ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA (FIEE)

DISEÑO DE UN NODO PROTOTIPO QUE OPERE CON GPS Y DETECTE MOVIMIENTO BASADO EN EL NODO LORA NODE 151.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

ALEX RUBÉN ZURITA CASTILLO

DIRECTOR: MSC. CARLOS ROBERTO EGAS

Quito, febrero 2022

CERTIFICACIONES

Certifico que el presente trabajo fue desarrollado por Alex Rubén Zurita Castillo, bajo mi supervisión.

MSC. CARLOS ROBERTO EGAS
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Alex Rubén Zurita Castillo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Alex Rubén Zurita Castillo

DEDICATORIA

El presente trabajo ve dedicado con mucho aprecio y gratitud a mi madre, Fanny Eulalia Castillo, quien nunca dejo de apoyarme en todas las etapas de mi vida, a pesar de las dificultades que en la vida me presentaba, a mis hermanos que fueron un apoyo y quienes ayudaron a motivarme de salir siempre adelante, a toda mi familia cercana que estuvieron presentes en momentos buenos y mejor aún en los malos.

AGRADECIMIENTO

Agradezco en primer lugar a Dios por tener el don de la vida, a mis padres, Jorge Rubén Zurita y Fanny Eulalia Castillo por haber permitido llegar a este mundo, por haberme apoyado cada uno en su momento con sus consejos, con su apoyo en lo académica y en lo personal para así lograr salir de la adversidad que la vida nos presenta día a día.

Agradezco a mis hermanos, a mi familia ya que unidos hemos logrado superarnos y tener un camino claro y tranquilo en nuestras vidas.

También agradezco a mis amigos que en cada etapa ellos han sabido darme una mano para cumplir este objetivo, también agradezco a todas las instituciones educativas en las cuales me supieron acoger y transmitir de la mejor manera todo el conocimiento necesario para lograr este objetivo académico.

De una manera especial agradezco a la Escuela Politécnica Nacional que me dio la oportunidad de ser parte de esta familia llena de gran sabiduría, con la misma intensidad agradezco al Msc. Carlos Roberto Egas director de la presente tesis, quien es un gran guía y la principal persona para que este proyecto se haga posible, compartiendo sus ideas, sabiduría para lograr este objetivo.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo de integración curricular realiza un diseño de un nodo prototipo que opera con *GPS* y detecta movimiento basado en el nodo Lora Node 151.

En el primer capítulo se recopila información de la tecnología *LoRaWAN*, para redes de larga distancia y bajo consumo de potencia. Se desarrolla un estudio del nodo Lora Node151, su entorno de desarrollo y lenguaje de programación.

En el capítulo dos se definen el sensor de movimiento *PIR* y el sensor *GPS* compatibles con el Lora Node151, la comunicación entre los periféricos y la interfaz de usuario.

En el tercer capítulo se presenta el diseño completo del prototipo. Las mediciones efectuadas sobre el prototipo se presentan en tablas, así mismo se discuten los resultados obtenidos.

En el capítulo cuatro constan las conclusiones y recomendaciones que se obtuvieron como consecuencia del presente trabajo.

PALABRAS CLAVE: Lora, Node151, STM32, ST LINK, PIR, GPS, GSM.

ABSTRACT

The present work carries out a design of a prototype node that operates with gps and detects movement based on the Lora Node 151 node.

The first chapter compiles information about LoRaWAN technology, for long distance networks and low power consumption. A study of the Lora Node151 node, its development environment and programming language is developed.

Chapter two defines the PIR motion sensor and GPS sensor compatible with the Lora Node151, the communication between the peripherals and the user interface.

In the third chapter the complete design of the prototype is presented. The measurements made on the prototype are presented in tables, and the results obtained are also discussed.

Chapter four contains the conclusions and recommendations that were obtained as a result of this work.

KEYWORDS: Lora, Node151, STM32, ST LINK, PIR, GPS, GSM.

1. INTRODUCCIÓN

La tecnología Lora/LoRaWAN presenta nuevos desafíos, comunicarnos a largas distancias con bajo consumo de potencia en sus nodos o dispositivos con una alta autonomía y sobre todo presenta bajo costo en comparación con otras tecnologías existentes como 802.15.4, ZigBee, proporcionando nuevas opciones para el desarrollo de aplicaciones en redes con topología lineal [1].

El impresionante desarrollo del internet de las cosas (IoT) [2], desafía campos de aplicaciones muy diversos, no solo en ámbitos de interconexiones personales sino también en sistemas y aplicaciones industriales, comerciales, etc. Al mismo tiempo, el consumo de energía eléctrica por parte de los dispositivos inmersos en el internet de las cosas es un parámetro que se debe tomar en cuenta, debido a que, dichos dispositivos necesitan energía permanente [3] para suplir la necesidad de disponibilidad por parte de los usuarios.

Una de las tecnologías utilizadas en IoT es LPWAN [4] [5] (Redes de Área Amplia de Baja Potencia), derivada de esta tecnología es LoRa/LoRAWAN que básicamente nace para satisfacer requerimientos del internet de las cosas y mantener un bajo consumo de energía eléctrica [6]; esta tecnología trabaja significativamente con equipos de bajo consumo (hasta 10 años con una batería) de esta manera los sensores o dispositivos son relativamente independientes de un suministro continuo de corriente eléctrica [3].

Es así que el bajo consumo de potencia de LoRaWAN, evita el uso permanente de energía eléctrica, los dispositivos en LoRa ocupan baterías que podrían durar hasta 10 años de manera continua, esto es de mucha ayuda para lugares donde se carece de energía eléctrica o las condiciones físicas dificultan el acceso de este [2].

Además, LoRa trabaja en distintas áreas y frecuencias en las cuales generalmente no existe una cobertura fija, Debido a que LoRa usa comunicación de radiofrecuencia no utiliza licencia, ya que ocupa el espectro de banda de radio ISM [7].

La versatilidad que ofrece el Lora Node151 para usar bandas de frecuencia, así como cambiar el consumo de potencia eléctrica durante su funcionamiento [3], entre otras funciones avanzadas, como cambiar de rol, pasando de ser un LoRa nodo a un LoRa Gateway [9], son grandes ventajas a la hora de elegir el dispositivo para desarrollar aplicaciones de IoT.

El estudio del Lora Node151, que se propone en el presente trabajo, permitirá obtener una estructura ordenada sobre el entorno de desarrollo, lenguaje de programación y los recursos que ofrece esta plataforma, que pueden ser utilizados como consulta para futuros

desarrollos de aplicaciones en IoT que requieran de las funciones avanzadas que ofrece este dispositivo.

1.1 Objetivo general

El objetivo general de este Proyecto Técnico es:

- Diseñar un nodo prototipo que opera con GPS y detecta movimiento basado en el nodo Lora Node151.

1.2 Objetivos específicos

- Estudiar la tecnología LoRaWAN.
- Estudiar el Lora Node151, su entorno de desarrollo y lenguaje de programación.
- Implementar un prototipo con el Lora Node151 en conjunto con un sensor de movimiento PIR y un sensor GPS.
- Realizar pruebas del prototipo.

1.3 ALCANCE

El presente trabajo se concentra en el estudio del entorno de desarrollo, lenguaje de programación, características y recursos que ofrece el nodo Lora Node 151.

El estudio conformará una estructura ordenada de la información recopilada sobre el Lora Node151. Se determinarán los recursos de programación que ayudarán a desarrollar el manejo de este dispositivo.

Se presentará un prototipo con Lora Node151 en el que se controlará un sensor de movimiento PIR y un sensor GPS para alcanzar el objetivo propuesto.

La propuesta se verificará realizando pruebas al prototipo implementado en el cual se verificará la validez de este, utilizando el software de desarrollo STM32 Cube IDE, STM32 Cube Programmer, un sensor de movimiento PIR, y un sensor GPS.

1.4 MARCO TEÓRICO

En esta sección se realiza un estudio de la tecnología LoRaWAN, para redes de larga distancia y bajo consumo de potencia. Así como, un estudio del nodo Lora Node151, su entorno de desarrollo y lenguaje de programación.

1.4.1 Revisión del protocolo LoRaWAN

El Internet de las Cosas (IoT) contempla un gran crecimiento de redes de sensores inalámbricos (WSNs) interconectados. Estas redes tienen limitaciones tales como la latencia, el ancho de banda, la cobertura, no siendo menos importante la limitación de energía ya que los nodos funcionan con baterías. Es así, que existen varias investigaciones en cuanto al tema de ahorro de energía cuyo objetivo es realizar la conexión de red optimizando el consumo de energía, y por lo tanto, aumenta la vida útil [4]. Para este propósito surge LoRa que suple las necesidades de cobertura y bajo consumo de energía.

LoRa es un tipo de modulación de espectro esparcido conocida como Long Range Modulation, por sus siglas en inglés, basado en la modulación de Chirp Spread Spectrum (CSS), proporciona largo alcance, bajo consumo de energía, seguridad y baja velocidad en la transmisión de datos,

LoRa está orientado a dos capas: i. una capa física que utiliza modulación de radio denomidada CSS (Chirp Spread Spectrum) y ii. Una capa MAC que utiliza un protocolo denominado LoRaWAN encargado de proporcionar acceso a la arquitectura LoRa [4]. La Figura 1.1 presenta las capas del protocolo LoRaWAN.

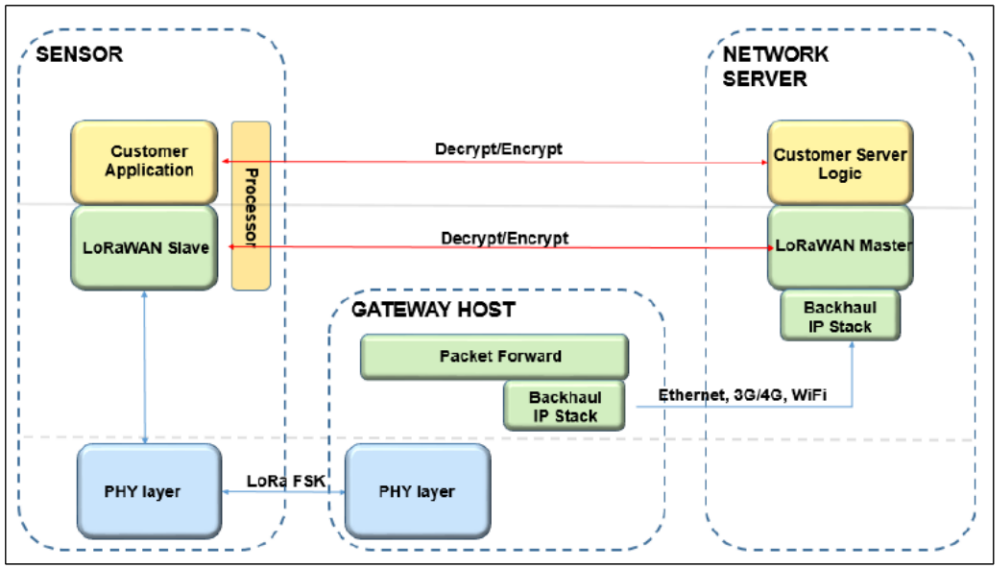


Figura 1.1. Representación de las capas de la red LoRaWAN [4].

El protocolo LoRaWAN, Long Range Wide Area Network por sus siglas en inglés, está orientado a la comunicación y arquitectura del sistema de red. Algunas características de este protocolo son [5]:

- Influye en la duración de la batería del nodo.
- Influye en la capacidad de red
- Tiene efecto en la calidad de servicio
- Tiene efecto en la seguridad

Como se observa en la Figura 1.1, LoRaWAN consta de nodos finales (sensores), gateways (concentradores) y servidores de red. Los datos transmitidos por un nodo final son retransmitidos por gateways hacia los servidores de red. La interfaz nodo final – gateway utiliza un protocolo propio de LoRa para la modulación de datos, por ejemplo, LoRa FSK; así mismo la interfaz gateway – servidor de red puede ser ethernet, red celular, wifi. En los paquetes enviados desde el nodo final hacia el servidor de red, viaja la carga útil de LoRaWAN que tiene como destino final un servidor de aplicaciones [4].

Para que los paquetes puedan ser reenviados tanto para enlaces ascendentes como descendentes, desde los nodos finales hacia los servidores y viceversa, en los gateways existe un software instalado denominado reenviador de paquetes. Este software utiliza multicanal para el reenvío de los paquetes [5]. Las funciones principales que ejecuta este software son:

- Reenviar los paquetes LoRa recibidos por el gateway al servidor de red a través de IP/UDP.
- Emitir paquetes LoRa que son enviados por el servidor de red.

Para realizar la modulación y demodulación de datos en la capa física, dependiendo de las especificaciones que requiere la aplicación, LoRa utiliza chips desarrollados por la Corporación Semtech [6], los cuales proveen de mecanismos para el procesamiento de datos tanto para transmisión como recepción. Ejemplos de estos chips son SX1276, SX1277, SX1279.

Sin embargo, este chip no posee periféricos como memoria, temporizadores, convertidores CAD-DAC, entre otros, por tanto, otro chip es el encargado de realizar el control de instrucciones que el usuario necesita programar. Este chip lo constituye un microcontrolador que es el encargado de proporcionar los recursos tales como memoria, timers, convertidores DAC – CAD, opciones de modos de operación en función de la

energía, etc. Los microcontroladores que se utilizan en LoRa, son desarrollados por la Compañía STMicroelectronics [7] y son principalmente los MCU STM32L151xCC.

1.4.2 CHIP SX1276

El chip SX1276 es un circuito integrado de alto rendimiento ideal para aplicaciones de RF en la banda ISM, es un transductor half-duplex de baja frecuencia intermedia (IF Intermediate Frequency), consta de bloques receptores y transmisores cada uno con su respectivo modem y utiliza modulación FSK, OOK y modulación de espectro esparcido LoRa [8]. La modulación LoRa usa una técnica propietaria de espectro esparcido que, a diferencia de la modulación FSK/OOK, permite el incremento del presupuesto del enlace y una mayor inmunidad a la interferencia. La Figura 1.2 muestra el diagrama de pines para el encapsulado tipo QFN.

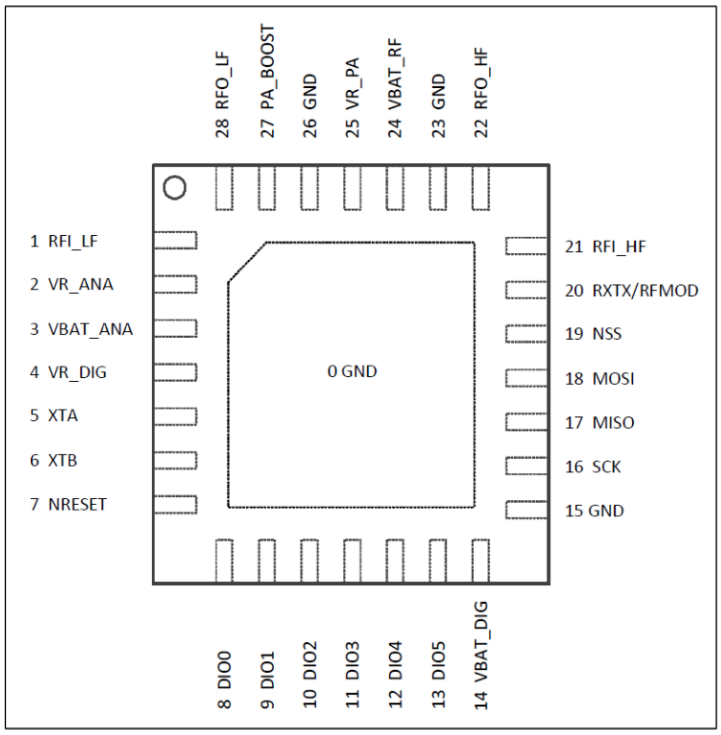


Figura 1.2. Encapsulado QFN del chip SX1276 [8].

El circuito integrado SX1276 tiene 28 pines entre los que se utilizan para la comunicación SPI los pines 15, 16, 17 y 18. La Tabla 1.1 muestra las especificaciones del dispositivo.

Este chip ofrece opciones de ancho de banda que van desde 7,8 kHz hasta 500 kHz con factores de dispersión SF que van de 6 a 12 y cubre todas las bandas ISM [8] [9]. La Figura 1.3 muestra la estructura de este chip.

Tabla 1.1. Especificaciones del chip SX1276 [8].

Voltaje de alimentación [V]	Temperatura de trabajo [°C]	Frecuencia de operación [MHz]	Bandas de frecuencia [MHz]	Potencia de salida [dBm]
3.3	25	32	169/433/868/915	13
Rango de Frecuencia [MHz]	**Factor de Dispersión (SF)**	**Ancho de Banda (BW) [KHz]**	**Tasa de datos efectiva [Kbps]**	**Sensibilidad [dBm]**
137 – 1020	6 - 12	7,8 – 500	0,018 – 37,5	-111 – -148

La Figura 1.3 muestra un buffer con sistema FIFO tanto para la modulación/demodulación FSK/OOK como para la modulación/demodulación LoRa, en este buffer se almacenan temporalmente los datos extraídos previamente de la carga útil.

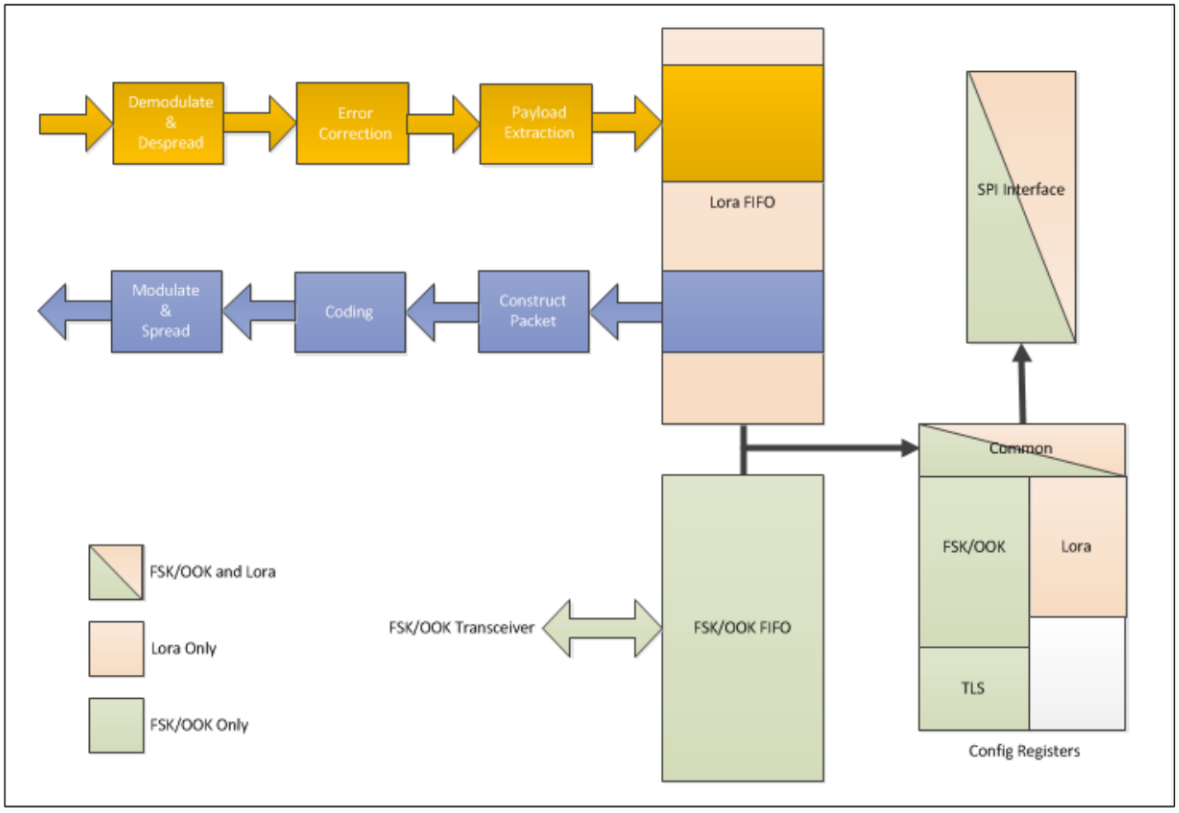


Figura 1.3. Diagrama de bloques del chip SX1276 [8].

