Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

INTEGRACIÓN DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGÍA LORA Y BLUETOOTH

José Carlos Samayoa Santiago
Asesorado por el Ing. Julio Solares

Guatemala, noviembre del 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

INTEGRACIÓN DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGÍA LORA Y BLUETOOTH

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSE CARLOS SAMAYOA SANTIAGO
ASESORADO POR EL ING JULIO SOLARES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO ELECTRÓNICO

GUATEMALA, NOVIEMBRE DEL 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Córdova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Julio César Solares Peñate
EXAMINADOR	Ing. Guillermo Antonio Puente Romero
EXAMINADOR	Ing. Armando Alonso Rivera Carrillo
SECRETARIO	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

INTEGRACIÓN DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGÍA LORA Y BLUETOOTH

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 9 de septiembre de 2019.

José Carlos Samayoa Santiago

Guatemala, 25 de agosto de 2021

Señor
Coordinador del Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC.

Estimado Ingeniero:

Por este medio me permito dar aprobación al trabajo de Graduación titulado **INTEGRACION DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGIA LORA Y BLUETOOTH**, desarrollado por el estudiante **José Carlos Samayoa Santiago**, ya que considero que cumple con los requisitos establecidos.

Por lo tanto, el autor de este trabajo y yo como asesor, nos hacemos responsables del contenido y conclusiones del mismo.

Sin otro en particular, aprovecho la oportunidad para saludarlo.

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Asesor

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 30 de agosto de 2021

Señor director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **INTEGRACION DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGIA LORA Y BLUETOOTH,** desarrollado por el estudiante **José Carlos Samayoa Santiago**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERIA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA

REF. EIME 141. 2021.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; JOSÉ CARLOS SAMAYOA SANTIAGO: INTEGRACIÓN DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGÍA LORA Y BLUETOOTH, procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA DE INGENIERIA MECANICA ELECTRICA
FACULTAD DE INGENIERIA

GUATEMALA, 7 DE SEPTIEMBRE 2,021.

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Decanato
Facultad de Ingeniería
24189101 – 24189102
secretariadecanato@ingenieria.usac.edu.gt

DTG. 595-2021

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **INTEGRACIÓN DE SENSORES Y DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO DE LARGO ALCANCE, BASADO EN TECNOLOGÍA LORA Y BLUETOOTH**, presentado por el estudiante universitario: **José Carlos Samayoa Santiago,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, noviembre de 2021

AACE/cc

ACTO QUE DEDICO A:

Dios	Por su bendición y gracia al permitirme alcanzar los objetivos propuestos.
Mis padres	Carlos Samayoa y Alma de Samayoa, por su amor y apoyo incondicional en todas mis decisiones.
Mi esposa	Nayeli de Samayoa, por su amor y apoyo en mi carrera y entre otras cosas.
Mi hijo	Alessandro Samayoa, por ser mi motivación para seguir adelante.
Mis hermanas	Sara y Ligia Samayoa. Por su paciencia y amor. Por ser una importante influencia en mi carrera, entre otras cosas.
Mis abuelos y familiares	Por ser parte importante y un gran apoyo en mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por ser la casa de estudio que me formo en mi carrera profesional.
Facultad de Ingeniería	Por haberme permitido forjar mi camino profesional y de conocimiento.
Mi asesor	Ing. Julio Solares, por su asesoría en la realización del presente trabajo de graduación.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
A	Amperio
ARM	*Advance RISC machine*
dB	Decibel
dBm	Decibel milivatio
f	Frecuencia
°C	Grados Centígrados
Ghz	Gigahercio
GPIO	*General purpouse in/out pin*
KBPS	Kilobits por segundo
kHz	Kilohercio
Km	Kilómetro
kΩ	Kiloohm
M	Metro
Mhz	Megahercio
mA	Miliamperio
mm	Milímetro
ms	Milisegundo
mW	Microvatios
nA	NanoAmperio
s	Segundo
USB	*Universal serial bus*
W	Vatio
V	Voltio

GLOSARIO

APK Es el tipo de extensión o formato para los archivos que utilizan los dispositivos móviles como un teléfono celular o una tableta para instalar aplicaciones.

Arduino Compañía de *software* y *hardware* libres.

ARM ARM, anteriormente *Advanced RISC Machine,* originalmente *Acorn RISC Machines,* es una arquitectura RISC (Ordenador con Conjunto Reducido de Instrucciones) de 32 bits y, con la llegada de su versión V8-A, también de 64 bits, desarrollada por ARM *Holdings*.

Banda ISM Banda del espectro electromagnético utilizada por industrias y científicos para la investigación, libre de regulaciones comerciales.

Blue Pill Seudónimo de microcontrolador con conectividad LoRa. Tarjeta STM32 es una placa desarrollo llamada *Blue Pill,* es de bajo costo que incorpora el núcleo RISC de 32 bits ARM®Cortex-M3 de alto rendimiento, esta tarjeta es adecuada para iniciar proyectos sobre el microcontrolador STM32.

Bluetooth	Tecnología utilizada por redes inalámbricas de área personal de comunicación, cuenta con su propio protocolo de comunicación.
C++	C++ es un lenguaje de programación diseñado en 1979 por *Bjarne Stroustrup*. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido.
Código C	Lenguaje de programación compilado de propósito general.
Conector	Estándar de conector para tecnología USB de microUSB dimensiones reducidas.
Dirección IP	Identificador, dentro del protocolo de Internet, de un dispositivo como parte de una red.
eTape	El sensor *eTape* es un sensor de estado sólido y continuo (multinivel) para medir agua, líquidos a base de agua no corrosivos y fluido seco (polvo).
Hardware	Son todos los componentes físicos eléctricos, electrónicos, electromecánicos y mecánicos de un sistema informático.

HTTP	Protocolo de transferencia de hipertexto.
IDE	Entorno integrado de desarrollo que se utiliza para programar.
IoT	Internet de las cosas, tecnología que consiste en la comunicación de máquinas, dispositivos y humanos a través de la red.
LoRaWAN	LoRaWAN ™ es una especificación de red de área amplia y baja potencia (LPWAN: *Low Power Wide Area Network*) diseñada para funcionar con "Cosas" alimentadas con baterías de forma inalámbrica en una red regional, nacional o global.
MCU	Un microcontrolador (a veces abreviado como µC, uC o MCU) es una pequeña computadora en un solo circuito integrado que contiene un núcleo de procesador, memoria y periféricos de entrada / salidas programables.
Noise Floor	El *Noise Floor* es una medida de la señal generada por la suma de todas las fuentes de ruido y las señales no deseadas en el sistema de medición, donde el ruido se define como cualquier señal que no sea la señal monitoreada.

P2P	Igual a igual, es un modelo de comunicación descentralizado donde todos los dispositivos pueden actuar como clientes y servidores.
PHP	PHP es un lenguaje de programación de uso general que se adapta especialmente al desarrollo web. Fue creado inicialmente por el programador danés-canadiense Rasmus Lerdorf en 1994. En la actualidad, la implementación de referencia de PHP es producida por *The PHP Group.*
Puerto	Canal lógico de comunicación de un dispositivo de red.
Pull-up	Es una resistencia conectada entre la salida del circuito y el voltaje de alimentación para evitar los valores lógicos indeterminados.
Resistencia RF	Radiofrecuencia.
Servidor web	Dispositivo especializado que presta un servicio de HTTP.
Software	Es el conjunto de componentes lógicos (programas o instrucciones secuenciales) que necesita un sistema informático para realizar tareas.

STM32	STM32 es una serie de circuitos integrados de microcontroladores de *STMicroelectronics*. El chip STM32 se basa en el núcleo RISC ARM de 32 bits de ARM *Holdings*, como los núcleos Cortex-M4F, Cortex-M3, Cortex-M0 + y Cortex-M0.
Switch	Dispositivo concentrador de red que conmuta la comunicación entre varios dispositivos finales e intermedios.
UART	Transmisor-receptor asíncrono universal.
URL	Localizador de recurso uniforme, es un identificador de la ubicación de un recurso dentro de la red.

RESUMEN

El siguiente trabajo de graduación presenta en el primer capítulo la descripción de los sensores a utilizar para obtener la precipitación y la evapotranspiración del agua en el suelo, así como los periféricos de salida, fuente eléctrica de alimentación que se utilizará para el diseño del nodo y el *Gateway*. La descripción incluye las reglas, terminología, procesos y topología del diseño propuesto.

En los capítulos posteriores, se da una descripción del concepto, terminología, componentes y las consideraciones del diseño de enlace de datos punto a punto de largo alcance, así como la definición, origen y descripción de la tecnología LoRa y *bluetooth*, además la forma en que se relaciona con las necesidades de los ingenios azucareros para realizar lecturas más precisas de los sensores para obtener balances hídricos cada vez más reales con el objetivo de optimizar el recurso hídrico.

Por último, se expone un ejemplo de las pruebas de funcionamiento de la integración de sensores y diseño planteado. Se detalla cada paso del diseño para que cualquiera que tenga acceso a este documento pueda replicar el ejemplo que se muestra o adaptarlo a su necesidad realizando las modificaciones que considere necesarias.

OBJETIVOS

General

Automatizar la lectura de datos mediante sensores e internet de las cosas para reducir tiempos y facilitar el acceso a la información sobre la programación de riego de ingenio Magdalena.

Específicos

1. Diseñar un radioenlace de datos de largo alcance basado en tecnología LoRa.

2. Combinar tecnologías de largo y corto alcance basados en estándares existentes.

3. Integrar sensores de diferentes fabricantes capaces de comunicarse bajo un mismo sistema de automatización y control.

4. Mejorar la eficiencia de la programación del riego de las fincas del ingenio Magdalena.

5. Facilitar el acceso a la información en tiempo real para la programación de riego.

INTRODUCCIÓN

La tecnología avanza cada año y las máquinas son cada vez más capaces de realizar tareas automatizadas en servicio de los seres humanos. Los ingenios, y la agroindustria en general, se están enfrentando a un cambio constante debido a la escases de los recursos naturales, esto ha abierto paso a la Agricultura 4.0, la cual incluye aplicaciones y nuevas tecnologías a los distintos procesos agrícolas que han permitido que productores y agricultores mejoren el uso eficiente y responsable de los recursos, aprovechando al máximo lo que la tierra produce y haciéndolo sostenible y amigable con el medioambiente.

Las empresas y fabricantes que ofrecen servicios de sistemas de comunicación inalámbrica y equipos de medición para aplicar la Agricultura 4.0 aumentan con frecuencia, sin embargo, cada empresa o fabricante utilizan protocolos propietarios para la intercomunicación de sus productos, mientas el mercado de dispositivos conectables en red aumenta y los productores de tecnología aplicable a la Agricultura 4.0 crecen en número, los protocolos propietarios no son una opción viable para favorecer la interconexión de los equipos, requisito importante del internet de las cosas. Es necesario el uso de protocolos basados en estándares existentes, considerablemente distribuidos en el mundo y adaptados a los requerimientos de un sistema de comunicación inalámbrica, para que puedan ser integrados por cualquier fabricante que utilice tecnologías de conexión en red estándar.

El propósito de este trabajo de graduación es el diseño de un enlace de datos punto a punto de largo alcance, basado en tecnología LoRa y *bluetooth* integrando los sensores de medición de precipitación y evapotranspiración basado en estándares que permitan la integración de aplicaciones existentes e interfaces de usuario.

Para entender el funcionamiento del diseño presentado en este trabajo es necesario introducir al lector en una base teórica sobre el campo de las tecnologías inalámbricas que existen en la actualidad y sobre las ventajas y desventajas de cada una de ellas, así mismo, es importante que se presente un resumen sobre las descripciones y especificaciones de la tecnología LoRa y los conceptos básicos de un enlace de radiofrecuencia que han sido considerados para la descripción del diseño propuesto.

1. RED DE SENSORES

1.1. Microcontrolador

STM32 es un tipo de microcomputadora de un solo chip, potente, de bajo costo, con gran cantidad de pines de entrada, información oficial completa, fácil de usar y de depuración. Es fácil de ver STM32 en los productos electrónicos integrados modernos. En productos de tecnología electrónica de alta calidad, STM32 representa un gran beneficio económico de la microcomputadora de un solo chip. Debido a sus obvias ventajas, no es imposible utilizar otros tipos de microcomputadoras de un solo chip. Es necesario usar STM32 como la microcomputadora de un solo chip de 32 bits representativa y el mercado determina la dirección de aprendizaje. El rendimiento STM32 es una MCU de 32 bits, pero siempre que el precio de la MCU de 8 bits, la velocidad sea varias veces mayor que la de la MCU de 8 bits, lo más importante es que es más fácil de dominar como un chip de nivel de entrada ARM.

La tarjeta de desarrollo de ST*Microelectronics,* también conocida como STM32 *Blue Pill* LoRaWAN *node*, es una tarjeta de desarrollo para un microcontrolador ARM Cortex M3. Muy similar al Arduino Nano, pero tiene un gran impacto debido que pueden superar fácilmente al Arduino con su CPU de 32 bits y la arquitectura ARM Cortex M3.

Estas tarjetas son extremadamente baratas en comparación con las tarjetas de desarrollo oficiales de Arduino y también el *hardware* es de código abierto. El microcontrolador es el STM32F103C8T6 de STM*icroelectronics*. Además del microcontrolador, la tarjeta de desarrollo también tiene dos osciladores de cristal,

uno es un cristal de 8 MHz y el otro es un cristal de 32 KHz, que se puede usar para impulsar el RTC interno (reloj de tiempo real). Debido a esto, la MCU puede operar en los modos de suspensión profunda, lo que la hace ideal para aplicaciones que funcionan con baterías.

El nombre de los microcontroladores STM32F103C8T6 tiene un significado detrás

- STM» significa el nombre del fabricante ST*Microelectronics*
- 32» significa arquitectura ARM de 32 bits
- F103» indica que la arquitectura ARM Cortex M3
- C »48 pines
- 8» memoria flash de 64 KB
- T» el tipo de paquete es LQFP
- 6» temperatura de funcionamiento -40 °C a + 85 °C

1.1.1. Características

La serie STM32F103C8T6 de densidad media contiene un núcleo RISC ARM Cortex-M3 de 32 bits de alto rendimiento que se ejecuta a 72 MHz, incluidas varias entradas y salidas mejoradas y periféricos conectados a dos buses APB. También proporciona más características de las que se muestran en la tabla. I.

Tabla I. **Características del microcontrolador STM32F103C8T6**

Tipo de microcontrolador	ARM 32 Cortex-M3 CPU
Frecuencia de operación	72 Mhz
Flash/SRAM	128 K Byte Flash, 20 KByte SRAM
Interfaces disponibles	2x SPI, 3x USART, 2x I2C, 1x CAN, 37x I / O *ports*

Continuación de la tabla I.

Conversión de analógico a digital	2x ADC (12-bit / 16-*channel*)
Timers	3 *timers* generales y 1 *timer* avanzado
Tipo de regulador	Regulador de 3,3 v, con una salida máxima de 300 mA
Número de pines	48

Fuente: elaboración propia.

De ahora en adelante al microcontrolador STM32F103C8T6 se le llamará por su seudónimo *Blue Pill* debido que en la red existe mucha documentación que se refiere a ella bajo ese nombre y no bajo su nombre real.

1.2. Sensor de precipitación

> La precipitación se medirá usando un pluviómetro de tipo balancín que se vacía automáticamente y tiene un interruptor de lámina magnética que se cierra momentáneamente cada vez que el medidor mide 0,011" (0,2794 mm) de lluvia y se puede conectar fácilmente a cualquier microcontrolador. Por lo general, se usará una resistencia de *pull-up* (4,7 kohms) para hacer un cambio de estado del pin de entrada a un nivel alto. [1]

[1] CD TECNOLOGIA. *Sensor de precipitacion balancin.* https://cdtechnologia.net/. Consulta: 3 de mayo de 2019.

Figura 1. **Pluviómetro tipo balancín**

Fuente: SPARKFUN. *Pluvímetro*. https://www.sparkfun.com/products/8942. Consulta: 3 de mayo de 2019.

Para colocar el pluviómetro, también se necesita una armadura secundaria. Esto se usa para mantener el pluviómetro alejado de los otros sensores para garantizar que pueda obtener una medición precisa.

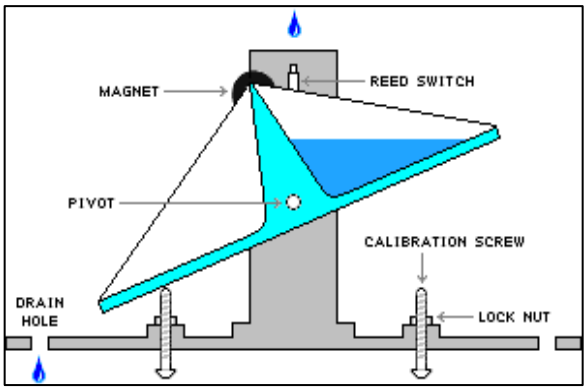
Figura 2. **Armadura secundaria del pluviómetro**

Fuente: ARGENT Data Systems. *Estación meteorológica.* https://www.argentdata.com/catalog/product_info.php?products_id=13. Consulta: 8 de julio de 2019.

El pluviómetro internamente utiliza dos pequeños cubos montados en un punto de apoyo (equilibrado como un balancín). Los cubos pequeños se fabrican con tolerancias ajustadas para garantizar que contengan una cantidad exacta de precipitación, por lo general, 0,011 pulgadas.

El conjunto del cucharón basculante se encuentra debajo del colector de lluvia, que canaliza la precipitación hacia los cubos. A medida que la lluvia llena el pequeño cubo, se desequilibra y se inclina, vaciándose mientras el otro cubo gira en su lugar para la siguiente lectura. La acción de cada evento de volteo activa un pequeño interruptor que activa los circuitos electrónicos para transmitir el conteo, registrando el evento como 0,011 pulgadas de lluvia.

Figura 3. **Funcionamiento interno del pluviómetro tipo balancín**



Fuente: WEATHER INNOVATIONS. *Interno del Pluvímetro.* https://www.weatherinnovations.com/technology.cfm. Consulta: 3 de julio de 2019.

1.3. Sensor de evapotranspiración

Es un sensor de nivel de líquido, un sensor de estado sólido con una salida resistiva que varía con el nivel del fluido. Elimina los voluminosos mecanismos mecánicos e interactúa fácilmente con los sistemas de control electrónico. La envoltura del sensor *eTape* está comprimida por la presión hidrostática del fluido en el que se encuentra inmerso. Esto resulta en un cambio en la resistencia que corresponde a la distancia desde la parte superior del sensor a la superficie del fluido. La salida resistiva del sensor es inversamente proporcional a la altura del líquido: cuanto más bajo es el nivel del líquido, más alta es la resistencia de salida; cuanto mayor sea el nivel de líquido, menor será la resistencia de salida. Dado que el sensor es resistivo, es fácil de leer con un microcontrolador por medio de un ADC.

Figura 4. **Sensor de evapotranspiración**

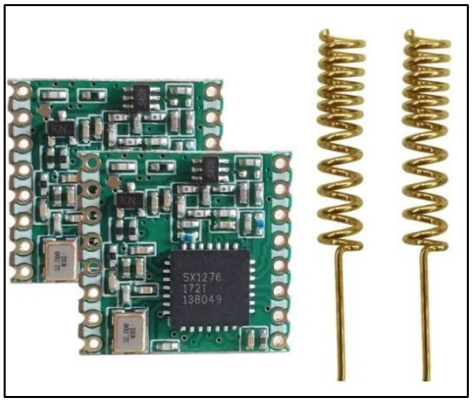
Fuente: IP MARKET. *Evaporímetro To19-TEVAP.SIAP*. https://ipmarket.cl/producto/evaporimetro-t019-. Consulta: 8 de mayo de 2019.

1.4. Módulo LoRa

Los transceptores SX1276/77/78/79 cuentan con el módem de largo alcance LoRa que proporciona una comunicación de espectro ensanchado de largo alcance y una alta inmunidad a las interferencias, a la vez que minimiza el consumo de corriente.

Usando la técnica patentada de modulación LoRa de *Semtech*, SX1276/77/78/79 puede lograr una sensibilidad de más de -148 dBm usando un cristal de bajo costo y una lista de materiales. La alta sensibilidad combinada con el amplificador de potencia integrado de +20 dBm proporciona un presupuesto de enlace líder en la industria que lo hace óptimo para cualquier aplicación que requiera rango o robustez. LoRa también proporciona ventajas significativas tanto en bloqueo como en selectividad sobre las técnicas de modulación convencionales, resolviendo el compromiso de diseño tradicional entre rango, inmunidad a la interferencia y consumo de energía.

Figura 5. **Módulo LoRa SX1276**



Fuente: DWM ZONE. *SX1276 LoRa Module*. https://dwmzone.com/en/lora-wireless-modules. Consulta: 13 de octubre de 2019.

1.4.1. Características

El *chip* SX1276 incorpora el módem de espectro ensanchado LoRa el cual es capaz de proveer una distancia de comunicación significativamente mayor que los sistemas basados en modulación FSK y OOK.

En su velocidad máxima de transmisión la sensibilidad de LoRa es 8 dB mayor que los módems basados en FSK, pero utilizando un cristal de 20 ppm, la sensibilidad del módem LoRa puede ser mejorada hasta en 20 dB comparado con módems basados en FSK. LoRa también proporciona mejoras significativas en selectividad e inmunidad al bloqueo, mejorando aun la fiabilidad de la comunicación. Para una mayor flexibilidad el usuario decide en ancho de banda del espectro ensanchado, el factor de ensanchamiento y el factor de corrección de errores. Otro beneficio de la modulación de espectro ensanchado es que cada factor de ensanchamiento es ortogonal, de forma tal que múltiples señales pueden compartir el mismo canal sin interferirse. En la tabla II, se describen las características del módulo SX1276.

Tabla II. **Características de módulo SX1276**

Presupuesto máximo del enlace	168 dB
Salida RF	+20 dBm - 100 mW
Eficiencia	14 dBm PA de alta eficiencia
Velocidad de bits	hasta 300 kbps programable
Alta sensibilidad	hasta -148 dBm
Corriente de recepción baja	9,9 mA, 200 nA retención de registro
Modulación	FSK, GFSK, MSK, GMSK, LoRa y OOK
Rango dinámico RSSI	127 dB
Sensores incorporados	temperatura e indicador de batería baja

Fuente: elaboración propia.

1.5. Módulo *bluetooth*

El módulo HC-05 es un módulo *bluetooth* SPP (Protocolo de puerto serial) fácil de usar, diseñado para una configuración de conexión serie inalámbrica transparente.

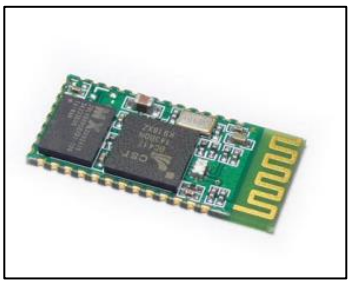
El módulo *bluetooth* del puerto serie es totalmente calificado para *bluetooth* V2.0+EDR (Velocidad de datos mejorada) 3 Mbps modulación con transmisor-receptor de radio completo de 2,4 GHz y banda base. Utiliza CSR *Bluecore* 04-Sistema mono-chip externo *bluetooth* con tecnología CMOS y con AFH (Función de salto de frecuencia). Tiene una huella tan pequeña como 12,7 mm x 27 mm. En la tabla III, se describen las características más importantes del módulo HC-05.

Tabla III. **Características de módulo HC-05**

Sensibilidad típica	-80 dBm
Potencia de transmisión RF	Hasta +4 dBm
Voltaje	1,8 a 3 6 V I/O
Interfaz	UART con velocidad de transmisión programable
Tipo de antena	Antena integrada

Fuente: elaboración propia.

Figura 6. **Módulo HC-05**



Fuente: ELECTRÓNICA ESTUDIO. *Módulo HC-05.* https://www.electronicaestudio.com/. Consulta: 8 de octubre de 2019.

1.6. Pantalla de visualización

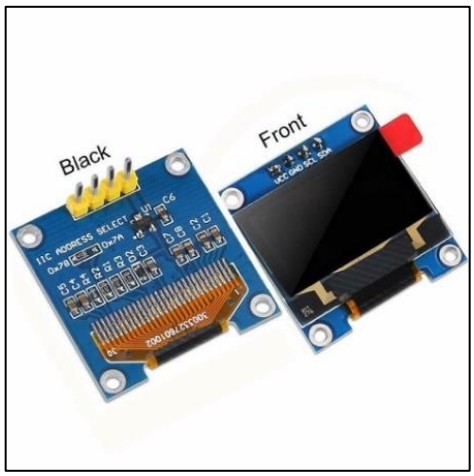
Se trata de un módulo de pantalla OLED de 0,96″ con controlador SSD1306 e interfaz I2C es ideal para usar con Arduino y STM32, compuesto por 128×64 pixeles OLED color blanco funciona sin luz de fondo, por lo que en ambientes obscuros las pantallas OLED son más brillantes comparados a los LCD. En la tabla IV, se describen las características de la pantalla OLED.

Tabla IV. **Características de pantalla OLED**

Tipo de pantalla	Led orgánico
Tensión de funcionamiento	3,3 V/ 5 V
Interfaz	Serie I2 C (CS, RS, SCL, SDA)
Resolución	128 x 64 pixeles
Ángulo de visión	>160°
Controlador	SSD1306
Consumo	0,06 W máximo
Default I2C *address*	0x3 C

Fuente: elaboración propia.

Figura 7. **Pantalla OLED con controlador SSD1306**



Fuente: EBAY. *Controlador SSD1306.* https://www.ebay.com/itm/293570812029?ul_noapp=true Consulta: 8 de octubre de 2019.

1.7. Batería

"El nodo y el *Gateway* utiliza como fuente eléctrica una batería recargable de Iones de litio de 3,7 V, el tipo que se usa a menudo en las cámaras fotográficas"[2].

1.7.1. Características

Son baterías muy ligeras, pero tienen alta capacidad energética y resistencia a la descarga, y casi no tienen efecto memoria. También pasan por muchos ciclos.

[2] MOUSER ELECTRONICS. *Tiny Circuits ASR00050.* https://www.mouser.com.gt/ProductDetail/TinyCircuits/ASR00050?qs=byeeYqUIh0Nbpw7ccioDew==. Consulta: 8 de octubre de 2019.

Estas pueden estar protegidas o desprotegidas. Las protegidas incluyen un circuito que controla cuándo la batería no debe descargarse, corta el voltaje de salida y corta el voltaje de entrada cuando la batería está completamente cargada. En la tabla V, se describen las características de la batería 18 650.

Tabla V. **Características de la batería**

Marca:	*TinyCircuits*
Química de la batería:	*Lithium Ion* (Li-Ion)
Voltaje:	3,7 V
Capacidad:	2 500 mAh
Empaquetado:	*Bulk*
Tamaño de batería:	18 650
Número de celdas:	1

Fuente: elaboración propia.

Figura 8. **Batería recargable**

Fuente: MOUSER ELECTRONICS. *Tiny Circuits ASR00050.* https://www.mouser.com.gt/ProductDetail/TinyCircuits/ASR00050?qs=byeeYqUIh0Nbpw7ccioDew==. Consulta: 8 de octubre de 2019.

1.8. Módulo de carga

Este módulo es perfecto para la carga de baterías LiPo o Li-ion de una sola celda de 3,7 V 1 Ah, cuenta con circuito de protección que le facilita la carga con pilas como 16 550 o 18 650 que no cuentan con dicho circuito de protección.

Basado en el chip TP4056 y el chip de protección de batería DW01 este módulo ofrecerá una corriente de carga de 1A, la luz roja indicará que se está cargando, mientras que el verde indicará que está completamente cargado (cortará la carga cuando haya terminado).

> El puerto de entrada es micro USB hembra, puede usar directamente el cargador de un smartphone o se puede conectar un panel solar para la entrada a la batería de litio, y aún retener las uniones de soldadura de cableado de voltaje de entrada[3].

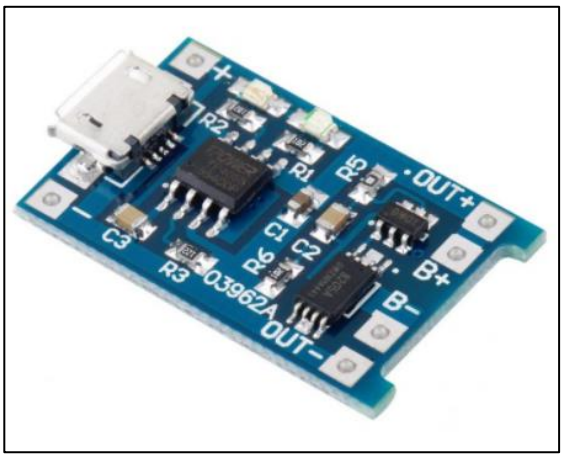
Tabla VI. **Características del módulo de carga**

Voltaje de entrada:	5 V
Corriente de carga máxima:	1 000 mA
Voltaje de corte de carga:	4,2 V ± 1%
Voltaje de protección de sobre carga de la batería:	2,5 V
Corriente de protección contra sobre corriente de la batería:	2,5 V
Interfaz de entrada:	Micro USB o panel solar
Dimensión:	2,6 x 1,7 cm

Fuente: elaboración propia.

[3] SANDOROBOTICS. *Módulo de carga con circuito de protección.* https://sandorobotics.com/producto/hr0140-1/. Consulta: 9 de julio de 2019.

Figura 9. **Módulo de carga TP4056**



Fuente: SANDOROBOTICS. *Módulo de carga con circuito de protección.* https://sandorobotics.com/producto/hr0140-1/. Consulta: 9 de julio de 2019.

1.9. Panel solar

> Los paneles solares son dispositivos que permiten el aprovechamiento de los rayos del sol para ser convertidos en energía eléctrica y así son empleados para el funcionamiento de diferentes equipos eléctricos, también son fuentes de energía renovable.
>
> Mediante placas reciben los rayos solares, no producen contaminación y son de fácil mantenimiento.[4]

El panel utilizado para alimentar la batería del nodo y *Gateway* tiene las siguientes características.

[4] ENERGYMASTER. *¿Para qué sirven los paneles solares?* https://energymaster.co/sirven-los-paneles-solares/. Consulta: 9 de julio de 2019.

Características:

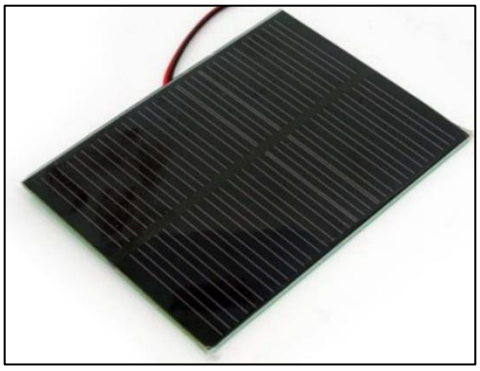
- Hecho de material monocristalino (monocristalino) que realiza una alta eficiencia de transformación de energía solar.
- Pequeño pero capaz de proporcionar suficiente energía para sus proyectos.
- Una fina superficie de resina que lo hace impermeable y un respaldo resistente adecuado para ambientes al aire libre.[5]

Tabla VII. **Características del panel solar de un 1W**

Voltaje típico:	5,5 V
Corriente típica:	170 mA
Voltaje de circuito abierto:	8,2 V
Voltaje de carga máxima:	6,4 V
Dimensión:	100x80x2,5 mm
Eficiencia	15,5 %

Fuente: elaboración propia.

Figura 10. **Panel solar de un 1W**



Fuente: SEEED. *Panel solar.* https://www.seeedstudio.com/1W-Solar-Panel-80X100.html

Consulta: 12 de octubre 2019.

[5] SEEED. *Small solar panel.* https://www.seeedstudio.com/1W-Solar-Panel-80X100.html. Consulta: 9 de julio de 2019.

2. DESCRIPCIÓN DE LA TECNOLOGÍA LORA & *BLUETOOTH*

2.1. IOT (Internet de las cosas)

En las últimas décadas, se han creado y desarrollado varios equipos (microcontrolador, actuador y sensores) con gran rapidez. Estos equipos están conectados a los usuarios finales a través de varias plataformas (móvil, ordenador). Esta conduce al concepto del Internet de las Cosas (IoT). “La IoT puede expresarse como una conectividad inteligente a través de internet en la que cada dispositivo intercambia información entre sí”[6].

El Internet de las cosas (IoT) es un ámbito tecnológico emergente que está creciendo rápidamente. Hoy en día, los sensores están entrando en el mercado del IoT y se espera que estos dispositivos inteligentes funcionen durante meses y años, pero por lo general se alimentan con baterías.

> La comunicación inalámbrica entre los sistemas de sensores distribuidos y las estaciones centrales puede resultar muy costosa en términos de consumo de energía, especialmente en redes que tienen que recorrer varios kilómetros. Por lo tanto, el diseño de la comunicación inalámbrica para un nodo moderno de la IoT merece una atención especial.[7]

---

[6] DAZA, Lennyn. *Beyond the internet of things: everything interconnected: technology, communications and computing*. https://www.semanticscholar.org/paper/Beyond-the-internet-of-things%3A-everything-and-%5Bbook-Daza-Misra/a6673b6613a3ba9d342232be7d1830317a070471. Consulta: 12 de octubre de 2019.

[7] MAGNO, Michele*. WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication*. https://www.researchgate.net/publication/316948751_WULoRa_An_energy_efficient_IoT_end-node_for_energy_harvesting_and_heterogeneous_communication. Consulta: 12 de octubre de 2019.

Beyond the internet of things: everything interconnected: technology, communications and computing [book review]

En el pasado, *Wi-Fi* y celular son los más populares para la implementación de IoT, *Wi-Fi* es de alta velocidad de datos, libre de costo, pero con un corto alcance de comunicación y un alto consumo de energía. Por lo tanto, los dispositivos IoT con *Wi-Fi* se utilizan en aplicaciones con un entorno de interiores como el hogar inteligente, la oficina inteligente. Mientras tanto, celular es de alta velocidad de datos, comunicación de largo alcance, pero de consumo de energía media y tiene cuotas mensuales del proveedor de telefonía celular.

Hoy en día, LPWAN (Red de área amplia de baja potencia) es una tecnología novedosa con las características de comunicación de largo alcance, baja velocidad de datos y bajo consumo de energía. “La tecnología LoRa es un estándar abierto desarrollado por LoRa *Alliance*. Utiliza espectro ISM sin licencia por debajo de 1 GHz, por lo que el tiempo de emisión es gratuito”[8].

2.2. Comparación entre comunicaciones inalámbricas

Al momento de transmitir datos, se han utilizado muchas tecnologías inalámbricas, como 3G / 4G, *ZigBee*, *bluetooth* o LoRa. Cada una de ellas tiene una serie de ventajas y desventajas que las hacen más o menos efectivos. Por tanto, la elección de la tecnología debe depender de los requisitos de una aplicación en particular, es decir, se debe encontrar un compromiso entre el precio, el consumo de energía y el ancho de banda que puede proporcionar.

[8] VATCHARATIANSAKUL, Nuttakit; TUWANUT, Panwit; PORNAVALAI, hotipat. *Experimental performance evaluation of LoRaWAN: A case study in Bangkok.* https://www.researchgate.net/publication/319591516_Experimental_performance_evaluation_of_LoRaWAN_A_case_study_in_Bangkok. Consulta: 12 de octubre de 2019.

A continuación, se presenta brevemente las diferentes tecnologías involucradas a evaluar.

2.2.1. Wi Fi

Aunque *bluetooth* y *ZigBee* son tecnologías de sensores inalámbricos de baja potencia y baja complejidad, tienen algunas limitaciones, como baja velocidad de datos, corto alcance y menor penetración a través de obstáculos. Con el avance de las tecnologías inalámbricas y de sistema en chip (SoC) en *fields*, se han desarrollado una serie de SoCs de sensores inalámbricos basados en *Wi-Fi* para aplicaciones de sensores inalámbricos de baja potencia"[9].

> *Wi-Fi* es una tecnología de red de área local (LAN) inalámbrica basada en el estándar IEEE 802.11. El estándar IEEE 802.11 proporciona un conjunto de especiaciones para el control de acceso a los medios (MAC) y la capa física (PHY) para la implementación de WLAN inalámbrica en la banda de frecuencias de 900 MHz, 2,4 GHz, 3,6 GHz, 5 GHz y 60 GHz.[10]

2.2.2. 3G/4G

La razón principal de la evolución de toda la red IP de 3G a 4G es para formar la misma plataforma para todas las redes preexistentes, de modo que satisface las necesidades de los usuarios en lo que se refiere a la mejora que se espera de ella. Además de todas las instalaciones 3G, la transmisión de datos se cree que sube por las nubes con velocidades que oscilan entre 100 MBPs a 1 GBPS debido a su rápida velocidad, es etiquetado como una red de conectar en

---

[9] NOREEN, Umber; BOUNCEUR, Ahcène; CLAVIER, Laurent. *A study of LoRa low power and wide area network technology.* https://ieeexplore.ieee.org/document/8075570. Consulta: 12 de octubre de 2019.

[10] KLEINROCK, L.; TOBAGI, F. *Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics*. https://scirp.org/reference/referencespapers.aspx?referenceid=807597. Consulta: 12 de octubre de 2019.

cualquier momento, en cualquier lugar, de cualquier manera. 4G lo hará proporcionan una itinerantica global muy fluida.

Actualmente, el *Wi-Fi* se utiliza en casi cualquier lugar y proporciona una conexión a internet de banda ancha de alta velocidad. Una conexión *Wi-Fi* ofrece eficiencia cuando se utilizan las últimas aplicaciones, sin embargo, la conexión es típicamente más lenta que 3G o 4G.

La principal diferencia entre 3G y 4G es la metodología de acceso, la velocidad de transferencia de datos, la terminología de transmisión y la seguridad. En cualquier momento, en cualquier lugar el usuario móvil puede acceder a los datos multimedia como vídeos y llamadas de voz con seguridad.

2.2.3. *Bluetooth*

Bluetooth es una tecnología inalámbrica de red de área personal (WPAN) basada en el estándar IEEE 802.15.1. Fue lanzado en 1994 por Ericsson. Utiliza una banda de frecuencia de 2,4 GHz con licencia libre y una banda de frecuencia de canal de hasta 1 MHz. *Bluetooth* adopta la técnica de transmisión de espectro ensanchado de salto de frecuencia (FHSS) y ofrece una velocidad máxima de transmisión de datos de hasta 1 Mb/s. El *bluetooth* convencional puede conectar ocho nodos entre sí con siete nodos esclavos y un nodo maestro.

Las redes basadas en *bluetooth* pueden desplegarse de punto a punto de forma maestro-esclavo. El consumo de energía de cualquier tipo de red de comunicación depende en gran medida de la distancia entre los nodos transmisor y receptor, de la potencia preferida que retendrá la señal y, lo que es más importante, del tipo de datos que se intercambien. Las redes basadas en

bluetooth son capaces de intercambiar más o menos todo tipo de datos como multimedia, texto, entre otros.

Otra extensión de *bluetooth* fue propuesta como *bluetooth* 4.0 también conocido como *bluetooth* de baja energía después de *bluetooth* 1.0, 2.0 y 3.0. Fue diseñado como una solución de bajo consumo alternativa al clásico *bluetooth. bluetooth* y BLE se utilizan para diferentes propósitos. El *bluetooth* convencional puede manejar casi toda la variedad de datos, pero consume más energía y dinero.

2.2.4. ZigBee

> El estándar IEEE 802.15.4 comúnmente conocido como ZigBee es la opción más popular en redes de área personal inalámbricas de baja velocidad (LR-WPAN) y WSN. El estándar IEEE 802.15.4 solo tiene definidas las características de la capa física (PHY) y la capa de Control de Acceso Medio (MAC). Estas características han sido adoptadas por ZigBee.[11]

Así también *ZigBee* tiene definidas las especificaciones para la capa de red y la capa de aplicación. En las redes inalámbricas, la capa MAC permite el acceso eficiente basándose en los requisitos de la aplicación, las redes basadas en *ZigBee* pueden ser centralizadas y/o descentralizadas.

2.2.5. LoRa

LoRa es una tecnología inalámbrica al igual que Wi-Fi, *Bluetooth*, LTE, *SigFox o Zigbee*. LoRa utiliza un tipo de modulación en radiofrecuencia, como la AM o la FM o el PSK; pero patentado por *Semtech* una importante empresa

[11] NOREEN, Umber; BOUNCEUR, Ahcène; CLAVIER, Laurent; KACIMI, Rahim. *Performance evaluation of IEEE 802.15.4 PHY with impulsive network interference in cupcarbon simulator*. https://ieeexplore.ieee.org/abstract/document/7746102. Consulta: 12 de octubre de 2019.

fabricante de chips de radio. Esta tecnología de modulación se llama *Chirp Spread Spectrum*, o CSS, y se usa en comunicaciones militares y espaciales desde hace décadas. La gran ventaja de la misma es que puede lograr comunicaciones a largas distancias (típicamente kilómetros) y tiene gran solidez frente a las interferencias.

