UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS

"RED DE SENSORES PARA MONITOREO DE RADIACIÓN NO IONIZANTE DE LOS SERVICIOS DE TELEFONÍA MÓVIL EN COLEGIOS Y HOSPITALES DEL DISTRITO DE LA PUNTA EN LA REGIÓN CALLAO"

PARA OBTENER EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

AUTORES:

Bach. ARRIETA CABALLERO, GERARDO ROBERTO

Bach. HUACHO ROJAS, GINA MILAGROS

Bach. RODRIGUEZ LENGUA, PAULO RENATTO

ASESOR: **MSc. Ing. JULIO CESAR BORJAS CASTAÑEDA**

Callao, 2019

PERÚ

140

ACTA para la obtención del Título profesional por la modalidad de Tesis Sin ciclo de Tesis

A los 11 días del mes de Abril del 2019 siendo las 14:00 horas se reunió el Jurado examinador de la facultad de ingeniería Eléctrica y Electrónica conformado por los siguientes docentes Ordinarios de la Universidad Nacional del Callao. (Res Dec N° 028-2014-DFIEE)

Mg. ing. Ricardo Raul Rodriguez Bustinza — Presidente
Mg. ing. Armando pedro, Cruz Ramirez — Secretario
Mg. ing. Cuzcano Rivas Abilio Bernardino — Suplente

Con el fin de dar inicio a la exposición de Tesis de los Señores Bachilleres en Ingeniería Electrónica quienes habiendo Cumplido con los requisitos establecidos en la Normativa sustentaron la Tesis Titulada: "RED DE Sensores Para Monitoreo de Radiación NO Ionizante de los Servicios de Telefonía Movil en Colegios y Hospitales del DISTRITO de la PUNTA en la Region CALLAO" Con el quorun reglamento de ley, se dio inicio a la exposición Considerando lo establecido en el Reglamento de Grados y Títulos, correspondiente al otorgamiento del Título profesional por la Modalidad de Tesis Sin ciclo de Tesis, efectuadas los deliberaciones pertinentes se acordo.

Dar por Aprobado Calificativo muy bueno nota 17. a los expositores:

Arrieta Caballero, Gerardo Roberto
Huacho Rojas, Gina Milagros
Rodriges lengua, Paulo Relatto David

Con lo cual se dio por concluida la sesión, siendo las 15:10 horas del día del mes y año en curso

Presidente
Ricardo Raul Rodrigz Bautista

Secretario
Armando pedro Cruz Ramirez

Vocal
Cuzcano Rivas Abilio

HOJA DE REFERENCIA DEL JURADO Y APROBACIÓN

PRESIDENTE : Mg. Ing. RICARDO RAUL RODRIGUEZ BUSTINZA

SECRETARIO : MSc. Ing. ARMANDO PEDRO CRUZ RAMIREZ

VOCAL : MSc. Ing. ABILIO BERNARDINO CUZCANO RIVAS

ASESOR : MSc. Ing. JULIO CESAR BORJAS CASTAÑEDA

DEDICATORIA

La presente investigación está dedicada a nuestros familiares, docentes y amigos que nos han brindado el soporte necesario a lo largo de nuestra formación universitaria.

AGRADECIMIENTO

Agradecemos a la Universidad Nacional del Callao por darnos la oportunidad de desarrollarnos profesionalmente y a nuestros docentes de la Facultad de Ingeniería Eléctrica y Electrónica por los conocimientos compartidos.

INDICE

TABLA DE CONTENIDOS

1. CONTENIDO DE FIGURAS

2. CONTENIDO DE CUADROS

3. CONTENIDO DE TABLAS

RESUMEN

El presente informe de tesis ““RED DE SENSORES PARA MONITOREO DE RADIACIÓN NO IONIZANTE DE LOS SERVICIOS DE TELEFONÍA MÓVIL EN COLEGIOS Y HOSPITALES DEL DISTRITO DE LA PUNTA EN LA REGIÓN CALLAO”, está orientado al área de supervisión de los servicios públicos en telecomunicaciones, esencialmente a la telefonía móvil.

La implementación de las estaciones radioeléctricas de telefonía móvil ha tenido un impacto negativo en la población, ya sea por su robustez o por los mitos que se dicen acerca de que las antenas provocan efectos en la salud.

Para la realidad peruana, el Ministerio de Transportes y Comunicaciones, en el marco de su función y competencia, ha establecido los límites máximos permisibles de las RNI, en la exposición ocupacional, poblacional y de población en áreas de uso público, promulgándose el Decreto Supremo N°038-2003-MTC, el mismo que fuera modificado por el Decreto Supremo N°038-2006-MTC.

Por lo que como parte del rol de control y fiscalizador es importante determinar el procedimiento a implementar considerando que en las experiencias de mediciones en Perú y a nivel internacional se observa que los niveles emitidos por las

estaciones base celular, son muy pequeños como para producir riesgos significativos en la salud, pero las personas al desconocer sobre el tema se oponen a la implementación de más estaciones de telecomunicaciones lo que trae como consecuencia la falta de cobertura del servicio, para ello la presente investigación busca brindar esta información mediante aplicativos multiplataformas bajo la supervisión del ente competente, en este caso el Ministerio de Transportes y Comunicaciones.

ABSTRACT

The present thesis project " SENSORS NETWORK FOR NON-IONIZING RADIATION MONITORING OF MOBILE TELEPHONE SERVICES IN COLLEGES AND HOSPITALS OF THE DISTRICT OF LA PUNTA IN THE CALLAO REGION ", is oriented to the area of supervision of public services in telecommunications, essentially to mobile telephony.

The implementation of mobile radio stations has had a negative impact on the population, either because of its robustness or because of the myths that are told about the effect of antennas on health.

For the Peruvian reality, the Ministry of Transport and Communications, within the framework of its function and competence, has established the maximum permissible limits of the RNI, in the occupational, population and population exposure in areas of public use, promulgating the Supreme Decree No. 038-2003-MTC, the same as that modified by Supreme Decree No. 038-2006-MTC.

Therefore, as part of the control and oversight role, it is important to determine the procedure to be implemented considering that in the measurement experiences in Peru and internationally it is observed that the levels emitted by the cellular base stations are too small to produce significant risks in health, but people who do not know about the issue are opposed to the implementation of more

telecommunication stations, which results in the lack of service coverage, for which the present investigation seeks to provide this information through multiplatform applications under the supervision of the competent authority, in this case the Ministry of Transport and Communications.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1. Identificación del problema

La perspectiva de la telefonía móvil es a seguir creciendo, no solo con el aumento de usuarios sino también al incremento de servicios, lo anterior mencionado va acompañado de la evolución constante de las tecnologías utilizadas en las redes de telecomunicaciones, como consecuencia es necesario el incremento de las estaciones radioeléctricas ya sea para mejorar la calidad y cobertura o para migrar a nuevas tecnologías.

Según la normativa en el Perú, el decreto supremo N°038-2003-MTC, publicado el 06 de junio del 2003 en el Diario Oficial El Peruano, ha establecido los Límites Máximos Permisibles de Radiaciones No Ionizantes en Telecomunicaciones; asimismo en la Resolución Ministerial N°120-2005-MTC se nombran algunas normas técnicas sobre las restricciones radioeléctricas en áreas de uso público, lugares definidos por la administración, en los que se considera que la población expuesta podría ser sensible a los campos electromagnéticos, como los centros educativos y centros de salud; por ello las Radiaciones No Ionizantes deben ser tan bajo como sea técnicamente posible.

Para que un operador de telefonía móvil instale una estación de radiocomunicación necesita presentar al Ministerio de Transportes y

Comunicaciones los requisitos establecidos en los artículos 12, 13, 14 y 15 del D.S. 003-2015-MTC que aprueba el Reglamento de la Ley N°29022 - Ley para el Fortalecimiento de la Expansión de Infraestructura en Telecomunicaciones; que conforme a lo dispuesto en el literal g) del artículo 15 requiere una “Carta de compromiso del Operador o del Proveedor de Infraestructura Pasiva, por la cual se compromete a adoptar las medidas necesarias para revertir y/o mitigar el ruido, las vibraciones u otro impacto ambiental durante la instalación de la Infraestructura de Telecomunicaciones, así como a cumplir los Límites Máximos Permisibles.”, esto quiere decir que solamente requiere un estudio teórico de los Límites Máximos Permisibles, mas no exige una medición de campo real en ninguna etapa de la implementación de la torre. Por lo antes expuesto, es como nace la necesidad de una constante monitorización de los niveles de Radiación No Ionizante comparados con la normativa nacional.

Cuando la implementación la estación radioeléctrica ha culminado y ya está operando, se debe realizar mediciones de los niveles del campo eléctrico, para poder determinar el aporte de esta nueva estación, con el objetivo de verificar el cumplimiento de los límites máximos permisibles que el Ministerio de Transportes y Comunicaciones exige, antes y después de la instalación.

2. Formulación del problema

2.1 Problema General

¿Es posible que una red de sensores permite monitorear la radiación no

ionizante de los servicios de telefonía móvil en colegios y hospitales del distrito de La Punta en la Región Callao?

2.2 Problemas Específicos

P.E.1 ¿Qué componentes son necesarios para realizar las medición, procesamiento y transmisión de los niveles de radiación no ionizante en el distrito de La Punta?

P.E.2 ¿Cuál es la manera más óptima de transmitir las mediciones de radiación no ionizante en la red de sensores?

P.E.3 ¿Dónde se deben ubicar los sensores en el distrito de La Punta para obtener la medición de los niveles de radiación no ionizante en las áreas de uso público?

P.E.4 ¿Cómo se puede visualizar los resultados de las mediciones obtenidas por la red de sensores en el distrito de la Punta?

3. Objetivos de la investigación

3.1 Objetivo General

Diseñar una red de sensores para el monitoreo de radiación no ionizante de los servicios de telefonía móvil en colegios y hospitales del distrito de La Punta en la región Callao.

3.2 Objetivos Específicos

O.E.1 Diseñar el hardware y software de los nodos que componen la red de

sensores.

O.E.2 Diseñar la topología usada en la red de sensores.

O.E.3 Planificar la distribución de nodos en el distrito de La Punta.

O.E.4 Programar la interfaz de visualización de las mediciones de radiación no ionizante.

4. Justificación

La radiación es una forma de energía que está presente de forma natural o artificial, y de alguna manera, todos estamos expuestos algún tipo de radiación, principalmente a la radiación solar y otras ondas electromagnéticas.

Los sistemas eléctricos, radio, televisión y telecomunicaciones en general, son emisores de radiaciones electromagnéticas, que afectan al entorno en el que existen.

En este contexto, las Radiaciones No Ionizantes que es aquella onda o partícula que no es capaz de arrancar electrones (ionizar) que ilumina produciendo como mucho, excitaciones electrónicas.

Para el contexto nacional, el Ministerio de Transportes y Comunicaciones, en el marco de su función y competencia, ha establecido como los niveles máximos permitidos de las RNI; En la exposición ocupacional, poblacional y de población en áreas de uso público, desde los 9 KHz hasta los 300 GHZ,

promulgándose el Decreto Supremo N°038-2003-MTC, el mismo que fuera modificado por el Decreto Supremo N°038-2006-MTC y en la Resolución Ministerial N°120-2005-MTC/03

En la presente investigación con la red de sensores se monitoreará la radiación no ionizante que emiten las estaciones base celular de telefonía móvil en áreas de uso público, siguiendo los protocolos establecidos por el MTC, estos resultados se podrán visualizar en tiempo real en una base de datos.

5. Importancia

Esta investigación es de suma importancia para mejorar el mecanismo de monitoreo de los niveles de radiación no ionizantes emitidos por los servicios de telefonía móvil, e informar a la población sobre dichos niveles, a través de la interfaces de visualización multiplataforma, de manera que se puedan evitar los rumores asociados a la radiación emitida por las estaciones radioeléctricas

CAPÍTULO II

MARCO TEORICO

1. Antecedentes del estudio

- Tesis 01

  Título: “Análisis de Mediciones de Radiaciones No Ionizantes en ambientes interiores y exteriores en predios de la ESPOL”

  Escuela Superior Politécnica del Litoral

  Facultad de Ingeniería en Electricidad y Computación

  Guayaquil- Ecuador

  Autor: Wilson Alejandro Díaz García

  Felipe Walkir Proaño Salvatierra

  Año: 2010

  Conclusiones: Del conjunto de análisis realizados, todas las mediciones de Campo Eléctrico promedio cumplen en todos los casos con los Niveles de Referencia ICNIRP pues la relación máxima obtenida no sobrepasa el 10% para mediciones de WLAN y 6% para mediciones de telefonía móvil tal forma se demuestra que no hay riesgo alguno para las personas que transiten en la zona poblacional, cumpliendo en todos los sentidos las recomendaciones internacionales y estatales para la exposición humana a radiofrecuencias

- Tesis 02

  Título: “Software Aplicativo para el Análisis Predictivo del Comportamiento de los Niveles de Campo Eléctrico y la Distribución de Potencia producida por las Estaciones de Telefonía Móvil”

  Universidad Católica de Santa María

  Facultad de Ciencias e Ingeniería Físicas y Formales

  Programa Profesional de Ingeniería Electrónica

  Arequipa, Perú

  Autor: Br. Gallegos Paz, Arturo Fernando

  Año: 2009

  Conclusiones: En la cuarta parte, se ha expuesto el paquete software que se ha programado para la realización de los cálculos necesarios en la evaluación de una nueva estación base. Con este paquete software, programado en Matlab se consiguen todos los datos necesarios y así cumplir con la normativa que exige la realización de un estudio teórico de radiaciones no ionizantes, solo introduciendo los datos de las tecnologías existentes y datos recogido al realizar las medidas radio eléctricas. Los datos que se obtienen son predicciones de los niveles de campo eléctrico, densidad de potencia, cociente de exposición poblacional, diagramas de radiación de las antenas que se instalan en los emplazamientos, distancias mínimas de seguridad, y graficas representativas de las distribuciones de potencia en 2D y 3D. De las simulaciones realizadas, en

ninguno de los casos se ha excedido el 1% de los LMP, lo que demuestra que el aporte de radiación electromagnética es considerablemente bajo.

- Tesis 03

  Título: “Predicción del Espectro de Emisiones Electromagnéticas desde Tarjetas de Circuito Impreso para Sistemas Digitales”

  Instituto Politécnico Nacional

  Escuela Superior de Ingeniería Mecánica y Eléctrica

  Unidad profesional “Adolfo López Mateos”

  Sección de Estudios de posgrado e Investigación

  México

  Autor: Rodrigo Jiménez López

  Año: 2002

  Conclusiones: La utilización de esta metodología en la predicción de emisiones radiadas por tarjetas de circuitos impresos nos ayuda de forma económica y eficiente, a prevenir problemas de interferencias electromagnéticas en las etapas de diseño de una forma confiable, sin la necesidad de utilizar programas específicos de simulación, los cuales son costosos y complicados y en la mayoría de los casos de uso específico.

- Tesis 04

  Título “Estudio real de las radiaciones no ionizantes en la provincia de Trujillo en las bandas de HF y VHF”

  Universidad Privada Antenor Orrego

  Facultad de Ingeniería Electrónica

Línea de Investigación: Radiodifusión y Comunicaciones inalámbricas

Trujillo- Perú

Autor: Br. César Obed Zavaleta Castro

Br. Frank Carlos Peralta Lujan

Año: 2016

Conclusiones: Se comprobó que el Campo Eléctrico en las en las cercanías de las radioemisoras en AM, FM y TV en estudio están bajo los límites permisibles dados por el MTC.

2. Fundamento Epistemológico

Epistemológicamente, la relación entre la medición y el resultado visualizado; es decir entre variables dependientes y las independientes se determinan por medio de un algoritmo de programación.

Para el caso de la red de sensores de radiación no ionizante, se realizará por medio de una cantidad de líneas de programación que transforman la señal obtenida por la medición de potencia en el microcontrolador para ser analizadas y luego visualizadas en una aplicación.

3. Fundamento Ontológico

El fundamento ontológico del marco teórico se describe de la siguiente manera: la red de sensores de radiación no ionizante consiste en la adquisición de datos mediante mediciones periódicas. El objetivo está en la

visualización de los datos previamente analizados y comparados con la normativa nacional a través de un aplicativo multiplataforma.

4. Fundamento Metodológico

La metodología a seguir para el diseño de una red de sensores para el monitoreo de radiación no ionizante de los servicios de telefonía móvil en colegios y hospitales del distrito de La Punta, se fundamenta en los siguientes pasos:

- Etapa de diseño del hardware y software de los nodos de la red.
- Etapa de ubicación de los puntos de medición.
- Etapa de selección de topología y protocolos de comunicación para la red de sensores.
- Etapa del desarrollo de la aplicación de visualización para la información adquirida.

5. Definiciones de términos básicos

- FDMA

Divide la banda de frecuencia en subcanales y asigna uno de estos subcanales de frecuencias a enlace de comunicación.

- TDMA

Divide un subcanal de frecuencia en diferentes tramas temporales y a su vez cada trama se subdivide en una secuencia de ranuras temporales. A

cada comunicación se le asigna una ranura dentro de cada trama (siempre la primera, o la segunda o la que corresponda, pero siempre la misma) siguiendo el principio de multiplexado en el tiempo.

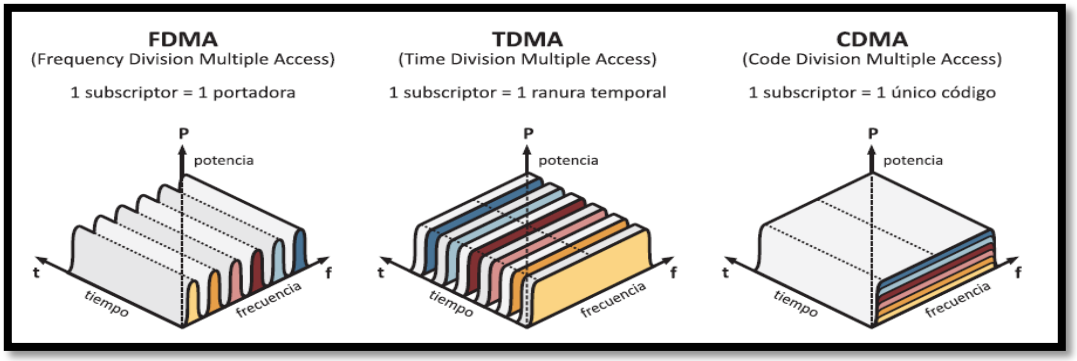
- CDMA

Procesa una secuencia de datos en cada extremo del canal de transmisión mediante el proceso de codificación a través de un esquema de codificación único, que es deliberadamente diferente de los esquemas codificación de todos los otros canales. Así, las señales de cada canal se pueden transmitir usando la misma banda de frecuencia y de forma simultánea. Los canales de comunicación se distinguen unos de otros en recepción a partir del esquema de codificación única utilizado para la codificación en el transmisor.

- OFDMA

Se basa en una combinación de acceso múltiple por división en tiempo y frecuencia. Los recursos disponibles se dividen en subportadoras en un dominio de frecuencia y también en varios intervalos de tiempo en el dominio de tiempo. Así pues, a los usuarios individuales se les asignado no sólo una o varias subportadoras sino también un intervalo de tiempo para la comunicación.

FIGURA Nº2. 1
DEFINICIÓN GRÁFICA DE FDMA, TDMA Y CDMA



- Celda

  Realmente, el elemento que nos da cobertura es la celda. Cada una de las antenas de un emplazamiento cubre un sector circular denominado celda. Además, si en el mismo sector circular tenemos varias tecnologías (2G, 3G, LTE), cada una es una celda distinta, aunque coincidan en el espacio. Es el mismo caso si tenemos la misma tecnología en dos bandas distintas serían dos celdas diferentes. Por ejemplo, 2G en la banda de 900 MHz y en la banda de 1800 MHz. Serían celdas distintas incluso si tienen la misma antena física. Un teléfono o dispositivo móvil solo están conectado a una celda, aunque mantiene información de todas las celdas próximas por si pierde la cobertura y tiene que conectarse a otra.

- Cobertura

  En realidad, la cobertura que da una celda está más limitada por el teléfono que por la antena de dicha celda. La potencia de una celda puede llegar

hoy en día hasta los 100 W y está en un lugar elevado por lo que puede llegar muy lejos. Sin embargo, el teléfono emite con 1 o 2 W dependiendo de la banda y suele estar en un bolsillo o en un bolso lo que limita mucho su capacidad de llegar hasta la antena de la celda. Por lo tanto, lo mayores problemas en la comunicación se producen desde el teléfono a la antena del operador. Por lo tanto, cualquier cosa que mejore la comunicación en el teléfono mejora la cobertura. Por ejemplo, utilizar un auricular que nos permite hablar y colocar el teléfono separado de nuestro cuerpo y en un sitio fijo. Hay teléfonos que permiten una antena externa que, colocada sobre el techo de un coche, aumenta mucho la capacidad de transmisión de nuestro teléfono. Otro punto a tener en cuenta es que las bandas de 700 MHz a 900 MHz permiten transmitir a 2 W mientras que el mismo teléfono en otras bandas solo puede transmitir con 1 W de potencia. Por lo tanto, se tendrá mejor cobertura en las bandas de 700 MHz a 900 MHz que en otras bandas de frecuencia superior.

- Banda

Se denomina banda al rango de frecuencias asignado, en este caso, para la telefonía móvil. Los gobiernos de cada país asignan en régimen de concesión por un tiempo a varias empresas el uso de esa banda. A cada empresa se le asigna un parte fija de esa banda y nadie más puede utilizarla. Normalmente la banda se identifica con la frecuencia central aunque realmente es un rango de frecuencias. En el caso de la banda de 900 MHz el rango, dependiendo del país, va desde 890 MHz a 915 MHz.

Inicialmente el sistema GSM comenzó en la banda de 900 MHz y en la banda de 1900 MHz en Estados Unidos (La banda de 900 MHz estaba ocupada). Posteriormente se utilizó la banda de 1800 MHz para dar más capacidad al sistema GSM. UMTS comenzó en la banda de 2100 MHz y actualmente GSM está dejando libre parte de la banda de 900 MHz en beneficio de UMTS. Esto está mejorando en gran medida la cobertura de UMTS ya que, como hemos indicado en el punto anterior, la banda de 900 MHz es la que mejor cobertura da. También ahora llega LTE y necesita también su espacio. LTE está utilizando la nueva banda de 800 MHz (que deja libre la banda de televisión), la de 1800 MHz (menos espacio para GSM y la banda de 2600 MHz (Ocupando hasta ahora por otras tecnologías como WIMAX o punto a multipunto). Una banda peculiar bastante utilizada en América es la banda AWS (Advanced Wireless Service). Utiliza la banda de 1700 MHz para la comunicación del teléfono a la antena y la banda de 2100 MHz para la comunicación de la antena al teléfono.

Por regla general cualquier tecnología funciona en cualquier banda y son los gobiernos de los países los que realizan esta asignación. Los operadores procuran emitir en la frecuencia más baja posible ya que la cobertura es mayor y eso le permite poner menos antenas. Pero esto no siempre es posible ya que la banda puede estar ocupada por otras tecnologías más antiguas o directamente no ha podido comprar la licencia necesaria. Esto se está notando sobre todo en la tecnología LTE que es la última que ha llegado y está ocupando los huecos que quedan libres.

Según los países hay una gran disparidad de bandas utilizadas. Debemos comprobar tanto la tecnología como la banda en la que se emite a la hora de comprar un teléfono o cuando viajamos ya que los teléfonos solo soportan cada tecnología en unas bandas concretas. Para más información consultar en la página cobertura.

- Radiación Electromagnética:

  La radiación electromagnética es una combinación de campos eléctricos y magnéticos oscilantes que se propagan a través del espacio transportando energía de un lugar a otro. A diferencia de otros tipos de onda, como el sonido, que necesitan un medio material para propagarse, la radiación electromagnética se puede propagar en el vacío. Los rayos X, las ondas de radio, los rayos gamma, los rayos infrarrojos y la luz visible son los tipos más importantes de radiación electromagnética. Las radiaciones ordenadas de acuerdo con su longitud de onda (λ) conforman el espectro electromagnético.

  La longitud de onda de una onda electromagnética (λ) está relacionada con su frecuencia (f) y su velocidad de propagación (v)

  según:

$$v=f*\lambda$$

  En el espacio libre, v es igual a la velocidad de la luz (aproximadamente 3.10^8m/s).

  El ser humano se encuentra expuesto principalmente a las radiaciones tanto de alta como de baja frecuencia. En el primer caso son las

provenientes de las antenas de telefonía celular y de las emisoras de radiodifusión (AM, FM y TV), mientras que el segundo corresponde a la red eléctrica domiciliaria. En este artículo se analizan estos casos particulares.

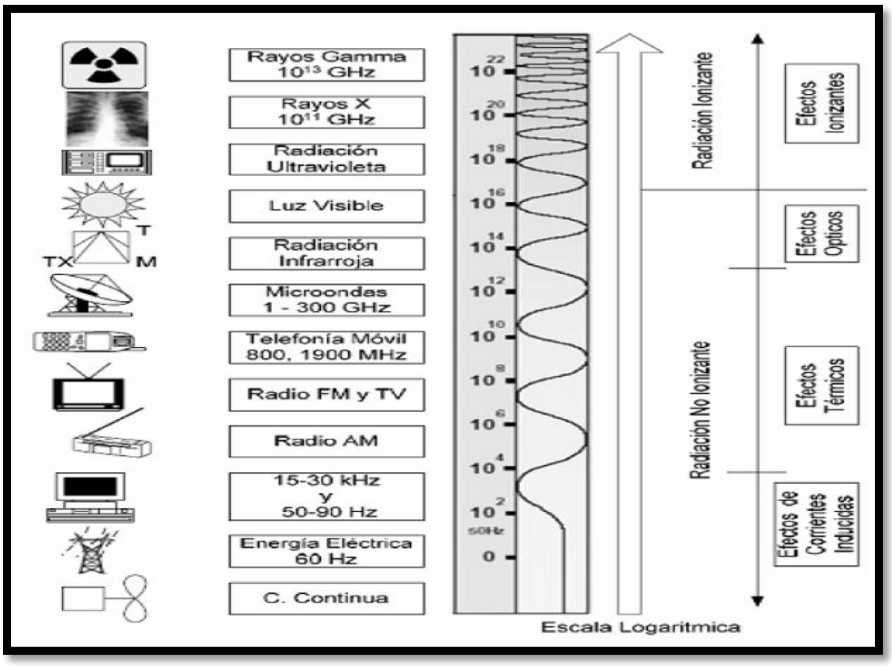
- Clasificación de Radiación Electromagnética:

  La ionización es el proceso por el cual se producen iones, que son átomos o moléculas cargadas eléctricamente debido al exceso o falta de electrones. Si la energía transportada por la onda electromagnética generada es suficiente para producir un ion, la radiación resultante se denomina ionizante.

  - Radiación Ionizante: Las radiaciones ionizantes corresponden a la porción de alta frecuencia del espectro electromagnético. Es bien conocido el uso medicinal de los rayos X y otros tipos de radiaciones de alta energía y también los efectos dañinos en exposiciones no controladas. Cabe señalar que es muy difícil encontrarse expuesto a este tipo de radiaciones, dado que normalmente estas fuentes se encuentran confinadas y su uso está estrictamente reglamentado y controlado.
  - Radiación No Ionizante: La zona del espectro electromagnético de menor frecuencia se la denomina región de Radiaciones No Ionizantes (RNI). A esta región pertenecen las emisiones de todo tipo de energía electromagnética como las utilizadas en comunicaciones inalámbricas, estaciones de radio de AM, FM y TV, radar, telefonía

celular, microondas, sistemas de láser y radiaciones infrarrojas. Estas son las radiaciones a las que habitualmente estamos expuestos involuntariamente porque forman parte de las condiciones ambientales en las que se desarrolla nuestra vida. Usualmente estamos irradiados por vivir o transitar en las inmediaciones de los mástiles donde están montadas las antenas transmisoras.

FIGURA Nº2. 2
CLASIFICACIÓN POR TIPO DE RADIACIÓN



- Redes Inalámbricas de Sensores:

Las redes inalámbricas de sensores es una red compuesta por elementos capaces de sensar, procesar y transmitir información, lo cual le permite a un usuario observar, medir y reaccionar a eventos detectados en su entorno.

Aplicaciones típicas de estos sistemas incluyen, recolección de datos, monitoreo, vigilancia, telemetría, etc.

Los componentes necesarios para la operación de una red de sensores son:

- Los nodos de la red capaces de sensar el ambiente a su alrededor.
- La interconexión entre los nodos para transmitir la información adquirida.
- Un nodo sumidero para recolectar y procesar la información.

Debido a que la gran cantidad de información que se puede obtener de la red de sensores, es necesario considerar métodos de administración y compresión de datos en su diseño. La infraestructura de procesamiento y comunicación que se usa en estos sistemas es específico a su aplicación, pero existen aspectos clave que deben ser considerados en el diseño de cualquier red de sensores, tales como consumo de potencia de los nodos, capacidad de baterías, etc. Uno de los retos del diseño de las redes de sensores es lograr la comunicación con baja potencia y el procesamiento con bajo costo. Otro reto importante es lograr extender el tiempo de vida de la red usando una fuente de energía limitada, como baterías.

Las redes de radares usados para control de tráfico aéreo, las estaciones de monitoreo del clima son ejemplos de redes de sensores, estos usan equipos especializados por lo que son muy costosos, sin embargo, es posible desplegar una red de sensores de bajo costo para otras aplicaciones como seguridad, salud, comercio, etc.

El estudio de redes de sensores es un campo multidisciplinario que involucra conocimientos en radio propagación, networking, procesamiento de señales, programación, base de datos, optimización de recursos, control de potencia.

Los diferentes tipos de sensores utilizados pueden ser detectores de campo eléctrico, magnético, frecuencia, sensores electro-ópticos, infrarrojos, radares, sensores de localización o navegación, sensores sísmicos, detectores de ondas de presión, sensores de parámetros ambientales como temperatura, humedad, etc. Estos sensores pueden ser pasivos o energizados por la batería dependiendo de la aplicación.

Los nodos que usan estos sensores están interconectados a través de una serie de enlaces inalámbricos de baja potencia y en distancias pequeñas. Estos enlaces usan técnicas de comunicación orientados a la conexión y que se pueden encontrar en los estándares IEEE 802.

Los nodos de estos sistemas son distribuidos de una manera muy densa para garantizar la conectividad entre ellos. Estos nodos son enlazados de manera lógica a través de una serie de protocolos de organización.

Las redes de sensores no cooperativas, usan una conexión directa entre los nodos fuente (source) y el nodo sumidero (sink) usan un enlace punto-a-punto. Estas redes no soportan comunicación entre nodos fuente, el rango de la red puede ser de cientos de metros y usan ruteo estático para comunicarse.

Las redes de sensores cooperativas, usan una conexión con múltiples saltos entre los nodos usan técnicas de ruteo dinámico, los nodos fuente pueden comunicarse entre sí y el rango de la red puede ser de kilómetros.

El bajo consumo de energía de los nodos es un factor clave para asegurar el funcionamiento prolongado de la red. Para lograr la eficiencia energética se usan algunas de estas técnicas:

- Funcionamiento con un bajo ciclo de trabajo.
- Procesamiento de la data adquirida para reducir el volumen de data transmitida y así reducir el tiempo de transmisión.
- Comunicación a través de saltos múltiples para evitar la necesidad de transmitir datos a largas distancias. Cada nodo en la red puede funcionar como repetidor, de manera que la distancia del enlace entre nodos se hace más pequeña y así se reduce la potencia de transmisión.

A inicios de los años 2000, los fabricantes buscaban introducir un estándar para comunicar los sensores en redes inalámbricas.

Para aplicaciones en ambientes cerrados, se descartó el uso de Wi-Fi (Estándar IEEE 802.11) pues es un protocolo muy complejo y usa un ancho de banda mucho mayor al que se necesita. Se descartó la comunicación por dispositivos infrarrojos pues no siempre se puede contar con línea de vista. Bluetooth (Estándar IEEE 802.15.1) fue descartada pues se considera muy costosa y limitada. Finalmente surgió el estándar IEEE 802.15.4, que describe los protocolos de comunicación para la capa física y capa de enlace y opera en la banda ISM de 2.4GHz, y este estaba diseñado para complementar las tecnologías de Wi-Fi y Bluetooth brindando conectividad punto-a-punto con bajo consumo de potencia y baja tasa de transmisión.

Luego, la aparición del estándar ZigBee, que es una capa de software que opera sobre las capas ya definidas en el estándar IEEE 802.15.4, facilita el desarrollo de aplicaciones ofreciendo tasas de transmisión de hasta 250kbps y rangos de hasta 80 metros.

CAPÍTULO III

VARIABLES E HIPÓTESIS

1. Variables de la investigación

1.1 Variables independientes

Nivel de radiación no ionizante emitida por las estaciones radioeléctricas de servicios de telecomunicaciones.

1.2 Variables dependientes

- Diseño de nodos sensores en la red.
- Topología de la red de sensores.
- Distribución geográfica de los nodos de la red.
- Interfaz web para visualizar los datos procesados.

2. Operacionalización de variables

CUADRO Nº 1
OPERACIONALIZACIÓN DE VARIABLES

Variables	Tipo de Variable	Indicadores
Nivel de radiación no ionizante emitida por las estaciones radioeléctricas de servicios de telecomunicaciones	Variable Independiente	Potencia radiada por antenas.
		Intensidad de campo eléctrico en las áreas públicas del distrito de La Punta.
		Densidad de Potencia en las áreas públicas del distrito de La Punta.
Nodos de la red de	Variable	Dimensiones físicas.

sensores	Dependiente	Capacidad de batería.
		Frecuencia de operación.
		Consumo de potencia.
		Tiempo de vida.
Topología de la red de sensores	Variable Dependiente	Tamaño de la red.
		Técnicas de transmisión y recepción.
		Tasa de transmisión de datos.
		Seguridad de la red.
Distribución geográfica de los nodos de la red	Variable Dependiente	Coordenadas.
		Altura.
Interfaz web	Variable Dependiente	Lenguaje de programación.
		Apariencia.
		Disponibilidad.

3. Hipótesis general e hipótesis específicos

3.1 Hipótesis General

El diseño de una red de sensores de radiación no ionizante emitida por los servicios de telecomunicaciones permitirá el monitoreo en tiempo real de los niveles de intensidad de campo en el distrito de la Punta en la región Callao que pueden ser comparados con los límites máximos permisibles establecidos en el Decreto Supremo N°38-2003-MTC y Resolución Ministerial N°120-2005-MTC para garantizar su cumplimiento, sin tener que depender únicamente en estudios teóricos.

3.2 Hipótesis Específicos

H.E.1 El hardware y software que conforman los nodos de la red de sensores será necesario para la adquisición, procesamiento y transmisión de las mediciones de los niveles de radiación no ionizante en el distrito de La Punta.

H.E.2 La topología de red del sistema optimizará la transmisión y recepción de datos correspondientes a la medición de los nivele s de radiación no ionizante en el distrito de La Punta.

H.E.3 La distribución adecuada de los nodos en las áreas de uso público en el distrito de La Punta permitirá la medición de los niveles de radiación no ionizante.

H.E.4 La interfaz web para visualizar las mediciones de la red de sensores facilitará el monitoreo en tiempo real de los niveles de radiación no ionizante en el distrito de La Punta.