Todos los parámetros principales de la interfaz de RF son totalmente configurables a través de una interfaz SPI que brinda acceso a los registros de configuración del SX1276.

El chip SX1276 es capaz de proporcionar corrección de errores en los datos y puede ser programado para elegir el tipo de modulación/demodulación mediante una interfaz SPI. Para marcar el reloj de conversión, el circuito integrado SX 1276 utiliza 2 osciladores, un oscilador RC y un oscilador de cristal de 32MHz [8].

Modulación Demodulación LoRa

El módem LoRa utiliza modulación de espectro esparcido y técnicas de corrección de errores de reenvío para aumentar el alcance y robustez de los enlaces de comunicación por radio en comparación con la modulación tradicional basada en FSK u OOK [8].

Para que sea posible optimizar la modulación LoRa para una aplicación determinada, el usuario puede configurar tres parámetros, los cuales son factor de dispersión (SF Spreadin Factor), tasa de codificación (CR Coding Rate) y ancho de banda (BW Bandwidth) de modulación.

Factor de dispersión (SF)

La modulación LoRa representa cada bit de información por múltiples bits de información, denominados chips. La tasa a la que se envía los chips de información se conoce como velocidad de símbolo (Rs), la relación entre la velocidad de símbolo y la velocidad de chips se conoce como factor de dispersión (SF) y representa el número de símbolos por bit de información [8].

Los valores que puede tomar el factor de dispersión se muestran en la siguiente Tabla 1.2.

Tabla 1.2. Rango de factores de dispersión.

Factor de dispersión	Tasa Chips/símbolo	SNR [dBm]
6	64	-5
7	128	-7,5
8	256	-10
9	512	-12,5
10	1024	-15
11	2048	-17,5
12	4096	-20

El factor de dispersión debe ser el mismo tanto en el lado de transmisión como en el de recepción del enlace, ya que los diferentes factores de dispersión son ortogonales entre si.

Tasa de codificación

La robustez del enlace puede mejorarse mediante el empleo de una codificación de error cíclica para errores de reenvío, detección y corrección. Tal codificación de errores incurre en una sobrecarga de transmisión: la sobrecarga de datos adicional resultante por la transmisión se muestra en la siguiente Tabla 1.3.

Tabla 1.3. Sobrecarga de codificación cíclica

Tasa de codificación	Tasa de codificación cíclica	Razón de sobrecarga
1	4/5	1.25
2	4/6	1.5
3	4/7	1.75
4	4/8	2

Para mejorar la confiabilidad del enlace ante la interferencia, se utiliza la corrección de errores, la tasa de codificación puede cambiarse de acuerdo a las condiciones del canal [8].

Ancho de banda de la señal

El aumento del ancho de banda de la señal incrementa la tasa de datos efectiva, reduciendo así el tiempo de transmisión, pero altera la reducción de la mejora de la sensibilidad. El módem LoRa utiliza un ancho de banda de doble banda lateral (o ancho de banda total del canal). La siguiente Tabla 1.4 muestra los anchos de banda para el modem LoRa [10].

Tabla 1.4. Anchos de Banda para el modem LoRa.

Ancho de banda [KHz]	Factor de dispersión	Tasa de codificación	Tasa de bits nominal [bps]
7,8	12	4/5	18
15,6	12	4/5	37
31,2	12	4/5	73
62,5	12	4/5	146
250	12	4/5	586
500	12	4/5	1172

Estructura del paquete LoRa

La Figura 1.4 muestra el formato de paquete para la modulación/demodulación LoRa.

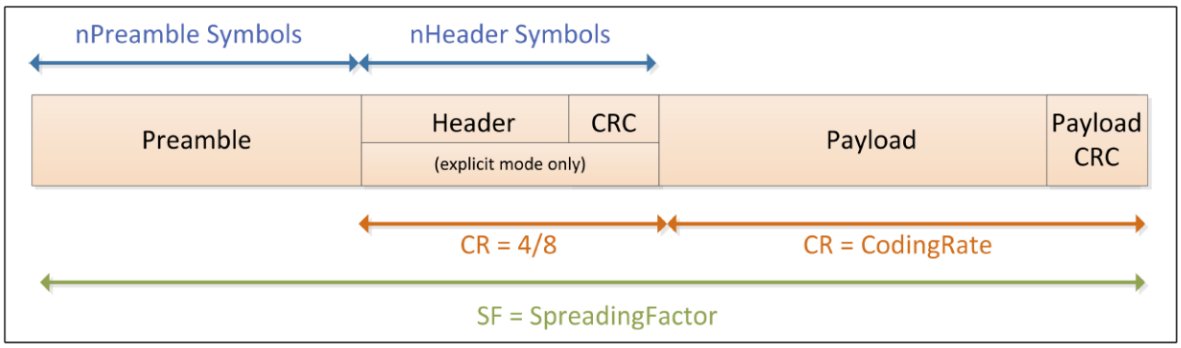


Figura 1.4. Formato de paquete para modem LoRa [8].

El tipo de modulación/demodulación LoRa proporciona dos formatos de paquetes los cuales son explícito e implícito. El paquete explícito contiene un encabezado corto con información sobre la cantidad de bytes, la tasa de codificación y el uso de CRC en el paquete, mientras que para el formato implícito se omite esta sección del paquete.

Los tres elementos principales del paquete son: preámbulo, cabecera y carga útil.

Preámbulo

El preámbulo sirve para la sincronización del receptor ante el flujo de datos entrante. El paquete está configurado por defecto con 12 secuencias de símbolos, este es un parámetro que se puede programar, por lo que la longitud del preámbulo puede variar.

El receptor detecta periódicamente el preámbulo, por consiguiente, la longitud del preámbulo debe programarse de la misma forma que en el transmisor [10].

Encabezado

LoRa tiene dos tipos de ebcabezados:

El encabezado explícito es el modo configurado por defecto. El encabezado proporciona información sobre la carga útil como:

- La longitud de los bits de información en bytes.
- La tasa de corrección de errores.
- Un CRC de 16 bits opcional para los bits de información.

El encabezado implícito omite el encabezado de la trama, ya que en algunas aplicaciones la carga útil, la tasa de codificación y el CRC son valores fijos conocidos de antemano [7].

Carga útil

La carga útil, o bits de información, tiene longitud variable y contiene los datos reales codificados con la tasa de error, como se especifica en el encabezado en modo explícito. En la programación de modo implícito, se puede agregar un CRC opcional.

Tiempo de transmisión del paquete LoRa

Para calcular el tiempo de transmisión del paquete LoRa se toma en cuenta el parámetro factor de dispersión SF. El ancho de banda BW y el factor de dispersión pueden ser elegidos mediante programación por el usuario.

Con los parámetros previamente elegidos por el usuario se puede definir la tasa de símbolo R_s de LoRa como [8]:

$$R_s = \frac{BW}{2^{FS}} \text{ [bps]} \tag{1.1}$$

Análogamente se obtiene el tiempo de símbolo T_S [8]:

$$T_s = \frac{1}{R_S} \text{ [s]} \tag{1.2}$$

La duración del paquete LoRa es la suma de los tiempos de duración del preámbulo y de la carga útil. El tiempo del preámbulo se calcula de la siguiente manera:

$$T_{preámbulo} = \left(n_{preámbulo} + 4{,}25\right)T_S \text{ [s]} \tag{1.3}$$

Donde $n_{preámbulo}$ es la longitud del preámbulo previamente programado por el usuario.

La duración de la carga útil se obtiene multiplicando el número de bits que contiene la carga útil $n_{carga\,útil}$ por el tiempo de símbolo T_S:

$$T_{carga\,útil} = n_{carga\,útil} \times T_S \text{ [s]} \tag{1.4}$$

Mediante las ecuaciones (1.3) y (1.4) se obtiene el tiempo del paquete LoRa:

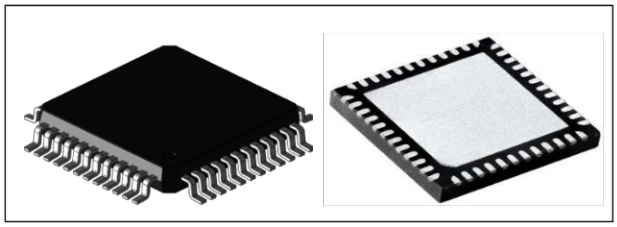
$$T_{paquete} = T_{preámbulo} + T_{carga\,útil} \text{ [s]} \tag{1.5}$$

La Corporación SEMTECH [6] proporciona módulos que contienen este chip para el procesamiento de la señal en la capa física de la red LoRaWAN. La figura 1.5 muestra un Módulo SX1276 DVK1.

Figura 1.5. Kit de desarrollo SX1276 DVK1.

1.4.3 Microcontrolador STM32L151CCU6

El MCU STM32L151CCU6 es un dispositivo de potencia ultra baja desarrollado por la Organización STMicroelectronics [7] basado en ARM® Cortex®-M3 con arquitectura RISC de 32bits, con eficiencia de código de alto rendimiento debido a la gama que posee el núcleo ARM, ofrece un rendimiento computacional excepcional y una respuesta avanzada del sistema a las interrupciones. La siguiente Figura 1.6 muestra los tipos de encapsulado.



(a (b

Figura 1.6. Encapsulados del MCU STM32L151CCU6. (a) LQFP48. (b) UFQFPN48 [7].

La siguiente Tabla 1.5 muestra las características del dispositivo [10].

Tabla 1.5. Características del MCU STM32L151CCU6.

Memoria FLASH [Kbytes]		256
Memoria EEPROM [Kbytes]		8
Memoria RAM [Kbytes]		32
Temporizadores	32 bits	1
	Propósito general	6
	Básicos	2
Interfases	SPI	8
	I^2S	2
	I^2C	2
	USART	3
	USB	1
Número de pines		37
Amplificadores operacionales		2
Convertidores ADC	Sincronizados de 12 bits	14
	No sincronizados de 12 bits	2
Comparadores		2
Frecuencia [MHz]		32
Voltaje [V]		1,8 – 3,6
Temperatura [°C]		-40 – 105
Encapsulado		LQFP48, UFQFPN48

EL microcontrolador incorpora núcleos de alta eficiencia energética con arquitectura Harvard y ejecución PIPELINE, por tanto, este dispositivo de potencia ultra baja es la mejor opción para desarrollar aplicaciones orientadas, por ejemplo, para medidores de gas/agua, teclado/mouse, aplicaciones para la salud, aplicaciones portátiles, etc. Además, cuenta con diferentes modos para baja energía que se muestran en la siguiente Tabla 1.6.

Tabla 1.6. Modos de baja energía [7].

Modo		**Recursos**
SLEEP		CPU detenida, Todos los periféricos continúan trabajando.
LOW POWER RUN		Minimiza la corriente de operación, limita la frecuencia y la actividad de los periféricos.
LOW POWER SLEEP		Desactiva los periféricos, excepto la interrupción wakeup.
STOP	con RTC	Consumo de energía más bajo, conserva la RAM, el contenido de los registros y el reloj en tiempo real.
	sin RTC	Consumo de energía más bajo, conserva la RAM y el contenido de los registros.
STANDBY	con RTC	Menor consumo de energía y reloj en tiempo real.
	Sin RTC	Menor consumo de energía.

Memoria

Este MCU tiene 32 Kbytes de RAM (Random Access Memory) integrada (lectura/escritura) a la velocidad del reloj del CPU. La memoria RAM se divide en tres arreglos: 256 Kbytes de memoria Flash, 8 Kbytes de memoria EEPROM, Bytes de opciones. Los bytes de opciones se utilizan para proteger la memoria contra escritura o lectura. La memoria RAM incorpora la función de código de corrección de errores (ECC Error Correction Code). Los códigos de corrección de errores (ECC) permiten, además de detectar el error, corregirle sin necesidad de repetir la transmisión [7].

Procesador ARM® CORTEX®-M3 con MPU

Este procesador ARM® Cortex®-M3 ha sido desarrollado para proporcionar una plataforma de bajo costo, con un número reducido de pines y un bajo consumo de energía, al mismo tiempo que ofrece rendimiento computacional sobresaliente y una respuesta avanzada del sistema a las interrupciones.

El procesador ARM® Cortex®-M3 RISC de 32 bits presenta una eficiencia de código tal que, entrega un alto rendimiento desde el núcleo ARM. La unidad de protección de memoria (MPU Memory Protection Unit) es una unidad programable que permite al

software, mejorar la confiabilidad del sistema. Los dispositivos STM32L151xC son compatible con todas las herramientas y software ARM [10].

GPIO (entradas/salidas de propósito general)

Cada uno de los pines GPIO se puede configurar por software como salida (push-pull o open-drain), como entrada (con o sin pull-up o pull-down) o como función alternativa periférica. La mayoría de los pines GPIO se comparten con funciones alternativas digitales o analógicas.Todos los GPIO tienen capacidad de alta corriente. La configuración de funciones alternativas de E/S se puede bloquear si es necesario siguiendo una secuencia para evitar escrituras espurias en los registros de E/S. El controlador de E/S es conectado con una velocidad de conmutación de hasta 16 MHz [7].

Potencia ultrabaja y voltaje de referencia

Los dispositivos STM32L151xC incorporan dos comparadores que comparten la misma polarización de corriente y voltaje de referencia. El voltaje de referencia puede ser interno o externo (procedente de una E/S). El voltaje de referencia interno está disponible externamente a través de una salida de baja potencia / baja corriente búfer (capacidad de corriente de conducción de 1 µA típico) [10].

El microcontrolador STM32L151CCU posee las siguientes interfaces:

Interfaz Autobús I²C

Tiene dos interfaces de bus I²C que pueden operar en modos multimaestro y esclavo. Admiten direccionamiento esclavo dual (solo 7 bits) y direccionamiento de 7 y 10 bits en modo maestro. Se incorpora verificación de CRC por hardware [10].

Interfaz Transmisor receptor síncrono/asíncrono universal (USART)

Tiene tres interfaces USART que pueden comunicarse a velocidades de hasta 4 Mbit/s, tiene capacidad LIN Master/Slave. Los tres USART proporcionan configuración por hardware de las señales CTS y RTS y cumplen con la norma ISO 7816. Todas las interfaces USART pueden ser atendidas por el controlador DMA, el cual es un dispositivo que controla el flujo de datos entre la memoria y un periférico sin que intervenga la CPU [10].

Interfaz periférica en serie (SPI)

Posee tres SPI que pueden comunicarse a una velocidad de hasta 16 Mbits/s en los modos esclavo y maestro en modos de comunicación full-duplex y half-duplex. El pre-escalador

de 3 bits ofrece 8 modos maestros frecuencias y la trama es configurable a 8 bits o 16 bits. El hardware CRC de verificación es compatible con los modos básicos de tarjeta SD/MMC. Los SPI pueden ser atendidos por el controlador DMA [7].

Interfaz I2S

Están disponibles dos interfaces I2S estándar (multiplexadas con SPI2 y SPI3), pueden operar en modo maestro o esclavo, y se puede configurar para operar con un 16-/32-bit resolución como canales de entrada o salida. Dispone de frecuencias de muestreo de audio desde 8 kHz hasta 192 kHz. Cuando una o ambas interfaces I2S están configuradas en el modo maestro, el reloj maestro se puede enviar al DAC/CODEC externo a 256 veces la frecuencia de muestreo. Los I2S pueden ser atendidos por el controlador DMA [7].

Bus serie universal (USB)

Los dispositivos STM32L151xC y STM32L152xC incorporan un dispositivo periférico USB compatible con USB full-speed 12 Mbit/s. La interfaz USB implementa una velocidad de 12 Mbit/s. Tiene una configuración por software y admite suspender/reanudar. Tiene un reloj dedicado de 48 MHz. Los SPI pueden ser atendidos por el controlador DMA [7].

Puerto de depuración JTAG de cable serie (SWJ-DP)

La interfaz ARM SWJ-DP es una depuración combinada de JTAG que permite conectar una depuración de cable serie o una sonda JTAG. Los pines JTAG JTMS y JTCK se comparten con SWDAT y SWCLK, respectivamente, y se utiliza una secuencia específica en el pin JTMS para cambiar entre JTAG-DP y SW-DP. El puerto JTAG se puede desactivar permanentemente con un fusible JTAG.

El dispositivo LoRa Node 151 se compone de estos dos dispositivos presentados como son, para la parte física, el chip SX1276 y para desarrollar la parte de control, el MCU STM32L151CCU6, por lo tanto, LoRa Node 151 es un dispositivo de alta gama que proporciona una eficiencia elevada en cuanto a consumo de energía y procesamiento de instrucciones [10].

1.4.4 LoRa Node 151

Es un módulo desarrollado por la compañía Chengdu Heltec Automation Technology Co., Ltd. [9] basado en un diseño de consumo de energía ultra bajo, está conformado por un MCU STM32L151CCU6 y el chip SX1276. LoRa Node 151 utiliza una batería AA ½ de 3,3V

de litio de sulfato de cloro. Se puede ingresar al modo de bajo consumo de energía mediante el estándar LoRaWAN, con un consumo de corriente menor a 1,8[mA] [3].

En la siguiente Figura 1.7 se muestra el módulo LoRa Node 151. El módulo consta de un terminal SMD para la conexión de una antena, la cual sirve para la comunicación RF en las bandas ISM, comandado por el chip SX1276.