2.3. Comparación de tecnologías

En la tabla VIII, se describen las comparaciones entre las tecnologías inalámbricas mencionadas en esta sección.

Tabla VIII. **Comparación de tecnologías**

Tecnología	Comunicación Inalámbrica	Especificaciones	Ventajas	Desventajas
Bluetooth	Corto Alcance	Rango: 10 m Potencia Tx: 2.5mW	Es muy fácil crear una red inalámbrica entre varios dispositivos para poder sincronizar e intercambiar información.	El reducido alcance por parte del protocolo para intercambiar información se debe a la baja potencia que maneja.
WIFI	Corto Alcance	Rango: ~50 m Potencia Tx: 80mW	11- 54 Mbps Una vez configuradas, las redes Wi-Fi permiten el acceso de múltiples ordenadores sin ningún problema ni gasto en infraestructura.	Pico de corriente: 30 mA Distancia limitada para la recepción de la señal.
3G/4G	Celular	Rango: 5000 m Potencia Tx: 500mW	Tiene largo alcance de la comunicación y su cobertura casi ubicua.	Las frecuencias licenciadas conllevan costes de operación, las altas velocidades de datos conllevan un significante consumo de energía.
ZigBee	LR-WPAN	Rango:10-100m Potencia Tx:100mW	Ideal para conexiones punto a punto y punto a multipunto.	Zigbee trabaja de manera que no puede ser compatible con Bluetooth en todos sus aspectos porque no llegan a tener las mismas tasas de transferencia, ni la misma capacidad de soporte para nodos.
LoRa	LPWAN	Rango: 2km-5km (área urbana) 5km-15km (área rural) >15km (Línea Vista) Potencia Tx: 20mW	Alta tolerancia a las interferencias. Conexión punto a punto.	No ofrece comunicaciones encriptadas.

Fuente: elaboracion propia.

2.3.1. Descripción de la tecnología LoRa

LoRa es un acrónimo de largo alcance y es una tecnología inalámbrica en la que un emisor de baja potencia transmite pequeños paquetes de datos (0,3 kbps a 5,5 kbps) a un receptor a larga distancia.

LPWAN está en rápido crecimiento en la industria de la comunicación. Entre las tecnologías de baja potencia y largo alcance recientemente introducidas, el fabricante de semiconductores *Semtech* ha introducido una amplia utilización de tecnologías avanzadas de espectro ensanchado con su línea de productos LoRaTM de largo alcance.

> En comparación con las técnicas de modulación tradicionales, la técnica de modulación de espectro ensanchado implícita en LoRa garantiza un mayor presupuesto de enlace, así como una mayor inmunidad a las interferencias de la red. LoRa utiliza una banda más ancha, generalmente de 125 kHz o más, para transmitir la señal. LoRa permite el uso de BW escalable de 125 kHz, 250 kHz o 500 kHz.[12]

2.3.2. Semtech

La tecnología inalámbrica LoRa fue desarrollada por una nueva empresa francesa, Cycleo, que desarrolló la tecnología de modulación LoRa, En 2012, *Semtech Corporation* (NASDAQ: SMTC) adquirió Cycleo. La parte de radio y modulación LoRa está patentada y su fuente está cerrada. *Semtech* ha licenciado su propiedad intelectual (ip) LoRa a otros fabricantes de chips, como HopeRF, Microchip, Dorji, entre otros.

---

[12] REYNDERS, Brecht; MEERT, Wannes; POLLIN, Sofie. *Range and coexistence analysis of long-range unlicensed communication*. https://ieeexplore.ieee.org/document/7500415. Consulta: 14 de octubre de 2019.

2.3.3. Rango de LoRa

El rango entre el nodo y el *Gateway* LoRa depende del entorno en el que opera el equipo. La cobertura interior depende en gran medida del tipo de material de construcción utilizado.

Tabla IX. **Rango de LoRa**

Entorno	Rango (km)
Áreas urbanas (pueblos y ciudades)	2-5
Áreas rurales	15
Línea vista directa	>15

Fuente: elaboración propia.

2.4. Casos de uso de la tecnología LoRa

En la siguiente sección se describen las aplicaciones más comunes de la tecnología LoRa.

2.4.1. Utilidades inteligentes

Las aplicaciones sectoriales de los sensores LoRa y la explotación de la información generada mediante redes LoRa son muy diversas: hidrología, agricultura, infraestructuras, control medioambiental, sensores en edificios entre otros. Algunos ejemplos son:

- Realizar investigaciones ambientales transdisciplinarias dentro y en las interfaces de los sistemas de agua urbanos.

- Monitoreo de combustible (monitoreo de niveles de combustible en tanques de combustible para calentar casas).

- Servicios de monitorización en tiempo real de los sensores conectados a la red LoRa (también disponible 2G/3G, 4G) en su *cloud* para múltiples aplicaciones: ciclo del agua, control medio ambiente, calidad ambiental interior, agricultura inteligente.

> LoRa aprovecha la red basada en LoRaWAN de Tata *Communications* para monitorear las condiciones del suelo, incluyendo la humedad y el pH, para tomar decisiones de riego más inteligentes, mejorando el rendimiento de los cultivos y reduciendo el consumo de agua y electricidad hasta en un 26 %.[13]

2.4.2. Eficiencia

- Gestión de activos (por ejemplo, seguimiento de contenedores, paletas, entre otros).

- Gestión de flotas (por ejemplo, seguimiento de automóviles, furgonetas, camiones, entre otros).

2.4.3. Agricultura

- Los sensores habilitados para LoRa detectan el celo del ganado, impulsan una mejor nutrición y predicen la aparición de enfermedades para ayudar a los ganaderos a controlar mejor su rebaño.

- Monitoreo del bienestar animal.

---

[13] SEMTECH. *LORA APPLICATIONS*. https://www.semtech.com. Consulta: 14 de octubre de 2019.

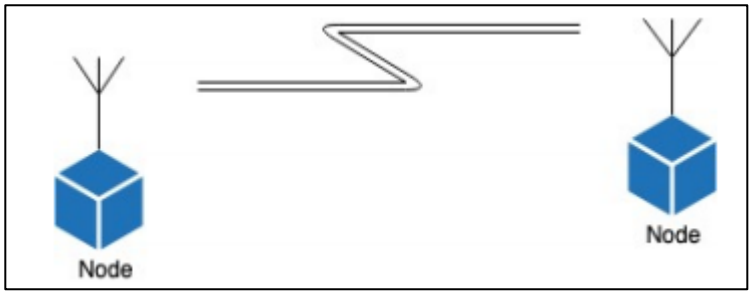
- Monitoreo de las condiciones de crecimiento de la planta.

2.5. *Peer to Peer* (P2P)

Algunas redes están configuradas para enviar datos a través de un servidor central. Por ejemplo, si hay una red LoRa que contenga 2 dispositivo y un *Gateway* que sea compatible para el enlace *Downlink*. Uno de estos dispositivos necesita datos del otro. Para que un nodo reciba datos de otro, los datos deben ser transmitidos primero al *Gateway* y transmitidos desde la *Gateway* para llegar al nodo final.

Tal información que es transmitida y recibida por los nodos finales se puede hacer sin un servidor central, estas implementaciones se llaman comunicación *peer-to-peer* o enlace punto a punto. Para que la comunicación entre pares funcione, es necesario establecer un vínculo entre los nodos. La forma en que se establece este enlace depende de la tecnología que se implemente.

Figura 11. **Representación de un enlace de comunicación P2P**



Fuente: elaboración propia, empleando Paint.

2.6. Nodo LoRa

Un nodo de LoRa consta de 2 partes:

- Un módulo de radio con antena.
- Un microprocesador para procesar, por ejemplo, los datos del sensor.
- Los nodos a menudo funcionan con baterías.
- Un nodo LoRa tiene un transceptor inalámbrico. Si este dispositivo también tiene sensores, este dispositivo actúa como un sensor remoto.

Figura 12. **Representación de un nodo LoRa**

Fuente: PBS. *Nodo LoRa*. https://pbs.twimg.com. Consulta: 2 de septiembre de 2019.

De ahora en adelante al nodo se le llamara también emisor.

2.7. *Gateway* de un solo canal

Un *Gateway* de un solo canal consta de 2 partes:

- Un módulo de radio con antena.
- Un microprocesador para procesar los datos.
- Un *Gateway* funcionan con corriente o batería y están conectadas a internet o algún servidor local.

Figura 13. **Representación de un *Gateway* de un solo canal**

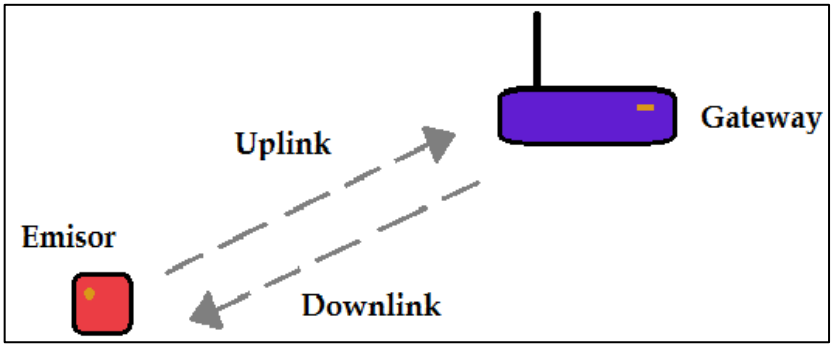
Fuente: *THE THINGS NETWORK. Gateway de un solo canal.* https://www.thethingsnetwork.org. Consulta: 2 de septiembre de 2019.

De ahora en adelante al *Gateway* de un solo canal se le llamará *Gateway* o receptor.

2.8. *Upload & downlink*

- Cuando un nodo transmite datos al *Gateway*, se denomina *uplink*.
- Cuando el *Gateway* transmite datos al nodo, se denomina *downlink.*

Figura 14. **Representación gráfica de *uplink & downlink***



Fuente: elaboración propia, empleando Paint.

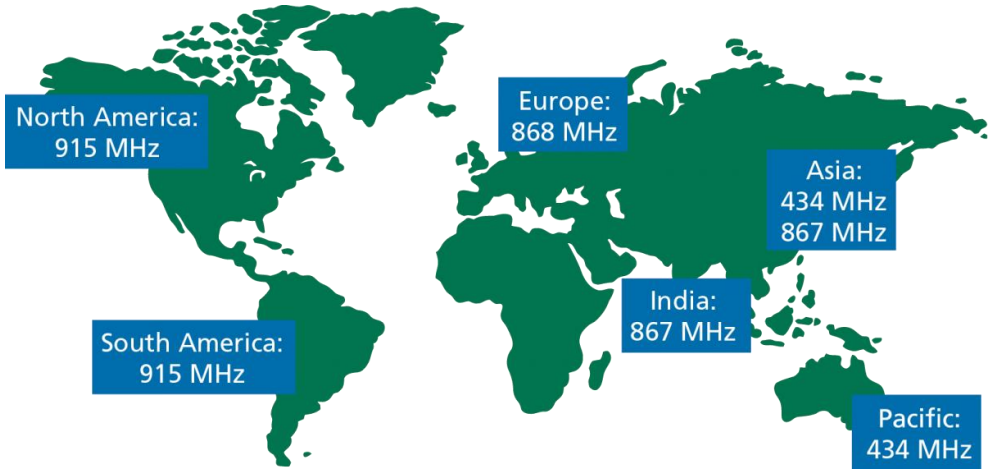
2.9. Reglas y regulaciones

Existen algunas reglas y regulaciones en LoRa y LoRaWAN, y se debe tener en cuenta estas reglas y regulaciones cuando se usa esta tecnología para llevar a cabo proyectos. LoRa opera en las bandas sin licencia industriales, científicas y médicas (ISM) respectivas de cada región. En la siguiente sección se describe los aspectos más importantes de las regulaciones de la tecnología.

2.9.1. Banda ISM

LoRa opera en las bandas de radio ISM (industriales, científicas y médicas) sin licencia que están disponibles en todo el mundo.

Figura 15. **Bandas de radio ISM disponibles en el mundo**



Fuente: PDA CONTROLES. *ISM*. http://pdacontroles.com. Consulta: 3 de septiembre de 2019.

- En los Estados Unidos, LoRaWAN y LoRa opera en la banda de frecuencia de 902-928 MHz.

- Los dispositivos como hornos microondas, equipos médicos o monitores para bebés usan la banda ISM.

2.9.2. Ventajas de banda ISM

- Cualquier persona puede usar estas frecuencias.
- No se requiere tarifa de licencia.

2.9.3. Desventajas de banda ISM

- Baja velocidad de datos.
- Mucha interferencia porque cualquiera puede usar estas frecuencias.

2.9.4. ETSI & FCC

Debido a que la banda ISM puede ser utilizada por todos, debe haber algunas reglas establecidas, de lo contrario esta banda quedará inutilizable. Piense en las muchas interferencias de señal.

- Existen varias organizaciones internacionales que gestionan el espectro de radio para garantizar una coexistencia segura entre todas las diferentes tecnologías de radio.

- En Europa, el Instituto Europeo de Normas de Telecomunicaciones (ETSI) crea normas que son utilizadas por las autoridades reguladoras de cada país. https://www.etsi.org/.

- En los Estados Unidos, la Comisión Federal de Comunicaciones (FCC) crea estos estándares.

- Todos los demás países están utilizando los conjuntos estándar de ETSI FCC. Excepto Japón, tienen el Centro de Ingeniería de Telecomunicaciones (TELEC) y Corea del Sur, tienen la Comisión de Comunicaciones de Corea (KCC).

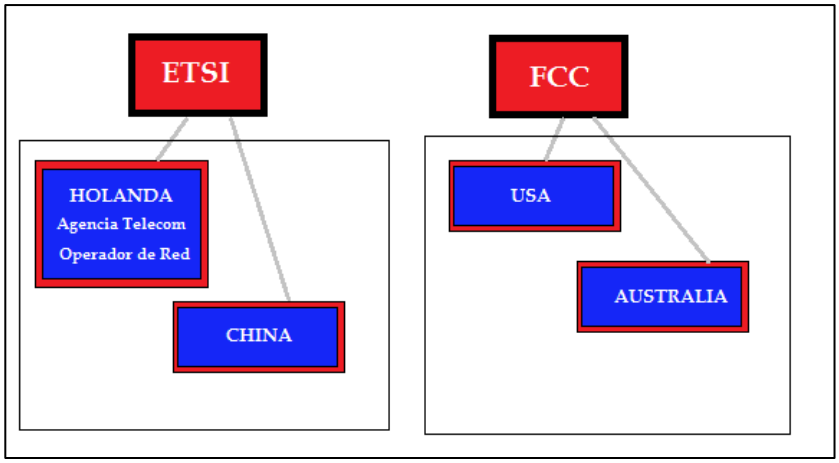
- Por ejemplo, en Holanda, su autoridad reguladora de telecomunicaciones se llama Agencia de Telecomunicaciones (en holandés: *Agentschap* Telecom), que forma parte del Ministerio de Asuntos Económicos y Clima (en holandés: *Ministerie van Economische Zaken en Klimaat*).

- En Guatemala no existe regulación de la tecnología LoRa, por lo tanto, el ciclo de trabajo y el tiempo en el aire no es restringido por la Superintendencia de Telecomunicaciones de Guatemala.

2.9.5. Organizaciones y autoridades reguladoras

Para definir los planes de canales LoRa en cada país o región, la LoRa Alliance ha formado un grupo de trabajo que crea, gestiona y mantiene las especificaciones de los parámetros regionales de LoRa y LoRaWAN. Debido a estas diferentes entidades reguladoras regionales, no es posible proporcionar un plan de canales único para todo el mundo. No obstante, el grupo de trabajo sobre parámetros regionales de LoRa y LoRaWAN ha combinado varios países en diversos planes regionales para reducir el número de regiones de LoRa y LoRaWAN. En la figura 16, se describen las organizaciones internacionales que gestionan el espectro de radio.

Figura 16. **ETSI & FCC**



Fuente: elaboración propia, empleando Paint.

2.9.6. Ciclo de trabajo

El ciclo de trabajo es la proporción de tiempo durante el cual un sistema es operado. El ciclo de trabajo puede expresarse como una relación o un porcentaje. Como ya se ha mencionado anteriormente, en Europa hay un ciclo de trabajo de 0,1 % y 1,0 % por día dependiendo del canal.

Cuando se usan las frecuencias de banda ISM (863 MHz - 870 MHz), los usuarios deben cumplir con las siguientes reglas:

- Para el *uplink,* la potencia de transmisión máxima está limitada a 25 mW (14 dBm). Para el *downlink* (para 869,525 MHz), la potencia de transmisión máxima está limitada a 0,5 W (27 dBm)

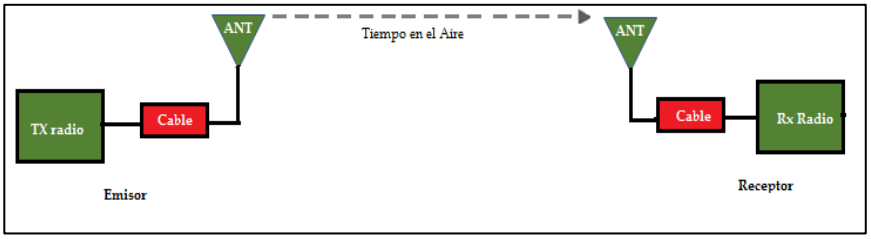
2.9.7. Tiempo en el aire

- Cuando se envía una señal desde un emisor, lleva cierto tiempo antes de que un *Gateway* reciba esta señal. Este tiempo se llama tiempo en el aire (ToA).

- El ciclo de trabajo es la proporción de tiempo durante el cual se opera un componente, dispositivo o sistema. El ciclo de trabajo se puede expresar como una relación o como un porcentaje.

- Por ejemplo: si nuestro ciclo de trabajo es del 1 % entonces el tiempo en el aire es igual a 530 ms, entonces tenemos que esperar 99 x 530 ms = 52,47 s antes de enviar un nuevo mensaje.

Figura 17. **Tiempo en el aire**



Fuente: elaboración propia, empleando Paint.

2.10. Bluetooth

Bluetooth es una especificación tecnológica para redes inalámbricas que permite la transmisión de voz y datos entre distintos dispositivos mediante una radiofrecuencia segura (2,4 GHz). Esta tecnología, por lo tanto, permite las comunicaciones sin cables ni conectores y la posibilidad de crear redes inalámbricas domésticas para sincronizar y compartir la información que se encuentra almacenada en diversos equipos.

En la siguiente sección se describe los datos más relevantes de la tecnología *bluetooth*.

2.10.1. Historia

> El nombre *Bluetooth* viene de Harald Blåtand (en inglés Harald Bluetooth) un rey noruego y danés, quien unificó tribus danesas, suecas y noruegas, y las convirtió al cristianismo. Esto serviría de analogía con *Bluetooth* principalmente porque permitiría la unificación de teléfonos móviles y computadores. Hoy en día siendo usado muchos más dispositivos.[14]

[14] GARIN, Dante; HAZARD, Mario. *Bluetooth*. http://profesores.elo.utfsm.cl/. Consulta: 2 de noviembre de 2019.

La historia de *bluetooth* se remonta a mediados de la década del 90′. Se inventó en 1994, aunque no fue hasta 1998 cuando empezó a comercializarse. Fue creado por ingenieros suecos de Ericsson, que buscaban una solución a la maraña de cables que entonces rodeaban los ordenadores. Ese proyecto era MCLink.

> Con el paso del tiempo, los grandes de la tecnología mostraron interés por el producto y formaron una SIG (*Special Interest Group),* eso en el campo de la tecnología corresponde a una especie de grupo de trabajo conformado por diferentes empresas, quienes aportan capital monetario y humano. Entre esos grandes está: Apple, Ericsson, Intel, Lenovo, Microsoft, Motorola, Nokia, Nordic Semiconductor y Toshiba. (Hoy hay más de 14 000 empresas que lo conforman).[15]

En 1998 *bluetooth* vio la luz, y se pensó en una posible competencia con Wi-Fi, lo cual era equivocado, ya que satisfacen necesidades completamente distintas: el primero está enfocado a conexión entre dispositivos y bajo consumo, mientras que el segundo podría definirse de manera simple como una Ethernet inalámbrica.

2.10.2. *Rango* bluetooth

Los dispositivos *bluetooth* cuentan con dos piezas fundamentales para su funcionamiento. El primero es un dispositivo de radio que se encarga de transmitir la señal. En segundo lugar, un CPU que se encarga de procesar las señales digitales. Los dispositivos *bluetooth* se clasifican en cuatro clases de acuerdo a su capacidad:

[15] WAYER, Fayer. *La historia del nacimiento de Bluetooth*. https://www.fayerwayer.com/2011/09/la-historia-del-nacimiento-de-bluetooth/. Consulta: 2 de noviembre de 2019.

Tabla X. **Clases de *bluetooth***

Clase	Potencia máx. permitida (mW)	Potencia máx. permitida (dBm)	Rango aproximado
Clase 1	100 mW	20 dBm	100 metros
Clase 2	2,5 mW	4 dBm	20 metros
Clase 3	1 mW	0 dBm	1 metro

Fuente: elaboración propia.

2.10.3. Velocidad de datos

Los dispositivos con *bluetooth* también pueden clasificarse según su capacidad de canal. En sus inicios, la tecnología *bluetooth* podía transmitir datos a una velocidad de 720 kbs, una capacidad increíble para la década de los noventa pero que hoy parece muy limitada. Después de más de dos décadas de mejoras, los diferentes tipos de *bluetooth* han llegado a contar con velocidades de hasta 50 Mbs. Además, el rango de conexión es otro de los aspectos que ha mejorado mucho. *Bluetooth* ha pasado de funcionar en distancias menores a un metro a los más de 100 metros que pueden alcanzar hoy en día. En la tabla XI, se describen la velocidad de datos de las versiones de *bluetooth.*

Tabla XI. **Velocidad de datos**

Versión	Ancho de banda (Mbit/s)
Versión 1,2	1
Versión 2,0 + ERD	3
UWB *bluetooth*	54-480

Fuente: elaboración propia.

2.10.4. Casos de uso de la tecnología

Las posibilidades que ofrece la tecnología *bluetooth* son numerosas e interesantes, en especial gracias a su fácil transmisión.

Auriculares inalámbricos para el móvil, teclados, ratones y un sin fin de pequeños *gadgets* que necesitan conectarse con un dispositivo mayor. Como no suelen tener necesidad de trasmitir una cantidad enorme de datos y suelen estar cerca de los otros dispositivos, el *bluetooth* es su conexión ideal.

GPS, equipamientos médicos y hasta militares se benefician de un sistema que no puede ser rastreado a más de unos metros.

3. DESCRIPCIÓN DE ENLACE DE DATOS PUNTO A PUNTO

3.1. ¿Qué es un radioenlace?

Se denomina radio enlace a cualquier interconexión entre los terminales de telecomunicaciones efectuados por ondas electromagnéticas. Además, si los terminales son fijos, el servicio se le denomina como tal y si algún terminal es móvil, se le denomina dentro de los servicios de esas características.

> Se puede definir al radio enlace del servicio fijo, como sistemas de comunicaciones entre puntos fijos situados sobre la superficie terrestre, que proporcionan una capacidad de información, con características de calidad y disponibilidad determinadas. Típicamente estos enlaces se explotan entre los 800 MHz y 42 GHz.[16]

3.2. Ecuaciones de Maxwell en la propagación de ondas electromagnéticas

Los campos electromagnéticos se propagan por el espacio en forma de ondas electromagnéticas, de forma que viajan a través de un medio en el vacío. Existen ecuaciones necesarias para describir el momento de propagación de las ondas, en presencia de materia o en campo vacío.

La existencia de las ondas electromagnéticas fue predicha en la Teoría Electromagnética presentada en 1865 por James Clerk Maxwell, y confirmada por primera vez en 1888 por Heinrich Rudolf Hertz, quien fue el primero en generarlas, detectarlas y estudiarlas.

---

[16] RUESCA, Pedro. *¿Qué es un radioenlace?* http://www.radiocomunicaciones.net/radio/radio-enlace-que-es-un-radioenlace/. Consulta: 8 de noviembre de 2019.

Las ondas electromagnéticas están compuestas por energía en forma de oscilaciones de campo eléctrico y campo magnético que se propagan como lo predicen las Ecuaciones de Maxwell.

$$\oint_s \vec{E}\, d\vec{s} = 0$$

$$\oint_s \vec{B}\, d\vec{s} = 0$$

$$\oint_C \vec{E}\, d\bar{l} = -\frac{\partial}{\partial t} \iint \bar{B}\, d\bar{s}$$

$$\oint_C \vec{B}\, d\vec{l} = u\epsilon \frac{\partial}{\partial t} \iint \vec{B}\, d\vec{s}$$

En función de la distancia del enlace la onda electromagnética va cambiando en todo el recorrido, cambiando en su forma y modificándose, perdiendo intensidad. El modelo de cálculo de propagación de onda que se puede encontrar es deducido por las ecuaciones de Maxwell, donde se pueden encontrar las pérdidas en el espacio libre de la señal. La ecuación de transmisión permite determinar la potencia recibida respecto una antena receptora en un enlace de comunicación por radio, pero sin prestar atención a las condiciones del medio, es decir, se interpretará que el medio tiene unas condiciones ideales de espacio libre.

La ecuación de transmisión

$$P_r = P_t * G_t * G_r * \left(\frac{\lambda}{4 * \pi * r}\right)^2 [W]$$

Donde:

P_r = la potencia recibida en los terminales de entrada de la antena receptora

P_t = la potencia de la antena transmisora

G_t = la ganancia de la antena transmisora

G_r = la ganancia de la antena receptora

λ = la longitud de onda

r = la distancia entre emisor y receptor

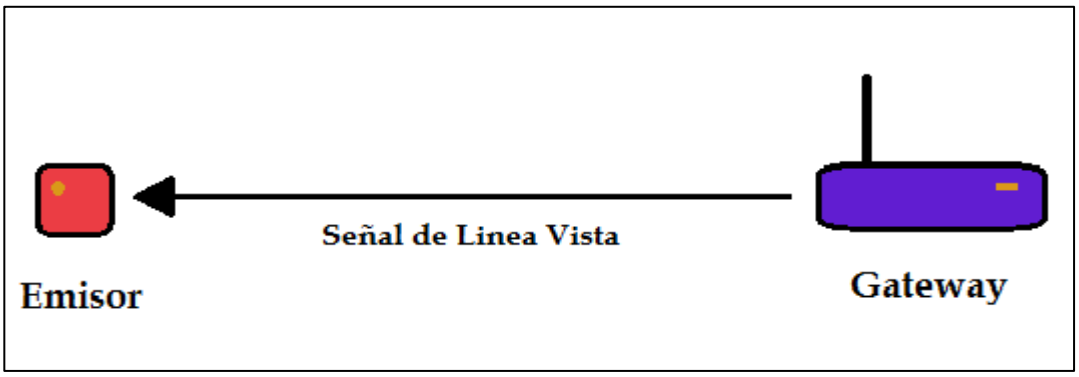
3.3. Tipos de propagación

La propagación es la forma en que las ondas de radio viajan a través del espacio libre (también conocido como medio). La forma en que viajan estas ondas puede afectar la intensidad de su señal. Por lo cual se tienen diferentes tipos de cómo se propagan las ondas electromagnéticas.

3.3.1. Propagación sin obstáculos

Las ondas de radio viajan directamente del emisor al receptor sin ningún obstáculo. Si la distancia entre el emisor y el receptor aumenta, la señal se debilitará. Esta pérdida se conoce como pérdida de espacio libre.

Figura 18. **Propagación de línea vista**



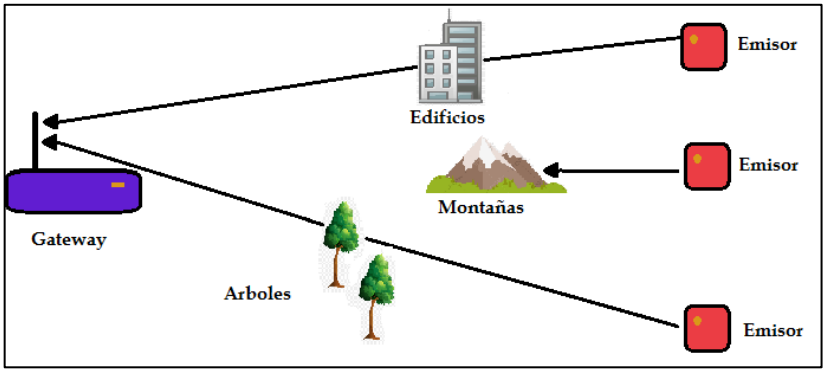
Fuente: elaboración propia, empleando Paint.

3.3.2. Propagación a través de obstáculos

Las ondas de radio pueden penetrar a través de obstáculos que aparecen en su camino. Las ondas de radio pierden fuerza si atraviesan obstáculos hechos de materiales más conductores.

Los edificios están hechos de materiales más conductores que debilitan más la señal en comparación con los árboles. Por esa razón se puede alcanzar mayor alcance con la tecnología LoRa en áreas rurales que en áreas pobladas con edificios. Una montaña bloquea la señal por completo.

Figura 19. **Propagación atreves de obstáculos**

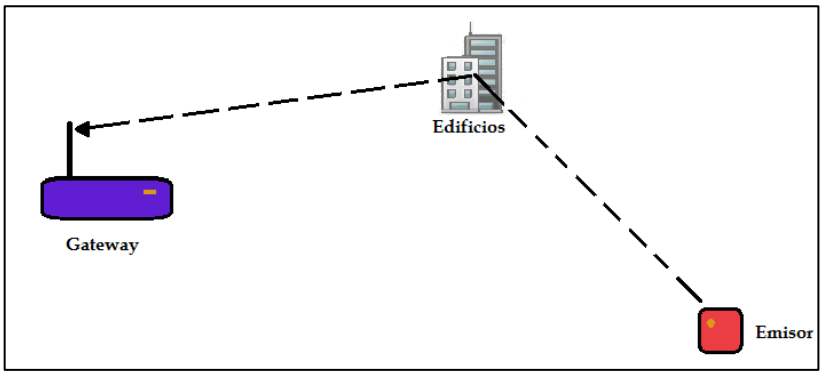


Fuente: elaboración propia, empleando Paint.

3.3.3. Propagación a través de reflexión

La reflexión es el cambio brusco en la dirección de propagación de una onda que golpea la frontera entre dos medios diferentes. Al menos una parte de la onda entrante permanece en el mismo medio. Como las ondas de radio pueden ser reflejadas por edificios que interfieren con su señal directa.

Figura 20. **Propagación a través de reflexión**



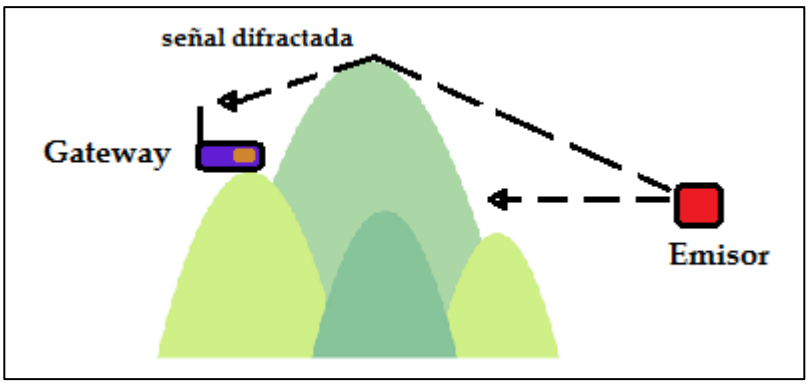
Fuente: elaboración propia, empleando Paint.

3.3.4. Propagación a través de difracción

> La zona de difracción de un transmisor se extiende desde la distancia con visibilidad directa (LoS) en la que el trayecto libre de obstáculos es igual al 60 % del radio de la primera zona de Fresnel, (R1), hasta una distancia más allá del horizonte del transmisor en la que predomina el mecanismo de dispersión troposférica.[17]

Este es el caso de la difracción de un transmisor de ondas utilizando la tecnología LoRa donde las ondas de radio se doblan alrededor de los bordes afilados. El *Gateway* puede recibir una señal de un transmisor, aunque esté sombreada por un gran obstáculo.

Figura 21. **Propagación a través de difracción**



Fuente: elaboración propia, empleando Paint.

La señal difractada si puede llegar al *Gateway*, pero la señal directa es afectada por las colinas y montañas.

[17] UTI. *Propagación por difracción.* https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.526-15-201910-I!!PDF-S.pdf. Consulta: 12 de noviembre de 2019.

3.3.5. Pérdidas en espacio libre

La pérdida de trayecto indica cuánta energía se pierde en el espacio libre a una distancia entre el emisor y receptor. Cuanto mayor sea la distancia entre emisor y receptor, menor será el nivel de energía.

La pérdida de espacio libre se puede calcular de la siguiente manera:

$$L_{fs} = 32{,}45 + 20 \log D + 20 \log f$$

Dónde:

L_{fs} = son las pérdidas en espacio libre en dB

D = distancia entre el emisor y receptor en Km

f = frecuencia en Mhz

3.4. Antenas

Una antena es un dispositivo normalmente construido a base de un material conductor, con el objetivo de permitir la transmisión y recepción de ondas electromagnéticas, y así lograr el intercambio de información de un punto a otro en un radio enlace.

Las antenas se pueden dividir en: elementales, arreglos de antenas, de onda progresiva y parabólicas; cada una de estas con parámetros específicos de los cuales dependerá su aplicación.

3.4.1. Parámetros de antenas

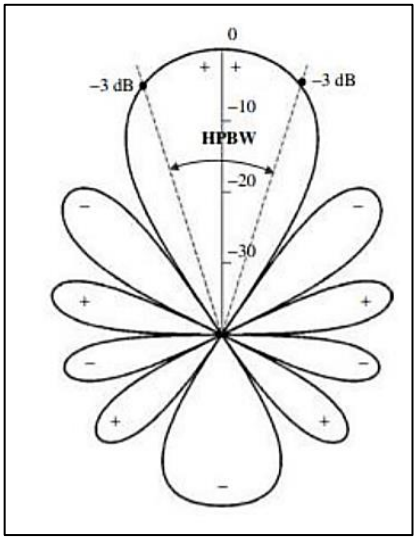
Entre los principales parámetros de las antenas, se puede encontrar el diagrama o patrón de radiación, ancho de banda, directividad, adaptación, eficiencia, impedancia y polarización.

Diagrama de radiación: en una antena, el diagrama o patrón de radiación representa de manera gráfica el comportamiento de las ondas electromagnéticas irradiadas, normalmente representa la densidad de potencia radiada.

Según el comportamiento del patrón de radiación, las antenas se pueden clasificar en: direccionales, omnidireccionales e isotrópicas.

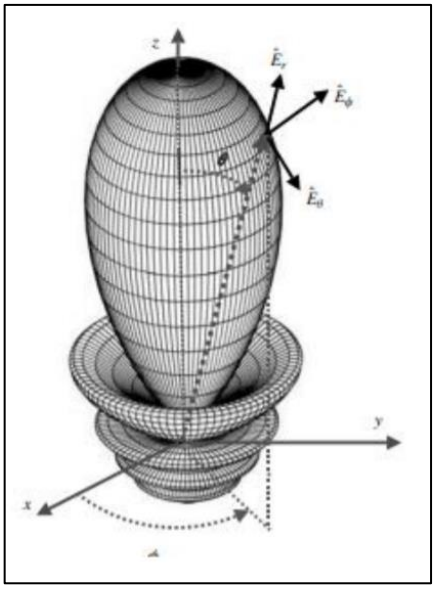
Una antena con patrón de radiación direccional o unidireccional es aquella que enfoca la mayor cantidad de potencia radiada en una dirección, logrando así una mayor distancia de transmisión. En la siguiente gráfica se observa el patrón de radiación de una antena direccional.

Figura 22. **Patrón de radiación direccional 2 dimensiones**



Fuente: BALANIS, Constantine. *Antenna Theory third edition analysis and design*. p. 29.

Figura 23. **Patrón de radiación direccional 3 dimensiones**

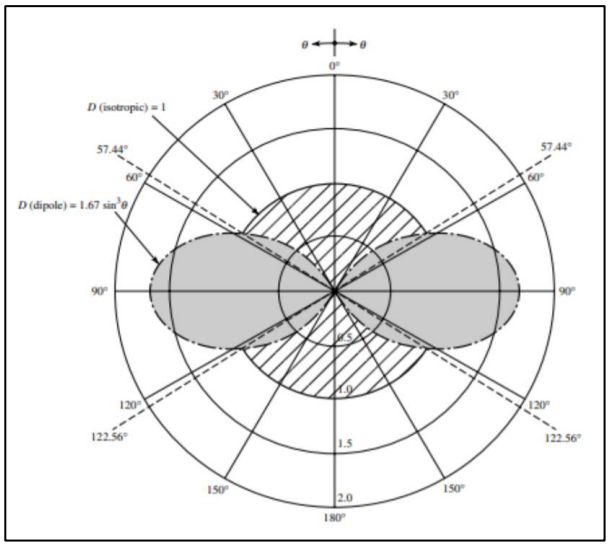


Fuente: BALANIS, Constantine. *Antenna Theory third edition analysis and design*. p. 31.

Una antena con patrón de radiación de una antena omnidireccional es aquella capaz de irradiar de manera uniforme en todas direcciones; sin embargo, la antena capaz de radiar de manera uniforme en 3 dimensiones es la isotrópica.

En la siguiente gráfica se observa el patrón de radiación de un dipolo, categorizado como “antena omnidireccional”[18].

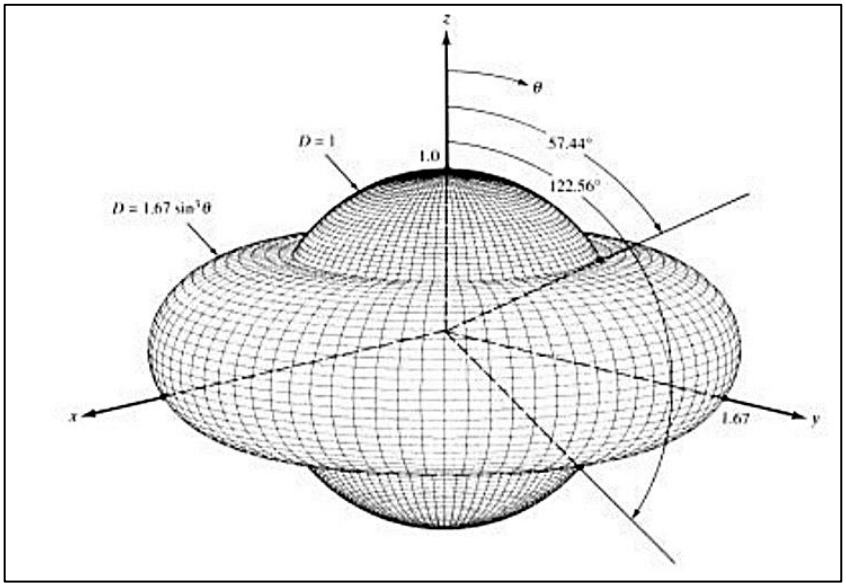
Figura 24. **Patrón de radiación antena omnidireccional 2 dimensiones**



Fuente: BALANIS, Constantine. *Antenna Theory third edition analysis and design*. p. 49.

[18] BALANIS, Constantine. *Antenna Theory third edition analysis and design*. p. 1073.

Figura 25. **Patrón de radiación antena omnidireccional 3 dimensiones**



Fuente: BALANIS, Constantine. *Antenna Theory third edition analysis and design*. p. 29.

Ancho de banda: se conoce como el rango de frecuencias en el cual la antena puede trabajar de manera óptima; está delimitado por las frecuencias en las cuales la intensidad de la señal que recibe la antena decrece 3 dB respecto de la intensidad con la que recibe las señales a la frecuencia para la que fue diseñada.

Directividad: es la relación de la potencia radiada en una dirección con la potencia radiada de forma isotrópica. Se puede calcular con la siguiente ecuación:

$$\mathrm{D}(\theta,\varphi) = \frac{\mathrm{P}(\theta,\varphi)}{\frac{N_r}{4\pi \mathrm{r}}}$$

Donde:

$P(\theta, \varphi)$ = representa la potencia radiada en una dirección

$\frac{N_r}{4\pi r}$ = representa la potencia radiada de forma isotrópica

- Adaptación: consiste en aplicar el teorema de la máxima transferencia de potencia, de tal manera que en el sistema la impedancia de entrada sea igual a la de salida. Si una antena no está adaptada, parte de la potencia será reflejada, siendo esto una pérdida para el sistema.

- Eficiencia: es la relación que existe entre la ganancia y la directividad, representada con el símbolo η en la siguiente ecuación:

$$\eta = \frac{\text{Ganancia}}{\text{Directividad}}$$

La impedancia se puede descomponer en varios elementos:

- Ra: resistencia física de la antena; esta depende del material utilizado para la construcción de la antena, así como el tipo de antena.

- Rr: resistencia de radiación.

- X: reactancia (este valor se encuentra en función de la frecuencia de operación).