CAPÍTULO IV

METODOLOGIA

1. Tipo de investigación

La investigación que se realizó para hacer este estudio se determina como un proyecto factible en el cual se demostrará los beneficios de este tipo de tecnología, este estudio se fundamenta en una investigación aplicada.

2. Diseño de la Investigación

Para el diseño de una red de sensores para el monitoreo de radiación no ionizante emitida por las estaciones radioeléctricas de telefonía, tiene el siguiente procedimiento.

- Recolección de data: La potencia emitida de las estaciones radioeléctricas de telefonía en los centros educativos y de salud del distrito de La Punta.
- Análisis de la data: Mediante algoritmos se transforma la potencia captada a campo eléctrico.
- Comparación de la data: del campo eléctrico se obtiene el porcentaje de radiación no ionizante que es emitida, para luego ser comparada con los Límites Máximos Permisibles relativas a la frecuencia de operación de cada operador, tal como se muestra en las siguientes tablas.
- Servicios Públicos de Telecomunicaciones móviles en Perú:

TABLA Nº4. 1
BANDA 698-806 MHz

Bloque	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida	Retorno		
A	703-718	758-773	Entel Perú S.A.C.	A nivel Nacional
B	718-733	733-788	América Móvil Perú S.A.C.	A nivel Nacional
C	733-748	788-803	Telefónica del Perú S.A.A.	A nivel Nacional

TABLA Nº4. 2
BANDAS DE 824 – 849 MHz Y 869 – 894 MHz

Bloque	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida	Retorno		
A	824-835	869-880	Telefónica del Perú S.A.A.	A nivel Nacional
	845-846.5	890-891.5		
B	835-845	880-890	América Móvil Perú S.A.C.	A nivel Nacional
	846.5-849	891.5-894		

TABLA Nº4. 3
BANDAS DE 894 – 899 MHz Y 939 – 944 MHz

Canal	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida BW:5MHz	Retorno BW:5MHz		
1	894-899	939-944	Telefónica del Perú S.A.A.	Las Provincias: Lima y Callao

TABLA Nº4. 4
BANDAS 894 – 902 MHz Y 939 – 947 MHz

Canal	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida BW:4MHz	Retorno BW:4MHz		
1	894-898	939-943	Telefónica del Perú S.A.A.	A nivel Nacional excepto Las Provincias de Lima y Callao
2	898-902	943-947	Telefónica del Perú S.A.A.	

TABLA Nº4. 5
BANDAS DE 899 – 915 MHz Y 944-960 MHz

Canal	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida BW:16MHz	Retorno BW:16MHz		
1	899-915	944-960	Viettel Perú S.A.C.	Las Provincias: Lima y Callao

TABLA Nº4. 6
BANDAS DE 902 – 915 MHz Y 947 – 960 MHz

Canal	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida BW:13MHz	Retorno BW:13MHz		
1	902-915	947-960	Viettel Perú S.A.C.	Las Provincias: Lima y Callao

TABLA Nº4. 7
BANDAS 1710 – 1771 MHz Y 2110 – 2170 MHz

Bloque	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida	Retorno		
A	1710-1730	2110-2130	Telefónica del Perú S.A.A.	A nivel Nacional

B	1730-1750	2130-2150	Entel Perú S.A.C.	A nivel Nacional
C	1750-1770	2150-2170		A nivel Nacional

TABLA Nº4. 8
BANDAS 1850 – 1910 MHz Y 1930 – 1990 MHz

Bloque	Rango de Frecuencias (MHz)		Empresa	Área de Asignación
	Ida	Retorno		
A	1850-1865	1930-1945	América Móvil Perú S.A.C.	A nivel Nacional
D	1865-1870	1945-1950	Entel Perú S.A.C.	A nivel Nacional
B	1870-1882.5	1950-1962.5	Telefónica del Perú S.A.A.	A nivel Nacional
E	1882.5-1895	1962.5-1975	Entel Perú S.A.C.	A nivel Nacional
F	1895-1897.5	1975-1977.5	América Móvil Perú S.A.C.	A nivel Nacional
C	1897.5-1910	1977.5-1990	Viettel Perú S.A.C.	A nivel Nacional

2. 1 Diseño del sistema

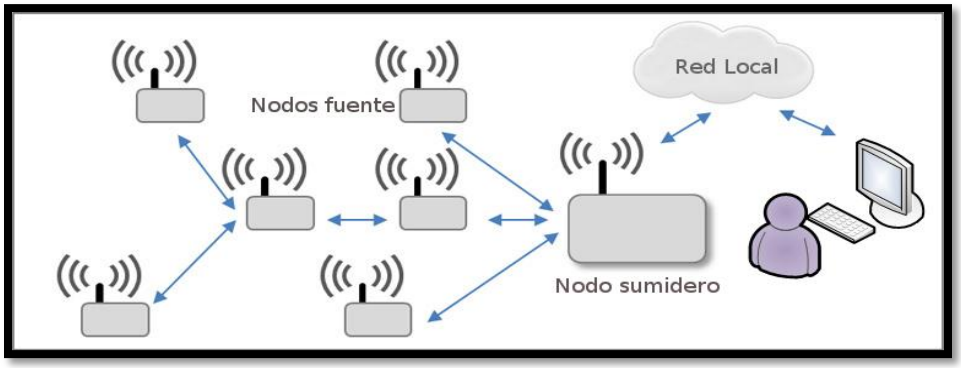
Una red inalámbrica de sensores, conocida en inglés como Wireless Sensor Network (WSN) es una red de nodos autónomos que incluyen sensores y actuadores para monitorear e interactuar con condiciones físicas y ambientales de su entorno.

Los nodos pueden ser implementados con dispositivos de bajo costo y estos pueden operar sin supervisión y mantenimiento por largos periodos de tiempo maximizando la duración de su batería, haciendo uso de técnicas de ahorro de energía tales como operación intermitente, operación en modo suspendido, hibernación profunda, transmisión por interrupciones, etc.

Los protocolos de comunicación usados en las WSN permiten que los nodos se organicen de manera autónoma, estos pueden ser agregados o retirados de la red sin afectar el funcionamiento de otros dispositivos. Esto permite que la red se mantenga operativa aun cuando ocurran fallas en los nodos, debido a que se encuentran fuera del rango de transmisión, modos de ahorro de energía o con batería baja.

Para el diseño de la red de sensores en la presente investigación, se ha considerado que los nodos que la componen, deben ser diseñados con dispositivos de bajo costo, fácil acceso y con documentación disponible.

FIGURA Nº4. 1
ESTRUCTURA DE UNA RED INALÁMBRICA DE SENSORES



Los nodos, independientes de su función, serán diseñados usando componentes similares, de manera que estos serán compatibles entre sí, y podrán ejecutar las mismas funciones si poseen el mismo hardware y software. También se ha tomado en cuenta que el hardware y software de los nodos deben soportar el protocolo de comunicación elegido.

Una vez determinados estos dos factores, se procederá a diseñar la distribución de los elementos de la red en el área de interés, en nuestro caso, el distrito de La Punta. Para esta etapa, se debe tomar en cuenta las limitaciones del diseño de los nodos, tales como el alcance de los transceptores de radiofrecuencia y la visibilidad entre dispositivos.

2.1.1 Protocolo de comunicación de la red de sensores

El protocolo de comunicación para la red de sensores definirá las reglas y formatos que los nodos deberán seguir para poder intercambiar datos entre sí. Algunas características de la red de sensores deben ser consideradas para la selección de un protocolo adecuado.

Debido a que los nodos son energizados desde una batería y estos deben mantenerse operativos por largos periodos de tiempo sin mantenimiento, el protocolo de comunicación debe minimizar el consumo de potencia para garantizar el mayor tiempo de vida posible para la red.

- **Protocolo Zigbee**

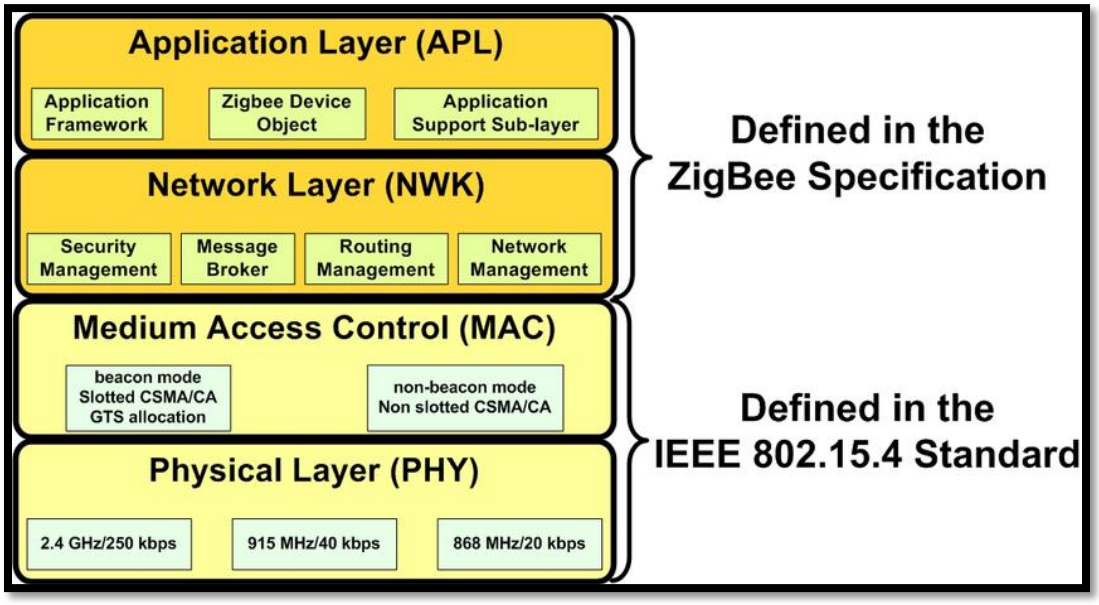
Para esta aplicación se ha seleccionado el protocolo Zigbee /IEEE 802.15.4. Este protocolo es usado para interconectar de manera inalámbrica a dispositivos que requieren bajo consumo de potencia, bajas tasas de transmisión y seguridad en la red usando encriptación simétrica, lo cual se adecua a esta aplicación.

El bajo consumo de potencia usado por este protocolo tiene una limitación en la distancia de transmisión entre dispositivos, sin embargo, el alcance

total puede aumentar según el tipo de topología de red usada en la aplicación. El protocolo Zigbee soporta las topologías estrella y árbol de forma nativa.

El protocolo Zigbee está basado en el estándar IEEE 802.15.4, que especifica la capa física y la capa de enlace de datos usados en redes inalámbricas de área personal de baja velocidad (LR-WPAN) para dispositivos con largo tiempo de vida.

FIGURA Nº4. 2
CAPAZ DEL PROTOCOLO ZIGBEE



❖ **Capa Física (PHY)**

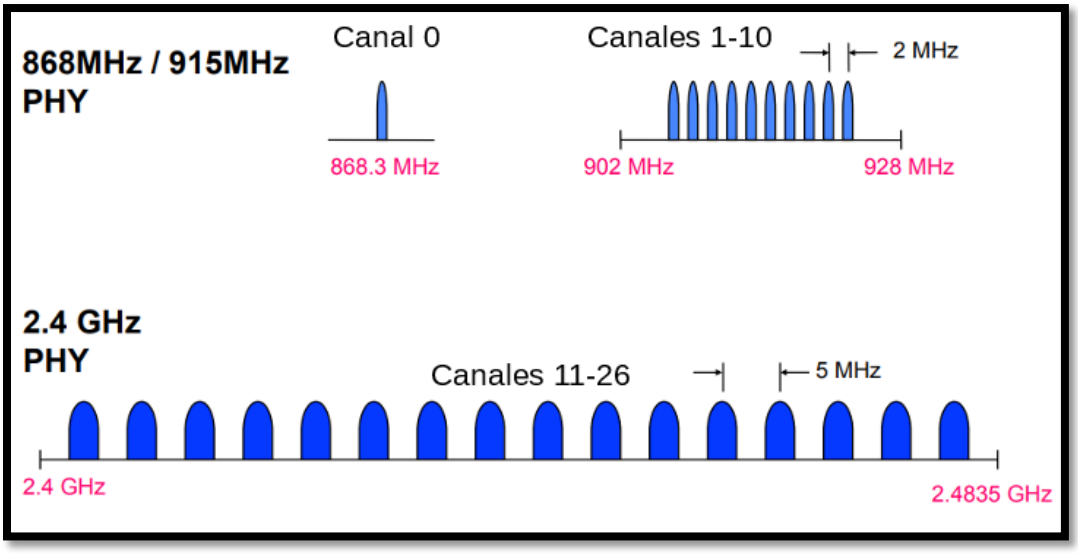
La capa física del estándar IEEE 802.15.4 define 3 bandas de frecuencia de operación con 27 canales. Estas bandas son 868MHz, 915MHz, 2.4GHz.

- El canal 0 está entre 868.0MHz y 868.6MHz con una tasa de transmisión de 20kbps.

- Los canales 1 a 10 están entre 902.0MHz y 928.0MHz y cada canal tiene una tasa de transmisión de 40kpbs.
- Los canales 11 a 26 están entre 2.4GHz y 2.48GHz y cada canal tiene una tasa de transmisión de 250kbps.

Los canales 0 a 10 usan modulación de fase binaria BPSK con una sensibilidad de al menos -92dBm mientras que los canales 11 a 26 usan modulación de cuadratura QPSK con una sensibilidad de al menos -85dBm. La potencia de transmisión debe ser de al menos -3dBm.

FIGURA Nº4. 3
CANALES USADOS POR EL ESTÁNDAR 802.15.4



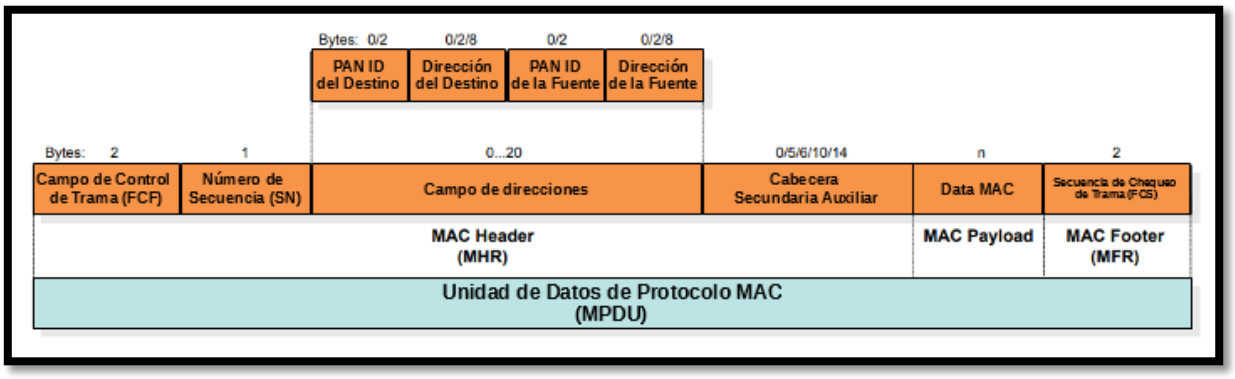
❖ **Capa de Control de Acceso de Medios (MAC)**

La capa de control de acceso de medios (MAC) del estándar IEEE 802.15.4 funciona como interfaz entre la capa física y la subcapa LLC. Esta maneja y valida las tramas, controla los canales de acceso y envía el acuse de

recibo. Usa el mecanismo de CSMA/CA para evitar colisiones de paquetes transmitidas por otros dispositivos en la red.

Las tramas son denominadas unidades de datos de protocolo MAC (MPDU) y tiene 3 secciones principales: La cabecera MAC (MHR), la data MAC (payload) y la cola MAC (MFR).

FIGURA Nº4. 4
ESTRUCTURA DE LA UNIDAD DE DATOS DEL PROTOCOLO MAC (MPDU)



- El campo de control de trama (FCF), contiene información acerca del tipo de trama y otros indicadores de control.
- El número de secuencia (SN), especifica el identificador de secuencia para la trama.
- El identificador de red (PAN ID) del destino, es un identificador único que pertenece a la red donde se encuentra el dispositivo destino que debe recibir la trama.
- La dirección de destino, especifica la dirección del dispositivo destino que debe recibir la trama.

- El identificador de red (PAN ID) de la fuente, es un identificador único que pertenece a la red donde se encuentra el dispositivo que originó la trama.
- La dirección de la fuente, especifica la dirección del dispositivo que originó la trama.
- La cabecera secundaria auxiliar, especifica información requerida para el procesamiento de seguridad.
- La data MAC (payload), contiene la información específica para enviar al dispositivo destino.
- La secuencia de chequeo de trama (FCS), es un cálculo de chequeo de redundancia cíclica (CRC).

Los campos de control de trama (FCF), número de secuencia (SN) y la secuencia de chequeo de trama (FCS) siempre son parte de una trama MPDU, sin embargo, los campos de direcciones, cabecera auxiliar y la data MAC pueden ser omitidos si no son necesarias.

Según qué campos están presentes en el MPDU estos pueden ser tramas beacon, tramas de datos, tramas de acuse de recibo o tramas de comando MAC.

CUADRO Nº 2
TIPOS DE TRAMAS USADAS EN LA CAPA MAC

Tipo de tramas MPDU	**Descripción**
Trama Beacon	Es usada por el coordinador de la red para sincronizar a todos los otros dispositivos dentro de la red. Esta trama incluye: - Campo de control de trama (FCF).

	- Número de secuencia (SN). - Identificador de red (PAN ID) de la fuente. - Dirección de la fuente. - Cabecera secundaria auxiliar - Data MAC - Secuencia de chequeo de trama (FCS)
Trama de Datos	Es usada para enviar datos a otros dispositivos en la red. Esta trama incluye: - Campo de control de trama (FCF). - Número de secuencia (SN). - Identificador de red (PAN ID) del destino. - Dirección del destino. - Identificador de red (PAN ID) de la fuente. - Dirección de la fuente. - Cabecera secundaria auxiliar - Data MAC - Secuencia de chequeo de trama (FCS)
Trama de Acuse de recibo	Es usada para indicar la recepción exitosa de una trama al dispositivo fuente. Esta trama incluye: - Campo de control de trama (FCF). - Número de secuencia (SN). - Secuencia de chequeo de trama (FCS)
Trama de comando MAC	Es usada para hacer solicitud de asociación de red, tramas de datos, beacons y otras notificaciones. Esta trama incluye: - Campo de control de trama (FCF). - Número de secuencia (SN). - Identificador de red (PAN ID) del destino. - Dirección del destino. - Identificador de red (PAN ID) de la fuente. - Dirección de la fuente. - Cabecera secundaria auxiliar

	- Data MAC - Secuencia de chequeo de trama (FCS)

Cada dispositivo que transmite datos debe tener en consideración que el destino tenga suficiente tiempo para procesar la data recibida antes de enviar la siguiente trama. Por lo tanto, las tramas consecutivas deben estar separadas por un periodo de espaciamiento entre tramas (IFS). Este periodo depende del tamaño de la trama anterior, de manera que tramas más extensas tendrán un largo periodo de espaciamiento (LIFS) y otras tramas tendrán un corto periodo de espaciamiento (SIFS).

- **Capa de Red (NWK)**

El protocolo Zigbee define la capa de red, que permite las transmisiones confiables y seguras entre dispositivos. La capa de red provee funciones que aseguran la operación adecuada de la capa MAC y para dar una interfaz a la capa de aplicación. Esta capa define dos entidades, la entidad de datos de la capa de red (NLDE) y la entidad de administración de la capa de red (NLME).

CUADRO Nº 3
ENTIDADES USADAS EN LA CAPA DE RED

Entidades de la capa de Red	**Descripción**
Entidad de Datos de la capa de red	Permite la transmisión de las unidades de datos del protocolo de aplicación (APDU) entre dos o más dispositivos ubicados en la misma red. Sus funciones son:

(NLDE)	- Generar las unidades de datos del protocolo de red (NPDU). - Transmitir el NPDU al dispositivo apropiado, sea el nodo destino o el siguiente dispositivo en la ruta.
Entidad de Administración de la capa de red (NLME)	Ofrece el servicio de administración para que la aplicación pueda interactuar con varios aspectos de la red. Sus funciones son: - Configurar el dispositivo como coordinador. - Iniciar una nueva red Zigbee. - Asociarse o desasociarse con una red Zigbee. - Asignar direcciones a los dispositivos que se unen a la red. - Descubrimiento de rutas. - Ruteo de paquetes para estos lleguen a su destino.

❖ **Capa de Aplicación (APL)**

La capa de aplicación definida en el protocolo Zigbee se divide en las subcapas de soporte de aplicación (APS) la subcapa de objetos de dispositivos Zigbee (ZDO) y el Framework de aplicación.

- La Framework de Aplicación, es el entorno que especifica un rango de perfiles, estándares, descriptores y formatos para que los objetos de aplicación puedan enviar y recibir datos. Los objetos de aplicación son definidos por el fabricante del producto Zigbee y cada objeto es asignado a una dirección endpoint, que tienen un valor entre 1 y 240.
- Los objetos de dispositivo Zigbee (ZDO), es el protocolo en la capa de aplicación Zigbee y es responsable de inicializar la subcapa de soporte de aplicación (APS), definir el modo de operación del dispositivo en la red

(coordinador, router, dispositivo final), administrar los procesos de descubrimiento de dispositivos y las reglas de seguridad.

- La subcapa de soporte de aplicación (APS), es el componente central de la capa de aplicación, este provee la interfaz entre la capa de red y la capa de aplicación a través de un conjunto de servicios, que son usados por los objetos de dispositivo Zigbee (ZDO) y los objetos de aplicación del Framework. Los servicios de la APS son realizados por la entidad de datos (APSDE) y la entidad de administración (APSME).

CUADRO Nº 4
ENTIDADES USADAS EN LA CAPA DE APLICACIÓN

Entidades de la APS	**Descripción**
Entidad de Datos de la subcapa de soporte de aplicación (APSDE)	Provee servicios a la capa de red, a los ZDO y a los objetos de aplicación que permiten la transmisión de las unidades de datos del protocolo de aplicación (APDU) entre dos o más dispositivos ubicados en la misma red. Esos servicios son: - Generación del APDU. - Transmisión entre dispositivos asociados a la red. - Incrementar la fiabilidad de las transmisiones en la capa de red. - Rechazo de mensajes duplicados. - Fragmentación y ensamblaje de mensajes.
Entidad de Administración de la subcapa de soporte de aplicación (APSME)	Provee servicios de administración que permite a la aplicación interactuar con la capa de red. Estos servicios son: - Relacionar dos dispositivos basados en sus características y formar una tabla para almacenar esta información. - Administrar dispositivos dentro de grupos que comparten una dirección para identificarlos.

❖ **Dispositivos de una red Zigbee**

En el estándar IEEE 802.15.4 se definen los dispositivos de función completa (FFDs) y dispositivos de función reducida (RFDs).

Un FFD puede ser designado como coordinador y su función es mantener activa la red y administrar otros dispositivos.

Un RFD solo puede enviar y recibir datos desde un coordinador con el cual esté asociado. A este dispositivo también se le denomina dispositivo final.

Según la función que un nodo cumple en la topología de la red Zigbee, estos pueden ser coordinadores, routers o dispositivos finales.

CUADRO Nº 5
TIPOS DE DISPOSITIVOS DENTRO UNA RED ZIGBEE

Tipo de Dispositivo	**Descripción**
Dispositivo Coordinador	Este tiene la función de formar la red. El coordinador es responsable de establecer el canal de operación y el identificador de la red de área personal (PAN ID). Una vez establecidos, el coordinador permitirá que otros dispositivos se unan. Cuando la red ya está formada, el coordinador funcionará como un router, y participa en el proceso de ruteo de paquetes a su destino. Sólo se admite un coordinador por red PAN, y este debe ser energizado por una fuente permanente (no usa batería).
Dispositivo Router	Este tiene la función de mantener la información de la red y usarla para determinar la mejor ruta para enviar datos. Un router debe unirse a una red antes que otros dispositivos finales puedan unirse.

	Varios routers pueden operar dentro una red PAN y este debe ser energizado por una fuente permanente (no usa batería).
Dispositivo Final	Este tiene la función de enviar datos, pero no son capaces de elegir rutas para la transmisión. Varios dispositivos finales pueden operar dentro de una red PAN y estos son energizados por baterías, por lo cual deben operar en modos de ahorro de energía para extender su tiempo de vida.

- El coordinador es el nodo encargado de iniciar la red. Este debe seleccionar un canal apropiado para que opere la red, y lo hace con un escaneo de energía en múltiples canales de la banda de frecuencia en el cual opera, formando así una lista de canales potenciales donde iniciar la red. Los canales que tienen altos niveles de potencia son descartados.

  Después del escaneo de canales, el coordinador escanea la lista de canales restantes para obtener una lista de redes vecinas. Para esto, el coordinador realiza una transmisión de solicitud de beacon en cada canal disponible, de manera que, si existe una red en dicho canal, los otros routers o coordinador de esa red, pueden responder al pedido con información de su propio identificador de red (PAN ID). Este proceso se denomina escaneo de red PAN. Al terminar dicho escaneo, el coordinador elige su propio canal y su propio identificador PAN ID, de manera aleatoria para iniciar la red.

- Los routers deben descubrir y unirse a la red Zigbee antes de que puedan participar en sus funciones. Después de que un router se haya unido a la red, este permite que otros dispositivos se unan a la red y podrá comunicarse a través de las rutas que formará. Para descubrir alguna red cercana, el router ejecuta un escaneo de redes PAN, de manera similar a como lo hizo el coordinador antes de iniciar la red. El router realiza una transmisión de solicitud de beacon en el primer canal que escaneará en su lista. Los otros routers y coordinador de la red que opera en dicho canal, responderán con la información de su identificador de red (PAN ID) e información sobre si unirse a la red está permitido o no. El router evaluará esta información y si encuentra una red válida, procederá a unirse a la red. Si no encuentra una red válida en dicho canal, procederá a escanear el siguiente.

  Para unirse a la red, el router envía una transmisión de solicitud de asociación. El dispositivo encargado en la red Zigbee le responderá ese pedido con un mensaje que puede aceptar o negar dicha solicitud. Si el router finalmente se une a la red, recibe una dirección de 16bits del dispositivo que le dio permiso.
- Los dispositivos finales son los nodos que sólo envían y reciben mensajes y no ejecutan ninguna otra función especial en la red. En la especificación Zigbee se nota que estos son los únicos dispositivos que pueden entrar en hibernación. De manera similar a los routers, los dispositivos finales, deben descubrir y unirse a una red Zigbee antes de poder participar en la comunicación. El dispositivo final realiza un

escaneo de redes PAN en cada canal de la banda de frecuencias disponible. Si encuentra una red válida, realiza una solicitud de asociación y si esta es exitosa, el dispositivo será asignado una dirección de 16bits.

Todos los dispositivos en la red Zigbee tienen una dirección única de 64bits y una dirección de red de 16bits.

La dirección de 64bits, a veces es denominada dirección MAC o dirección extendida, es asignada al dispositivo durante su fabricación.

La dirección de 16bits, es asignada al dispositivo cuando este se une a una red Zigbee. La dirección 0x0000 es reservado para el coordinador de la red, y el resto de dispositivos reciben un valor aleatorio generado por el coordinador. Estas direcciones no son estáticas y pueden cambiar bajo ciertas condiciones.

Todas las transmisiones Zigbee y las tablas de ruteo, usan las direcciones de 16bits, sin embargo, es posible enviar datos usando la dirección de 64bits para asegurar que la transmisión llegue al dispositivo correcto.

- **Transmisiones en la red Zigbee**

Las transmisiones en la red Zigbee pueden ser Unicast o Broadcast.

CUADRO Nº 6
TIPO DE TRANSMISIÓN EN UNA RED ZIGBEE

Tipo de transmisión	Descripción
Unicast	Son mensajes enviados desde un dispositivo fuente, hacia un dispositivo destino. El destino puede ser un dispositivo vecino o puede requerir múltiples saltos a través de una ruta para recibir el mensaje.
Broadcast	Son mensajes que se propagan por toda la red, de manera que todos los nodos reciben la transmisión. Los coordinadores y routers que reciben un mensaje broadcast, lo retransmiten 3 veces para asegurar su difusión. Para evitar que los mensajes se envíen de manera infinita, los nodos que retransmiten mantienen un registro de cuantas veces se han enviado los mensajes. El uso excesivo de mensajes broadcast puede congestionar la red.

Cuando se envía un mensaje Unicast, la capa de red del protocolo Zigbee usa la dirección de 16bits para identificar el destino y la ruta a usar. Si la dirección es desconocida, el protocolo realiza una etapa de descubrimiento de direcciones antes de enviar los datos.

Para descubrir la dirección de red de un dispositivo remoto, el dispositivo que inicia el descubrimiento envía una transmisión broadcast de descubrimiento de dirección. Este mensaje incluye la dirección de 64bits del dispositivo remoto. Todos los nodos que reciben la transmisión, revisan la dirección de 64bits del dispositivo remoto y la comparan con su propia

dirección de 64bits. Si la dirección es la misma, el dispositivo envía una respuesta con la información de su dirección de 16bits al dispositivo que inició la transmisión broadcast.

Todos los dispositivos Zigbee mantienen una tabla de direcciones que mapea las direcciones de 64bits con las direcciones de red de 16bits. Cuando una transmisión usa la dirección de 64bits como destino, el protocolo buscará cual es la dirección de red que corresponde al valor de 64bits, y si no encuentra dicha dirección, realiza el proceso de descubrimiento de direcciones. Esta tabla puede almacenar hasta 20 direcciones.

- **Ruteo de paquetes en la red Zigbee**

El protocolo Zigbee establece enlaces bidireccionales y usa el mecanismo de estatus de enlace para determinar cuál es la calidad de los enlaces entre los nodos de la red. Este mecanismo consiste en los routers y coordinador enviando mensajes periódicos de estatus de enlace a sus dispositivos vecinos. En el mensaje contiene una lista de los dispositivos vecinos, cada uno con un valor de calidad para su enlace. Con esta información, un dispositivo puede determinar cuál enlace es más apropiado para enviar información y seleccionar una ruta apropiada.

Las técnicas de ruteo usadas en el protocolo Zigbee son ADOV Mesh-Routing, Many-to-One Routing y Source Routing.

CUADRO Nº 7
TÉCNICAS DE RUTEO EN UNA RED ZIGBEE

Técnica de Ruteo	**Descripción**	**Aplicación Recomendada**
Ad hoc On-demand Distance Vector (AODV) Mesh-Routing	Las rutas son creadas entre el nodo fuente y el destino, con saltos a través de múltiples nodos.	Usada en redes pequeñas que no contienen más de 40 dispositivos.
Many-to-One Routing	Se realiza una transmisión broadcast para hallar todas las rutas en todos los dispositivos hacia el nodo central que envía el mensaje.	Usada en redes donde varios dispositivos remotos envían datos a un solo dispositivo recolector.
Source Routing	El paquete de datos incluye la ruta que el mensaje debe seguir para llegar a su destino.	Usada en redes grandes, con más de 40 dispositivos.

- AODV Mesh-Routing, Permite que los routers y coordinadores participen en la creación de rutas entre dispositivos fuente y destino usando el proceso de descubrimiento de rutas. Este proceso está basado en el algoritmo de ruteo Ad hoc On-demand Distance Vector (AODV).

  Este algoritmo AODV usa tablas en cada nodo, que almacena el próximo salto para llegar al nodo destino. Si el próximo salto es desconocido, se inicia el proceso de descubrimiento de rutas. Debido a que el número de rutas que puede almacenar un router es limitado, el descubrimiento de rutas se hace más frecuente en redes más grandes.

Cuando un nodo fuente encuentra una ruta al nodo destino, este envía un mensaje broadcast de comando de solicitud de ruta. El comando de solicitud de ruta, incluye la dirección del nodo fuente, la dirección del nodo destino y el valor de costo de la ruta. A medida que el mensaje broadcast se propaga por la red, cada nodo que recibe la ruta actualiza su tabla de rutas.

Cuando el nodo destino recibe la solicitud de ruta, este transmite un mensaje de respuesta de ruteo al nodo que originó la solicitud. Los nodos intermedios que reciben esta respuesta se encargan de reenviarlo al nodo fuente.

- Many-to-One Routing, en redes donde varios dispositivos deben enviar datos a un nodo central, la técnica AODV Mesh-Routing es demasiado engorrosa. Si cada dispositivo tuviese que descubrir la ruta al dispositivo central antes de enviar datos, la red quedaría inundada de transmisiones broadcast. La técnica Many-to-One (MTO) está especialmente optimizada para este tipo de redes. En lugar de que varios dispositivos hagan su propio descubrimiento de rutas, el nodo central realiza una sola transmisión broadcast para establecer la ruta en todos los otros dispositivos.

  El broadcast many-to-one es un mensaje de solicitud de ruta con la dirección del nodo central y cada dispositivo que recibe el mensaje almacena la información en su tabla de rutas. Este mensaje se envía de manera periódica para actualizar las rutas de la red.

- Source Routing, para aplicaciones donde un dispositivo debe enviar datos a múltiples nodos remotos, la técnica AODV Mesh-Routing tendría que realizar el proceso de descubrimiento de rutas para cada dispositivo destino. Esto puede resultar en congestión en la red. Con la técnica Source-Routing permite que un dispositivo central almacene rutas para varios nodos remotos. Este debe realizar transmisiones broadcast many-to-one de manera periódica, para crear rutas MTO en todos los otros dispositivos.

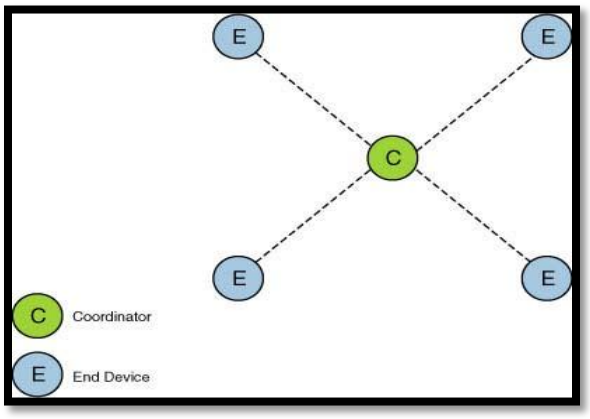
❖ **Topologías de redes Zigbee**

El protocolo Zigbee soporta tres tipos de topologías de red, que afecta como los mensajes son enviados en la red y que dispositivos se pueden comunicar entre sí.

- La topología Estrella, es la más simple que soporta el protocolo. Todos los dispositivos finales se conectan directamente al nodo coordinador y este se encarga de enviar y recibir los mensajes.

  Con esta topología, la velocidad de la red es limitada por el coordinador y si esta falla, toda la red falla. El rango de la red es limitado por el rango del coordinador. En este tipo de red, no hace falta realizar ruteo de paquetes, sin embargo, routers pueden ser parte de la red.