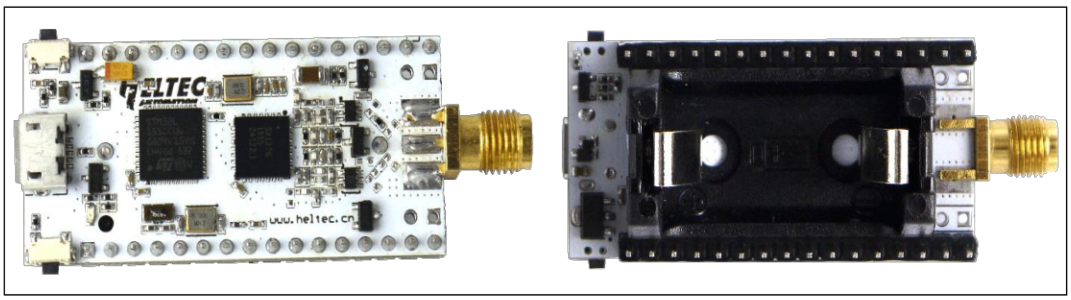


Figura 1.7. LoRa Node 151.

Lora Node 151 tiene una antena que se conecta al plug SMA soldado a la placa, se puede montar y desmontar dependiendo de la interfaz a usar para la transición de datos. Así mismo la placa dispone de un zócalo para conectar una batería de litio que suministrará la energía para el funcionamiento de la placa, esta implementación tiene la característica de detectar si se conecta también la interfaz microUSB, en tal caso se desconecta automáticamente el suministro de energía por parte de la pila.

Además, la placa cuenta con dos pulsadores que sirven para reestablecer el nodo y también permiten ingresar en modo DFU. Este modo se utiliza cuando se conecta por primera vez el LoRa Node 151 a un ordenador, para instalar el software a la placa mediante la interfaz micro USB. En la Figura 1.8 se muestra un diagrama de bloques con los principales componentes de LoRa Node 151.

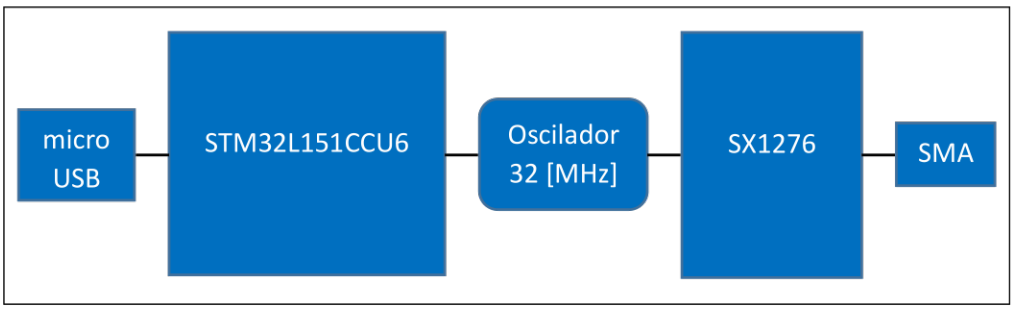


Figura 1.8. Diagrama de bloques simplificado del LoRa Node 151.

La Figura 1.8 muestra que tanto el bloque gobernado por el MCU STM 32L151CCU6 y el bloque que constituye el chip SX1276 trabajan con osciladores de 32 [MHz]. Sin embargo, mediante programación se puede disminuir el ciclo de trabajo.

En la Tabla 1.7 se presenta las principales características de LoRa Node 151. Cabe destacar que su reducido tamaño permite a la placa que se pueda ubicar dentro de un chasis fácilmente. Debido a que el MCU STM32L151CCU6 que contiene los periféricos tales como los convertidores ADC y en general el control de los pines, se pueden conectar sensores cuyas señales de salida no superen los 5 [V] y una corriente de 60 [mA] [7].

Tabla 1.7. Parámetros Técnicos de LoRa Node 151.

RECURSO		**PARÁMETRO**
Microcontrolador		STM32L151CCU6
Chip LoRa		SX1276
Cobertura LoRaWAN		137 – 1020 [MHz]
Máxima salida de potencia		19 [dB] ± 1 [dB]
Recursos de Hardware		3 UART, 2 SPI, 2 I2C, 2 I2S, ADC de 12 bits, DAC de 12 bits, 29 pines salida/entrada.
Interfaz		1 microUSB, antena LoRa SMA.
Dimensiones		56,6 x 24 x 20 [mm]
Batería		½ AA Litio
Bajo consumo de energía		2,5 [µA]
Memoria	FLASH	256 [KB]
	RAM	32 [KB]

La mayoría de los sensores proveen señales TTL por lo que el LoRa Node 151 admite una compatibilidad con una gran variedad de sensores existentes en el mercado.

En la siguiente Tabla 1.8 se muestra las características eléctricas de LoRa Node 151.

Tabla 1.8. Características Eléctricas de LoRa Node 151.

CARACTERÍSTICA ELÉCTRICA	CONDICIÓN	VALOR TÍPICO
Fuente de Alimentación	USB	5 [V]
	Batería de Litio	3,6 [V]
	Pin de 3,3 [V]	3,3 [V]
	Pin de 5 [V]	5 [V]
Consumo de Corriente	Salida de 10 [dB]	50 [mA]
	Salida de 12 [dB]	60 [mA]
	Salida de 15 [dB]	110 [mA]
	Salida de 20 [dB]	130 [mA]
Salida	Pin de 3,3 [V]	500 [mA]
	Pin de 5 [V] (Alimentado por USB)	Igual a la corriente de alimentación.

Lora Node 151 se puede programar de dos diferentes maneras, una es mediante el modo DFU en el que se necesita únicamente un cable microUSB para conectar la placa a un computador, y la otra forma es mediante el uso de un módulo extra llamado ST-LINK como se muestra en la Figura 1.9.

Figura 1.9. ST-LINK V.2.

La ventaja de usar el método ST-LINK es que puede realizar una depuración en un solo paso con el software STM32CubeIDE, pero la desventaja es que se requiere la herramienta de depuración ST-LINK. En ambos casos se necesita del programa STM32CubeIDE.

Software STM32CubeIDE

STM32CubeIDE es un entorno de desarrollo integrado (IDE) basado en el sistema Eclipse®. está dirigido a usuarios que desarrollan software integrado en C/C++ para los microcontroladores STMicroelectronics STM32. Utiliza una herramienta mejorada para STM32, basada en GNU Arm Embedded. Tiene una versión integrada de STM32CubeMX y MCUFinder, que permite una fácil configuración del proyecto, así como la generación de código en lenguaje C a través de un proceso paso a paso. Además, STM32CubeIDE integra la versión de línea de comandos de STM32CubeProgrammer (STM32CubeProg) para el manejo de memoria Flash mientras se usa el servidor ST-LINK GDB. Esto permite la programación del dispositivo STM32 a través de interfaces de depuración (JTAG y SWD). La siguiente Figura 1.10 muestra la interfaz de STM32CubeIDE [11].

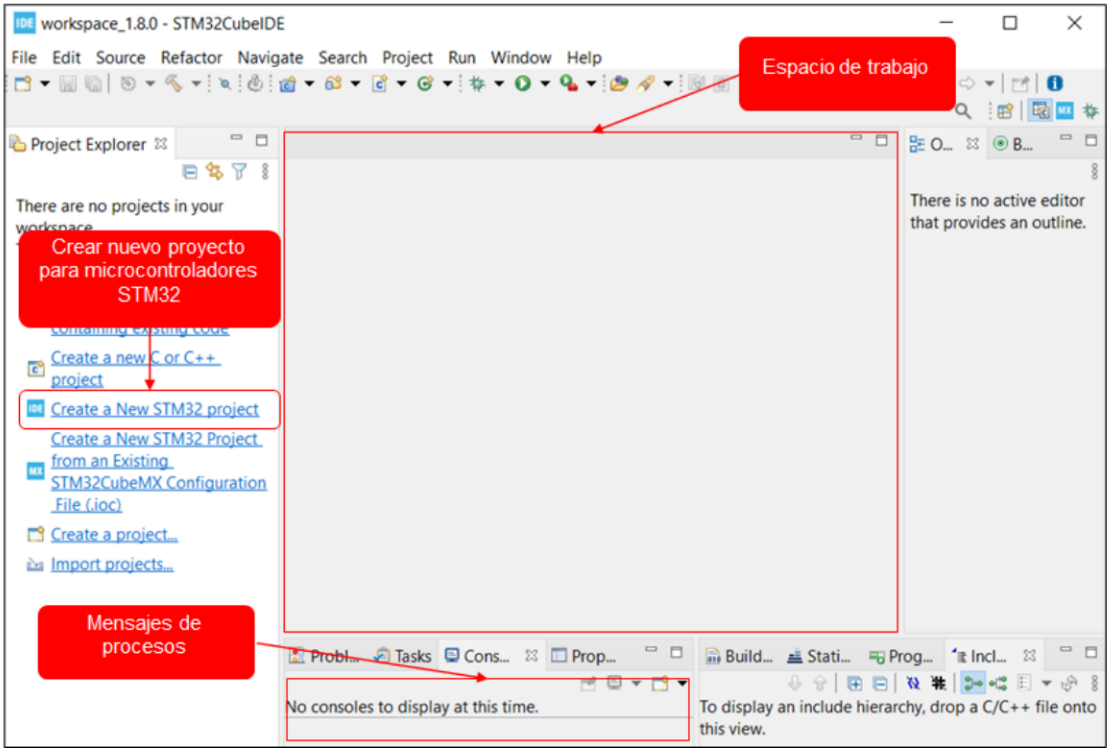


Figura 1.10. Interfaz del software STM32CubeIDE [11].

Ejemplo de programación

Como ejemplo para utilizar este software de programación se realiza un programa utilizando el LoRa Node 151. El programa consiste en controlar un diodo LED mediante un pulsador, mientras se encuentre presionado el pulsador se encenderá el diodo LED, caso contrario permanecerá apagado. La Figura 1.11 muestra el diagrama de flujo para realizar este programa.

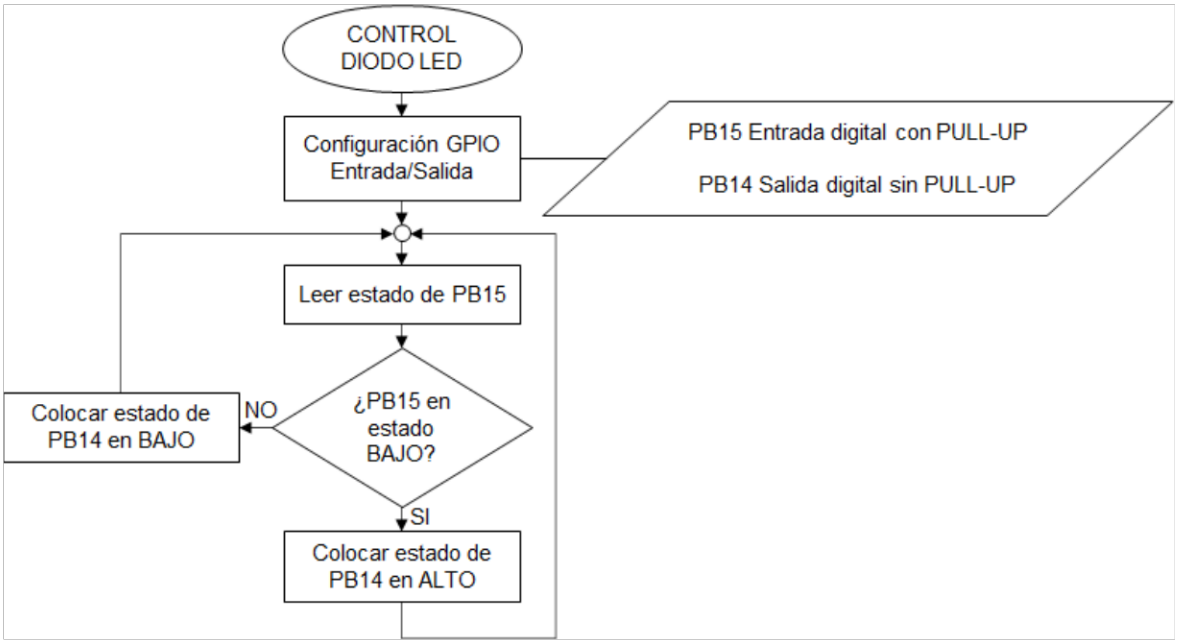
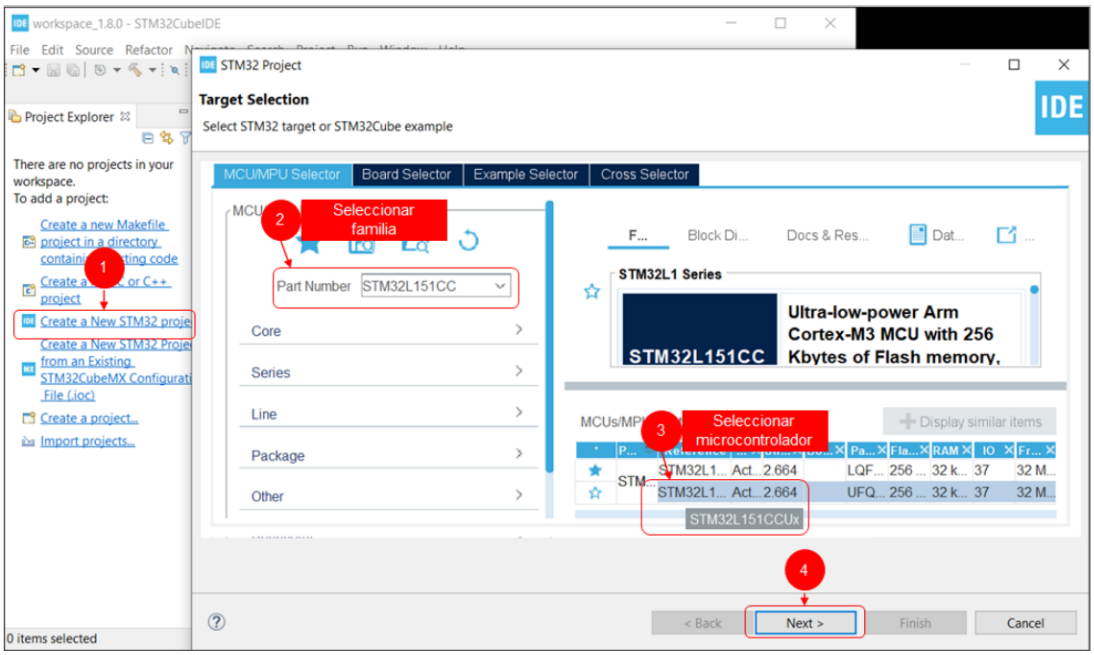
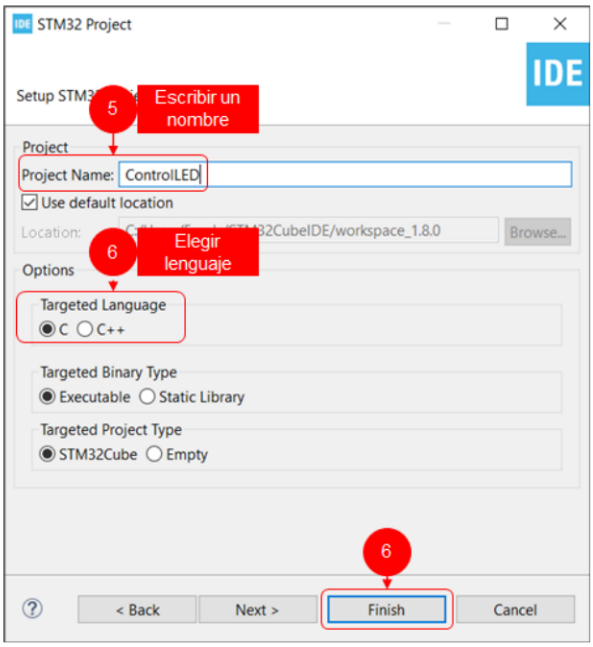


Figura 1.11. Diagrama de flujo para controlar encendido de un diodo LED.

El software STM32CubeIDE permite configurar los GPIO de los microcontroladores de la familia STM32 mediante una interfaz gráfica, en la cual, se puede seleccionar el microcontrolador específico a programar y se muestra el diagrama de pines donde se puede seleccionar el pin de interés y configurarlo de acuerdo con la necesidad del programa. LoRa Node 151 utiliza el microcontrolador STM32L151CCU6, por tanto, se selecciona este microcontrolador al momento de crear un nuevo proyecto. La siguiente Figura 1.12 muestra los pasos para seleccionar el microcontrolador.



(a)



(b)

Figura 1.12. (a) y (b) Pasos para iniciar un proyecto con STM32CubeIDE.

A continuación, se configura los pines, seleccionando en el espacio de trabajo la opción *Pinout & Configuration*. En esta sección se eligen los pines que se van a configurar como entrada o salida, debe tomarse en cuenta que un pin ya configurado, se torna de color verde, indicando que ya está ocupado dicho pin, la siguiente Figura 1.13 ilustra como configurar un pin.

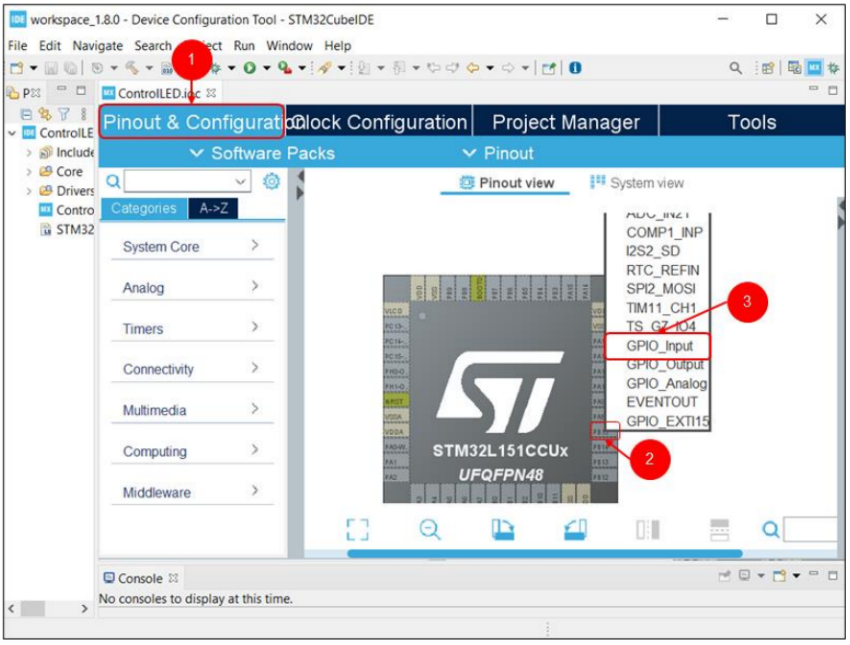


Figura 1.13. Pasos para configurar un GPIO con la interfaz gráfica.

Para configurar la resistencia interna PULL-UP, se elige la opción System Core, luego la opción GPIO. Se puede observar una lista con los pines utilizados y seleccionando un pin

se tiene la opción de activar la resistencia PULL-UP. La siguiente Figura 1.14 muestra como activar esta resistencia PULL-UP.