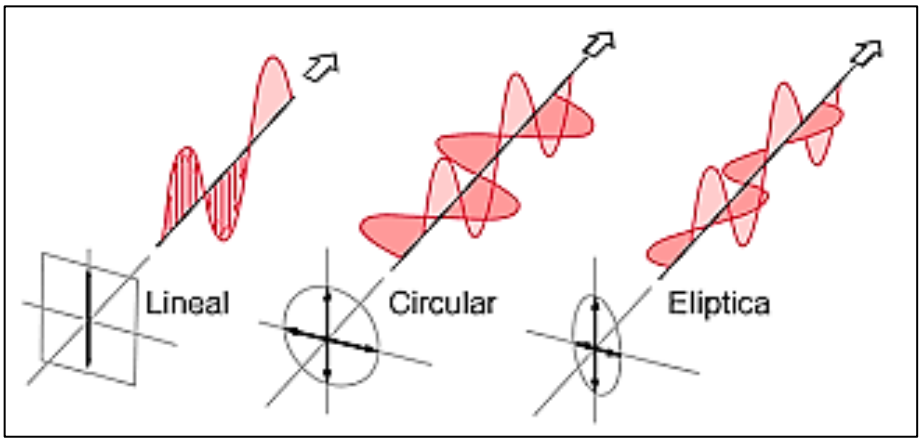
La impedancia de una antena se puede obtener a partir de la siguiente ecuación:

$$Z = (Ra + Rr) + jX$$

Se tiene que la componente real de la impedancia se obtiene por la suma de la resistencia física y la radiación; la componente compleja está dada por la reactancia de la antena.

Polarización: la polarización en las ondas electromagnéticas sirve para conocer el comportamiento de los campos eléctricos y magnéticos en una onda transversal; existe la polarización lineal, circular y elíptica.

Figura 26. **Tipos de polarización**



Fuente: HYPERPHYSICS. *Tipos de polarización.* http://hyperphysics.phyastr.gsu.edu/hbasees/phyopt/polclas.html Consulta: 3 de octubre 2019.

3.4.2. Tipos de antenas

Según las características de las antenas y su aplicación, es posible dividirlas en: elementales, como el dipolo y la espira; de onda progresiva como las de apertura y de bocina; los arreglos de antenas, como las antenas Yagi; para finalizar con las antenas parabólicas o tipo plato.[19]

[19] ANGUERA, Jaume; PÉREZ, Antonio. *Teoría de antenas.* p. 3.

- Dipolo elemental: es un conductor de corriente con longitud L, el cual es recorrido por una corriente uniforme, cuyas dimensiones son pequeñas comparadas con su longitud de onda; la mayor parte de las antenas que trabajan en frecuencias inferiores a 1 MHz se comportan como dipolos elementales.

- Arreglo de antenas: consisten en la agrupación de uno o más tipos de antenas simples unidas; se pueden tener arreglos de antenas lineales, de antenas planas y cilíndricos; su utilización dependerá de la aplicación.

- Antenas lineales: utilizadas normalmente para la transmisión de emisoras de radio y televisión, entre ellas se puede encontrar la antena Yagi, que es una antena direccional compuesta de una serie de dipolos colocados uno tras otro.

- Arreglo de antenas planas: son utilizados normalmente en las comunicaciones satelitales; este tipo de arreglo se consigue colocando todos los elementos sobre un plano, prácticamente de 2 dimensiones; consiguiendo así una mayor directividad y control del patrón de radiación; los arreglos de antenas planas más comunes son los reticulares y los circulares.

- Antenas de apertura: son aquellas que buscan concentrar la radiación electromagnética en un punto, de tal manera que consiguen concentrar las ondas de radio en una dirección, para conseguir una mayor directividad. Entre estas se pueden mencionar las antenas de bocina y los reflectores parabólicos.

- Antenas de bocina: son utilizadas, en la mayoría de los casos, para trabajar con microondas; tienen un ancho de banda bastante amplio; las antenas tipo bocina consisten en una guía de onda cuya área va en incremento hasta un extremo abierto en el cual ingresan las ondas electromagnéticas.

- Reflector parabólico: este concentra la energía en un punto conocido como foco, al hacer rebotar las ondas electromagnéticas en una superficie reflectora.

3.5. Zona de Fresnel

La zona de Fresnel es un cuerpo de forma elíptica alrededor de la línea directa de la ruta de visión entre el nodo y el *Gateway.* Cualquier obstáculo dentro de este volumen, por ejemplo, edificios, árboles, valles o terrenos pueden debilitar la señal transmitida incluso si hay una línea de visión directa entre el emisor y el *Gateway.* El radio máximo de la zona de Fresnel, ubicada a la mitad de la distancia entre el emisor y el *Gateway*, se calcula de la siguiente manera:

$$R_{max} = 8{,}657 \sqrt[2]{\frac{D}{f}}$$

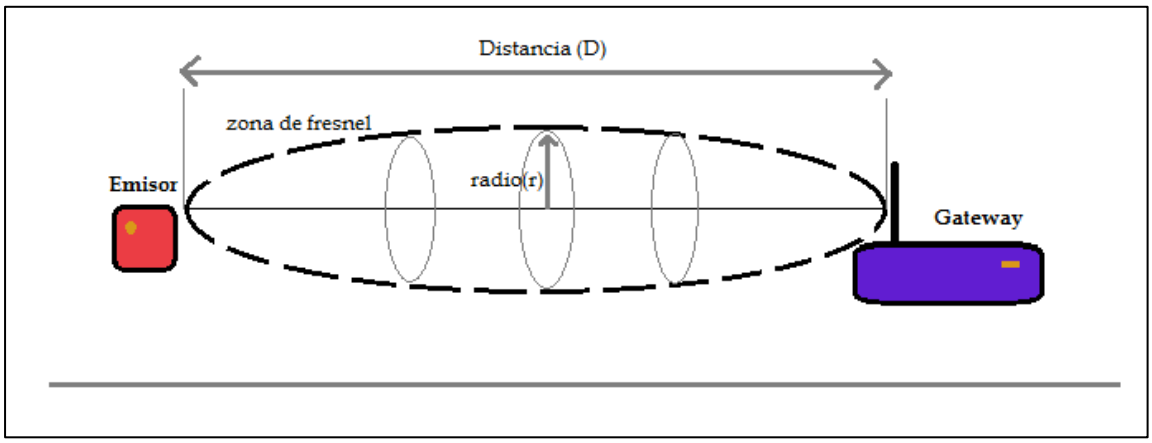
Dónde:

R_{max} = es el radio máximo de la zona de Fresnel en metros

D = es la distancia en Km

f = es la frecuencia en GHz

Figura 27. **Zona de Fresnel**



Fuente: elaboración propia, empleando Paint.

En el caso ideal el emisor envía una señal al *Gateway* sin ninguna interferencia u obstáculo dentro de la zona de Fresnel, pero si se aumenta la distancia entre emisor y el *Gateway* entonces el radio de la zona de Fresnel aumenta por lo tanto la tierra estaría dentro de la zona de Fresnel, lo cual significa que la señal va ser reflejada la cual interfiere con la señal directa.

Existen 2 posibles casos cuando una señal es reflejada:

- La señal directa y la señal reflejada están en fase.
- La señal directa y la señal reflejada están desfasadas.

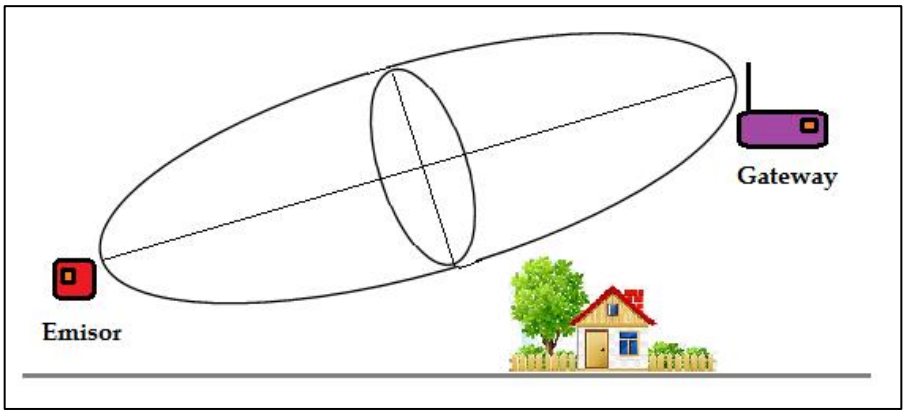
El peor de estos casos es cuando la señal directa y la señal reflejada están desfasadas debido que se cancelan mutuamente y se pierde la señal transmitida.

En general hay que evitar obstáculos dentro de la zona de Fresnel para evitar impactos negativos en la intensidad de la señal.

3.6. Posición de la antena del *Gateway*

Para evitar objetos dentro de la zona de Fresnel se puede emplear la siguiente técnica, localizar la antena del *Gateway* en una ubicación más alta tal como se ve en la imagen, el emisor envía una señal al *Gateway* sin ninguna interferencia sin obstáculos dentro de la zona de Fresnel.

Figura 28. **Representación de la posición de antena**

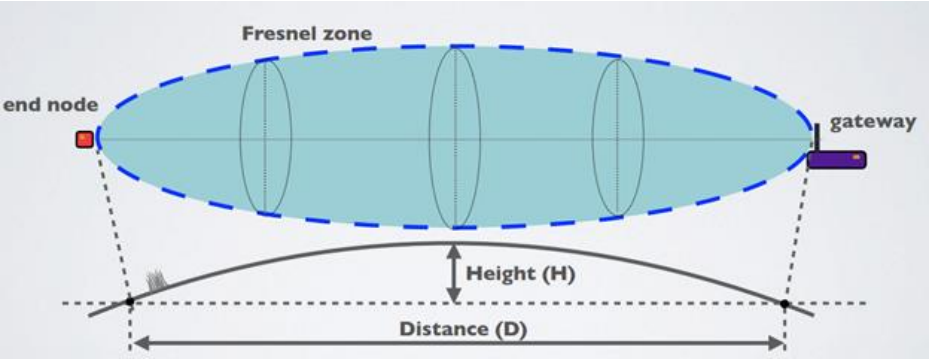


Fuente: elaboración propia, empleando Paint.

3.7. Consideración de la curvatura de la tierra

La ecuación de la zona de Fresnel desprecia la curvatura de la tierra, se basa en una tierra plana. No toma en cuenta la curvatura de la tierra.

Figura 29. **Curvatura de la Tierra**



Fuente: MOBILE FISH. *Zona Fresnel*. https://www.mobilefish.com. Consulta: 3 de septiembre de 2019.

Para calcular la altura H se usa la siguiente ecuación:

$$H = \frac{1\,000\,D^2}{8R}$$

Dónde:

H = es la altura (margen de curvatura de la tierra) en metros.

D = es la distancia entre el emisor y receptor en Km.

R = es el radio de la tierra en Km que tiene el valor de 8 504 Km.

En la siguiente tabla se puede observar que radioenlaces menores a 2 kilómetros la curvatura de la tierra es despreciable, pero si la distancia es de 5 km o más entonces la curvatura de la tierra debe ser considerada.

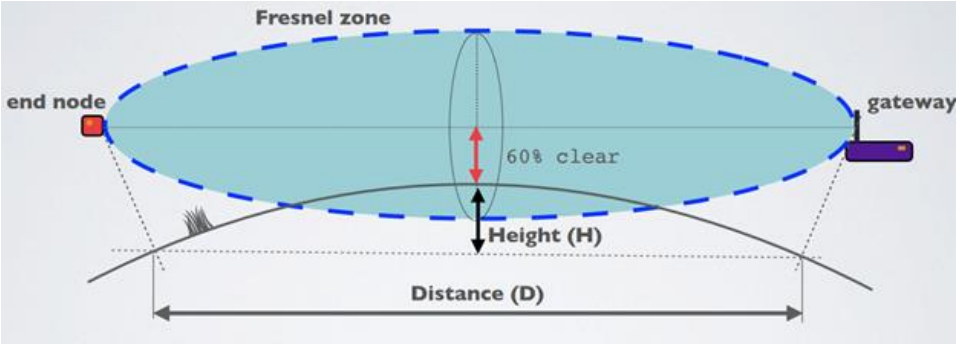
Tabla XII. **Distancia vs. curvatura de la Tierra**

Distancia [Km]	H [m]
0,1	Despreciable
0,5	Despreciable
1	Despreciable
2	Despreciable
5	0,4
10	1,5
15	3,3
20	5,9
25	9,2
30	13,2

Fuente: elaboración propia.

Como regla general, la zona de Fresnel siempre debe estar libre de obstrucciones, pero esto puede ser poco práctico, por lo que se dice que más allá del 40 % de bloqueo, la pérdida de señal será significativa.

Figura 30. **Zona de Fresnel vs curvatura de la Tierra**



Fuente: MOBILE FISH. *Curvatura de la tierra*. https://www.mobilefish.com. Consulta: 4 de septiembre de 2019.

3.7.1. Ejemplo de cálculo usando un enlace LoRa

Si se tiene un *Gateway* y un nodo que transmiten a una frecuencia de 915 MHz, calcular el radio de la zona de Fresnel a diferentes distancias, también considerar los siguientes casos:

- Libre de obstáculos a un 100 % (caso ideal)
- Libre de obstáculos a un 60 %

  - $f = 0{,}915\ [GHz]$
  - $H = Altura\ en\ [m]$

$$R_{max} = 8{,}657\sqrt[2]{\frac{0{,}6D}{f}}\quad H = \frac{1\ 000\ D^2}{8R}$$

Tabla XIII. **Consideraciones de la zona de Fresnel**

D(m)	0,6D[Km]	R[m]	R + H[m]
100	0,06	2,22	2,22
500	0,3	4,96	4,96
1 000	0,6	7,01	7,01
2 000	1,2	9,91	9,91
5 000	3	15,68	16,08
10 000	6	22,17	23,67

Fuente: elaboración propia.

La cuarta columna de la tabla XIII se puede encontrar el radio de la zona de Fresnel más la altura a considerar por la curvatura de la tierra a un 60 % libre de

obstáculos, se puede notar que la altura de localización del nodo y el *Gateway* a considerar es mucho menor que a un 100 % libre de obstáculos.

3.8. Cómo mejorar el rendimiento de la señal de radio

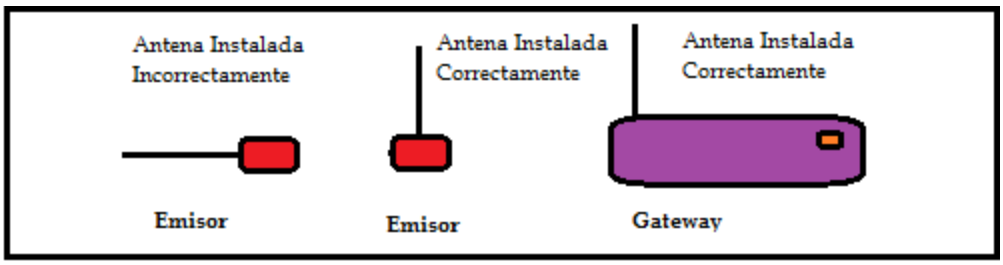
Para el mejor rendimiento de la señal de radio:

- La antena del *Gateway* debe colocarse al aire libre en una ubicación alta (evitando obstáculos en la zona de Fresnel).

- El diseño de la antena para el emisor y el *Gateway* debe optimizarse para su frecuencia regional.

- Mantenga la polarización de la antena vertical tanto para el *Gateway* como para el nodo y use una antena omnidireccional para cubrir una mayor área.

Figura 31. **Polarización de antena**



Fuente: MOBILE FISH. *Polarización de antena.* https://www.mobilefish.com. Consulta: 4 de septiembre de 2019.

3.9. Presupuesto de enlace y margen de enlace

Un presupuesto de radio enlace completo es simplemente la suma de todos los aportes (en decibeles) en el camino de las tres partes principales. Las cuales son:

- El lado de transmisión con potencia efectiva de transmisión.
- Pérdidas en la propagación.
- El lado de recepción con efectiva sensibilidad receptiva.

No es suficiente que la señal que llega al receptor sea mayor que la sensibilidad del mismo, sino que además se requiere que haya cierto margen para garantizar el funcionamiento adecuado. La relación entre el ruido y la señal se mide por la tasa de señal a ruido.

En la siguiente sección se describe los aspectos más importantes del presupuesto de enlace y margen de enlace.

3.9.1. Modulación y demodulación

> La modulación es un paso muy importante en la transmisión de una señal. Nuestra señal de mensaje es generalmente una señal de baja frecuencia y la pérdida de la señal en el trayecto es proporcional al cuadrado de la longitud de onda (y por lo tanto inversamente proporcional al cuadrado de la frecuencia).[20]

Por lo tanto, se debe encontrar un método para idear una manera de codificar nuestra señal de mensaje a una señal de frecuencia más alta. Podemos

[20] AERO IITB. *Modulacion y Demodulacion.* https://www.aero.iitb.ac.in/satelliteWiki/index.php/Modulation_and_Demodulation. Consulta: 4 de septiembre de 2019.

jugar con diferentes propiedades de la señal portadora como la frecuencia, la fase, la amplitud para codificar nuestra señal de mensaje en la señal portadora.

La demodulación es una técnica para obtener la señal de mensaje de la señal de recepción. La modulación y la demodulación van de la mano. El siguiente diagrama de bloques da una idea general:

Figura 32. **Modulación y demodulación**

Fuente: MOBILE FISH. *Modulación & Demodulación.* https://www.mobilefish.com. Consulta: 5 de septiembre de 2019.

3.10. Presupuesto de enlace

Un presupuesto de enlace es la suma de todas las ganancias y pérdidas del transmisor, a través del medio (también conocido como espacio libre), al receptor en un sistema de telecomunicaciones. Es una forma de cuantificar el rendimiento del enlace.

Cuando una señal se propaga a través del medio, la señal pierde fuerza. Esto se llama pérdida de ruta o atenuación de ruta. Un buen presupuesto de enlace es esencial para el funcionamiento del mismo. Una ecuación de presupuesto de enlace simple se ve como la ecuación siguiente:

$$Potencia_{recibida} = Potencia_{transmitida} - Pérdidas$$

3.11. Potencia del transmisor

La potencia de transmisión es la potencia de salida del radio y debe especificarse en dbm; de lo contrario, no se conoce su valor. El límite superior depende de las regulaciones vigentes en cada país, dependiendo de la frecuencia de operación y puede cambiar al variar el marco regulatorio. En general, los radios con mayor potencia de salida son más costosos.

3.12. Ganancia de antena del lado del transmisor

La ganancia de una antena varía entre 2 dBi y 30 dBi. Tenga en cuenta que hay muchos factores que disminuyen la ganancia real de una antena. Las pérdidas pueden ocurrir por muchas razones, principalmente relacionadas con una incorrecta instalación (pérdidas en la inclinación, en la polarización, objetos metálicos adyacentes). Esto significa que solo puede esperar una ganancia completa de antena, si está instalada en forma óptima. La ganancia de antena debe especificarse en dBi y vienen dados en varios rangos tal como se muestra en la tabla XIV.

Tabla XIV. **Ganancia de antena**

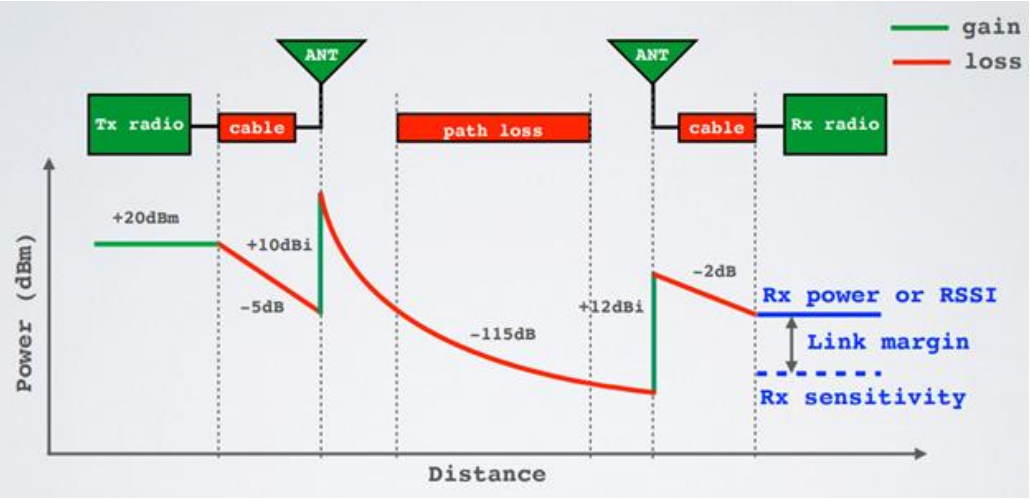
Ganancia	Características
2 dBi	Antena simple
8 dBi	Omnidireccional estándar
21-31 dBi	Parabólica

Fuente: elaboración propia.

3.13. Pérdidas

Las pérdidas de propagación están relacionadas con la atenuación que ocurre en la señal cuando esta sale de la antena de transmisión hasta que llega a la antena receptora. Tales como pérdidas en cables y conectores. La señal que se propaga a través del medio se mide en dB.

Figura 33. **Pérdidas de la señal**



Fuente: MOBILE FISH. *Pérdidas de señal.* https://www.mobilefish.com. Consulta: 5 de septiembre de 2019.

3.13.1. Ejemplo de pérdidas de señal de un enlace LoRa

- Potencia de salida del transmisor = 20 dBm
- Pérdida en guía de onda de antena del nodo = - 5 dB
- Ganancia de antena omnidireccional del nodo = 10 dBi
- Pérdidas de trayectoria = - 115 dB
- Ganancia de antena omnidireccional del *Gateway* = 12 dBi

- Pérdida de guía de onda de antena del *Gateway* = - 2 dB

$$P_r = 20 - 5 + 10 - 115 + 12 - 2$$
$$P_r = -80\ dBm$$

Donde:

P_r = la potencia recibida en el *Gateway*.

3.14. Margen de enlace

El margen de enlace es la diferencia entre la potencia mínima esperada recibida por el extremo receptor y la sensibilidad de recepción.

$$Margen_{enlace} = P_r - S_r[\text{dBm}]$$

Donde:

P_r = la potencia recibida

S_r = la sensibilidad del receptor

3.15. Potencia recibida

Es el equilibrio del enlace, que tiene en cuenta todas las ganancias y pérdidas de potencia experimentadas por la señal de comunicación.

$$P_r = P_t + G - P\ [\text{dBm}]$$

Donde:

P_r = la potencia recibida

P_t = la potencia transmitida

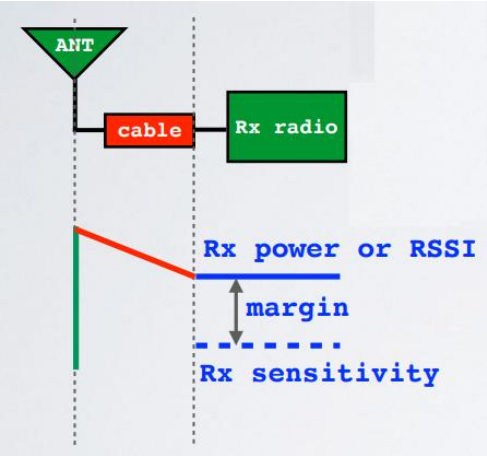
G = todas las ganancias

P = todas las pérdidas

3.16. Sensibilidad de receptor

La sensibilidad del receptor es el nivel de potencia más bajo en el que el *Gateway* puede recibir o demodular la señal. Los receptores LoRa son muy sensibles y ofrecen una sensibilidad de hasta - 148 dBm, debido al uso de *Chirp Spread Spectrum*. La Sensibilidad del receptor se mide en dBm.

Figura 34. **Sensibilidad de receptor**



Fuente: MOBILE FISH. *Sensibilidad de receptor.* https://www.mobilefish.com. Consulta: 6 de septiembre de 2019.

3.16.1. Ejemplo de sensibilidad de enlace LoRa

A continuación, se demuestra el cálculo del margen de enlace, que tiene en cuenta la potencia recibida y la sensibilidad del receptor.

$$Margen_{enlace} = P_r - S_r\ [dBm]]$$

- Potencia recibida = - 80 dBm
- Sensibilidad del receptor = - 90 dBm

$$Margen_{enlace} = -80 - (-90)$$
$$Margen_{enlace} = 10\ dBm = 10\ mW$$

3.17. Máximo presupuesto de enlace

El presupuesto de enlace máximo se puede utilizar como valor de referencia para comparar una radio con la siguiente.

3.17.1. Ejemplo de máximo presupuesto de enlace

A continuación, se demuestra que el máximo presupuesto de enlace es mayor que el límite inferior de la sensibilidad del receptor.

- Potencia máxima del nodo = 20 dBm
- Sensibilidad del *Gateway* más baja = - 148 dBm

$$P_{max} = P_{t-max} - S_{r-min}\ [dBm]$$

Donde:

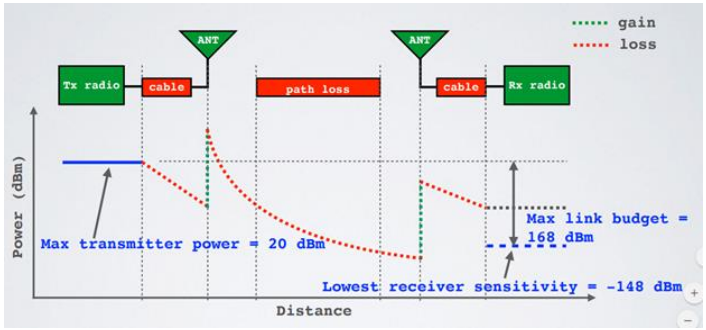
P_{max} = presupuesto de enlace máximo

P_{t-mx} = potencia máxima del nodo

S_{r-min} = sensibilidad del *Gateway* más baja

$$P_{max} = 20 - (-148)$$
$$P_{max} = 168\ dBm$$

Figura 35. **Máximo presupuesto de enlace**

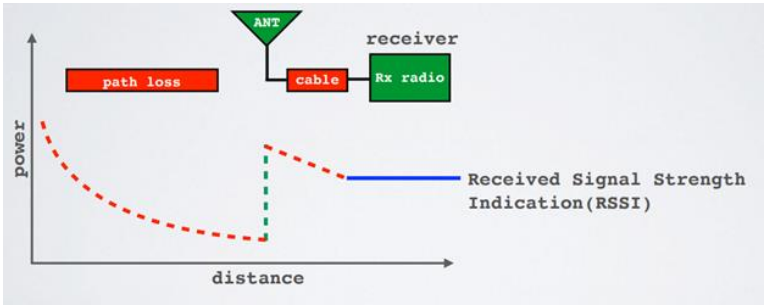


Fuente: MOBILE FISH. *Máximo Presupuesto de enlace*. https://www.mobilefish.com. Consulta: 13 de septiembre de 2019.

3.18. RSSI

La indicación de intensidad de señal recibida (RSSI) es la potencia de señal recibida en milivatios y se mide en dBm. Este valor se puede usar como una medida de qué tan bien un receptor puede "escuchar" una señal de un nodo.

Figura 36. **RSSI**



Fuente: MOBILE FISH. *RSSI.* https://www.mobilefish.com. Consulta: 12 de septiembre de 2019.

RSSI está en dBm y es un valor negativo. Cuanto más cerca de 0, mejor será la señal.

El valor típico de LoRa RSSI es:

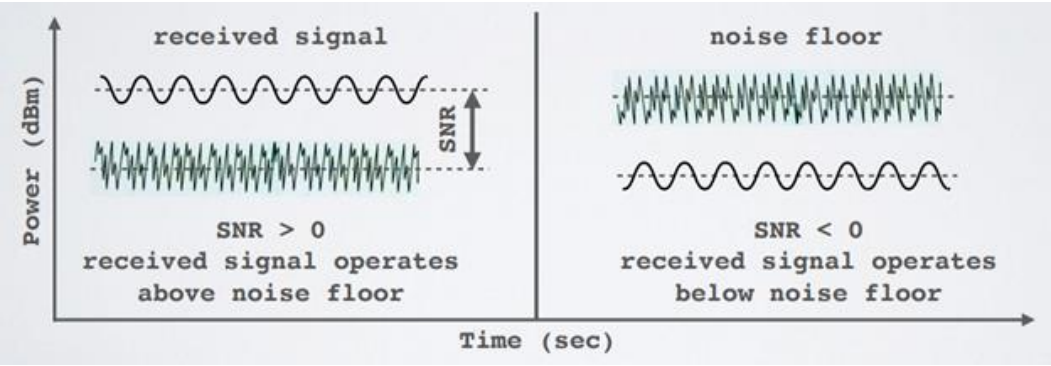
- RSSI mínimo = -120 dBm.
- Si RSSI = -30 dBm: la señal es fuerte.
- Si RSSI = -120 dBm: señal débil.

3.19. SNR

La relación señal / ruido (SNR) es la relación entre la señal de potencia recibida y el nivel de potencia del *noise floor*.

El *noise floor* es un área de todas las fuentes de señales interferentes no deseadas que pueden corromper la señal transmitida y, por lo tanto, se presentarán retransmisiones

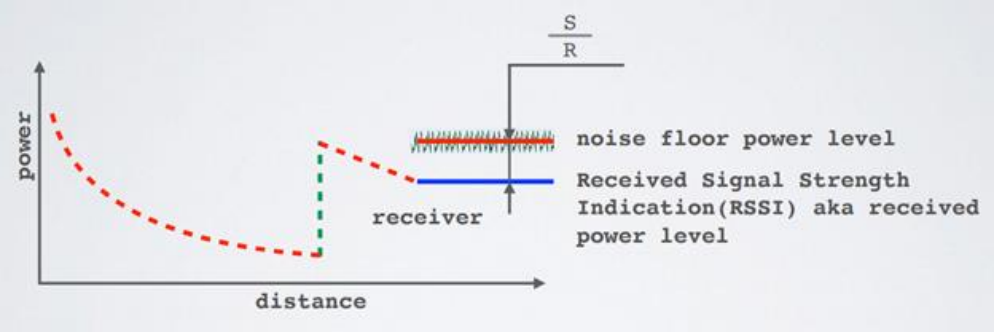
Figura 37. **Relación señal/ruido**



Fuente: MOBILE FISH. *Relación Señal/Ruido.* https://www.mobilefish.com. Consulta: 14 de septiembre de 2019.

Normalmente, el *noise floor* es el límite físico de sensibilidad, sin embargo, LoRa funciona por debajo del nivel de ruido.

Figura 38. ***Noise floor***



Fuente: MOBILE FISH. *Noise Floor.* https://www.mobilefish.com. Consulta: 18 de septiembre de 2019.

Los valores típicos de LoRa SNR están entre: -20 dB y + 10 dB Un valor más cercano a + 10 dB significa que la señal recibida está menos dañada.

LoRa puede demodular señales que están entre -7,5 dB y -20 dB por debajo del nivel de ruido.

3.19.1. SNR LIMIT

Cada factor de ensanchamiento tiene un límite SNR, si se alcanza este límite, el receptor no podrá demodular la señal.

En la tabla XV, puede encontrar el límite de SNR para cada factor de ensanchamiento además Si el SF aumenta en 1, el límite de SNR cambia en -2,5 dB.

Tabla XV. **SNR *limit***

Factor de ensanchamiento	**Chips/ *symbol***	**SNR *limit* [dB]**
7	128	-7,5
8	256	-10
9	512	-12,5
10	1 024	-15
11	2 048	-17,5
12	4 096	-20

Fuente: elaboración propia.

Para calcular la sensibilidad recibida se usa la siguiente ecuación:

$$S_r = -174 + 10\log_{10}(BW) + NF + SNR_{limit}\ [dBm]$$

Donde:

BW = ancho de banda en Hz

NF = figura de ruido del receptor (NF) en dB

SNR_{limit} = límite de señal/ruido en dB

El NF es fijo para los chips transceptores de LoRa SX1272 y SX1276 utilizan NF = 6 dB.

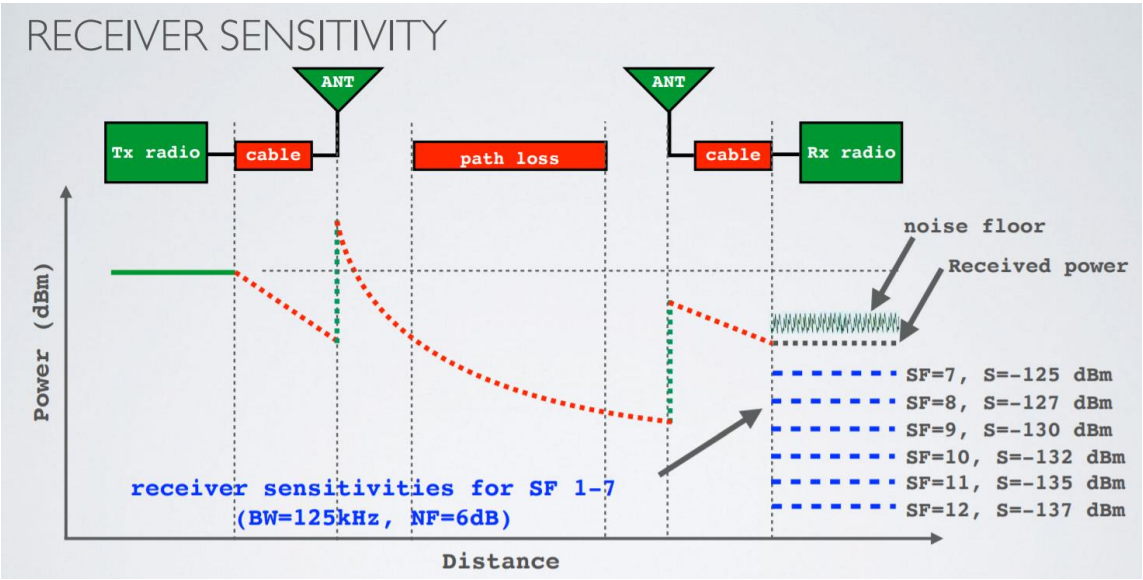
Por ejemplo, si el ancho de banda tiene el valor de 125 kHz, NF=6 dB se podrá calcular la sensibilidad del receptor para SF 7-12 utilizando el *limit* SNR de la tabla XV.

Tabla XVI. **Sensibilidad del receptor variando SF**

SF	S [dBm]
7	128
8	256
9	512
10	1 024
11	2 048
12	4 096

Fuente: elaboración propia.

Figura 39. **Sensibilidad del receptor variando SF**



Fuente: MOBILE FISH. *Sensibilidad del receptor.* https://www.mobilefish.com. Consulta: 23 de septiembre de 2019.

Si la distancia entre el nodo y el *Gateway* aumenta, la señal se vuelve más débil y, por lo tanto, se necesita un factor de propagación mayor para que la sensibilidad del receptor sea menor y sea capaz de demodular la señal recibida.

Para demodular la señal recibida. El factor de dispersión es cercano a 7 cuando se está cerca del *Gateway* y 12 cuando se está lejos del *Gateway*.

4. DISEÑO DE ENLACE DE DATOS PUNTO A PUNTO BASADO EN LORA Y *BLUETOOTH*

4.1. Descripción del diseño

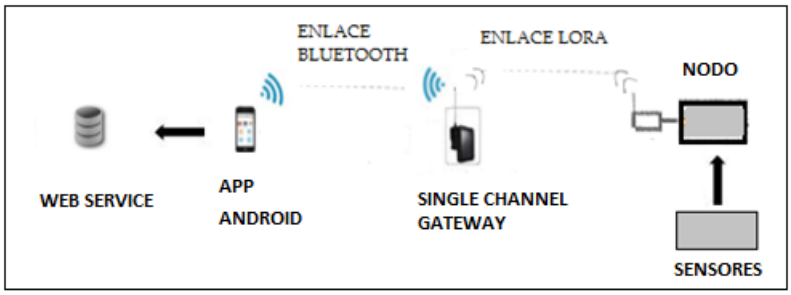
Con la finalidad de probar el funcionamiento del sistema diseñado, en este documento, se han realizado pruebas del sistema. El sistema consiste en un nodo, un *Gateway*, un celular Android y un servidor web. El nodo incluye una red de sensores que envía los datos obtenidos por un enlace de comunicación LoRa y posteriormente los datos los envía por un enlace *bluetooth* a un teléfono Android donde una aplicación muestra los datos en una interfaz de usuario, cada 15 minutos la aplicación envía los datos a un servicio web que se encarga de almacenar los datos en una base de datos.

Con el objetivo de mantener el ejemplo de funcionamiento lo más claro y simple posible no se han añadido medidas de seguridad adicionales para el acceso a la aplicación Android. El ente que tenga el instalador de la aplicación, usuario y contraseña del servicio web puede tener acceso a los datos de los sensores. No obstante, si se desea reproducir este ejemplo en una aplicación comercial, se recomienda agregar más medidas de seguridad.

Cada sección de este ejemplo de implementación se detallará, para que cualquier persona con conocimientos básicos de redes de computadoras y programación en App *Inventor,* PHP y C++ pueda replicar el mismo ejemplo o modificar características que considere.

En la figura 40, se expone la estructura del diseño propuesto.

Figura 40. **Estructura del diseño**



Fuente: elaboración propia, empleando Paint.

4.2. Procedimiento

En las subsecciones siguientes se detalla el diseño del nodo y *Gateway,* adicionalmente las pruebas del funcionamiento y sus configuraciones. A continuación, se detallan las fases de su diseño.

4.2.1. *Hardware* del diseño del nodo y *Gateway*

A continuación, se listan los elementos utilizados para las pruebas de funcionamiento, tanto de *hardware* como de *software*.

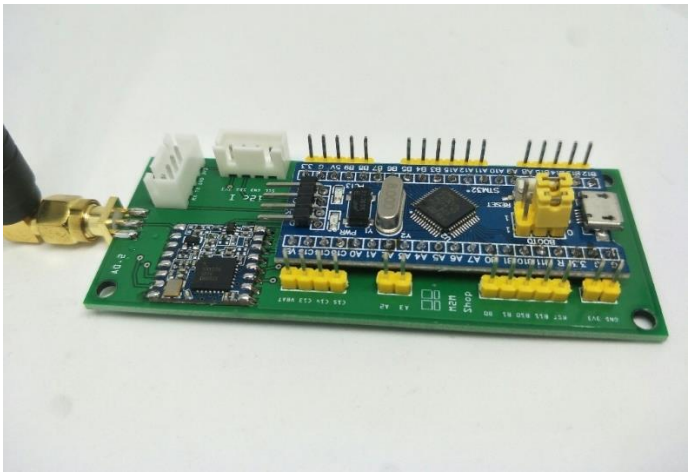
- Elementos de *hardware*
  - Nodo STM32 *Blue Pill* LoRa
  - Módulo de comunicación LoRa
  - Módulo de comunicación *bluetooth*
  - Circuitos de adquisición de datos de sensores
  - Módulo de alimentación

- Elementos de *software*
  - Entorno de desarrollo integrado de Arduino
  - Entorno de diseño de diagramas esquemáticos *Eagle* 7.6.0 profesional
  - Entorno de desarrollo de aplicaciones Android App *Inventor*
  - Entorno de servicio de *Hosting Web*

4.2.1.1. Construcción del nodo y *Gateway* usando STM32 *Blue Pill* LoRa

La tarjeta de desarrollo STM32 *blue Pill* LoRa ha sido construido y diseñado por M2M *Shop* en Tailandia y fue comprado en la página web tindie.com, la ventaja de esta tarjeta es que ya trae el módulo LoRa listo para configurarse para la aplicación que se requiera, por lo tanto, se utilizó en la construcción del nodo y *Gateway* de este diseño.

Figura 41. **Tarjeta de desarrollo STM32 *blue Pill***



Fuente: TINDIE. *STM32 Blue Pill LoRaWAN node.* https://www.tindie.com/products/m2m/stm32-blue-pill-lorawan-node/ consulta: 15 de enero 2020.

4.2.1.2. Lista de componentes

El diseño del nodo y la red de sensores consta de los siguientes componentes:

Tabla XVII. **Lista de componentes del nodo**

Código	**Cantidad**	**Nombre**	**Descripción**
U1	1	STM32 *Blue Pill* LoRaWAN node	Tarjeta de desarrollo con modulo LoRa incluido
s1	1	*Tipping Bucket Rain Gauge*	Pluviómetro tipo balancín
J1, J2	2	Conectores	Conectores tipo RJ11 hembra
C1	1	Capacitor	Capacitor de cerámica 10nF
s2	1	*eTape Liquid Level sensor*	Sensor de nivel de estado solido
OLED1	1	SSD1306	Módulo de pantalla OLED con controlador
U2	1	TP4056	Cargador de batería de iones de litio
BAT1	1	Batería	Batería recargable de 3,7V 2 500mAh
P1	1	Panel Solar	1W Solar Panel 80X100 mm
X1	1	AK500/2	Bloque de terminal PTR 250V 2-*Pole Grey*
R1, R3	2	Resistencias	Resistencia de 10 kΩ para 0,25 W
R2	1	Resistencia	Resistencia de 2 690 Ω para 0,25 W
SW1	1	interruptor	interruptor de 2 posiciones

Fuente: elaboración propia.