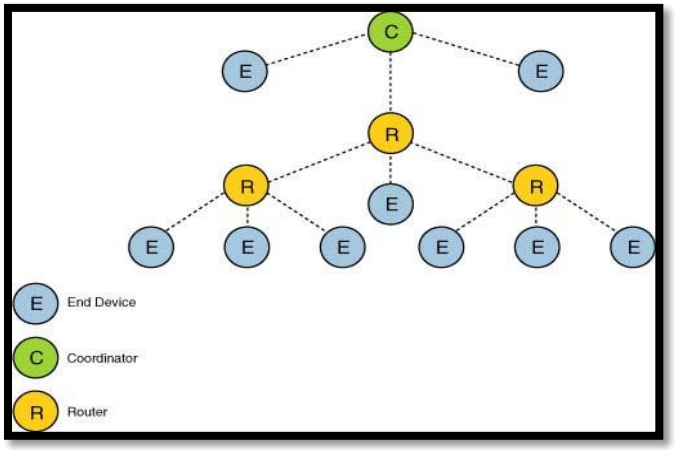
FIGURA Nº4. 5
TOPOLOGÍA DE RED TIPO ESTRELLA



- La topología Árbol, se considera que el nodo coordinador es un “nodo raíz” y los dispositivos finales son “nodos hojas”. La conexión entre estos se hace a través de nodos routers. Los nodos routers ayudan a extender el alcance de la red pues estos permiten que otros nodos se conecten a la red sin necesidad de acceder al nodo coordinador.

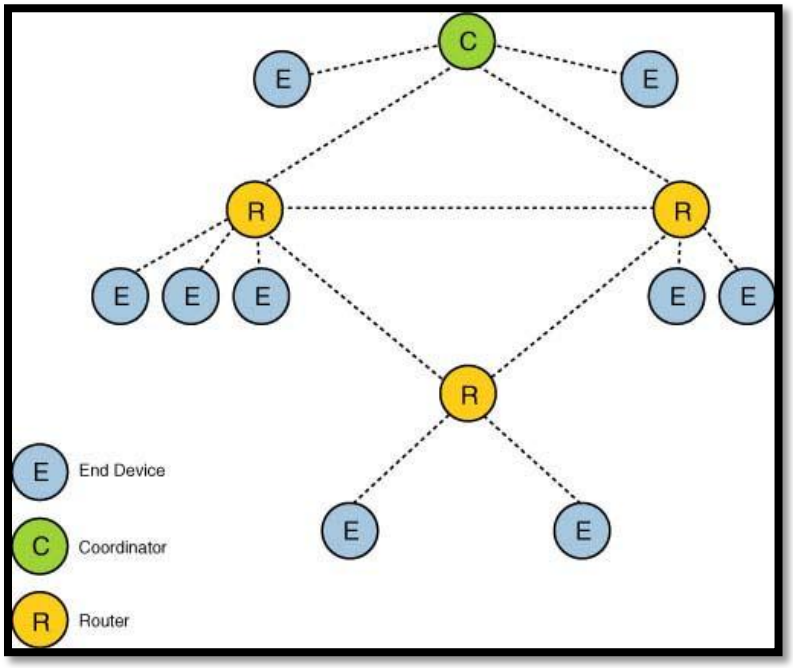
  La desventaja de esta topología es que, si un nodo router falla, los dispositivos conectados a él no tendrán acceso a la red.

FIGURA Nº4. 6
TOPOLOGÍA DE RED TIPO ÁRBOL



- La topología Mesh, es la más flexible ofrecida en el protocolo Zigbee. También es conocida como red Peer to Peer. Es similar a la red Árbol debido a que los routers permiten extender el alcance de la red, sin embargo, en la topología Mesh, los routers se pueden comunicar directamente entre sí o con el coordinador, según estén dentro del rango de transmisión. De esta manera la transmisión de mensajes se realiza a través de múltiples saltos y pueden seguir múltiples rutas. El protocolo Zigbee incluye la capacidad de descubrimiento de rutas, por lo cual si un dispositivo en la red falla, los mensajes aún tienen posibilidad de llegar a su destino.

FIGURA Nº4. 7
TOPOLOGÍA DE RED TIPO MESH



2.1.2 Diseño de Hardware de los Nodos

Los nodos son la parte más importante de la red y está conformado por la etapa de alimentación, un controlador, sensores, módulo de localización y un transceptor de radio. Se pueden identificar dos tipos distintos de nodos para el diseño de esta red de sensores.

Según la función que cumplirán estos nodos en la red de sensores, estos pueden ser nodos fuente o nodos sumidero.

- Los nodos fuente son los encargados de recolectar datos a través de sus sensores de manera periódica y reportar dicha información a los nodos sumidero con un enlace inalámbrico.
- Los nodos sumidero se encargan de recibir, almacenar y procesar la data de los nodos fuente Ese nodo sumidero se considera como un Gateway que tiene la capacidad de comunicarse con otras redes LAN, WLAN, celular, etc. de manera que pueden reenviar la información que será legible para el usuario final.

FIGURA Nº4. 8
DIAGRAMA DE BLOQUES DE UN NODO FUENTE

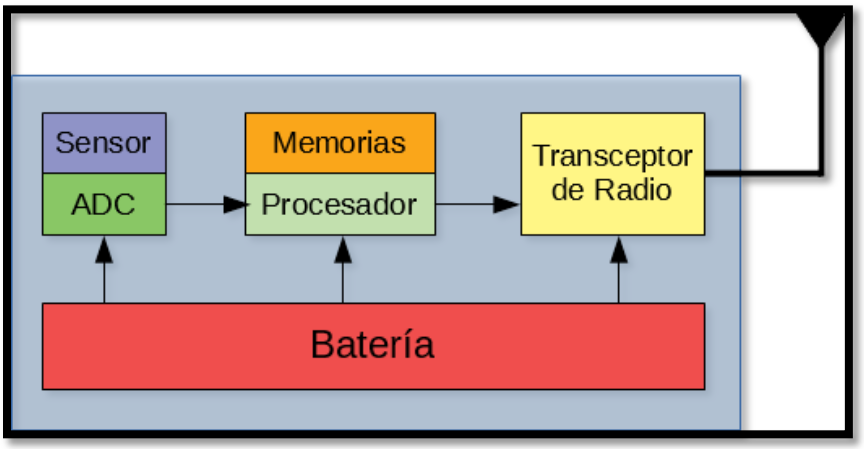
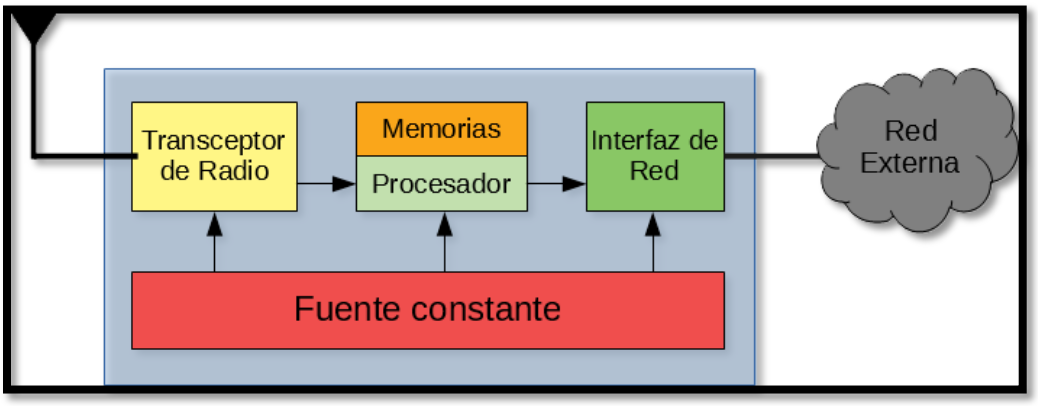


FIGURA Nº4. 9
DIAGRAMA DE BLOQUES DE UN NODO SUMIDERO



El diseño de ambos tipos de nodos se realizó de manera que ambos usen los mismos componentes. De esta manera se simplifica el diseño de los dispositivos, y su comportamiento será definido por software.

- **Sensor**

Los sensores son los componentes que detectan y responden a un evento o estímulo del ambiente que los rodea. Este estímulo puede ser temperatura, movimiento, humedad o, en el caso de esta investigación, radiación no ionizante.

Este componente es central para el diseño de los nodos fuente pues es la única forma en la cual el dispositivo puede adquirir datos de su entorno para después procesarla y enviarla a su destino.

Para esta aplicación, se necesita un sensor capaz de detectar la radiación no ionizante presente en las bandas usadas por los servicios públicos de telecomunicaciones.

Se ha elegido el módulo RTL-SDR para operar como el sensor de los nodos fuente. Este módulo funciona a base de 2 circuitos integrados especializados, el R820T2 y el RTL2832U.

El R820T2 es un circuito sintonizador integrado que contiene un LNA, mezclador, PLL, VGA, reguladores y filtros que están diseñados para recepción de señales RF entre 24 MHz a 1766 MHz. Las técnicas de integración en este dispositivo permiten un alto rendimiento y bajo consumo de energía los cuales son ideales para esta aplicación.

El RTL2832U es un demodulador COFDM que soporta una interfaz USB 2.0, lo que facilita su uso con el controlador seleccionado para el nodo, y un ADC de 7bits para leer las señales RF recibidas por el R820T2.

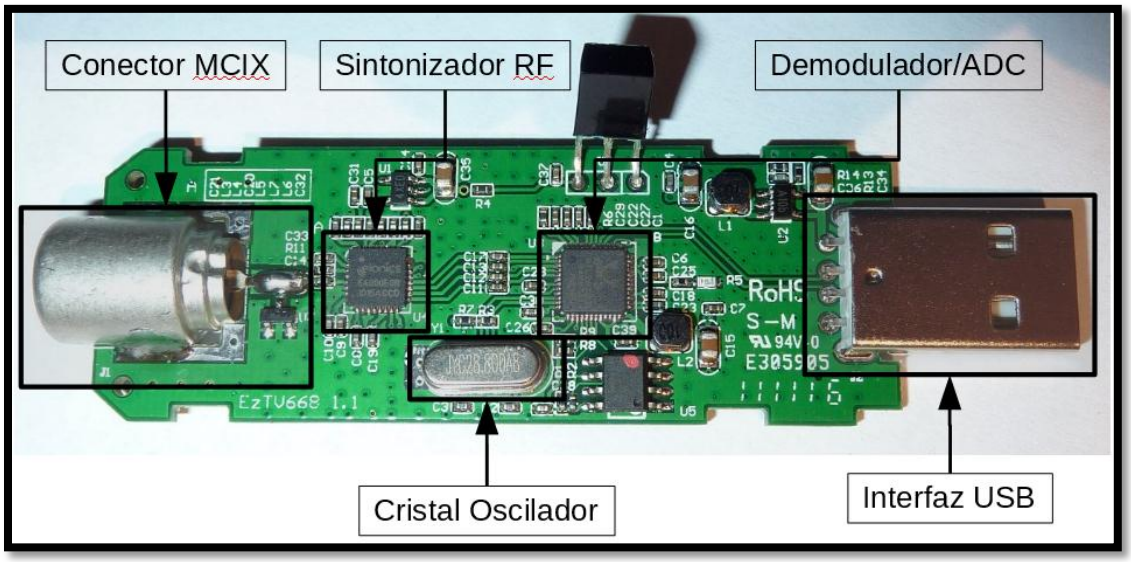
CUADRO Nº 8
CARACTERÍSTICAS DEL MÓDULO RTL-SDR

Módulo RTL-SDR	
Costo	$12
Frecuencia de operación	24MHz – 1766MHz
Figura de Ruido	3.5dB
Ganancia de LNA	-7.5dB - +35dB
Sensibilidad	-91.5dBm
Impedancia	50ohm
Frecuencia de muestreo	2.56Msps
Voltaje de operación	5V USB
Consumo de corriente	100mA

El RTL-SDR fue originalmente concebido como un receptor para televisión digital, pero sus características han sido útiles para el desarrollo de nuevas aplicaciones de software definido por radio (SDR). El módulo trabaja con frecuencias intermedias IF de 3.57MHz 4.57MHz con un error de aproximadamente 20ppm. Este usa un oscilador basado con un PLL que alimenta el mezclador. La salida es adquirida por una ADC a una frecuencia de muestreo de 28.8Msps, sin embargo, el resultado es muestreado una segunda vez, a 2.56Msps y finalmente el resultado que se envía a través del interfaz USB es de números de 8bits, sin signo, por lo que se debe restar -127 para obtener el rango completo.

Existen múltiples opciones de software para visualizar e interpretar la data recibida por el módulo RTL-SDR, muchas de estas son open-source y bien documentadas, lo que facilitará el diseño del firmware para el controlador del nodo fuente.

FIGURA Nº4. 10
MÓDULO RTL-SDR



- **Controlador**

El controlador tiene la labor de procesar datos y controlar el funcionamiento de los otros elementos dentro del nodo. Generalmente el controlador es implementado con un microcontrolador debido a su bajo costo, bajo consumo de energía, flexibilidad para comunicarse con otros dispositivos y facilidad para programar, sin embargo, un microcontrolador no tendrá el rendimiento adecuado para esta aplicación debido a que se necesitará una gran cantidad de memoria RAM para capturar y procesar la información del receptor RF. Para cumplir con estos requisitos se ha decidido usar una computadora de tarjeta reducida o Single Board Computer (SBC) debido a que poseen gran potencial de procesamiento a un bajo costo, bajo consumo de energía y en un tamaño reducido.

Se ha elegido la tarjeta Raspberry Pi, uno de los productos más reconocidos en esta serie de dispositivos y que posee gran cantidad de soporte, comunidad y aplicaciones en múltiples campos de ingeniería y educación.

El Raspberry Pi es una computadora de tarjeta reducida (SBC) de bajo costo, desarrollado por la Fundación Raspberry Pi con el objetivo de promover la enseñanza de ciencias de computación en las escuelas del Reino Unido y países en desarrollo. Este dispositivo no incluye periféricos como teclado, mouse o pantalla.

El modelo Raspberry Pi Zero es la tarjeta más pequeña y económica de la serie, siendo la más adecuada para la aplicación del nodo en la red de sensores.

CUADRO Nº 9
CARACTERÍSTICAS DEL RASPBERRY PI ZERO

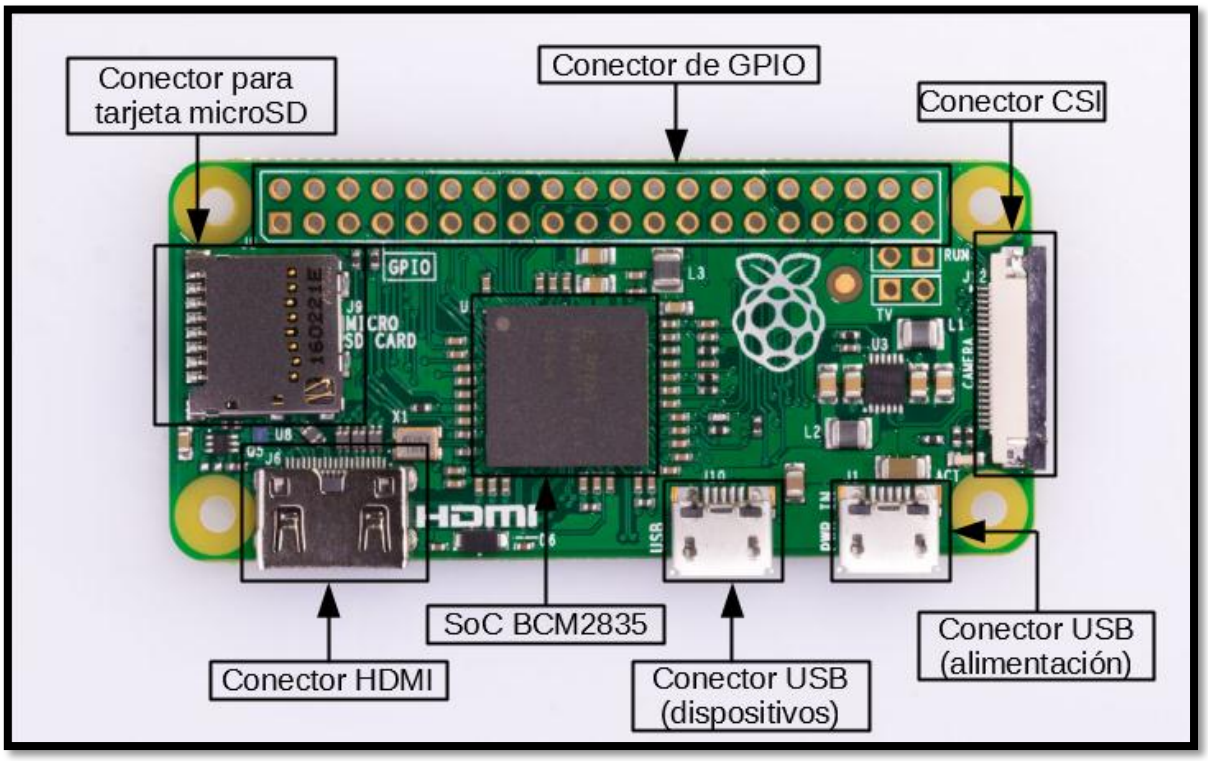
Raspberry Pi Zero	
Costo	$5
Procesador	ARM1176JZF-S, reloj de 1GHz
Memoria RAM	512MB
Memoria de almacenamiento	MicroSDHC
Puertos y Periféricos	1 Micro-USB OTG
	1 Mini-HDMI 1080p60
	17 GPIOs
	1 I^2C
	1 SPI
	1 UART
Voltaje de alimentación	5V Micro-USB
Voltaje de operación	3.3V
Consumo de corriente (estado idle)	80mA (0.4W)
Consumo de corriente (carga)	200mA (1.0 W)

El Raspberry Pi Zero está basado en el sistema en chip (SoC) Broadcom BCM2835. Este circuito integrado incluye un procesador ARM1176JZF-S con una señal de reloj de 1GHz, una unidad de procesamiento de gráficos (GPU) VideoCore IV, 512MB de memoria RAM y periféricos como:

- Interfaz de Receptor y Transmisor Asíncrono Universal (UART), permite la comunicación serial con otros dispositivos.

- Controlador Serial Broadcom (BSC), es un periférico propietario de Broadcom que es compatible con la interfaz I2C de Phillips. Este soporta el funcionamiento de maestro, con direcciones de 7bits y 10bits y reloj programable.
- Interfaz Serial de Periféricos (SPI), es un protocolo de comunicación que permite transmitir data serial de manera síncrona entre múltiples circuitos integrados.
- Controlador de Acceso Directo de Memoria (DMA), permite que otros periféricos del SoC accedan a la memoria del sistema sin intervención del procesador principal.
- Controlador de Almacenamiento Externo (eMMC), provee una interfaz para leer memorias MMC y tarjetas SD.
- Entradas y Salidas de Propósito General (GPIO), permiten la interacción con otros dispositivos a través de niveles de voltaje lógicos de 0V o 3.3V. Muchos de estos pines pueden ser configurados con funciones alternativas para usar otros periféricos.
- Modulador de Ancho de Pulso (PWM), es una técnica de modulación que permite cambiar el ciclo de trabajo de una señal periódica.
- Interfaz USB, permite la conexión de teclados, mouse y otros periféricos típicos de una computadora.

FIGURA Nº4. 11
TARJETA RASPBERRY PI ZERO



La tarjeta Raspberry Pi Zero posee un conector para cable HDMI, para conectar la tarjeta a una pantalla, y un conector CSI, para conectar una cámara. En esta aplicación no se usará ninguno de estos dos conectores.

En la tarjeta existen dos conectores USB. El conector USB al extremo de la tarjeta es usado para alimentar la tarjeta con un cable microUSB-B y una fuente de voltaje de 5V. La tarjeta incluye un regulador de voltaje a 3.3V, que es el voltaje necesario para operar el BCM2835 y sus periféricos.

El consumo de corriente dependerá de que operaciones esté realizando la tarjeta y si esta está energizando otros dispositivos. El consumo máximo de corriente es de 1000mA.

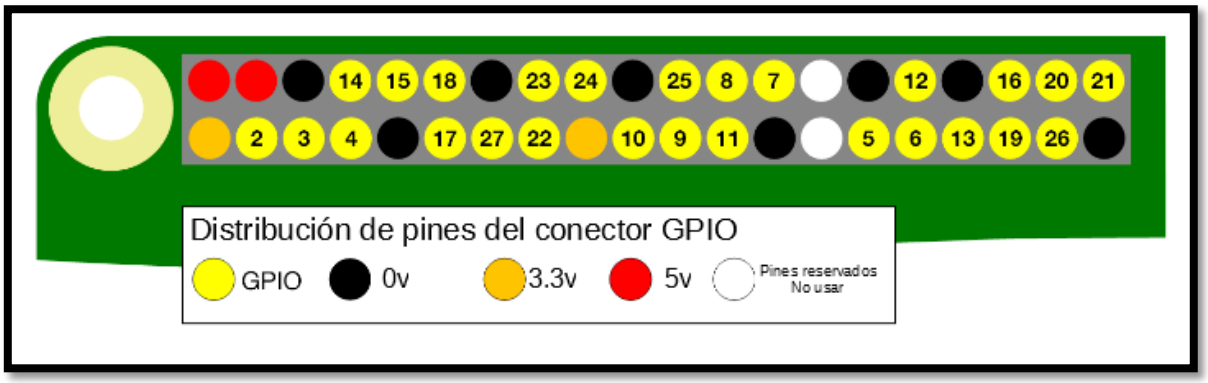
El segundo conector USB está directamente conectado a la interfaz USB del BCM835, lo que le da la capacidad de operar como un dispositivo USB On-The-Go (OTG). Un dispositivo USB OTG es capaz de conectarse con otros dispositivos USB como memorias, teclados, mouse, etc.

Este puerto USB OTG será usado para conectar el Raspberry Pi Zero con el módulo RTL-SDR.

La tarjeta también deja disponibles una sección con los pines del BCM2835, lo cual nos da acceso a sus periféricos, incluidos el UART y los GPIO.

La distribución de pines en la tarjeta es la siguiente.

FIGURA Nº4. 12
DISTRIBUCIÓN DE PINES DEL CONECTOR GPIO EN EL RASPBERYY PI ZERO



El conector ofrece pines conectados a la fuente de 5V, a la salida del regulador de 3.3V y a la conexión a 0V. Estos pueden ser usados para energizar la tarjeta, de manera separada de su conector USB, o para alimentar otros dispositivos que serán usados junto al Raspberry Pi.

El conector deja disponible 26 pines GPIO, los cuales solo soportan niveles lógicos de hasta 3.3V y un flujo de corriente de 8mA. Los pines GPIO pueden ser configurados con resistencias internas pull-up o pull-down, según sea necesario en la aplicación, además muchos de los pines soportan funciones alternativas propias de los periféricos del BCM2835.

Las funciones y comportamiento de los pines serán definidos a través de software.

CUADRO Nº 10
PERIFÉRICOS DEL RASPBERRY PI ZERO

Periférico del BCM2835	**Función en el Periférico**	**GPIO**
UART	TX: Transmisor	GPIO14
	RX: Receptor	GPIO15
I2C	SDA: Línea de datos	GPIO02
	SCL: Línea de reloj	GPIO03
SPI0	SCK: Reloj	GPIO11
	MOSI: Datos de salida	GPIO10
	MISO: Datos de entrada	GPIO09
	CE0: Selector de chip 0	GPIO08
	CE1: Selector de chip 1	GPIO07
SPI1	SCK: Reloj	GPIO21
	MOSI: Datos de salida	GPIO20
	MISO: Datos de entrada	GPIO19
	CE0: Selector de chip 0	GPIO18
	CE1: Selector de chip 1	GPIO17
	CE2: Selector de chip 2	GPIO16
PWM	Generado por Hardware	GPIO12, GPIO13, PIO18, GPIO19

	Generado por Software	Disponible en todos los pines

El Raspberry Pi Zero usa una tarjeta microSD para realizar las funciones de almacenamiento externo. Esta es equivalente al disco duro HDD de una computadora, y almacena el sistema operativo con el cual funcionará el dispositivo, además de ofrecer espacio para guardar archivos y aplicaciones.

Cuando el Raspberry Pi Zero es encendido, este ejecuta un código especializado llamado bootloader, el cual leerá el sistema operativo que se encuentra en la tarjeta para iniciar el sistema. Si la tarjeta microSD no está presente, el Raspberry Pi no realizará ninguna operación. El tamaño recomendado para la tarjeta microSD es de 8GB.

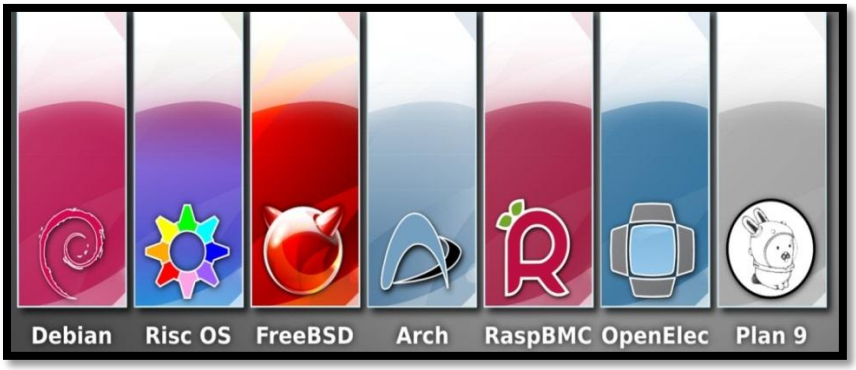
El Raspberry Pi es capaz de usar un sistema operativo Linux para administrar los componentes de hardware que posee. La fundación de Raspberry Pi soporta de manera oficial las distribuciones Raspbian, Ubuntu MATE, Windows 10 IoT y RISC-OS.

Se la elegido la distribución Raspbian para Raspberry Pi en esta aplicación pues es el sistema operativo recomendado por la fundación de Raspberry Pi.

Raspbian es un sistema operativo libre, está basado en Debian y es optimizado para funcionar con el hardware del Raspberry Pi. Incluye drivers, aplicaciones y herramientas que facilitan el uso del sistema operativo.

La programación del comportamiento del Raspberry Pi se realizará en el lenguaje Python, el cual se encuentra instalado por defecto en la distribución Raspbian.

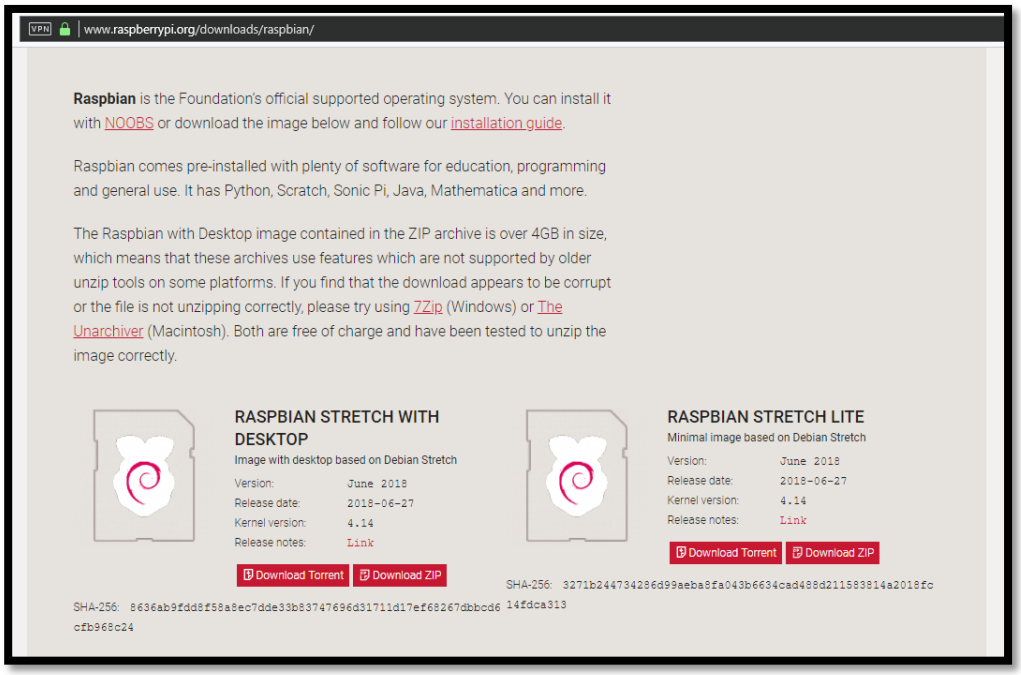
FIGURA Nº4. 13
DISTRIBUCIONES DE SISTEMAS OPERATIVOS PARA RASPBERRY PI



Para instalar el sistema operativo Raspbian en una tarjeta microSD para el Raspberry Pi, se necesita otra computadora con un lector de tarjetas.

Primero se descarga la imagen del sistema operativo desde la página oficial en la dirección web https://www.raspberrypi.org/downloads/raspbian/

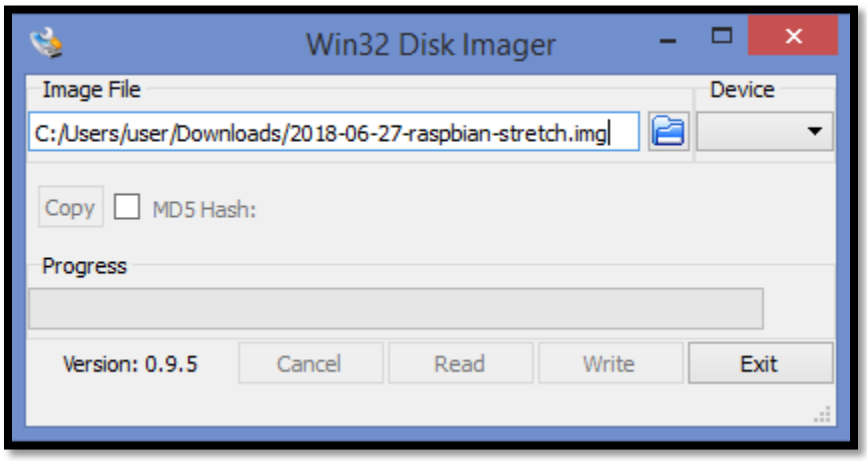
FIGURA Nº4. 14
DESCARGA DEL SISTEMA OPERATIVO RASPBIAN



En una computadora con sistema operativo Windows, se puede usar la aplicación libre Win32DiskImager que permite escribir datos en tarjetas SD, dispositivos USB, etc. La aplicación se puede descargar desde la dirección web https://sourceforge.net/projects/win32diskimager/

Dentro de la aplicación, se debe seleccionar la imagen del sistema operativo y la tarjeta microSD donde será instalada y darle clic al botón de escritura.

FIGURA Nº4. 15
ESCRITURA DE LA IMAGEN EN LA TARJETA MICROSD



❖ **Transceptor de Radio**

El transceptor de radio es un dispositivo compuesto por un transmisor y un receptor combinados en un solo circuito. Este dispositivo permite la transmisión y recepción inalámbrica de datos entre nodos. El transceptor de radio operará en la banda ISM, la cual está reservada para usos no comerciales. El transceptor es el componente que consume la mayor cantidad de energía dentro del nodo. Debido a que los nodos deben operar alimentados de una batería, se requiere un dispositivo que tenga el menor consumo de potencia posible pero aún ser capaz de mantener un enlace funcional y seguro para el funcionamiento de la red.

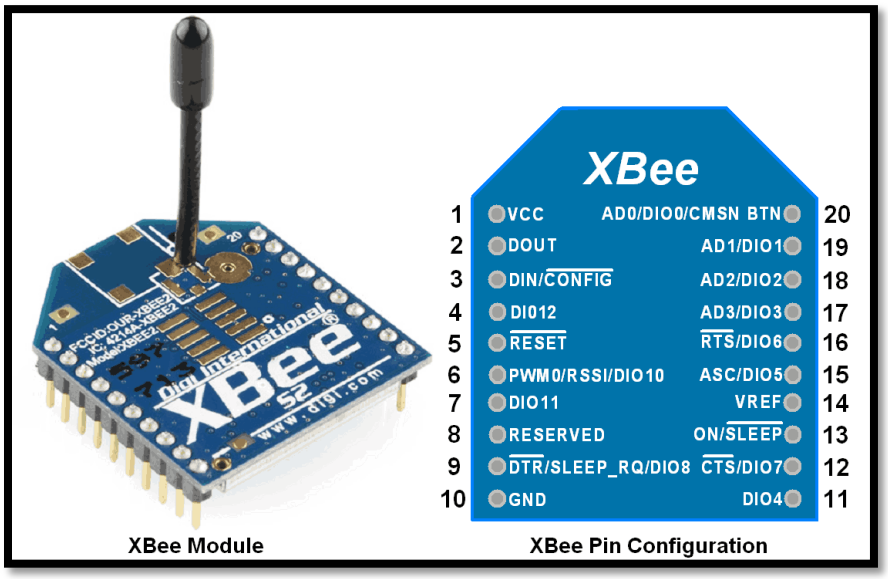
Se ha elegido el transceptor Xbee Zigbee S2C del fabricante Digi, este es un módulo RF embebido que proveen conectividad inalámbrica entre dispositivos en la banda de 2.4GHz usando el protocolo Zigbee y son ideales para aplicaciones de bajo costo y baja potencia.

CUADRO Nº 11
CARACTERÍSTICAS DEL MÓDULO XBEE ZIGBEE S2C

Módulo Xbee Zigbee S2C	
Costo	$30
Chipset	Silicon Labs EM357 SoC
Tasa de transmisión	250kbps
Alcance (con línea de vista)	1200m
Potencia de transmisión	3.1mW (+5dBm)
	6.3mW (+8dBm) modo boost
Sensibilidad de recepción	-100dBm
Banda de frecuencia	2.4GHz
Técnica de modulación	DSSS (Direct Sequence Spread Spectrum)
Canales disponibles	16 canales
Cifrado	128-bit AES
Interfaces	UART, SPI
Voltaje de operación	2.1 - 3.6V
Consumo de corriente en Transmisión	33mA
Consumo de corriente en Recepción	28mA
Consumo de corriente en Hibernación	<1uA

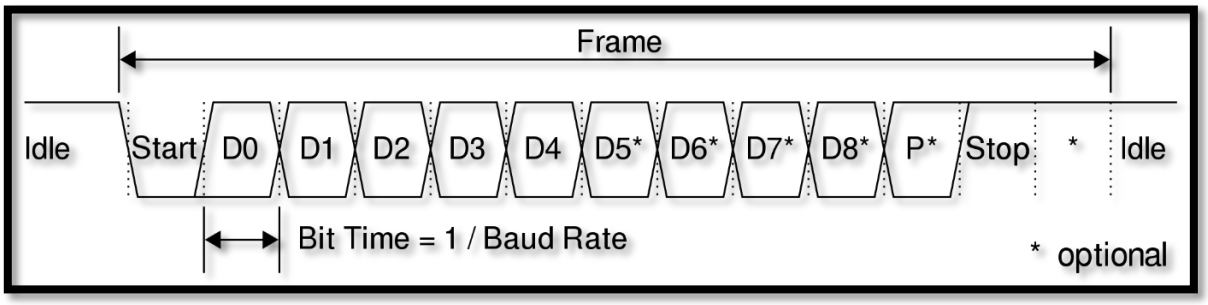
Se debe notar que los módulos Xbee incluyen un microcontrolador embebido que posee pines digitales, y entradas analógicas, sin embargo, su capacidad de procesamiento es muy limitada y no será usada en el diseño de esta aplicación.