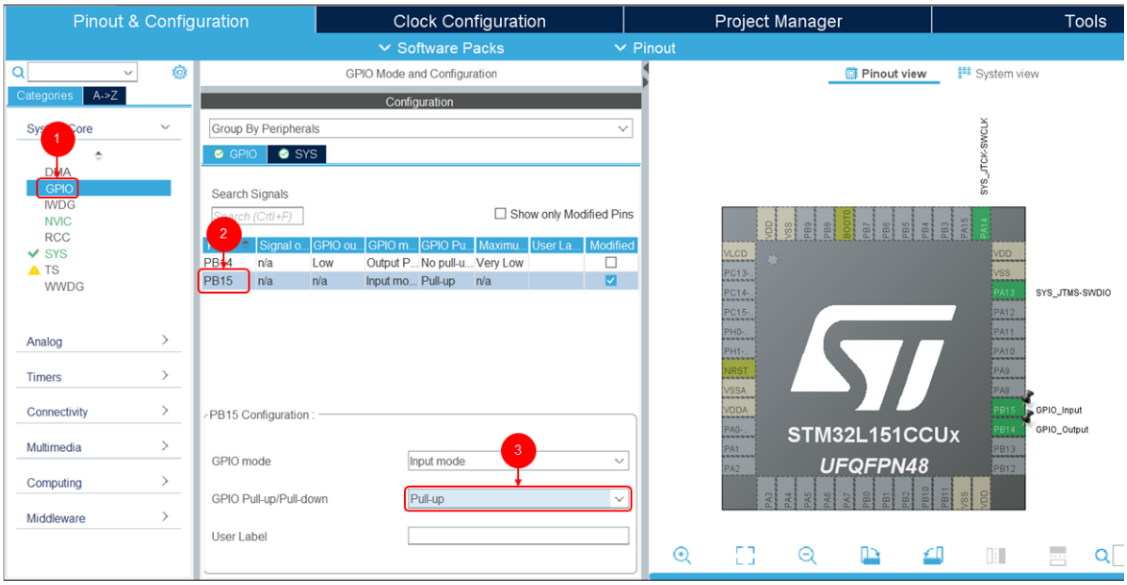


Figura 1.14. Activación de resistencia PULL-UP.

Para generar el código correspondiente a la configuración realizada mediante la interfaz gráfica, se debe seleccionar el ícono Device Configuration Tool Code Generation, se generan los archivos para el microcontrolador ST32L151CCU6 en el lenguaje de programación que se había seleccionado anteriormente, entre los programas que se generan, se encuentra un archivo main.c, en el cual, se programa el encendido del diodo LED. La siguiente Figura 1.15 muestra este proceso.

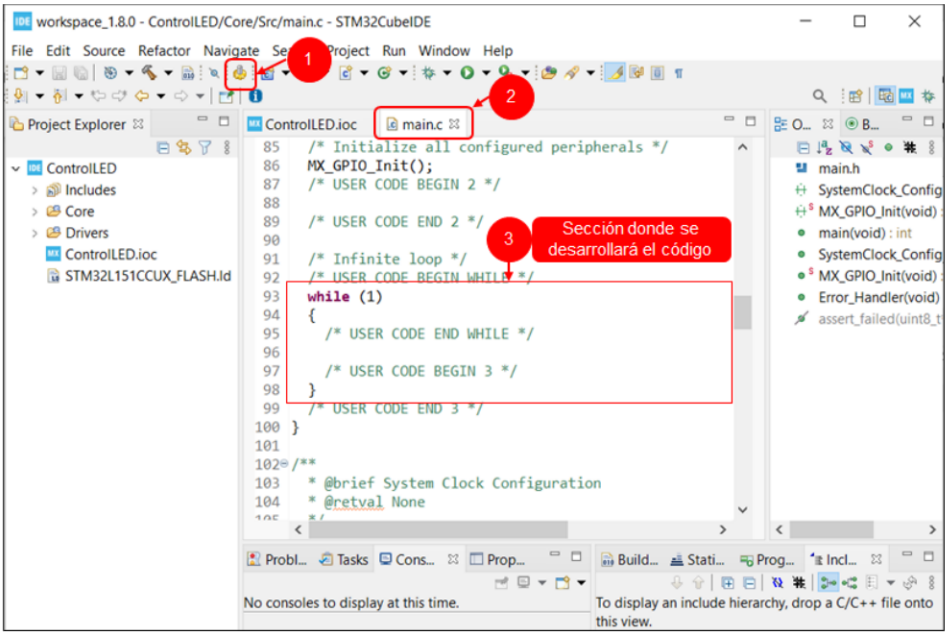


Figura 1.15. Generación de código en lenguaje C.

Para escribir el código se debe tomar en cuenta que existe un espacio predeterminado donde el usuario puede escribir el código, por ejemplo, para el bucle *while* se observa el comentario /* UDER CODE END WHILE */, indicando de esta manera que el código desarrollado por el usuario debe escribirse antes de este comentario. Para facilitar la escritura de instrucciones, el software ofrece el recurso de autocompletar la instrucción, para ello, se debe escribir la primera letra de la instrucción, seguido debe presionarse la combinación de teclas Ctrl+Space y se desplegará una lista de la que se deberá seleccionar la instrucción deseada. La siguiente Figura 1.16 muestra el código para controlar el diodo LED.

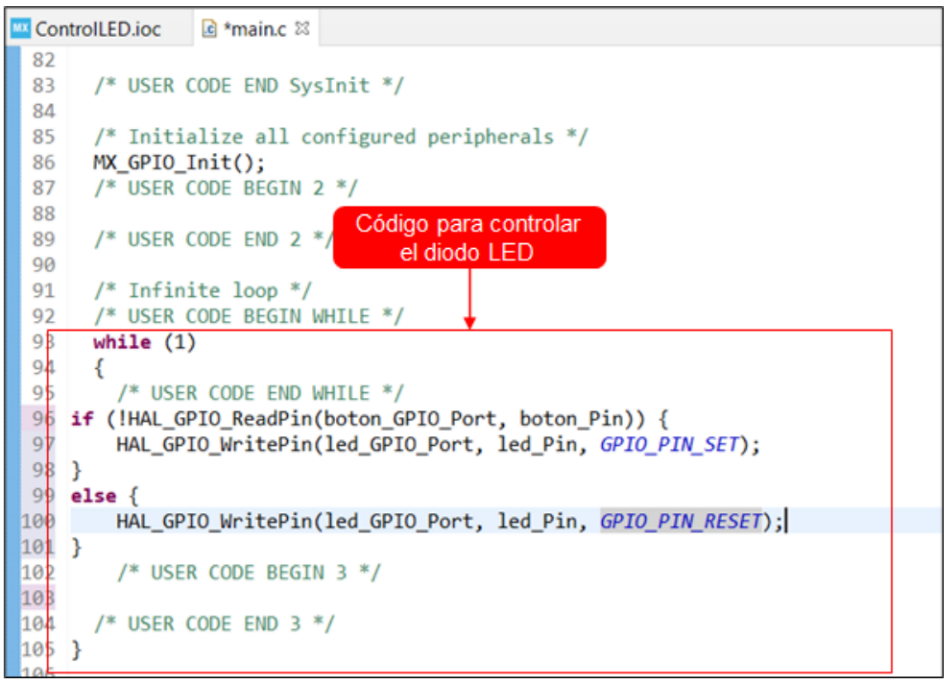


Figura 1.16. Código para controla el diodo LED.

En la siguiente Figura 1.17 se muestra el diagrama esquemático para la interconexión de elementos.

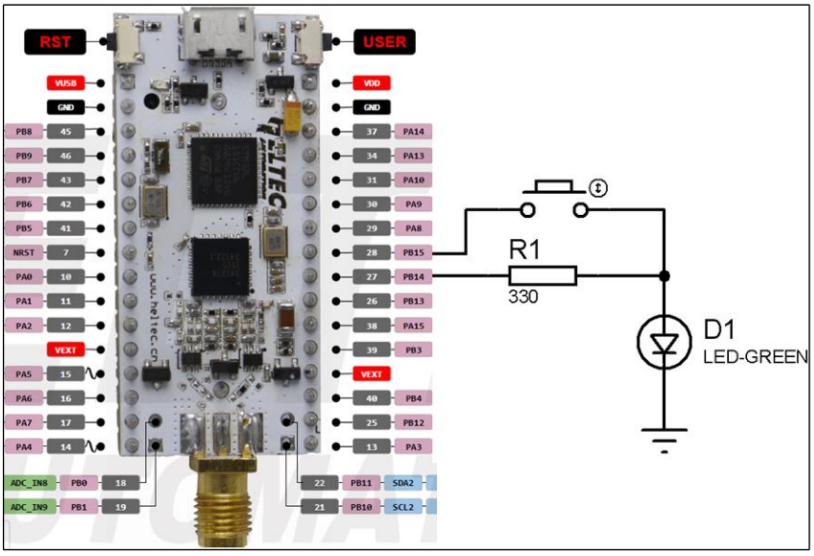
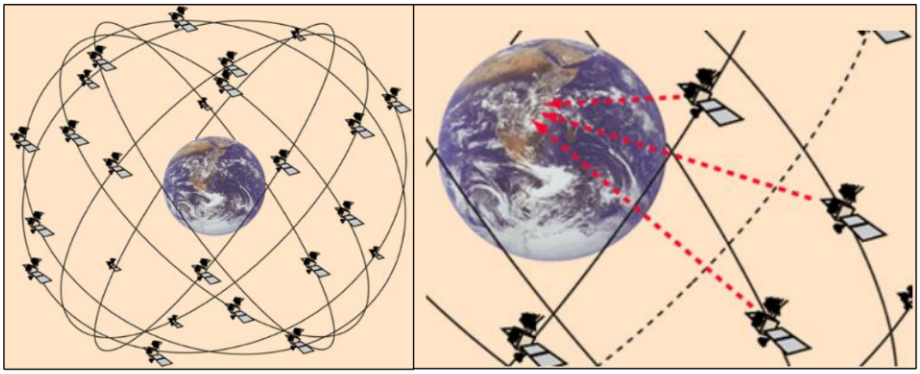


Figura 1.17. Diagrama esquemático para controlar el diodo LED.

1.4.5 Sensor de posicionamiento global GPS

El posicionamiento global (GPS Global Position System) es un sistema americano que consiste en la localización de un objeto sobre la superficie terrestre con una precisión de pocos metros [10]. Existen 4 órbitas satelitales con 4 satélites en cada órbita, de este modo, siempre se tiene como mínimo 3 satélites visibles desde cualquier parte de la superficie terrestre, como se muestra en la Figura 1.18.



(a) (b)

Figura 1.18. (a) Distribución de satélites en órbitas satelitales. (b) Triangulación con tres satélites [10].

La señal RF que un satélite transmite consta de:

- La hora de la semana
- El estado en que se encuentra el satélite
- El número de semana GPS

En la superficie terrestre, el sistema GPS se compone por 16 estaciones secundarias distribuidas en la superficie terrestre y una estación principal (MCS Master Control Station). Dichas estaciones rastrean los satélites, receptan su información, calculan la ubicación exacta de los satélites, y reenvían esta información a los satélites [11].

La medición precisa del tiempo mediante el reloj atómico permite calcular el tiempo que tarda una señal de radio en llegar al receptor GPS en la Tierra desde el satélite. Los sensores GPS de los diferentes dispositivos como los celulares, poseen osciladores de cristal de cuarzo que no son exactos. En su lugar, los dispositivos GPS utilizan el propio sistema satelital para actualizar el tiempo con mucha precisión. Por otra parte, los dispositivos que no tienen GPS, como los computadores, realizan la actualización del tiempo mediante el protocolo NTP (Network Time Protocol), el cual, sincroniza el tiempo de los dispositivos con el tiempo de las estaciones secundarias [11].

Por otro lado, la Unión Soviética también formó su Sistema Global de Navegación por Satélite (GLONASS Global Navigation Satellite System) operando desde 1995. China también desarrolló su propio sistema satelital llamado Sistema de Navegación Beidou (BDS Beidou Navigation System) en el año 2020. Es así como, frente a estos sistemas regionales de posicionamiento global desarrollados por los diferentes países u organizaciones, se les pueden caracterizar como un sistema continuo y de alta precisión, dentro de la comunidad de los Sistemas Globales de Navegación por Satélite (GNSS Global Navigation Satellite Systems) [11] [12]. Estas características del GNSS implican amplias y potenciales aplicaciones. Cada satélite GNSS transmite continuamente señales de radio en dos o más frecuencias en la banda L (1–2 GHz) con una longitud de onda de alrededor de 20 cm, las señales directas se utilizan para navegación, posicionamiento y temporización [12].

1.4.6 Sensor GPS U-blox NEO-6M

En el presente trabajo se utilizará un módulo GPS U-blox NEO-6M-0-001 V2 [14]. Los módulos NEO-6 son receptores GPS que operan por sí mismos y son de alto rendimiento. La Figura 1.19 muestra el GPS que se utilizará en el desarrollo de este trabajo.

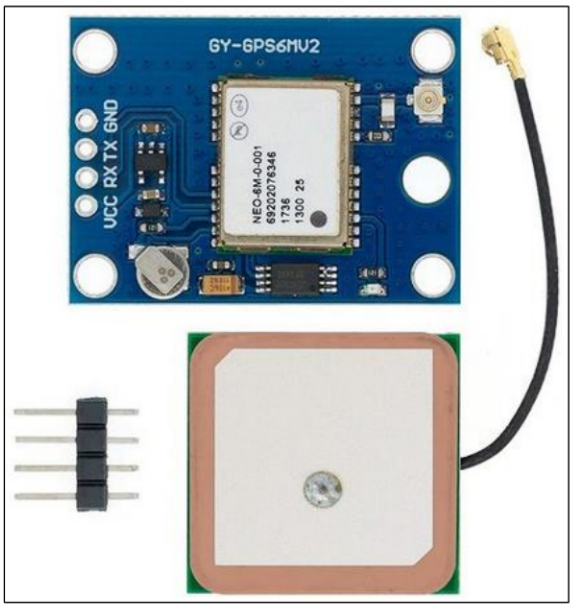


Figura 1.19. Módulo GPS U-blox NEO-6M-0-001 V2 [14].

Para el proceso de posicionamiento, el módulo u-blox 6 encuentra los satélites instantáneamente mediante la utilización de 50 canales, ya que realiza búsquedas

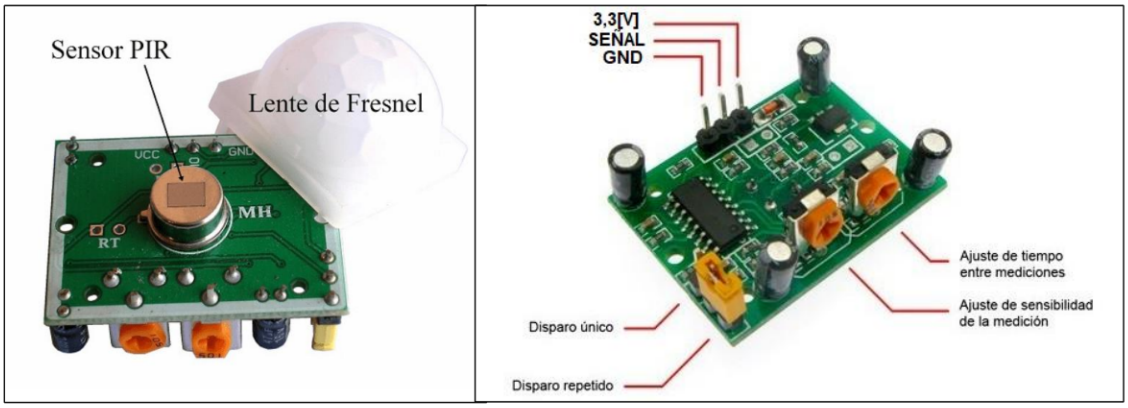
simultáneas en tiempo, espacio y frecuencia. La siguiente Tabla 1.9 muestra las características de este módulo GPS.

Tabla 1.9. Características del módulo GPS Ublox NEO-8M GY-NEO8MV2 [14].

Modelo	NEO-6M-0-001
Categoría	Precisión estándar GNSS
Número de GNSS concurrentes	3
Fuente de alimentación	1.65 [V] – 3.6 [V]
Interfaces	UART, SPI, I2C
Oscilador	RTC
Protocolo de comunicación	UBX
Dimensiones	16 x 12,2 x 2,4 [mm]

1.4.7 Sensor de movimiento PIR

El sensor de movimiento basado en Sensores Infrarrojos pasivos Piroeléctricos (PIR Pyroelectric passive InfraRed sensors) se compone de sensores piroeléctricos de infrarrojos que detectan la radiación infrarroja de los objetos. El sensor infrarrojo está constituido por un FET con un recubrimiento de fresnel sensible a la radiación infrarroja. Este sensor contiene un chip que convierte las lecturas del sensor infrarrojo en pulsos de voltaje. La Figura 1.20 muestra el sensor PIR [14].



(a) (b)

Figura 1.20. Sensor PIR. (a) Vista superior. (b) Vista Inferior.

El sensor PIR que se utilizará en el presente trabajo presenta 3 pines y funciona con voltaje de 3,3 [V]. En la siguiente Tabla 1.10 se muestra las características del sensor PIR a utilizarse.

Tabla 1.10. Especificaciones técnicas del sensor PIR [15].

Modelo	HC-SR501
Consumo de energía	3,3 – 5 [V]. 1[mA]
Rango de detección	3 – 7 [m]. 110°
Modos de disparo	Pulso y continuo
Temperatura de operación	-15°C – 70°C

2. METODOLOGÍA

Para realizar la comunicación entre LoRa Node 151 y el PC se utiliza un módulo ST-LINK. Se prepara el nodo mediante la instalación de su respectivo firmware y luego se procede con la programación del nodo utilizando los softwares STM32CubeIde [16] y STM32CubeProgrammer [17]. La Figura 2.1 muestra el diagrama de flujo para seguir la metodología.

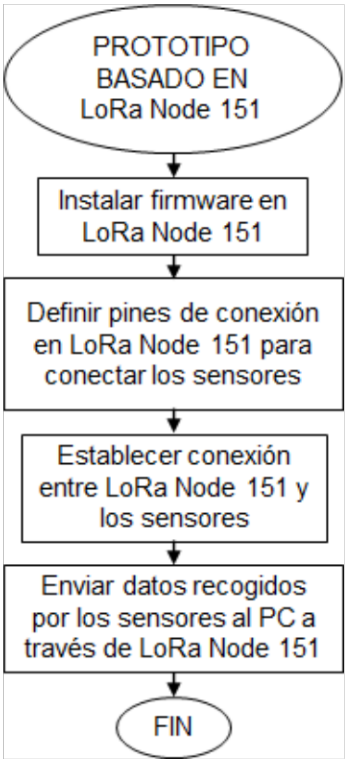


Figura 2.1. Diagrama de flujo para la metodología.