El diseño del *Gateway* consta de los siguientes componentes:

Tabla XVIII. **Lista de componentes del *Gateway***

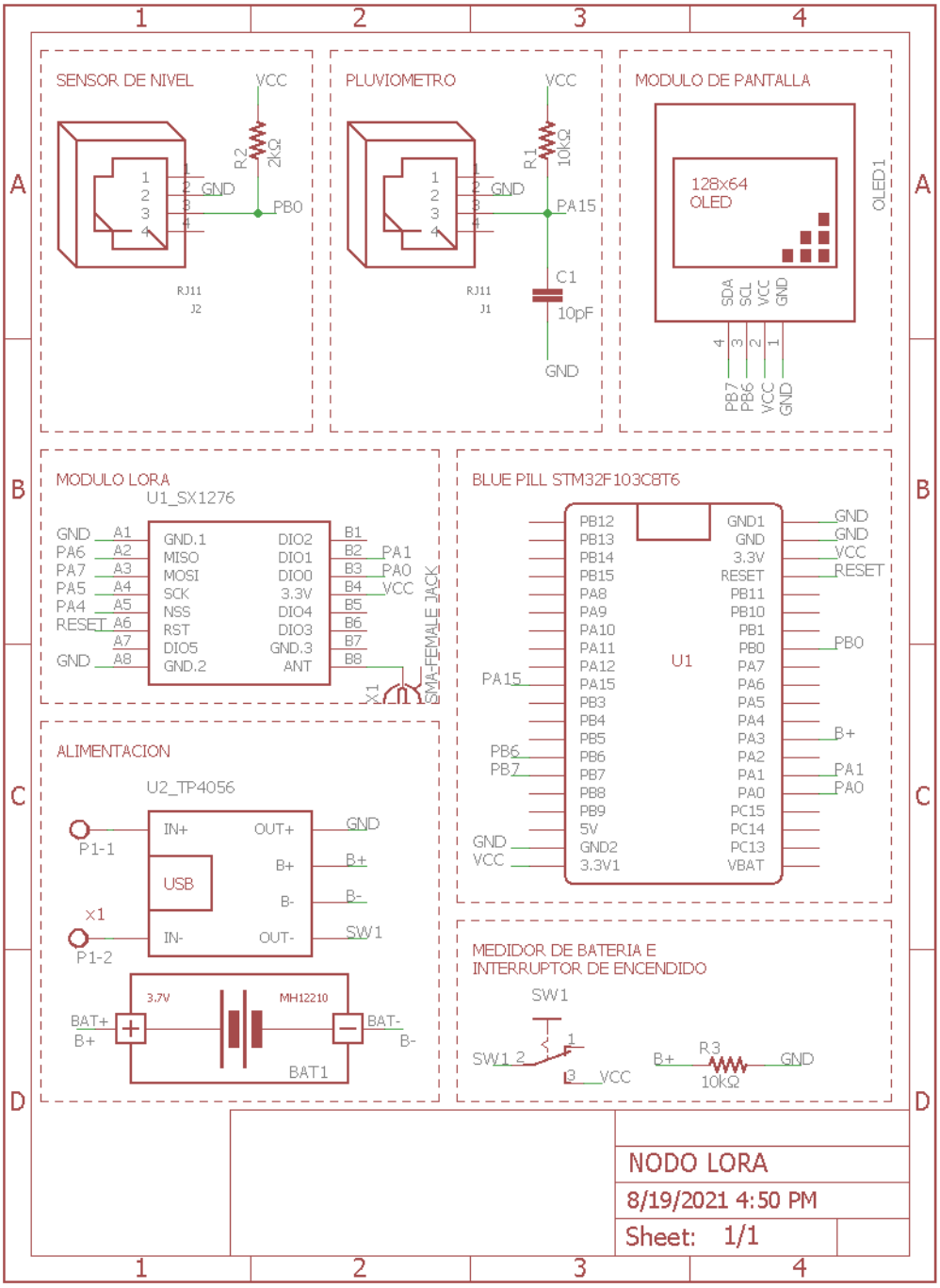
Código	**Cantidad**	**Nombre**	**Descripción**
U3	1	STM32 *Blue Pill* LoRaWAN *node*	Tarjeta de desarrollo con módulo LoRa incluido
OLED2	1	SSD1306	Módulo de pantalla OLED con controlador
BLE1	1	HC-05	Módulo de comunicación *bluetooth*
U4	1	TP4056	Cargador de batería de iones de litio
BAT2	1	Batería	Batería recargable de 3,7 V 2 500 mHh
P2	1	Panel solar	1 W solar panel 80 X 100 mm
X2	1	AK500/2	Bloque de terminal PTR 250 V 2-*Pole Grey*
R4	1	Resistencia	Resistencia de 10 kΩ para 0,25 W
SW2	1	Interruptor	Interruptor de 2 posiciones

Fuente: elaboración propia.

4.2.1.3. Esquemático

El esquemático es el diseño de las conexiones de los componentes que utiliza el nodo y el *Gateway*.

Figura 42. **Esquemático del nodo LoRa**



Fuente: elaboración propia, empleando EAGLE 7.6.0.

El circuito consta de tres partes principales:

Alimentación de voltaje: esta sección está compuesta por un módulo TP4056(U2), este circuito permite conectarle una fuente de energía eléctrica (P1) a su entrada y una batería (BAT1) a su salida para que pueda cargarse de forma adecuada, en caso de emergencia el módulo TP4056 cuenta con un conector micro USB hembra que permite utilizar cualquier cargador de 5 V que provea al menos 300 mA como alimentación principal, se agrega una resistencia (R3) en configuración *pull-down* que se utiliza para medir el voltaje de la batería y como elemento final de esta parte se agrega un interruptor (SW1) de 2 posiciones encargado de encender y apagar el nodo.

Adquisición de datos: los componentes principales de esta etapa son los circuitos para obtener los datos de los sensores, el pluviómetro se conecta con la resistencia (R2) en configuración *pull-up*, adicionalmente se agrega un capacitor (C1) conectado a tierra debido que el *reed switch* que internamente tiene el pluviómetro genera transiciones falsas de apertura/ cierre cuando envía un pulso, debido a problemas mecánicos y físicos, estas transiciones pueden leerse como múltiples pulsos en un muy corto tiempo engañando al programa. Por esa razón el capacitor tiene como objetivo eliminar el rebote del *reed switch* y como elemento final de esta parte se tiene un circuito divisor de voltaje para convertir los diferentes valores de resistencia del sensor *eTape* a voltaje que puede ser medido por el ADC del microprocesador.

Control y configuración: en esta sección está la tarjeta de desarrollo que incluye el microcontrolador y el módulo LoRa, el centro del nodo. el módulo (U1) es el que se encarga de establecer al enlace punto a punto por medio de LoRa, este módulo se comunica por medio de SPI con el microcontrolador, la *blue pill* (U1) es el cerebro del nodo, envía las instrucciones al *Gateway* y recibe instrucciones del *Gateway*. Otro módulo importante es el de la pantalla, esta se comunica por medio de I2 C con el microcontrolador permitiendo desplegar los parámetros más importantes del nodo.

4.2.1.4. Esquemático del *Gateway*

El circuito consta de 2 partes principales.

En la figura 43, se expone el esquemático del *Gateway*.

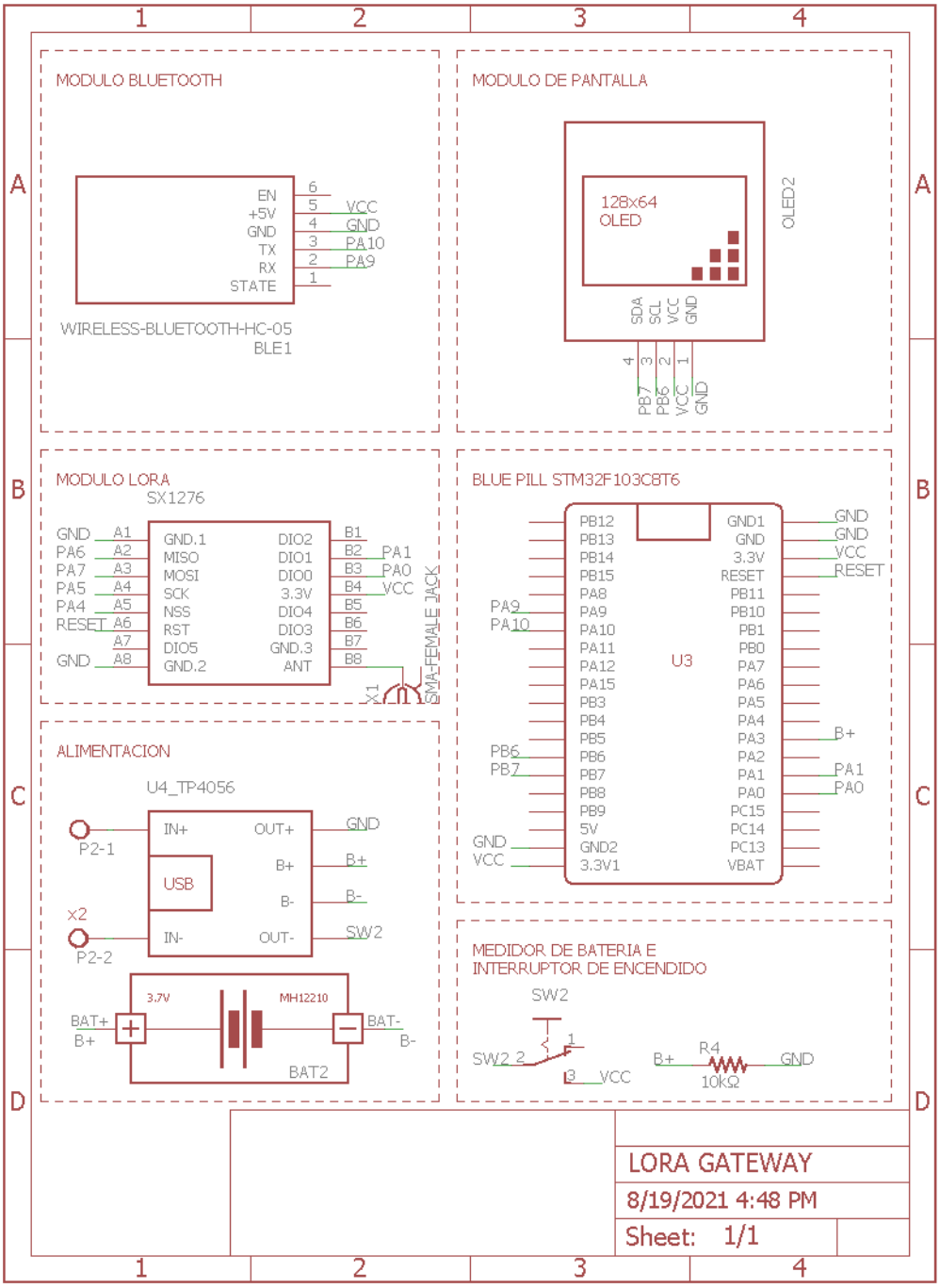
Alimentación de voltaje: esta sección está compuesta por un módulo TP4056(U4), este circuito permite conectarle una fuente de energía eléctrica (P2) a su entrada y una batería (BAT2) a su salida para que pueda cargarse de forma adecuada, en caso de emergencia el módulo TP4056 cuenta con un conector micro USB hembra que permite utilizar cualquier cargador de 5 V que provea al menos 300 mA como alimentación principal, se agrega una resistencia (R4) en configuración *pull-down* que se utiliza para medir el voltaje de la batería y como elemento final de esta parte se agrega un interruptor (SW2) de 2 posiciones encargado de encender y apagar el *Gateway.*

Control y configuración: en esta sección está la tarjeta de desarrollo que incluye el microcontrolador y el módulo LoRa, el centro del nodo.

El módulo es el que se encarga de establecer al enlace punto a punto por medio de LoRa, este módulo se comunica por medio de SPI al microcontrolador, la *blue pill* (U3) es el cerebro del *Gateway*, él recibe las instrucciones del nodo y él envía instrucciones al nodo.

Otro módulo importante es el de comunicación *bluetooth*, este se conecta por medio de comunicación serial al microcontrolador permitiendo enviar instrucciones a la aplicación Android, y como elemento final de esta parte se agrega el módulo de pantalla, este se comunica por medio de I2C con el microcontrolador permitiendo desplegar los parámetros más importantes del enlace punto a punto de la tecnología LoRa.

Figura 43. **Esquemático del *Gateway***



Fuente: elaboración propia, empleando EAGLE 7.6.0.

4.2.2. Diseño del nodo

En la siguiente sección se describen los pasos para realizar el diseño del nodo.

4.2.2.1. Diagrama de flujo

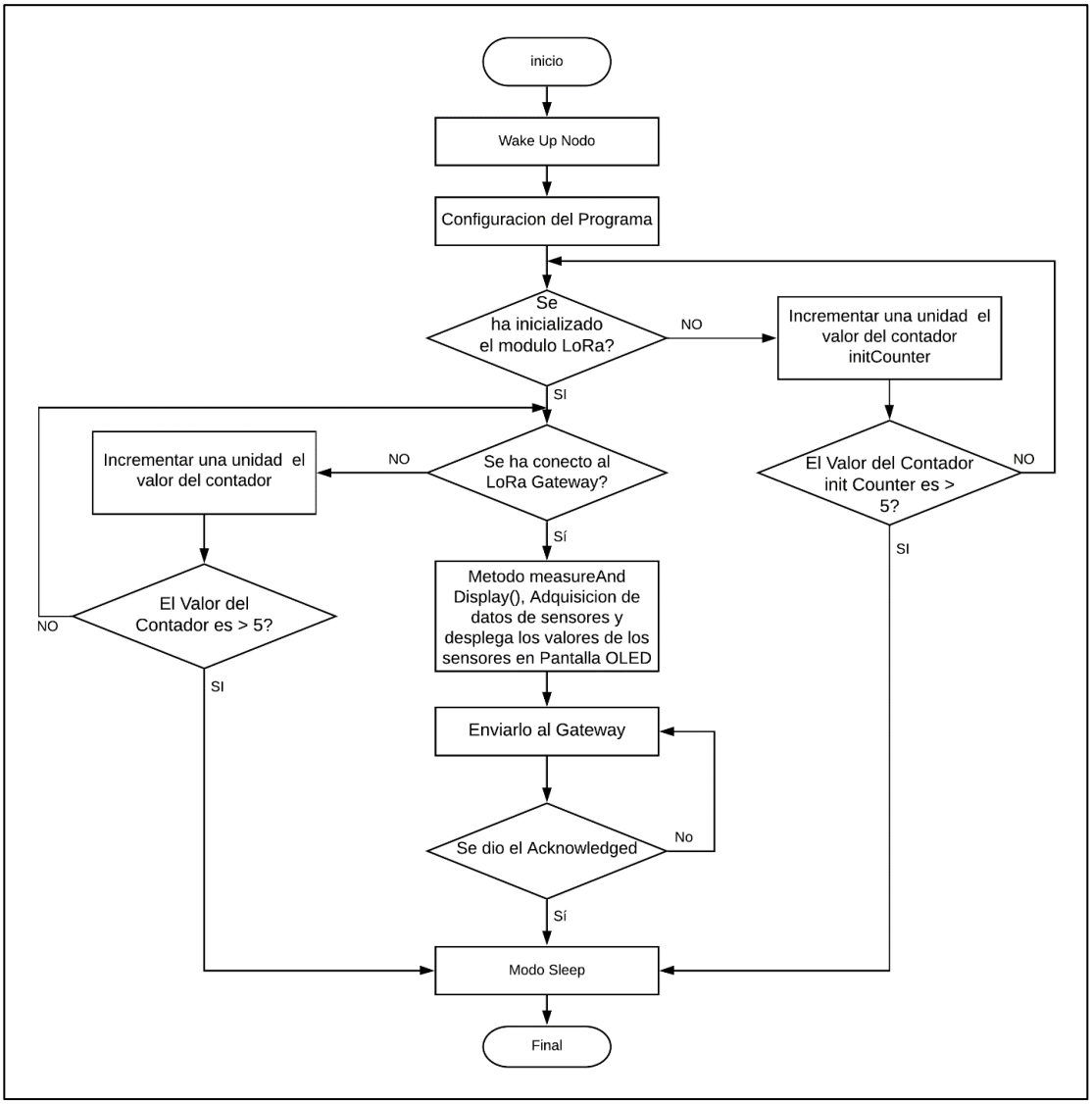
El diagrama de flujo es una representación gráfica del algoritmo del código fuente del nodo, la lógica de la comunicación punto a punto se explica en esta sección para una mejor comprensión de la programación.

Las funciones principales del programa son:

- Configuración del programa.
- Inicialización del módulo LoRa.
- Adquisición de datos de los sensores.
- Visualización de datos.
- Envío de trama de datos por medio del protocolo LoRa.

El diagrama de flujo siguiente resume la programación del nodo:

Figura 44. **Diagrama de flujo nodo**



Fuente: elaboración propia, empleando Visio 2019.

4.2.2.2. Algoritmo de medición de precipitación

El volumen de agua de cada compartimiento corresponde a la altura de una columna de agua lluvia recibida por precipitación y determinada por la siguiente ecuación.

$$Lámina = \frac{V[cm^3] * 10}{\pi \left(\frac{d\ [cm]}{2}\right)^2}\ [mm]$$

Donde:

V = volumen de cada uno de los compartimientos medido en cm^3

d = diámetro mayor del embudo del pluviómetro medido en cm

> El pluviómetro a utilizar en este diseño es fabricado por *Argent Data System* y según el fabricante cada volteo en el balancín representa 0,2794 mm de lluvia y un pequeño imán produce una señal eléctrica que es detectada por un sensor llamado *reed switch*, y puede ser detectada por una entrada de interrupción de microcontrolador.[21]

El *reed switch* es un interruptor eléctrico activado por un campo magnético, los contactos están normalmente abiertos se cierran en la presencia de un campo magnético, cuenta con 2 terminales y está conectado a los dos conductores centrales de un cable con terminación RJ11.

4.2.2.3. Algoritmo de medición de evapotranspiración

Las entradas analógicas son usadas por el comando *analog Read,* con el cual se podrá capturar los valores de diferentes sensores.

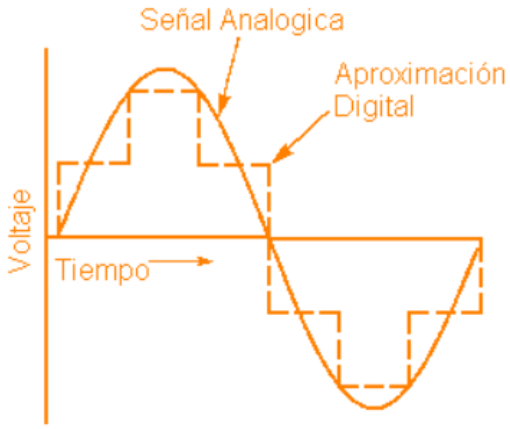
---

[21] SPARKFUN. *Electronics.* https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly.pdf. Consulta: 8 de enero de 2020.

Todas las señales físicas que captamos con nuestros sentidos, al igual que toda variable que existe en el entorno (temperatura, velocidad, peso, caudal, entre otras) son señales analógicas, es decir señales que varían con el tiempo, y que pueden tomar diferentes valores a diferencia de la señal digital que solo presenta dos valores.

En la figura 42 se puede ver una señal analógica, que representa una función sinusoidal junto con una aproximación discreta.

Figura 45. **Conversión analógica – digital (ADC) de una señal senoidal**



Fuente: CAE. *Conversión Analógica*. https://controlautomaticoeducacion.com. Consulta: 8 de enero de 2021.

La *blue pill* tiene la capacidad de leer señales analógicas realizando aproximaciones discretas a través de pequeños rectángulos digitales como fue observado en la figura anterior, donde se puede ver fácilmente, que entre más pequeños sean los rectángulos, más parecida será la aproximación digital con relación a la señal analógica.

4.2.2.3.1. ADC blue pill

Cuando se trabaja con el ADC (conversor análogo digital) se debe tener en consideración que solo se pueden colocar voltajes de 3,3 v, de lo contrario se puede quemar la placa.

A continuación, se muestra una tabla con los voltajes máximos de operación y la máxima resolución de la *blue pill*.

Tabla XIX. **Voltajes máximos de operación**

Placa	Voltaje operación	Pines	Max resolución
Stm32f103c8t6	3,3 V	PA0-PA7 PB0-PB1	12 bits

Fuente: elaboración propia.

Como se puede ver en la tabla anterior, la lectura que haga la *blue pill* en la entrada análoga va a depender de la resolución ADC que tenga. Por ejemplo, como es de 12 bits en realidad la *blue pill* va a ver una variación de un entero entre 0 a 4 095.

Un bit es un binario que puede tomar dos valores (0 o 1), o sea que si tiene una resolución de 12 bits indica que ($2^{12} = 4\,096$), pero como empezamos desde 0, se dice que el valor entero toma valores entre 0 a 4 095. Esto quiere decir que si la *blue pill* mide:

- El máximo voltaje (3,3 v) va a almacenar un valor entero de 4 095.
- Voltaje intermedio (1,65 v) va a almacenar un valor entero de 2 048.

- Voltaje mínimo (0v) va a almacenar un entero de 0.

A través de lo visto anteriormente podemos mostrar entonces la ecuación de la resolución del ADC

$$R = \frac{V_{ref}}{2^N - 1}$$

Donde:
R = resolución
V_{ref} = voltaje de referencia ADC de la Placa (por defecto es el voltaje de alimentación de la placa)
N = bits del convertidor ADC (análogo digital).

Es decir que para la *blue pill* la resolución es la siguiente:

$$R = \frac{3{,}3\ v}{2^{12} - 1} = \frac{3{,}3\ v}{4\ 095} = 0{,}8058\ mV$$

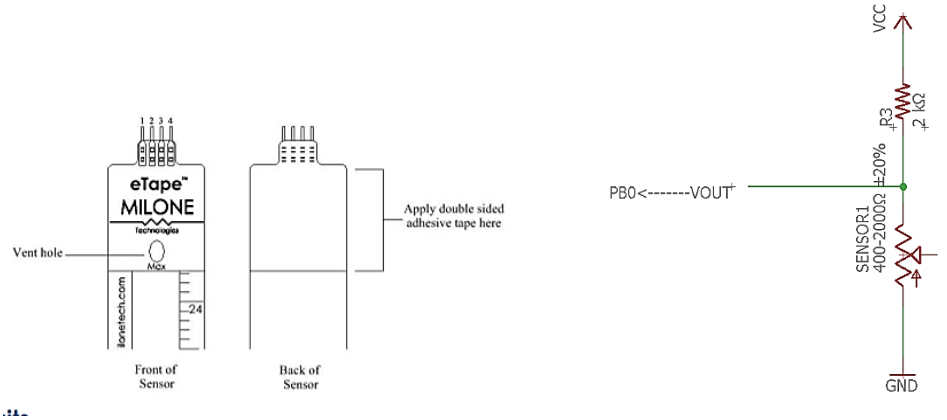
Eso quiere decir que, en el caso de un la *blue pill*, el valor de 0 V analógico es expresado en digital como B000000000000 (0) y el valor de 3,3 V analógico es B111111111111 (4 096). Por lo tanto, todo valor analógico intermedio es expresado con un valor entre 0 y 4 095, es decir, se suma 1 en binario cada 0,8058 mV.

4.2.2.3.2. Sensor a nivel *eTape*

La conexión del sensor de nivel (*eTape*) se realiza mediante un conector de 4 pines con cables presoldados a los pines *Crimpflex*. Las dos terminales internas (terminales 2 y 3) son la salida del sensor. Las terminales externas (terminales 1

y 4) son la resistencia de referencia que puede utilizarse para la compensación de temperatura. Se debe colocar el sensor en el fluido que se va a medir en este caso agua. Para que funcione correctamente, el sensor debe permanecer recto y no debe doblarse ni vertical ni longitudinalmente. Para obtener mejores resultados, Sin embargo, se debe permitir que el líquido interactúe libremente con ambos lados del sensor. El orificio de ventilación situado por encima de la línea máxima permite que el sensor de nivel se equilibre con la presión atmosférica. El orificio de ventilación está equipado con una membrana de filtro hidrofóbico para evitar que el sensor de nivel se inunde si se sumerge inadvertidamente.

Figura 46. **Sensor *eTape***



Fuente: ELECTRÓNICOS CALDAS. *Sensor (eTape)*: https://www.electronicoscaldas.com. Consulta: 12 de enero de 2020.

En este circuito, el voltaje es directamente proporcional a la resistencia del sensor de nivel.

$$V = \frac{Vcc \times R\ sensor}{[2K\Omega] + R\ sensor}$$

El sensor tiene menor resistencia en los niveles de líquido más altos y mayor resistencia en los niveles de líquido más bajos. Por lo tanto, la lectura digital máxima de voltaje corresponde a un tanque vacío y la lectura digital mínima de voltaje corresponde a un tanque lleno. La figura 46 muestra esta relación.

4.2.2.3.3. Calibración de cenirrómetro

El código será calibrado para reportar la lámina de líquido medido, este código leerá la resistencia real del sensor y le ayudará a calibrar para las mediciones de la lámina.

En la parte de la programación del nodo se puede ajustar los siguientes valores #define basados en el *hardware*.

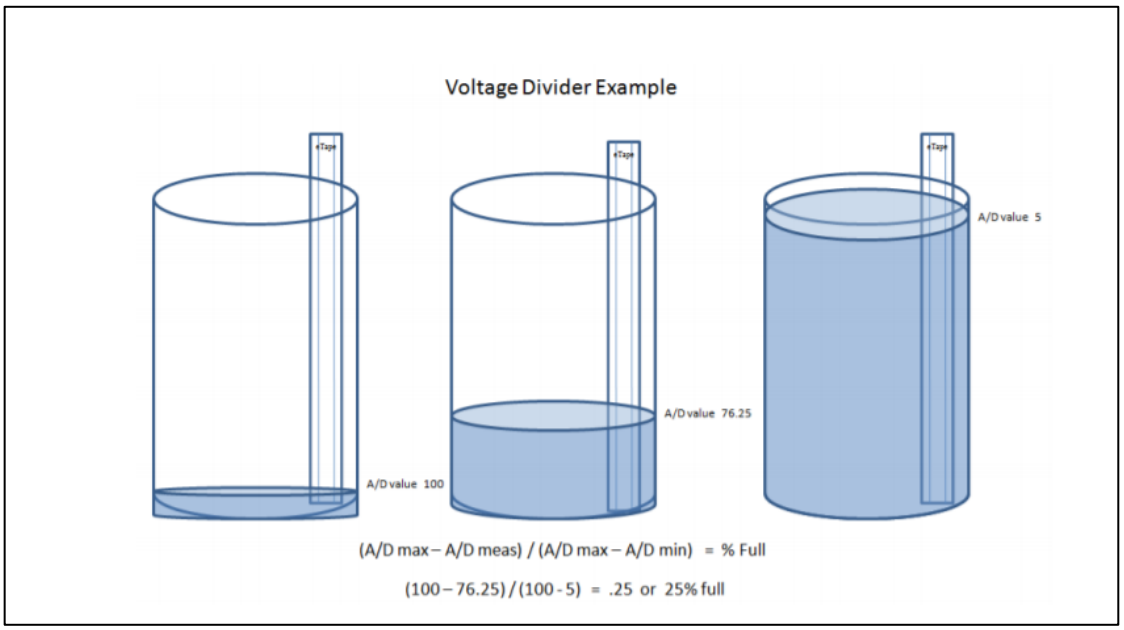
SERIES_RESISTOR - Este es el valor en ohmios de la resistencia que se ha conectado al *hardware* en serie con el sensor, en este caso 2 690 Ω.

SENSOR_PIN - Este es el pin de entrada analógica que está conectado al sensor.

Para calibrar el sensor para medir la lámina, se mide lo siguiente:

- El valor de la resistencia sin que el líquido toque el sensor.
- El valor de resistencia cuando una lámina conocida de líquido está tocando el sensor.
- La lámina del líquido utilizado para encontrar el valor de resistencia anterior.

Figura 47. **Calibración de cenirrómetro**



Fuente: CDN Shop. *eTape*. https://cdn-shop.adafruit.com/datasheets/*eTape*App.pdf-. Consulta: 8 de enero de 2020.

Una vez que haya determinado los valores de calibración anteriores, se debe actualizar los #defines en la parte superior de la programación del nodo apropiadamente:

ZERO_LAMINAMM_RESISTANCE – valor de resistencia cuando no hay agua presente en el cenirrómetro, en este caso 2 631,40 Ω.

CALIBRATION_RESISTANCE - valor de resistencia cuando el agua está al máximo en el cenirrómetro, en este caso 750,00 Ω

CALIBRATION_LAMINAMM – milímetros cuando el agua está al máximo en el cenirrómetro, en este caso 211,00 mm.

4.2.2.4. *Software* del nodo

En la siguiente sección se describe el conjunto de programas y rutinas que permiten al microcontrolador realizar determinadas tareas de configuración. Entre ellas la configuración del radio permitiendo la transmisión de datos al *Gateway*, la configuración de entradas digitales y análogas permitiendo la adquisición de datos y configuración de periféricos de salida permitiendo la visualización de los datos.

4.2.2.4.1. Configuración del entorno de desarrollo

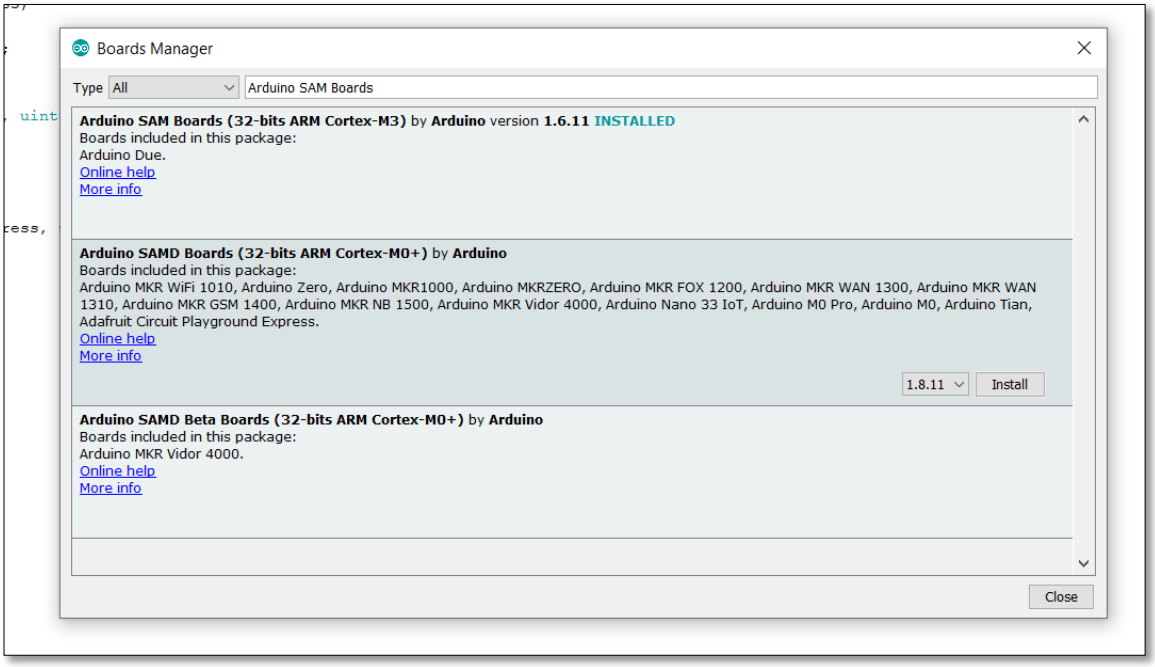
El microcontrolador *Blue Pill* se puede programar mediante STM32 Cube IDE, Eclipse, PlatformIO, *Visual Studio Code*, código de Arduino, que es una variante de C. Se ha seleccionado el código de Arduino para su programación por ser el código para el que existe mayor documentación y el más sencillo de interpretar por una persona que lea este documento.

Antes de comenzar a programar el código fuente del nodo, es necesario configurar el entorno de desarrollo integrado de Arduino, pues originalmente este no ha sido diseñado para la configuración del microcontrolador *blue pill*. Para esto es necesario contar con la versión 1.6.4 o superior del IDE, este se puede descargar desde la página oficial de Arduino https://www.arduino.cc.

Después de descargar e instalar Arduino IDE, es necesario instalar Arduino SAM Boards (ARM Cortex-M3 de 32 bits) de Arduino versión 1.6.11 desde:

Tools -> Boards -> Boards Manager en Arduino IDE.

Figura 48. **Gestor de tarjetas del IDE de Arduino**



Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Posteriormente, para que el microcontrolador reconozca los módulos basados en el núcleo de Roger Clark, se debe descargar el núcleo desde el repositorio oficial https://github.com/rogerclarkmelbourne/Arduino_STM32, se descomprime en el directorio de *hardware* de Arduino IDE. Al reiniciar el IDE de Arduino se habrá agregado, en el menú herramientas.

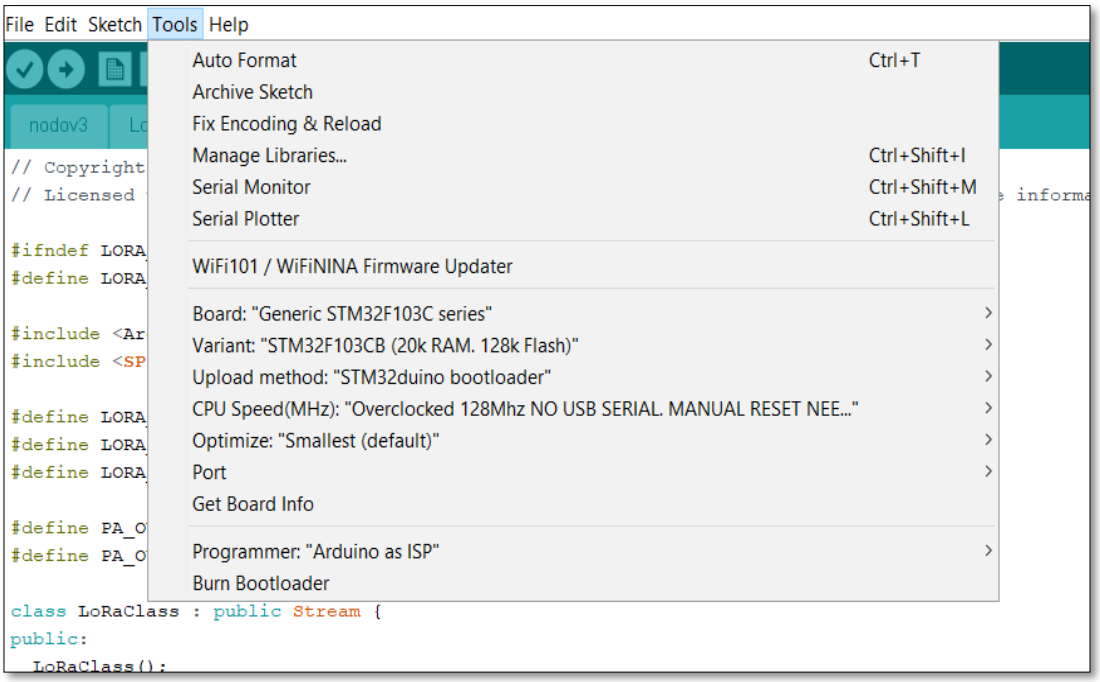
Hay varias cosas que deben suceder para que cualquier placa de microcontrolador, incluida la STM32duino, sea compatible con Arduino IDE.

El núcleo de Arduino en sí debe estar precargado de alguna manera en el microcontrolador. La *blue pill* cuenta con la ventaja que tiene Stm32duino

preinstalado. Por lo tanto, puede usar Arduino IDE para actualizar el programa de la misma manera que un Arduino.

Después de la instalación de la tarjeta, se pueden configurar los demás parámetros. Desde el menú de herramientas, en la sección tarjetas, se selecciona *Generic* STM32F103C *serie*, en la sección variación, se selecciona STM32F103CB (20k RAM. 128k Flash), en la sección método de carga, se selecciona STM32duino *bootloader* y en la sección programador, se selecciona Arduino as ISP, habiendo realizado las configuraciones mencionadas en esta sección, se puede iniciar a programar.

Figura 49. **Configuración de parámetros de tarjeta *blue pill***



Fuente: elaboración propia, empleando Arduino ide 1.8.15.

4.2.2.4.2. Programación del nodo

Para programar el nodo con una computadora es necesario conectar el microcontrolador por medio de un puerto USB. Una vez conectado se puede conectar a la computadora y utilizar el IDE de Arduino como si de un dispositivo Arduino se tratará. A continuación, se detallará el código fuente que se ha utilizado para programar el nodo. Este código se puede encontrar completo en el apéndice 1.

En primer lugar, es necesario importar la librería *armtronix's Modified lib of sandeepmistry arduino*-LoRa for STM32F103 de forma local, ya que esta librería no es para la *blue pill* (STM32f103c8t6) sino para STM32f103CB entonces se debe modificar 2 archivos de la librería, en el primer archivo llamado LoRa_STM32.h se modifica los pines de la interfaz SPI.

El archivo LoRa_STM32.h contiene la configuración de la librería, pero lo único que se debe modificar en este archivo es la configuración de los pines de la comunicación SPI entre la *blue pill* y el módulo LoRa, todas las configuraciones adicionales se mantienen sin cambio, disponible completo en el apéndice 3.

Figura 50. **Código fuente del nodo, segmento 1**

```
10  #define LORA_DEFAULT_SS_PIN    PA4
11  #define LORA_DEFAULT_RESET_PIN 0xFF
12  #define LORA_DEFAULT_DIO0_PIN  PA0
13
14  #define PA_OUTPUT_RFO_PIN      0
15  #define PA_OUTPUT_PA_BOOST_PIN 1
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

El archivo LoRa_STM32.cpp contiene una colección de funciones de la comunicación LoRa que pueden ser llamadas en el programa, disponible completa en el apéndice 2.

En LoRa_STM32.cpp se cambia #include <LoRa_STM32.h> a #include "LoRa_STM32.h" porque LoRa_STM32.h ahora es una librería local.

Figura 51. **Código fuente del nodo, segmento 2**

```
1   #include "LoRa_STM32.h"
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

En segundo lugar, es necesario agregar las bibliotecas que se utilizarán.