FIGURA Nº4. 16
MÓDULO XBEE Y ADAPTADOR USB



El módulo Xbee se puede comunicar con otros dispositivos host a través de su interfaz serial UART. Para esta comunicación, los dispositivos deben tener los mismos parámetros de velocidad de baudios (baudrate), paridad, bit de inicio, bit de parada y cantidad de bits a transmitir. El módulo Xbee usa los pines DOUT (Transmisor) y DIN (Receptor) para esta comunicación. Cada transmisión comienza con un bit de inicio (estado de 0V), seguido por los 8 bits de datos (bit menos significativo primero), y termina con un bit de parada (estado de 3.3V).

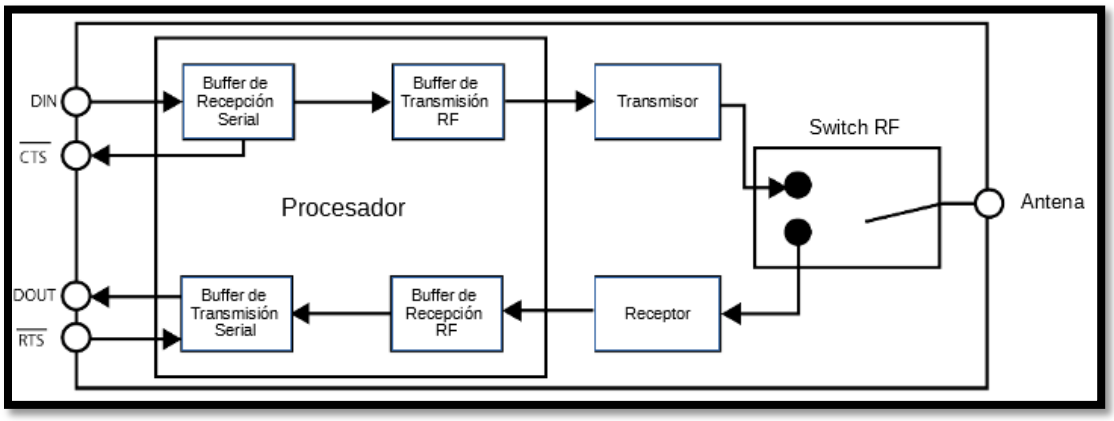
FIGURA Nº4. 17
TRANSMISIÓN SERIAL POR UART



El módulo Xbee contiene unos buffers internos de recepción para recolectar la data serial y la data RF que recibe. La data serial que envía el host es almacenada en el buffer de recepción serial hasta que el procesador interno pueda procesarla y moverla al buffer de transmisión RF. Si el dispositivo recibe datos cuando el buffer serial está lleno, la nueva data recibida será descartada. Esto puede suceder si la data serial llega al dispositivo más rápido de lo que puede transmitirla por su interfaz inalámbrica.

De manera similar, el módulo posee un buffer de recepción RF que almacena la data RF y esta es llevada al buffer de transmisión serial, donde la data recibida es comunicada al host. Si este buffer de recepción RF está lleno, descartará cualquier paquete nuevo que reciba. Esto sucede si el dispositivo recibe datos más rápido de lo que puede transmitirlos a través de la interfaz serial.

FIGURA Nº4. 18
BUFFERS DE RECEPCIÓN Y TRANSMISIÓN DEL MÓDULO XBEE



Según como el módulo Xbee se comunica con el host, este soporta dos modos de operación, el modo transparente y el de interfaz de programación de aplicación (API).

- El modo de operación transparente, permite que los dispositivos inalámbricos actúen como una línea serial. El módulo Xbee almacenará la información que recibe de su interfaz UART y la envía de manera inalámbrica. Cuando la data RF es recibida, esta es enviada a través del puerto serial del receptor.
- El modo de operación API, es una alternativa al modo transparente y este extiende la capacidad del host de interactuar con las funciones de red del dispositivo Xbee. En el modo API, se puede configurar dispositivos y definir rutas de los datos al nivel de capa de aplicación. El host puede enviar paquetes que contienen la dirección y los datos a enviar al dispositivo final. Esto facilita la transmisión de datos a múltiples

destinos, identificar la dirección fuente de cada paquete recibido y revisar el estatus de cada paquete transmitido.

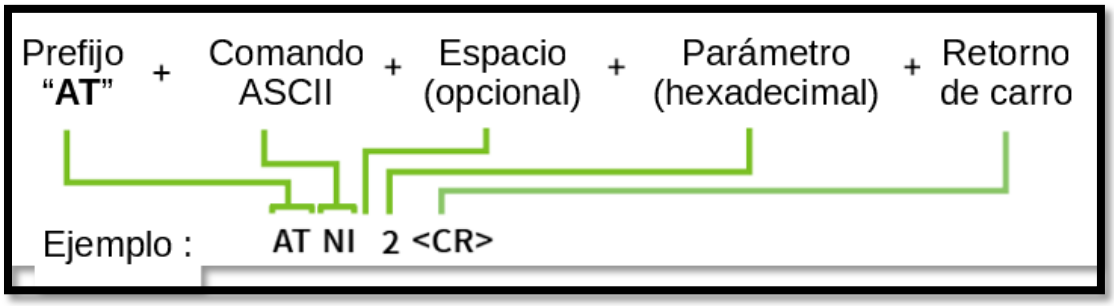
Según qué condiciones externas se cumplan, el módulo Xbee puede estar en modo de recepción, modo de transmisión, modo de hibernación y modo de comandos AT.

- El modo de recepción, es el modo por defecto del módulo Xbee, el dispositivo permanece en modo de recepción hasta que el buffer de recepción serial recibe data para ser transmitida.
- El modo de transmisión, cuando la data esta lista para ser transmitida, el módulo se asegura que la dirección destino y la ruta hayan sido establecidas. Cuando la data es transmitida de un nodo a otro, el nodo destino transmite un mensaje de acuse de recibo hacia el nodo fuente, para indicar que la transmisión fue exitosa, de lo contrario, el nodo fuente reenviará la data.
- El modo de hibernación, permite que el módulo reduzca su consumo de energía cuando no está en uso. El módulo Xbee soporta hibernación activada por pin (donde una señal de voltaje le indica al dispositivo cuando iniciar el modo hibernación) y la hibernación cíclica (donde el dispositivo hiberna por un intervalo de tiempo). En este modo, el módulo es incapaz de enviar o recibir datos.
- El modo de comandos, hace que el dispositivo interprete los datos de la interfaz serial UART como comandos de configuración. Para leer o

configurar algún parámetro del módulo Xbee usando este modo, se debe enviar un comando AT.

- Los comandos AT fueron desarrollados por Dennis Hayes en 1981 con el objetivo de controlar las funciones de módems. AT es una abreviación de la palabra “atención”, cada comando es una cadena de texto que incluye el prefijo AT que indica que función debe cumplir.

FIGURA Nº4. 19
FORMATO DE COMANDOS AT



Cuando se envía un comando AT para configurar parámetros, el dispositivo responderá con el mensaje *OK\r* si la operación fue exitosa, o *ERROR\r* si no lo fue.

Cuando se leen parámetros con comandos AT, el dispositivo responderá con el parámetro deseado en lugar del mensaje *OK\r*.

Para esta aplicación, se usarán los siguientes comandos AT para los módulos Xbee.

CUADRO Nº 12
COMANDOS AT PARA CONFIGURAR EL MÓDULO XBEE

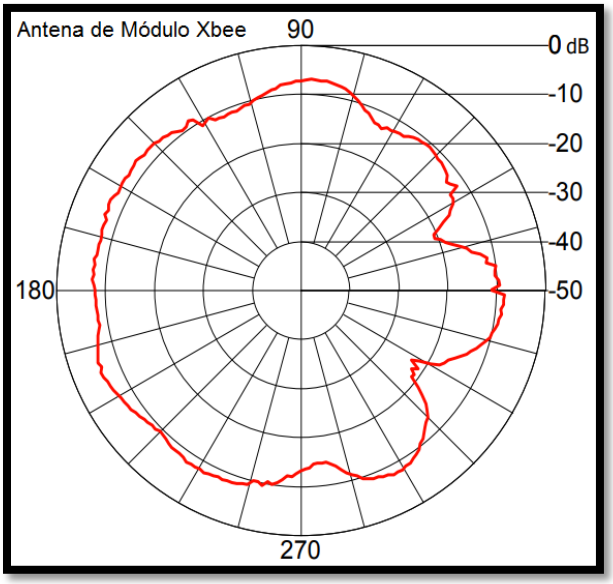
Comando AT	**Nombre y Descripción**	**Rango de Parámetros**	**Ejemplo (pie de página)**
+++	Ingreso al modo AT: Usar este comando y esperar 3 segundos para que el dispositivo entre al modo de comandos AT.	Ninguno	**+++***OK*
AT	Atención: Al escribir este comando, el dispositivo Xbee debe responder **OK**	Ninguno	**AT** *OK*
ATDH	Dirección Destino HIGH: Lee/Escribe los primeros 32bits de la dirección destino de 64bits. La dirección es representada en hexadecimal.	0 – 0xFFFFFFFF	**ATDH 13A200** *OK* **ATDH** *13A200*
ATDL	Dirección Destino LOW: Lee/Escribe los últimos 32bits de la dirección destino de 64bits. La dirección es representada en hexadecimal.	0 – 0xFFFFFFFF	**ATDL 2018** *OK* **ATDL** *2018*
ATSH	Número Serial HIGH: Lee los primeros 32bits de la dirección única del módulo. La dirección es representada en hexadecimal.	0 – 0xFFFFFFFF [solo lectura]	**ATSH** *13A200*
ATSL	Número Serial LOW: Lee los últimos	0 – 0xFFFFFFFF	**ATSL**

	32bits de la dirección única del módulo. La dirección es representada en hexadecimal.	[solo lectura]	*98E480A7*
ATID	Identificador de PAN: Lee/Escribe el identificador de 64bits de la red de área personal (PAN). La dirección es representada en hexadecimal.	0 – 0xFFFFFFFFFFFFFFFF	**ATID** **EF15F0875BE179E8** *OK* **ATID** *EF15F0875BE179E8*
ATCE	Habilitación de modo coordinador: Lee/Selecciona si el módulo operará como coordinador de la red.	0 = no coordinador 1 = coordinador	**ATCE 1** *OK* **ATCE** *1*
ATJV	Verificación de canal: Permite que un dispositivo router o final, verifique que el coordinador esté en el mismo canal de operación cuando se une a la red.	0 = desactiva verificación de canal. 1 = activa verificación de canal.	**ATJV 1** *OK* **ATJV** *1*
ATPL	Nivel de Potencia: Lee/Selecciona el nivel de potencia con el cual el módulo RF transmite.	0 = -5dbm 1 = -1dbm 2 = +1dbm 3 = +3dbm 4 = +5dbm	**ATPL 4** *OK* **ATPL** *4*
ATPM	Modo de Potencia: Lee/Selecciona el modo Boost de potencia. En el modo Boost, la potencia de transmisión aumenta 3dB y mejora	0 = modo Boost desactivado 1 = modo Boost activado	**ATPM 1** *OK* **ATPM**

	la sensibilidad de recepción en 2dB.		*1*
ATSM	Modo Hibernación: Configura el modo de hibernación del módulo.	0 = hibernación desactivada 1 = hibernación por pin activada 4 = hibernación cíclica activada 5 = hibernación cíclica por pin activada	**ATSM 1** *OK*
ATRE	Restaurar configuración por defecto: Usar este comando para restaurar los parámetros del módulo definidos por fábrica.	Ninguno	**ATRE** *OK*
ATAC	Aplicar cambios: Usar este comando para aplicar los cambios realizados con otros comandos AT.	Ninguno	**ATAC** *OK*
ATWR	Escritura: Usar este comando para escribir los parámetros actuales en la memoria no volátil. No se deben enviar datos al dispositivo hasta que este responda con *OK*.	Ninguno	**ATWR** *OK*
ATCN	Salir de modo AT: Usar este comando para terminar el modo de comandos AT del dispositivo y reanudar operación de transmisión y recepción.	Ninguno	**ATCN**
ATFR	Reinicio por software: Usar este comando para reiniciar el módulo. Este responderá con OK y se reiniciará después de 2 segundos.	Ninguno	**ATFR** *OK*

La antena incluida en el módulo Xbee es un monopolo tipo A24-QI de baja ganancia y tiene el siguiente patrón de radiación.

FIGURA Nº4. 20
PATRÓN DE RADIACIÓN DE LA ANTENA EN MÓDULO XBEE



Usando la ecuación de transmisión de Friis y los datos técnicos de los radios transceptores podemos determinar cuál es la distancia máxima en la cual dos módulos Xbee pueden entablar un enlace de red.

$$P_{RX}(dB) = P_{TX}(dB) + G_{TX}(dB) + G_{RX}(dB) + 20\log\left(\frac{c}{4\pi \times f \times d}\right)$$

Sabemos que la mínima sensibilidad de recepción es de -100dBm, la potencia de transmisión es de +5dBm y la ganancia de las antenas es de aproximadamente 1.5dB en la banda 2.4GHz. Entonces reemplazando en la fórmula de Friis.

$$-100 = 5 + (1.5) + (1.5) + 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times d}\right)$$

$$-108 = 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times d}\right)$$

$$-5.4 = \log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times d}\right)$$

$$3.981 \times 10^{-6} = \frac{9.9403 \times 10^{-3}}{d}$$

$$d = 2496.89m$$

Entonces concluimos que la distancia máxima teórica entre dos nodos en la red no será mayor a 2496.89m. Este dato será necesario en el plan de distribución de nodos de esta investigación.

En esta aplicación, el módulo Xbee se comunica con el procesador Raspberry Pi a través de la interfaz UART a 9600 baudios, usando comandos AT para su configuración y enviando los datos en modo de operación trasparente para comunicarse con el nodo sumidero.

Para identificar los nodos de manera sencilla, se leerá y usará las direcciones de 64bits.

El módulo Xbee tiene un consumo de corriente relativamente alto y no es necesario que se encuentre activo durante todo el ciclo de operación del nodo, por lo tanto se mantendrá en modo hibernación durante la mayoría de tiempo y solo será usado durante un corto lapso de tiempo para transmitir la data de los sensores. Este modo de hibernación se aplica para los dispositivos fuente, ya que estos son energizados por una batería, y porque

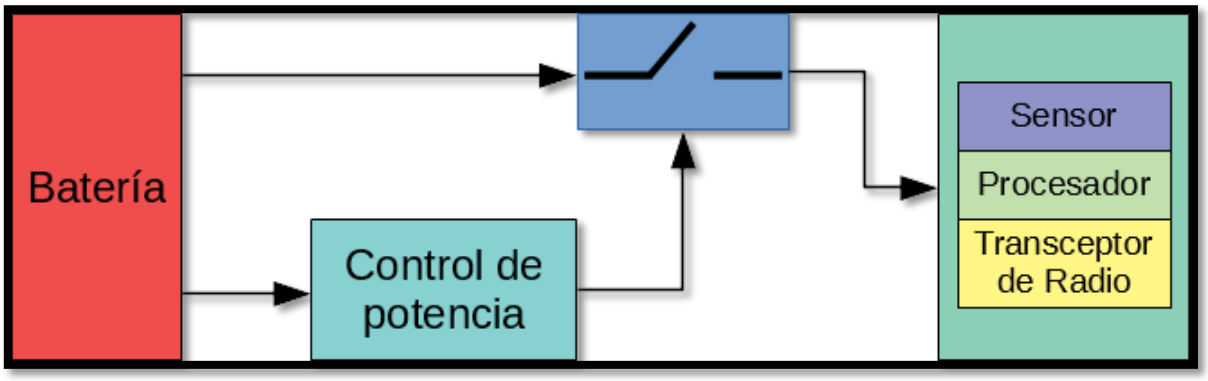
según el protocolo Zigbee, el coordinador debe permanecer activo y energizado desde una fuente permanente.

❖ **Etapa de Alimentación**

Las baterías son la principal fuente de energía para el nodo en la red de sensores. Esta debe tener la capacidad suficiente para mantener en funcionamiento el sistema sin necesidad de supervisión o mantenimiento.

Para reducir el consumo de potencia del nodo y alargar el tiempo de vida de la batería y la red, se introducirá un circuito de control de potencia, que permita encender y apagar los componentes del nodo de manera periódica para reducir el consumo innecesario de energía cuando el nodo no esté en uso. Esta es la misma técnica que usan otros dispositivos cuando entran en modo de hibernación.

FIGURA Nº4. 21
DIAGRAMA DE LA ETAPA DE ALIMENTACIÓN



Los elementos de mayor consumo de energía en el nodo son el procesador, sensor y el módulo transceptor de radio, y estos elementos no estarán activos durante todo el ciclo de vida del nodo, sino que solo serán

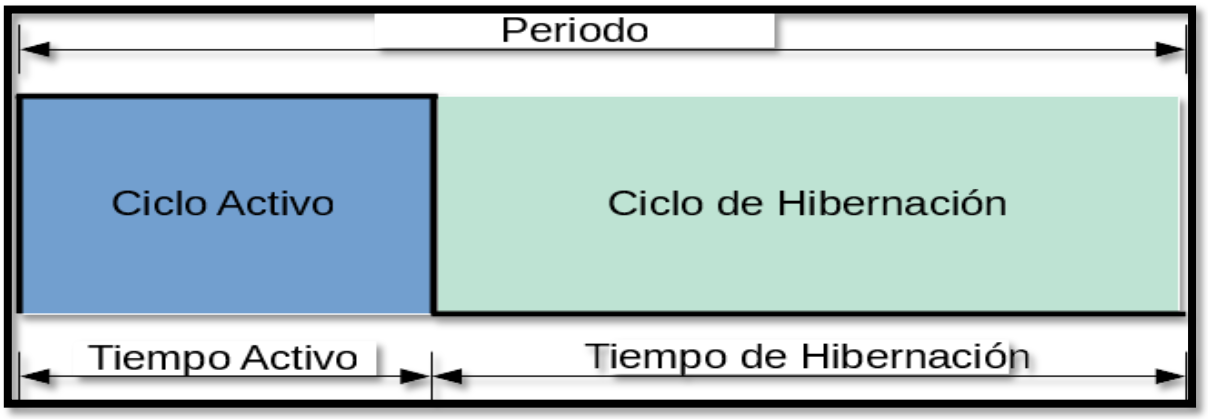
encendidos cuando sea necesario realizar las mediciones de campo electromagnético y la transmisión de la data adquirida al nodo sumidero.

Se ha elegido la tarjeta Raspberry Pi Zero para cumplir el rol de procesador principal en el nodo, sin embargo, esta tarjeta no posee el hardware necesario para realizar operaciones de encendido o apagado automático, como lo hacen otras computadoras. Es necesario agregar otro controlador de baja potencia para realizar estas operaciones.

Este controlador de potencia podrá encender los componentes relevantes del nodo, permitiendo la operación principal, que consiste en la medición de campo electromagnético y finalizada dicha operación, puede remover la energía del procesador, sensor y transceptor de radio, haciendo que el nodo entre en su propio modo de hibernación, para ahorrar energía y extender el tiempo de vida de la batería.

El controlador de potencia usará un temporizador para volver a encender los componentes del nodo, de manera que sus operaciones sean resumidas de manera cíclica.

FIGURA Nº4. 22
DIAGRAMA DE CICLO DE OPERACIÓN DEL NODO FUENTE



Para el circuito de control de potencia se usará un ATtiny45, un pequeño microcontrolador AVR que puede ser programado con el lenguaje C, en conjunto con un transistor de potencia, el cual actuará como un switch que permitirá encender y apagar los otros componentes del nodo.

CUADRO Nº 13
CARACTERÍSTICAS DEL MICROCONTROLADOR ATtiny45

ATtiny45	
Costo	$2
Arquitectura	AVR 8bits
Memoria RAM	256B
Memoria FLASH	4kB
Frecuencia de reloj	0 – 20MHz
Interfaces	UART, SPI, I^2C, GPIO
Voltaje de operación	2.7V – 5.5V
Consumo de corriente (estado idle)	0.35mA
Consumo de corriente (activo)	1.5mA
Consumo de corriente (modo	<10uA

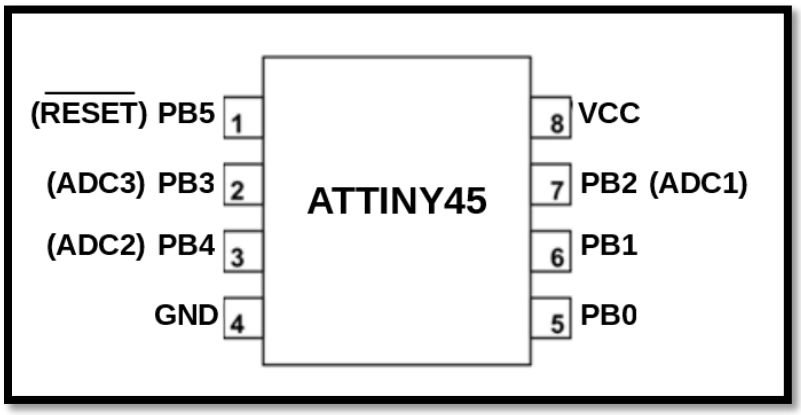
hibernación de baja potencia)	

El ATtiny45 es un microcontrolador CMOS de baja potencia que usa una arquitectura Harvard de 8bits y ejecuta 1 instrucción por cada ciclo de reloj. Este microcontrolador cuenta con periféricos como:

- Temporizadores de 8bits con dos canales de señales de PWM.
- Interrupciones Externas.
- Módulo de Interfaz Serial Universal.
- Conversor Analógico-Digital (ADC) de 10bits.
- Comparador analógico.
- Temporizador de perro guardián (watchdog).

La señal de reloj que usa este microcontrolador puede provenir de una fuente de reloj externa, del oscilador RC interno (8MHz), el oscilador interno de baja potencia (128kHz) o de un cristal oscilador externo (hasta 20MHz). Por defecto, el ATtiny45 usa el oscilador RC interno de 8MHz con un divisor de frecuencia de 8.

FIGURA Nº4. 23
MICROCONTROLADOR ATtiny45



Para aplicaciones que requieren bajo consumo de energía, el ATtiny45 soporta 3 tipos de hibernación, cada uno restringe ciertos componentes del microcontrolador para ahorrar energía.

- Hibernación idle, detiene el funcionamiento del CPU y la memoria FLASH y admite el funcionamiento de todos los otros periféricos. En el modo Idle, el microcontrolador puede despertar de la hibernación por fuentes externas, como interrupciones en los pines, o internas, como los temporizadores.
- Hibernación de reducción de ruido del ADC, detiene el funcionamiento del CPU, la memoria FLASH y los pines del microcontrolador. Este modo es usado para mejorar la calidad de las lecturas del conversor analógico-digital. El microcontrolador solo puede despertar de este modo usando una señal de RESET externa, RESET por el perro guardián, RESET por el detector de bajo voltaje (brownout) o una interrupción externa.

- Hibernación de baja potencia, detiene el funcionamiento de todos los periféricos y solo es posible despertar al dispositivo usando una señal de RESET externa, RESET por el perro guardián, RESET por el detector de bajo voltaje (brownout) o una interrupción externa.

La señal de RESET externa se ejecuta cuando un nivel de 0V está presente en el pin RESET (PB5) del microcontrolador.

El RESET por el perro guardián ocurre cuando el temporizador de perro guardián, que opera con un reloj interno de 128kHz, termina su cuenta.

El RESET por el detector de bajo voltaje o brownout, ocurre cuando el voltaje de alimentación del microcontrolador cae por debajo de un nivel umbral programable.

Una interrupción externa ocurre cuando el microcontrolador detecta un flanco de subida o bajada en alguno de sus pines.

El ATtiny45 cuenta con un conversor analógico-digital (ADC) de 10bits, con 4 canales multiplexados en los pines PB2, PB3, PB4 y PB5 y una frecuencia de muestreo de hasta 200kHz.

También cuenta con un comparador analógico con entradas en los pines PB0 y PB1. Cuando el voltaje en el pin PB0 es mayor al voltaje en PB1, el comparador genera una señal de interrupción.

El ATtiny45 cuenta con un temporizador de 8bits de propósito general (TIMER0) con la capacidad de general señales PWM. Este temporizador puede usar la señal de reloj interna del microcontrolador, o una fuente de reloj externa en el pin PB2.

Cuando el contador ascendente del TIMER0 llega a su valor máximo de 255, la cuenta reiniciará desde 0. Este evento genera una interrupción.

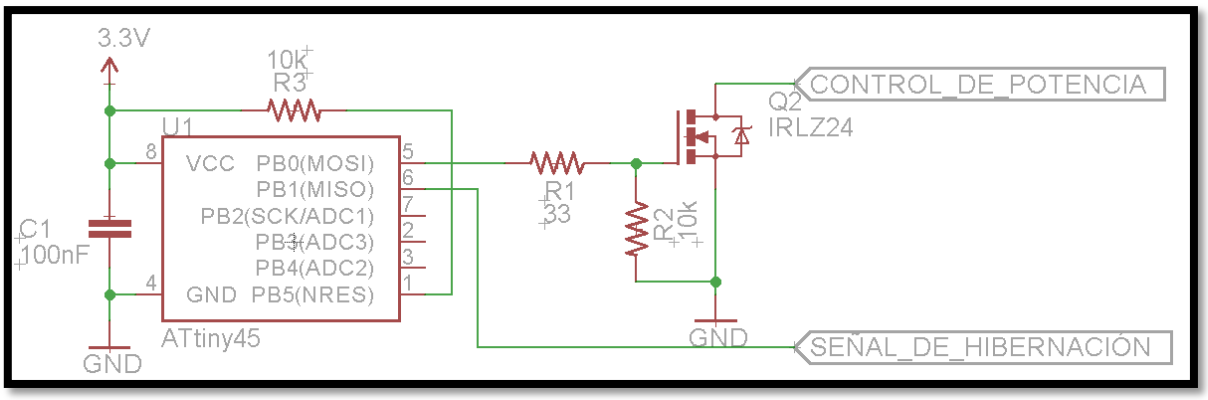
El ATtiny45 usa un transistor como switch para controlar la energía enviada a los otros elementos del nodo. Para esta investigación se usará el MOSFET IRLZ24, un transistor de efecto de campo capaz de manejar cargas de mediana potencia, tiene una baja resistencia de encendido y puede ser activado con bajos niveles de voltaje.

CUADRO Nº 14
CARACTERÍSTICAS DEL MOSFET IRLZ24

IRLZ24	
Costo	$2
Voltaje Drenador-Fuente (VDS)	Hasta 60V
Voltaje Compuerta-Fuente (VGS)	5V
Resistencia de Encendido (RDS)	0.1Ω
Corriente Continua (ID)	Hasta 12ª
Máxima Potencia Disipada	60W

El esquema de conexión para la etapa de alimentación será la siguiente:

FIGURA Nº4. 24
ESQUEMA DEL CIRCUITO DE CONTROL DE POTENCIA



El ATtiny45 también usa una entrada digital en el pin PB1, conectada al procesador Raspberry Pi Zero, que le indicará cuándo entrar en modo hibernación.

Debemos notar que este circuito de control de potencia solo es necesario en el nodo fuente, debido a que según el protocolo Zigbee, el nodo sumidero, que actúa como coordinador de la red, debe tener una fuente de alimentación permanente y por lo tanto no necesita ingresar en modo hibernación.

Cuando el Raspberry Pi está en modo hibernación, el ATtiny45 también entrará en hibernación de baja potencia, para ayudar a reducir el consumo de corriente del nodo. Se usará una interrupción del temporizador de perro guardián (watchdog) para despertar al microcontrolador e iniciar el siguiente ciclo activo.

- **Batería**

La limitación más significante en el funcionamiento de una red de sensores es operar con una reserva limitada de energía. De manera general, los nodos de una red de sensores son energizados por baterías. Cuando un nodo descarga su batería, este queda descartado de la red.

Los nodos deben operar de manera autónoma y sin supervisión, por lo tanto estos deben usar técnicas de ahorro de energía para mantener operativa la red durante el mayor tiempo posible.

En esta investigación, el funcionamiento del nodo fuente incluye una etapa de hibernación. La función principal del nodo es medir los niveles de radiación no ionizante, esta tarea será realizada una vez al día durante aproximadamente 6 minutos. El resto del tiempo, el nodo permanecerá en modo hibernación para alargar el tiempo de vida de la batería y la red.

En la actualidad, la mayoría de dispositivos electrónicos portátiles (teléfonos móviles, laptops, tablets, cámaras) usan baterías recargables de Ión de litio (Li-Ion).

El desarrollo de circuitos dedicados a el monitoreo de este tipo de baterías y el avance en los procesos de producción de las baterías, hacen de estas, una opción viable para esta aplicación.

Para esta investigación se ha elegido un banco de baterías comercial (power-bank) para energizar los componentes del nodo.

Un power-bank, a veces denominado batería externa, es un arreglo de baterías de Li-Ion conectados a un circuito con un microcontrolador, regulador de voltaje y controlador de carga, con protección de sobrecorriente y cortocircuito. Este dispositivo ofrece salidas en forma de conectores USB que son generalmente usados para recargar dispositivos electrónicos como teléfonos móviles, tablets, etc.

La ventaja de usar el power-bank es su disponibilidad, pues puede ser adquirido en cualquier centro comercial, su portabilidad y su facilidad de uso para esta aplicación, ya que este nos da una fuente de 5V necesario para energizar los componentes del nodo.

El precio de un power-bank está determinado por su capacidad medida en mAh, para determinar cuál es la capacidad necesaria para energizar un nodo fuente por un largo periodo de tiempo, debemos realizar el cálculo de consumo de corriente de los dispositivos del nodo en su ciclo de operación.

- **Cálculo de consumo de corriente**

Generalmente, la duración de una batería conectada a una carga se calcula basado en el consumo de corriente de dicha carga, en miliamperios (mA), y la capacidad de la batería en miliamperios-hora (mAh), de manera que una carga con bajo consumo de corriente tendrá un largo tiempo de vida.

Para realizar el cálculo usaremos la siguiente fórmula:

$$t = \frac{Q \times 0.8}{i}$$

Dónde:

t es la duración estimada de la batería, en horas. (h)

Q es la capacidad de la batería, en miliamperio-horas. (mAh)

i es el consumo de corriente de la carga, en miliamperios. (mA)

El factor de 0.8 es usado para simular pérdidas de factores externos como temperatura, resistencias internas, corrientes de fuga, etc.

Del diagrama de ciclo de operación del nodo (Véase la Figura N°4.22 en la página 86), concluimos que el consumo de corriente no es constante debido a que en cada etapa del funcionamiento del nodo, algunos componentes son activados y/o desactivados para reducir el consumo de energía.

Para aplicar la fórmula de cálculo de duración de batería, debemos hallar el consumo de corriente promedio en un ciclo de operación del nodo fuente.

Para esto usaremos la siguiente fórmula:

$$i_{promedio} = \frac{i_{ciclo\ activo} \times t_{ciclo\ activo} + i_{ciclo\ hibernación} \times t_{ciclo\ hibernación}}{t_{ciclo\ activo} + t_{ciclo\ hibernación}}$$

Dónde:

$$t_{ciclo\ hibernación} = t_{periodo\ de\ operación} - t_{ciclo\ activo}$$

El periodo de operación en esta aplicación será de 24horas, en las cuales los componentes más importantes del nodo fuente se encontrarán activos durante una fracción de ese periodo para realizar las funciones de la aplicación.

Durante el ciclo activo, todos los componentes del nodo están activados, sin embargo, no todos los componentes están funcionando a su máximo potencial.

El módulo RTL por ejemplo consume 70mA cuando está conectado, pero su consumo aumenta a 100mA cuando está siendo usado por alguna aplicación.

De igual manera el módulo Xbee, consumirá más potencia cuando está transmitiendo datos que cuando está estático. Dentro del ciclo activo del nodo fuente, existen diferentes etapas que son detalladas a continuación:

CUADRO Nº 15
ETAPAS DE OPERACIÓN DEL CICLO ACTIVO DEL NODO FUENTE

Etapa	**Descripción**	**T**	**Consumo de corriente**		
			Componente	**mA**	**Total**

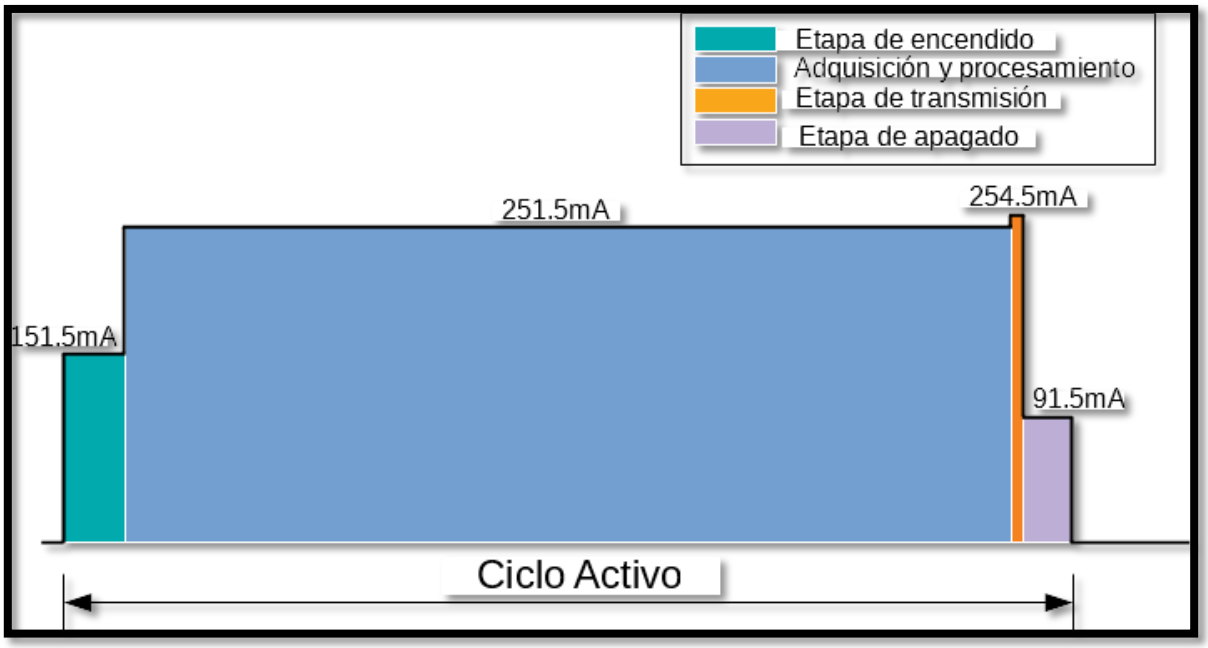
Encendido	El Raspberry Pi inicia su sistema operativo antes de iniciar su programa principal.	25s	ATtiny45	1.5	151.5mA
			Raspberry Pi Zero	80	
			Módulo RTL-SDR	70	
			Módulo Xbee	0.001	
Adquisición y Procesa-miento	Se realiza la lectura del sensor y el cálculo de los niveles de radiación no ionizante.	360s	ATtiny45	1.5	251.5mA
			Raspberry Pi Zero	150	
			Módulo RTL-SDR	100	
			Módulo Xbee	0.001	
Transmisión de datos	El nodo fuente transmite los datos procesados hacia el nodo sumidero.	5s	ATtiny45	1.5	254.5mA
			Raspberry Pi Zero	150	
			Módulo RTL-SDR	70	
			Módulo Xbee	33	
Apagado	El Raspberry Pi termina sus operaciones y apaga el sistema operativo.	10s	ATtiny45	1.5	91.5 mA
			Raspberry Pi Zero	20	
			Módulo RTL-SDR	70	
			Módulo Xbee	0.001	

El consumo del nodo fuente durante el ciclo activo será la suma de consumo durante cada una de sus etapas.

$$i_{ciclo\ activo} \times t_{ciclo\ activo} = 151.5 \times 25 + 251.5 \times 360 + 254.5 \times 2 + 91.5 \times 10$$

$$i_{ciclo\ activo} \times t_{ciclo\ activo} = 95751.5mAs$$

FIGURA Nº4. 25
DIAGRAMA DEL CICLO ACTIVO DEL NODO FUENTE



Durante el ciclo de hibernación, solo se encuentra activo el microcontrolador ATtiny45 y este a su vez se encuentra en modo hibernación de baja potencia. En este modo, el microcontrolador consume menos de 10uA y permanecerá en este modo durante 86000 segundos.

$$i_{ciclo\ hibernación} \times t_{ciclo\ hibernación} = 0.01 \times 86000$$

$$i_{ciclo\ hibernación} \times t_{ciclo\ hibernación} = 860mAs$$

Al juntar los valores calculados en la fórmula anterior tenemos:

$$i_{promedio} = \frac{95751.5mAs + 860mAs}{86400s}$$

$$i_{promedio} = 1.118mA$$

Para esta aplicación se ha elegido el power-bank Xiaomi Mi 2 de 20000mAh. Este usa baterías Li-Ion del fabricante Lishen y los circuitos

integrados fabricados por Texas Instrument. El Xioami Mi incluye circuito de protección de sobrevoltaje en su entrada y sus salidas, posee protección contra cortocircuito, protección de sobrecarga y detector de desconexión.