2.1. Preparación del LoRa Node 151

LoRa Node 151 es un dispositivo el cual posee su propio número de identificación ID (ID Identification), Mediante este ID se puede descargar información de la placa, así como ejemplos de programación de la página oficial [3]. Para obtener ese número de identificación, se debe descargar un firmware en el LoRa Node 151. Para cargar un programa al LoRa se utiliza el módulo ST-LINK que servirá de interfaz entre el LoRa Node 151 y el ordenador. Una vez instalado este firmware en el nodo, luego se descargará el ejemplo de programa Ping Pong de la página oficial para realizar pruebas de comunicación entre la PC y el nodo.

En la Figura 2.2 se muestra el diagrama de flujo a seguir para preparar el Lora Node 151.

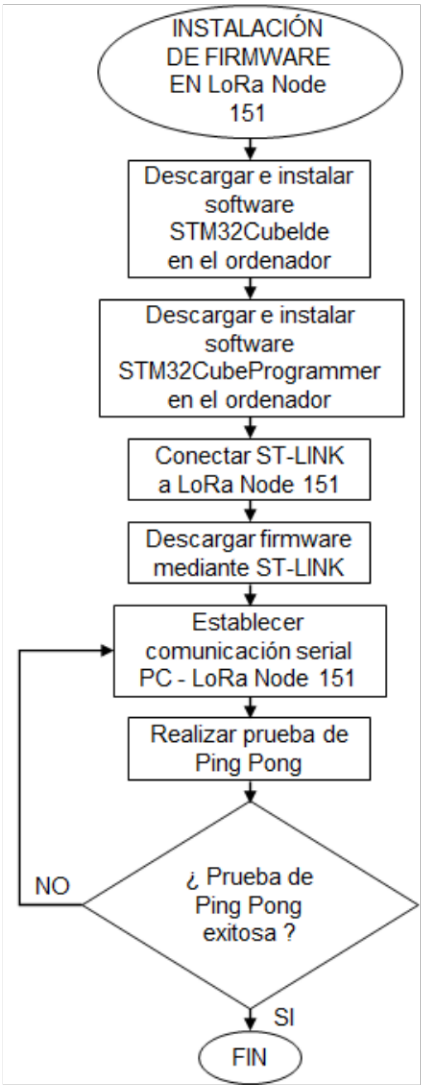


Figura 2.2. Diagrama de flujo para preparar LoRa Node 151.

2.1.2 Instalación de firmware en LoRa Node 151

Los materiales para instalar el firmware en el nodo se muestran en la Tabla 2.1.

Tabla 2.1. Materiales para instalar firmware en LoRa Node 151.

RECURSO	DESCRIPCIÓN
Laptop	Intel Core i5, 8GB RAM
LoRaWAN	LoRa Node 151
Programador	ST-LINK V.2
Cables	Dupont hembra-hembra
Programa 1	STM32Cubelde
Programa 2	STM32CubeProgrammer
Programa 3	STM32 ST-LINK Utility

Se descargan los softwares *STM32CubeIde*, *STM32CubeProgrammer*, STM32 ST-LINK Utility y el programa que permite obtener el ID, de la página oficial [16] [17], para ello, debe registrarse en la página y luego validar el correo. Se debe seleccionar la versión del programa y el sistema operativo que tiene el ordenador a utilizarse. En este caso se seleccionó la última versión y el sistema operativo Windows de 64bits, como se muestra en la Figura 2.3.

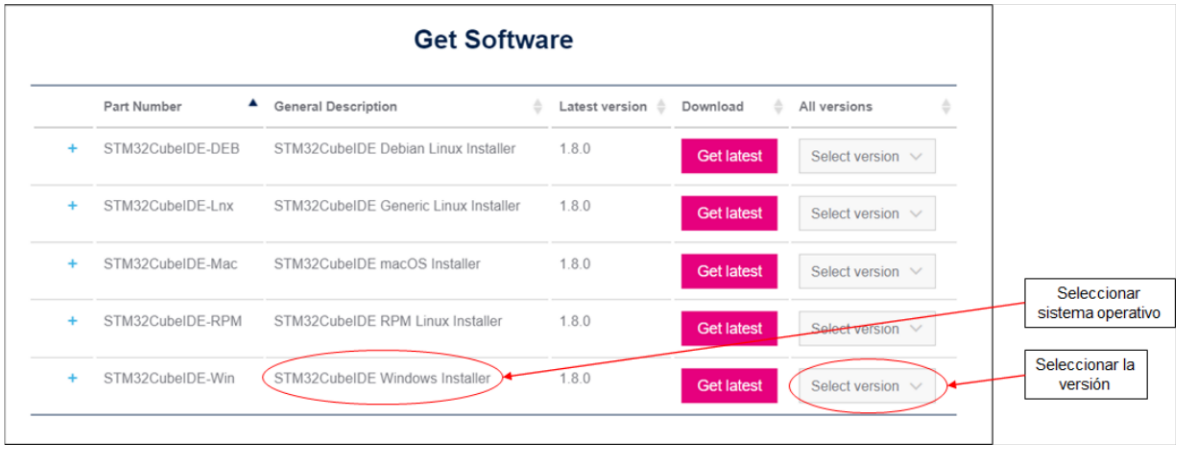


Figura 2.3. Descarga del programa STM32CubeIde.

Con los programas instalados, ejecutar el programa *STM32CubeIde*, conectar el módulo ST-LINK al LoRa Node 151, siguiendo el orden de pines descritos en la Tabla 2.2.

Tabla 2.2. Pines de conexión ST-LINK y LoRa Node 151.

Pin ST-LINK	Pin LoRa Node 151
3,3V	3,3V
GND	GND
SWCLK	PA14
SWDIO	PA13

La Figura 2.4 muestra la conexión entre el ST-LINK y el LoRa Node 151 mediante cables dupont hembra – hembra. En el ANEXO 1 se presenta el diagrama de pines del LoRa Node 151. El nodo también permite la conexión utilizando un cable microUSB, por lo que se debe tener cuidado en cuanto a sobrealimentar la placa, ya que esta sobrealimentación podría dañar al LoRa Node 151, por tal razón, si se conecta el cable microUSB, debe suprimirse el uso del cable que se conecta en el pin 3,3V.

Figura 2.4. Conexión entre ST-LINK y LoRa Node 151.

Con el módulo ST-LINK conectado a un puerto USB del ordenador, el software *STM32CubeIde* detectará automáticamente la placa conectada. En el STM32CubeIde se debe seleccionar el programa que se descargará en el nodo, en este caso se selecciona el programa que permite obtener el ID de la placa. Debe seleccionar en la pestaña *File* la opción abrir un proyecto desde los archivos del sistema, como se muestra en la Figura 2.5.

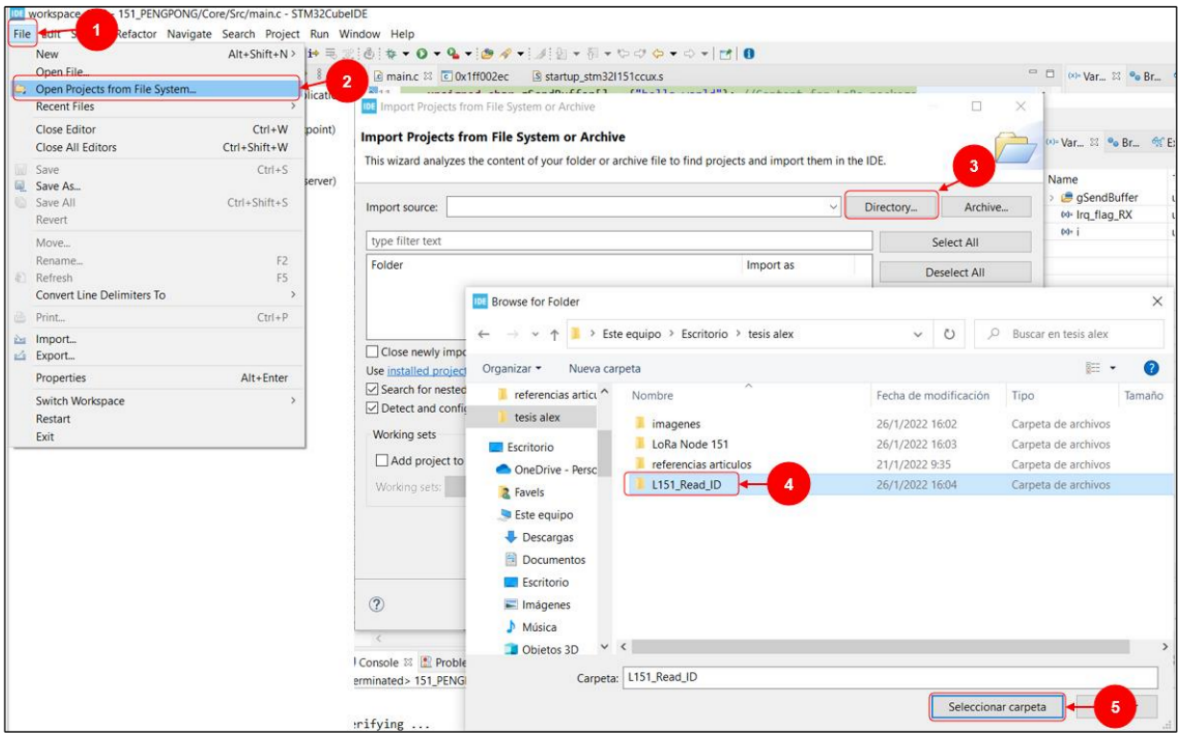


Figura 2.5. Pasos para seleccionar un programa a cargar en el nodo.

Para descargar el frimware a la placa, click en el ícono *Debuger*, como se muestra en la Figura 2.6. Cuando se haya finalizado la carga del programa, se mostrará un mensaje de

verificación de descarga satisfactoria. Con este paso realizado también se comprueba que ya existe comunicación entre ST-LINK y el LoRa Node 151.

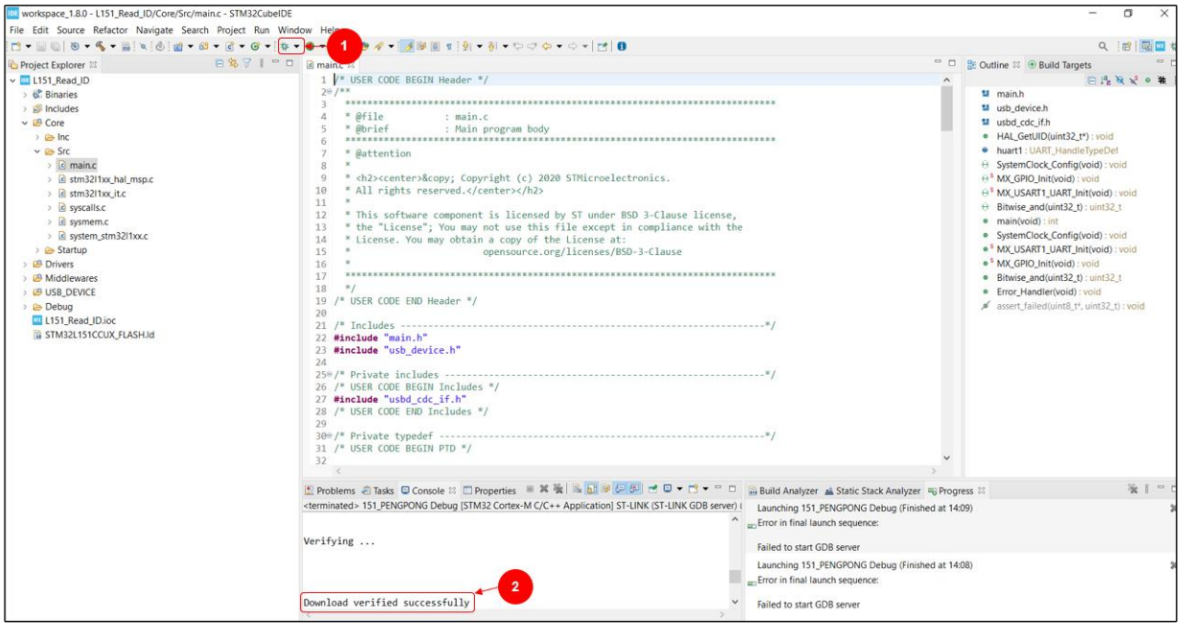


Figura 2.6. Descarga del firmware al LoRa Node 151.

En el caso que no se descargue el firmware al LoRa Node 151, se debe actualizar, en primer lugar, el firmware del ST-LINK, para ello se debe utilizar el software STM32 ST-LINK Utility. Una vez ejecutado este software se debe conectar el ST-LINK y proceder a actualizar, seleccionar la opción actualizar firmware, como se ilustra en la Figura 2.7.

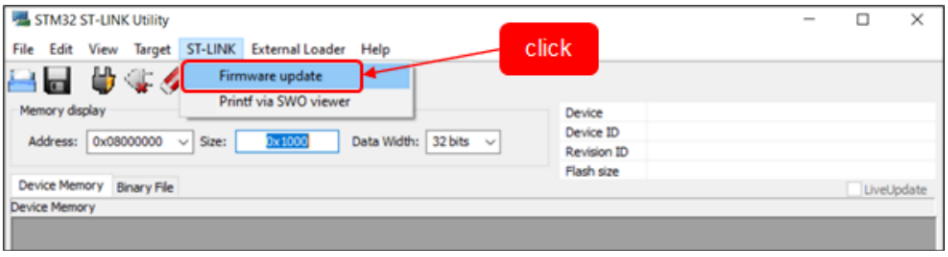


Figura 2.7. Actualización del firmware del ST-LINK con STM32 ST-LINK Utility.

Una vez descargado el firmware para obtener el ID en el LoRa Node 151, se retira el módulo ST-LINK y se conecta un cable microUSB. Para leer el ID se necesita de un emulador de terminal que soporte comunicación serial, en este caso se utilizará el emulador *PuTTY*. Se debe configurar el puerto serial con los mismos parámetros que el programa descargado al nodo, en la figura Figura 2.8 se muestra la porción de código que contiene los parámetros configurados para establecer la comunicación serial del nodo.

```
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}
```

Figura 2.8. Parámetros para la comunicación serial del LoRa Node 151.

Por consiguiente, se configura el puerto serial del emulador *PuTTY* con los parámetros mostrados en la Figura 2.9.

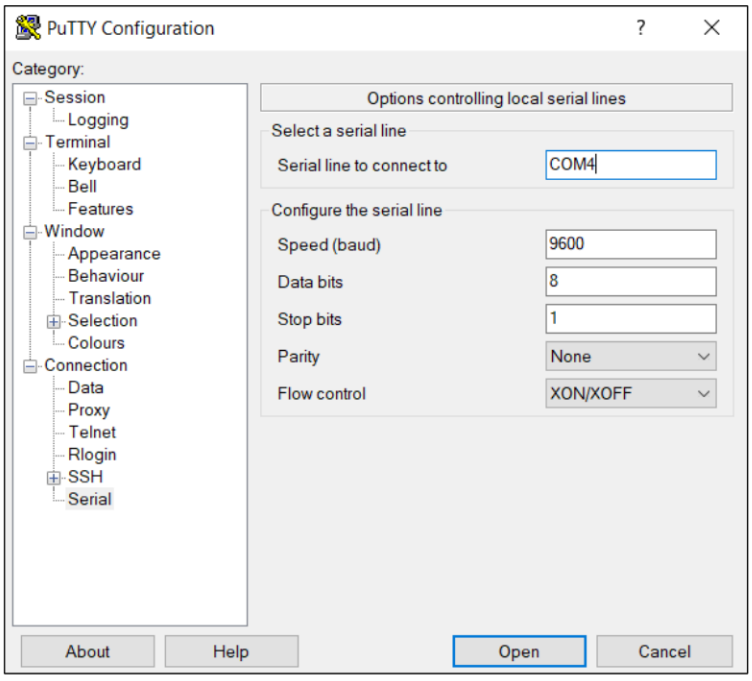


Figura 2.9. Parámetros para el puerto serie del emulador *PuTTY*.

A continuación, se obtiene el identificador del LoRa Node 151 mediante el emulador *PuTTY* como se muestra en la Figura 2.10.

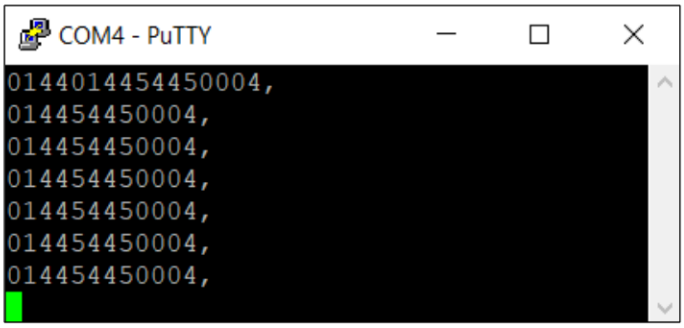


Figura 2.10. Obtención del ID del LoRa Node 151 mediante el emulador PuTTY.

Con este número identificador, se busca los recursos, para el LoRa Node 151 que se utiliza en el presente trabajo, en la *URL*: https://resource.heltec.cn/search [3], como se muestra en la siguiente Figura 2.11.

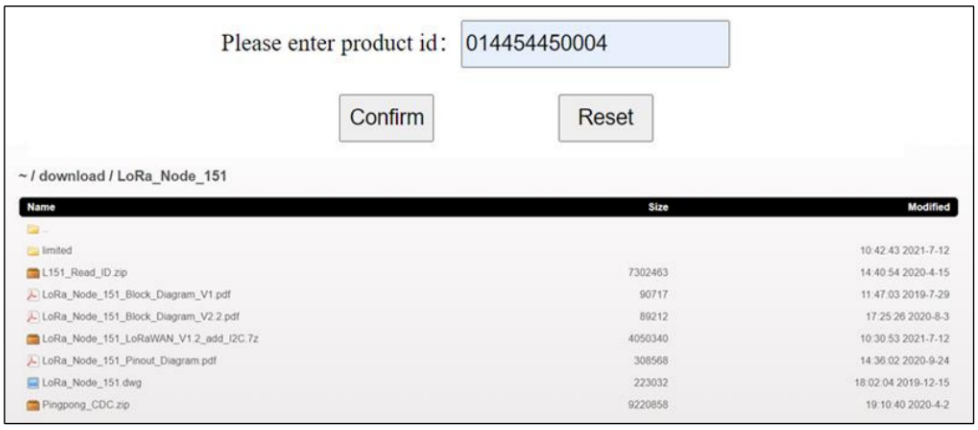


Figura 2.11. Descarga de recursos mediante ID del LoRa Node 151.

Finalizado este proceso se ha comprobado que existe comunicación serial entre LoRa Node 151 y el ordenador, por consiguiente, se continúa con la prueba de Ping Pong, esta prueba permite comprobar la comunicación entre nodos utilizando el protocolo LoRa.

