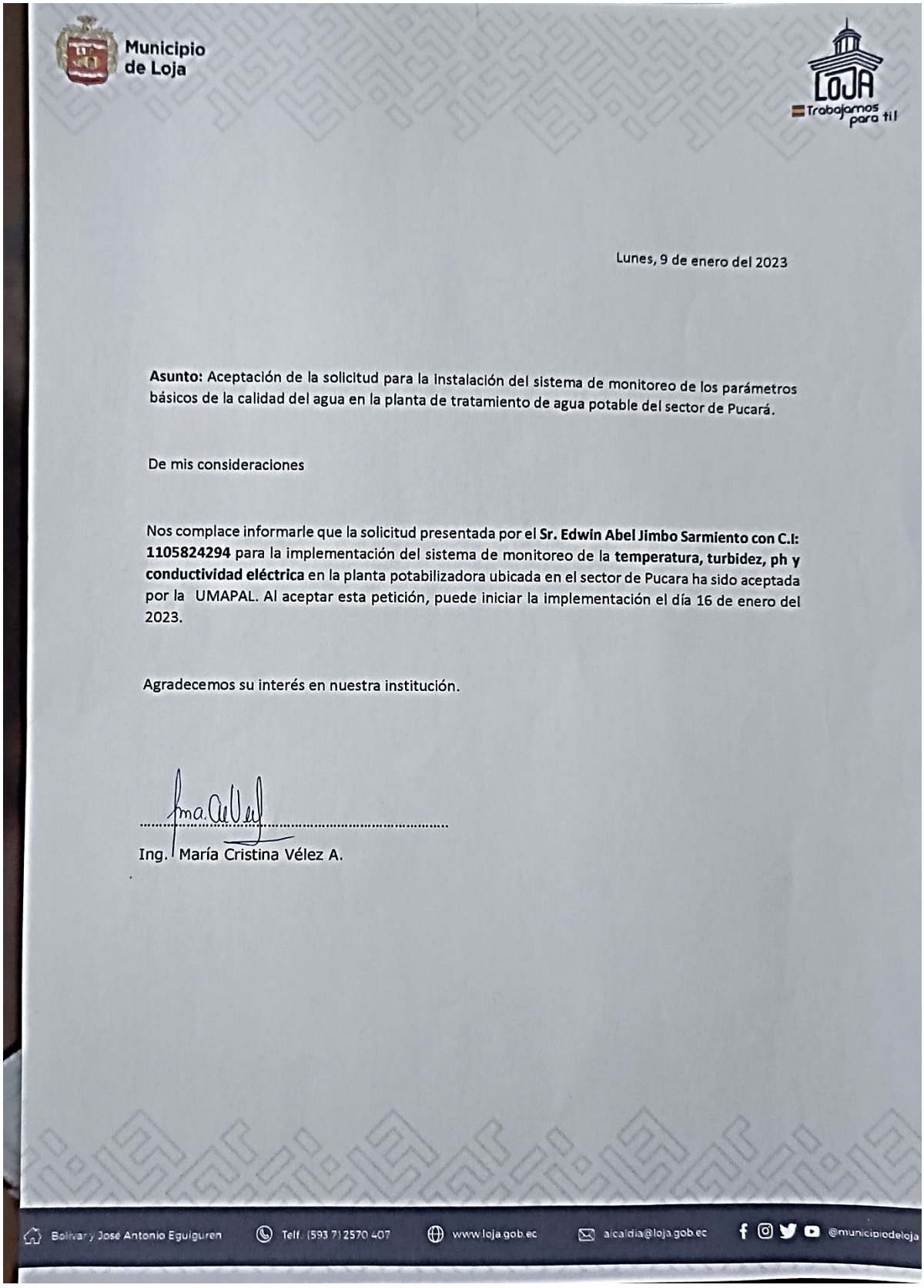
- Diseñar la red inalámbrica de baja potencia basada en IoT con tecnología de acceso radio "LoRa" para el monitoreo en tiempo real de la temperatura, el potencial de hidrógeno (pH), conductividad y turbidez del agua.
- Implementar y evaluar la red IoT en la planta potabilizadora del sector de Pucará.
- Presentar y analizar en tiempo real los datos obtenidos por la red mediante una API de una plataforma IoT.