CUADRO Nº 16
CARACTERÍSTICAS DEL POWER-BANK XIOAMI Mi2

Power-bank Xiaomi Mi 2	
Costo	$35
Capacidad	20000mAh
Tipo de batería	Li-Ion
Dimensiones	135,5 x 67,6 x 23,9 cm
Interfaz de Entrada	Micro USB (5V @ 2A)
Interfaz de Salida	USB tipo A (5.1V @ 3.6A)
Masa	331g

Al aplicar la fórmula de cálculo de duración de batería con el valor de consumo de corriente promedio y la capacidad del Power-bank Xiaomi Mi 2, tendremos:

$$t = \frac{20000mAh}{1.118mA} \times 0.8$$

$$t = 14308.86h$$

El tiempo de vida del nodo fuente será aproximadamente 14308.86 horas o 596.2 días.

- **Interfaz de red LAN**

Esta será la interfaz por la cual un nodo sumidero se conecta a una red LAN. El Raspberry Pi Zero no posee una interfaz de red Ethernet, por lo tanto será necesario usar un adaptador USB-RJ45 para comunicar el dispositivo con una red externa y transmitir los datos recibidos.

Se ha elegido el adaptador USB 2.0 OTG Micro-B to Ethernet del fabricante Plugable. Este adaptador compacto que usa un conector microUSB para conectarse con dispositivos USB OTG como smartphones, tablets, y el Raspberry Pi Zero.

CUADRO Nº 17
CARACTERÍSTICAS DEL ADAPTADOR USB-ETHERNET

Adaptador Plugable USB 2.0 OTG Micro-B to Ethernet	
Costo	$15
Estándares de Red	IEEE 802.3 (10Base-T) IEEE 802.3u (100Bbase-TX)
Interfaz de Red	Conector RJ45 hembra, con soporte de auto MDIX
Voltaje de entrada	5V del bus USB
Chipset	ASIX AX88772

El adaptador está basado en el chip de aplicación específica (ASIC) AX88772 del fabricante ASIX. Este es un controlador especializado y de bajo costo, capaz de establecer una interfaz entre una conexión de red local y un dispositivo externo con puerto USB. Este implementa las funciones de red LAN a velocidades de 10Mbps y 100Mbps basados en los estándares IEEE 802.3 y 802.3u.

El AX88772 opera a 3.3V y usa un reloj de 12MHz para las operaciones del puerto USB y 25MHz para las operaciones de Ethernet. Cuenta con 20kB de memoria SRAM para el buffer de recepción y 8kB de SRAM para el buffer de transmisión.

FIGURA Nº4. 26
ADAPTADOR USB 2.0 OTG Micro-B TO ETHERNET

Los drivers para este controlador ya están incluidos en los sistemas operativos Windows, Mac, Linux (a partir del kernel 2.6.35) y ChromeOS, por lo tanto, este funcionará directamente en el Raspberry Pi Zero, sin necesidad de configuraciones adicionales.

- **Fuente de Voltaje Continuo**

En el diseño de esta aplicación, el nodo sumidero actuará como el coordinador de la red inalámbrica de sensores. Según el protocolo Zigbee, un nodo coordinador debe tener una fuente de alimentación continua, y a

diferencia de los nodos fuente, el coordinador no realizará el ciclo de hibernación, o de ahorro de energía.

El nodo sumidero estará basado en un Raspberry Pi Zero, y necesita una fuente de voltaje de 5V capaz de suministrar al menos 1A. El procesador en el nodo no consumirá más de 200mA, y los otros componentes del nodo no exceden el consumo de 100mA cada uno.

Para este diseño, se ha elegido la fuente de voltaje DR-15-5 del fabricante Mean-Well. Esta fuente soporta un voltaje de entrada AC de rango completo desde 85VAC hasta 264VAC con protecciones de sobrecarga, sobrevoltage y cortocircuito. La fuente DR-15 puede ser montada en un riel DIN TS35 lo que facilita su instalación en un ambiente industrial.

CUADRO Nº 18
CARACTERÍSTICAS DE LA FUENTE DE VOLTAJE DR-15-5

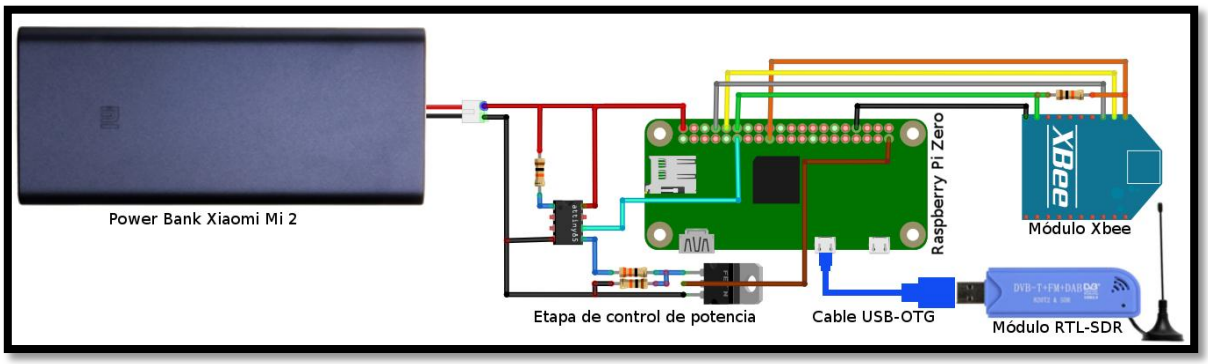
Fuente DR-15-5	
Costo	$13
Voltaje de salida	5V
Rango de voltaje ajustable	4.75 ~ 5.5V
Corriente de salida máxima	2.4A
Potencia	12W
Rango de voltaje de entrada	85 ~ 264VAC
Rango de frecuencia	47 ~ 63Hz
Corriente de entrada	0.48A @ 230VAC
Eficiencia	77%

FIGURA Nº4. 27
FUENTE DE VOLTAJE DR-15-5

❖ **Diagramas de Conexión para los componentes del Nodo Fuente**

El software de diseño electrónico Fritzing permite visualizar las conexiones entre los componentes usados en el diseño del nodo fuente.

FIGURA Nº4. 28
CONEXIONES FÍSICAS DE LOS COMPONENTES EN EL NODO FUENTE



Se ha usado el software de diseño de circuitos Autodesk Eagle para realizar el diseño de circuito de tarjeta impresa (PCB) del nodo fuente.

FIGURA Nº4. 29
ESQUEMA DE CONEXIONES DEL NODO FUENTE

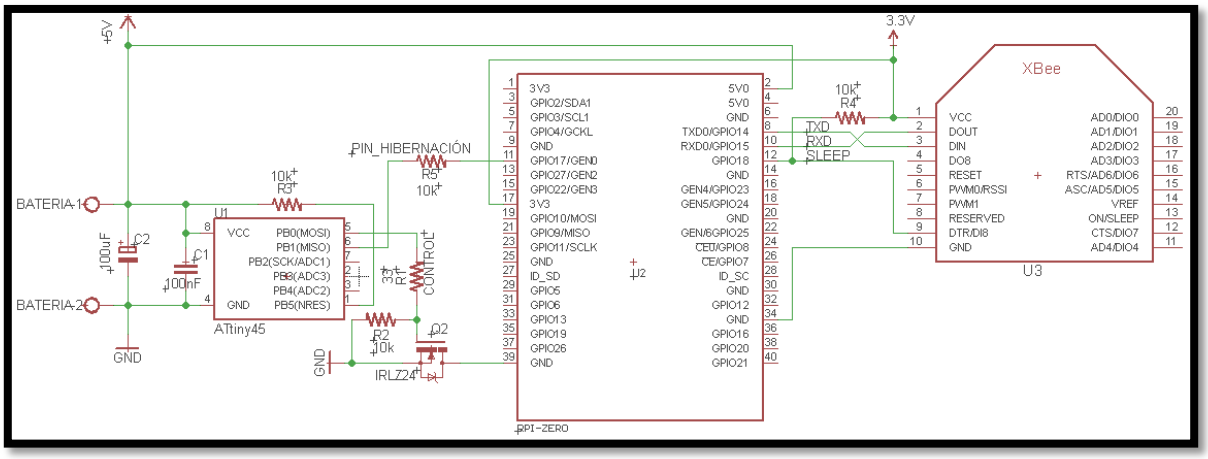
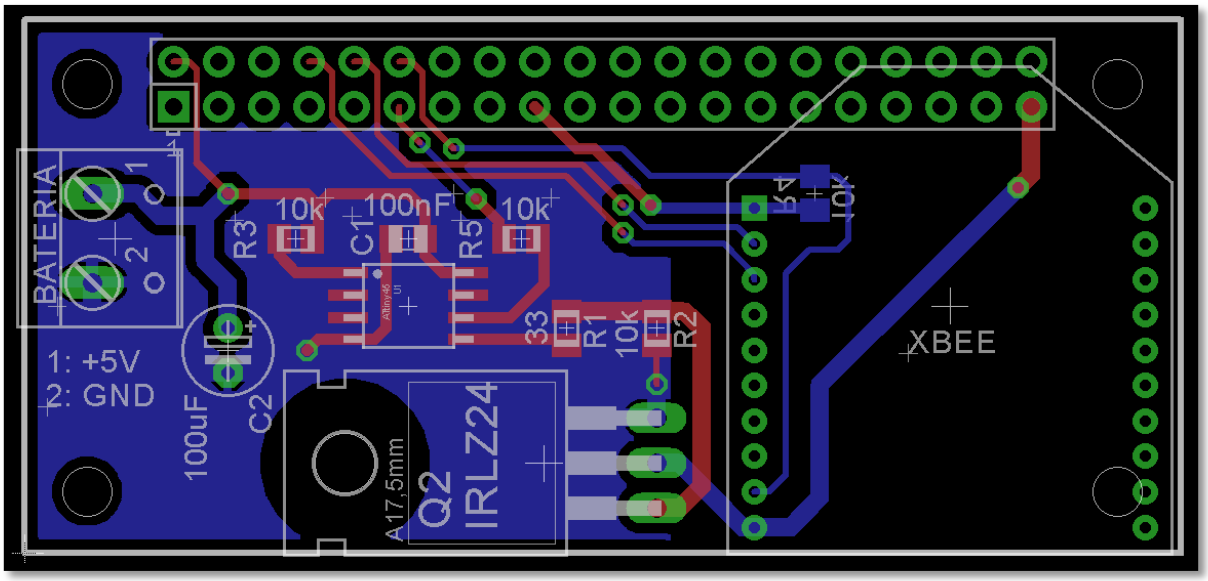


FIGURA Nº4. 30
DISEÑO DE TARJETA IMPRESA DEL NODO FUENTE

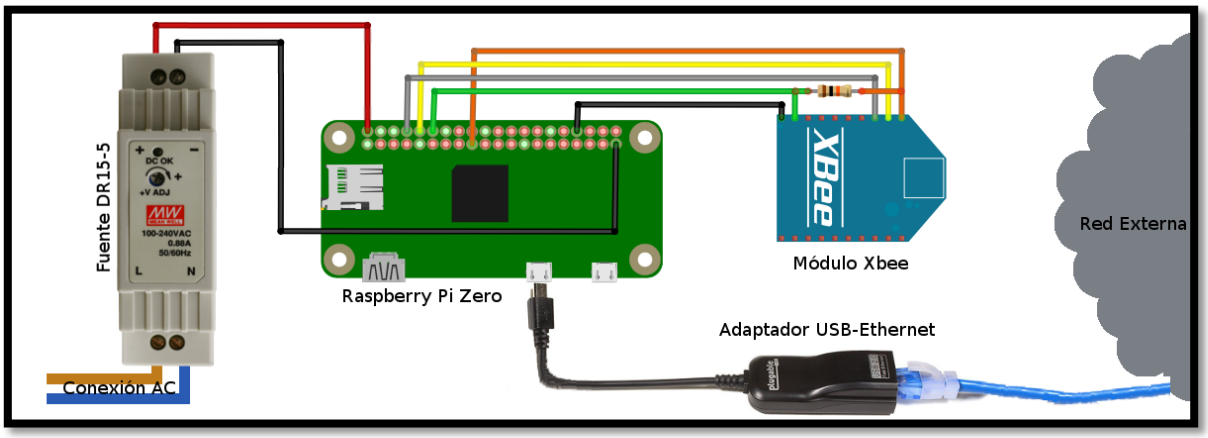


El PCB del nodo fuente tiene la misma forma y las dimensiones que el Raspberry Pi Zero, y con este diseño se pueden conectar los componentes haciendo uso del conector de GPIO.

❖ **Diagramas de Conexión para los componentes del Nodo Sumidero**

El diseño del nodo sumidero es un poco más sencillo, pues no requiere los componentes de la etapa de control de potencia que son necesarios en el nodo fuente, pero se realizará con características similares de tamaño y forma.

FIGURA Nº4. 31
CONEXIONES FÍSICAS DE LOS COMPONENTES EN EL NODO SUMIDERO



Se ha usado el software de diseño de circuitos Autodesk Eagle para realizar el diseño de circuito de tarjeta impresa (PCB) del nodo sumidero.

FIGURA Nº4. 32
ESQUEMA DE CONEXIONES DEL NODO SUMIDERO

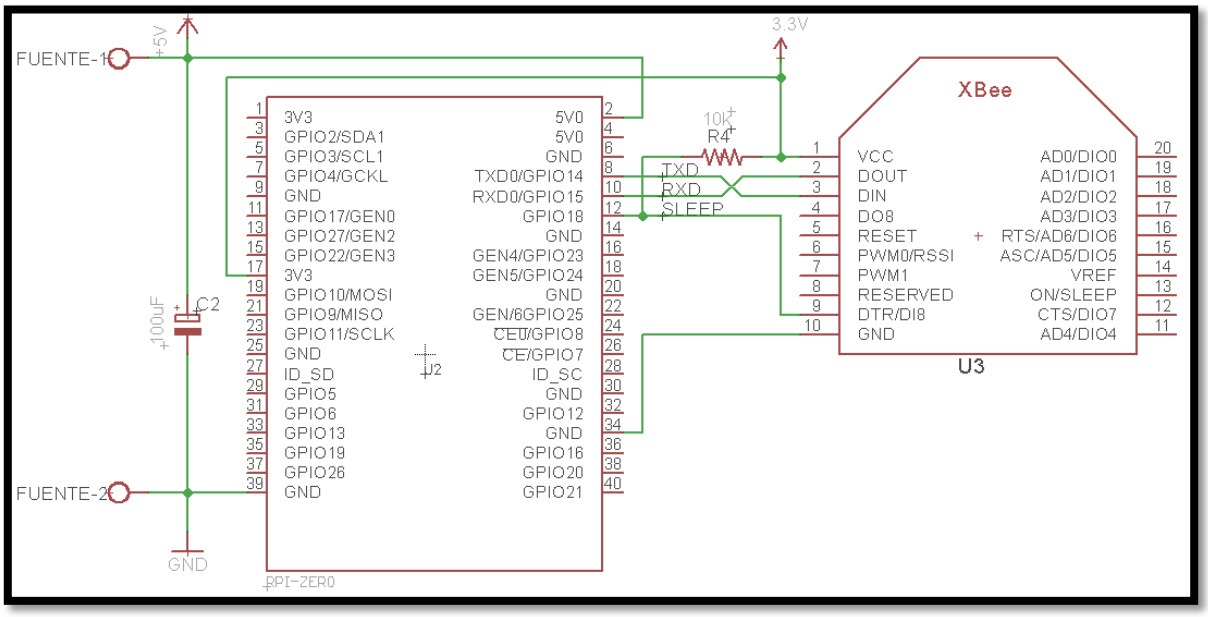
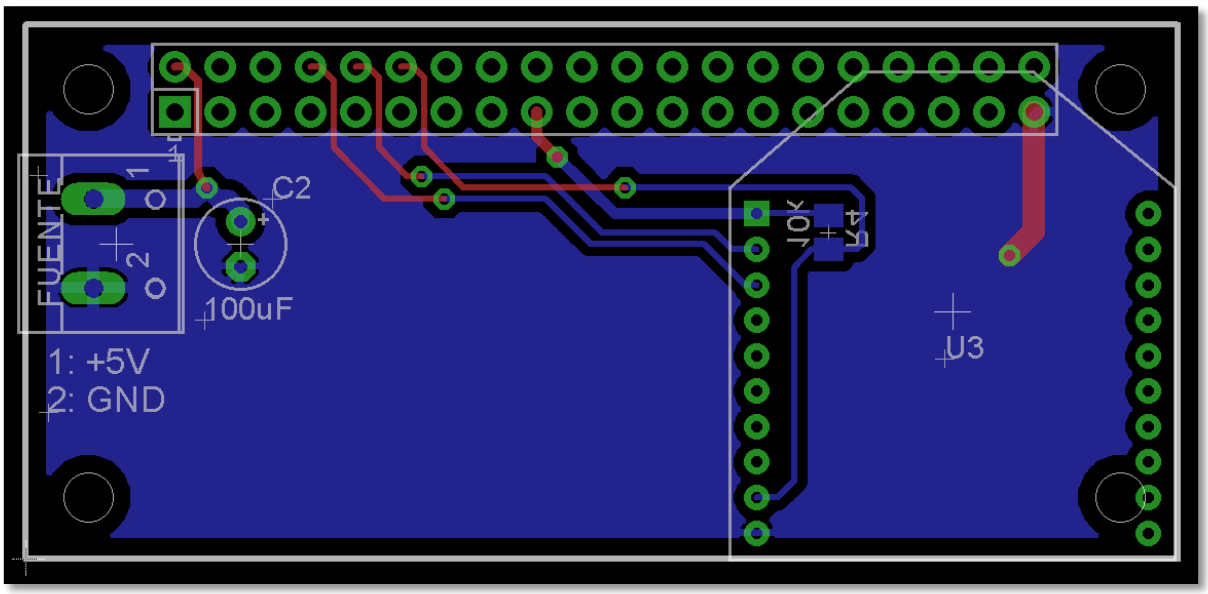


FIGURA Nº4. 33
DISEÑO DE TARJETA IMPRESA DEL NODO SUMIDERO



Debemos notar que en ambos casos, los dispositivos que se comunican a través de la interfaz USB-OTG del Raspberry Pi no aparecen en el diseño. Estos deben ser conectados de manera física.

2.1.3 Planificación del despliegue de nodos en el distrito de La Punta

La distribución de nodos se ha realizado cerca a zonas de exposición poblacional, tales como colegios, parques, centros de salud, etc.

FIGURA Nº4. 34
DISTRIBUCIÓN DE LOS NODOS EN EL DISTRITO DE LA PUNTA

Se ha elegido la topología Estrella para la red Zigbee, debido a que es la más simple y adecuada para un número pequeño de nodos que se encuentran en cercanía, además sólo requiere que el dispositivo coordinador tenga una fuente de alimentación continua.

CUADRO Nº 19
DETALLES DE LOS NODOS DE LA RED DE SENSORES

Nodo	Función	Coordenadas	Referencia
Nodo0	Nodo coordinador de la red Zigbee	Latitud: -12.0727667 Longitud: -77.1634638888889	Municipalidad
Nodo1	Nodo fuente de la red Zigbee	Latitud: -12.0744333 Longitud: -77.16321666666667	I.E Jardín Municipal
Nodo2	Nodo fuente de la red Zigbee	Latitud: -12.0739083 Longitud: -77.16340277777779	I.E. Parroquia Clara Cogorno de Cogorno
Nodo3	Nodo fuente de la red Zigbee	Latitud: -12.0716139 Longitud: -77.16446944444445	I.E Mini Skool
Nodo4	Nodo fuente de la red Zigbee	Latitud: -12.0684583 Longitud: -77.15809166666668	Centro de Salud La Punta

- **Enlace del nodo1**

El nodo 1 es ubicado para monitorear los niveles de intensidad de campo eléctrico cerca del instituto educativo Jardín Municipal. Para ver las características de este enlace se usará la fórmula de Friis con los siguientes datos.

TABLA Nº4. 9
DETALLES DEL ENLACE ENTRE EL NODO 1 Y EL NODO SUMIDERO

Distancia del enlace	Potencia de Transmisión	Ganancia del transmisor	Ganancia del receptor	Pérdidas cables/conectores
187.26m	5dB	1.5dB	1.5dB	3dB

$$P_{RX}(dB) = P_{TX}(dB) + G_{TX}(dB) + G_{RX}(dB) + 20\log\left(\frac{c}{4\pi \times f \times d}\right)$$

$$P_{RX}(dB) = 5 + 1.5 + 1.5 - 3 + 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times 187.26}\right)$$

$$P_{RX}(dB) = -80.5dB$$

- **Enlace del nodo2**

El nodo 2 es ubicado para monitorear los niveles de intensidad de campo eléctrico cerca del instituto educativo Parroquia Clara Cogorno de Cogorno. Para ver las características de este enlace se usará la fórmula de Friis con los siguientes datos.

TABLA Nº4. 10
DETALLES DEL ENLACE ENTRE EL NODO 2 Y EL NODO SUMIDERO

Distancia del enlace	Potencia de Transmisión	Ganancia del transmisor	Ganancia del receptor	Pérdidas cables/conectores
127.11m	5dB	1.5dB	1.5dB	3dB

$$P_{RX}(dB) = P_{TX}(dB) + G_{TX}(dB) + G_{RX}(dB) + 20\log\left(\frac{c}{4\pi \times f \times d}\right)$$

$$P_{RX}(dB) = 5 + 1.5 + 1.5 - 3 + 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times 127.11}\right)$$

$$P_{RX}(dB) = -77.136dB$$

- **Enlace del nodo3**

El nodo 3 es ubicado para monitorear los niveles de intensidad de campo eléctrico cerca del instituto educativo Mini Skool. Para ver las características de este enlace se usará la fórmula de Friis con los siguientes datos.

TABLA Nº4. 11
DETALLES DEL ENLACE ENTRE EL NODO 3 Y EL NODO SUMIDERO

Distancia del enlace	Potencia de Transmisión	Ganancia del transmisor	Ganancia del receptor	Pérdidas cables/conectores
168.48m	5dB	1.5dB	1.5dB	3dB

$$P_{RX}(dB) = P_{TX}(dB) + G_{TX}(dB) + G_{RX}(dB) + 20\log\left(\frac{c}{4\pi \times f \times d}\right)$$

$$P_{RX}(dB) = 5 + 1.5 + 1.5 - 3 + 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^9 \times 168.48}\right)$$

$$P_{RX}(dB) = -79.583dB$$

- **Enlace del nodo4**

El nodo 4 es ubicado para monitorear los niveles de intensidad de campo eléctrico cerca del Centro de Salud de La Punta. Para ver las características de este enlace se usará la fórmula de Friis con los siguientes datos.

TABLA Nº4. 12
DETALLES DEL ENLACE ENTRE EL NODO 4 Y EL NODO SUMIDERO

Distancia del enlace	Potencia de Transmisión	Ganancia del transmisor	Ganancia del receptor	Pérdidas cables/conectores
755.48m	5dB	1.5dB	1.5dB	3dB

$$P_{RX}(dB) = P_{TX}(dB) + G_{TX}(dB) + G_{RX}(dB) + 20\log\left(\frac{c}{4\pi \times f \times d}\right)$$

$$P_{RX}(dB) = 5 + 1.5 + 1.5 - 3 + 20\log\left(\frac{299\,792\,458}{4\pi \times 2.4 \times 10^{9} \times 755.48}\right)$$

$$P_{RX}(dB) = -92.616dB$$

2.1.4 Diseño del firmware de los nodos

Ya que hemos planteado el hardware sobre el cual va a operar la red de sensores, la siguiente etapa en esta investigación consiste en el diseño del firmware que regirá el comportamiento de los nodos.

A continuación, se describirá que herramientas de software será usado para el diseño de esta etapa y cuáles son los algoritmos que se usarán para la lectura de sensores, administración de batería y transmisión de datos.

2.1.4.a. Firmware del nodo fuente

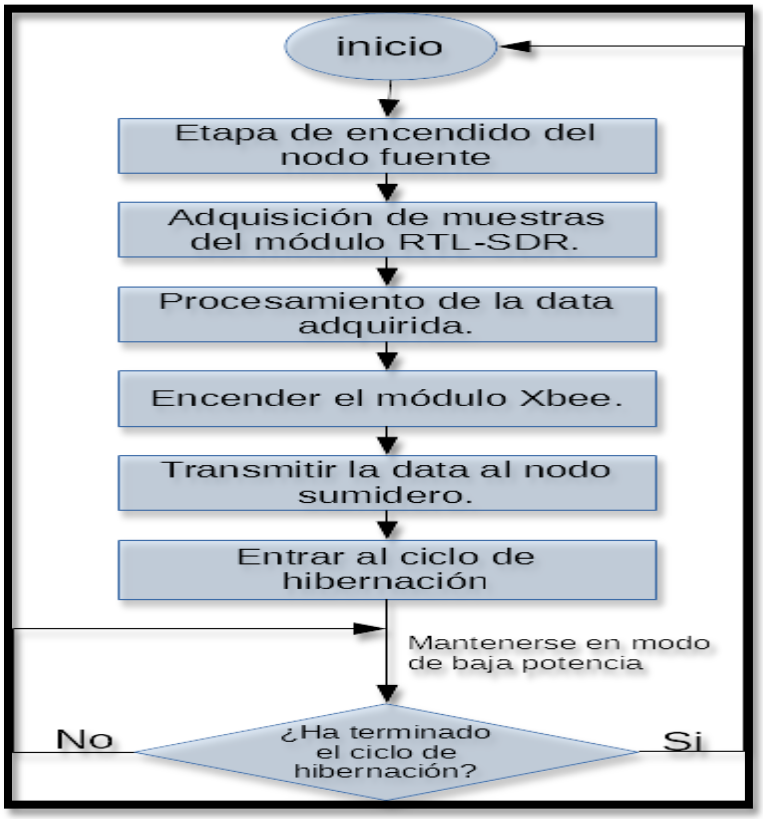
Los nodos fuente son los encargados de recolectar la data de los sensores y transmitirlas al nodo sumidero. Los nodos usan un controlador basado en el Raspberry Pi, el cual funciona con el sistema operativo Raspbian. Esto permite que el controlador sea programado con un lenguaje de programación de alto nivel para facilitar el desarrollo y agilizar el proceso de la investigación.

Se ha elegido el lenguaje Python para esta aplicación, el cual es un lenguaje de programación interpretado reconocido por énfasis en el orden y sencillez de su sintaxis. Python viene instalado por defecto en el sistema operativo Raspbian.

Con el lenguaje Python podemos escribir un script, que se encargue de la adquisición de los sensores del nodo, su pre-procesamiento y finalmente la transmisión de a través de los módulos radio transceptores Xbee.

El nodo fuente seguirá el siguiente diagrama de flujo para su funcionamiento:

FIGURA Nº4. 35
DIAGRAMA DE FLUJO DEL NODO FUENTE



Para poder usar los módulos RTL-SDR y los pines GPIO del Raspberry Pi se han descargado e instalado las siguientes herramientas.

- Numpy: Es una librería para el lenguaje de programación Python que agrega funciones para manipular arreglos, matrices, funciones de álgebra lineal, transformadas de Fourier, etc.
- Matplotlib: Es una librería para el lenguaje de programación Python que permite realizar gráficas y operaciones matemáticas de manera similar a MATLAB con la ventaja de ser software libre.
- Libusb: Es una librería para acceder dispositivos USB en plataformas Linux, OSX, Windows y BSD.
- librtlsdr: Es una librería que contiene comandos y herramientas para realizar la transferencia de datos del dispositivo RTL-SDR hacia la computadora.
- Pyrtlsdr: Es una librería para el lenguaje de programación Python que funciona como interfaz para los dispositivos RTL-SDR que usan el chipset RTL2832U basado en la librería librtlsdr.
- Pyserial: Es una librería para el lenguaje de programación Python que permite acceder al puerto serial del Raspberry pi, a través de su interfaz física o el puerto USB.

En el terminal de comandos del Raspberry pi, conectado a internet, se han ingresado los siguientes comandos para instalar las librerías de Python y la librería libusb que será necesaria para usar los drivers del módulo RTL-SDR.

FIGURA Nº4. 36
COMANDOS DE INSTALACIÓN DE LIBRERÍAS PARA EL RASPBERRY PI

```
1. sudo apt-get update
2. sudo apt-get install cmake build-essential python-pip libusb-1.0-0-dev python-numpy git pandoc
3. sudo pip install matplotlib
4. sudo pip install pyserial
```

La instalación de los drivers del módulo RTL-SDR para ser reconocido a través de la interfaz USB del Raspberry Pi se realizó ingresando los siguientes comandos en el terminal.

FIGURA Nº4. 37
COMANDOS DE INSTALACIÓN DE DRIVERS DEL MÓDULO RTL-SDR

```
1. git clone git://git.osmocom.org/rtl-sdr.git
2. cd rtl-sdr
3. mkdir build
4. cd build
5. cmake ../ -DINSTALL_UDEV_RULES=ON -DDETACH_KERNEL_DRIVER=ON
6. make
7. sudo make install
8. sudo ldconfig
```

Para finalizar la instalación, se descargó la librería pyrtlsdr que es usada en el script de Python para manejar el módulo RTL-SDR.

FIGURA Nº4. 38
COMANDO DE INSTALACIÓN DE LA LIBRERÍA PYRTLSDR

```
1. sudo pip install pyrtlsdr
```

Para verificar que todas las librerías han sido instaladas correctamente, se inició una consola de Python en el Raspberry Pi y se escribieron las siguientes funciones.

FIGURA Nº4. 39
VERIFICACIÓN DE LAS LIBRERÍAS INSTALADAS PARA EL LENGUAJE PYTHON

```
1. import numpy
2. import matplotlib
3. import rtlsdr
4. import serial
```

Si la instalación fue exitosa, la ejecución de dichos códigos no retornará ningún error.

- **Programación de la Etapa de Encendido**

La etapa de encendido es la etapa inicial del ciclo activo. El Raspberry Pi demorará aproximadamente 25 segundos en iniciar su sistema operativo. Cuando el sistema operativo Raspbian esté listo para funcionar, el Raspberry Pi del nodo fuente deberá iniciar sus funciones de manera automática y sin intervención de algún usuario. Para esto se creó un archivo

que especifica que scripts deben ejecutarse y dar el funcionamiento del nodo fuente. Este es un archivo Bash, el cual es un archivo de texto que incluye comandos de consola de Linux que se ejecutarán de manera secuencial y nos permitirá automatizar el comportamiento de nuestro dispositivo.

El archivo fue creado en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tiene el nombre de ***nodoFuente.sh***.

FIGURA Nº4. 40
PROGRAMACIÓN DEL ARCHIVO NODO FUENTE .SH

```
1. #!/bin/bash
2.
3. echo "Iniciando el funcionamiento del nodo fuente"
4.
5. python /home/pi/procesamiento.py
6. python /home/pi/apagado.py
```

La primera línea #!/bin/bash sirve como indicador de que este es un archivo Bash.

La función echo, sirve para mostrar un mensaje en la consola del terminal Linux en el Raspberry Pi. En el funcionamiento normal del nodo fuente en la red de sensores, este mensaje no será visible y esta línea puede ser removida del archivo, pero es útil para depurar el funcionamiento del script cuando se tiene conectado un monitor o una conexión ssh hacia el Raspberry Pi.

Las dos últimas líneas del archivo Bash indican que se debe ejecutar dos archivos de Python de forma secuencial. El primer archivo, ***procesamiento.py*****,** es un script de Python que contendrá la programación principal del nodo fuente, donde se realiza la lectura del sensor RTL-SDR, el procesamiento de las muestras adquiridas y la transmisión de datos hacia el nodo sumidero.

El segundo archivo, ***apagado.py***, es un script donde el nodo fuente se prepara para el modo hibernación y se comunica con el microcontrolador ATtiny45 para lograr el control de potencia. La creación y programación de ambos archivos será detallado en la presente investigación.

Para probar el funcionamiento del archivo ***nodoFuente.sh*** se puede escribir el siguiente comando la consola de Linux del Raspberry Pi.

FIGURA Nº4. 41
COMANDO PARA EJECUTAR EL ARCHIVO NODO-FUENTE .SH

```
1. sh /home/pi/nodoFuente.sh
```

Con el archivo ***nodoFuente.sh*** ya tenemos la secuencia de comandos que harán que el nodo fuente realice sus operaciones. Para que este archivo se ejecute de manera automática cada vez que el Raspberry Pi inicie el sistema operativo, se ha usado la herramienta Cron.

Cron es un programa de Linux que permite al usuario ejecutar comandos o scripts de manera automática a alguna hora o fecha específica. Para esta

aplicación se usará el prefijo @reboot que permite ejecutar funciones al iniciar el sistema operativo.

Cron usa un archivo llamado crontab, el cual es un archivo de texto con la lista de los scripts a ejecutar. Para editar el archivo crontab en el Raspberry Pi, hemos ingresado lo siguiente en la consola de comandos.

FIGURA Nº4. 42
COMANDO PARA EDITAR EL ARCHIVO CRONTAB DEL RASPBERRY PI

```
1. crontab -e
```

Finalmente agregamos la siguiente línea en la última parte del archivo crontab para hacer que el script ***nodoFuente.sh*** se ejecute a la hora de iniciar el sistema.

FIGURA Nº4. 43
COMANDO PARA EJECUTAR EL ARCHIVO NODO-FUENTE.SH AL INICIAR EL OS

```
1. @reboot sh /home/pi/nodoFuente.sh
```

Después de guardar los cambios al archivo crontab y reiniciar el Raspberry Pi, vemos que cada vez que se inicie el sistema operativo, los comandos del archivo ***nodoFuente.sh*** se ejecutarán de manera automática en la secuencia especificada.