2.1.3 Prueba de Ping Pong

Los materiales que se van a utilizar en esta prueba se detallan en la Tabla 2.3. Cabe destacar que se necesitan dos ordenadores para comprobar que se envía información mediante la interfaz RF del LoRa Node 151. Ya que la descarga del programa en el nodo se realiza con el software STM32CubeIde, no es necesario tener instalado este programa en ambos ordenadores, en un solo ordenador se realizará la descarga del ejemplo Ping

Pong a los dos nodos LoRa. A diferencia del modo de descargar el programa en los nodos, para realizar la configuración del protocolo LoRa en los nodos, se llevará a cabo mediante un emulador de puerto serial, así mismo la lectura de la información enviada entre nodos se visualizará en dicho emulador, por tanto, sí es necesario tener instalado el emulador de puerto serial en ambos ordenadores.

Tabla 2.3. Recursos necesarios para realizar la prueba de Ping Pong.

RECURSO	CANTIDAD
Ordenador	2
LoRa Node 151	2
Cable microUSB	2
Antena RF	2
ST-LINK	1

A continuación, en el software STM32CubeIde seleccionar el ejemplo Ping Pong, conectar el LoRa Node 151 al ordenador mediante el módulo ST-LINK y descargar el programa en el nodo. Concluida la descarga con el mensaje de verificación de descarga satisfactoria, se desconecta el módulo ST-LINK del nodo y en su lugar se conecta un cable microUSB, enseguida realizar la conexión del cable al ordenador, se verifica en que puerto del ordenador se reconoció al nodo. Se configura los parámetros de comunicación serial en el emulador *PuTTY* según los parámetros establecidos en el nodo.

Para utilizar correctamente el ejemplo Ping Pong, se debe ingresar parámetros de configuración del protocolo LoRa, para establecer la comunicación entre los nodos mediante la interfaz RF. Los parámetros se especifican en la Tabla 2.4, esta configuración se la realiza mediante un terminal serial, por consiguiente, se utiliza el emulador *PuTTY* para la configuración del LoRa Node 151. Se debe tomar en cuenta que se debe establecer los mismos parámetros en los dos nodos LoRa que van a llevar a cabo la prueba de Ping Pong.

Tabla 2.4. Parámetros para configurar el protocolo LoRa en los nodos Lora Node 151.

	PARÁMETRO	VALOR
1	Banda de frecuencia	868 [MHz]
2	Factor de dispersión	12
3	Tasa de codificación	1
4	Potencia de salida	18 [dBm]

En la Figura 2.12 se muestra los parámetros configurados en ambos nodos mediante el emulador *PuTTY*.

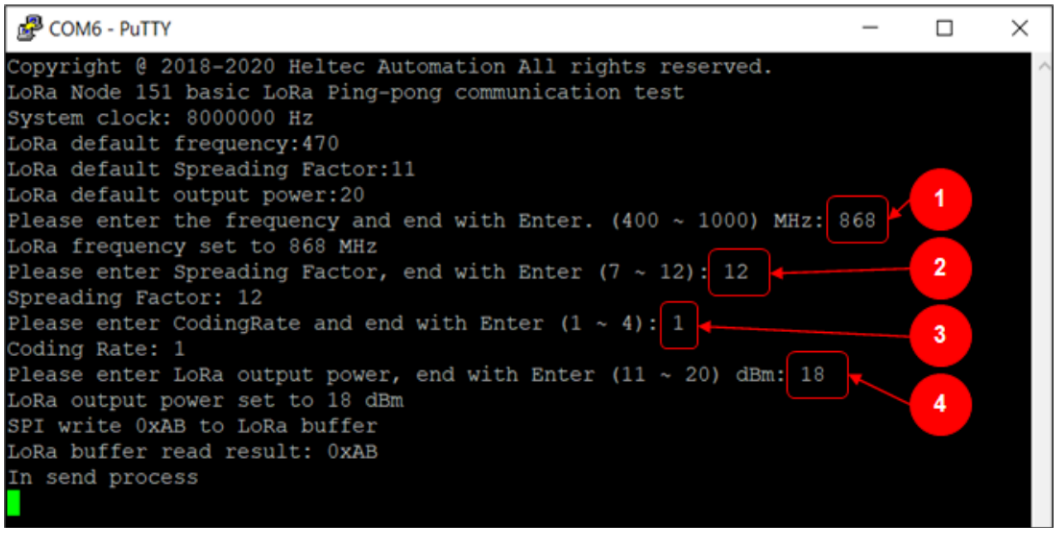


Figura 2.12. Configuración del protocolo LoRa en los nodos LoRa Node 151.

Los dos nodos se conectan a dos ordenadores con sus respectivos emuladores PuTTY, y se puede verificar el funcionamiento de la prueba Ping Pong. En la siguiente Figura 2.13 se muestra la conexión de los nodos LoRa Node 151 a los ordenadores, cada uno con su respectivo cable microUSB.

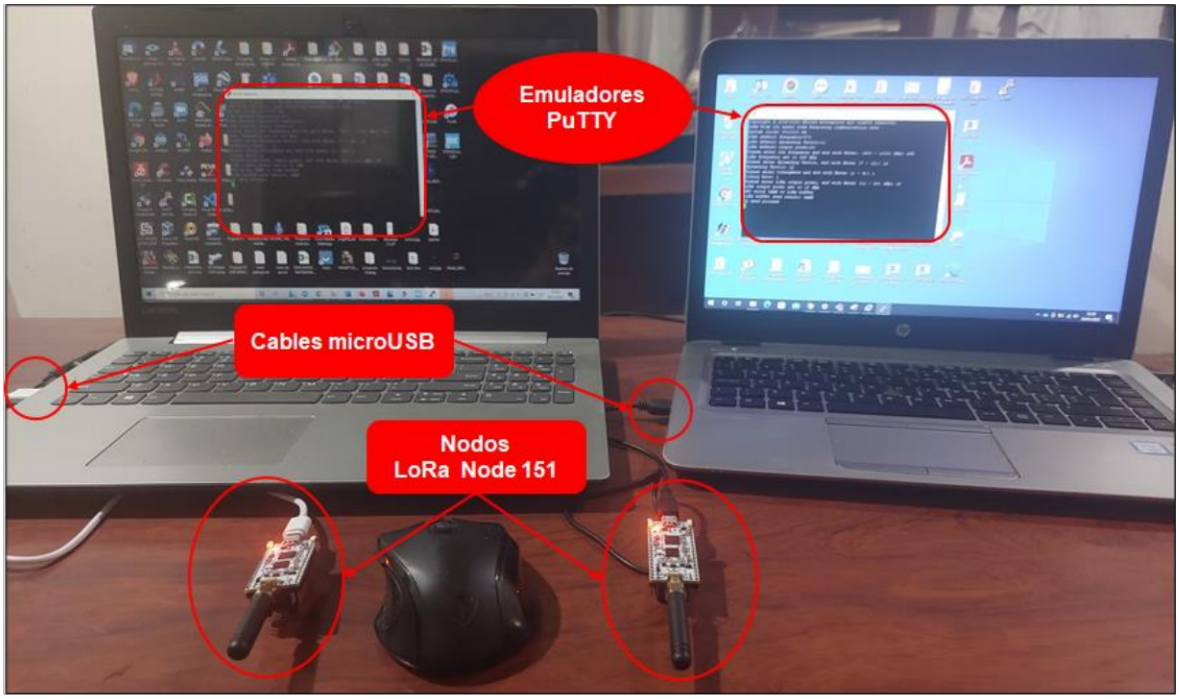
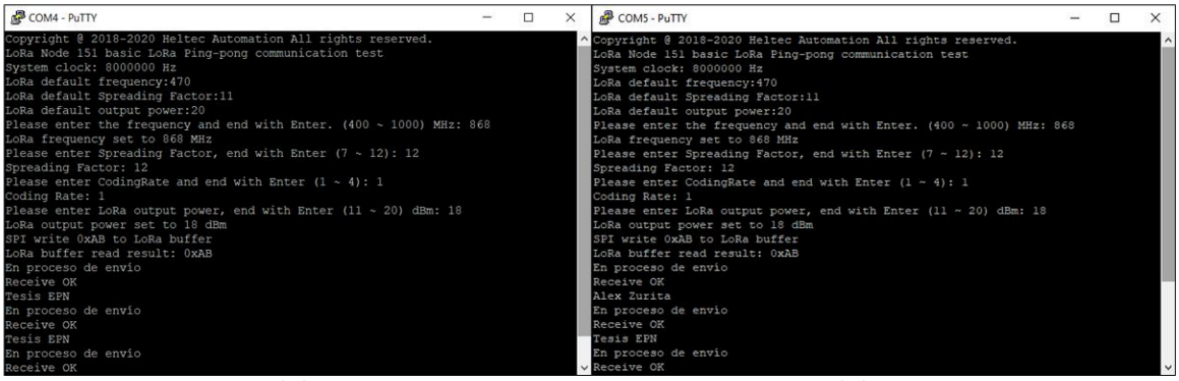


Figura 2.13. Conexión de los dos LoRa Node 151 a dos ordenadores.

En este proceso, el primer LoRa Node 151 envía el mensaje “Tesis EPN”, cuando el segundo Lora Node 151 al recibir este mensaje contesta con otro mensaje “Alex Zurita”, el ciclo continúa indefinidamente concluyendo así esta prueba de conexión mediante la interfaz RF utilizando el protocolo LoRa. La siguiente Figura 2.14 muestra los mensajes recibidos en sus respectivos emuladores *PuTTY.*



(a) (b)

Figura 2.14. Verificación de mensajes. (a) Primer nodo en COM4. (b) Segundo nodo en COM5.

2.2. Desarrollo del prototipo

El prototipo que se desarrolla en el presente trabajo consiste en obtener las coordenadas del sensor GPS y enviarlas desde el primer nodo a un segundo nodo mediante la interfaz RF utilizando el protocolo LoRa. Las coordenadas se visualizarán en un ordenador con emulador *PuTTY*.

2.2.1. Diagrama de flujo del funcionamiento del prototipo

El sensor de movimiento PIR se conecta a un pin del nodo LoRa, cada vez que este sensor detecte que un movimiento, envía un pulso hacia el nodo LoRa, el cual se encarga de realizar la petición de coordenadas al Arduino UNO mediante interfaz UART. Las coordenadas se obtienen del sensor GPS mediante una placa Arduino UNO, utilizando la interfaz UART, en la placa Arduino se utiliza la librería TinyGPS desarrollada por Mikal Hart [18], se utiliza el protocolo UBX para procesar los datos recibidos de los satélites más cercanos. Estas coordenadas se envían hacia LoRa Node 151 mediante interfaz UART, LoRa se encarga de insertar en la carga útil estas coordenadas para luego ser enviadas mediante la interfaz RF utilizando el protocolo LoRa hacia otro nodo LoRa.

El segundo nodo LoRa recibe la trama que contiene las coordenadas GPS, extrae los datos de la carga útil y los envía a un ordenador mediante interfaz UART para que estos puedan

ser visualizados en un emulador PuTTY. En la siguiente Figura 2.15 se muestra el diagrama de flujo para la lógica del problema propuesto en este trabajo.

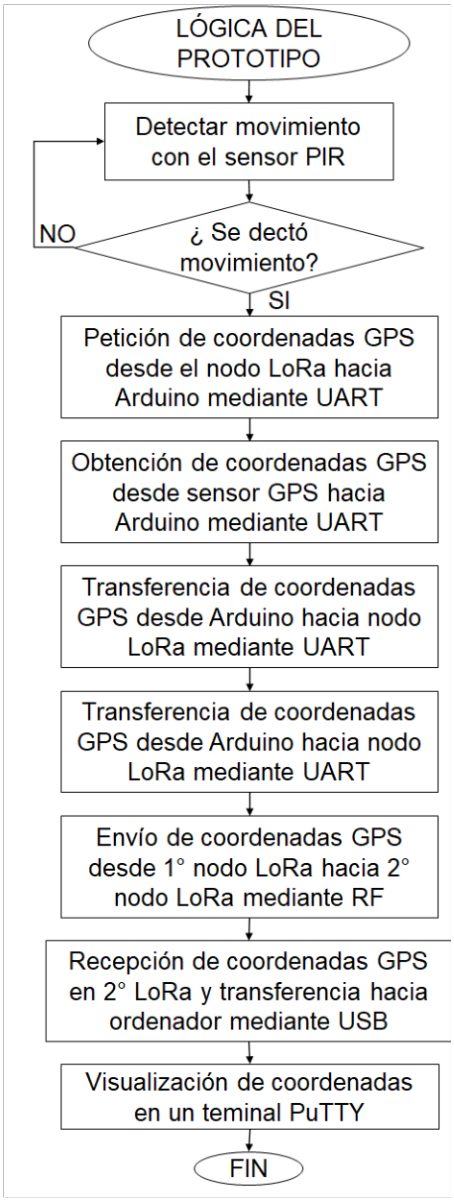


Figura 2.15. Diagrama de flujo de la lógica del prototipo.

La Figura 2.16 muestra un diagrama de bloques con los dispositivos interconectados entre sí y las interfaces utilizadas para la comunicación entre dispositivos. Cabe destacar que tanto la placa Arduino UNO como LoRa Node 151, poseen en sus periféricos varias interfaces UART. En el caso de la placa Arduino UNO, se utiliza la librería SoftwareSerial para designar dos interfaces UART, una interfaz se comunica con el sensor GPS y la otra interfaz con LoRa Node 151.

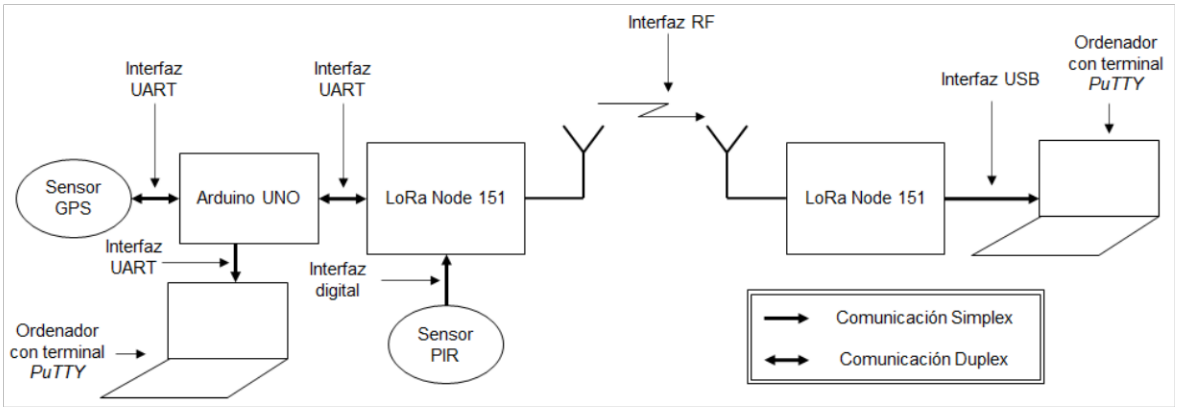


Figura 2.16. Diagrama de bloques del trabajo propuesto.

Ya que Arduino UNO se encarga de la comunicación con el sensor GPS, se realiza un programa para esta placa y se utiliza la librería TinyGPS [18]. Por otra parte, Lora Node 151 se comunica con Arduino UNO y con otro LoRa Node 151 utilizando el protocolo Lora, por consiguiente, también se desarrolla un programa para cada LoRa Node 151.

2.2.2. Programa para Arduino UNO

La placa Arduino UNO se comunica con el sensor GPS mediante interfaz UART y con LoRa Node 151 también mediante interfaz UART, por consiguiente, se utiliza la librería SoftwareSerial que incluye el IDE de Arduino. Arduino UNO recibirá un dato serial proveniente de LoRa Node 151, el cual, le indicará que Arduino debe iniciar el sensor GPS para la obtención de las coordenadas GPS. Cuando el sensor GPS haya concluido el proceso de obtención de coordenadas, enviará estas coordenadas hacia Arduino por interfaz UART, esta interfaz se designará a los pines 3 y 4 mediante la librería SoftwareSerial. Luego, estas coordenadas serán enviadas hacia LoRa Node 151 mediante otra interfaz UART, designada en los pines 8 y 9. La Tabla 2.5 muestra la designación de pines para Arduino UNO.

Tabla 2.5. Designación de pines de Arduino UNO.

PIN	DESIGNACIÓN	DISPOSITIVO DESTINO
3,4	UART 1	Sensor GPS
8,9	UART 2	LoRa Node 151
GND	Señal tierra	Sensor GPS, LoRa Node 151
3,3V	Señal Vdd	Sensor GPS

La siguiente Figura 2.17 muestra el diagrama de flujo para realizar la codificación en la placa Arduino UNO.

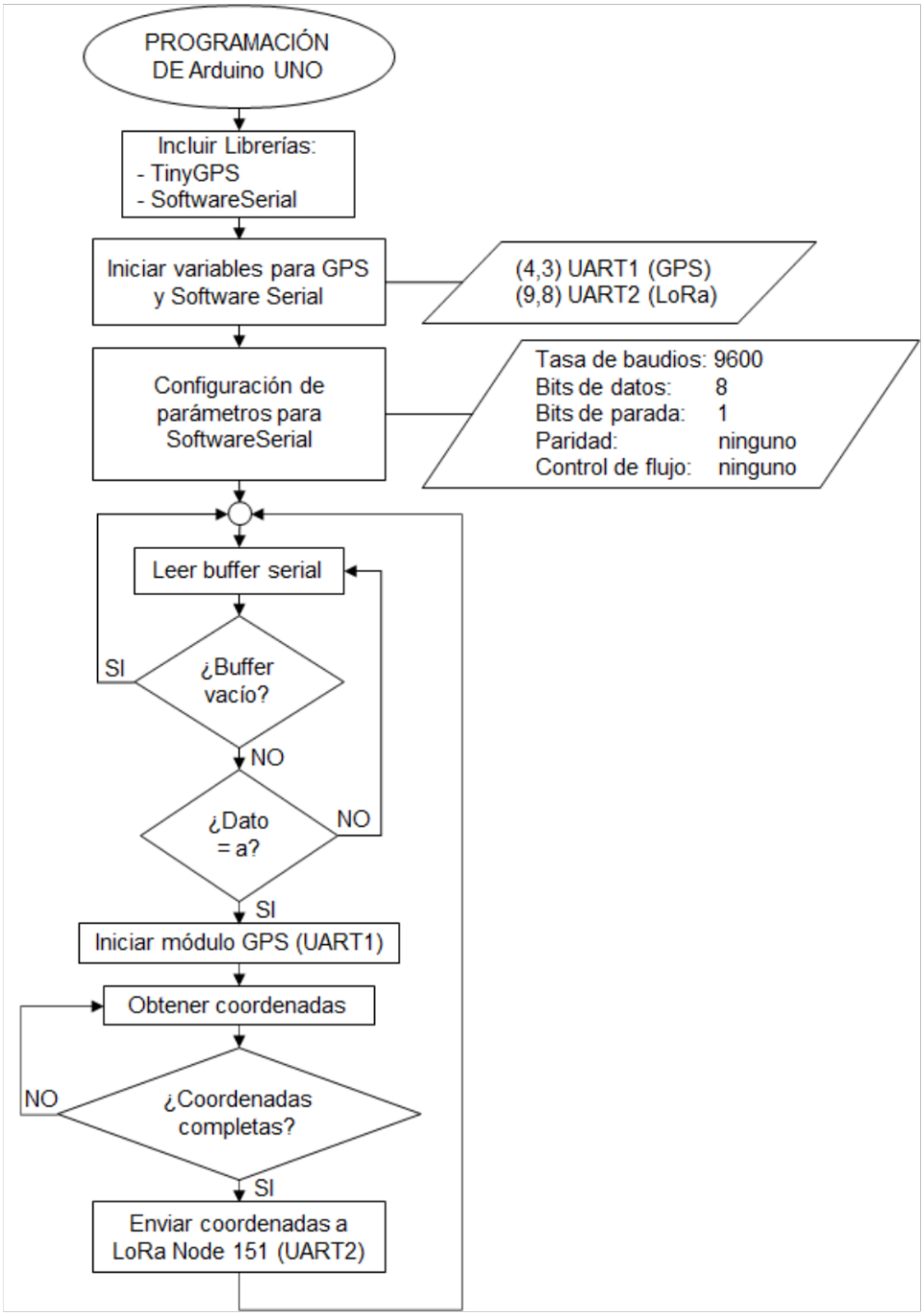


Figura 2.17. Diagrama de flujo del programa para Arduino UNO.