Figura 52. **Código fuente del nodo, segmento 3**

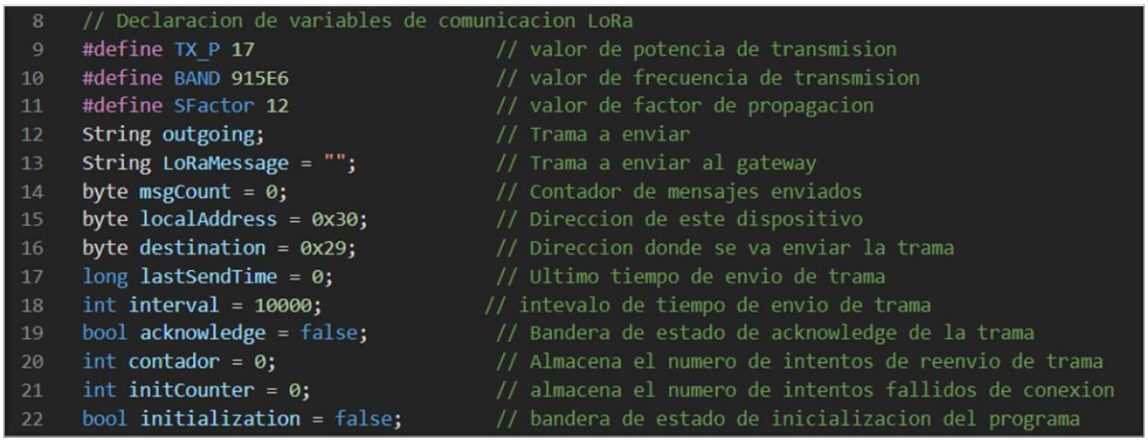
```
1   //Librerias
2   #include <SPI.h>
3   #include "LoRa_STM32.h"
4   #include <Wire.h>
5   #include <Adafruit_GFX.h>
6   #include <Adafruit_SSD1306.h>
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La biblioteca SPI le permite comunicarse con dispositivos SPI, con la *blue pill* como dispositivo maestro, mientras LoRa_STM32.h es una colección de funciones de la comunicación LoRa que pueden ser llamadas en el programa, wire.h le permite comunicarse con dispositivos I2C, adafruit_gfx.h proporciona

una sintaxis común y un conjunto de funciones gráficas para la pantalla OLED, Adafruit_SSD1306.h esta librería es la específica para las pantallas OLED basadas en el controlador SSD1306.

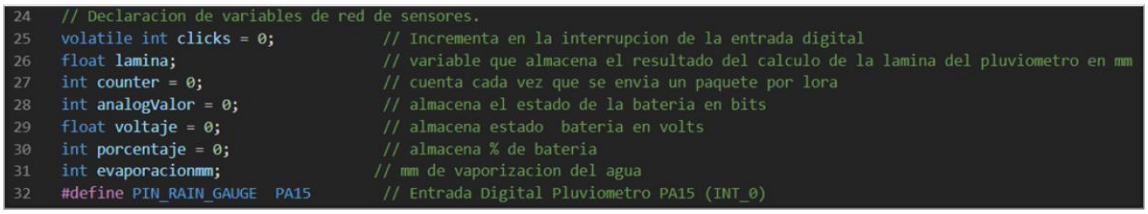
Figura 53. **Código del nodo, segmento 4**



```
// Declaracion de variables de comunicacion LoRa
#define TX_P 17                     // valor de potencia de transmision
#define BAND 915E6                  // valor de frecuencia de transmision
#define SFactor 12                  // valor de factor de propagacion
String outgoing;                    // Trama a enviar
String LoRaMessage = "";            // Trama a enviar al gateway
byte msgCount = 0;                  // Contador de mensajes enviados
byte localAddress = 0x30;           // Direccion de este dispositivo
byte destination = 0x29;            // Direccion donde se va enviar la trama
long lastSendTime = 0;              // Ultimo tiempo de envio de trama
int interval = 10000;              // intevalo de tiempo de envio de trama
bool acknowledge = false;           // Bandera de estado de acknowledge de la trama
int contador = 0;                   // Almacena el numero de intentos de reenvio de trama
int initCounter = 0;                // almacena el numero de intentos fallidos de conexion
bool initialization = false;        // bandera de estado de inicializacion del programa
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Figura 54. **Código del nodo, segmento 5**



```
// Declaracion de variables de red de sensores.
volatile int clicks = 0;            // Incrementa en la interrupcion de la entrada digital
float lamina;                       // variable que almacena el resultado del calculo de la lamina del pluviometro en mm
int counter = 0;                    // cuenta cada vez que se envia un paquete por lora
int analogValor = 0;                // almacena el estado de la bateria en bits
float voltaje = 0;                  // almacena estado  bateria en volts
int porcentaje = 0;                 // almacena % de bateria
int evaporacionmm;                 // mm de vaporizacion del agua
#define PIN_RAIN_GAUGE  PA15        // Entrada Digital Pluviometro PA15 (INT_0)
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Figura 55. **Código del nodo, segmento 6**

```
34  // Configuracion de Valores de Evaporimetro
35  #define SERIES_RESISTOR             2690      // Valor de la resistencia en serie en ohms.
36  #define SENSOR_PIN          PA2               // Pin analogico el cual esta conectado al sensor.
37  #define ZERO_LAMINAMM_RESISTANCE    2631.40   // Valor de resistencia (en ohms) cuando no hay agua presente en el Cenirrometro.
38  #define CALIBRATION_RESISTANCE      750.00    // Valor de resistencia (en ohms) cuando el agua esta al maximo en el Cenirrometro.
39  #define CALIBRATION_LAMINAMM        211.00    // milimetros cuando el agua esta al maximo en el Cenirrometro.
40  // Configuracion de Pantalla OLED
41  #define SCREEN_WIDTH                128       // OLED display ancho, en pixeles
42  #define SCREEN_HEIGHT               64        // OLED display alto, en pixeles
43  //Declaracion de pantalla SSD1306 conectado mediante I2C (pines SDA, SCL)
44  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Después de la inclusión de bibliotecas y librerías locales, se colocan las definiciones, objetos y variables globales. Las definiciones serán valores que no cambiarán en el tiempo y se traducen al momento de la compilación, no se guardan como variables. Los objetos y variables globales son entidades a las que puede tener acceso cualquier función del código fuente.

Cualquier programación en el IDE de Arduino se compone de dos funciones principales, una función de configuración (*setup*) y otra de ejecución periódica (*loop*). La primera solamente se ejecuta una vez al inicio y la segunda se ejecuta de forma cíclica indeterminadamente. Dentro de la función de configuración se ha agregado la parte de configuración inicial que debe realizar el nodo.

Figura 56. **Código del nodo, segmento 7**

```
46  void setup() {
47
48   //Inicializacion de SSD1306
49  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {              // Direccion 0x3C para 128x64
50          SSD1306errormsg();                                     // mostrar alerta
51           for (;;);
52     }
53    // configuracion de interrupcion de pluviometro
54    pinMode(PIN_RAIN_GAUGE, INPUT_PULLUP);                      // asignacion de pin de entrada
55    attachInterrupt(PIN_RAIN_GAUGE, countRainGauge, FALLING);   // configuracion de interrupcion
56
57   // configuracion de ADC de medidor de bateria
58
59    pinMode(PA3, INPUT_PULLDOWN);                               // configuracion de pull-down para estado de bateria
60
61   // metodo de inicializacion de enlace punto a punto
62    initSX1276();
63    }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Lo primero que se configura es la inicialización de la pantalla OLED en la dirección 0x3C, sino es exitosa llama a la función SSD1306errormsg().

Figura 57. **Código del nodo, segmento 8**

```
286  void SSD1306errormsg(){
287          display.clearDisplay();                                // limpiar buffer
288          display.setTextSize(1);                                // tamaño del texto
289          display.setCursor(0, 32);                              //posicion del texto
290          display.print("Fallo en la Asignacion de SSD1306");
291          display.display();                                     // enviar a pantalla
292          delay(2000);                                           // tiempo de visualizacion
293    }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Cuando llama a la función SSD1306errormsg(), lo primero que ejecuta es la limpieza del *buffer* para evitar sobrescribir sobre el texto visualizado previamente a continuación, ajusta el tamaño del texto y su posición en coordenadas cartesianas posterior, despliega el mensaje que fallo la asignación de la pantalla

SSD1306 y por último, se visualiza el mensaje en un lapso de tiempo de 2 segundos.

Regresando al archivo del código fuente del nodo, se configura la interrupción empleada por el pluviómetro, lo primero es asignar el nombre *PIN_RAIN_GAUGE* al pin de entrada digital y adicionalmente se configura la resistencia interna *pull up*, luego se configura el *CountRainGauge* que es el ISR a llamar cuando ocurre la interrupción, por último, se define cuándo debe activarse la interrupción, *FALLING* en este caso que es cuando el pin pasa de un estado alto a bajo.

Figura 58. **Código del nodo, segmento 9**

```
175  void countRainGauge() {                          //metodo de interrupcion del pluviometro
176    clicks++;
177  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Cuando llama al ISR *countRainGauge*() la variable *clicks* que inicialmente tiene el valor de 0 incrementa su valor a 1 y así sucesivamente cada que se le interrumpe.

Regresando al archivo fuente del nodo, por último, se llama a la función initSX1276() que inicializa el enlace LoRa punto a punto.

Figura 59. **Código del nodo, segmento 10**

```
void initSX1276(){
  LoRa.setTxPower(TX_P);            // Configuracion de potencia de transmision
  LoRa.setSpreadingFactor(SFactor); // Configuracion de factor de propagacion
  pinMode(PC13, OUTPUT);            // indicador de envio de trama

      while (initCounter < 5) {      // Contador de intentos de reconexion
  if (!LoRa.begin(BAND)) {           // Configuracion de Frecuencia
          initfailed();              // mostrar alerta
          initCounter++;
          delay(5000);               // intervalo de tiempo de reintentos
  }

  else {
          initsucceeded();           // mensaje de inicializacion correcta
          initialization = true;     // bandera de inicializacion correcta
          break;
  }
}
  if (initialization == false) {     // bandera de inicializacion incorrecta
    sleepmsg();                      // mensaje de inicio de modo ahorro de energia
    LoRa.sleep();                    // se activa modo sleep

  }
}
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Lo primero que se configura es la potencia de transmisión, la cual tiene un rango entre 2 y 17 dB, su valor por defecto es de 17 dB que es la máxima potencia de transmisión, por lo tanto, TX_P tiene valor de 17 dB. Luego se configura el factor de ensanchamiento, el cual tiene un rango entre 6 y 12, su valor por defecto es de 7. S*Factor* tiene valor de 12. Posterior se configura el pin PC13 como salida para indicar el envío de una trama de datos. Por último, se configura la frecuencia de transmisión, para Europa se utiliza 868Mhz, Asia 433 Mhz y para América 915 Mhz, por lo tanto, *BAND* tiene el valor de 915E6.

El diagrama de flujo muestra que se debe validar si la inicialización del enlace punto a punto fue exitosa, si lo es entonces *initialization* que es una

variable tipo binaria (*bool*) funciona como una bandera indicando que la inicialización fue exitosa, por lo tanto, permite continuar con el programa. en caso no es exitoso debe realizar 5 intentos, luego sino logra inicializar en 5 intentos entonces ingresa en modo *sleep* el chip LoRa por 15 minutos y luego vuelve intentar inicializar de nuevo.

Figura 60. **Código del nodo, segmento 11**

```
void loop() {

  if (initialization == true){        // bandera de inicializacion exitosa
  while (contador < 5) {              // intenta comunicarse con el gateway 5 veces
    if (millis() - lastSendTime > interval) {
      measureAndDisplay();            // metodo de medicion de sensores y visualizacion
      sendMessage(LoRaMessage);       //Envio de trama  de datos al receptor
      lastSendTime = millis();        // marca de tiempo de envio de mensaje
      interval = 10000;               // 10 segundos entre reintentos de envio de trama
      contador++;
    }
    else {

      if (contador > 0) {
      onReceive(LoRa.parsePacket());  // analizar un paquete y llamar a onReceive con el resultado
      }
    }

    if (acknowledge == true) {        // Llamar metodo sleep
     contador = 0;
      break;
      }
  }
  }

  else{
    sleepmsg2();                      // mostrar alerta
    delay(900000);                    // espera de 15 minutos
    initCounter = 0;
    initSX1276();
  }


  if (acknowledge == true || contador > 4) {
    sleepAndRestart();                // Ingresa a metodo si recibio acknowledge o contador mayor a 4
  }
}
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Dentro de la función de ejecución aparece la condición si anidada donde se evalúa varios escenarios, primero se verifica si la inicialización fue exitosa, en caso contrario ingresa en modo *sleep* durante 15 minutos, luego reinicia el valor del contador de inicialización y vuelve ingresar a la función de inicialización del enlace LoRa punto a punto para intentar inicializar nuevamente, si la inicialización fue exitosa hay 2 posibles escenarios, el primero sucede cuando el nodo no logra enviar la trama de datos, intenta reenviarlo 5 veces, sino lo logra entonces entra en modo *sleep* por 15 minutos. El segundo escenario sucede cuando se envía la trama de datos exitosamente, entonces procede a llamar a las funciones *measureAndDisplay()* que se encarga de la adquisición de los datos de los sensores y también procede a desplegar los valores actuales de los sensores en la pantalla OLED en tiempo real, luego envía la trama de datos al *Gateway* por medio del enlace LoRa punto a punto y posteriormente se pone el nodo en modo escucha y espera que el *Gateway* le dé *acknowledge* de la trama enviada, si recibe algún paquete el nodo entonces procede a llamar a la función *onReceive* para analizarlo. Por último, si el *Gateway* le da *acknowledge* exitosamente o el contador es mayor a 4 el nodo entra en modo *sleep* por 15 minutos y al despertar vuele a repetir la función de ejecución perpetuamente.

Figura 61. **Código del nodo, segmento 12**

```
104   void sendMessage(String outgoing) {
105     LoRa.beginPacket();                   // paquete inicial
106     LoRa.write(destination);              // incluir la dirección de destino
107     LoRa.write(localAddress);             // incluir la dirección del emisor
108     LoRa.write(msgCount);                 // incluir el ID de la trama
109     LoRa.write(outgoing.length());        // incluir la longitud de la trama
110     LoRa.print(outgoing);                 // incluir trama
111     LoRa.endPacket();                     // terminar el paquete y enviarlo
112     msgCount++;                           // incrementar el ID del mensaje
113   }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *sendMessage*(), forma una cadena de texto, llamada *outgoing,* la cadena de texto es también conocida como trama y se establece de la siguiente manera.

Tabla XX. **La función *sendMessage*()**

Dir. Destino	Dir. Local Nodo	ID MSG	Long. Trama	*Outgoing*

Fuente: elaboración propia.

La variable *outgoing* de tipo *string* es un conjunto ordenado caracteres, por lo tanto, envía la adquisición de datos de los sensores por medio de esta variable y se compone de la siguiente manera.

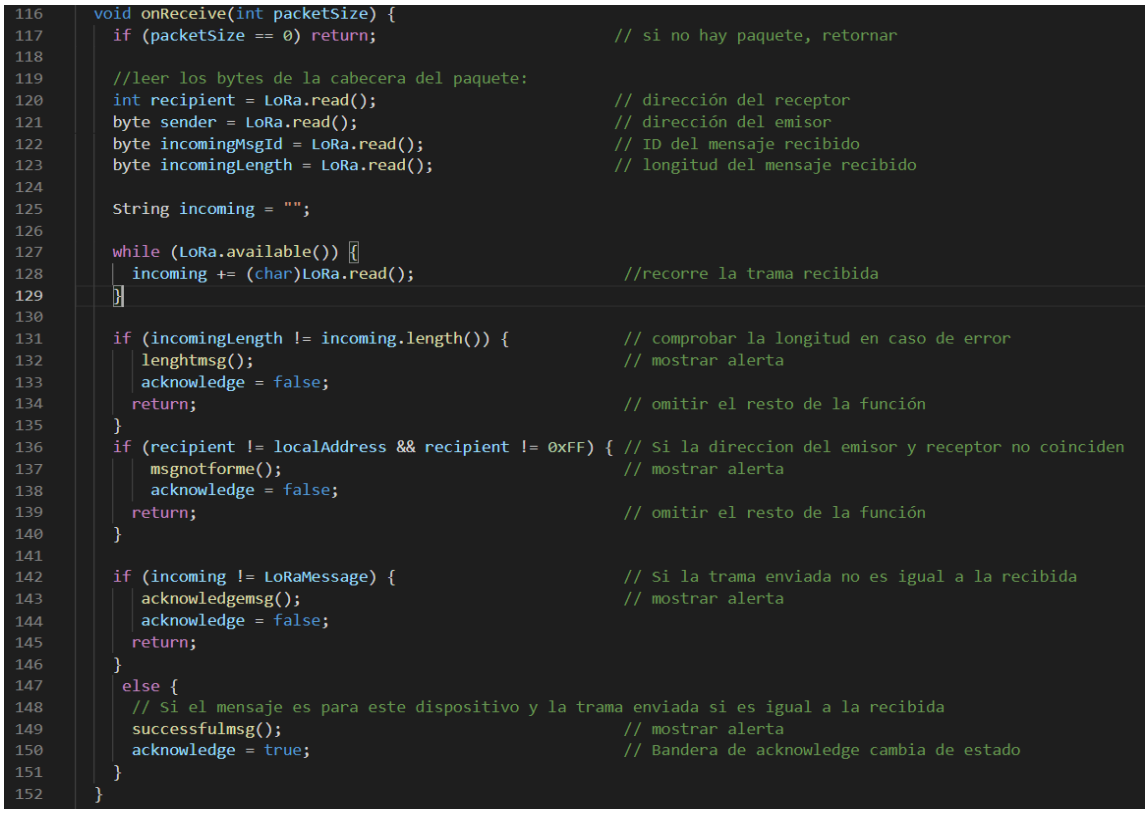
Tabla XXI. **La variable *outgoing***

Clicks	/	Lámina Pluviómetro	&	Resistencia Cenirrómetro	#	Lámina Cenirrómetro	@	Porcentaje de batería

Fuente: elaboración propia.

Los caracteres intermedios a cada variable sirven para indicarle al *Gateway* donde empieza y donde termina el dato de cada sensor.

Figura 62. **Código del nodo, segmento 13**



```
void onReceive(int packetSize) {
  if (packetSize == 0) return;                          // si no hay paquete, retornar

  //leer los bytes de la cabecera del paquete:
  int recipient = LoRa.read();                          // dirección del receptor
  byte sender = LoRa.read();                            // dirección del emisor
  byte incomingMsgId = LoRa.read();                     // ID del mensaje recibido
  byte incomingLength = LoRa.read();                    // longitud del mensaje recibido

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();                       //recorre la trama recibida
  }

  if (incomingLength != incoming.length()) {            // comprobar la longitud en caso de error
     lenghtmsg();                                       // mostrar alerta
     acknowledge = false;
    return;                                             // omitir el resto de la función
  }
  if (recipient != localAddress && recipient != 0xFF) { // Si la direccion del emisor y receptor no coinciden
      msgnotforme();                                    // mostrar alerta
      acknowledge = false;
    return;                                             // omitir el resto de la función
  }

  if (incoming != LoRaMessage) {                        // Si la trama enviada no es igual a la recibida
     acknowledgemsg();                                  // mostrar alerta
     acknowledge = false;
    return;
  }
   else {
    // Si el mensaje es para este dispositivo y la trama enviada si es igual a la recibida
    successfulmsg();                                    // mostrar alerta
    acknowledge = true;                                 // Bandera de acknowledge cambia de estado
  }
}
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *onReceive (int packetSize)*, es la encargada de realizar una serie de pruebas para comprobar que la trama haya sido enviada correctamente, Primero verifica si ha recibido un paquete, si lo recibe entonces procede a leer los bytes de la cabecera del paquete, luego recorre la trama recibida y valida la longitud de la trama sino coincide la longitud de la trama enviada con la recibida entonces procede a llamar a la función *lenghmsg*() encargada de mostrar la alerta en la pantalla OLED, después de eso valida si la dirección del nodo y del *Gateway* coinciden, sino coinciden manda a llamar a la función *msgnotforme*() encargada de mostrar la alerta en la pantalla OLED, posterior a esto hace una última

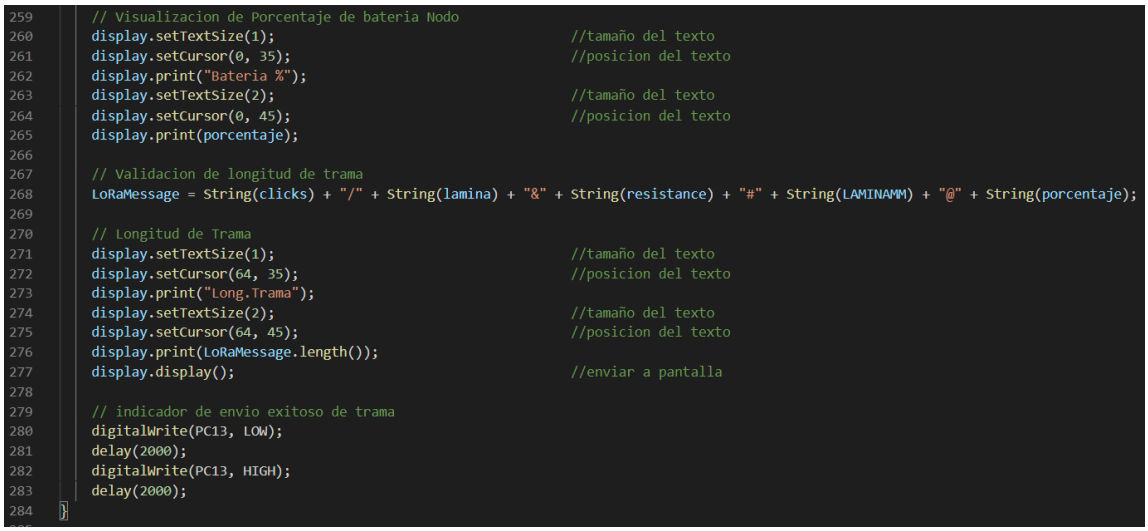
validación verifica si la trama enviada por el nodo al Gateway es igual a la trama enviada por el *Gateway* al nodo, sino son iguales, manda a llamar a la función *acknowledgemsg*() encargada de mostrar la alerta en la pantalla OLED sin embargo, si las tramas son iguales procede a cambiar el estado de la variable que funciona como bandera *acknowledge* de tipo binario(*bool*) a true y también procede a llamar a la función *successfulmsg*() encargada de mostrar la alerta en la pantalla OLED.

Figura 63. **Código del nodo, segmento 14**

```
224  void measureAndDisplay() {
225    // Medicion de resistencia del sensor.
226    float resistance = readResistance(SENSOR_PIN, SERIES_RESISTOR);
227    // Map resistencia a LAMINAMM.
228    float LAMINAMM = resistanceToLA MINAMM(resistance, ZERO_LAMINAMM_RESISTANCE, CALIBRATION_RESISTANCE, CALIBRATION_LAMINAMM);
229    evaporacionmm = int(LAMINAMM);
230    delay(1000);
231    lamina = clicks * 0.2794;              //factor de conversion como lo indica el datasheet del pluviometro
232    analogValor = analogRead(PA3);         //ADC estado de bateria
233    voltaje = 0.001025390 * analogValor;   //voltaje actual de bateria
234    porcentaje = (voltaje - 3.2) * 100;    //porcentaje actual de bateria
235
236    // Visualizacion de Lamina de Pluviometro
237    display.clearDisplay();                                  // limpiar buffer
238    display.setTextSize(1);                                  //tamaño del texto
239    display.setCursor(0, 0);                                 //posicion del texto
240    display.print("Lluvia");
241
242    display.setTextSize(2);                                  //tamaño del texto
243    display.setCursor(0, 10);                                //posicion del texto
244    display.print(lamina);
245    display.setTextSize(1);                                  //tamaño del texto
246    display.print("mm");
247
248    // Visualizacion de Lamina de Cenirrometro
249    display.setTextSize(1);                                   //tamaño del texto
250    display.setCursor(60, 0);                                 //posicion del texto
251    display.print("Evaporacion");
252
253    display.setTextSize(2);                                   //tamaño del texto
254    display.setCursor(64, 10);                                //posicion del texto
255    display.print(evaporacionmm);
256    display.setTextSize(1);                                   //tamaño del texto
257    display.print("mm");
258
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Figura 64. **Código del nodo, segmento 15**



```
259     // Visualizacion de Porcentaje de bateria Nodo
260     display.setTextSize(1);                                 //tamaño del texto
261     display.setCursor(0, 35);                               //posicion del texto
262     display.print("Bateria %");
263     display.setTextSize(2);                                 //tamaño del texto
264     display.setCursor(0, 45);                               //posicion del texto
265     display.print(porcentaje);
266
267     // Validacion de longitud de trama
268     LoRaMessage = String(clicks) + "/" + String(lamina) + "&" + String(resistance) + "#" + String(LAMINAMM) + "@" + String(porcentaje);
269
270     // Longitud de Trama
271     display.setTextSize(1);                                 //tamaño del texto
272     display.setCursor(64, 35);                              //posicion del texto
273     display.print("Long.Trama");
274     display.setTextSize(2);                                 //tamaño del texto
275     display.setCursor(64, 45);                              //posicion del texto
276     display.print(LoRaMessage.length());
277     display.display();                                      //enviar a pantalla
278
279     // indicador de envio exitoso de trama
280     digitalWrite(PC13, LOW);
281     delay(2000);
282     digitalWrite(PC13, HIGH);
283     delay(2000);
284 }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *measureAndDisplay*() realiza la adquisición de los datos de la red de sensores y luego procede a desplegar los datos más importantes para el usuario en la pantalla OLED.

Primero declara una variable local tipo flotante para almacenar el valor real de la resistencia del sensor de nivel, esto se logra por intermedio de un divisor de voltaje entre el *SENSOR_PIN y SERIES_ RESISTOR*. Luego con ayuda de la función *map*() de Arduino que permite transformar un valor de un rango de entrada de resistencia al valor correspondiente a otro rango de salida en este caso lámina en mm del cenirrómetro.

Después procede a realizar una conversión entre número de volteos del pluviómetro por un factor de conversión como lo indica la hoja de datos del pluviómetro 0,2794 para obtener milímetros de precipitación.

Posteriormente, es necesario poder medir el voltaje remanente en la batería del nodo, la variable voltaje almacena el voltaje actual de la batería, sabiendo este valor la variable porcentaje de tipo entero procede a indicar el estado de la batería en porcentaje en un intervalo de 0 a 100 %.

Los parámetros que se visualizan en la pantalla OLED del nodo son los siguientes:

- Precipitación [mm]
- Evaporación del cenirrómetro [mm]
- Porcentaje de batería nodo [0-100 %]
- Longitud de la trama [unidades en decimal]

Por último, un led indicador conectado en el pin PC13 configurado como salida se enciende por un lapso de 2 segundos indicando que la trama ha sido enviada exitosamente por el enlace LoRa.

Figura 65. **Código del nodo, segmento 16**

```
155  float readResistance(int pin, int seriesResistance) {
156    float resistance = analogRead(pin);                  // Consigue valor de ADC.
157    resistance = (4095.0 / resistance) - 1.0;            // Convierte la lectura del ADC a resistencia.
158    return resistance;
159  }
160
161  float resistanceToLAMINAMM(float resistance, float zeroResistance, float calResistance, float calLAMINAMM) {
162    if (resistance > zeroResistance || (zeroResistance - calResistance) == 0.0) {
163      // Se detiene si el valor esta por encima del umbral de cero, o sino se a ingresado el valor de resistencia maxima en
164      //los valores de calibracion.
165      return 0.0;
166    }
167    // Calcula el factor de escala mapeando la resistencia en un rango de 0...1.0+ en relación con el valor máximo de la resistencia.
168    float scale = (zeroResistance - resistance) / (zeroResistance - calResistance);
169    // Escala maxLAMINAMM basada en el factor de escala calculado.
170    return calLAMINAMM * scale;
171  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *readResistance* consigue el valor de ADC del sensor y convierte la lectura del ADC a resistencia, luego se calcula el factor de escala mapeando la resistencia en relación con el valor máximo de la resistencia.

Figura 66. **Código del nodo, segmento 17**

```
207  void sleepAndRestart() {
208    display.clearDisplay();                                    // limpiar buffer
209    display.setTextSize(1);                                    // tamaño del texto
210    display.setCursor(0, 32);                                  // posicion del texto
211    display.print("Ingresando a sleep mode por 15 minutos...");
212    display.display();                                         // enviar a pantalla
213    delay(900000);                                             // tiempo en modo sleep
214    acknowledge = false;
215    contador = 0;
216    display.clearDisplay();                                    //limpiar buffer
217    display.setTextSize(1);                                    //tamaño del texto
218    display.setCursor(0, 32);                                  //posicion del texto
219    display.print("Despertando despues de 15 minutos, enviando nuevo mensaje...");
220    display.display();                                         // enviar a pantalla
221    delay(2000);                                               // tiempo de visualizacion
222  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La *function SleepAndRestart*() es llamada cuando la trama de datos ha sido enviada exitosamente, procede a cambiar el estado de las variables contador y *acknowledge* y esperan 15 minutos para despertar y volver a enviar una trama de datos.

Figura 67. **Código del nodo, segmento 18**

```
295  void initfailed(){
296        display.clearDisplay();                                  // limpiar buffer
297        display.setTextSize(1);                                  // tamaño del texto
298        display.setCursor(0, 32);                                // posicion del texto
299        display.print("Fallo en el inicio de LoRa, reintentando...");
300        display.display();                                       // enviar a pantalla
301        delay(2000);                                             // tiempo de visualizacion
302   }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Por otra parte, Las funciones auxiliares que son *inifailed(), initsucceeded(), sleepmsg(), sleepmsg2(), lenghtmsg(), msgnotforme(), acknowledgemsg() y successfullmsg()* que permiten desplegar los mensajes de alerta en la pantalla OLED. Tienen la misma lógica de limpiar el buffer, configurar el tamaño del texto, la posición del texto, enviar el texto a visualizar a la pantalla, y configurar un tiempo de visualización, por consiguiente, el código se puede encontrar completo en el apéndice 1.

Por último, al subir el código a la *blue pill* se ha terminado de programar el nodo.

4.2.3. Diseño de Gateway de un solo canal

En la siguiente sección se describe los pasos para realizar el diseño del *Gateway*.

4.2.3.1. Diagrama de flujo

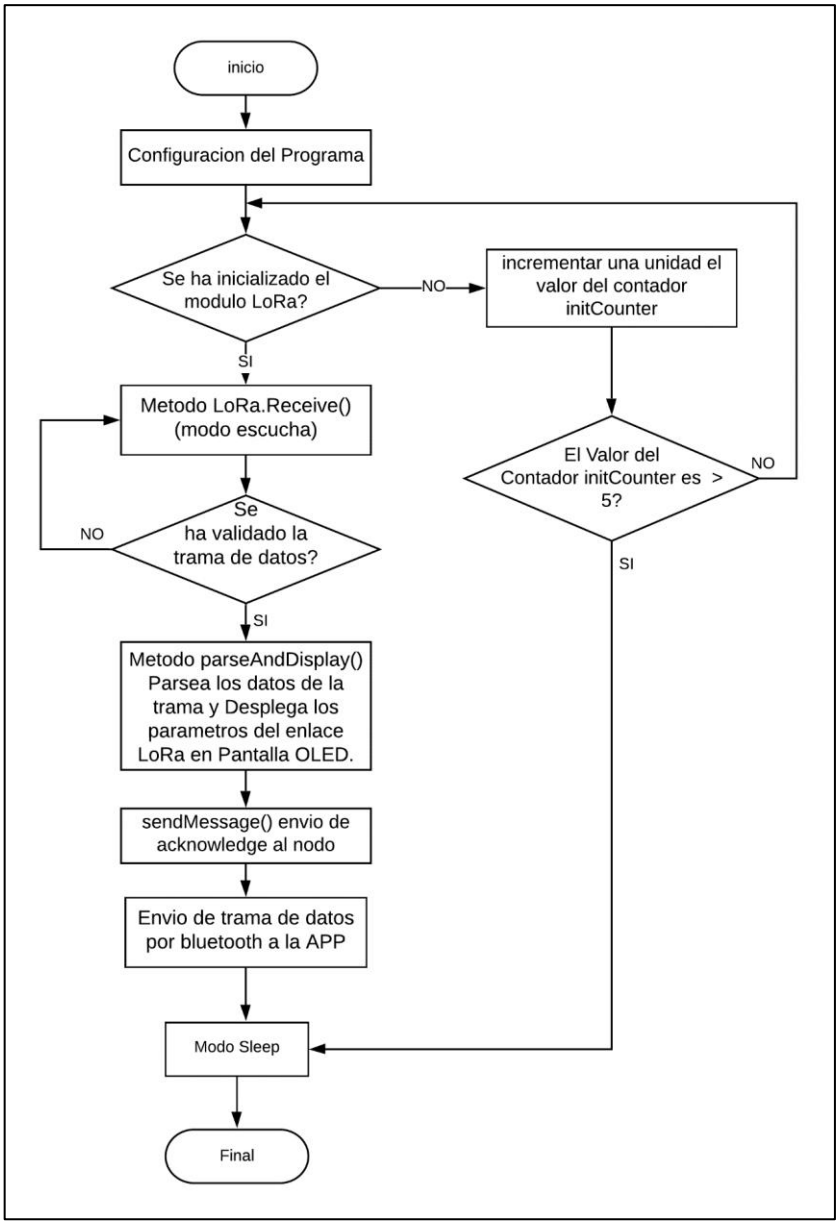
El diagrama de flujo es una representación gráfica del algoritmo del código fuente del *Gateway*, la lógica de la comunicación punto a punto se explica en esta sección para una mejor comprensión de la programación.

Las funciones principales del programa son:

- Configuración del programa.
- Inicialización del módulo LoRa.
- Almacenamiento e interpretación de la trama de datos.
- Visualización de datos.
- Envío de trama de datos por medio del protocolo *bluetooth.*

El diagrama de flujo siguiente resume la programación del *Gateway*:

Figura 68. **Diagrama de flujo *Gateway***



Fuente: elaboración propia, empleando Visio 2019.

4.2.3.2. Software del *Gateway*

En la siguiente sección se describe el conjunto de programas y rutinas que permiten al microcontrolador realizar determinadas tareas de configuración, entre ellas la configuración del radio permitiendo la recepción de datos y configuración de periféricos de salida que permiten la visualización de los datos así también la transmisión de datos por medio del módulo bluetooth hacia la aplicación Android.

4.2.3.2.1. Programación del *Gateway*

Para programar el *Gateway* con una computadora es necesario conectar el microcontrolador por medio de un puerto USB. Una vez conectado se puede conectar a la computadora y utilizar el IDE de Arduino como si de un dispositivo Arduino se tratará. A continuación, se detallará el código fuente que se ha utilizado para programar el *Gateway*. Este código se puede encontrar completo en el apéndice 4.

La configuración de la librería *armtronix's Modified lib of sandeepmistry* arduino-LoRa *for* STM32F103 de forma local se realiza utilizando el mismo procedimiento que se ha empleado con el nodo, por lo tanto, se encuentra disponible el código completo en el apéndice 2 y 3.

En segundo lugar, es necesario agregar las bibliotecas que se utilizarán, son las mismas bibliotecas utilizadas en la programación del nodo.

Figura 69. **Código del *Gateway*, segmento 1**

```
// Configuracion de Pantalla OLED
#define SCREEN_WIDTH 128 // OLED display ancho, en pixeles
#define SCREEN_HEIGHT 64 // OLED display alto, en pixeles

// Declaracion de variables de comunicacion LoRa
#define TX_P 17               // valor de potencia de transmision
#define BAND 915E6            // valor de frecuencia de transmision
#define SFactor 12            // valor de factor de propagacion
String incoming = "";         // trama a recibir
byte msgCount = 0;            // contador de tramas de salida
byte localAddress = 0x29;     // direccion de este dispositivo
byte destination = 0x30;      // direccion donde se enviara la trama
long lastSendTime = 0;        // ultimo tiempo de envio de trama
int interval = 10000;         // intervalo de tiempo entre envio de tramas
bool validated = false;       // bandera de validacion
String LoRaData = "";         // trama a envia
String LoRaMessage = "";      // trama a recibir
int recipient;                // direccion del receptor
byte sender;                  // direccion del emisor
byte incomingMsgId;           // ID de la trama entrante
byte incomingLength;          // longitud de mensaje entrante
int initCounter = 0;          // contador de intentos de inicializacion
bool initialization = false;  // bandera de inicializacion
int analogValor = 0;          // almacena el estado de la bateria en bits
float voltaje = 0;            // almacena estado  bateria en volts
int porcentajegateway = 0;    // almacena % de bateria del gateway
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Figura 70. **Código del *Gateway*, segmento 2**

```
// Declaracion de variables de interpretacion  de la trama entrante
int pos1, pos2, pos3, pos4;   // variables para separar la trama
String clicks;                // almacena el numero de clicks del pluviometro
String lamina;                // almacena el valor de la lamina de lluvia
String resistance;            // almacena el valor de resistencia del evaporimetro
String mm_evaporacion;                // almacena el valor de la lamina de evaporacion
String porcentaje;            // almacena el porcentaje de bateria del nodo

//Declaracion de pantalla SSD1306 conectado mediante I2C (pines SDA, SCL)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Después de la inclusión de bibliotecas y librerías locales, se colocan las definiciones, objetos y variables globales. Las definiciones serán valores que no cambiarán en el tiempo y se traducen al momento de la compilación, no se guardan como variables. Los objetos y variables globales son entidades a las que puede tener acceso cualquier función del código fuente.

Cualquier programación en el IDE de Arduino se compone de dos funciones principales, una función de configuración (*setup*) y otra de ejecución periódica (*loop*). La primera solamente se ejecuta una vez al inicio y la segunda se ejecuta de forma cíclica indeterminadamente. Dentro de la función de configuración se ha agregado la parte de configuración inicial que debe realizar el *Gateway*.

Figura 71. **Código del *Gateway*, segmento 3**

```
45  void setup() {
46      Serial1.begin(9600);       //inicializacion de la transmision de datos por uart
47
48    if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Direccion 0x3C para 128x64
49      SSD1306errormsg();         //mostrar alerta
50      for (;;);
51    }
52    delay(2000);
53    // configuracion de ADC de medidor de bateria
54
55    pinMode(PA3, INPUT_PULLDOWN);   // configuracion de pull-down para estado de bateria
56    pinMode(PC13, OUTPUT);          // configuracion del pin PC13 como salida
57    initSX1276();                   // metodo de inicializacion de enlace punto a punto
58  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Lo primero que se configura es la inicialización de la trasmisión de datos por uart (Serial1), debido que el módulo *bluetooth* transmite datos de forma serial, luego se inicializa la pantalla OLED en la dirección 0x3C, sino es exitosa llama a la función auxiliar SSD1306errormsg() para mostrar alerta, por último se procede a configurar el pin de entrada analógica y el pin de salida digital, el pin PA3 es la

entrada analógica que se utiliza con configuración *pulldown* con el objetivo de saber el estado actual de la batería, el pin PC13 es la salida digital que se utiliza como indicador.

Por último, se llama a la función initSX1276() que inicializa el enlace LoRa punto a punto.

Figura 72. **Código del *Gateway*, segmento 4**

```
126  void initSX1276() {
127    LoRa.setTxPower(TX_P);                // Configuracion de potencia de transmision
128    LoRa.setSpreadingFactor(SFactor);     // Configuracion de factor de propagacion
129    while (initCounter < 5) {             // Contador de intentos de reconexion
130      if (!LoRa.begin(BAND)) {            // Configuracion de Frecuencia
131        initfailed();                     // mostrar alerta
132        initCounter++;
133        delay(5000);                      // intervalo de tiempo de reintentos
134      } else {
135        LoRa.onReceive(onReceive);        // se pone modo escucha
136        LoRa.receive();                   // llama al metodo modo escucha
137        initsucceeded();                  // mostrar alerta
138        initialization = true;            // cambio de estado de bandera de inicializacion
139        break;
140      }
141    }
142    if (initialization == false) {        //bandera de inicializacion incorrecta
143      sleepmsg();                         // mensaje de inicio de modo ahorro de energia
144      LoRa.sleep();                       // se activa modo sleep
145
146    }
147  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Lo primero que se configura es la potencia de transmisión, la cual tiene un rango entre 2 y 17 dB, su valor por defecto es de 17dB que es la máxima potencia de transmisión, por lo tanto, TX_P tiene valor de 17 dB. Luego se configura el factor de ensanchamiento, el cual tiene un rango entre 6 y 12, su valor por defecto es de 7. *SFactor* tiene valor de 12. Por último, configuramos la frecuencia de

transmisión, para Europa se utiliza 868Mhz, Asia 433 Mhz y para América 915 Mhz, por lo tanto, BAND tiene el valor de 915E6.

El diagrama de flujo muestra que se debe validar si la inicialización del enlace punto a punto fue exitosa, si lo es entonces *initialization* cambia de valor a true ya que es una variable tipo binaria(*bool*) y funciona como una bandera, por lo tanto, permite continuar con el programa y el *Gateway* se coloca en modo escucha llamando a la función LoRa.*onReceive(onReceive),* en caso no es exitosa la inicialización se debe realizar 5 intentos, luego sino logra inicializar en 5 intentos entonces ingresa en modo *sleep* el chip LoRa por 15 minutos y luego vuelve intentar inicializar de nuevo.

Figura 73. **Código del *Gateway*, segmento 5**

```
95  void onReceive(int packetSize) {
96    if (packetSize == 0) return;          // si no hay paquete, retornar
97
98    incoming = "";
99    // leer los bytes de la cabecera del paquete:
100   recipient = LoRa.read();          // dirección del receptor
101   sender = LoRa.read();             // dirección del emisor
102   incomingMsgId = LoRa.read();      // ID del mensaje recibido
103   incomingLength = LoRa.read();     // longitud del mensaje recibido
104
105   while (LoRa.available()) {
106     incoming += (char)LoRa.read();  //recorre la trama recibida
107   }
108
109   if (incomingLength != incoming.length()) {   // comprobar la longitud en caso de error
110     lenghtmsg();                               // mostrar alerta
111     return;                                    // omitir el resto de la función
112   }
113
114   // if the recipient isn't this device or broadcast,
115   if (recipient != localAddress && recipient != 0xFF) { // Si la direccion del emisor y receptor no coinciden
116     msgnotforme();
117     return;                             // omitir el resto de la función
118   }
119
120   validated = true;
121   // Si el mensaje es para este dispositivo y la trama enviada si es igual a la recibida
122   successfulmsg();                      //mostrar alerta
123
124 }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *onReceive(int packetSize),* es la encargada de realizar una serie de pruebas para comprobar que la trama haya sido enviada correctamente, Primero verifica si ha recibido un paquete, si lo recibe entonces procede a leer los bytes de la cabecera del paquete, luego recorre la trama recibida y valida la longitud de la trama sino coincide la longitud de la trama enviada con la recibida entonces procede a llamar a la función *lenghmsg()* encargada de mostrar la alerta en la pantalla OLED, después de eso valida si la dirección del nodo y del *Gateway* coinciden, si no coinciden manda a llamar a la función *msgnotforme()* encargada de mostrar la alerta en la pantalla OLED, por lo contrario sí coinciden procede a cambiar de valor *Validated* a *true* ya que es una variable tipo binaria (*bool*) y funciona como una bandera, por lo tanto, permite continuar con el programa y también procede a llamar a la función *successfulmsg*() encargada de mostrar la alerta en la pantalla OLED.

Figura 74. **Código del *Gateway*, segmento 6**

```
60  void loop() {
61    if (initialization == true) {   // bandera de inicializacion exitosa
62      if (validated == true) {      // bandera de validacion de mensaje
63        parseAndDisplay();          // metodo de interpretacion de trama y visualizacion
64        sendMessage(incoming);      // envio de acknowledge
65        //trama a enviar a aplicacion android
66        LoRaMessage = String(clicks) + "," + String(lamina) + "," + String(resistance) + ","
67        + String(mm_evaporacion) + "," + String(porcentaje) + "," + String(porcentajegateway) + ","
68        + String(LoRa.packetRssi())+ "," + String(LoRa.packetSnr())+ ","
69        + String(sender, HEX) + "," + String(recipient, HEX)+ "," + String(incomingMsgId);
70
71        Serial1.println(LoRaMessage); // envio de trama por bluetooth
72        validated = false;            // cambia de estado a bandera de validacion
73        LoRa.receive();               // se pone en modo escucha por defecto
74
75      }
76    } else {
77      void sleepmsg2()                // muestra alerta
78      delay(900000);                  // espera de 15 minutos
79      initCounter = 0;                // cambia estado de contador
80      initSX1276();                   // metodo de inicializacion de enlace punto a punto
81    }
82  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Dentro de la función de ejecución aparece la condición si anidada donde se evalúan 2 condiciones, si la inicialización fue exitosa y si la validación de la trama de datos fue exitosa, si se cumplen ambas condiciones entonces procede a llamar a las función *parseAndDisplay()*.