❖ **Programación de configuración del radio Xbee en el nodo fuente**

Esta no es una etapa propia del ciclo de operación del nodo. Esta configuración solo debe ejecutarse una vez y definirán parámetros como la dirección del dispositivo, Identificador de la red, tasa de transmisión y otros valores.

Los módulos Xbee son dispositivos muy accesibles para los usuarios. Usan una interfaz UART para comunicarse con los microcontroladores o computadoras a las cuales son conectados de manera física. En nuestro caso, el módulo Xbee se conecta al Raspberry Pi a través de su puerto UART en los pines GPIO14 (TXD) y GPIO15 (RXD). Para enviar los comandos de configuración del módulo Xbee, este debe operar en el modo de comandos AT, el cuál es accesible desde la interfaz UART.

La conexión serial y los comandos serán enviados a través del script de Python usando las librerías instaladas previamente. Este script se ha creado en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tiene el nombre de ***config_nodo_fuente.py***.

FIGURA Nº4. 44
PROGRAMACIÓN DE PYTHON DEL ARCHIVO CONFIG_NODO_FUENTE.PY

```
1. import serial
2. import time
3. from gpiozero import DigitalOutputDevice
4.
5. """ constantes del módulo xbee """
6. PUERTO_XBEE = 'dev/ttyAMA0'
```

```
7.  PIN_XBEE = 18
8.
9.  """ funciones del módulo xbee """
10. #Función para despertar el módulo Xbee de hibernación
11. def despertar_xbee(pin_xbee):
12.     pin_xbee.off()
13.
14. #Función para transmitir datos seriales al módulo Xbee
15. def transmitir_xbee(modulo_xbee, data):
16.     modulo_xbee.write(data) #enviar la data
17.
18. #Función para leer la respuesta del módulo Xbee
19. def leer_xbee(modulo_xbee):
20.     respuesta = modulo_xbee.readline()
21.     return respuesta
22.
23.
24. if __name__ == '__main__':
25.     #configurar el puerto serial y el GPIO18
26.     modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudr
    ate=9600)
27.     pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_va
    lue=True)
28.
29.     despertar_xbee(pin_xbee)
30.     time.sleep(1)
31.
32.     #iniciar modo comandos AT
33.     transmitir_xbee(modulo_xbee, '+++')
34.     respuesta = leer_xbee(modulo_xbee)
35.
```

```
#restaurar configuración por defecto
transmitir_xbee(modulo_xbee, 'ATRE\r')
respuesta = leer_xbee(modulo_xbee)

#configurar dispositivo final
transmitir_xbee(modulo_xbee, 'ATCE 0\r')
respuesta = leer_xbee(modulo_xbee)

#configurar el PAN ID
transmitir_xbee(modulo_xbee, 'ATID 2018\r')
respuesta = leer_xbee(modulo_xbee)

#activar verificación de canal
transmitir_xbee(modulo_xbee, 'ATJV 1\r')
respuesta = leer_xbee(modulo_xbee)

#configurar la dirección destino del nodo sumidero
transmitir_xbee(modulo_xbee, 'ATDH 0013A200\r')
respuesta = leer_xbee(modulo_xbee)
transmitir_xbee(modulo_xbee, 'ATDL 4103604A\r')
respuesta = leer_xbee(modulo_xbee)

#Potencia de Transmisión = 5dBm
transmitir_xbee(modulo_xbee, 'ATPL 4\r')
respuesta = leer_xbee(modulo_xbee)

#Modo boost desactivado
transmitir_xbee(modulo_xbee, 'ATPM 0\r')
respuesta = leer_xbee(modulo_xbee)

#Modo hibernación activado por pin
```

```
transmitir_xbee(modulo_xbee, 'ATSM 1\r')
respuesta = leer_xbee(modulo_xbee)

#aplicar cambios
transmitir_xbee(modulo_xbee, 'ATAC\r')
respuesta = leer_xbee(modulo_xbee)

#escribir a memoria no volatil
transmitir_xbee(modulo_xbee, 'ATWR\r')
respuesta = leer_xbee(modulo_xbee)

#reinicia el dispositivo
transmitir_xbee(modulo_xbee, 'ATFR\r')
respuesta = leer_xbee(modulo_xbee)

#esperar 3 segundos para que se reinicie
time.sleep(3)

#poner el módulo xbee en hibernación
pin_xbee.close()
```

La primera parte del script es la inclusión de librerías que fueron descargadas e instaladas previamente en el Rasbperry Pi.

FIGURA Nº4. 45
INCLUSIÓN DE LIBRERÍAS DEL ARCHIVO CONFIG_NODO_FUENTE.PY

```
import serial
import time
from gpiozero import DigitalOutputDevice
```

La librería ***serial*** (proveniente de pyserial) será usada para configurar el puerto UART del Raspberry Pi y permitirá establecer la comunicación entre este y el módulo Xbee.

La librería ***time*** será usada para generar tiempos de espera, para que el módulo Xbee termine ciertas operaciones después de recibir comandos.

La librería ***gpiozero*** incluye el objeto ***DigitalOutputDevice***, que permite configurar los pines GPIO como salidas digitales que generarán valores de 3.3V (estado ON o HIGH) y 0V (estado OFF o LOW).

Después de la inclusión de librerías, se definieron las constantes y funciones que serán usadas en el script para comunicarse con el módulo Xbee.

FIGURA Nº4. 46
FUNCIONES PARA EL MÓDULO XBEE DEL ARCHIVO CONFIG_NODO_FUENTE.PY

```
""" constantes del módulo Xbee """
PUERTO_XBEE = 'dev/ttyAMA0'
PIN_XBEE = 18

""" funciones del módulo xbee """
#Función para despertar el módulo Xbee de hibernación
def despertar_xbee(pin_xbee):
    pin_xbee.off()

#Función para transmitir datos seriales al módulo Xbee
def transmitir_xbee(modulo_xbee, data):
    modulo_xbee.write(data) #enviar la data
```

```
17.
18. #Función para leer la respuesta del módulo Xbee
19. def leer_xbee(modulo_xbee):
20.     respuesta = modulo_xbee.readline()
21.     return respuesta
```

La variable ***PUERTO_XBEE*** contiene una cadena de texto que almacena la dirección que el Raspberry Pi usa para acceder el puerto serial UART con interfaz en los pines GPIO14 y GPIO15.

La variable ***PIN_XBEE*** contiene el valor 18, haciendo referencia al pin GPIO18 del Raspberry Pi que está conectado al pin SLEEP_RQ del Xbee para controlar el modo hibernación del módulo. El pin SLEEP_RQ mantiene al módulo en hibernación cuando está conectado a 3.3V y despierta de hibernación cuando es conectado a 0V. En nuestro diagrama de conexiones, este pin está conectado a un resistor pull-up, el cual mantiene el dispositivo en modo hibernación por defecto. Cuando el Raspberry Pi está listo para comunicarse con el módulo Xbee, primero este debe despertar el módulo, y esto ocurre cuando el GPIO18 genera un voltaje de 0V en su salida.

La función ***despertar_xbee*** toma como argumento a ***pin_xbee***, este será un objeto que controla el pin GPIO18 del Raspberry Pi y será definido en la sección principal (main) del script usando la librería ***gpiozero***. Esta función despertará al módulo Xbee de su estado de hibernación para poder enviarle datos por el puerto serial UART.

La función ***transmitir_xbee*** es usada para enviar datos desde el Raspberry Pi hacia el módulo Xbee. Esta función tiene como argumentos, ***módulo_xbee***, el cual es un objeto configurado como la conexión de puerto serial UART del Raspberry Pi usando la librería ***serial*** y definido en la sección principal (main) del script. El argumento ***data***, el cual contiene el mensaje que será enviado en forma de texto.

La función ***leer_xbee*** es usada para leer la respuesta que el módulo Xbee envía al Raspberry Pi. El argumento ***modulo_xbee*** es el objeto del puerto serial UART del Raspberry Pi, y será el mismo usado en la función ***transmitir_xbee***.

La sección principal (main) del script, es donde se realiza la configuración del puerto serial y del pin GPIO18 que se conecta al pin SLEEP_RQ del módulo Xbee y la transmisión de comandos AT para configurar los parámetros de operación del módulo Xbee en el nodo fuente.

FIGURA Nº4. 47
SECCIÓN PRINCIPAL DEL ARCHIVO CONFIG_NODO_FUENTE.PY

```
24. if __name__ == '__main__':
25.     #configurar el puerto serial y el GPIO18
26.     modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrate=9600
    )
27.     pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_value=True
    )
28.
29.     despertar_xbee(pin_xbee)
30.     time.sleep(1)
```

```
31.
32.  #iniciar modo comandos AT
33.  transmitir_xbee(modulo_xbee, '+++')
34.  respuesta = leer_xbee(modulo_xbee)
35.
36.  #restaurar configuración por defecto
37.  transmitir_xbee(modulo_xbee, 'ATRE\r')
38.  respuesta = leer_xbee(modulo_xbee)
39.
40.  #configurar dispositivo final
41.  transmitir_xbee(modulo_xbee, 'ATCE 0\r')
42.  respuesta = leer_xbee(modulo_xbee)
43.
44.  #configurar el PAN ID
45.  transmitir_xbee(modulo_xbee, 'ATID 2018\r')
46.  respuesta = leer_xbee(modulo_xbee)
47.
48.  #activar verificación de canal
49.  transmitir_xbee(modulo_xbee, 'ATJV 1\r')
50.  respuesta = leer_xbee(modulo_xbee)
51.
52.  #configurar la dirección destino del nodo sumidero
53.  transmitir_xbee(modulo_xbee, 'ATDH 0013A200\r')
54.  respuesta = leer_xbee(modulo_xbee)
55.  transmitir_xbee(modulo_xbee, 'ATDL 4103604A\r')
56.  respuesta = leer_xbee(modulo_xbee)
57.
58.  #Potencia de Transmisión = 5dBm
59.  transmitir_xbee(modulo_xbee, 'ATPL 4\r')
60.  respuesta = leer_xbee(modulo_xbee)
61.
```

```
62.  #Modo boost desactivado
63.  transmitir_xbee(modulo_xbee, 'ATPM 0\r')
64.  respuesta = leer_xbee(modulo_xbee)
65.
66.  #Modo hibernación activado por pin
67.  transmitir_xbee(modulo_xbee, 'ATSM 1\r')
68.  respuesta = leer_xbee(modulo_xbee)
69.
70.  #aplicar cambios
71.  transmitir_xbee(modulo_xbee, 'ATAC\r')
72.  respuesta = leer_xbee(modulo_xbee)
73.
74.  #escribir a memoria no volátil
75.  transmitir_xbee(modulo_xbee, 'ATWR\r')
76.  respuesta = leer_xbee(modulo_xbee)
77.
78.  #reinicia el dispositivo
79.  transmitir_xbee(modulo_xbee, 'ATFR\r')
80.  respuesta = leer_xbee(modulo_xbee)
81.
82.  #esperar 3 segundos para que se reinicie
83.  time.sleep(3)
84.
85.  #poner el módulo Xbee en hibernación
86.  pin_xbee.close()
```

Después de enviar un comando AT al módulo Xbee con la función ***transmitir_xbee***, se debe esperar a que el módulo responda antes de

ejecutar el siguiente comando. Para identificar esta respuesta se ha usado la función ***leer_xbee*** que guarda el resultado en una variable.

Después de configurar todos los parámetros necesarios, se ejecuta el comando *ATWR*, para escribir los valores en memoria no volátil y el comando *ATFR*, para reiniciar el módulo, sacarlo de modo de comandos AT y ahora el módulo Xbee operará con los cambios aplicados.

No es necesario ejecutar este script de configuración cada vez que se enciende el nodo, pues la información será almacenada en la memoria no volátil del radio Xbee.

Para probar el funcionamiento del script ***config_nodo_fuente.py*** se puede escribir el siguiente comando la consola de Linux del Raspberry Pi.

FIGURA Nº4. 48
COMANDO PARA EJECUTAR EL ARCHIVO CONFIG_NODO_FUENTE.PY

```
1. python /home/pi/config_nodo_fuente.py
```

- **Programación de Adquisición, Procesamiento y transmisión de datos**

Es la etapa principal del funcionamiento del nodo fuente. En esta etapa se ha definido cuáles son las bandas de frecuencias a analizar, se configura el módulo RTL-SDR para realizar la adquisición de muestras de radio, se almacenan dichos valores para calcular los valores de intensidad de campo y finalmente se transmiten las lecturas al nodo sumidero.

La conexión con el módulo RTL-SDR y el módulo Xbee, configurado como dispositivo final, además del procesamiento de las muestras adquiridas, se realizaron a través de un script de Python usando las librerías instaladas previamente. Este script se ha creado en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tiene el nombre de ***procesamiento.py***.

FIGURA Nº4. 49
PROGRAMACIÓN DE PYTHON DEL PROCESAMIENTO.PY

```
1.  import rtlsdr
2.  import numpy as np
3.  import matplotlib
4.  import time
5.  import serial
6.  from gpiozero import DigitalOutputDevice
7.
8.  """ constantes de adquisición"""
9.  C = 299792458   #velocidad de la luz
10. GAIN = 1.5      #ganancia de la antena RTL-SDR
11.
12. FS = 2e6        #frecuencia de muestreo
13. FDIV = 2        #divisor de frecuencia
14. FSTEP = FS/FDIV #resolución de muestreo
15. N_FFT = 128     #N puntos en la transformada de fourier
16.
17. N_MUESTRAS = 16*1024  #número de muestrar para adquisición
18.
19. """ funciones """
20. #genera arreglo de frecuencias que serán analizadas por el sensor
```

```
21. def arreglo_de_frecuencias(f0, fn):
22.     frecuencias = np.arange(f0, fn+FSTEP, FSTEP)  #arreglo de frecu
    encias
23.     return frecuencias
24.
25. #función para leer muestras en el arreglo de frecuencias generado
26. def adquirir_muestras(sdr, f0, fn):
27.     frecuencias = arreglo_de_frecuencias(f0, fn)
28.     muestras = []   #arreglo de salida
29.     for fcentral in frecuencias:
30.         sdr.center_freq = fcentral          #configura frecuencia central
31.         ss = sdr.read_samples(N_MUESTRAS)   #leer n muestras
32.         muestras = np.append(muestras, ss)  #adjuntar las muestras a l
    a salida
33.     return muestras     #retornar las muestras capturadas
34.
35. #función para procesar la data adquirida por el sensor
36. def medicion_modulo_RTL(sdr, f0, fn):
37.     frecuencias = arreglo_de_frecuencias(f0, fn)
38.     muestras = adquirir_muestras(sdr, f0, fn)
39.
40.     m = int((len(frecuencias)+(FDIV-
    1))*N_FFT/FDIV) #dimensiones de la salida
41.     arreglo_P = np.zeros(m)   #arreglo de niveles de potencia
42.     arreglo_E = np.zeros(m)   #arreglo de intensidad de campo
43.     #iterar sobre cada frecuencia central
44.     for i in range(len(frecuencias)):
45.         fcentral = frecuencias[i]   #fcentral del arreglo de frecuencias
46.         j = i*N_MUESTRAS        #indice para el arreglo de muestras
47.         jj = int(i*N_FFT/FDIV)  #indice para el arreglo de salida del FFT
48.         ss = muestras[j:j+N_MUESTRAS]
```

```
49.         P, f = matplotlib.mlab.psd(ss, NFFT=N_FFT, Fs=FS)   #cálculo d
    e FFT
50.         f = f + fcentral        #ajuste de las frecuencias del FFT
51.         w = np.max([P, arreglo_P[jj:jj+N_FFT]], axis=0)   #sobreposicion

52.         arreglo_P[jj:jj+N_FFT] = w
53.         #cálculo de densidad de potencia e intensidad de campo
54.         longitud_onda = C / f
55.         Prx = w/(10**(sdr.gain/100))/1.5    #potencia de Rx (CALIBRAR)

56.         Ae = (GAIN*longitud_onda**2)/(4*np.pi)  #apertura de antena
57.         S = Prx/Ae                  #densidad de potencia
58.         E = np.sqrt(S*120*np.pi)    #intensidad de campo
59.         arreglo_E[jj:jj+N_FFT] = E  #adjuntar los caluclos al arreglo
60.     #retornar la intensidad de campo
61.     n = int(N_FFT/2)    #límites para el arreglo de salida
62.     return arreglo_E[n:-n].mean()
63.
64.
65. """ constantes del módulo xbee """
66. PUERTO_XBEE = 'dev/ttyAMA0'
67. PIN_XBEE = 18
68.
69. """ funciones del módulo xbee """
70. #Función para despertar el módulo Xbee de hibernación
71. def despertar_xbee(pin_xbee):
72.     pin_xbee.off()
73.
74. #Función para transmitir datos seriales al módulo Xbee
75. def transmitir_xbee(modulo_xbee, data):
76.     modulo_xbee.write(data) #enviar la data
```

```
77.
78.
79. """ proceso principal """
80. if __name__ == '__main__':
81.     #Configuración del módulo RTL
82.     sdr = rtlsdr.RtlSdr()   #iniciar el módulo RTL-SDR
83.     sdr.sample_rate = FS    #configura frecuencia de muestreo
84.     sdr.gain = 125          #ganancia interna de 1.25dB
85.     #Bandas de frecuencia
86.     bandas = {'banda1':[[758e6, 773e6], []],    #758-773MHz
87.               'banda2':[[773e6, 788e6], []],    #773-788MHz
88.               'banda3':[[788e6, 803e6], []],    #788-803MHz
89.               'banda4':[[869e6, 880e6], []],    #869-880MHz
90.               'banda5':[[880e6, 890e6], []],    #880-890MHz
91.               'banda6':[[944e6, 960e6], []]}    #944-960MHz
92.     #Límites de tiempo
93.     duracion = 360          #6 minutos
94.     t_inicio = time.time()  #tiempo actual
95.     #bucle principal
96.     while( (time.time() - t_inicio) <= duracion):
97.         #iterando sobre el diccionario
98.         for indice, contenido in bandas.items():
99.             f0 = contenido[0][0]    #frecuencia inicial
100.                fn = contenido[0][1]    #frecuencia final
101.                intensidad = medicion_modulo_RTL(sdr, f0, fn)
102.                contenido[1] = np.append(contenido[1], intensidad)
103.    #terminar el funcionamiento del módulo RTL-SDR
104.    sdr.close()
105.    #intensidad de campo promedio en mV/m en cada banda
106.    for indice, contenido in bandas.items():
107.        contenido[1] = contenido[1].mean()*1000
```

```
108.
109.
110.    #Configuración del módulo Xbee y su pin de hibernación
111.    modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrate=9600)
112.    pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_value=True)
113.
114.    #Despertar al módulo de hibernación
115.    despertar_xbee(pin_xbee)
116.    time.sleep(3)
117.
118.    #Enviar data al nodo sumidero
119.    data = 'N1:{},{},{},{},{},{}\n'.format(
120.            bandas['banda1'][1],bandas['banda2'][1],
121.            bandas['banda3'][1],bandas['banda4'][1],
122.            bandas['banda5'][1],bandas['banda6'][1])
123.    transmitir_xbee(modulo_xbee, data)
124.    time.sleep(2)
125.    pin_xbee.close()    #terminar el funcionamiento del módulo Xbee
```

La primera parte del script es la inclusión de librerías

FIGURA Nº4. 50
INCLUSIÓN DE LIBRERÍAS DEL ARCHIVO PROCESAMIENTO.PY

```
1.  import rtlsdr
2.  import numpy as np
3.  import matplotlib
4.  import time
5.  import serial
6.  from gpiozero import DigitalOutputDevice
```

La librería ***rtlsdr*** es la que permite la interacción entre el módulo RTL-SDR y el script de Python, e incluye funciones de inicio, configuración y lectura del módulo.

La librería ***numpy*** incluye funciones para realizar operaciones y manipular arreglos, en el archivo, se usará el prefijo ***np*** para usar las funciones de esta librería.

La librería ***matplotlib*** está basada en numpy e incluye algunas funciones matemáticas que serán usadas en el cálculo de la transformada rápida de Fourier (FFT).

La librería ***time*** será usada para medir lapsos de tiempo, específicamente para que se ejecute un bucle durante cierto periodo definido por el usuario.

La librería ***serial*** (proveniente de pyserial) será usada para configurar el puerto UART del Raspberry Pi y permitirá establecer la comunicación entre este y el módulo Xbee.

Después de la inclusión de librerías, se definieron las constantes y funciones que son usadas en la lectura del módulo RTL-SDR y el procesamiento de muestras adquiridas.

FIGURA Nº4. 51
FUNCIONES PARA EL MÓDULO RTL-SDR DEL ARCHIVO PROCESAMIENTO.PY

```
8.  """ constantes de adquisición"""
9.  C = 299792458   #velocidad de la luz
10. GAIN = 1.5      #ganancia de la antena RTL-SDR
11.
```

```
12. FS = 2e6        #frecuencia de muestreo
13. FDIV = 2        #divisor de frecuencia
14. FSTEP = FS/FDIV #resolución de muestreo
15. N_FFT = 128     #N puntos en la transformada de fourier
16.
17. N_MUESTRAS = 16*1024  #número de muestrar para adquisición
18.
19. """ funciones """
20. #genera arreglo de frecuencias que serán analizadas por el sensor
21. def arreglo_de_frecuencias(f0, fn):
22.     frecuencias = np.arange(f0, fn+FSTEP, FSTEP)  #arreglo de frecu
    encias
23.     return frecuencias
24.
25. #función para leer muestras en el arreglo de frecuencias generado
26. def adquirir_muestras(sdr, f0, fn):
27.     frecuencias = arreglo_de_frecuencias(f0, fn)
28.     muestras = []   #arreglo de salida
29.     for fcentral in frecuencias:
30.         sdr.center_freq = fcentral          #configura frecuencia central
31.         ss = sdr.read_samples(N_MUESTRAS)   #leer n muestras
32.         muestras = np.append(muestras, ss)  #adjuntar las muestras a l
    a salida
33.     return muestras     #retornar las muestras capturadas
34.
35. #función para procesar la data adquirida por el sensor
36. def medicion_modulo_RTL(sdr, f0, fn):
37.     frecuencias = arreglo_de_frecuencias(f0, fn)
38.     muestras = adquirir_muestras(sdr, f0, fn)
39.
40.     m = int((len(frecuencias)+(FDIV-
```

```
    1))*N_FFT/FDIV) #dimensiones de la salida
41.     arreglo_P = np.zeros(m)   #arreglo de niveles de potencia
42.     arreglo_E = np.zeros(m)   #arreglo de intensidad de campo
43.     #iterar sobre cada frecuencia central
44.     for i in range(len(frecuencias)):
45.         fcentral = frecuencias[i]   #fcentral del arreglo de frecuencias
46.         j = i*N_MUESTRAS        #indice para el arreglo de muestras
47.         jj = int(i*N_FFT/FDIV)  #indice para el arreglo de salida del FFT
48.         ss = muestras[j:j+N_MUESTRAS]
49.         P, f = matplotlib.mlab.psd(ss, NFFT=N_FFT, Fs=FS)   #cálculo d
    e FFT
50.         f = f + fcentral        #ajuste de las frecuencias del FFT
51.         w = np.max([P, arreglo_P[jj:jj+N_FFT]], axis=0)   #sobreposicion

52.         arreglo_P[jj:jj+N_FFT] = w
53.         #cálculo de densidad de potencia e intensidad de campo
54.         longitud_onda = C / f
55.         Prx = w/(10**(sdr.gain/100))/1.5    #potencia de Rx (CALIBRAR)

56.         Ae = (GAIN*longitud_onda**2)/(4*np.pi)  #apertura de antena
57.         S = Prx/Ae                  #densidad de potencia
58.         E = np.sqrt(S*120*np.pi)    #intensidad de campo
59.         arreglo_E[jj:jj+N_FFT] = E  #adjuntar los caluclos al arreglo
60.     #retornar la intensidad de campo
61.     n = int(N_FFT/2)    #límites para el arreglo de salida
62.     return arreglo_E[n:-n].mean()
```

Las constantes de adquisición son valores que serán usados durante el procesamiento y cálculo de las muestras adquiridas por el módulo RTL-

SDR como para la función de la transformada rápida de Fourier, longitud de onda, cálculo de apertura de antena, etc.

La función ***arreglo_de_frecuencias*** usa la función de ***arange*** de la librería ***numpy*** para generar un arreglo que contiene un rango de frecuencias que usará el módulo RTL-SDR para realizar la adquisición. Esta función tiene dos argumentos, ***f0*** y ***fn*** indican la frecuencia inicial y frecuencia final respectivamente. El resultado de esta función es usada en otras funciones de la aplicación.

La función ***adquirir_muestras*** se encarga de iniciar la lectura del módulo RTL-SDR y retornar las muestras en un arreglo de salida. Esta función tiene los argumentos, ***sdr***, ***f0*** y ***fn***. El primer argumento es el objeto que establece la conexión con el módulo RTL-SDR y será definido en la sección principal (main) del código usando la librería ***rtlsdr***. Los últimos dos argumentos son las frecuencias de inicio y fin de la banda que el módulo RTL-SDR analizará. Esta función hace uso de la librería ***rtlsdr*** para configurar la frecuencia central del sintonizador en el módulo RTL-SDR y leer una cantidad determinada de muestras, esta cantidad es definida en la variable ***N_MUESTRAS***.

La función ***medicion_modulo_RTL*** es la función principal de esta aplicación. Esta función tiene los argumentos, ***sdr***, ***f0*** y ***fn***. El primer argumento es el objeto que establece la conexión con el módulo RTL-SDR y será definido en la sección principal (main) del código usando la librería ***rtlsdr***. Los últimos dos arreglos son las frecuencias de inicio y fin de la

banda sobre la cual se realizará la medición de intensidad de campo eléctrico.

Esta función depende de las dos funciones anteriores, ***arreglo_de_frecuencias*** y ***adquirir_muestras***, para adquirir las muestras del módulo RTL-SDR en la banda de frecuencias especificadas. Después de obtener las muestras, la función realiza los cálculos para obtener los niveles de intensidad de campo eléctrico. Haciendo uso de las funciones matemáticas de la librería ***matplotlib*** y ***numpy*** se realizar el cálculo de la transformada rápida de Fourier y convertir las muestras adquiridas con el módulo RTL-SDR del dominio de tiempo al dominio de frecuencias. Esta función retorna la intensidad de campo eléctrico promedio en la banda de frecuencias que se está analizando.

En el script ***procesamiento.py***, una vez que se obtiene el valor de intensidad de campo eléctrico, esta información es transmitida hacia el nodo sumidero haciendo uso de la red Zigbee. Por este motivo, en el script se incluyen las constantes y funciones que permiten acceder al módulo Xbee y establecer comunicación con el nodo sumidero.

FIGURA Nº4. 52
FUNCIONES PARA EL MÓDULO XBEE DEL ARCHIVO PROCESAMIENTO.PY

```
66. """ constantes del módulo xbee """
67. PUERTO_XBEE = 'dev/ttyAMA0'
68. PIN_XBEE = 18
69.
70. """ funciones del módulo xbee """
```

```
71. #Función para despertar el módulo Xbee de hibernación
72. def despertar_xbee(pin_xbee):
73.     pin_xbee.off()
74.
75. #Función para transmitir datos seriales al módulo Xbee
76. def transmitir_xbee(modulo_xbee, data):
77.     modulo_xbee.write(data) #enviar la data
```

La variable ***PUERTO_XBEE*** contiene una cadena de texto que almacena la dirección que el Raspberry Pi usa para acceder el puerto serial UART con interfaz en los pines GPIO14 y GPIO15.

La variable ***PIN_XBEE*** contiene el valor 18, haciendo referencia al pin GPIO18 del Raspberry Pi que está conectado al pin SLEEP_RQ del Xbee para controlar el modo hibernación del módulo. El pin SLEEP_RQ mantiene al módulo en hibernación cuando está conectado a 3.3V y despierta de hibernación cuando es conectado a 0V. En nuestro diagrama de conexiones, este pin está conectado a un resistor pull-up, el cual mantiene el dispositivo en modo hibernación por defecto. Cuando el Raspberry Pi está listo para comunicarse con el módulo Xbee, primero este debe despertar el módulo, y esto ocurre cuando el GPIO18 genera un voltaje de 0V en su salida.

La función ***despertar_xbee*** toma como argumento a ***pin_xbee***, este será un objeto que controla el pin GPIO18 del Raspberry Pi y será definido en la sección principal (main) del script usando la librería ***gpiozero***. Esta función

despertará al módulo Xbee de su estado de hibernación para poder enviarle datos por el puerto serial UART.

La función ***transmitir_xbee*** es usada para enviar datos desde el Raspberry Pi hacia el módulo Xbee. Esta función tiene como argumentos, ***módulo_xbee***, el cual es un objeto configurado como la conexión de puerto serial UART del Raspberry Pi usando la librería ***serial*** y definido en la sección principal (main) del script. El argumento ***data***, el cual contiene el mensaje que será enviado en forma de texto. El módulo Xbee ha sido configurado para operar en modo transparente, esto quiere decir que cualquier dato que sea enviado por la interfaz UART será transmitida de manera inalámbrica hacia el nodo destino, en nuestro caso, el nodo sumidero.

En la sección principal (main) del script, es donde se utilizan todas estas funciones y constantes para realizar la tarea principal del nodo fuente.

FIGURA Nº4. 53
SECCIÓN PRINCIPAL DEL ARCHIVO PROCESAMIENTO.PY

```
79. """ proceso principal """
80. if __name__ == '__main__':
81.     #Configuración del módulo RTL
82.     sdr = rtlsdr.RtlSdr()   #iniciar el módulo RTL-SDR
83.     sdr.sample_rate = FS    #configura frecuencia de muestreo
84.     sdr.gain = 125          #ganancia interna de 1.25dB
85.     #Bandas de frecuencia
86.     bandas = {'banda1':[[758e6, 773e6], []],    #758-773MHz
87.               'banda2':[[773e6, 788e6], []],    #773-788MHz
```

```
            'banda3':[[788e6, 803e6], []],    #788-803MHz
            'banda4':[[869e6, 880e6], []],    #869-880MHz
            'banda5':[[880e6, 890e6], []],    #880-890MHz
            'banda6':[[944e6, 960e6], []]}    #944-960MHz
    #Límites de tiempo
    duracion = 360          #6 minutos
    t_inicio = time.time()  #tiempo actual
    #bucle principal
    while( (time.time() - t_inicio) <= duracion):
        #iterando sobre el diccionario
        for indice, contenido in bandas.items():
            f0 = contenido[0][0]    #frecuencia inicial
            fn = contenido[0][1]    #frecuencia final
            intensidad = medicion_modulo_RTL(sdr, f0, fn)
            contenido[1] = np.append(contenido[1], intensidad)

        #terminar el funcionamiento del módulo RTL-SDR
        sdr.close()
        #intensidad de campo promedio en mV/m en cada banda
        for indice, contenido in bandas.items():
            contenido[1] = contenido[1].mean()*1000


        #Configuración del módulo Xbee y su pin de hibernación
        modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrat
    e=9600)
        pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_valu
    e=True)

        #Despertar al módulo de hibernación
        despertar_xbee(pin_xbee)
```

```
            time.sleep(3)

            #Enviar data al nodo sumidero
            data = 'N1:{},{},{},{},{},{}\n'.format(
                    bandas['banda1'][1],bandas['banda2'][1],
                    bandas['banda3'][1],bandas['banda4'][1],
                    bandas['banda5'][1],bandas['banda6'][1])
            transmitir_xbee(modulo_xbee, data)
            time.sleep(2)

    pin_xbee.close()    #terminar el funcionamiento del módulo Xbee
```

En la sección principal del programa se muestra la configuración del módulo RTL-SDR usando las funciones de la librería ***rtlsdr***. Se configura la frecuencia de muestreo, la cual ya se ha definido en la sección de constantes de adquisición, y la ganancia interna del módulo, la cual se configuró a 1.25dB.

Las bandas de frecuencia de los servicios públicos de telecomunicaciones que se van a analizar son las siguientes:

TABLA Nº4. 13
BANDAS DE FRECUENCIA ANALIZADA POR EL NODO FUENTE

	Frecuencia inicial	**Frecuencia final**	**Frecuencia central**	**Operador**
Banda 1	758 MHz	773 MHz	765.5MHz	Entel
Banda 2	773 MHz	788 MHz	780.5MHz	Claro
Banda 3	788 MHz	803 MHz	795.5MHz	Movistar

Banda 4	869 MHz	880 MHz	874.5MHz	Movistar
Banda 5	880 MHz	890 MHz	885 MHz	Claro
Banda 6	944 MHz	960 MHz	952 MHz	Bitel

Para representar esta información en el script, se creará una variable de Python del tipo diccionario. Un diccionario de Python es una estructura de datos que permite almacenar cualquier tipo de datos como números, cadenas, arreglos e incluso funciones, en este caso se creará el diccionario con el nombre ***bandas***, donde se almacenará los rangos de frecuencia de cada banda que se va a analizar y un arreglo vacío donde se almacenarán las lecturas de intensidad de campo en cada iteración. Podemos ver cada parte de esta variable en la siguiente tabla.

TABLA Nº4. 14
ESTRUCTURA DEL DICCIONARIO DE PYTHON DE LA VARIABLE “BANDAS”

Estructura de la variable “bandas”		
Índices del diccionario	Contenido del diccionario	
	Contenido[0]: Rangos de frecuencia	Contenido[1]: Arreglos de las medidas de intensidad de campo eléctrico
‘banda1’	[758000000, 773000000]	Recolección de lecturas en la banda1
‘banda2’	[773000000, 788000000]	Recolección de lecturas en la banda2
‘banda3’	[788000000, 803000000]	Recolección de lecturas en la banda3
‘banda4’	[869000000, 880000000]	Recolección de lecturas en la banda4
‘banda5’	[880000000, 890000000]	Recolección de lecturas en la banda5
‘banda6’	[944000000, 960000000]	Recolección de lecturas en la banda6

La medición de los niveles de intensidad de campo eléctrico se realiza durante 360 segundos, se han creado la variable duración y la variable tiempo de inicio como ***duracion*** y ***t_inicio*** respectivamente para ser usadas en el script.

Usando un bucle y la librería ***time*** se ejecuta la lectura, adquisición y cálculo de la intensidad de campo eléctrico con el módulo RTL-SDR, de manera continua hasta que hayan transcurrido tantos segundos como los que se han definido en la variable de duración. En cada iteración de este bucle, el resultado de la medición de intensidad de campo eléctrico es almacenado en la sección respectiva del diccionario ***bandas***.

Al finalizar el bucle, se realiza un promedio de todas las medidas adquiridas y dicho valor se multiplica por 1000 para indicar que se está expresando su magnitud en milivoltios por metro (mV/m).

Con la data ya adquirida, el nodo fuente está listo para transmitir la información hacia el nodo sumidero. Usando la librería ***serial*** y ***gpiozero*** se configura la interfaz UART y el pin GPIO18 para comunicarse con el módulo Xbee.