A continuación, se presenta el código desarrollado en el IDE de Arduino.

```
#include <SoftwareSerial.h>
#include <TinyGPS.h>
```

```
TinyGPS gps;                    //Definición de variable para utilizar libreria del módulo GPS
SoftwareSerial serial_GPS(4, 3);    //Definicion de pines para comunicación serial con módulo GPS
SoftwareSerial serial_LoRa(9, 8);    //Definicion de pines para comunicación serial con LoRa Node 151


void setup()
{
  pinMode(4, INPUT);
  pinMode(3, OUTPUT);
  pinMode(9, INPUT);
  pinMode(8, OUTPUT);

  Serial.begin(9600);             //Serial entre terminal de ordenador y arduino
  serial_GPS.begin(9600);         //Serial entre arduino y módulo GPS
  serial_LoRa.begin(9600);        //Serial entre arduino y LoRa Node 151

  Serial.print("Escuela Politécnica Nacional");
  Serial.println("Prototipo con LoRa Node 151");
  Serial.println("Alex Zurita");
}

void loop()
{

  bool datosGPS = false;                    //Variable para verificar datos de GPS
  bool datosLoRa = false;                   //Variable para verificar datos de LoRa Node 151
  unsigned long datoscaracteres;
  unsigned short cadenas;
  unsigned short fallo;
```

```
while (serial_LoRa.available())
   {
     char l = serial_LoRa.read();
     if (l=='a')
       datosLoRa = true;
   }

     if (datosLoRa)
 {

 //Reportar valores obtenidos por GPS por un segundo
 for (unsigned long start = millis(); millis() - start < 1000;)
 {
   while (serial_GPS.available())
   {
     char c = serial_GPS.read();
     if (gps.encode(c))
       datosGPS = true;
   }
 }

 if (datosGPS)
 {
   float GPS_lat;
   float GPS_long;
   unsigned long age;
   gps.f_get_position(&GPS_lat, &GPS_long, &age);
   Serial.print("LATITUD=");
   Serial.print(GPS_lat == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : GPS_lat, 6);
   Serial.print(" LONGITUD=");
   Serial.print(GPS_long == TinyGPS::GPS_INVALID_F_ANGLE ? 0.0 : GPS_long, 6);
   serial_LoRa.write(GPS_lat);
```

```
    serial_LoRa.write(GPS_long);
  }

  gps.stats(&datoscaracteres, &cadenas, &fallo);
  Serial.print(" datoscaracteres=");
  Serial.print(datoscaracteres);
  Serial.print(" cadenas=");
  Serial.print(cadenas);
  Serial.print(" CSUM ERR=");
  Serial.println(fallo);
  if (datoscaracteres == 0)
    Serial.println("** No characters received from GPS: check wiring **");
}
}
```

2.2.3. Programa para LoRa Node 151

EL presente trabajo utiliza dos Lora Node 151, que se comunican entre si mediante la interfaz RF utilizando el protocolo LoRa. El primer LoRa Node 151 mantiene conexión digital con el sensor de movimiento PIR a través de un GPIO ubicado en el pin PB5, este pin es configurado como entrada digital sin *PULL-UP*.

El sensor PIR emite una señal en ALTO (3,3V) al pin PB15 de LoRa Node 151, este pin al detectar un estado ALTO indica que debe generarse un estado ALTO en el pin PB14 previamente configurado como salida digital sin PULL-UP, y encenderá un diodo *LED* conectado a este pin a través de una resistencia con el fin de indicar visualmente cuando el sensor PIR ha detectado movimiento. Al mismo tiempo, se envía mediante comunicación serial en la interfaz UART1 el carácter “a” hacia Arduino UNO, este carácter indica que el nodo LoRa está realizando la petición de coordenadas GPS a Arduino UNO. Luego, el nodo LoRa queda a la espera de recibir las coordenadas en su buffer serial, cuando recibe las coordenadas GPS, inserta estos datos en la carga útil de la trama para luego enviar esta trama a través de la interfaz RF utilizando el protocolo LoRa hacia el segundo LoRa Node 151.

El segundo LoRa Node 151 se encuentra programado con los mismos parámetros en la interfaz RF que el primer nodo LoRa, también está configurado con una interfaz USB para comunicarse con un ordenador. Al recibir la trama, extrae de la carga útil las coordenadas y las envía serialmente al ordenador y finalmente se puede observar estas coordenadas GPS en un emulador PuTTY. La Tabla 2.6 muestra la designación de pines para los dos nodos LoRa.

Tabla 2.6. Designación de GPIOs a los dos LoRa Node 151.

	GPIO	**DESIGNACIÓN**	**DISPOSITIVO DESTINO**
Primer LoRa Node 151	PB15	Entrada digital sin *PULL-UP*	Sensor *PIR*
	PB14	Salida digital sin *PULL-UP*	Diodo *LED*
	PA10, PA9	UART1	Arduino UNO
	ANT	RF	Segundo LoRa Node 151
Segundo LoRa Node 151	-	USB	Ordenador
	ANT	RF	Primer LoRa Node 151

La Figura 2.18 muestra el diagrama de flujo para programar el primer LoRa Node 151, los sensores utilizados en este trabajo se alimentan con un voltaje de 3,3V. Con este parámetro se asegura que todo el circuito trabaje con un voltaje de 3,3V asegurando de esta manera que el nodo LoRa no sufra daños por exceso de voltaje.

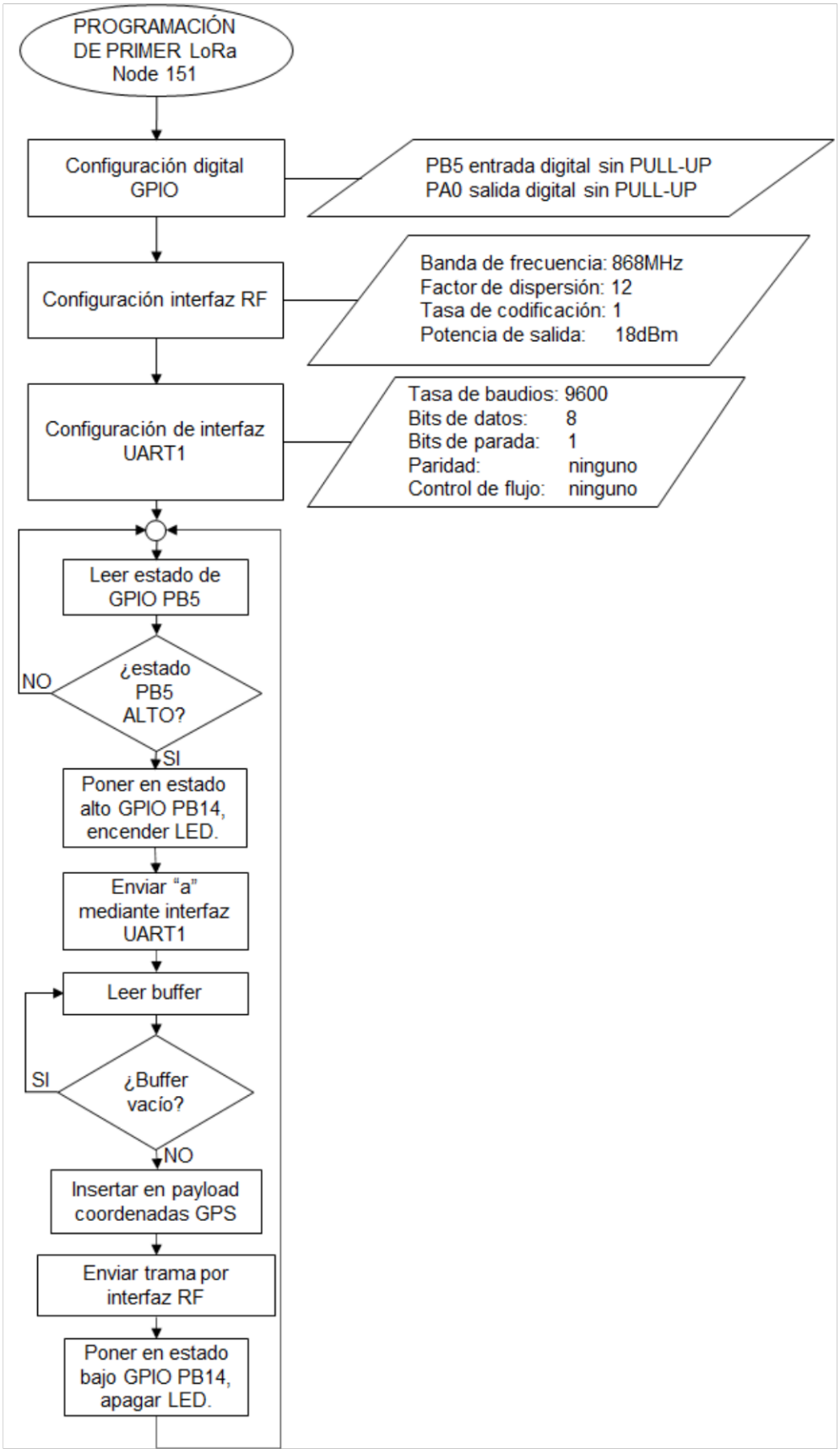


Figura 2.18. Diagrama de flujo para programar primer LoRa Node 151.

El código se desarrolla sobre el ejemplo Ping Pong, añadiendo la comunicación UART y la programación para controlar el sensor PIR y el diodo LED. A continuación, se presenta el código correspondiente a la programación de la detección de movimiento y el envío del carácter “a” para habilitar el módulo GPS y la posterior obtención de las coordenadas GPS.

```
#include "main.h"
#include "usb_device.h"
#include "board.h"
#include "SX127x.h"


while (1)
 {
        if (HAL_GPIO_ReadPin(sensor_gps_GPIO_Port, sensor_gps_Pin)==0) {
                        HAL_GPIO_WritePin(led_GPIO_Port, led_Pin, 1);
                        HAL_UART_Transmit(&huart1, datot, 1, 50);
                        if (!HAL_UART_Receive(&huart1, dator, sizeof(dator),
HAL_MAX_DELAY)) {



                    FUN_RF_SENDPACKET(dator,sizeof(dator));
                    SX127xReadBuffer( REG_LR_IRQFLAGS );
                    HAL_Delay(500);
                    while(1)
                    {
                            if((Irq_flag_RX = SX127xReadBuffer( REG_LR_IRQFLAGS
)))
                             {
                                        SX127x_Interupt();
                                        SX127xWriteBuffer( REG_LR_IRQFLAGS
,Irq_flag_RX);
                                        if(RX_LoRa_Flag)
                                        {
                                                HAL_Delay(200);
                                                RX_LoRa_Flag = 0;
                                                break;
                                        }
                            }
                    }
```

```
                    HAL_Delay(5000);
            }
            }
}
```

La programación que se utiliza en el segundo LoRa Node 151 se indica en el diagrama de flujo de la Figura 2.19.

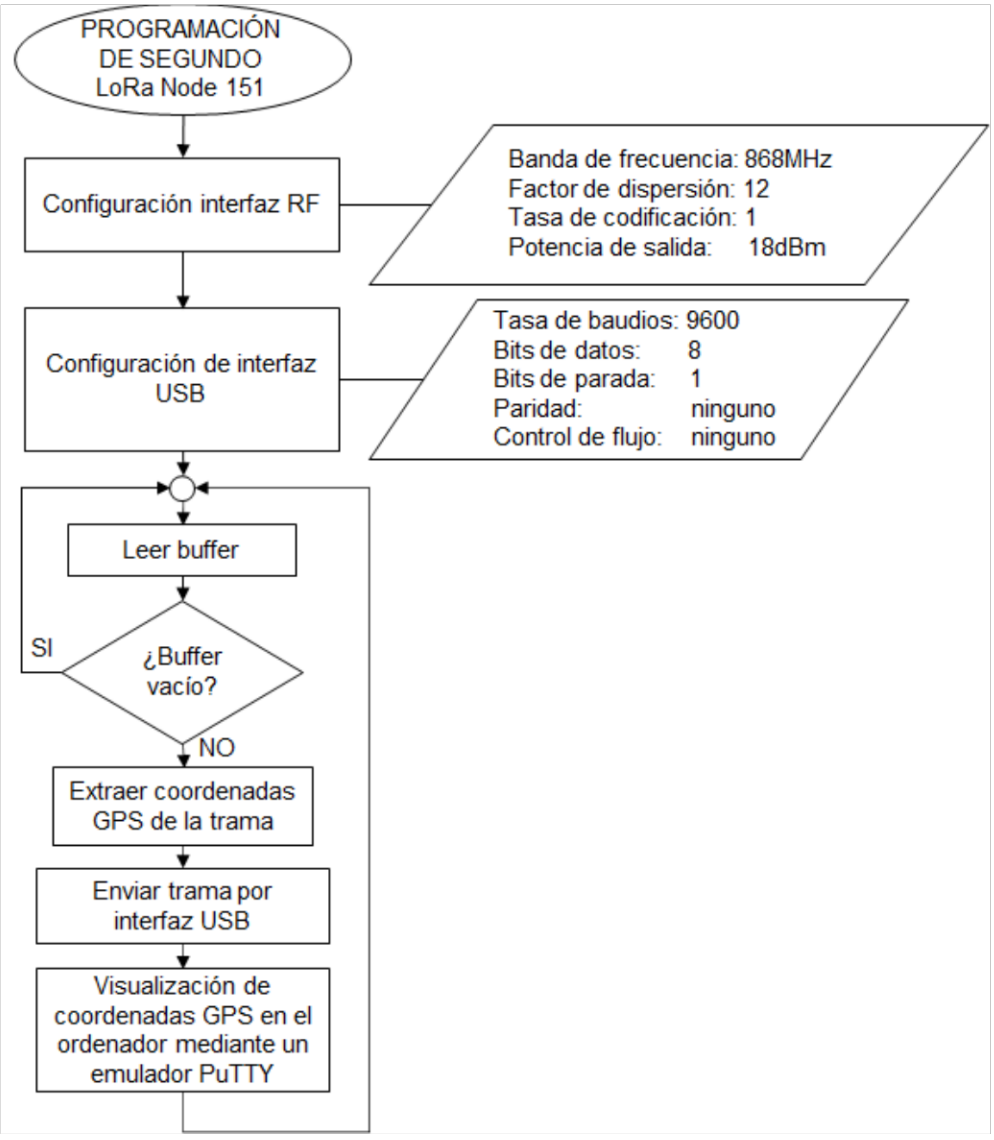


Figura 2.19. Diagrama de flujo de la programación para el segundo LoRa Node 151.

A continuación, se presenta el código correspondiente al archivo main.c del segundo LoRa Node 151.

```
#include "main.h"
#include "board.h"
#include "SX127x.h"

extern uint16_t RX_LoRa_Flag ;
uint16_t                begin_flag   = 0;
uint16_t                num          = 0;

int main(void)
{
        unsigned char gSendBuffer[] = {""}; //Variable donde se almacenarán las
coordenadas
        unsigned char Irq_flag_RX = 0;

        Board_Init();
        usb_printf("Facultad de Ingeniería Electrónica y Telecomunicaciones\r\n");
        HAL_Delay(10);
        usb_printf("Alex Zurita\r\n");
        HAL_Delay(10);
        usb_printf("Protocolo LoRa Node 151\r\n");
        HAL_Delay(10);

        SX127x_RLEN = 0;
        SX127xReset();
        Parameter_Init();
        SX127xLORA_INT();

        uint16_t i   = 0;


        while (1)
        {
                usb_printf("Recibiendo datos\r\n");
                SX127xReadBuffer( REG_LR_IRQFLAGS );
                HAL_Delay(500);
                while(1)
                {
                        if((Irq_flag_RX = SX127xReadBuffer( REG_LR_IRQFLAGS )))
                          {
                                SX127x_Interupt();
                                SX127xWriteBuffer( REG_LR_IRQFLAGS
,Irq_flag_RX);
                                if(RX_LoRa_Flag)
                                {
                                        HAL_Delay(200);
                                        RX_LoRa_Flag = 0;
                                        break;
                                }
                          }
                }
                HAL_Delay(5000);
        }
}
```

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presenta el diseño completo del prototipo propuesto con el LoRa Node 151. Mediante un emulador de terminal PuTTY se observa las coordenadas GPS enviadas desde el primer LoRa Node 151 hacia el segundo LoRa Node 151. Las coordenadas también se verifican en un emulador PuTTY en el primer nodo, es decir que en el lado del transmisor también se colocará un ordenador con el fin de visualizar los datos que se enviarán posteriormente, y comprobar en el lado de recepción que efectivamente se tienen los mismos datos.

Se realizan las pruebas colocando la antena en sus respectivos nodos, luego se toman los datos ubicando el segundo LoRa a diferentes distancias del primer LoRa con el fin de obtener errores en la lectura de las coordenadas en el segundo LoRa, y con esto, determinar la distancia a la que se realiza la comunicación RF satisfactoriamente.

3.1 Implementación completa del prototipo

La siguiente Figura 3.1 presenta el diagrama esquemático del prototipo. El terminal del lado del receptor se conecta a la placa Arduino UNO, en este terminal se observa las coordenadas que Arduino obtiene del módulo GPS, este dato será considerado como el valor medido para el cálculo del error relativo. El valor real será considerado en el lado del receptor, es decir los datos que se leen en el terminal PuTTY del segundo nodo.