Figura 75. **Código del *Gateway*, segmento 7**

```
149  void parseAndDisplay() {
150    pos1 = incoming.indexOf('/');       // almacena el valor de la trama de la pos 1
151    pos2 = incoming.indexOf('&');       // almacena el valor de la trama de la pos 2
152    pos3 = incoming.indexOf('#');       // almacena el valor de la trama de la pos 3
153    pos4 = incoming.indexOf('@');       // almacena el valor de la trama de la pos 4
154
155    clicks = incoming.substring(0, pos1);                // extrae el valor de la pos 1
156    lamina = incoming.substring(pos1 + 1, pos2);         // extrae el valor de la pos 2
157    resistance = incoming.substring(pos2 + 1, pos3);     // extrae el valor de la pos 3
158    mm_evaporacion = incoming.substring(pos3 + 1, pos4); // extrae el valor de la pos 4
159    porcentaje = incoming.substring(pos4 + 1, incoming.length()); // extrae el valor de la pos 5
160    analogValor = analogRead(PA3);                       //ADC estado de bateria
161    voltaje = 0.001025390 * analogValor;                 //voltaje actual de bateria
162    porcentajegateway = (voltaje - 3.2) * 100;           //porcentaje actual de bateria
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *parseAndDisplay()* se encarga de analizar la trama para determinar su estructura lógica, y luego con ayuda de la función *incoming.indexOf()* buscará caracteres dentro del *string incoming* que es la trama recibida, si encuentra el carácter que se busca *incoming.indexOf* devolverá la posición en la que se encuentra dicho carácter dentro del *String incoming*.

Las variables pos1, pos2, pos3 y pos4 almacenan el valor de la trama según la posición donde se encuentre los caracteres buscados.

Luego la función *incoming.substring()* se encarga de extraer el valor de las variables pos1, pos2, pos3 y pos4 y los almacena en las nuevas variables.

Posteriormente, es necesario medir el voltaje remanente en la batería del nodo, la variable voltaje almacena el voltaje actual de la batería, sabiendo este valor la variable porcentaje *Gateway* de tipo entero procede a indicar el estado de la batería en porcentaje en un intervalo de 0 a 100 %.

Figura 76. **Código del *Gateway*, segmento 8**

```
164   display.clearDisplay();               // limpiar buffer
165
166   // RSSI
167   display.setTextSize(1);               // tamaño del texto
168   display.setCursor(0, 0);              // posicion del texto
169   display.print("RSSI");
170   display.setTextSize(2);               // tamaño del texto
171   display.setCursor(0, 10);             // posicion del texto
172   display.print(LoRa.packetRssi());
173   display.setTextSize(1);               // tamaño del texto
174   display.print("dBm");
175
176   // SNR
177   display.setTextSize(1);               // tamaño del texto
178   display.setCursor(60, 0);             // posicion del texto
179   display.print("SNR");
180
181   display.setTextSize(2);               // tamaño del texto
182   display.setCursor(56, 10);            // posicion del texto
183   display.print(String(LoRa.packetSnr()));
184   display.setTextSize(1);               // tamaño del texto
185   display.print("dB");
186
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Figura 77. **Código del *Gateway*, segmento 9**

```
188    // Porcentaje de Bateria
189    display.setTextSize(1);              // tamaño del texto
190    display.setCursor(0, 35);            // posicion del texto
191    display.print("Bateria %");
192    display.setTextSize(2);              // tamaño del texto
193    display.setCursor(0, 45);            // posicion del texto
194    display.print(porcentajegateway);
195
196  // Longitud de Trama
197    display.setTextSize(1);              // tamaño del texto
198    display.setCursor(64, 35);           // posicion del texto
199    display.print("Long.Trama");
200    display.setTextSize(2);              // tamaño del texto
201    display.setCursor(64, 45);           // posicion del texto
202    display.print(incoming.length());
203    display.display();                   // enviar a pantalla
204
205    // led indicador
206    digitalWrite(PC13, LOW);
207    delay(2000);
208    digitalWrite(PC13, HIGH);
209    delay(2000);
210
211  }
212
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Los parámetros que se visualizan en la pantalla OLED del *Gateway* son los siguientes:

- RSSI [dBm]
- SNR [db]
- Porcentaje de batería *Gateway* [0-100 %]
- Longitud de la trama [unidades en decimal]

Por último, un led indicador conectado en el pin PC13 configurado como salida se enciende por un lapso de 2 segundos indicando que la trama ha sido recibida exitosamente por el enlace LoRa.

Figura 78. **Código del *Gateway*, segmento 10**

```
84  void sendMessage(String outgoing) {
85    LoRa.beginPacket();                   // paquete inicial
86    LoRa.write(destination);              // incluir la dirección de destino
87    LoRa.write(localAddress);             // incluir la dirección del emisor
88    LoRa.write(msgCount);                 // incluir el ID de la trama
89    LoRa.write(outgoing.length());        // incluir la longitud de la trama
90    LoRa.print(outgoing);                 // incluir trama
91    LoRa.endPacket();                     // terminar el paquete y enviarlo
92    msgCount++;                           // incrementar el ID del mensaje
93  }
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

La función *sendMessage*(), forma una cadena de texto, llamada *outgoing,* la cadena de texto es también conocida como trama y se establece de la misma manera que se estableció en el nodo solo que en esta ocasión se envía la trama para dar *acknowledge* al nodo.

Regresando a la función ejecución del código fuente principal, después de analizar, separar y desplegar los datos de la trama, se procede a enviar *acknowledge* al nodo, después se procede a enviar una nueva trama llamada LoRa*Message,* esta nueva trama está formada de la siguiente manera:

Figura 79. ***String* LoRaMessage**

Clicks	,	Lam Pluviometro	,	R Cenirrometro	,	Lam Cenirrometro	,	% bat nodo	,	% bat gateway	,	RSSI	,	SNR	,	Dir. Local Nodo	,	Dir. Destino	,	Long.Trama

Fuente: elaboración propia.

Los caracteres intermedios a cada variable sirven para indicarle a la aplicación Android donde empieza y donde termina el dato de cada parámetro.

Posterior, se procede a enviar la trama *LoRaMessage* por el serial1 a la aplicación Android y la bandera *validated* cambia de estado a false y se pone el *Gateway* en modo escucha para iniciar nuevamente con la secuencia del programa.

Por último, si alguna de las condiciones de inicialización o validación no fue exitosa entonces procede a llamar a la función *sleepmsg2()*, entra en modo *sleep* por 15 minutos, reinicia el contador *initCounter*, y vuelve a ingresar al método initsx1 276() para comenzar la secuencia del programa.

Por otra parte, las funciones auxiliares que son ssd1303errormsg (), *inifailed*(), *initsucceeded*(), *sleepmsg*(), *sleepmsg2*(), *lenghtmsg*(), *msmnotforme*(), *acknowledgemsg*() y *successfulmsg*() que permiten desplegar los mensajes de alerta en la pantalla OLED. Tienen la misma lógica de limpiar el buffer, configurar el tamaño del texto, la posición del texto, enviar el texto a visualizar a la pantalla, y configurar un tiempo de visualización, por consiguiente, el código se puede encontrar completo en el apéndice 4.

Por último, al subir el código a la *blue pill* se ha terminado de programar el *Gateway*. Por otra parte, las funciones auxiliares que son ssd1303*errormsg*(), *inifailed*(), *initsucceeded*().

4.2.4. Diseño de aplicación de Android

AppInventor es una herramienta de programación muy útil que permite el desarrollo de aplicaciones para dispositivos móviles con sistema operativo Android, este desarrollo se realiza en el dispositivo a través de internet y los cambios se guardan *online*. Además de crear el código, también debe juntar las distintas partes. Dado que cada parte del código requerido tiene la forma de un rompecabezas, esto lo hace más interesante y más fácil para las personas que no están familiarizadas con la programación o la creación de aplicaciones para el sistema operativo Android. La herramienta tiene dos partes principales: módulo web y editor de bloques.

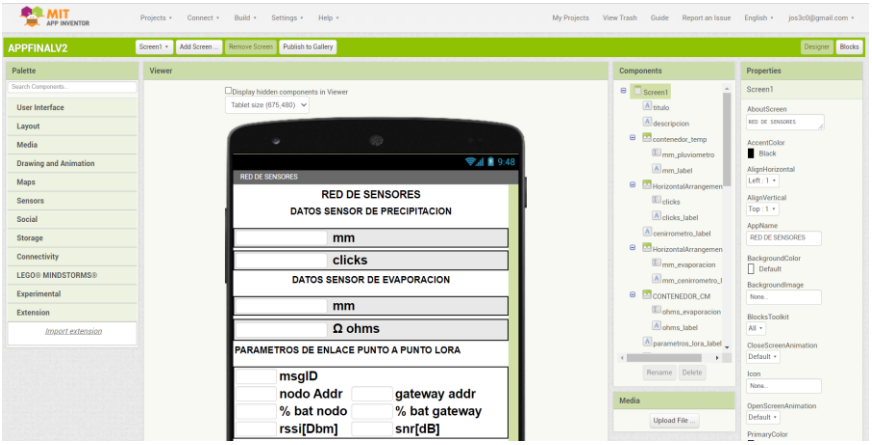
Módulo web: es la parte donde se puede acceder al proyecto, donde se pueden agregar componentes y su configuración. Allí, puede tener una vista previa de cómo aparecerá en el dispositivo que se instalará. Para que el desarrollador se familiarice con el desarrollo de aplicaciones Android.

Editor de bloques: responsable de establecer una conexión entre el teléfono para ser probada la ejecución de aplicaciones. Se pueden utilizar varios bloques de código para crear el código necesario para la aplicación que se está ejecutando en ese momento.

4.2.4.1. Interfaz de usuario de aplicación

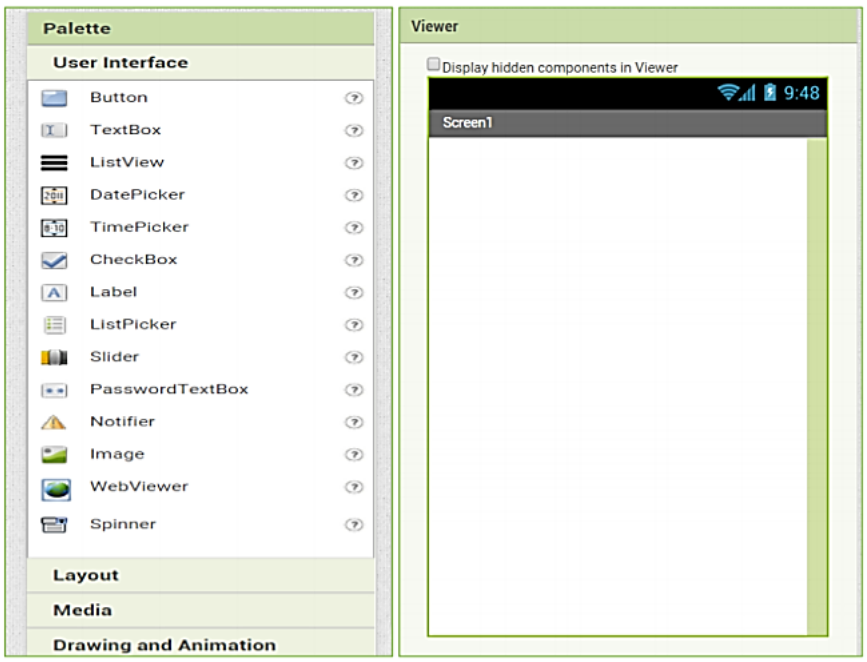
La primera parte del diseño de la aplicación de Android es la interfaz de usuario. Inicialmente, la página de diseño de App*Inventor* debe estar habilitada, lo que proporciona el espacio necesario para continuar con la interfaz. Entre ellos, el uso de campos de texto, etiquetas y botones.

Figura 80. **Captura del entorno de trabajo de App*Inventor***



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Figura 81. **La interfaz de usuario y visualización previa**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Los elementos del diseño de la aplicación se encuentran en la sección Paleta, y el diseño y el espacio de trabajo de vista previa se encuentran en la sección del visor.

Figura 82. **Propiedades de elementos**

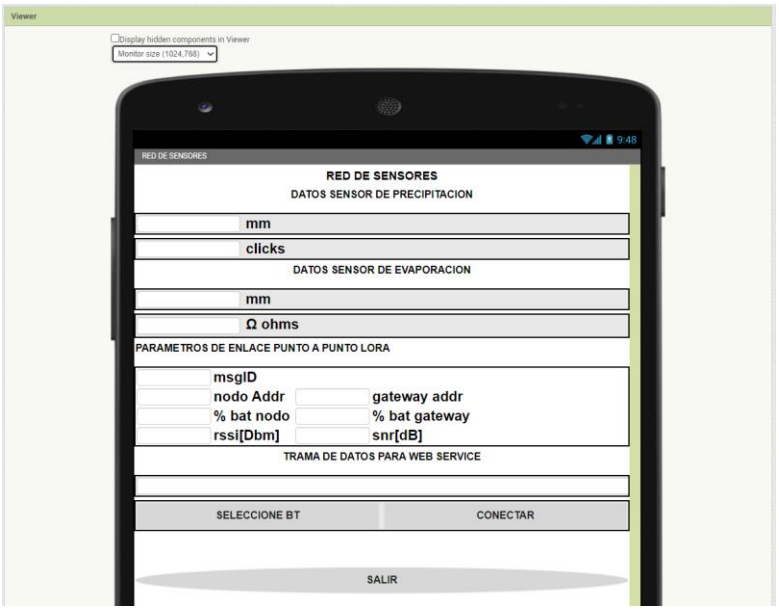
Fuente: elaboración propia, empleando MIT App Inventor V2.

En la sección de componentes, encontrará los diferentes atributos de cada tipo de elemento a ocupar, así como una lista de su ocupación.

Para crear esta pantalla denominada Red de sensores, se utilizan los siguientes elementos. 12 campos de texto donde se visualizarán los datos de la trama enviada por el *Gateway,* 17 etiquetas donde identificarán cada uno de los datos de la trama, 3 botones encargados de realizar las funciones de la aplicación Android, un componente de cliente *bluetooth* encargado de la comunicación

serial entre el *Gateway* y el teléfono móvil, un componente web encargado de la comunicación entre el *web service* y la aplicación Android, la cual proporciona funciones para solicitudes HTTP *GET*, *POST*, *PUT* y *DELETE*, un componente de notificación que muestra alarmas temporales cuando se produce algún error en la aplicación Android y 2 temporizadores que proporcionan el instante en el tiempo utilizando el reloj interno del teléfono. Puede disparar un temporizador a intervalos para que realice una acción, así como seis disposiciones horizontales que permiten organizar los campos de texto y etiquetas en forma horizontal y una disposición de tabla que permite agrupar varios campos de texto y etiquetas en un número determinado de filas y columnas; todos estos elementos se pueden encontrar en las pestañas de la sección de paleta. El color de fondo, el tipo y color de la letra, y el tamaño del botón se configuran a la derecha, donde se pueden encontrar los atributos de cada elemento al seleccionar.

Figura 83. **Desarrollo de la aplicación**

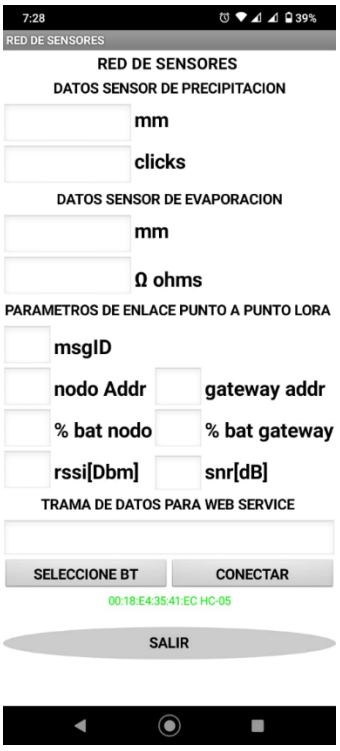


Fuente: elaboración propia, empleando MIT App Inventor V2.

El botón Seleccione BT, tiene la función de seleccionar el nombre del dispositivo *bluetooth* previamente emparejado, luego se tiene el botón conectar, que tiene la función de establecer la comunicación entre el *Gateway* y la aplicación, por último, se tiene el botón salir que tiene la función de salir de la aplicación.

En este *dashboard* los usuarios que ocuparán esta aplicación podrán obtener toda información necesaria sobre la red de sensores, el estado de la batería del nodo y *Gateway*, el número de mensaje enviado, la dirección del nodo y el *Gateway,* los parámetros de transmisión LoRa (SNR y RSSI) y por último se puede validar la trama de datos que enviara al servicio web.

Figura 84. ***Dashboard* de la aplicación Android**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

4.2.4.1. Programación de aplicación Android

La programación de la aplicación Android, se basa en la plataforma llamada App *Inventor*, este entorno de desarrollo es de uso público y gratuito, esto quiere decir que cualquier persona con un correo electrónico valido puede acceder a este sitio y empezar a desarrollar su propio *software* para dispositivos basados en el sistema operativo Android, de forma gratuita e ilimitada. A pesar de que todo el entorno de desarrollo es totalmente gráfico, la programación de las aplicaciones sigue el orden y las reglas de la programación, el objetivo es la simplicidad y facilidad de desarrollar nuevas aplicaciones, además cualquiera puede compartir sus aplicaciones de forma libre en la tienda de aplicaciones de *Google.*

App *Inventor* por la forma en que trabaja y como fue creado, permite la creación de aplicaciones en menos tiempo que otros leguajes de programación, incluyendo aplicaciones más complejas que llevan mucho tiempo de compilación basados en lenguajes de programación de texto.

En la siguiente sección se describe los pasos para realizar la programación de la aplicación Android.

4.2.4.1.1. Diagrama de flujo

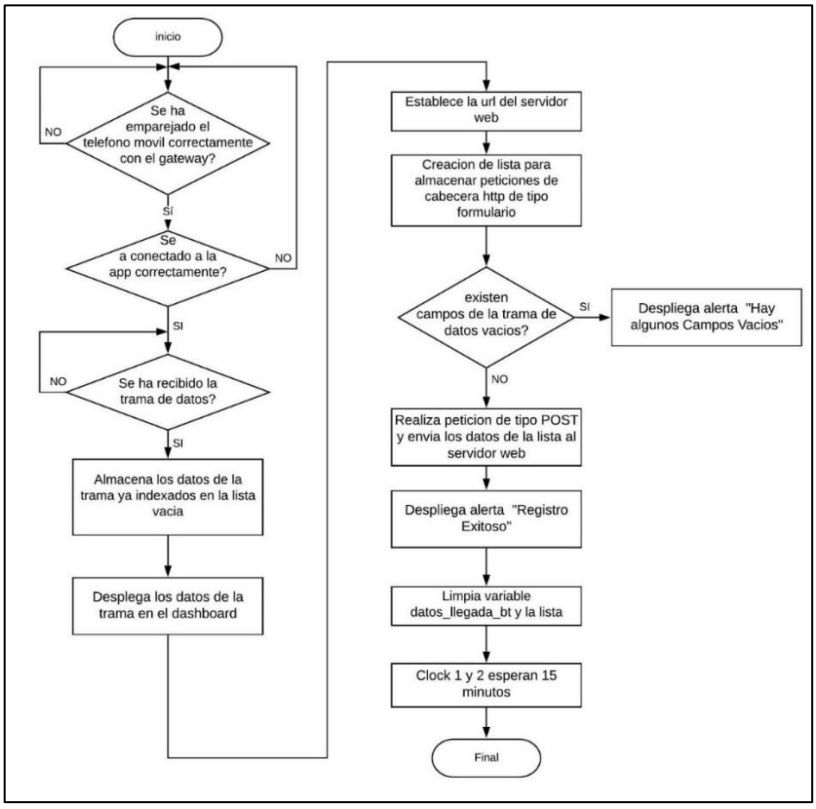
Bluetooth, es un protocolo de comunicación que necesita emparejamiento entre dispositivos, es una forma de registrar información para vincular dispositivos, tras emparejar los dispositivos por primera vez, no será necesario repetir emparejamiento para volver a conectarlos. Por tanto, las funciones de programación no funcionaran sino se cumple esta condición.

Las funciones principales del programa son:

- Almacenar e indexar los datos de llegada en una lista vacía.
- Desplegar los datos en el *dashboard*.
- Establecer comunicación con el servidor web.
- Realizar una petición http de tipo post al servidor.

El siguiente diagrama de flujo resume la programación de la aplicación Android:

Figura 85. **Diagrama de flujo App**



Fuente: elaboración propia, empleando Visio 2019.

4.2.4.1.2. Generación de código

Esta herramienta tiene una forma muy especial de generar el código, ya que cada función tiene la forma de un rompecabezas que conduce a la correcto y que no haya errores en el código.

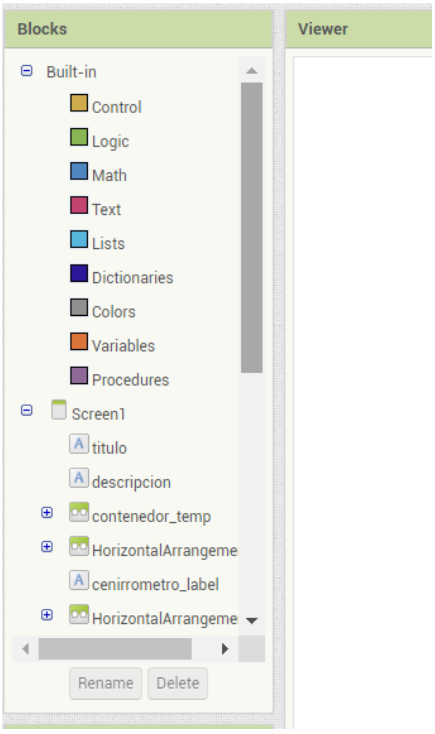
Para comenzar a generar código, es necesario hacer clic en el botón *blocks* en la esquina superior izquierda, donde se pueden encontrar todos los tipos de códigos.

Figura 86. **Botón *blocks***

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

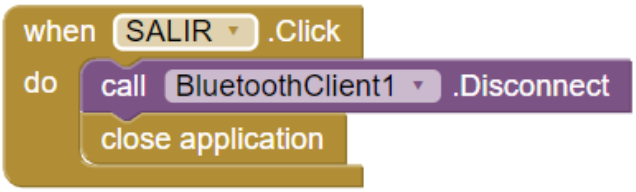
Cada pieza se puede sacar del lado izquierdo de la sección *blocks*, pero todo dependerá de lo que se quiera hacer; pero cada elemento también tiene su propio código, es decir, al seleccionar el elemento a utilizar se despliega una ventana transparente donde se muestran las piezas con los diferentes tipos de códigos que se pueden utilizar.

Figura 87. **Sección de programación tipo *blocks***



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Figura 88. **Código de aplicación Android, segmento 1**



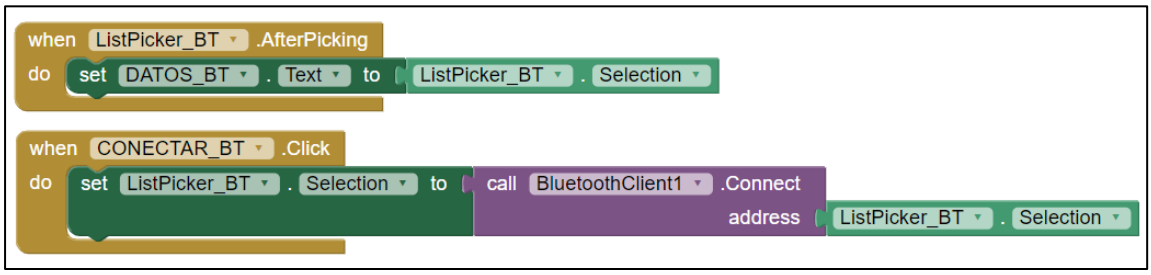
Fuente: elaboración propia, empleando MIT App *Inventor* V2.

En este caso este bloque de código lo que hace es que cuando presione el botón salir va desconectar el *bluetooth* y cerrar la aplicación, y esto se realizará mediante un *when-do*.

Primero se debe emparejar el módulo *bluetooth* con el teléfono móvil, el nombre del dispositivo *bluetooth* será HC-05 y la contraseña 1234, después HC-05 aparecerá en la lista de dispositivos emparejados previamente.

Cuando se presiona el botón seleccione *bluetooth* lo que hace el siguiente bloque es almacenar en la variable *ListPicker_BT* el listado de dispositivos emparejados previamente a mi teléfono móvil, y esto se realizó mediante un *when-do*.

Figura 89. **Código de aplicación Android, segmento 2**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

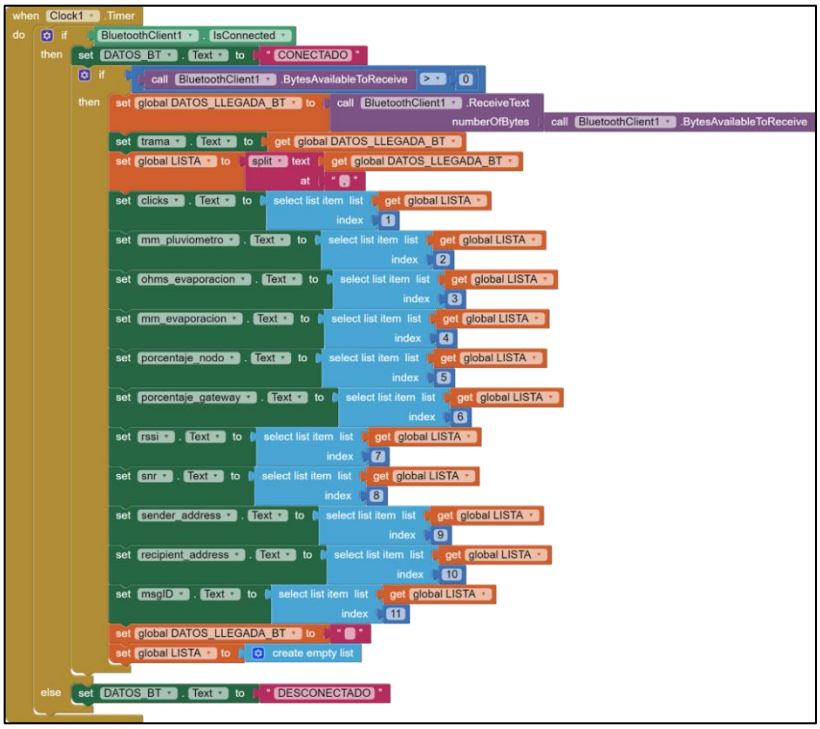
Luego que ya se encuentra enlazada la aplicación Android con el *Gateway* mediante *bluetooth* procede activarse el temporizador *clock* 1.

Figura 90. **Código de aplicación Android, segmento 3**

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Se crearon 2 variables una llamada lista que es la que se encarga de recibir la trama de los datos y está esperando que se separen por comas y otra llamada datos_llegada_bt ambas se inicializan con textos vacíos.

Figura 91. **Código de aplicación Android, segmento 4**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

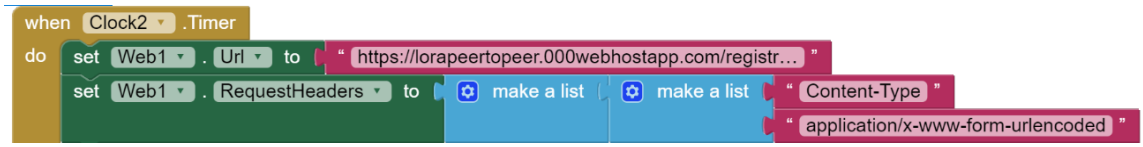
El *clock* 1 se configuro a 900 000 ms que es el equivalente a 15 minutos, cuando el temporizador es mayor a 15 minutos ejecuta la sentencia IF y pregunta si el *bluetooth* se encuentra conectado, si lo está entonces procede a mostrar en la etiqueta datos_bt que está conectado, de lo contrario procede a mostrar desconectado.

Luego realiza otra sentencia *IF* y pregunta si se tiene datos mayores a 0 en *BluetoothClient*1, si son mayores a cero significa que ha recibido la trama de datos entonces procede almacenar la trama en la variable datos de llegada *bluetooth* y cada vez que encuentre una coma procede a separar los datos en la variable lista, luego la variable lista comienza a irse indexando en la primera posición, segunda posición, entre otras.

Para evitar que los datos del *bluetooth* se llenen a cada rato en datos de llegada y la lista este almacenando muchas variables se necesita borrar eso.

La variable datos de llegada se limpian por medio de una variable vacía en otras palabras que no tenga ningún texto y la variable lista se vuelve a limpiar con el fin de borrar todos los datos para que cada vez que llegue un dato nuevo simplemente quede borrado y se puedan visualizar los datos nuevos.

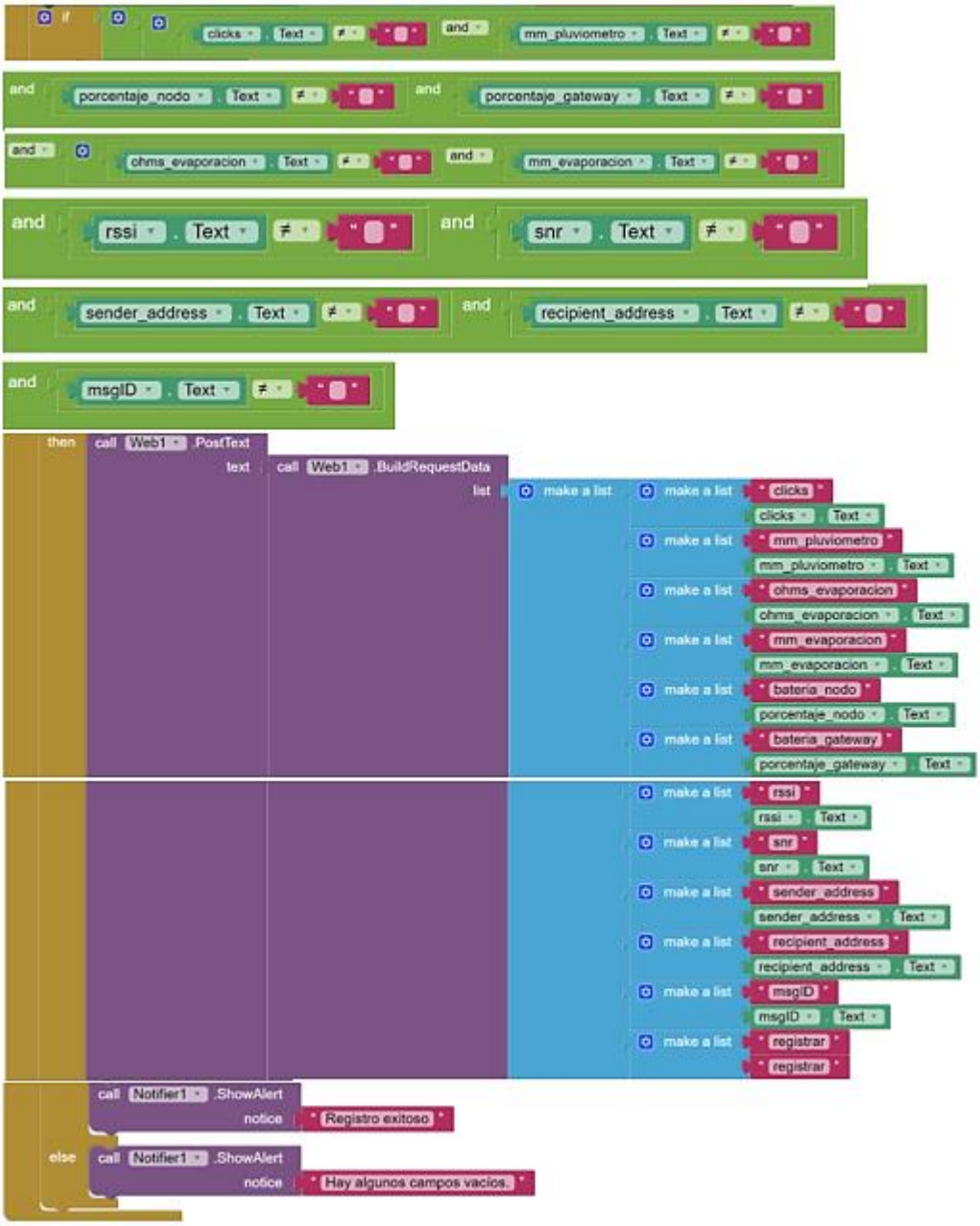
Figura 92. **Código de aplicación Android, segmento 5**



Fuente: elaboración propia, empleando MIT App Inventor V2.

El *clock* 2 se configuro a 900 000 ms que es el equivalente a 15 minutos, cuando el temporizador es mayor a 15 minutos entonces se establece la url https://lorapeertopeer.000webhostapp.com/registrar.php del servicio web donde se realizarán las peticiones, luego se establece la petición de la cabecera por lo tanto se crea una lista que contendrá una cabecera en http, el protocolo http es utilizado para acceder a páginas web y tienen normalmente una cabecera donde se especifican los datos que se enviarán, en este caso se le indica que se envía información de tipo formulario, y esto se realizó mediante un bloque *when-do*.

Figura 93. **Código de aplicación Android, segmento 6**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Posteriormente por medio de una sentencia *IF* se verifica que los campos que se enviarán en el formulario no se encuentren vacíos, si todos los campos están llenos entonces realiza una petición de solicitud tipo *POST* y se envía la lista de los atributos con sus valores.

Si la petición post ha sido exitosa se mostrará una alerta que informa al usuario que el registro ha sido exitoso de lo contrario mostrará una alerta que informa al usuario que existen campos vacíos.

Por último, se generan los archivos de instalación de la aplicación Android para el uso de la misma en un dispositivo móvil.

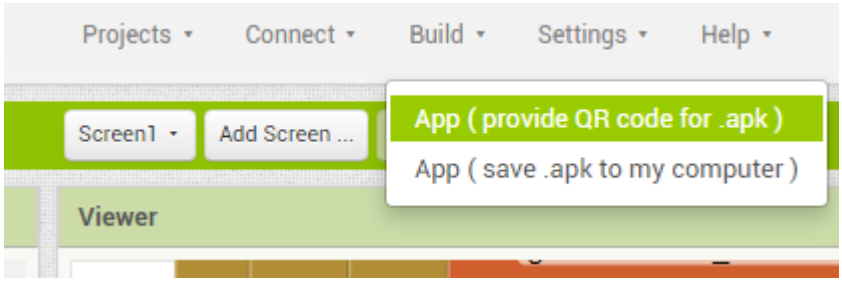
4.2.4.2. Generador de archivos de instalación

Hay dos formas de generar el archivo de instalación, una es crear un código QR, descargarlo e instalarlo en el dispositivo móvil, y la otra es crear un archivo APK para instalarlo en el dispositivo móvil.

El primer método de instalación se muestra a continuación:

- Primero, ir al menú "Construcción" o "*Build*", y luego se selecciona la opción "App(*provide QR code for .*apk)".

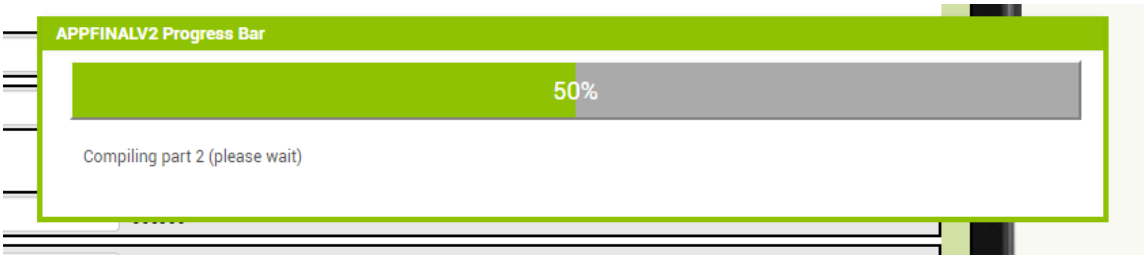
Figura 94. **Generador de archivo, segmento 1**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Inmediatamente se comenzará a generar el código QR, para que se pueda descargar a través de un dispositivo móvil.

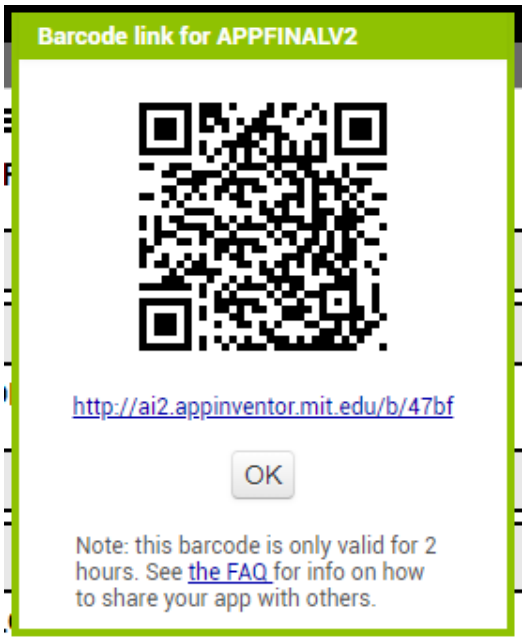
Figura 95. **Generador de archivo, segmento 2**



Fuente: elaboración propia, empleando MIT App Inventor V2.

La siguiente imagen es el código que se va a escanear con el dispositivo móvil.

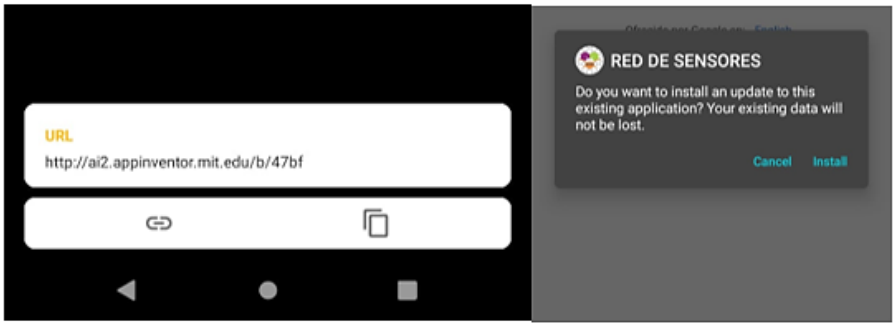
Figura 96. **Generador de archivo, segmento 3**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

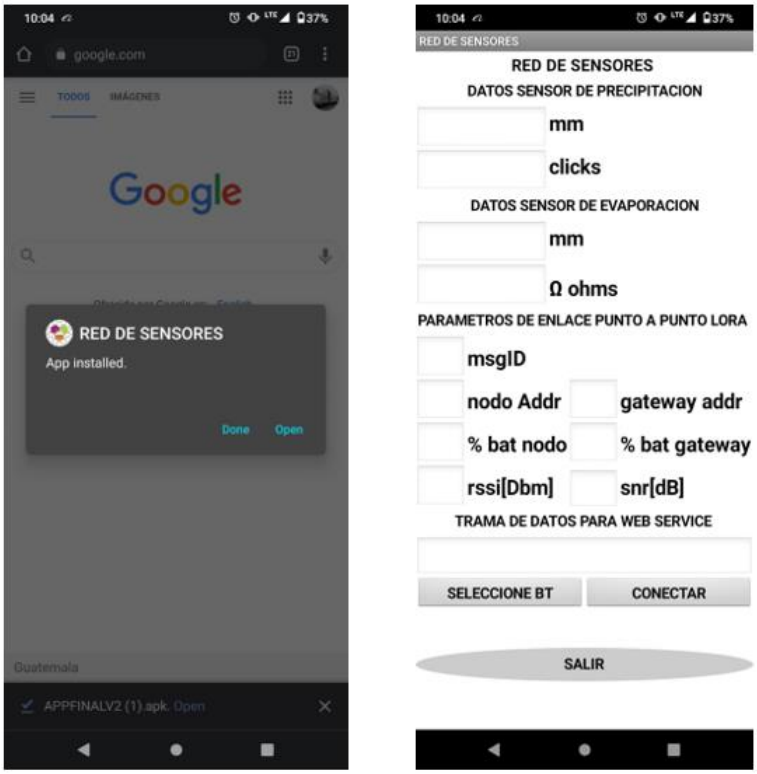
Ya en el dispositivo, después de haber escaneado el código aparecerá la siguiente imagen en donde se le dará abrir, para que en seguida comience a descargar el archivo red_de_sensores.apk, cuando finalmente el archivo este completamente descargado, se selecciona instalar, para que al finalizar la instalación se muestre la aplicación ya instalada en el dispositivo móvil.