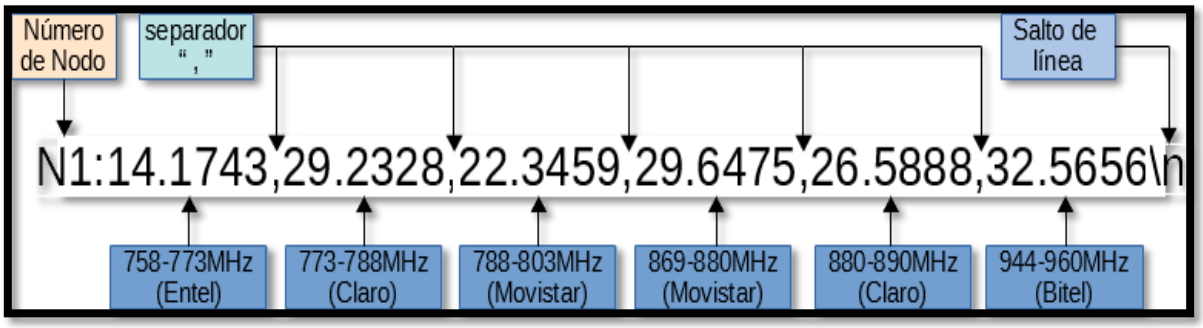
Usando la función ***despertar_xbee***, se despierta al módulo de su estado de hibernación y esperamos un tiempo para que se conecte a la red Zigbee. Luego se crea el mensaje que se enviará al nodo sumidero. Este mensaje es una cadena de texto que incluye el nombre del nodo y el valor de la lectura de intensidad de campo promedio.

Usando la función ***transmitir_xbee*** se envía la información usando la interfaz UART y el mensaje almacenado en la variable ***data***. Al finalizar la

función, se agrega un tiempo de espera para que el módulo envíe de manera exitosa el mensaje antes de terminar la aplicación y pasar a la siguiente etapa del ciclo activo.

La estructura del mensaje transmitido por el módulo Xbee es la siguiente:

FIGURA Nº4. 54
FORMATO DE LA TRAMA ENVIADA POR EL NODO FUENTE



La primera parte de la trama identifica al nodo transmisor. Seguido del indicador, están los valores medidos en cada banda de frecuencias analizada. Los valores están expresados en milivoltios por metro (mV/m) y son separados por comas.

El mensaje termina con un carácter de salto de línea para indicar su fin. La estructura de esta trama debe ser considerada en la programación del nodo sumidero, para ser interpretada de manera correcta.

Para probar el funcionamiento del script ***procesamiento.py*** se puede escribir el siguiente comando la consola de Linux del Raspberry Pi.

FIGURA Nº4. 55
COMANDO PARA EJECUTAR EL ARCHIVO PROCESAMIENTO.PY

```
1. python /home/pi/procesamiento.py
```

❖ **Programación de la Etapa de Apagado**

Esta es la etapa final del funcionamiento del nodo fuente. Debido a que los nodos fuente de la red de sensores usan una batería como fuente de energía, el ahorro de energía de los dispositivos en el nodo es un factor muy importante. En esta aplicación, el ahorro de energía se logrará cortando el voltaje que alimenta los dispositivos más relevantes, específicamente, el radio transceptor Xbee, el procesador Raspberry Pi y el sensor RTL-SDR. Debido a que el Raspberry Pi no cuenta con el hardware necesario para realizar el proceso de apagado y encendido de manera automática, se ha optado por usar otro microcontrolador, que se encargará de realizar esta tarea.

El microcontrolador ATtiny45 controlará una salida digital con un transistor que se encargará de encender o apagar los otros componentes del nodo. Este microcontrolador también posee una entrada digital, la cual está conectada al pin GPIO17 del Raspberry Pi. De esta manera el Raspberry Pi puede anunciar al microcontrolador exactamente cuándo se debe cortar su fuente de voltaje, esto ocurre después de haber completado la tarea principal de medición y transmisión de datos.

El script de la etapa de apagado se ha creado en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tiene el nombre de ***apagado.py***.

FIGURA Nº4. 56
PROGRAMACIÓN DE PYTHON DEL ARCHIVO APAGADO.PY

```
import time
from gpiozero import DigitalOutputDevice
import os

""" constante del pin de hibernación """
PIN_MCU = 17

""" funciones para el modo hibernación """
#Función para señalar al microcontrolador ATtiny45
def pulso_hibernacion(pin_mcu):
    pin_mcu.on()
    time.sleep(0.1)
    pin_mcu.off()


if __name__ == '__main__':
    #configurar el pin conectado al MCU como salida digital
    pin_mcu = DigitalOutputDevice(pin=PIN_MCU, initial_value=False)

    #dar señal al microcontrolador ATtiny45
    pulso_hibernacion(pin_mcu)
    pin_mcu.close() #desahibiltar la salida digital

    #apagar el nodo Raspberry pi
    os.system('sudo shutdown -h now')
```

La librería ***os*** incluye la función ***system***, que permite que el usuario ejecute un comando de consola Linux desde el script de Python. En este script, se usa el comando ***shutdown*** que es usado para apagar el Raspberry Pi de manera segura, antes de realizar el corte de energía.

La señal de hibernación que envía el Raspberry Pi al microcontrolador será un pulso digital, de 0.1 segundos, esto se ve en la función ***pulso_hibernacion*** y toma como argumento el objeto ***pin_mcu*** que es un pin de salida digital configurado en la sección principal del programa, usando el GPIO17.

Después de haber cortado la energía, el microcontrolador entrará en un modo de hibernación prolongado, activando un temporizador. El microcontrolador ATtiny45 será programado para despertar de su hibernación cuando el temporizador termine su cuenta y ejecute una interrupción. Este evento indica que el próximo ciclo activo del nodo ha iniciado, por lo tanto el microcontrolador volverá a encender los componentes principales del nodo y la operación se repite de manera periódica.

Para facilitar el manejo de registros, interrupciones y otros componentes de bajo nivel del microcontrolador, se usará el lenguaje de programación Arduino, que es compatible con muchas variantes de microcontroladores AVR, incluido el attiny45.

FIGURA Nº4. 57
PROGRAMACIÓN DE ARDUINO DEL ATTINY45

```
#include <tinysnore.h>

#define PIN_TRANSISTOR  0
#define PIN_HIBERNACION 1

#define encendido     0
#define procesamiento 1
#define apagado       2
#define hibernacion   3
int etapa = 0;

void setup()
{
  //Configura el pin de control de potencia como salida
  pinMode(PIN_TRANSISTOR, OUTPUT);
  digitalWrite(PIN_TRANSISTOR, LOW);

  //Configura el pin de hibernación del Raspberry como entrada
  pinMode(PIN_HIBERNACION, INPUT);
  digitalWrite(PIN_HIBERNACION, LOW);
}

void loop()
{
  //En la etapa de encendido, activar el transistor
  if(etapa == encendido)
  {
    digitalWrite(PIN_TRANSISTOR, HIGH);
```

```
    etapa = procesamiento;
    snore(25000); //esperar 25s para que inicie el RaspberryPi
  }

  //En la etapa de procesamiento, esperar el pulso de apagado
  if(etapa == procesamiento)
  {
    int pulso = pulseIn(PIN_HIBERNACION, HIGH);
    if( pulso >= 100000 ) //Pulso es medido en microsegundos
    { etapa = apagado;  }
  }

  //En la etapa de apagado, desactivar el transistor
  if(etapa == apagado)
  {
    snore(10000); //esperar 10s para que se apague el RaspberryPi
    digitalWrite(PIN_TRANSISTOR, LOW);
    etapa = hibernacion;
  }

  //En la etapa hibernación, entrar a modo hibernación idle
  if(etapa == hibernacion)
  {
    snore(86000000); //hibernar durante 86000 segundos
    etapa = encendido;  //reiniciar el ciclo
  }
}
```

La programación del ATtiny45 en Arduino está compuesta por 2 funciones principales. La función ***setup()*** es la primera en ejecutarse cuando el

dispositivo se enciende y en esta configuramos los pines de entrada y salida digital que usa el microcontrolador para controlar el transistor y comunicarse con el Raspberry Pi.

La función ***loop()*** es un bucle infinito, y en esta se encuentran las etapas del ciclo de operación del nodo, estas son controladas por una variable global definida como ***etapa*** en la primera parte de la programación.

La programación incluye la librería ***tinysnore.h***, la cual incluye la función ***snore()***. Esta función controla el modo de hibernación de baja potencia del microcontrolador ATtiny45. El argumento de esta función es un número entero de 32bits, que define cuánto tiempo el microcontrolador estará en hibernación. Una vez terminado ese periodo, el microcontrolador resume su funcionamiento secuencial.

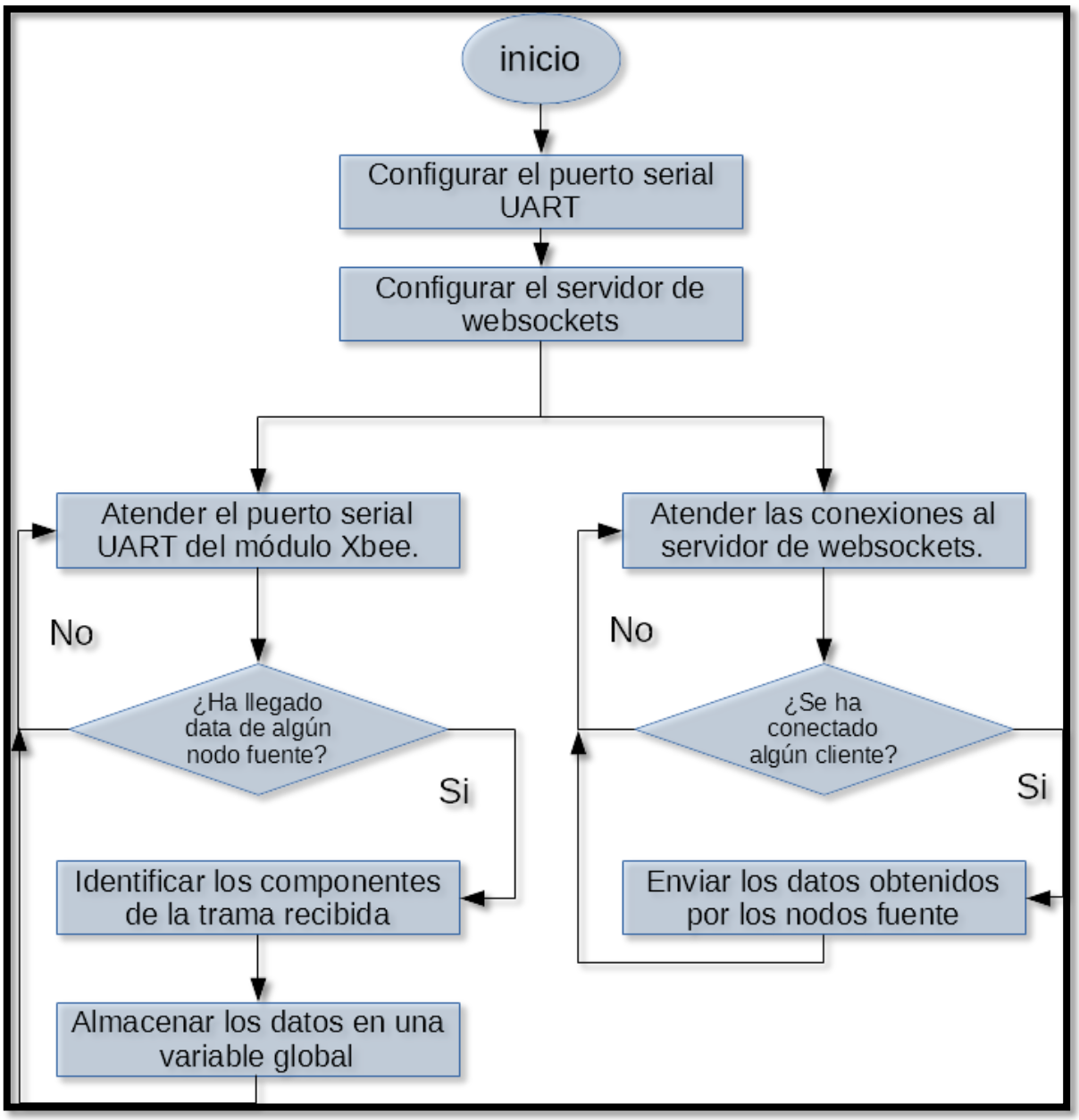
2.1.4.b. Firmware del nodo sumidero

El nodo sumidero es el elemento central de la red de sensores, este será el coordinador de red, por lo tanto, su función es crear y mantener la red Zigbee operativa. También debe recibir y agregar todos los mensajes de los nodos fuente que contienen la adquisición, procesamiento y cálculos de los niveles de radiación no ionizante en su vecindad. Este dispositivo actúa como un Gateway, para comunicar esta información con una red externa, donde otros dispositivos podrán visualizarla e interpretarla.

De manera similar al nodo fuente, el nodo sumidero usa un Raspberry Pi como su procesador, usando el sistema operativo Raspbian y un script en

lenguaje interpretado Python, que realizará la lectura de otros nodos, y conectar estos datos a una red externa a través de un servidor de websockets.

FIGURA Nº4. 58
DIAGRAMA DE FLUJO DEL NODO SUMIDERO



Para la programación de los componentes del nodo sumidero en el lenguaje Python, se deben instalar las siguientes librerías.

- Numpy: Es una librería para el lenguaje de programación Python que agrega funciones para manipular arreglos, matrices, funciones de álgebra lineal, transformadas de Fourier, etc.
- Pyserial: Es una librería para el lenguaje de programación Python que permite acceder al puerto serial del Raspberry pi, a través de su interfaz física o el puerto USB.
- Websocket-Server: Es una librería para el lenguaje de programación Python que incluye funciones para implementar un servidor de websockets.

En el terminal de comandos del Raspberry pi del nodo sumidero, conectado a internet, se han ingresaremos los siguientes comandos.

FIGURA Nº4. 59
COMANDOS EN EL TERMINAL DEL RASPBERRY

```
1. sudo apt-get update
2. sudo apt-get install python-pip
3. sudo pip install pyserial
4. sudo pip install numpy
5. sudo pip install websocket-server
```

Para verificar que todas las librerías han sido instaladas correctamente, se inició una consola de Python en el Raspberry Pi y se escribieron las siguientes funciones.

FIGURA Nº4. 60
VERIFICACION DE TODAS LAS LIBRERÍAS

```
1. import numpy
2. import serial
3. import websocket_server
```

Si la instalación fue exitosa, la ejecución de dichos códigos no retornará ningún error.

❖ **Programación de la Etapa de Encendido**

Cuando el nodo sumidero es encendido, el Raspberry Pi demorará aproximadamente 25 segundos en iniciar su sistema operativo. Cuando el sistema operativo Raspbian esté listo para funcionar, el Raspberry Pi deberá iniciar sus funciones de manera automática y sin intervención de algún usuario.

El comportamiento del Raspberry Pi del nodo sumidero es establecido en un script de Python, que será llamado ***nodoSumidero.py*** y este es almacenado en el directorio principal con dirección ***/home/pi.***

De manera similar al inicio del nodo fuente, en el nodo sumidero usa la q herramienta Cron para que este archivo se ejecute de manera automática cada vez que el Raspberry Pi inicie el sistema operativo. Para esta aplicación se ha usado el prefijo @reboot que permite ejecutar funciones al iniciar el sistema operativo.

Primero se edita el archivo crontab en el Raspberry Pi ingresando el siguiente en la consola de comandos.

FIGURA Nº4. 61
COMANDO PARA EDITAR EL ARCHIVO CRONTAB DEL RASPBERRY PI

```
1. crontab -e
```

Finalmente se agrega la siguiente línea en la última parte del archivo crontab para hacer que el script ***nodoSumidero.py*** se ejecute a la hora de iniciar el sistema.

FIGURA Nº4. 62
COMANDO PARA EJECUTAR EL ARCHIVO NODOSUMIDERO.PY AL INICIAR EL OS.

```
1. @reboot python /home/pi/nodoSumidero.py
```

Después de guardar los cambios al archivo crontab y reiniciar el Raspberry Pi, vemos que cada vez que se inicie el sistema operativo, los comandos del archivo ***nodoSumidero.py*** se ejecutarán de manera automática en la secuencia especificada.

- **Programación de la Configuración del radio Xbee en el nodo sumidero**

Esta configuración solo debe ejecutarse una vez y definirán parámetros del módulo Xbee para funcionar como coordinador de red.

El módulo Xbee se conecta al Raspberry Pi a través de su puerto UART en los pines GPIO14 (TXD) y GPIO15 (RXD). Para enviar los comandos de configuración del módulo Xbee, este debe operar en el modo de comandos AT, el cuál es accesible desde la interfaz UART.

La conexión serial y los comandos serán enviados a través del script de Python usando las librerías instaladas previamente. Este script se guardará en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tendrá el nombre de ***config_nodo_sumidero.py***.

FIGURA Nº4. 63
PROGRAMACIÓN DE PYTHON DEL ARCHIVO CONFIG_NODO_SUMIDERO.PY

```
1.  import serial
2.  import time
3.  from gpiozero import DigitalOutputDevice
4.
5.  """ constantes del módulo xbee """
6.  PUERTO_XBEE = 'dev/ttyAMA0'
7.  PIN_XBEE = 18
8.
9.  """ funciones del módulo xbee """
10. #Función para despertar el módulo Xbee de hibernación
11. def despertar_xbee(pin_xbee):
12.     pin_xbee.off()
13.
14. #Función para transmitir datos seriales al módulo Xbee
15. def transmitir_xbee(modulo_xbee, data):
16.     modulo_xbee.write(data) #enviar la data
17.
```

```
#Función para leer la respuesta del módulo Xbee
def leer_xbee(modulo_xbee):
    respuesta = modulo_xbee.readline()
    return respuesta


if __name__ == '__main__':
    #configurar el puerto serial y el GPIO18
    modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrate=9600)
    pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_value=True)

    despertar_xbee(pin_xbee)
    time.sleep(1)

    #iniciar modo comandos AT
    transmitir_xbee(modulo_xbee, '+++')
    respuesta = leer_xbee(modulo_xbee)

    #restaurar configuración por defecto
    transmitir_xbee(modulo_xbee, 'ATRE\r')
    respuesta = leer_xbee(modulo_xbee)

    #configurar dispositivo final
    transmitir_xbee(modulo_xbee, 'ATCE 1\r')
    respuesta = leer_xbee(modulo_xbee)

    #configurar el PAN ID
    transmitir_xbee(modulo_xbee, 'ATID 2018\r')
    respuesta = leer_xbee(modulo_xbee)
```

```
47.
48.     #Potencia de Transmisión = 5dBm
49.     transmitir_xbee(modulo_xbee, 'ATPL 4\r')
50.     respuesta = leer_xbee(modulo_xbee)
51.
52.     #Modo boost activado (Tx+3dB, Rx+2dB)
53.     transmitir_xbee(modulo_xbee, 'ATPM 1\r')
54.     respuesta = leer_xbee(modulo_xbee)
55.
56.     #Modo hibernación desactivado
57.     transmitir_xbee(modulo_xbee, 'ATSM 0\r')
58.     respuesta = leer_xbee(modulo_xbee)
59.
60.     #aplicar cambios
61.     transmitir_xbee(modulo_xbee, 'ATAC\r')
62.     respuesta = leer_xbee(modulo_xbee)
63.
64.     #escribir a memoria no volatil
65.     transmitir_xbee(modulo_xbee, 'ATWR\r')
66.     respuesta = leer_xbee(modulo_xbee)
67.
68.     #reinicia el dispositivo
69.     transmitir_xbee(modulo_xbee, 'ATFR\r')
70.     respuesta = leer_xbee(modulo_xbee)
71.
72.     #esperar 3 segundos para que se reinicie
73.     time.sleep(3)
74.
75.     #poner el módulo xbee en hibernación
76.     pin_xbee.close()
```

Las librerías ***serial*** (proveniente de pyserial), **time** y **gpiozero** serán usadas para configurar el puerto UART del Raspberry Pi, generar tiempos de espera y configurar los pines GPIO como salidas digitales para comunicarse con el módulo Xbee, respectivamente.

La conexión del módulo Xbee al Raspberry Pi en el nodo sumidero es igual a la conexión mostrada en el nodo fuente. Los pines GPIO14 (TXD) y GPIO15 (RXD) del Raspberry Pi son conectados a los pines UART del módulo Xbee y el pin GPIO18 del Raspberry Pi está conectado al pin SLEEP_RQ del Xbee para controlar el modo hibernación del módulo. El pin SLEEP_RQ mantiene al módulo en hibernación cuando está conectado a 3.3V y despierta de hibernación cuando es conectado a 0V.

En la sección principal (main) se realiza la configuración del puerto serial y del pin GPIO18 que se conecta al pin SLEEP_RQ del módulo Xbee y una vez terminada la configuración, se realiza la transmisión de comandos AT para configurar los parámetros de operación del módulo Xbee en el nodo sumidero.

Después de configurar todos los parámetros necesarios, se ejecuta el comando *ATWR*, para escribir los valores en memoria no volátil y el comando *ATFR*, para reiniciar el módulo, sacarlo de modo de comandos AT y ahora el módulo Xbee operará con los cambios aplicados.

No es necesario ejecutar este script de configuración cada vez que se enciende el nodo, pues la información será almacenada en la memoria no volátil del radio Xbee.

Para probar el funcionamiento del script ***config_nodo_sumidero.py*** se puede escribir el siguiente comando la consola de Linux del Raspberry Pi.

FIGURA Nº4. 64
COMANDO PARA EJECUTAR EL ARCHIVO CONFIG_NODO_SUMIDERO.PY

```
1. python /home/pi/config_nodo_sumidero.py
```

❖ **Programación de Recepción de datos en el nodo sumidero**

Esta es la tarea principal del nodo sumidero y es ejecutada al iniciar el sistema operativo del Raspberry Pi. El nodo sumidero tiene su módulo Xbee actuando como coordinador de la red Zigbee, por lo tanto este se encarga de iniciar y mantener la red. Esto requiere que el nodo sumidero tenga una fuente de voltaje continuo y no entrará en modo de hibernación, a diferencia de los otros nodos en la red. El Raspberry Pi en el nodo sumidero tiene la función de leer y agregar los mensajes recibidos por los nodos fuente a través de la red Zigbee. Para esta función se usará la interfaz UART y será controlada por la librería ***serial*** en un script de Python. La otra función del Raspberry Pi en el nodo sumidero es de actuar como un dispositivo Gateway, que comunica la información recibida con una red externa. Esta función se realizará programando un servidor de websockets usando la librería ***websocket_server***. Los clientes que se conecten a este servidor recibirán los valores de las mediciones obtenidas de los nodos fuente.

Hemos concluido que el Raspberry Pi dentro del nodo sumidero tiene dos funciones muy importantes, que deben ser atendidas de manera asíncrona,

pues este dispositivo no tiene conocimiento de previo de cuándo llegará la lectura de un nodo fuente o cuándo se conectará un nuevo cliente al servidor para recibir los datos de la red. Para manejar esta situación se ha usado la librería ***threading*** que permite realizar tareas que se ejecutan de manera simultánea.

Este script se ha creado en el directorio principal del Raspberry Pi, el cual tiene la dirección ***/home/pi*** y este script tiene el nombre de ***nodoSumidero.py***.

FIGURA Nº4. 65
PROGRAMACIÓN DE PYTHON DEL ARCHIVO NODOSUMIDERO.PY

```
1.  import websocket_server
2.  import serial
3.  from gpiozero import DigitalOutputDevice
4.  import numpy as np
5.  import json
6.  import threading
7.
8.  """ datos de los nodos de la red y bandas de frecuencias"""
9.  nodo1 = {'ID': 'I.E Jardin Municipal',
10.         'gps': [-12.0744333,-77.16321666666667],
11.         'banda1':['Entel',     758, 765.5, 773, 0],
12.         'banda2':['Claro',     773, 780.5, 788, 0],
13.         'banda3':['Movistar',  788, 795.5, 803, 0],
14.         'banda4':['Movistar',  869, 874.5, 880, 0],
15.         'banda5':['Claro',     880, 885, 890, 0],
16.         'banda6':['Bitel',     944, 952, 960, 0]}
17.
```

```
nodo2 = {'ID': 'I.E. Parroquia Clara Cogorno de Cogorno',
        'gps': [-12.0739083,-77.16340277777779],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodo3 = {'ID': 'I.E Mini Skool',
        'gps': [-12.0716139,-77.16446944444445],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodo4 = {'ID': 'Centro de Salud La Punta',
        'gps': [-12.0684583,-77.15809166666668],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodos = {'N1':nodo1,
        'N2':nodo2,
        'N3':nodo3,
        'N4':nodo4 }
```

```
49.
50.
51. """ constantes del módulo xbee """
52. PUERTO_XBEE = 'dev/ttyAMA0'
53. PIN_XBEE = 18
54.
55. """ funciones del módulo xbee """
56. #Función para despertar el módulo Xbee de hibernación
57. def despertar_xbee(pin_xbee):
58.     pin_xbee.off()
59.
60. #Función para leer el UART Xbee de manera continua
61. def leer_xbee(modulo_xbee):
62.     while(True):
63.         respuesta = modulo_xbee.readline()  #fmt: 'Nx:E1,E2,E
    3,E4,E5,E6\n'
64.         nodo = respuesta[:2]
65.         lecturas = np.fromstring(respuesta[3:], sep=',')
66.         nodos[nodo]['banda1'][4] = lecturas[0]
67.         nodos[nodo]['banda2'][4] = lecturas[1]
68.         nodos[nodo]['banda3'][4] = lecturas[2]
69.         nodos[nodo]['banda4'][4] = lecturas[3]
70.         nodos[nodo]['banda5'][4] = lecturas[4]
71.         nodos[nodo]['banda6'][4] = lecturas[5]
72.
73.
74. """ constantes del servidor """
75. PUERTO = 8888          #puerto donde operará el servidor
76. DIRECCION = '0.0.0.0'   #permite que otros clientes se con
    ecten
77.
```

```
78. """ funciones del servidor """
79. #Cuando un nuevo cliente se conecta al servidor
80. def nuevo_cliente(client, server):
81.     mensaje = json.dumps(nodos)
82.     server.send_message(client, mensaje)
83.
84.
85. if __name__ == '__main__':
86.     #configurar el puerto serial y el GPIO18
87.     modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrate=9600)
88.     pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_value=True)
89.     despertar_xbee(pin_xbee)    #sacar al xbee de modo hibernacion
90.     #iniciar el hilo de lectura serial
91.     threading.Thread(target=leer_xbee, args=(modulo_xbee,)).start()
92.
93.     #Configura e inicia el servidor de websockets
94.      servidor = websocket_server.WebsocketServer(port=PUERTO, host=DIRECCION)
95.     servidor.set_fn_new_client(nuevo_cliente)
96.     #iniciar el bucle del servidor de websockets
97.     servidor.run_forever()
```

La primera parte del script es la inclusión de librerías

FIGURA Nº4. 66
INCLUSIÓN DE LIBRERÍAS DEL ARCHIVO NODOSUMIDERO.PY

```
1. import websocket_server
2. import serial
3. from gpiozero import DigitalOutputDevice
4. import numpy as np
5. import json
6. import threading
```

La librería ***websocket_server*** es usada para iniciar un servidor de websockets en el Raspberry Pi.

La librería ***serial*** (proveniente de pyserial) es usada para configurar el puerto UART del Raspberry Pi y permitirá establecer la comunicación entre este y el módulo Xbee.

La librería ***gpiozero*** incluye el objeto ***DigitalOutputDevice***, que permite configurar los pines GPIO como salidas digitales que generarán valores de 3.3V (estado ON o HIGH) y 0V (estado OFF o LOW).

La librería ***numpy*** incluye funciones para realizar operaciones y manipular arreglos, en el archivo, se usará el prefijo ***np*** para usar las funciones de esta librería.

La librería ***json*** permite usar el formato de notación de objetos de javascript para enviar datos a través del servidor.

La librería ***threading*** es usada para realizar tareas de manera simultánea.

Después de la inclusión de librerías, se han definido las variables que contienen los datos de los nodos fuente en la red de sensores.

FIGURA Nº4. 67
ESTRUCTURAS DE DATOS PARA LOS MENSAJES RECIBIDOS

```
""" datos de los nodos de la red y bandas de frecuencias"""
nodo1 = {'ID': 'I.E Jardin Municipal',
        'gps': [-12.0744333,-77.16321666666667],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodo2 = {'ID': 'I.E. Parroquia Clara Cogorno de Cogorno',
        'gps': [-12.0739083,-77.16340277777779],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodo3 = {'ID': 'I.E Mini Skool',
        'gps': [-12.0716139,-77.16446944444445],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

```

```
nodo4 = {'ID': 'Centro de Salud La Punta',
        'gps': [-12.0684583,-77.15809166666668],
        'banda1':['Entel',     758, 765.5, 773, 0],
        'banda2':['Claro',     773, 780.5, 788, 0],
        'banda3':['Movistar',  788, 795.5, 803, 0],
        'banda4':['Movistar',  869, 874.5, 880, 0],
        'banda5':['Claro',     880, 885, 890, 0],
        'banda6':['Bitel',     944, 952, 960, 0]}

nodos = {'N1':nodo1,
        'N2':nodo2,
        'N3':nodo3,
        'N4':nodo4 }
```

La variable de cada nodo es un diccionario, que incluye información como la referencia del lugar donde se realiza la medición, las coordenadas de GPS del nodo, y los valores de intensidad de campo eléctrico en cada banda de frecuencias.

TABLA Nº4. 15
ESTRUCTURA DEL DICCIONARIO DE PYTHON DE LAS VARIABLES “NODO”

Estructura de las variables nodo1, nodo2, nodo3 y nodo4					
Índices	**Contenidos del diccionario**				
‘ID’	Lugar de referencia del nodo				
‘gps’	Coordenadas de GPS del nodo (latitud, longitud)				
'banda1'	Operador (Entel)	Frecuencia inicial en MHz (758)	Frecuencia central en MHz (765.5)	Frecuencia final en MHz (773)	Intensidad de campo eléctrico.

'banda2'	Operador (Claro)	Frecuencia inicial en MHz (773)	Frecuencia central en MHz (780.5)	Frecuencia final en MHz (788)	Intensidad de campo eléctrico.
'banda3'	Operador (Movistar)	Frecuencia inicial en MHz (788)	Frecuencia central en MHz (795.5)	Frecuencia final en MHz (803)	Intensidad de campo eléctrico.
'banda4'	Operador (Movistar)	Frecuencia inicial en MHz (869)	Frecuencia central en MHz (874.5)	Frecuencia final en MHz (880)	Intensidad de campo eléctrico.
'banda5'	Operador (Claro)	Frecuencia inicial en MHz (880)	Frecuencia central en MHz (885)	Frecuencia final en MHz (890)	Intensidad de campo eléctrico.
'banda6'	Operador (Bitel)	Frecuencia inicial en MHz (944)	Frecuencia central en MHz (952)	Frecuencia final en MHz (960)	Intensidad de campo eléctrico.

Toda esta información se agrega en una variable global del tipo diccionario llamada ***nodos***, la cual es usada en las funciones de recepción de datos del módulo Xbee y en la transmisión de datos hacia los clientes que se conectan al servidor de websockets.

El script también incluye las constantes y funciones del módulo Xbee que son usadas para leer la data enviada por los nodos fuente.

FIGURA Nº4. 68
FUNCIONES PARA EL MÓDULO XBEE DEL ARCHIVO NODOSUMIDERO.PY

```
51. """ constantes del módulo xbee """
52. PUERTO_XBEE = 'dev/ttyAMA0'
53. PIN_XBEE = 18
```

```
54.
55. """ funciones del módulo xbee """
56. #Función para despertar el módulo Xbee de hibernación
57. def despertar_xbee(pin_xbee):
58.     pin_xbee.off()
59.
60. #Función para leer el UART Xbee de manera continua
61. def leer_xbee(modulo_xbee):
62.     while(True):
63.         respuesta = modulo_xbee.readline()  #fmt: 'Nx:E1,E2,E3,E4,E5,
    E6\n'
64.         nodo = respuesta[:2]
65.         lecturas = np.fromstring(respuesta[3:], sep=',')
66.         nodos[nodo]['banda1'][4] = lecturas[0]
67.         nodos[nodo]['banda2'][4] = lecturas[1]
68.         nodos[nodo]['banda3'][4] = lecturas[2]
69.         nodos[nodo]['banda4'][4] = lecturas[3]
70.         nodos[nodo]['banda5'][4] = lecturas[4]
71.         nodos[nodo]['banda6'][4] = lecturas[5]
72.
```

La variable ***PUERTO_XBEE*** contiene una cadena de texto que almacena la dirección que el Raspberry Pi usa para acceder el puerto serial UART con interfaz en los pines GPIO14 y GPIO15.

La variable ***PIN_XBEE*** contiene el valor 18, haciendo referencia al pin GPIO18 del Raspberry Pi que está conectado al pin SLEEP_RQ del Xbee para controlar el modo hibernación del módulo. El pin SLEEP_RQ mantiene al módulo en hibernación cuando está conectado a 3.3V y despierta de

hibernación cuando es conectado a 0V. En nuestro diagrama de conexiones, este pin está conectado a un resistor pull-up, el cual mantiene el dispositivo en modo hibernación por defecto. Para la operación del nodo sumidero, el Raspberry Pi despertará el módulo de hibernación y lo mantendrá activo durante todo el resto de la operación.

La función ***despertar_xbee*** toma como argumento a ***pin_xbee***, este será un objeto que controla el pin GPIO18 del Raspberry Pi y será definido en la sección principal (main) del script usando la librería ***gpiozero***. Esta función despertará al módulo Xbee de su estado de hibernación para leer los datos por el puerto serial UART.

La función ***leer_xbee*** incluye un bucle infinito, en el cual el Rasbperry Pi se mantendrá atento a su puerto serial UART para recibir los datos enviados por el módulo Xbee. Una vez recibidos los datos de algún nodo fuente en la red, esta función descompone el mensaje para identificar el nodo transmisor y las mediciones de intensidad de campo en cada banda de frecuencia. Después de separar estos valores, la función almacena dicha información en la variable ***nodos***.

Las constantes y función del servidor son usadas para configurar, iniciar el servidor y responder a los clientes que se conectan al Raspberry Pi a través de un websocket.

FIGURA Nº4. 69
FUNCIONES PARA EL SERVIDOR WEBSOCKETS DEL ARCHIVO NODOSUMIDERO.PY

```
74. """ constantes del servidor """
75. PUERTO = 8888          #puerto donde operará el servidor
76. DIRECCION = '0.0.0.0'   #permite que otros clientes se conecten
77.
78. """ funciones del servidor """
79. #Cuando un nuevo cliente se conecta al servidor
80. def nuevo_cliente(client, server):
81.     mensaje = json.dumps(nodos)
82.     server.send_message(client, mensaje)
```

La variable ***PUERTO*** contiene un valor numérico que indica en qué puerto TCP del Raspberry Pi se implementará el servidor de websockets

La variable ***DIRECCION*** contiene una cadena de texto con la dirección IP 0.0.0.0. Este valor indica que el servidor se implementará en la dirección local del Raspberry Pi y que aceptará conexiones de clientes que se encuentren en la misma red.