Esperando su respuesta favorable desde ya le expreso mis sinceros agradecimientos.
Atentamente.



Edwin Abel Jimbo Sarmiento
CI. 1105824294
Correo: edwinjimbosarmiento@gmail.com
Cel: 0962544629

Anexo 13. Aceptación de la solicitud



Municipio de Loja

LOJA Trabajamos para ti!

Lunes, 9 de enero del 2023

Asunto: Aceptación de la solicitud para la instalación del sistema de monitoreo de los parámetros básicos de la calidad del agua en la planta de tratamiento de agua potable del sector de Pucará.

De mis consideraciones

Nos complace informarle que la solicitud presentada por el **Sr. Edwin Abel Jimbo Sarmiento con C.I: 1105824294** para la implementación del sistema de monitoreo de la **temperatura, turbidez, ph y conductividad eléctrica** en la planta potabilizadora ubicada en el sector de Pucara ha sido aceptada por la UMAPAL. Al aceptar esta petición, puede iniciar la implementación el día 16 de enero del 2023.

Agradecemos su interés en nuestra institución.

Ing. María Cristina Vélez A.

Bolívar y José Antonio Eguiguren Telf.: (593 7) 2570 407 www.loja.gob.ec alcaldia@loja.gob.ec @municipiodeloja

Anexo 14. Certificado de la implementación del prototipo

Municipio de Loja

Loja ¡Con honestidad te reconstruiremos!

Loja, 20 de marzo del 2023

CERTIFICACIÓN

Por medio del presente, tengo a bien certificar que el **Sr. Edwin Abel Jimbo Sarmiento** portador de la cédula **Nro. 1105824294**, estudiante de la Carrera de Ingeniería en Telecomunicaciones de la facultad de la Energía, las Industrias y los Recursos Naturales No Renovables de la Universidad Nacional de Loja, realizo la implementación del proyecto denominado: **"DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO BASADO EN IOT PARA EL MONITOREO DE LOS PARÁMETROS BÁSICOS DE LA CALIDAD DEL AGUA EN LA PLANTA POTABILIZADORA DEL SECTOR DE PUCARÁ"**, la cual empezó el 23 de enero y culminó el 15 de marzo del 2023. La implementación se cumplió en un total de 200 horas en horarios de lunes a viernes.

Atentamente

Ing. María Cristina Vélez A

TÉCNICA RESPONSABLE DE LA PLANTA DE PUCARA

Bolívar y José Antonio Eguiguren Telf.: (593 7) 2570 407 www.loja.gob.ec alcaldia@loja.gob.ec @municipiodeloja

Anexo 15. Certificado de la fiel traducción del español-inglés del resumen

Mg. Sc. María Patricia Rodríguez Ludeña

DOCENTE DE LA CARRERA DE PEDAGOGÍA DE LOS IDIOMAS NACIONALES Y EXTRANJEROS DE LA UNL

C E R T I F I C A:

Que el documento aquí expuesto es fiel traducción del idioma español al idioma inglés del resumen del Trabajo de Integración Curricular titulado: "**DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO BASADO EN IOT PARA EL MONITOREO DE LOS PARÁMETROS BÁSICOS DE LA CALIDAD DEL AGUA EN LA PLANTA POTABILIZADORA DEL SECTOR DE PUCARA**" de autoría de EDWIN ABEL JIMBO SARMIENTO con cédula de ciudadanía 1105824294, de la Carrera de Ingeniería en Telecomunicaciones de la Universidad Nacional de Loja.

Lo certifico y autorizo al interesado hacer uso del presente en lo que a sus intereses convenga.

Loja, 11 de julio de 2023

MARIA PATRICIA RODRIGUEZ LUDENA

Mg. Sc. María Patricia Rodríguez Ludeña

DOCENTE DE LA CARRERA DE PEDAGOGÍA DE LOS IDIOMAS NACIONALES Y EXTRANJEROS DE LA UNL