Figura 97. **Generador de archivo, segmento 4**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Figura 98. **Generador de archivo, segmento 5**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

4.2.4.3. Procedimiento de levantado de servicio web

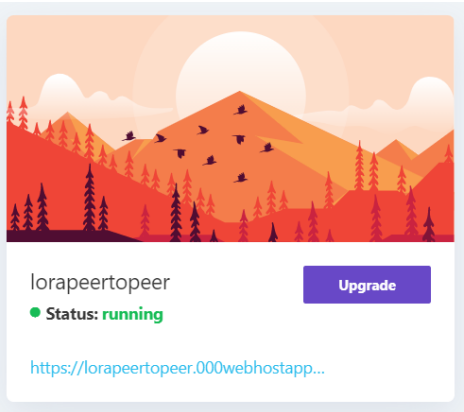
En esta sección se muestra cómo guardar información en una base de datos MySQL. Para ello se necesita contratar un *hosting*. En este caso se utilizará uno gratuito. A continuación, se creará y configurará la base de datos, se creará una tabla con 12 columnas para almacenar los datos de la trama enviada por la aplicación Android. Por otro lado, se usará un *script* en PHP para recibir los datos mediante una petición web y se guardará en la tabla correspondiente.

4.2.4.3.1. Configuración de servidor

Primero se contrató un plan de *hosting* para poner el archivo php para manejar todas las peticiones y guardar el contenido que se desea en la base de datos.

La url del *hosting* que se utiliza es el siguiente: https://www.000webhost.com aquí se crea una cuenta gratuita para tener acceso al servidor.

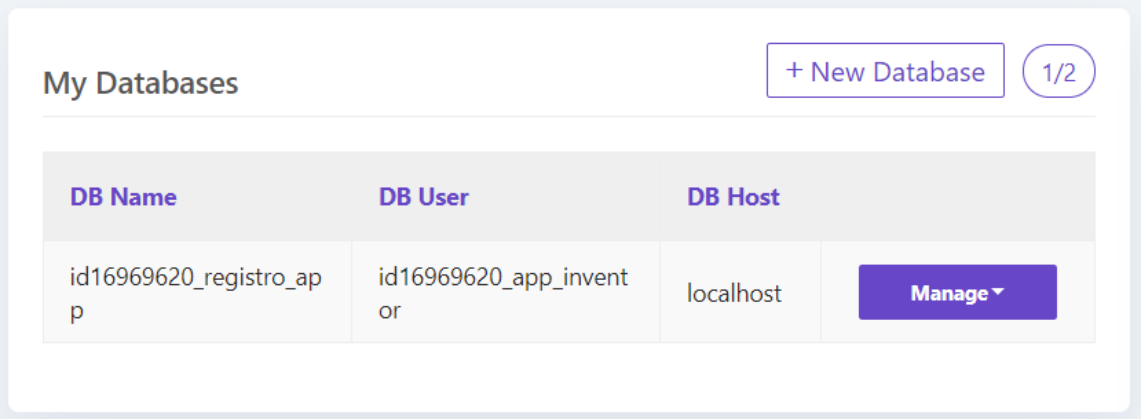
Figura 99. **Configuración de servidor web, segmento 1**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Después se crear la base de datos se podrán ver los detalles como se muestra en la siguiente figura.

Figura 100. **Configuración de servidor web, segmento 2**

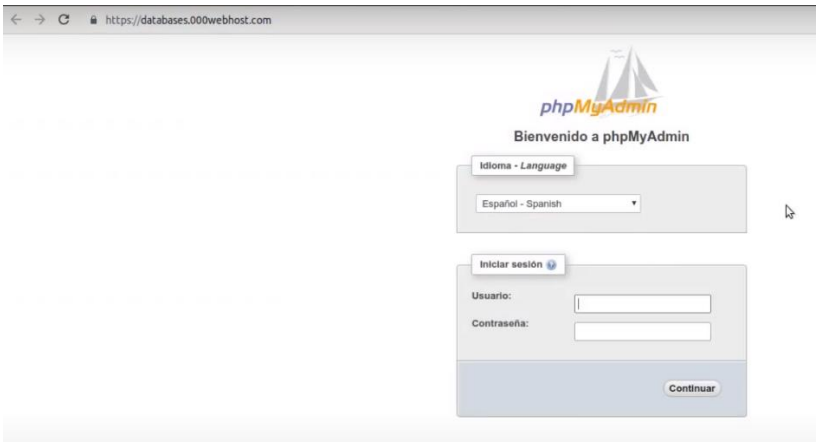


Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Una vez creada la base de datos el web *hosting* asigna una ID debido que esta base de datos es compartida entre diversos usuarios por lo tanto el nombre y usuario que se ha elegido para la base de datos va ir acompañado de esta Id.

Se puede gestionar la base de datos mediante phpmyadmin.

Figura 101. **Configuración de servidor web, segmento 3**



Fuente: elaboración propia, empleando MIT App Inventor V2.

Phpmyadmin es un *framework* hecho para gestionar bases de datos, y se puede acceder mediante el usuario y contraseña configurado previamente.

Las características del servidor de base de datos se muestran a continuación:

Figura 102. **Configuración de servidor web, segmento 4**

Fuente: elaboración propia, empleando MIT App Inventor V2.

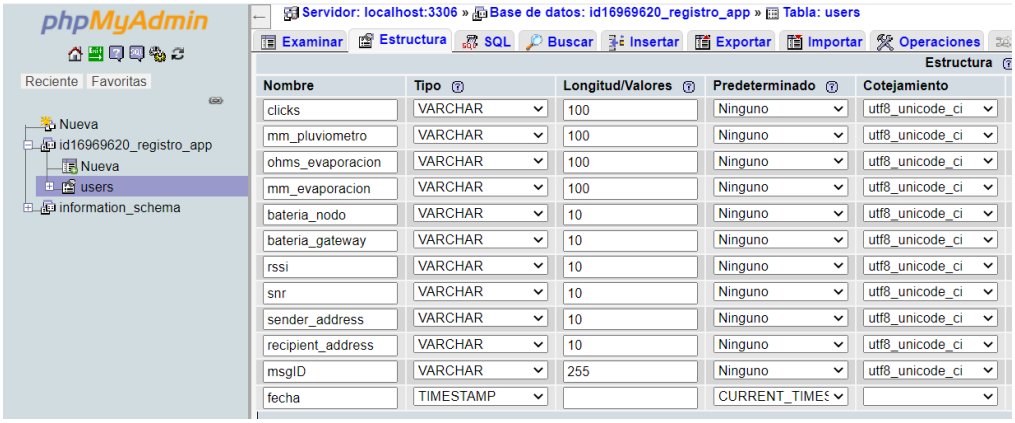
En la sección de estructura se encuentra la opción de crear tablas, se procede asignar el nombre de la tabla, así como el número de columnas.

Figura 103. **Configuración de servidor web, segmento 5**

Fuente: elaboración propia, empleando MIT App Inventor V2.

Usando PhpMyAdmin se creó la tabla "*users*" con 12 columnas (*clicks*, mm_pluviometro, ohms_evaporacion, mm_evaporacion, bateria_nodo, bateria_gateway, rssi, sender_address, recipient_address, msgID y fecha).

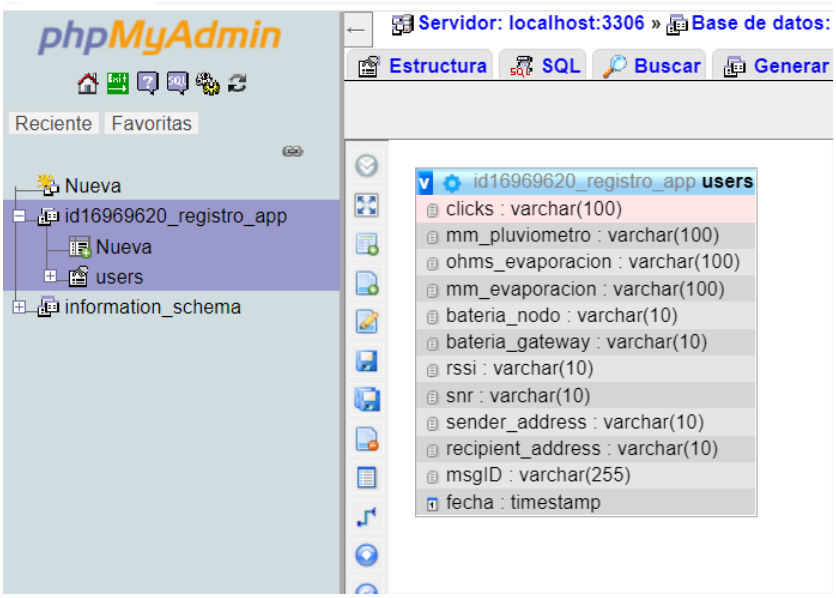
Figura 104. **Configuración de servidor web, segmento 6**



Fuente: elaboración propia, empleando MIT App Inventor V2.

A continuación, se muestra el esquema lógico de la única tabla de la base de datos.

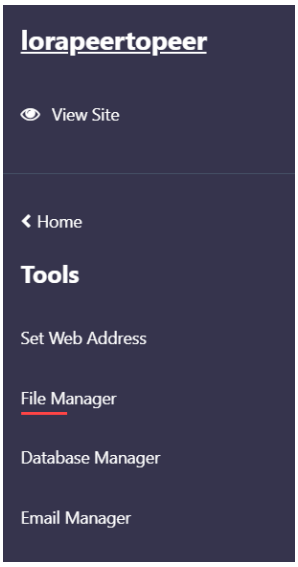
Figura 105. **Configuración de servidor web, segmento 7**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Luego se vuelve al *hosting* y se selecciona la sección *file manager.*

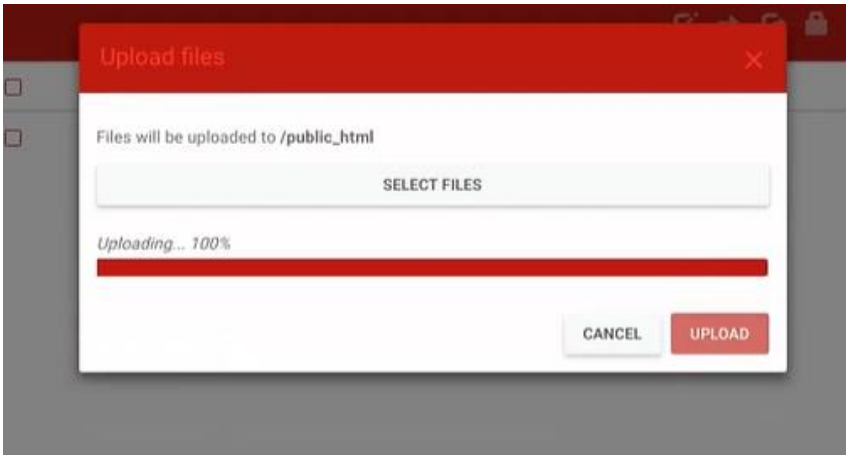
Figura 106. **Configuración de servidor web, segmento 8**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Dentro de *file manager* se necesita cargar el *script* de registro.

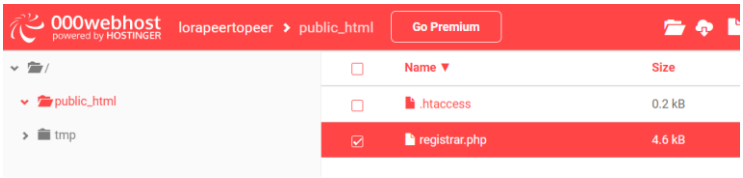
Figura 107. **Configuración de servidor web, segmento 9**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Una vez completada la carga del archivo registro.php aparecerá dentro la carpeta public_html.

Figura 108. **Configuración de servidor web, segmento 10**



Fuente: elaboración propia, empleando MIT App *Inventor* V2.

4.2.4.3.2. Programación del script en php

En esta sección se describe la programación del script diseñado para el servidor web. En la base de datos de pruebas se utiliza un *Web Hosting* gratuito llamado 000Webhost.com el cual es alimentado por la aplicación Android, a continuación, en la figura 109 se presenta la programación del *script* utilizando el lenguaje de programación php.

Figura 109. **Programación del *script,* segmento 1**

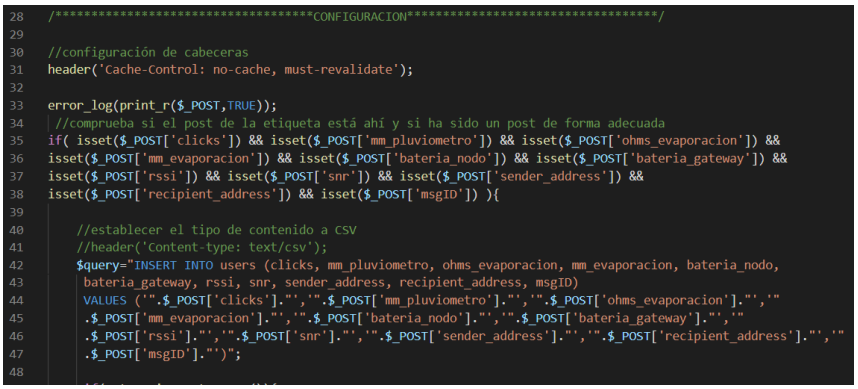
```php
<?php
//Configuracion de Web Service
/*********************************CONFIGURACION*******************************/
//DETALLES DE BASE DE DATOS//
$DB_ADDRESS="localhost";
$DB_USER="id16969620_app_inventor";
$DB_PASS="Guatemala2021*";
$DB_NAME="id16969620_registro_app";


//CONFIGURACIONES//

if( isset($_GET['clicks'])) {
    $_POST['clicks']=$_GET['nombre'];
    $_POST['mm_pluviometro']=$_GET['mm_pluviometro'];
    $_POST['ohms_evaporacion']=$_GET['ohms_evaporacion'];
    $_POST['mm_evaporacion']=$_GET['mm_evaporacion'];
    $_POST['bateria_nodo']=$_GET['bateria_nodo'];
    $_POST['bateria_gateway']=$_GET['bateria_nodo'];
    $_POST['rssi']=$_GET['rssi'];
    $_POST['snr']=$_GET['snr'];
    $_POST['sender_address']=$_GET['sender_address'];
    $_POST['recipient_address']=$_GET['recipient_address'];
    $_POST['msgID']=$_GET['msgID'];
}

```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Lo primero que se realiza es configurar el servidor, se establece el nombre de la base de datos en DB_NAME, luego se colocan las credenciales para tener acceso a la base de datos en DB_USER y DB_PASS y por último se establece la dirección de la base de datos en DB_ADDRESS.

Posteriormente, se utiliza la sentencia IF, pregunta si la variable *clicks* existe, entonces proceder a insertar los valores de la trama de datos en su variable correspondiente.

Figura 110. **Programación del *script,* segmento 2**

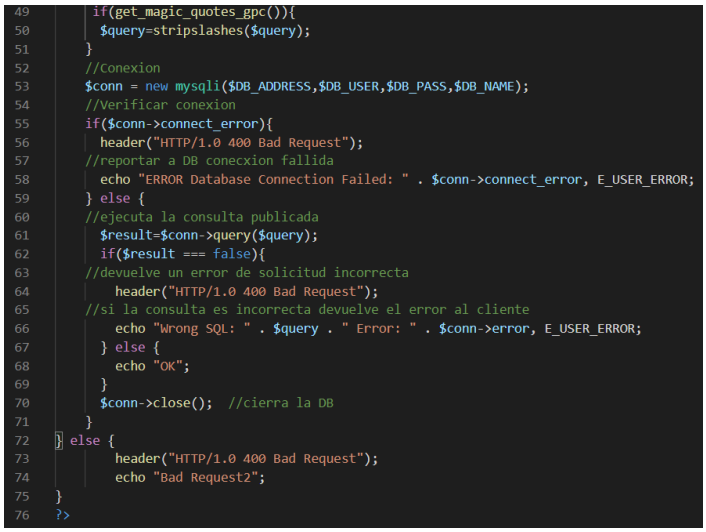


```
/*********************************CONFIGURACION*********************************/

//configuración de cabeceras
header('Cache-Control: no-cache, must-revalidate');

error_log(print_r($_POST,TRUE));
 //comprueba si el post de la etiqueta está ahí y si ha sido un post de forma adecuada
if( isset($_POST['clicks']) && isset($_POST['mm_pluviometro']) && isset($_POST['ohms_evaporacion']) &&
isset($_POST['mm_evaporacion']) && isset($_POST['bateria_nodo']) && isset($_POST['bateria_gateway']) &&
isset($_POST['rssi']) && isset($_POST['snr']) && isset($_POST['sender_address']) &&
isset($_POST['recipient_address']) && isset($_POST['msgID']) ){

    //establecer el tipo de contenido a CSV
    //header('Content-type: text/csv');
    $query="INSERT INTO users (clicks, mm_pluviometro, ohms_evaporacion, mm_evaporacion, bateria_nodo,
     bateria_gateway, rssi, snr, sender_address, recipient_address, msgID)
     VALUES ('".$_POST['clicks']."','".$_POST['mm_pluviometro']."','".$_POST['ohms_evaporacion']."','"
     .$_POST['mm_evaporacion']."','".$_POST['bateria_nodo']."','".$_POST['bateria_gateway']."','"
     .$_POST['rssi']."','".$_POST['snr']."','".$_POST['sender_address']."','".$_POST['recipient_address']."','"
     .$_POST['msgID']."')";
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Luego se configuran las cabeceras y comprueba si el *post* de la etiqueta está ahí y si ha sido un *post* de forma adecuada. Luego establece el tipo de contenido por medio de *INSERT INTO*.

Figura 111. **Programación del *script*, segmento 3**



```
    if(get_magic_quotes_gpc()){
      $query=stripslashes($query);
    }
    //Conexion
    $conn = new mysqli($DB_ADDRESS,$DB_USER,$DB_PASS,$DB_NAME);
    //Verificar conexion
    if($conn->connect_error){
      header("HTTP/1.0 400 Bad Request");
    //reportar a DB conecxion fallida
      echo "ERROR Database Connection Failed: " . $conn->connect_error, E_USER_ERROR;
    } else {
    //ejecuta la consulta publicada
      $result=$conn->query($query);
      if($result === false){
    //devuelve un error de solicitud incorrecta
        header("HTTP/1.0 400 Bad Request");
    //si la consulta es incorrecta devuelve el error al cliente
        echo "Wrong SQL: " . $query . " Error: " . $conn->error, E_USER_ERROR;
      } else {
        echo "OK";
      }
      $conn->close();  //cierra la DB
    }
} else {
        header("HTTP/1.0 400 Bad Request");
        echo "Bad Request2";
}
?>
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Por último, establece conexión con la base de datos, verifica conexión, ejecuta la consulta publicada, si es incorrecta la consulta devuelve un error, pero si es correcta cierra la base de datos.

4.3. Resultados de las pruebas del diseño

A continuación, se documenta con fotografías, el funcionamiento del ejemplo que se ha propuesto en este capítulo.

4.3.1. Conectando los sensores y periféricos de salida al nodo

El pluviómetro y el cenirrómetro se conectan al nodo por medio de un conector RJ11, los contactos son tiras de metal que permiten que la corriente fluya desde el sensor hasta los pines de la *blue pill,* evitando el ruido e interferencias en las líneas de conducción, otra ventaja que ofrece este conector al usuario es una forma práctica y fácil de instalación.

La pantalla OLED y la alimentación se conectan por medio de un conector tipo *Glove* XH2.54 JST de 4 pines, es comúnmente usado para sensores I2C.

El panel solar se conecta al módulo TP4056 por medio de una terminal PTR AK500/02.

4.3.2. Pruebas de funcionamiento del nodo

Para conectar el nodo, se acciona el interruptor *ON/OFF*, el cable vivo del circuito se conecta en las terminales NO y COM del nodo. Una vez energizado el nodo la pantalla OLED mostrará el siguiente mensaje de alerta.

Figura 112. **Pruebas de funcionamiento, segmento 1**

Fuente: elaboración propia.

Luego el nodo realiza la adquisición de datos de los sensores y procede a mostrarlos en la pantalla como se observa en la siguiente figura.

Figura 113. **Pruebas de funcionamiento, segmento 2**

Fuente: elaboración propia.

Después procede a enviar la trama de datos por medio del enlace LoRa, cuando el envió es exitoso muestra la alerta de la siguiente figura.

Figura 114. **Pruebas de funcionamiento, segmento 3**

Fuente: elaboración propia.

Posteriormente, al envío de la trama de datos el nodo se pone en modo escucha, esperando recibir *acknowledge* del *Gateway*, si lo recibe exitosamente despliega la alerta mostrada en la figura siguiente y entra en *sleep mode* por 15 minutos.

Figura 115. **Pruebas de funcionamiento, segmento 4**

Fuente: elaboración propia.

Por último, pasados los 15 minutos el nodo despierta y empieza a realizar el ciclo de la función ejecución de nuevo, así también despliega la alerta mostrada en la siguiente figura.

Figura 116. **Pruebas de funcionamiento, segmento 5**

Fuente: elaboración propia.

Existen otras alertas que despliega el nodo cuando el ciclo no es exitoso, en la sección de anexos se encontrarán las fotografías de estas alertas.

4.3.3. Pruebas de funcionamiento del *Gateway*

Para conectar el *Gateway*, se acciona el interruptor *ON/OFF*, el cable vivo del circuito se conecta en las terminales de RETORNO y COM del *Gateway*. Una vez energizado el *Gateway* la pantalla OLED mostrará el siguiente mensaje de alerta.

Figura 117. **Pruebas de funcionamiento, segmento 6**

Fuente: elaboración propia.

Luego el *Gateway* realiza la medición del estado de la batería, verifica la trama de datos recibida y procede a mostrar los parámetros del enlace LoRa en la pantalla juntamente con la longitud de la trama y el porcentaje de batería como se observa en la figura siguiente.

Figura 118. **Pruebas de funcionamiento, segmento 7**

Fuente: elaboración propia.

Posteriormente el *Gateway* envia *acknowledge* al nodo y si ha sido enviado exitosamente despliega la alerta mostrada en la siguiente figura.

Figura 119. **Pruebas de funcionamiento, segmento 8**

Fuente: elaboración propia.

Por último, la trama de datos es enviada a la aplicación Android por medio del módulo *bluetooth* y si ha sido enviada exitosamente despliega la alerta mostrada en la siguiente figura.

Figura 120. **Pruebas de funcionamiento, segmento 9**

Fuente: elaboración propia.

Luego el *Gateway* queda en modo escucha en espera que el nodo envie la siguiente trama de datos para comenzar el ciclo de la función ejecución nuevamente.

Existen otras alertas que despliega el *Gateway* cuando el ciclo no es exitoso de alguna parte del programa, en la sección de anexos se encontrarán las fotografías de estas alertas.

4.3.4. Pruebas de funcionamiento de la aplicación Android

Después de emparejar el módulo *bluetooth*, se procede a ingresar a la aplicación red de sensores, luego se seleccionar el botón seleccione BT y se busca en la lista de dispositivos emparejados el siguiente 00:18: E4:35:41:EC HC-05, tal como se muestra en la siguiente figura.

Figura 121. **Pruebas de funcionamiento, segmento 10**

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Después se procede a seleccionar el botón conectar, si se conecta exitosamente se mostrará una alerta de color verde que dice conectado tal como se muestra en la figura siguiente.

Figura 122. **Pruebas de funcionamiento, segmento 11**

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Una vez está establecida la comunicación entre el *Gateway* y la aplicación Android se coloca en modo escucha, una vez recibe la trama de datos procede a llenar el *dashboard,* tal como se muestra en la figura siguiente.

Figura 123. **Pruebas de funcionamiento, segmento 12**

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

Por último, la aplicación procede a establecer comunicación con el servidor web, si el envío de la trama de datos al servidor web es exitosa procede a mostrar la alerta de registro exitoso, tal como se muestra en la figura siguiente.

Figura 124. **Pruebas de funcionamiento, segmento 13**

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

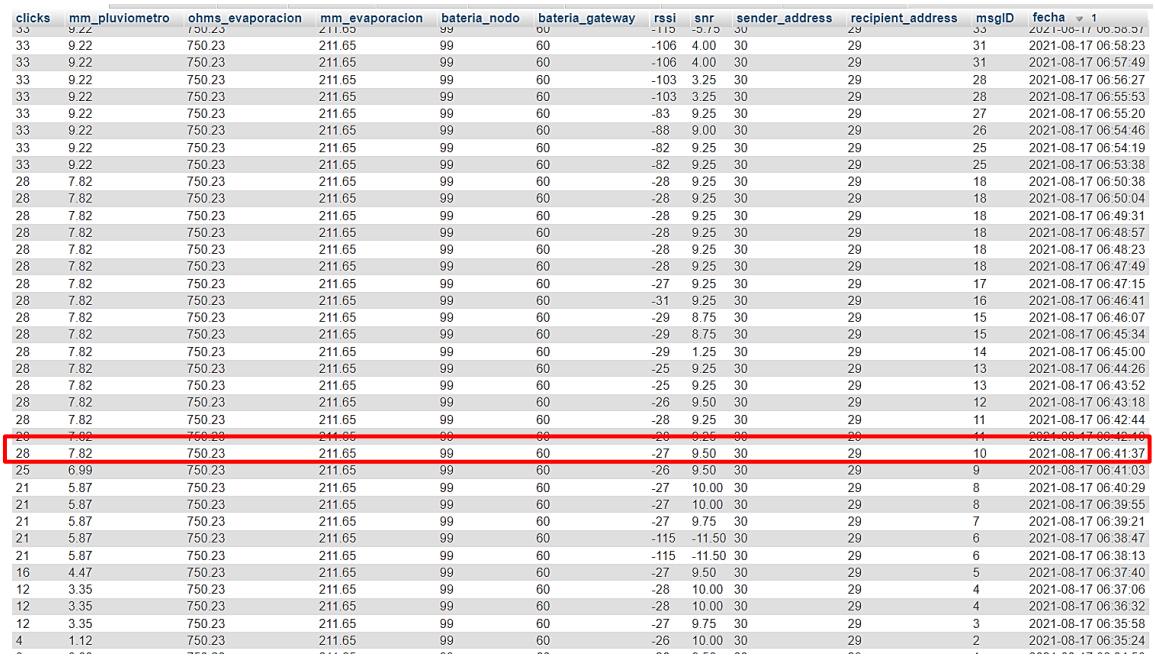
Existen otras alertas que despliega la aplicación Android cuando el ciclo no es exitoso, en la sección de anexos se encontrarán las fotografías de estas alertas.

4.3.5. Pruebas de funcionamiento del servidor Web

Primero se ingresar al *Hosting* https://www.000webhost.com, después se procede a iniciar sesión y acceder a la base de datos que es gestionado por phpmyadmin.

Posteriormente, se procede actualizar la base de datos y se despliegan los valores de los parámetros enviados desde la aplicación Android, tal como se muestra en la figura siguiente.

Figura 125. **Pruebas de funcionamiento, segmento 14**



clicks	mm_pluviometro	ohms_evaporacion	mm_evaporacion	bateria_nodo	bateria_gateway	rssi	snr	sender_address	recipient_address	msgID	fecha
33	9.22	750.23	211.65	99	60	-115	-5.75	30	29	33	2021-08-17 06:58:57
33	9.22	750.23	211.65	99	60	-106	4.00	30	29	31	2021-08-17 06:58:23
33	9.22	750.23	211.65	99	60	-106	4.00	30	29	31	2021-08-17 06:57:49
33	9.22	750.23	211.65	99	60	-103	3.25	30	29	28	2021-08-17 06:56:27
33	9.22	750.23	211.65	99	60	-103	3.25	30	29	28	2021-08-17 06:55:53
33	9.22	750.23	211.65	99	60	-83	9.25	30	29	27	2021-08-17 06:55:20
33	9.22	750.23	211.65	99	60	-88	9.00	30	29	26	2021-08-17 06:54:46
33	9.22	750.23	211.65	99	60	-82	9.25	30	29	25	2021-08-17 06:54:19
33	9.22	750.23	211.65	99	60	-82	9.25	30	29	25	2021-08-17 06:53:38
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:50:38
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:50:04
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:49:31
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:48:57
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:48:23
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	18	2021-08-17 06:47:49
28	7.82	750.23	211.65	99	60	-27	9.25	30	29	17	2021-08-17 06:47:15
28	7.82	750.23	211.65	99	60	-31	9.25	30	29	16	2021-08-17 06:46:41
28	7.82	750.23	211.65	99	60	-29	8.75	30	29	15	2021-08-17 06:46:07
28	7.82	750.23	211.65	99	60	-29	8.75	30	29	15	2021-08-17 06:45:34
28	7.82	750.23	211.65	99	60	-29	1.25	30	29	14	2021-08-17 06:45:00
28	7.82	750.23	211.65	99	60	-25	9.25	30	29	13	2021-08-17 06:44:26
28	7.82	750.23	211.65	99	60	-25	9.25	30	29	13	2021-08-17 06:43:52
28	7.82	750.23	211.65	99	60	-26	9.50	30	29	12	2021-08-17 06:43:18
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	11	2021-08-17 06:42:44
28	7.82	750.23	211.65	99	60	-28	9.25	30	29	11	2021-08-17 06:42:10
28	7.82	750.23	211.65	99	60	-27	9.50	30	29	10	2021-08-17 06:41:37
25	6.99	750.23	211.65	99	60	-26	9.50	30	29	9	2021-08-17 06:41:03
21	5.87	750.23	211.65	99	60	-27	10.00	30	29	8	2021-08-17 06:40:29
21	5.87	750.23	211.65	99	60	-27	10.00	30	29	8	2021-08-17 06:39:55
21	5.87	750.23	211.65	99	60	-27	9.75	30	29	7	2021-08-17 06:39:21
21	5.87	750.23	211.65	99	60	-115	-11.50	30	29	6	2021-08-17 06:38:47
21	5.87	750.23	211.65	99	60	-115	-11.50	30	29	6	2021-08-17 06:38:13
16	4.47	750.23	211.65	99	60	-27	9.50	30	29	5	2021-08-17 06:37:40
12	3.35	750.23	211.65	99	60	-28	10.00	30	29	4	2021-08-17 06:37:06
12	3.35	750.23	211.65	99	60	-28	10.00	30	29	4	2021-08-17 06:36:32
12	3.35	750.23	211.65	99	60	-27	9.75	30	29	3	2021-08-17 06:35:58
4	1.12	750.23	211.65	99	60	-26	10.00	30	29	2	2021-08-17 06:35:24

Fuente: elaboración propia, empleando MIT App *Inventor* V2.

En este ejemplo del funcionamiento del servidor web se enmarca con un rectángulo rojo la trama recibida en la base de datos.

4.3.6. Pruebas de campo

En la siguiente sección se describe las actividades que se realizaron en las pruebas de campo.

4.3.6.1. Modificación del pluviómetro

Se contrato el servicio de una empresa de impresión 3D para la elaboración del cono y la pieza de acople del cono con el pluviómetro, tal como se muestra en la figura siguiente.

Figura 126. **Pluviómetro modificado**

Fuente: Ingenio Magdalena.

4.3.6.2. Instalación del nodo

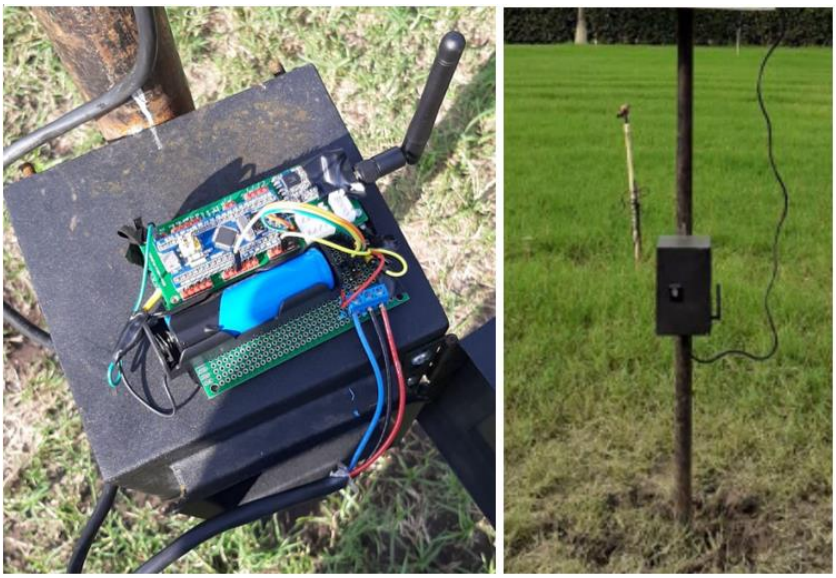
Luego se procedió a realizar las primeras pruebas en el edificio administrativo de Ingenio Magdalena oficinas costa, con apoyo del departamento del Recurso hídrico se fabricó y se instaló la base del nodo con su pluviómetro y su cenirrómetro.

Figura 127. **Instalación del nodo de enlace punto a punto**

Fuente: Ingenio Magdalena.

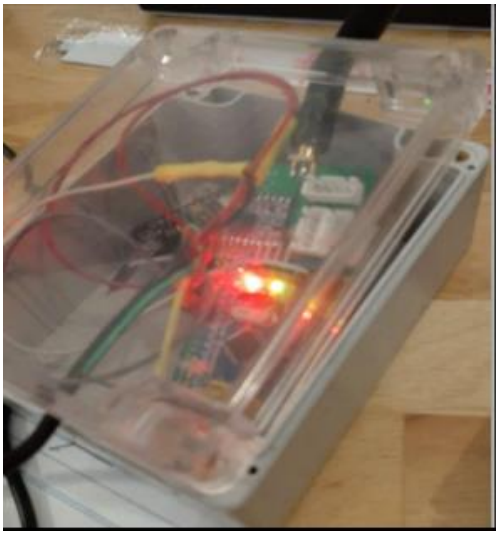
Posteriormente se procedió a colocar la electrónica del nodo dentro de la carcasa eléctrica de metal para protegerlo del ambiente.

Figura 128. **Nodo de pruebas**



Fuente: Ingenio Magdalena.

Figura 129. ***Gateway* de pruebas**



Fuente: Ingenio Magdalena.

Por último, el *Gateway* se colocó en una caja de derivación como se muestra en la figura anterior para realizar pruebas de conectividad del enlace LoRa tal como se muestra en la sección de resultados de pruebas del diseño.

CONCLUSIONES

1. La combinación de tecnologías de largo y corto alcance basadas en estándares existentes promueven la integración de Internet de las cosas en la agroindustria, porque los sensores de diferentes fabricantes pueden comunicarse dentro de los sistemas de automatización y control.

2. La implementación del diseño propuesto automatiza la lectura de datos de pluviómetros y cenirrómetros de las fincas más distantes, donde no hay señal celular 3G / 4G, de manera homogénea y también permite una reducción de costos de personal debido al hecho de que no se requiere ninguna intervención humana y también se reduce el número de tareas repetitivas.

3. La red de sensores inalámbricos de baja potencia es una solución de bajo costo para la transmisión y recopilación de datos en el campo de Internet de las cosas. Al conservar y limitar la energía utilizada por los dispositivos remotos, las redes inalámbricas de baja potencia han creado varias aplicaciones para mejorar la eficiencia, mejorando así la capacidad de recopilar datos al tiempo que se eliminan los errores humanos.

4. El protocolo de transferencia de hipertexto se utiliza en muchos servidores web del mundo y está integrado en todos los *Gateways* con conexiones *Wi-Fi* o Ethernet. Por lo tanto, el servidor web propuesto se puede implementar en cualquier *Gateway* que utilice el protocolo de internet, con capacidades mínimas de procesamiento.

5. La aplicación de Android puede mostrar información de la red de sensores tanto en modo *online* como fuera de línea, en caso de que alguna finca no pueda acceder a internet. La aplicación permite a los supervisores de la finca acceder a información en tiempo real para la programación del riego.

RECOMENDACIONES

1. Agregar medidas de seguridad adicionales para evitar ataques de interceptación de información cuando el diseño propuesto se implemente en una red pública.

2. Implementar la tecnología LoRaWAN en topología estrella cuando existan muchas fincas dentro de un radio de 10 km porque puede cubrir hasta 10 000 nodos usando un solo *Gateway*.

3. Fabricar las tarjetas de circuito impreso PCB con un proveedor que utiliza el método industrial, ya que el PCB tendrá acabados más profesionales y estéticos, además es protegido con distintas capas de pintura alargando el tiempo de vida útil del mismo.

4. Utilizar una caja de conexiones IP67 para resguardar las tarjetas electrónicas para evitar problemas de implementación ya que el diseño estará expuesto a extremos ambientales.

5. Capacitar al personal responsable de ensamblar las tarjetas de circuito impreso PCB para evitar daños y el desperdicio de materiales de construcción.

BIBLIOGRAFÍA

1. AERO IITB. *Modulacion y Demodulacion.* [en línea]. <https://www.aero.iitb.ac.in/satelliteWiki/index.php/Modulation_and_Demodulation>. [Consulta: 4 de septiembre de 2019].

2. ARGENT Data Systems. *Estación meteorológica.* [en línea]. <https://www.argentdata.com/catalog/product_info.php?products_id=13>. [Consulta: 8 de julio de 2019].

3. BALANIS, Constantine A. *Antenna Theory: third edition analysis and design*. 3 ed. EEUU: John Wiley & Sons, 2012. p. 1136. ISBN 1118585739.

4. CAE. *Conversión Analógica*. [en línea]. <https://controlautomaticoeducacion.com>. [Consulta: 8 de enero de 2021].

5. CD TECNOLOGÍA. *Sensor de precipitación balancin.* [en línea]. <https://cdtechnologia.net/>. [Consulta: 3 de mayo de 2019].

6. CDN Shop. *eTape*. [en línea]. <https://cdn-shop.adafruit.com/datasheets/*eTape*App.pdf->. [Consulta: 8 de enero de 2020].

7. CULTURA SONORA. *tipos de bluetooth y diferencias*. [en línea]. < https://www.culturasonora.es/blog/tipos-de-bluetooth-y-diferencias/ >. [Consulta: 14 de octubre de 2021].

8. DAZA, Lennyn. *Beyond the internet of things: everything interconnected: technology, communications and computing*. [en línea]. <https://www.semanticscholar.org/paper/Beyond-the-internet-of-things%3A-everything-and-%5Bbook-Daza-Misra/a6673b6613a3ba9d342232be7d1830317a070471>. [Consulta: 12 de octubre de 2019].

9. DEFINICIÓN. *bluetooth*. [en línea]. < https://definicion.de >. [Consulta: 14 de octubre de 2021].

10. DWM Zone. *SX1276 LoRa Module*. [en línea]. <https://dwmzone.com/en/lora-wireless-modules>. [Consulta: 13 de octubre de 2019].

11. EBAY. *Controlador SSD1306.* [en línea]. <https://www.ebay.com/itm/293570812029?ul_noapp=true>. [Consulta: 8 de octubre de 2019].

12. ELECTRÓNICA Estudio. *Módulo HC-05.* [en línea]. <https://www.electronicaestudio.com/>. [Consulta: 8 de octubre de 2019].

13. ELECTRÓNICOS CALDAS. *Sensor (eTape)*. [en línea]. <https://www.electronicoscaldas.com>. [Consulta: 12 de enero de 2020].

14. ENERGYMASTER ¿Para qué sirven los paneles solares? [en línea]. <https://energymaster.co/sirven-los-paneles-solares/>. [Consulta: 9 de julio de 2019].

15. GARIN, Dante; HAZARD, Mario. *Bluetooth*. [en línea]. <http://profesores.elo.utfsm.cl/>. [Consulta: 2 de noviembre de 2019].

16. HYPERPHYSICS. *Tipos de polarización.* [en línea]. <http://hyperphysics.phyastr.gsu.edu/hbasees/phyopt/polclas.html>. [Consulta: 3 de octubre 2019].

17. IP *Market*. *Evaporímetro To19-TEVAP.SIAP*. [en línea]. <https://ipmarket.cl/producto/evaporimetro-t019->. [Consulta: 8 de mayo de 2019].