En el diseño se incluye un diodo LED con el propósito de indicar de manera visual que se detectó movimiento y en ese instante inicia el proceso de adquirir las coordenadas del sensor GPS. El LoRa Node 151 se conecta a un ordenador mediante la interfaz USB, por consiguiente, el voltaje presente en cada GPIO del nodo se tiene un voltaje de 3,3V. El diodo LED empleado en el presente trabajo necesita una corriente de 15mA para su iluminación adecuada, con estos datos se calcula la resistencia que se debe colocar en el GPIO PB14 del nodo LoRa. Mediante la ley de Ohm se calcula el valor de la resistencia:

$$R = \frac{V}{I}\ [\Omega] \tag{1.6}$$

$$R = \frac{3{,}3}{0{,}015}\ [\Omega]$$

$$R = 220\ [\Omega]$$

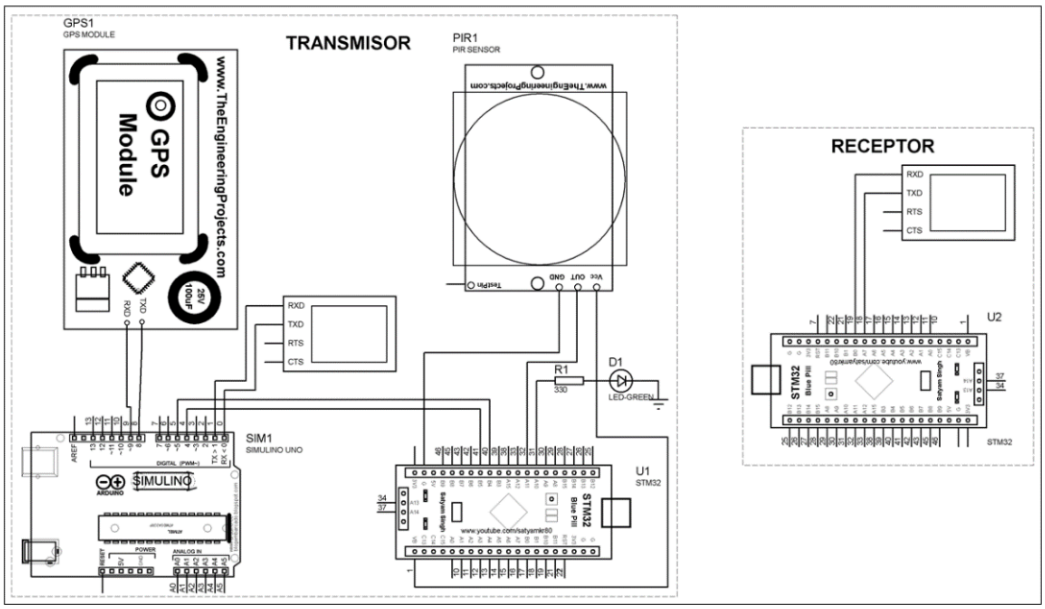


Figura 3.1. Diagrama esquemático del prototipo.

Calculado el valor de la resistencia, se implementa el prototipo, en un principio se realiza la primera prueba de transmisión sobre un escritorio, posteriormente se realiza la transmisión en un área despejada. La siguiente Figura 3.2 muestra la implementación final del prototipo. La prueba sobre escritorio muestra la obtención de las coordenadas GPS del

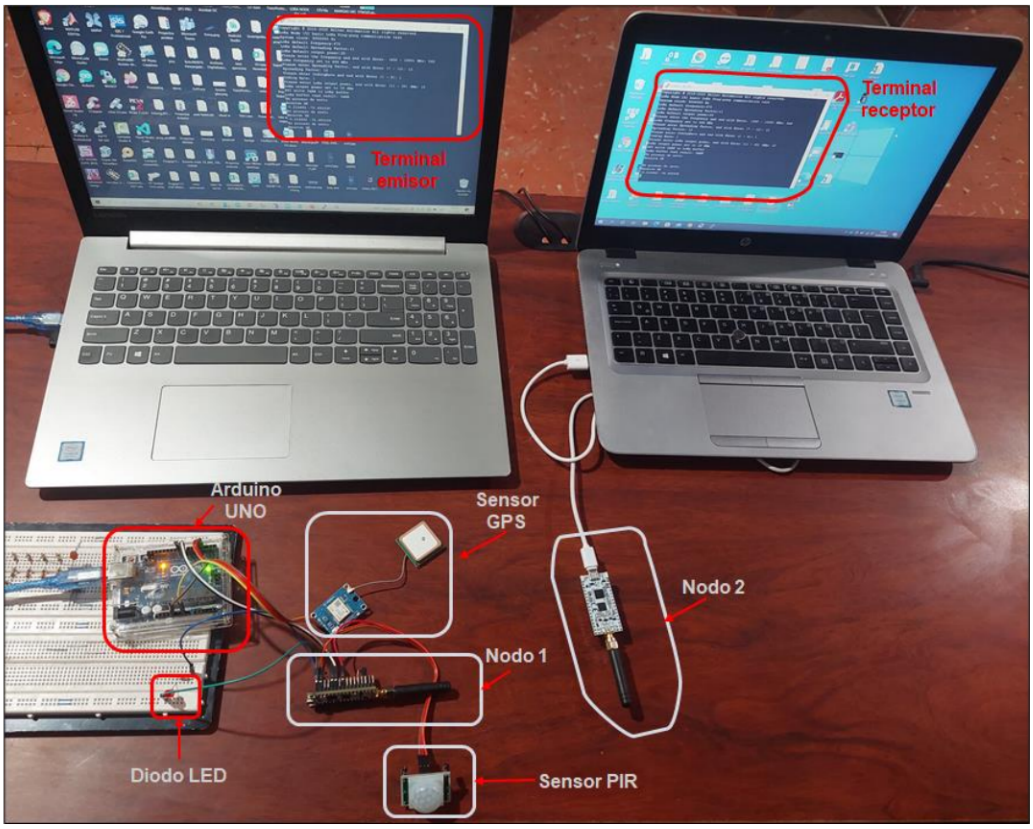
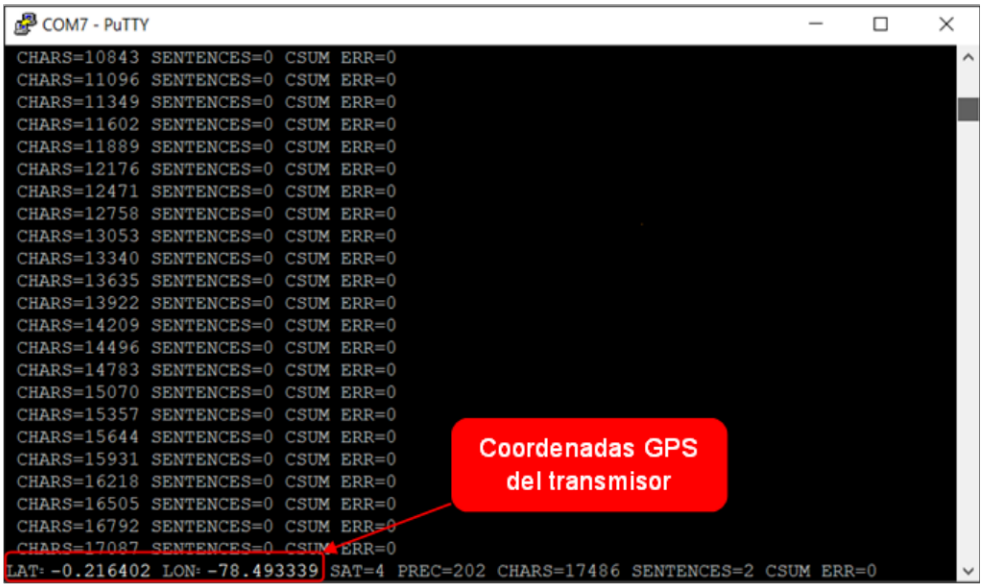


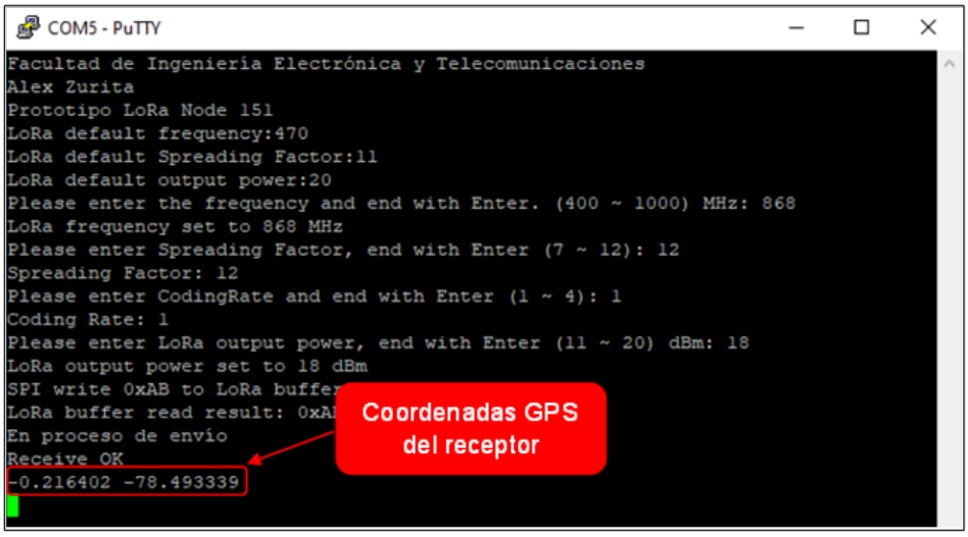
Figura 3.2. Implementación final del prototipo.

lado del receptor en un teminal PuTTy, estos datos son enviados desde la placa Arduino UNO. Estos datos son transferidos a LoRa Node 151 y este a su vez envía el paquete LoRa hacia el segundo LoRa Node 151 y se observa las coordenas GPS en el terminal PuTTY del lado del receptor, se puede constatar que las coordenadas coinciden y son receptadas sin errores. La siguiente

Figura 3.3 muestra las coordenadas en sus respectivos terminales.



(a)



(b)

Figura 3.3. Verificación de coordenadas GPS. (a) Coordenadas GPS del lado del transmisor. (b) Coordenadas GPS del lado del receptor.

Se verificó que el prototipo se encuentra funcional por lo que se procede a realizar pruebas a una distancia de 2 metros y 30 metros entre el nodo emisor y el nodo receptor, sin

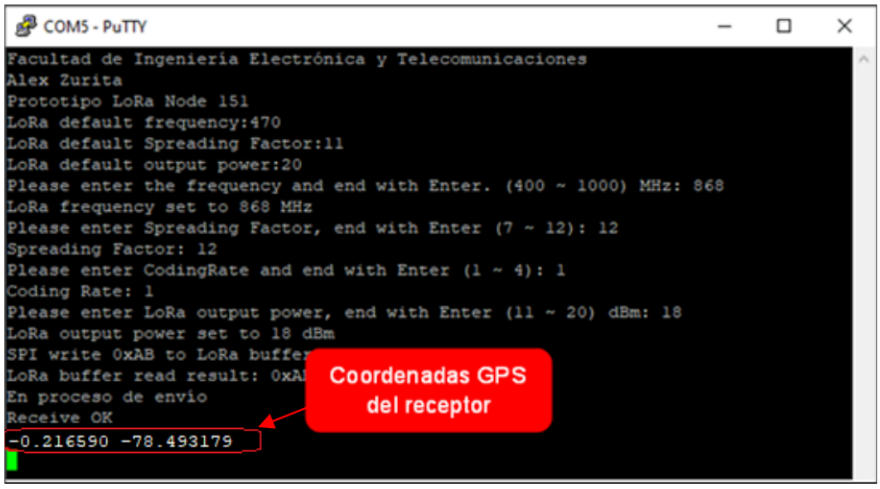
embargo se encontró un problema con la detección de movimiento puesto que la luz del sol interfería en la detección, para solucionar este problema se ajustó el potenciómetro de la sensibilidad del sensor para que se reduzca la luz de incidencia en el sensor, además se utilizó una sombrilla para limitar la luz del sol, de esta manera se logró una detección de movimiento sin problemas. En el lado del segundo nodo se comprobó la recepción de los datos sin ningún error.

La siguiente Figura 3.4 muestra la implementación inicialmente a una distancia de 2 metros.

Figura 3.4. Prueba del prototipo a una distancia de 2 metros.

Los resultados que se obtienen tanto a una distancia de 2 metros y de 30 metros son los mismos y se presentan en la siguiente figura.

(a)



(b)

Figura 3.5. Resultados en terminales. (a) Terminal transmisor. (b) Terminal receptor.

En el terminal del receptor se tienen los mismos datos enviados desde el terminal transmisor sin ningún error, por tanto, en esta instancia finaliza con éxito la prueba del prototipo.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Se diseñó un nodo prototipo que opera con GPS y detecta movimiento basado en el nodo Lora Node151.

Para la modulación LoRa de espectro expandido se puede concluir de acuerdo con las ecuaciones (1) y (2) que, para factores de dispersión grandes, la tasa de símbolo disminuye y el tiempo de transmisión aumenta.

De acuerdo con la conclusión anterior, con el fin de tener menor interferencia, los datos deberían permanecer el menor tiempo posible de transmisión, por tanto, se utiliza factores de dispersión pequeños.

Se recopiló información concreta sobre la utilización y programación del LoRa Node 151 de manera ordenada, concluyendo con pruebas exitosas de funcionamiento, tanto utilizando entradas y salidas digitales como utilizando el recurso RF para la transmisión de datos entre nodos.

El presente trabajo servirá de base y consulta para futuras investigaciones ya que se ha investigado sobre el uso y los recursos para esta tecnología nueva que utiliza LoRa Node 151.

El tiempo empleado en reunir e investigar la información necesaria para la utilización y programación del LoRa Node 151 se tardó aproximadamente 90 horas distribuidas en 3 semanas, ya que se recopiló información por separado de los dos componentes principales de esta placa, los cuales son el microcontrolador STM32L151CCU6 y el chip SX1276.

4.2. RECOMENDACIONES

Para cargar al LoRa Node 151 un programa por primera vez se recomienda entrar en modo DFU utilizando directamente un cable microUSB, ya que si se carga por primera vez utilizando el método de ST-LINK, hay la posibilidad que el controlador del ST-LINK este desactualizado por lo que no podrá cargar los programas al LoRa Node 151.

Para ahorrar tiempo en la programación, se recomienda generar el código desde la interfaz gráfica del microcontrolador, ya que esta interfaz permite configurar los GPIOs, recursos de comunicación de manera rápida e intuitiva, y simplemente seleccionando la opción Device Configuration Tool Code Generation de la barra de herramientas, se obtienen los códigos correspondientes a lo configurado en esta interfaz.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia y N. Strachan, «Evaluation of LoRa and LoRaWAN for wireless sensor networks,» de *2016 IEEE SENSORS*, 2016.

[2] L. Ntseane y B. Isong, «Analysis of lora/lorawan challenges,» de *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2019.

[3] H. Automation, «LoRa Node 151,» 2019.

[4] J. De Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic y A. L. L. Aquino, «LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities,» de *2017 2nd International Multidisciplinary Conference on Computer and* Energy *Science (SpliTech)*, 2017.

[5] P. Seneviratne, Beginning LoRa Radio Networks with Arduino: Build Long Range, Low Power Wireless IoT Networks, Apress, 2019.

[6] SEMTECH, «What is LoRa?,» 2019.

[7] STMicroelectronics, «STM32L151CC Ultra-low-power Arm Cortex-M3 MCU,» 2017.

[8] SEMTECH, «SX1216/77/78/79 Datasheet,» *Online: http://www. semtech. com/images/datasheet/sx1276. pdf,* 2015.

[9] SEMTECH, «SX1216 DevKit_UserGuide_STD_V1_1,» *Online: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HUNq/GP6b1RUcXK6FAcpZxv1EBx3hJJQn8RDDINBagqekxtw,* 2014.

[10] M. O. R. Nave, «Sistema de Posicionamiento Global GPS,» *Online: http://hyperphysics.phy-astr.gsu.edu/hbasees/gps.html,* 2018.

[11] R. J. Aughey, «Applications of GPS technologies to field sports,» *International journal of sports physiology and performance,* vol. 6, pp. 295-310, 2011.

[12] S. Jin, E. Cardellach y F. Xie, GNSS remote sensing, vol. 16, Springer, 2014.

[13] TecBolivia.com, «Módulo GPS Ublox NEO-8M GY-NEO8MV2,» *http://www.tecbolivia.com/index.php/venta-de-componentes-electronicos-11/sensores/m%C3%B3dulo-gps-ublox-neo-8m-gy-neo8mv2-detail,* 2018.

[14] Ó. T. Artero, Arduino. Curso Práctico de Formación, RC libros, 2013.

[15] S. A. Punto Flotante, «Sensor Infrarrojo de movimiento PIR HC-SR501 Datasheet,» *https://www.puntoflotante.net/MANUAL-DEL-USUARIO-SENSOR-DE-MOVIMIENTO-PIR-HC-SR501.pdf,* 2017.

[16] . Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 8 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[17] S. Vashi, J. Ram, J. Modi, S. Verma y C. Prakash, «Internet of Things (IoT): A vision, architectural elements, and security issues,» de *2017 international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, 2017.

[18] STMicroelectronics, «STM32L151xC Datasheet,» *Online: http://www. semtech. com/images/datasheet/sx1276. pdf,* 2017.

[19] Y. Song, J. Lin, M. Tang y S. Dong, «An Internet of energy things based on wireless LPWAN,» *Engineering,* vol. 3, pp. 460-466, 2017.

[20] N. C. Luong, D. T. Hoang, P. Wang, D. Niyato, D. I. Kim y Z. Han, «Data collection and wireless communication in Internet of Things (IoT) using economic analysis and pricing models: A survey,» *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 2546-2590, 2016.

[21] L. Leonardi, F. Battaglia, G. Patti y L. L. Bello, «Industrial LoRa: A novel medium access strategy for LoRa in industry 4.0 applications,» de *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, 2018.

[22] C. Hayes, «Gestión de la potencia IoT con tecnología LPWAN,» 2019.

[23] L. Chengdu Heltec Automation Technology Co., «SX1216 DevKit_UserGuide_STD_V1_1,» *Online: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R000000HUNq/GP6b1RUcXK6FAcpZxv1EBx3hJJQn8RDDINBagqekxtw,* 2014.

[24] M. Chen, U. Challita, W. Saad, C. Yin y M. Debbah, «Artificial neural networks-based machine learning for wireless networks: A tutorial,» *IEEE Communications Surveys & Tutorials,* vol. 21, pp. 3039-3071, 2019.

[25] BehrTech, «LPWAN | MYTHINGS, LoRa, NB-IoT, Sigfox | BehrTech,» 2020.

[26] E. Bäumker, A. M. Garcia y P. Woias, «Minimizing power consumption of LoRa® and LoRaWAN for low-power wireless sensor nodes,» de *Journal of Physics: Conference Series*, 2019.

ANEXOS

ANEXO A. Datasheet STM32L151CCU6

ANEXO B. Datasheet SX1276

ANEXO C. Diagrama de pines LoRa Node 151

ORDEN DE EMPASTADO