18. ITRAINONLINE. *Cálculo de radioenlace guía.* [en línea]. < http://www.itrainonline.org/itrainonline/mmtk/wireless_es/files/06_es_calculo-de-radioenlace_guia_v02.pdf>. [Consulta: 9 de julio de 2019].

19. KLEINROCK, L.; TOBAGI, F. *Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics*. [en línea]. <https://scirp.org/reference/referencespapers.aspx?referenceid=807597>. [Consulta: 12 de octubre de 2019].

20. MAGNO, Michele. *WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication*. [en línea].

<https://www.researchgate.net/publication/316948751_WULoRa_An_energy_efficient_IoT_end-node_for_energy_harvesting_and_heterogeneous_communication>. [Consulta: 12 de octubre de 2019].

21. MOBILE FISH. *Zona Fresnel*. [en línea]. <https://www.mobilefish.com>. [Consulta: 3 de septiembre de 2019].

22. MOUSER Electronics. *Tiny Circuits ASR00050.* [en línea]. <https://www.mouser.com.gt/ProductDetail/TinyCircuits/ASR00050?qs=byeeYqUIh0Nbpw7ccioDew==>. [Consulta: 8 de octubre de 2019].

23. NOREEN, Umber; BOUNCEUR, Ahcéne; CLAVIER, Laurent. *A study of LoRa low power and wide area network technology.* [en línea]. <https://ieeexplore.ieee.org/document/8075570>. [Consulta: 12 de octubre de 2019].

24. PBS. *Nodo LoRa*. [en línea]. <https://pbs.twimg.com>. [Consulta: 2 de septiembre de 2019].

25. PDA CONTROLES. *ISM*. [en línea]. <http://pdacontroles.com>. [Consulta: 3 de septiembre de 2019].

26. REYNDERS, Brecht; MEERT, Wannes; POLLIN, Sofie. *Range and coexistence analysis of long-range unlicensed communication*. [en línea]. <https://ieeexplore.ieee.org/document/7500415>. [Consulta: 14 de octubre de 2019].

27. RUESCA, Pedro. *¿Qué es un radioenlace?* [en línea]. <http://www.radiocomunicaciones.net/radio/radio-enlace-que-es-un-radioenlace/>. [Consulta: 8 de noviembre de 2019].

28. SANDOROBOTICS. *Módulo de carga con circuito de protección.* [en línea]. <https://sandorobotics.com/producto/hr0140-1/>. [Consulta: 9 de julio de 2019].

29. SEEED. *Panel Solar.* [en línea]. <https://www.seeedstudio.com/1W-Solar-Panel-80X100.html>. [Consulta: 12 de octubre 2019].

30. SEEED. *Small Solar Panel.* [en línea]. <https://www.seeedstudio.com/1W-Solar-Panel-80X100.html>. [Consulta: 9 de julio de 2019].

31. SEMTECH. *LoRa Applications*. [en línea]. <https://www.semtech.com>. [Consulta: 14 de octubre de 2019].

32. SEMTECH. *SX1276-7-8-9 Datasheet.* [en línea]. < https://www.semtech.com/products/wireless-rf/lora-core/sx1276>. [Consulta: 9 de octubre de 2021].

33. SPARKFUN. *Electronics.* [en línea]. <https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly.pdf>. [Consulta: 8 de enero de 2020].

34. SPARKFUN. *Pluvímetro*. [en línea]. <https://www.sparkfun.com/products/8942>. [Consulta: 3 de mayo de 2019].

35. THE THINGS NETWORK. *Gateway de un solo canal.* [en línea]. <https://www.thethingsnetwork.org>. [Consulta: 2 de septiembre de 2019].

36. TINDIE. *STM32 Blue Pill LoRaWAN node.* [en línea]. <https://www.tindie.com/products/m2m/stm32-blue-pill-lorawan-node/>. [Consulta: 15 de enero 2020].

37. UTI. *Propagación por difracción.* [en línea]. <https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.526-15-201910-I!!PDF-S.pdf>. [Consulta: 12 de noviembre de 2019].

38. VATCHARATIANSAKUL, Nuttakit; TUWANUT, Panwit; PORNAVALAI, hotipat. *Experimental performance evaluation of LoRaWAN: A case study in Bangkok.* [en línea]. <https://www.researchgate.net/publication/319591516_Experimental_performance_evaluation_of_LoRaWAN_A_case_study_in_Bangkok>. [Consulta: 12 de octubre de 2019].

39. WAYER, Fayer. *La historia del nacimiento de Bluetooth*. [en línea]. <https://www.fayerwayer.com/2011/09/la-historia-del-nacimiento-de-bluetooth/>. [Consulta: 2 de noviembre de 2019].

40. WEATHER INNOVATIONS. *Interno del pluvímetro.* [en línea]. <https://www.weatherinnovations.com/technology.cfm>. [Consulta: 3 de julio de 2019].

APÉNDICES

Apéndice 1. **Código fuente del nodo**

```
#include <SPI.h>
#include "LoRa_STM32.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
// Declaración de variables de comunicación LoRa
#define TX_P 17
#define BAND 915E6
#define SFactor 12
String outgoing;
String LoRaMessage = "";
byte msgCount = 0;
byte localAddress = 0x30;
byte destination = 0x29;
long lastSendTime = 0;
int interval = 10000;
bool acknowledge = false;
int contador = 0;
int initCounter = 0;
bool initialization = false;

// Declaración de variables de red de sensores.
volatile int clicks = 0;
float lamina;
int counter = 0;
int analogValor = 0;
float voltaje = 0;
int porcentaje = 0;
int evaporacionmm;
#define PIN_RAIN_GAUGE  PA15
```

Continuación del apéndice 1.

```
// Configuración de Valores de Evaporímetro
#define SERIES_RESISTOR             2690
#define SENSOR_PIN          PA2
#define ZERO_LAMINAMM_RESISTANCE    2631.40
#define CALIBRATION_RESISTANCE      750.00
#define CALIBRATION_LAMINAMM        211.00
// Configuración de Pantalla OLED
#define SCREEN_WIDTH                128
#define SCREEN_HEIGHT               64
//Declaracion de pantalla SSD1306 conectado mediante I2C (pines SDA, SCL)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {

 //Inicialización de SSD1306
if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
       SSD1306errormsg();
        for (;;);
    }
  // configuración de interrupción de pluviómetro
  pinMode(PIN_RAIN_GAUGE, INPUT_PULLUP);
  attachInterrupt(PIN_RAIN_GAUGE, countRainGauge, FALLING);
 // configuración de ADC de medidor de batería
    pinMode(PA3, INPUT_PULLDOWN);
 // método de inicialización de enlace punto a punto
  initSX1276();
  }
void loop() {

    if (initialization == true){
    while (contador < 5) {
      if (millis() - lastSendTime > interval) {
        measureAndDisplay();
        sendMessage(LoRaMessage);
        lastSendTime = millis();
        interval = 10000;
        contador++;
      }
      else {
```

Continuación del apéndice 1.

```
        if (contador > 0) {
        onReceive(LoRa.parsePacket());
        }
      }

      if (acknowledge == true) {
       contador = 0;
        break;
        }
    }
    }

    else{
      sleepmsg2();
      delay(900000);
      initCounter = 0;
      initSX1276();
    }


    if (acknowledge == true || contador > 4) {
      sleepAndRestart();
    }
  }

void sendMessage(String outgoing) {
    LoRa.beginPacket();
    LoRa.write(destination);
    LoRa.write(localAddress);
    LoRa.write(msgCount);
    LoRa.write(outgoing.length());
    LoRa.print(outgoing);
    LoRa.endPacket();
    msgCount++;
  }
```

Continuación del apéndice 1.

```
void onReceive(int packetSize) {
  if (packetSize == 0) return;
   int recipient = LoRa.read();
  byte sender = LoRa.read();
  byte incomingMsgId = LoRa.read();
  byte incomingLength = LoRa.read();

  String incoming = "";

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }

  if (incomingLength != incoming.length()) {
     lenghtmsg();
     acknowledge = false;
    return;
  }
  if (recipient != localAddress && recipient != 0xFF) {
      msgnotforme();
      acknowledge = false;
    return;
  }

  if (incoming != LoRaMessage) {
     acknowledgemsg();
     acknowledge = false;
    return;
  }
   else {
    successfulmsg();
    acknowledge = true;
  }
}
float readResistance(int pin, int seriesResistance) {
  float resistance = analogRead(pin);
  resistance = (4095.0 / resistance) - 1.0;
  return resistance;
}
```

Continuación del apéndice 1.

```
  float resistanceToLAMINAMM(float resistance, float zeroResistance, float
 calResistance, float calLAMINAMM) {
    if (resistance > zeroResistance || (zeroResistance - calResistance) ==
 0.0) {
      return 0.0;
    }
    float scale = (zeroResistance - resistance) / (zeroResistance - calRes
istance)
    return calLAMINAMM * scale;
  }
// metodos del programa
  void countRainGauge() {
    clicks++;
  }
  void initSX1276(){
    LoRa.setTxPower(TX_P);
    LoRa.setSpreadingFactor(SFactor);
    pinMode(PC13, OUTPUT);

        while (initCounter < 5) {
    if (!LoRa.begin(BAND)) {
            initfailed();
            initCounter++;
            delay(5000);
    }

    else {
            initsucceeded();
            initialization = true;
            break;
    }
  }
    if (initialization == false) {
      sleepmsg();
      LoRa.sleep();

    }
  }
```

Continuación del apéndice 1.

```
  void sleepAndRestart() {
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 32);
    display.print("Ingresando a sleep mode por 15 minutos...");
    display.display();
    delay(900000);
    acknowledge = false;
    contador = 0;
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 32);
    display.print("Despertando despues de 15 minutos, enviando nuevo mensa
je...");
    display.display();
    delay(2000);
  }

  void measureAndDisplay() {
    // Medicion de resistencia del sensor.
    float resistance = readResistance(SENSOR_PIN, SERIES_RESISTOR);
    // Map resistencia a LAMINAMM.
    float LAMINAMM = resistanceToLAMINAMM(resistance, ZERO_LAMINAMM_RESIST
ANCE, CALIBRATION_RESISTANCE, CALIBRATION_LAMINAMM);
    evaporacionmm = int(LAMINAMM);
    delay(1000);
    lamina = clicks * 0.2794;
    analogValor = analogRead(PA3);
    voltaje = 0.001025390 * analogValor;
    porcentaje = (voltaje - 3.2) * 100;

    // Visualizacion de Lamina de Pluviometro
    display.clearDisplay();
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print("Lluvia");
    display.setTextSize(2);
    display.setCursor(0, 10);
    display.print(lamina);
```

Continuación del apéndice 1.

```
    display.setTextSize(1);
    display.print("mm");

    // Visualizacion de Lamina de Cenirrometro
    display.setTextSize(1);
    display.setCursor(60, 0);
    display.print("Evaporacion");

    display.setTextSize(2);
    display.setCursor(64, 10);
    display.print(evaporacionmm);
    display.setTextSize(1);
    display.print("mm");

    // Visualizacion de Porcentaje de bateria Nodo
    display.setTextSize(1);
    display.setCursor(0, 35);
    display.print("Bateria %");
    display.setTextSize(2);
    display.setCursor(0, 45);
    display.print(porcentaje);

    // Validacion de longitud de trama
    LoRaMessage = String(clicks) + "/" + String(lamina) + "&" + String(res
istance) + "#" + String(LAMINAMM) + "@" + String(porcentaje);

    // Longitud de Trama
    display.setTextSize(1);
    display.setCursor(64, 35);
    display.print("Long.Trama");
    display.setTextSize(2);
    display.setCursor(64, 45);
    display.print(LoRaMessage.length());
    display.display();
    // indicador de envio exitoso de trama
    digitalWrite(PC13, LOW);
    delay(2000);
    digitalWrite(PC13, HIGH);
    delay(2000);
}
```

Continuación del apéndice 1.

```
void SSD1306errormsg(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0, 32);
        display.print("Fallo en la Asignacion de SSD1306");
        display.display();
        delay(2000);
  }

 void initfailed(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0, 32);
        display.print("Fallo en el inicio de LoRa, reintentando...");
        display.display();
        delay(2000);
  }

 void initsucceeded(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0, 32);
        display.print("LoRa inicializado con éxito");
        display.display();
        delay(2000);
  }

 void sleepmsg(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0, 32);
        display.print("El inicio de LoRa falló después de 5 intentos, entr
ando en sleep mode");
        display.display();
        delay(2000);
  }
```

Continuación del apéndice 1.

```
void sleepmsg2(){
      display.clearDisplay();
      display.setTextSize(1);
      display.setCursor(0, 32);
      display.print("La inicialización de LoRa ha fallado, espera 15 minut
os para volver a intentarlo");
      display.display();
      delay(2000);
}

void lenghtmsg() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setCursor(0, 32);
      display.print("error: la longitud del mensaje no coincide con la lon
gitud");
      display.display();
      delay(2000);
}

void msgnotforme() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setCursor(0, 32);
      display.print("Este mensaje no es para mi");

      display.display();
      delay(2000);
    }

void acknowledgemsg() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setCursor(0, 32);
      display.print("El mensaje recibido no es el mismo");
      display.display();
      delay(2000);
}
```

Continuación del apéndice 1.

```
void successfulmsg() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setCursor(0, 32);
      display.print("la trama ha sido enviado con éxito");
      display.display();
      delay(2000);
}
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Apéndice 2. **Código fuente de librería local LoRa_STM32.cpp**

```
#include "LoRa_STM32.h"

// registers
#define REG_FIFO                0x00
#define REG_OP_MODE             0x01
#define REG_FRF_MSB             0x06
#define REG_FRF_MID             0x07
#define REG_FRF_LSB             0x08
#define REG_PA_CONFIG           0x09
#define REG_LNA                 0x0c
#define REG_FIFO_ADDR_PTR       0x0d
#define REG_FIFO_TX_BASE_ADDR   0x0e
#define REG_FIFO_RX_BASE_ADDR   0x0f
#define REG_FIFO_RX_CURRENT_ADDR 0x10
#define REG_IRQ_FLAGS           0x12
#define REG_RX_NB_BYTES         0x13
#define REG_PKT_SNR_VALUE       0x19
#define REG_PKT_RSSI_VALUE      0x1a
#define REG_MODEM_CONFIG_1      0x1d
#define REG_MODEM_CONFIG_2      0x1e
#define REG_PREAMBLE_MSB        0x20
#define REG_PREAMBLE_LSB        0x21
#define REG_PAYLOAD_LENGTH      0x22
#define REG_MODEM_CONFIG_3      0x26
```

Continuación del apéndice 2.

```
#define REG_RSSI_WIDEBAND         0x2c
#define REG_DETECTION_OPTIMIZE    0x31
#define REG_DETECTION_THRESHOLD   0x37
#define REG_SYNC_WORD             0x39
#define REG_DIO_MAPPING_1         0x40
#define REG_VERSION               0x42

// modes
#define MODE_LONG_RANGE_MODE      0x80
#define MODE_SLEEP                0x00
#define MODE_STDBY                0x01
#define MODE_TX                   0x03
#define MODE_RX_CONTINUOUS        0x05
#define MODE_RX_SINGLE            0x06

// PA config
#define PA_BOOST                  0x80

// IRQ masks
#define IRQ_TX_DONE_MASK            0x08
#define IRQ_PAYLOAD_CRC_ERROR_MASK 0x20
#define IRQ_RX_DONE_MASK            0x40

#define MAX_PKT_LENGTH            255

LoRaClass::LoRaClass() :
  _spiSettings(8E6, MSBFIRST, SPI_MODE0),
  _ss(LORA_DEFAULT_SS_PIN), _reset(LORA_DEFAULT_RESET_PIN), _dio0(LORA_DEF
AULT_DIO0_PIN),
  _frequency(0),
  _packetIndex(0),
  _implicitHeaderMode(0),
  _onReceive(NULL)
{
  // overide Stream timeout value
  setTimeout(0);
}
int LoRaClass::begin(long frequency)
{
```

Continuación del apéndice 2.

```
  // setup pins
  pinMode(_ss, OUTPUT);

  // set SS high
  digitalWrite(_ss, HIGH);

  if (_reset != -1) {
    pinMode(_reset, OUTPUT);

    // perform reset
    digitalWrite(_reset, LOW);
    delay(10);
    digitalWrite(_reset, HIGH);
    delay(10);
  }

  // start SPI
  SPI.begin();

  // check version
  uint8_t version = readRegister(REG_VERSION);
  if (version != 0x12) {
    return 0;
  }

  // put in sleep mode
  sleep();

  // set frequency
  setFrequency(frequency);

  // set base addresses
  writeRegister(REG_FIFO_TX_BASE_ADDR, 0);
  writeRegister(REG_FIFO_RX_BASE_ADDR, 0);

  // set LNA boost
  writeRegister(REG_LNA, readRegister(REG_LNA) | 0x03);
```

Continuación del apéndice 2.

```
  // set auto AGC
  writeRegister(REG_MODEM_CONFIG_3, 0x04);

  // set output power to 17 dBm
  setTxPower(20);
  // put in standby mode
  idle();

  return 1;
}

void LoRaClass::end()
{
  // put in sleep mode
  sleep();

  // stop SPI
  SPI.end();
}

int LoRaClass::beginPacket(int implicitHeader)
{
  // put in standby mode
  idle();

  if (implicitHeader) {
    implicitHeaderMode();
  } else {
    explicitHeaderMode();
  }

  // reset FIFO address and paload length
  writeRegister(REG_FIFO_ADDR_PTR, 0);
  writeRegister(REG_PAYLOAD_LENGTH, 0);

  return 1;
}
int LoRaClass::endPacket()
{
```

Continuación del apéndice 2.

```
  // put in TX mode
  writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_TX);

  // wait for TX done
  while ((readRegister(REG_IRQ_FLAGS) & IRQ_TX_DONE_MASK) == 0) {
    yield();

  }

  // clear IRQ's
  writeRegister(REG_IRQ_FLAGS, IRQ_TX_DONE_MASK);

  return 1;
}

int LoRaClass::parsePacket(int size)
{
  int packetLength = 0;
  int irqFlags = readRegister(REG_IRQ_FLAGS);

  if (size > 0) {
    implicitHeaderMode();

    writeRegister(REG_PAYLOAD_LENGTH, size & 0xff);
  } else {
    explicitHeaderMode();
  }

  // clear IRQ's
  writeRegister(REG_IRQ_FLAGS, irqFlags);

  if ((irqFlags & IRQ_RX_DONE_MASK) && (irqFlags & IRQ_PAYLOAD_CRC_ERROR_M
ASK) == 0) {
    // received a packet
    _packetIndex = 0;

    // read packet length
    if (_implicitHeaderMode) {
      packetLength = readRegister(REG_PAYLOAD_LENGTH);
```

Continuación del apéndice 2.

```
    } else {
      packetLength = readRegister(REG_RX_NB_BYTES);
    }

    // set FIFO address to current RX address
    writeRegister(REG_FIFO_ADDR_PTR, readRegister(REG_FIFO_RX_CURRENT_ADDR
));

    // put in standby mode
    idle();
  } else if (readRegister(REG_OP_MODE) != (MODE_LONG_RANGE_MODE | MODE_RX_
SINGLE)) {
    // not currently in RX mode

    // reset FIFO address
    writeRegister(REG_FIFO_ADDR_PTR, 0);

    // put in single RX mode
    writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_RX_SINGLE);
  }

  return packetLength;
}

int LoRaClass::packetRssi()
{
  return (readRegister(REG_PKT_RSSI_VALUE) - (_frequency < 868E6 ? 164 : 1
57));
}

float LoRaClass::packetSnr()
{
  return ((int8_t)readRegister(REG_PKT_SNR_VALUE)) * 0.25;
}

size_t LoRaClass::write(uint8_t byte)
{
  return write(&byte, sizeof(byte));
}
```

Continuación del apéndice 2.

```
size_t LoRaClass::write(const uint8_t *buffer, size_t size)
{
  int currentLength = readRegister(REG_PAYLOAD_LENGTH);

  // check size
  if ((currentLength + size) > MAX_PKT_LENGTH) {
    size = MAX_PKT_LENGTH - currentLength;
  }

  // write data
  for (size_t i = 0; i < size; i++) {
    writeRegister(REG_FIFO, buffer[i]);
  }

  // update length
  writeRegister(REG_PAYLOAD_LENGTH, currentLength + size);

  return size;
}

int LoRaClass::available()
{
  return (readRegister(REG_RX_NB_BYTES) - _packetIndex);
}

int LoRaClass::read()
{
  if (!available()) {
    return -1;
  }

  _packetIndex++;

  return readRegister(REG_FIFO);
}

int LoRaClass::peek()
{
  if (!available()) {
```

Continuación del apéndice 2.

```
    return -1;
  }

  // store current FIFO address
  int currentAddress = readRegister(REG_FIFO_ADDR_PTR);

  // read
  uint8_t b = readRegister(REG_FIFO);

  // restore FIFO address
  writeRegister(REG_FIFO_ADDR_PTR, currentAddress);

  return b;
}

void LoRaClass::flush()
{
}

void LoRaClass::onReceive(void(*callback)(int))
{
  _onReceive = callback;

  if (callback) {
    writeRegister(REG_DIO_MAPPING_1, 0x00);

    attachInterrupt(_dio0, LoRaClass::onDio0Rise, RISING);
  } else {
    detachInterrupt(_dio0);
  }
}

void LoRaClass::receive(int size)
{
  if (size > 0) {
    implicitHeaderMode();

    writeRegister(REG_PAYLOAD_LENGTH, size & 0xff);
  } else {
```

Continuación del apéndice 2.

```
    explicitHeaderMode();
  }

  writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_RX_CONTINUOUS);
}

void LoRaClass::idle()
{
  writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_STDBY);
}

void LoRaClass::sleep()
{
  writeRegister(REG_OP_MODE, MODE_LONG_RANGE_MODE | MODE_SLEEP);
}

void LoRaClass::setTxPower(int level, int outputPin)
{
  if (PA_OUTPUT_RFO_PIN == outputPin) {
    // RFO
    if (level < 0) {
      level = 0;
    } else if (level > 14) {
      level = 14;
    }

    writeRegister(REG_PA_CONFIG, 0x70 | level);
  } else {
    // PA BOOST
    if (level < 2) {
      level = 2;
    } else if (level > 17) {
      level = 17;
    }

    writeRegister(REG_PA_CONFIG, PA_BOOST | (level - 2));
  }
}
```

Continuación del apéndice 2.

```
void LoRaClass::setFrequency(long frequency)
{
  _frequency = frequency;

  uint64_t frf = ((uint64_t)frequency << 19) / 32000000;

  writeRegister(REG_FRF_MSB, (uint8_t)(frf >> 16));
  writeRegister(REG_FRF_MID, (uint8_t)(frf >> 8));
  writeRegister(REG_FRF_LSB, (uint8_t)(frf >> 0));
}

void LoRaClass::setSpreadingFactor(int sf)

{
  if (sf < 6) {
    sf = 6;
  } else if (sf > 12) {
    sf = 12;
  }

  if (sf == 6) {
    writeRegister(REG_DETECTION_OPTIMIZE, 0xc5);
    writeRegister(REG_DETECTION_THRESHOLD, 0x0c);
  } else {
    writeRegister(REG_DETECTION_OPTIMIZE, 0xc3);
    writeRegister(REG_DETECTION_THRESHOLD, 0x0a);
  }

  writeRegister(REG_MODEM_CONFIG_2, (readRegister(REG_MODEM_CONFIG_2) & 0x
0f) | ((sf << 4) & 0xf0));
}

void LoRaClass::setSignalBandwidth(long sbw)
{
  int bw;

  if (sbw <= 7.8E3) {
    bw = 0;
  } else if (sbw <= 10.4E3) {
```

Continuación del apéndice 2.

```
    bw = 1;
  } else if (sbw <= 15.6E3) {
    bw = 2;
  } else if (sbw <= 20.8E3) {
    bw = 3;
  } else if (sbw <= 31.25E3) {
    bw = 4;
  } else if (sbw <= 41.7E3) {
    bw = 5;
  } else if (sbw <= 62.5E3) {
    bw = 6;
  } else if (sbw <= 125E3) {
    bw = 7;
  } else if (sbw <= 250E3) {
    bw = 8;
  } else /*if (sbw <= 250E3)*/ {
    bw = 9;
  }

  writeRegister(REG_MODEM_CONFIG_1, (readRegister(REG_MODEM_CONFIG_1) & 0x
0f) | (bw << 4));
}

void LoRaClass::setCodingRate4(int denominator)
{
  if (denominator < 5) {
    denominator = 5;
  } else if (denominator > 8) {
    denominator = 8;
  }

  int cr = denominator - 4;

  writeRegister(REG_MODEM_CONFIG_1, (readRegister(REG_MODEM_CONFIG_1) & 0x
f1) | (cr << 1));
}

void LoRaClass::setPreambleLength(long length)
{
```

Continuación del apéndice 2.

```
  writeRegister(REG_PREAMBLE_MSB, (uint8_t)(length >> 8));
  writeRegister(REG_PREAMBLE_LSB, (uint8_t)(length >> 0));
}

void LoRaClass::setSyncWord(int sw)
{
  writeRegister(REG_SYNC_WORD, sw);
}

void LoRaClass::enableCrc()
{
  writeRegister(REG_MODEM_CONFIG_2, readRegister(REG_MODEM_CONFIG_2) | 0x0
4);
}

void LoRaClass::disableCrc()
{
  writeRegister(REG_MODEM_CONFIG_2, readRegister(REG_MODEM_CONFIG_2) & 0xf
b);
}

byte LoRaClass::random()
{
  return readRegister(REG_RSSI_WIDEBAND);
}

void LoRaClass::setPins(int ss, int reset, int dio0)
{
  _ss = ss;
  _reset = reset;
  _dio0 = dio0;
}

void LoRaClass::setSPIFrequency(uint32_t frequency)
{
  _spiSettings = SPISettings(frequency, MSBFIRST, SPI_MODE0);
}

void LoRaClass::dumpRegisters(Stream& out)
```

Continuación del apéndice 2.

```
{
  for (int i = 0; i < 128; i++) {
    out.print("0x");
    out.print(i, HEX);
    out.print(": 0x");
    out.println(readRegister(i), HEX);
  }
}

void LoRaClass::explicitHeaderMode()
{
  _implicitHeaderMode = 0;

  writeRegister(REG_MODEM_CONFIG_1, readRegister(REG_MODEM_CONFIG_1) & 0xf
e);
}

void LoRaClass::implicitHeaderMode()
{
  _implicitHeaderMode = 1;

  writeRegister(REG_MODEM_CONFIG_1, readRegister(REG_MODEM_CONFIG_1) | 0x0
1);
}

void LoRaClass::handleDio0Rise()
{
  int irqFlags = readRegister(REG_IRQ_FLAGS);

  // clear IRQ's
  writeRegister(REG_IRQ_FLAGS, irqFlags);

  if ((irqFlags & IRQ_PAYLOAD_CRC_ERROR_MASK) == 0) {
    // received a packet
    _packetIndex = 0;

    // read packet length
    int packetLength = _implicitHeaderMode ? readRegister(REG_PAYLOAD_LENG
TH) : readRegister(REG_RX_NB_BYTES);
```

Continuación del apéndice 2.

```
    // set FIFO address to current RX address
    writeRegister(REG_FIFO_ADDR_PTR, readRegister(REG_FIFO_RX_CURRENT_ADDR
));

    if (_onReceive) {
      _onReceive(packetLength);
    }

    // reset FIFO address
    writeRegister(REG_FIFO_ADDR_PTR, 0);
  }
}

uint8_t LoRaClass::readRegister(uint8_t address)
{
  return singleTransfer(address & 0x7f, 0x00);
}

void LoRaClass::writeRegister(uint8_t address, uint8_t value)
{
  singleTransfer(address | 0x80, value);
}

uint8_t LoRaClass::singleTransfer(uint8_t address, uint8_t value)
{
  uint8_t response;

  digitalWrite(_ss, LOW);

  SPI.beginTransaction(_spiSettings);
  SPI.transfer(address);
  response = SPI.transfer(value);
  SPI.endTransaction();

  digitalWrite(_ss, HIGH);

  return response;
}
```

```
void LoRaClass::onDio0Rise()
{
  LoRa.handleDio0Rise();
}

LoRaClass LoRa;
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Apéndice 3. **Código librería local LoRa_STM32.h**

```
#ifndef LORA_H
#define LORA_H

#include <Arduino.h>
#include <SPI.h>

#define LORA_DEFAULT_SS_PIN    PA4
#define LORA_DEFAULT_RESET_PIN 0xFF
#define LORA_DEFAULT_DIO0_PIN  PA0

#define PA_OUTPUT_RFO_PIN      0
#define PA_OUTPUT_PA_BOOST_PIN 1

class LoRaClass : public Stream {
  public:
    LoRaClass();

    int begin(long frequency);
    void end();

    int beginPacket(int implicitHeader = false);
    int endPacket();

    int parsePacket(int size = 0);
    int packetRssi();
    float packetSnr();

    // from Print
    virtual size_t write(uint8_t byte);
```

Continuación del apéndice 3.

```
    virtual size_t write(const uint8_t *buffer, size_t size);

    // from Stream
    virtual int available();
    virtual int read();
    virtual int peek();
    virtual void flush();

    void onReceive(void(*callback)(int));
    void receive(int size = 0);
    void idle();
    void sleep();

    void setTxPower(int level, int outputPin = PA_OUTPUT_PA_BOOST_PIN);
    void setFrequency(long frequency);
    void setSpreadingFactor(int sf);
    void setSignalBandwidth(long sbw);
    void setCodingRate4(int denominator);
    void setPreambleLength(long length);
    void setSyncWord(int sw);
    void enableCrc();
    void disableCrc();

    // deprecated
    void crc() {
      enableCrc();
    }
    void noCrc() {
      disableCrc();
    }

    byte random();

    void setPins(int ss = LORA_DEFAULT_SS_PIN, int reset = LORA_DEFAULT_RE
SET_PIN, int dio0 = LORA_DEFAULT_DIO0_PIN);
    void setSPIFrequency(uint32_t frequency);

    void dumpRegisters(Stream& out);
```

Continuación del apéndice 3.

```
  private:
    void explicitHeaderMode();
    void implicitHeaderMode();

    void handleDio0Rise();

    uint8_t readRegister(uint8_t address);
    void writeRegister(uint8_t address, uint8_t value);
    uint8_t singleTransfer(uint8_t address, uint8_t value);

    static void onDio0Rise();

  private:
    SPISettings _spiSettings;
    int _ss;
    int _reset;
    int _dio0;
    int _frequency;
    int _packetIndex;
    int _implicitHeaderMode;
    void (*_onReceive)(int);
};

extern LoRaClass LoRa;

#endif
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Apéndice 4. **Código fuente del *Gateway***

```
#include <SPI.h>
#include "LoRa_STM32.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Configuracion de Pantalla OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

// Declaracion de variables de comunicacion LoRa
#define TX_P 17
#define BAND 915E6
#define SFactor 12
String incoming = "";
byte msgCount = 0;
byte localAddress = 0x29;
byte destination = 0x30;
long lastSendTime = 0;
int interval = 10000;
bool validated = false;


String LoRaData = "";
String LoRaMessage = "";
int recipient;
byte sender;
byte incomingMsgId;
byte incomingLength;
int initCounter = 0;
bool initialization = false;
int analogValor = 0;
float voltaje = 0;
int porcentajegateway = 0;

// Declaracion de variables de interpretacion de la trama entrante
int pos1, pos2, pos3, pos4;
String clicks;
String lamina;
```

Continuación del apéndice 4.

```
String resistance;
String mm_evaporacion;
String porcentaje;

//Declaracion de pantalla SSD1306 conectado mediante I2C (pines SDA, SCL)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
   Serial1.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    SSD1306errormsg();
    for (;;);
  }
  delay(2000);
  // configuracion de ADC de medidor de bateria

  pinMode(PA3, INPUT_PULLDOWN);
  pinMode(PC13, OUTPUT);
  initSX1276();
}



void loop() {
  if (initialization == true) {
    if (validated == true) {
      parseAndDisplay();
      sendMessage(incoming);
      //trama a enviar a aplicacion android
      LoRaMessage = String(clicks) + "," + String(lamina) + "," + String(r
esistance) + ","
      + String(mm_evaporacion) + "," + String(porcentaje) + "," + String(p
orcentajegateway) + ","
      + String(LoRa.packetRssi())+ "," + String(LoRa.packetSnr())+ ","
      + String(sender, HEX) + "," + String(recipient, HEX)+ "," + String(i
ncomingMsgId);
```

Continuación del apéndice 4.

```
      Serial1.println(LoRaMessage);
      validated = false;
      LoRa.receive();

    }
  } else {
    void sleepmsg2();
    delay(900000);
    initCounter = 0;
    initSX1276();
  }
}

void sendMessage(String outgoing) {
  LoRa.beginPacket();
  LoRa.write(destination);
  LoRa.write(localAddress);
  LoRa.write(msgCount);
  LoRa.write(outgoing.length());
  LoRa.print(outgoing);
  LoRa.endPacket();
  msgCount++;
}

void onReceive(int packetSize) {
  if (packetSize == 0) return;


  incoming = "";
  // leer los bytes de la cabecera del paquete:
  recipient = LoRa.read();
  sender = LoRa.read();
  incomingMsgId = LoRa.read();
  incomingLength = LoRa.read();

  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }
```

Continuación del apéndice 4.

```
  if (incomingLength != incoming.length()) {
    lenghtmsg();
    return;
  }

  // if the recipient isn't this device or broadcast,
  if (recipient != localAddress && recipient != 0xFF) {
    msgnotforme();
    return;
  }
  successfulmsg();
  validated = true;
}

void initSX1276() {
  LoRa.setTxPower(TX_P);
  LoRa.setSpreadingFactor(SFactor);
  while (initCounter < 5) {
    if (!LoRa.begin(BAND)) {
      initfailed();
      initCounter++;
      delay(5000);
    } else {
      LoRa.onReceive(onReceive);
      LoRa.receive();
      initsucceeded();
      initialization = true;
      break;

    }
  }
  if (initialization == false) {
    sleepmsg();
    LoRa.sleep();

  }
}
```

Continuación del apéndice 4.

```
void parseAndDisplay() {
  pos1 = incoming.indexOf('/');
  pos2 = incoming.indexOf('&');
  pos3 = incoming.indexOf('#');
  pos4 = incoming.indexOf('@');

  clicks = incoming.substring(0, pos1);
  lamina = incoming.substring(pos1 + 1, pos2);
  resistance = incoming.substring(pos2 + 1, pos3);
  mm_evaporacion = incoming.substring(pos3 + 1, pos4);
  porcentaje = incoming.substring(pos4 + 1, incoming.length());
  analogValor = analogRead(PA3);
  voltaje = 0.001025390 * analogValor;
  porcentajegateway = (voltaje - 3.2) * 100;

  display.clearDisplay();

  // RSSI
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.print("RSSI");
  display.setTextSize(2);
  display.setCursor(0, 10);
  display.print(LoRa.packetRssi());
  display.setTextSize(1);
  display.print("dBm");

  // SNR
  display.setTextSize(1);
  display.setTextColor(WHITE);

  display.setCursor(60, 0);
  display.print("SNR");

  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(56, 10);
  display.print(String(LoRa.packetSnr()));
```

Continuación del apéndice 4.

```
  display.setTextSize(1);
  display.print("dB");


  // Porcentaje de Bateria
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 35);
  display.print("Bateria %");
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 45);
  display.print(porcentajegateway);

// Longitud de Trama
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(64, 35);
  display.print("Long.Trama");
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(64, 45);
  display.print(incoming.length());
  display.display();

  // led indicador
  digitalWrite(PC13, LOW);
  delay(2000);
  digitalWrite(PC13, HIGH);
  delay(2000);

}


void SSD1306errormsg(){
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0, 32);
```

Continuación del apéndice 4.

```
        display.print("Fallo en la Asignacion de SSD1306");
        display.display();
        delay(2000);
  }

void initfailed() {
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0, 32);
        display.print("Fallo en el inicio de LoRa, reintentando...");
        display.display();
        delay(2000);
  }

 void initsucceeded() {
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0, 32);
        display.print("LoRa inicializado con éxito");
        display.display();
        delay(2000);
  }

  void sleepmsg() {
        display.clearDisplay();
        display.setTextSize(1);
        display.setTextColor(WHITE);
        display.setCursor(0, 32);
        display.print("El inicio de LoRa falló después de 5 intentos, entr
ando en sleep mode");
        display.display();
        delay(2000);
  }

void sleepmsg2(){
      display.clearDisplay();
      display.setTextSize(1);
```

Continuación del apéndice 4.

```
      display.setTextColor(WHITE);
      display.setCursor(0, 32);
      display.print("La inicialización de LoRa ha fallado, espera 15 minut
os para volver a intentarlo");
      display.display();
      delay(2000);
}
void lenghtmsg() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE);
      display.setCursor(0, 32);
      display.print("error: la longitud del mensaje no coincide con la lon
gitud");
      display.display();
      delay(2000);
}
void msgnotforme() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE);
      display.setCursor(0, 32);
      display.print("Este mensaje no es para mi");

      display.display();
      delay(2000);
    }
void successfulmsg() {
      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE);
      display.setCursor(0, 32);
      display.print("la trama ha sido enviado con éxito");
      display.display();

      delay(2000);
}
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.

Apéndice 5. **Código fuente del servidor web**

```
<?php
//Configuracion de Web Service
/**********CONFIGURACION************/
//DETALLES DE BASE DE DATOS//
$DB_ADDRESS="localhost";
$DB_USER="id16969620_app_inventor";
$DB_PASS="Guatemala2021*";
$DB_NAME="id16969620_registro_app";


//CONFIGURACIONES//

if( isset($_GET['clicks'])) {
    $_POST['clicks']=$_GET['nombre'];
    $_POST['mm_pluviometro']=$_GET['mm_pluviometro'];
    $_POST['ohms_evaporacion']=$_GET['ohms_evaporacion'];
    $_POST['mm_evaporacion']=$_GET['mm_evaporacion'];
    $_POST['bateria_nodo']=$_GET['bateria_nodo'];
    $_POST['bateria_gateway']=$_GET['bateria_nodo'];
    $_POST['rssi']=$_GET['rssi'];
    $_POST['snr']=$_GET['snr'];
    $_POST['sender_address']=$_GET['sender_address'];
    $_POST['recipient_address']=$_GET['recipient_address'];
    $_POST['msgID']=$_GET['msgID'];
}









/*********CONFIGURACION********/

//configuración de cabeceras
header('Cache-Control: no-cache, must-revalidate');
```

Continuación del apéndice 5.

```
error_log(print_r($_POST,TRUE));
 //comprueba si el post de la etiqueta está ahí y si ha sido un post de fo
rma adecuada
if( isset($_POST['clicks']) && isset($_POST['mm_pluviometro']) && isset($_
POST['ohms_evaporacion']) &&
isset($_POST['mm_evaporacion']) && isset($_POST['bateria_nodo']) && isset(
$_POST['bateria_gateway']) &&
isset($_POST['rssi']) && isset($_POST['snr']) && isset($_POST['sender_addr
ess']) &&
isset($_POST['recipient_address']) && isset($_POST['msgID']) ){

    //establecer el tipo de contenido a CSV
    //header('Content-type: text/csv');
    $query="INSERT INTO users (clicks, mm_pluviometro, ohms_evaporacion, m
m_evaporacion, bateria_nodo,
     bateria_gateway, rssi, snr, sender_address, recipient_address, msgID)
     VALUES ('".$_POST['clicks']."','".$_POST['mm_pluviometro']."','".$_PO
ST['ohms_evaporacion']."','"
     .$_POST['mm_evaporacion']."','".$_POST['bateria_nodo']."','".$_POST['
bateria_gateway']."','"
     .$_POST['rssi']."','".$_POST['snr']."','".$_POST['sender_address']."'
,'".$_POST['recipient_address']."','"
     .$_POST['msgID']."')";

     if(get_magic_quotes_gpc()){
      $query=stripslashes($query);
    }
    //Conexion
    $conn = new mysqli($DB_ADDRESS,$DB_USER,$DB_PASS,$DB_NAME);
    //Verificar conexion
    if($conn->connect_error){
      header("HTTP/1.0 400 Bad Request");
    //reportar a DB conecxion fallida
      echo "ERROR Database Connection Failed: " . $conn-
>connect_error, E_USER_ERROR;
    } else {
```

Continuación del apéndice 5.

```
    //ejecuta la consulta publicada
      $result=$conn-
>query($query);
      if($result === false){
    //devuelve un error de solicitud incorrecta
        header("HTTP/1.0 400 Bad Request");
    //si la consulta es incorrecta devuelve el error al cliente
        echo "Wrong SQL: " . $query . " Error: " . $conn-
>error, E_USER_ERROR;
      } else {
        echo "OK";
      }
      $conn->close();  //cierra la DB
    }
} else {
        header("HTTP/1.0 400 Bad Request");
        echo "Bad Request2";
}
?>
```

Fuente: elaboración propia, empleando Arduino ide 1.8.15.