La función ***nuevo_cliente*** es una función que se ejecutada cada vez que un nuevo cliente se conecta al servidor de websockets. El Raspberry Pi le enviará un mensaje que contiene la data de la variable global ***nodos***, que contiene toda la información obtenida de los nodos fuente. Este mensaje se encuentra en formato JSON, el cual es un formato de texto que es independiente al lenguaje de programación, pero es fácilmente interpretado por otras aplicaciones usando otros lenguajes como C, C++, Javascript,

Perl, etc. Esto facilitará el diseño de una interfaz para visualizar la información enviada por este servidor.

En la sección principal (main) del script, es donde se utilizan estas funciones y constantes para realizar la tarea principal del nodo sumidero.

FIGURA Nº4. 70
SECCIÓN PRINCIPAL DEL ARCHIVO NODOSUMIDERO.PY

```
85. if __name__ == '__main__':
86.     #configurar el puerto serial y el GPIO18
87.     modulo_xbee = serial.Serial(port=PUERTO_XBEE, baudrate=9600
    )
88.     pin_xbee = DigitalOutputDevice(pin=PIN_XBEE, initial_value=True
    )
89.     despertar_xbee(pin_xbee)    #sacar al xbee de modo hibernacion
90.     #iniciar el hilo de lectura serial
91.     threading.Thread(target=leer_xbee, args=(modulo_xbee,)).start()
92.
93.     #Configura e inicia el servidor de websockets
94.     servidor = websocket_server.WebsocketServer(port=PUERTO, ho
    st=DIRECCION)
95.     servidor.set_fn_new_client(nuevo_cliente)
96.     #iniciar el bucle del servidor de websockets
97.     servidor.run_forever()
```

En la sección principal del programa se inicia con la configuración del módulo Xbee.

Usando la librería ***serial*** y ***gpiozero*** se configura la interfaz UART y el pin GPIO18 para comunicarse con el módulo Xbee.

Usando la función ***despertar_xbee***, despertamos al módulo de su estado de hibernación. A partir de ahora, el módulo Xbee actuará como coordinador de la red Zigbee y permanecerá en estado activo y será capaz de recibir datos de otros nodos que se conecten a su red. Para leer los datos recibidos por la interfaz UART, usaremos la librería ***threading*** para ejecutar la función ***leer_xbee*** en un hilo paralelo a la secuencia del programa principal.

Después de haber iniciado la interfaz UART, se configura el servidor de websockets en el Raspberry Pi, usando la librería ***websocket_server***. Se crea el objeto de referencia al servidor usando como argumentos, los valores de las variables ***PUERTO*** y ***DIRECCION*** definidos anteriormente.

Luego se usa el método ***set_fn_new_client*** para asignar la función ***nuevo_cliente*** como un callback, de manera que cada vez que un cliente se conecte al servidor, esta función se ejecutará.

Finalmente, para iniciar el servidor con el método ***run_forever***, el cual inicia un bucle infinito que atenderá el servidor websockets.

Para probar el funcionamiento del script ***nodoSumidero.py*** se puede escribir el siguiente comando la consola de Linux del Raspberry Pi.

FIGURA Nº4. 71
COMANDO PARA EJECUTAR EL ARCHIVO NODOSUMIDERO.PY

```
1. python /home/pi/nodoSumidero.py
```

2.1.5 Diseño de la interfaz para visualizar resultados

Con el diseño del hardware y software de los nodos de la red de sensores, la última etapa de esta investigación, consiste en visualización de los datos adquiridos.

El nodo sumidero actúa como Gateway de la red de sensores y permite que dispositivos en una red LAN accedan a la información de los nodos fuente.

Para esta función el nodo sumidero realiza la función de un servidor de websockets, programado para enviar dicha información a los clientes que se conecten.

Debido a las limitaciones del hardware y software del nodo sumidero, la red LAN externa deberá incluir componentes como firewalls, routers y otros para evitar la sobrecarga, posibles ataques al servicio del nodo sumidero y para brindar la información a otros usuarios que se encuentran en otras redes y a través de internet. El diseño de la red LAN externa no es parte de esta investigación.

Para los usuarios que inicien un cliente de websockets para conectarse al servidor del nodo sumidero, podrán visualizar la data recibida usando una interfaz Web diseñada con HTML5.

HTML5 es la última versión del lenguaje marcado de hipertexto o HTML. HTML el lenguaje más básico de la web. Este lenguaje describe y define el contenido y la estructura de los elementos de una página web. El desarrollo del lenguaje HTML5 es regulado por el grupo W3C. La parte de hipertexto se refiere a los enlaces que conectan una página web con otra. La parte de lenguaje marcado se refiere al sintaxis de los elementos que conforman una página web, por ejemplo <title>, <head>, , etc. El archivo HTML5 será guardado con el formato .html.

Junto al diseño de la web con HTML5 se usará el lenguaje de hojas de estilo en cascada o CSS. CSS el lenguaje de diseño gráfico que describe la presentación de los elementos de la página web, definiendo colores, tipos de letra, dimensiones, etc. El archivo CSS se crea de manera separada y se guarda con el formato .css.

La programación de los eventos y elementos de la web se realizarán con el lenguaje Javascript. Javascript es un lenguaje de programación orientada a objetos que permite agregar funciones interactivas y complejas a una página web, por ejemplo, interactuar con imágenes, tablas, mapas, etc. Un archivo de Javascript se guarda con el formato .js.

Para la interfaz de visualización se han creado los 3 archivos necesarios. El archivo interfaz.html contiene todos los elementos de la página web y su estructura, además del enlace hacia los otros archivos necesarios. El archivo interfaz.css configurará la presentación de los componentes en la página web y el archivo interfaz.js contiene la programación necesaria para iniciar el

cliente websocket y mostrar la información en los elementos del archivo HTML.

FIGURA Nº4. 72
PROGRAMACIÓN DE HTML DEL ARCHIVO INTERFAZ.HTML

```
1.  <html>
2.  <head>
3.  <script src="interfaz.js"></script>
4.  <link rel="stylesheet" href="interfaz.css">
5.     <title> Intefaz Web de Visualización de data </title>
6.     <h1> Red de Sensores para Monitoreo de RNI en Colegios Y Hos
    pitales del Distrito de la Punta</h1>
7.     <body>
8.        <div id="mapa"></div>
9.        <script src="https://maps.googleapis.com/maps/api/js?key=LLA
    VE_API&callback=iniciar_mapa"></script>
10.       </br>
11.       <table id="tabla">
12.          <tr>  <th>Referencia</th>     <td id="ref" colspan="4"></td>
    </tr>
13.          <tr>  <th>Coordenadas</th>    <td id="gps" colspan="4"></t
    d>  </tr>
14.          <tr>  <th>Banda</th>  <th>F.Central</th>  <th>Operador</t
    h>   <th>Intensidad de Campo (E)</th>    <th>Comparación</th>
    </tr>
15.          <tr>  <td id="b1"></td>   <td id="fc1"></td>  <td id="op1"></t
    d>  <td id="E1"></td>   <td id="pct1"></td> </tr>
16.          <tr>  <td id="b2"></td>   <td id="fc2"></td>  <td id="op2"></t
    d>  <td id="E2"></td>   <td id="pct2"></td> </tr>
17.          <tr>  <td id="b3"></td>   <td id="fc3"></td>  <td id="op3"></t
```

```
d>  <td id="E3"></td>   <td id="pct3"></td> </tr>
18.         <tr>  <td id="b4"></td>   <td id="fc4"></td>  <td id="op4"></t
    d>  <td id="E4"></td>   <td id="pct4"></td> </tr>
19.         <tr>  <td id="b5"></td>   <td id="fc5"></td>  <td id="op5"></t
    d>  <td id="E5"></td>   <td id="pct5"></td> </tr>
20.         <tr>  <td id="b6"></td>   <td id="fc6"></td>  <td id="op6"></t
    d>  <td id="E6"></td>   <td id="pct6"></td> </tr>
21.    </body>
22. </head>
23. </html>
```

El archivo intefaz.html de la interfaz de visualización consiste de un mapa, que mostrará la ubicación de los nodos de la red de sensores según la información enviada por el nodo sumidero, y una tabla donde se muestran los detalles de las mediciones realizadas por los nodos fuente.

La primera parte del archivo interfaz.html usa la sintaxis HTML para indicar cuales son los archivos CSS y JS que se usarán para la interfaz web.

El mapa de la interfaz usa el servicio de Google Maps usando su API para mostrar la ubicación de los nodos usando marcadores. El mapa requiere de una función callback para configurar sus parámetros, esta función fue llamada ***iniciar_mapa*** y será definida en el archivo de javascript. En el archivo HTML, el mapa se encuentra dentro de un divisor con identificador “mapa”. Este identificador será usado en los archivos CSS y JS para modificar los elementos del mapa.

La tabla de datos donde se mostrará la información de los nodos fuente, posee filas donde se detalla la ubicación de referencia del nodo, sus

coordenadas GPS y columnas para separar la frecuencia central, operador de servicio de telecomunicaciones, lectura de intensidad de campo eléctrico y comparación con los Límites Máximos Permitidos en cada banda de frecuencia que analiza el nodo.

La tabla de datos tiene un identificador llamado “tabla” y será usado en los archivos CSS y JS.

FIGURA Nº4. 73
PROGRAMACIÓN DE CSS DEL ARCHIVO INTERFAZ.CSS

```
#mapa
{
    width: 100%;
    height: 400px;
    color: black;
}

#tabla
{
    border-collapse: collapse;
    display: none;
}

td, th
{
    border: 1px solid #dddddd;
    text-align: center;
    padding: 8px;
}
```

El archivo intefaz.css de la interfaz de visualización contiene las características externas de los componentes del mapa y la tabla que fueron definidos en el archivo interfaz.html.

El elemento #mapa hace referencia al divisor con identificador “mapa” y define las dimensiones de este elemento y el color de fondo negro que se usará en la aplicación.

El elemento #tabla hace referencia a la tabla con identificador “tabla”. El estilo display:none hará que la tabla sea invisible al iniciar la aplicación. De manera que el usuario no verá la tabla vacía. Este comportamiento se puede modificar de manera dinámica en el archivo JS.

Los elementos td, th configuran el estilo de las celdas y cabeceras de la tabla. Aquí se configura la alineación de texto y el color y anchura de los bordes de la tabla.

FIGURA Nº4. 74
PROGRAMACIÓN DE JAVASCRIPT DEL ARCHIVO INTERFAZ.JS

```
1. var ws;            //variable de la conexión websockets
2.
3. function iniciar_mapa()
4. {
5.     //define las coordenadas del centro del mapa
6.     var centro = {lat: -12.0717667, lng: -77.1614638888889};
7.     //inicia el mapa con el API de google
8.     var mapa = new google.maps.Map( document.getElementById('m
   apa'),
```

```
9.          {zoom: 16, center: centro} );
10.    mapa.setMapTypeId('satellite'); //establece el tipo de mapa
11.
12.    //define ubicación del nodo sumidero
13.    var sumidero = {lat: -12.0727667, lng: -77.1634638888889};
14.    //crea marcador para el nodo sumidero
15.    var marcador_sumidero = new google.maps.Marker({position:sumi
    dero, map:mapa});
16.    //selecciona ícono para el marcador
17.    marcador_sumidero.setIcon('http://maps.google.com/mapfiles/ms/i
    cons/green-dot.png');
18.    //crea ventana de información para el nodo sumidero
19.    var info = new google.maps.InfoWindow({ content: "Nodo Sumider
    o" });
20.    //adjunta la ventana de info al nodo sumidero
21.    marcador_sumidero.addListener('click', function()
22.    { info.open(mapa, marcador_sumidero); } );
23.
24.    //inicio de la conexión al servidor websocket
25.    ws = new WebSocket("ws://192.168.0.10:8888/");
26.    //establece la función de recepción de datos por websockets
27.    ws.onmessage = function(e)
28.    {   var nodos = JSON.parse(e.data); //decodificar la data en format
    o JSON
29.        //iterando sobre la información adquirida en la variable nodos
30.        for (var N in nodos)
31.        {   //seleccionar un nodo en la lista adquirida
32.            var nodo = nodos[N]
33.            //obtiene las coordenadas de GPS del nodo fuente
34.            var posicion = {lat: nodo.gps[0], lng: nodo.gps[1]};
35.            //crear un marcador para el nodo fuente
```

```
36.         var marcador = new google.maps.Marker({position:posicion,
    map:mapa});
37.         //selecciona ícono para el marcador
38.         marcador.setIcon('http://maps.google.com/mapfiles/ms/icons/
    blue-dot.png');
39.         //crea ventana de información para el nodo fuente
40.         var info = new google.maps.InfoWindow({content: nodo.ID});
41.         //Función
    cuando el usuario haga click en el marcador del nodo:
42.         marcador.addListener('click', function()
43.         {   //mostrar la ventana de información
44.             info.open(mapa, marcador);
45.             //mostrar la tabla oculta
46.             document.getElementById('tabla').style.display="block";
47.             //mostrar la información del nodo en la tabla
48.             document.getElementById('ref').innerHTML = nodo.ID;
49.             //mostrar las coordenadas del nodo en la tabla
50.             document.getElementById('gps').innerHTML = nodo.gps;
51.             //mostrar la información de la banda de frecuencias 1 en la
    tabla
52.             var LMP = 26975.98; //Limite maximo permisible en la ban
    da1
53.             document.getElementById('b1').innerHTML = nodo.banda1[
    1]+'-'+nodo.banda1[3]+'MHz';
54.             document.getElementById('fc1').innerHTML = nodo.banda1
    [2]+'MHz';
55.             document.getElementById('op1').innerHTML = nodo.banda
    1[0];
56.             document.getElementById('E1').innerHTML = nodo.banda1
    [4]+' mv/m';
57.             document.getElementById('pct1').innerHTML = (nodo.band
```

```
a1[4]/LMP*100)+"%";
58.             //mostrar la información de la banda de frecuencias 2 en la tabla
59.             var LMP = 27238.99; //Limite maximo permisible en la banda2
60.             document.getElementById('b2').innerHTML = nodo.banda2[1]+'-'+nodo.banda2[3]+'MHz';
61.             document.getElementById('fc2').innerHTML = nodo.banda2[2]+'MHz';
62.             document.getElementById('op2').innerHTML = nodo.banda2[0];
63.             document.getElementById('E2').innerHTML = nodo.banda2[4]+' mv/m';
64.             document.getElementById('pct2').innerHTML = (nodo.banda2[4]/LMP*100)+"%";
65.             //mostrar la información de la banda de frecuencias 3 en la tabla
66.             var LMP = 27499.49;;    //Limite maximo permisible en la banda3
67.             document.getElementById('b3').innerHTML = nodo.banda3[1]+'-'+nodo.banda3[3]+'MHz';
68.             document.getElementById('fc3').innerHTML = nodo.banda3[2]+'MHz';
69.             document.getElementById('op3').innerHTML = nodo.banda3[0];
70.             document.getElementById('E3').innerHTML = nodo.banda3[4]+' mv/m';
71.             document.getElementById('pct3').innerHTML = (nodo.banda3[4]/LMP*100)+"%";
72.             //mostrar la información de la banda de frecuencias 4 en la tabla
```

```
73.             var LMP = 28832.65; //Limite maximo permisible en la ban
    da4
74.             document.getElementById('b4').innerHTML = nodo.banda4[
    1]+'-'+nodo.banda4[3]+'MHz';
75.             document.getElementById('fc4').innerHTML = nodo.banda4
    [2]+'MHz';
76.             document.getElementById('op4').innerHTML = nodo.banda
    4[0];
77.             document.getElementById('E4').innerHTML = nodo.banda4
    [4]+' mv/m';
78.             document.getElementById('pct4').innerHTML = (nodo.band
    a4[4]/LMP*100)+"%";
79.             //mostrar la información de la banda de frecuencias 5 en la
    tabla
80.             var LMP = 29005.23; //Limite maximo permisible en la ban
    da5
81.             document.getElementById('b5').innerHTML = nodo.banda5[
    1]+'-'+nodo.banda5[3]+'MHz';
82.             document.getElementById('fc5').innerHTML = nodo.banda5
    [2]+'MHz';
83.             document.getElementById('op5').innerHTML = nodo.banda
    5[0];
84.             document.getElementById('E5').innerHTML = nodo.banda5
    [4]+' mv/m';
85.             document.getElementById('pct5').innerHTML = (nodo.band
    a5[4]/LMP*100)+"%";
86.             //mostrar la información de la banda de frecuencias 6 en la
    tabla
87.             var LMP = 30083.13; //Limite maximo permisible en la ban
    da6
88.             document.getElementById('b6').innerHTML = nodo.banda6[
```

```
    1]+'-'+nodo.banda6[3]+'MHz';
89.             document.getElementById('fc6').innerHTML = nodo.banda6
    [2]+'MHz';
90.             document.getElementById('op6').innerHTML = nodo.banda
    6[0];
91.             document.getElementById('E6').innerHTML = nodo.banda6
    [4]+' mv/m';
92.              document.getElementById('pct6').innerHTML = (nodo.band
    a6[4]/LMP*100)+"%";  } );
93.       }
94.       //terminar la conexion
95.       ws.close();
96.       //fin
97.     };
98.}
```

El archivo intefaz.js de la interfaz de visualización define las funciones necesarias para inicializar el mapa usando el API de Google e iniciar la conexión hacia el servidor websocket del nodo sumidero.

El archivo inicia definiendo la variable ***ws***. Esta variable será usada al establecer la conexión con el servidor usando websockets.

La función ***iniciar_mapa*** es la función principal de esta parte de la aplicación, y es la misma función que usa el archivo HTML para configurar los parámetros del mapa usando el API de Google. Dentro de esta función se definen coordenadas para usar como centro del mapa, y el API de Google permite crear el objeto mapa, que permite que la aplicación muestre la región de interés en la interfaz web.

FIGURA Nº4. 75
FUNCIÓN PRINCIPAL DEL ARCHIVO INTERFAZ.JS

```
1.  var ws;            //variable de la conexión websockets
2.
3.  function iniciar_mapa()
4.  {
5.      //define las coordenadas del centro del mapa
6.      var centro = {lat: -12.0717667, lng: -77.1614638888889};
7.      //inicia el mapa con el API de google
8.      var mapa = new google.maps.Map( document.getElementById('m
    apa'),
9.          {zoom: 16, center: centro} );
10.     mapa.setMapTypeId('satellite'); //establece el tipo de mapa
11.
```

Para identificar donde se encuentran los nodos de la red de sensores en el mapa, se usarán marcadores, estos son elementos del API de Google y se crean definiendo sus coordenadas, en latitud y longitud, y usando la variable mapa que se creó previamente en la programación.

Primero se crea el marcador para el nodo sumidero, usando la información de sus coordenadas geográficas y haciendo referencia al objeto mapa donde se ubicará el marcador. Luego se crea una ventana de información en la variable ***info*** que contiene una pequeña descripción de este marcador. Para mostrar esta información se creará una función que se ejecuta cada vez que el usuario hace click sobre el marcador del nodo sumidero. Usando el método ***addListener*** podemos crear este comportamiento, usando el

argumento click y creando una función que muestra el contenido de la variable ***info*** usando se método ***open***.

FIGURA Nº4. 76
CREACIÓN DE MARCADOR PARA EL NODO SUMIDERO

```
12.  //define ubicación del nodo sumidero
13.  var sumidero = {lat: -12.0727667, lng: -77.1634638888889};
14.  //crea marcador para el nodo sumidero
15.  var marcador_sumidero = new google.maps.Marker({position:sumi
     dero, map:mapa});
16.  //selecciona ícono para el marcador
17.  marcador_sumidero.setIcon('http://maps.google.com/mapfiles/ms/i
     cons/green-dot.png');
18.  //crea ventana de información para el nodo sumidero
19.  var info = new google.maps.InfoWindow({ content: "Nodo Sumider
     o" });
20.  //adjunta la ventana de info al nodo sumidero
21.  marcador_sumidero.addListener('click', function()
22.  { info.open(mapa, marcador_sumidero); } );
```

La función ***iniciar_mapa*** también iniciará la conexión hacia el servidor usando websockets, para esto se usa la función ***Websocket*** usando como argumento la dirección del servidor y el puerto donde está implementado el servicio. Esta conexión será guardada en la variable ***ws*** definida en parte superior del archivo.

Después se configura el método ***onmessage*** de la variable ***ws*** donde se programa el comportamiento cuando el cliente reciba información del servidor de websockets.

Debido a que el servidor, en este caso, será el Raspberry Pi del nodo sumidero y este dispositivo envía la información de los nodos fuente de manera automática a cualquier cliente que se conecte. En el método ***onmessage*** se recibirá la data del servidor y se decodificará usando JSON para almacenar el contenido en una variable llamada ***nodos***. Ahora la variable ***nodos*** contiene toda información de los nodos de la red de sensores en una estructura ordenada.

FIGURA Nº4. 77
INICIO DE CONEXIÓN AL SERVIDOR DE WEBSOCKETS

```
24.  //inicio de la conexión al servidor websocket
25.  ws = new WebSocket("ws://192.168.0.10:8888/");
26.  //establece la función de recepción de datos por websockets
27.  ws.onmessage = function(e)
28.  {   var nodos = JSON.parse(e.data); //decodificar la data en format
     o JSON
```

Para acceder a cada elemento de la variable nodos, se puede usar un bucle ***for***, lo que nos permite obtener la información de cada uno de sus elementos y poner dicha información en la tabla para ser visualizada por el usuario.

Para cada nodo se crea un marcador, usando el API de Google, y de manera similar al caso anterior, se inicia el marcador con las coordenadas GPS del nodo fuente, y se crea una ventana de información que contiene el lugar de referencia de la medición.

FIGURA Nº4. 78
RECEPCIÓN DE DATOS DEL SERVIDOR WEBSOCKETS

```
28.
    {   var nodos = JSON.parse(e.data); //decodificar la data en formato J
    SON
29.         //iterando sobre la información adquirida en la variable nodos
30.         for (var N in nodos)
31.         {   //seleccionar un nodo en la lista adquirida
32.             var nodo = nodos[N]
33.             //obtiene las coordenadas de GPS del nodo fuente
34.             var posicion = {lat: nodo.gps[0], lng: nodo.gps[1]};
35.             //crear un marcador para el nodo fuente
36.             var marcador = new google.maps.Marker({position:posicion,
    map:mapa});
37.             //selecciona ícono para el marcador
38.             marcador.setIcon('http://maps.google.com/mapfiles/ms/icons/
    blue-dot.png');
39.             //crea ventana de información para el nodo fuente
40.             var info = new google.maps.InfoWindow({content: nodo.ID});
```

Para que el usuario pueda ver el resto de datos del nodo fuente en la tabla de la interfaz web, se creará una función para el marcador usando el método ***addListener***, de manera que si el usuario hace click sobre este

elemento, se mostrará la ventana de información creada previamente y la tabla de datos de la interfaz web se llenará con los datos de mediciones de radiación no ionizante y la comparación de dichas mediciones con el valor de los límites máximos permisibles. Este proceso se repite para cada banda analizada por el nodo.

FIGURA Nº4. 79
PROCESAMIENTO DE LA INFORMACIÓN RECIBIDA DEL SERVIDOR

```
41.         //Función
    cuando el usuario haga click en el marcador del nodo:
42.         marcador.addListener('click', function()
43.         {   //mostrar la ventana de información
44.             info.open(mapa, marcador);
45.             //mostrar la tabla oculta
46.             document.getElementById('tabla').style.display="block";
47.             //mostrar la información del nodo en la tabla
48.             document.getElementById('ref').innerHTML = nodo.ID;
49.             //mostrar las coordenadas del nodo en la tabla
50.             document.getElementById('gps').innerHTML = nodo.gps;
51.             //mostrar la información de la banda de frecuencias 1 en la
    tabla
52.             var LMP = 26975.98; //Limite maximo permisible en la ban
    da1
53.             document.getElementById('b1').innerHTML = nodo.banda1[
    1]+'-'+nodo.banda1[3]+'MHz';
54.             document.getElementById('fc1').innerHTML = nodo.banda1
    [2]+'MHz';
55.             document.getElementById('op1').innerHTML = nodo.banda
    1[0];
```

```
56.             document.getElementById('E1').innerHTML = nodo.banda1
    [4]+' mv/m';
57.             document.getElementById('pct1').innerHTML = (nodo.band
    a1[4]/LMP*100)+"%";
58.             //mostrar la información de la banda de frecuencias 2 en la
    tabla
59.             var LMP = 27238.99; //Limite maximo permisible en la ban
    da2
60.             document.getElementById('b2').innerHTML = nodo.banda2[
    1]+'-'+nodo.banda2[3]+'MHz';
61.             document.getElementById('fc2').innerHTML = nodo.banda2
    [2]+'MHz';
62.             document.getElementById('op2').innerHTML = nodo.banda
    2[0];
63.             document.getElementById('E2').innerHTML = nodo.banda2
    [4]+' mv/m';
64.             document.getElementById('pct2').innerHTML = (nodo.band
    a2[4]/LMP*100)+"%";
65.             //mostrar la información de la banda de frecuencias 3 en la
    tabla
66.             var LMP = 27499.49;;    //Limite maximo permisible en la b
    anda3
67.             document.getElementById('b3').innerHTML = nodo.banda3[
    1]+'-'+nodo.banda3[3]+'MHz';
68.             document.getElementById('fc3').innerHTML = nodo.banda3
    [2]+'MHz';
69.             document.getElementById('op3').innerHTML = nodo.banda
    3[0];
70.             document.getElementById('E3').innerHTML = nodo.banda3
    [4]+' mv/m';
71.             document.getElementById('pct3').innerHTML = (nodo.band
```

```
a3[4]/LMP*100)+"%";
72.             //mostrar la información de la banda de frecuencias 4 en la tabla
73.             var LMP = 28832.65; //Limite maximo permisible en la banda4
74.             document.getElementById('b4').innerHTML = nodo.banda4[1]+'-'+nodo.banda4[3]+'MHz';
75.             document.getElementById('fc4').innerHTML = nodo.banda4[2]+'MHz';
76.             document.getElementById('op4').innerHTML = nodo.banda4[0];
77.             document.getElementById('E4').innerHTML = nodo.banda4[4]+' mv/m';
78.             document.getElementById('pct4').innerHTML = (nodo.banda4[4]/LMP*100)+"%";
79.             //mostrar la información de la banda de frecuencias 5 en la tabla
80.             var LMP = 29005.23; //Limite maximo permisible en la banda5
81.             document.getElementById('b5').innerHTML = nodo.banda5[1]+'-'+nodo.banda5[3]+'MHz';
82.             document.getElementById('fc5').innerHTML = nodo.banda5[2]+'MHz';
83.             document.getElementById('op5').innerHTML = nodo.banda5[0];
84.             document.getElementById('E5').innerHTML = nodo.banda5[4]+' mv/m';
85.             document.getElementById('pct5').innerHTML = (nodo.banda5[4]/LMP*100)+"%";
86.             //mostrar la información de la banda de frecuencias 6 en la tabla
```

```
87.             var LMP = 30083.13; //Limite maximo permisible en la ban
    da6
88.             document.getElementById('b6').innerHTML = nodo.banda6[
    1]+'-'+nodo.banda6[3]+'MHz';
89.             document.getElementById('fc6').innerHTML = nodo.banda6
    [2]+'MHz';
90.             document.getElementById('op6').innerHTML = nodo.banda
    6[0];
91.             document.getElementById('E6').innerHTML = nodo.banda6
    [4]+' mv/m';
92.              document.getElementById('pct6').innerHTML = (nodo.band
    a6[4]/LMP*100)+"%";  } );
```

Una vez obtenido la información relevante para la aplicación, no hace falta mantener activa la conexión al servidor. Por lo tanto se cerrará la conexión usando el método ***close*** de la variable ***ws*** que se usó para iniciar la conexión.

FIGURA Nº4. 80
FIN DE LA CONEXIÓN AL SERVIDOR DE WEBSOCKETS

```
93.         }
94.         //terminar la conexion
95.         ws.close();
96.         //fin
97.       };
98. }
```

Para iniciar la interfaz de visualización, se debe hacer abrir el archivo interfaz.html con un navegador de internet que soporte HTML5, como por ejemplo Chrome, Firefox, Opera, Brave, Edge, etc. Los archivos CSS y JS deben estar en la misma carpeta que el archivo HTML.

FIGURA Nº4. 81
ARCHIVOS DE LA INTERFAZ DE VISUALIZACIÓN

FIGURA Nº4. 82
ESTADO INICIAL DE LA INTERFAZ WEB DE VISUALIZACIÓN DE DATOS

FIGURA Nº4. 83
INTERFAZ WEB DE VISUALIZACIÓN DE DATOS CON INFORMACIÓN DETALLADA

Referencia	Centro de Salud La Punta			
Coordenadas	-12.0684583,-77.15809166666668			
Banda	F.Central	Operador	Intensidad de Campo (E)	Comparación
758-773MHz	765.5MHz	Entel	7.987220447284345 mv/m	0.029608638675163407%
773-788MHz	780.5MHz	Claro	8.626198083067093 mv/m	0.031668568045537277%
788-803MHz	795.5MHz	Movistar	4.15335463258786 mv/m	0.015103387854057874%
869-880MHz	874.5MHz	Movistar	21.405750798722046 mv/m	0.07424135762311838%
880-890MHz	885MHz	Claro	14.37699680511182 mv/m	0.04956691191592626 4%
944-960MHz	952MHz	Bitel	20.766773162939298 mv/m	0.06903129150104825%

2. 2 Población y muestra

Por la naturaleza de la investigación no corresponde determinar la población ni el tamaño de la muestra, ni otros factores estadísticos.

2. 3 Técnicas e instrumentos de recolección de datos

Para la recolección de datos se utilizará una red de sensores, estratégicamente ubicados en los centros educativos y de salud del distrito de La Punta, cabe mencionar que los datos no serán parte de un análisis estadístico.

2. 4 Procedimientos de recolección de datos

Dada la naturaleza de la investigación, el método del control realimentado por asignación de polos no utiliza técnicas e instrumentos de recolección de datos.

2. 5 Procesamiento estadístico y análisis de datos

La naturaleza de la investigación, no genera un plan de análisis estadístico de datos.

CAPÍTULO V

RESULTADOS

1. Introducción

Para verificar el funcionamiento y comportamiento de los nodos en la red de sensores, se realizaron simulaciones de las diferentes etapas de la investigación, que han sido detallados en el capítulo 4.

2. Simulaciones

2.1 Simulación del nodo fuente

Para simular el funcionamiento del nodo fuente se usó el entorno de desarrollo integrado Spyder, el cual permite crear aplicaciones orientadas a las ciencias e ingeniería usando el lenguaje de programación Python.

Se simuló la etapa de adquisición y procesamiento de los niveles de radiación no ionizante usando el módulo RTL-SDR, pues es la más importante en el funcionamiento del nodo sumidero.

Los resultados de la adquisición serán graficados usando las herramientas de matplotlib que vienen incluidas con el entorno Spyder.

FIGURA Nº5. 1
GRÁFICA DE LA BANDA DE 758-773MHZ

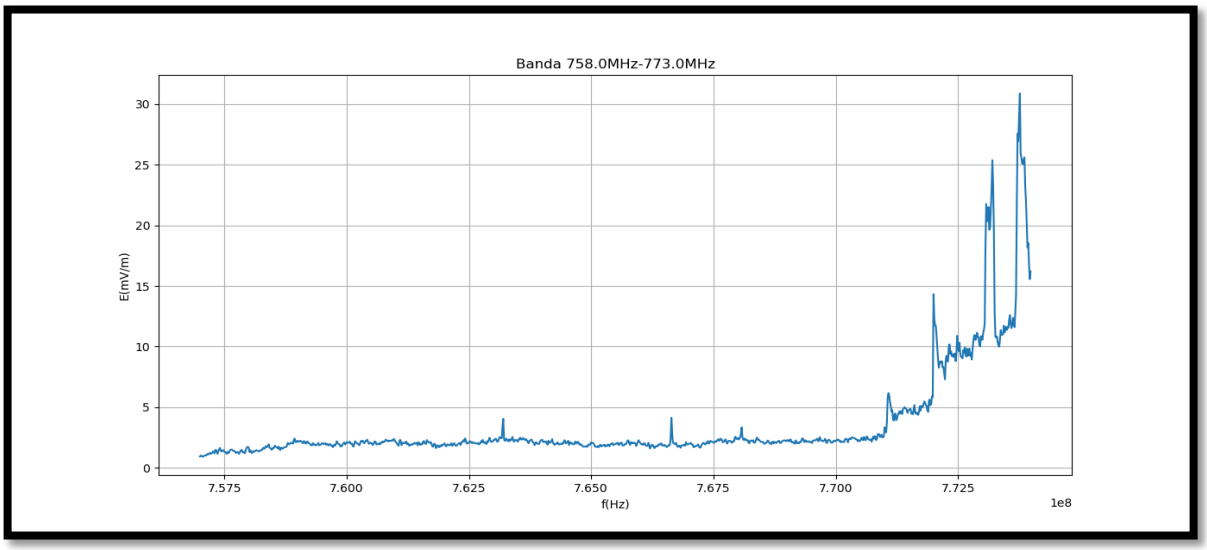


FIGURA Nº5. 2
GRÁFICA DE LA BANDA DE 773-788MHZ

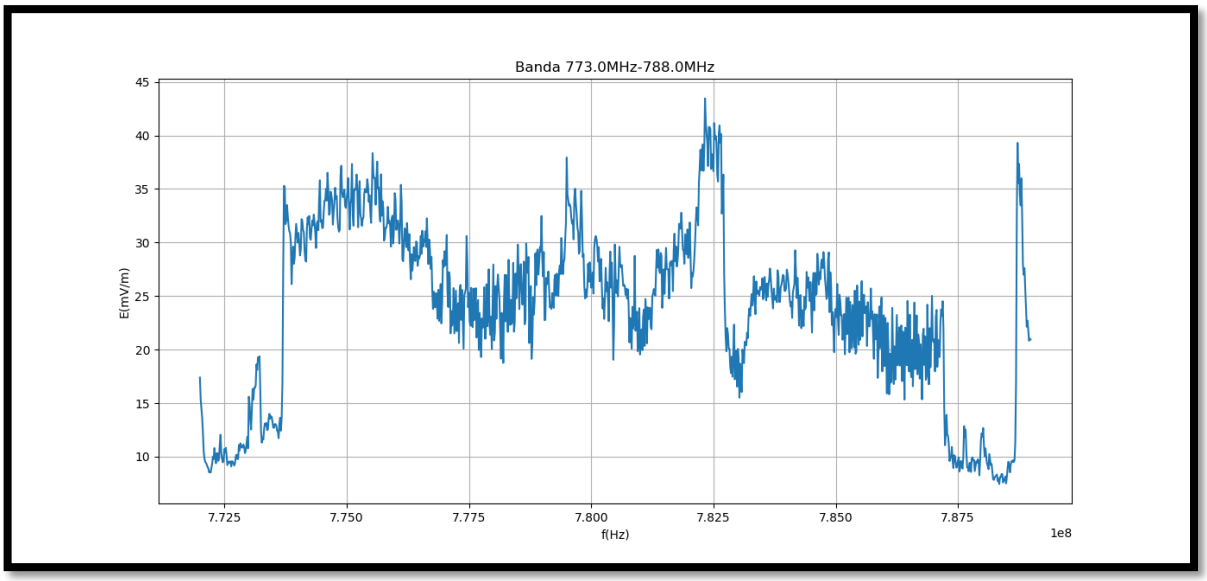


FIGURA Nº5. 3
GRÁFICA DE LA BANDA DE 768-803MHZ

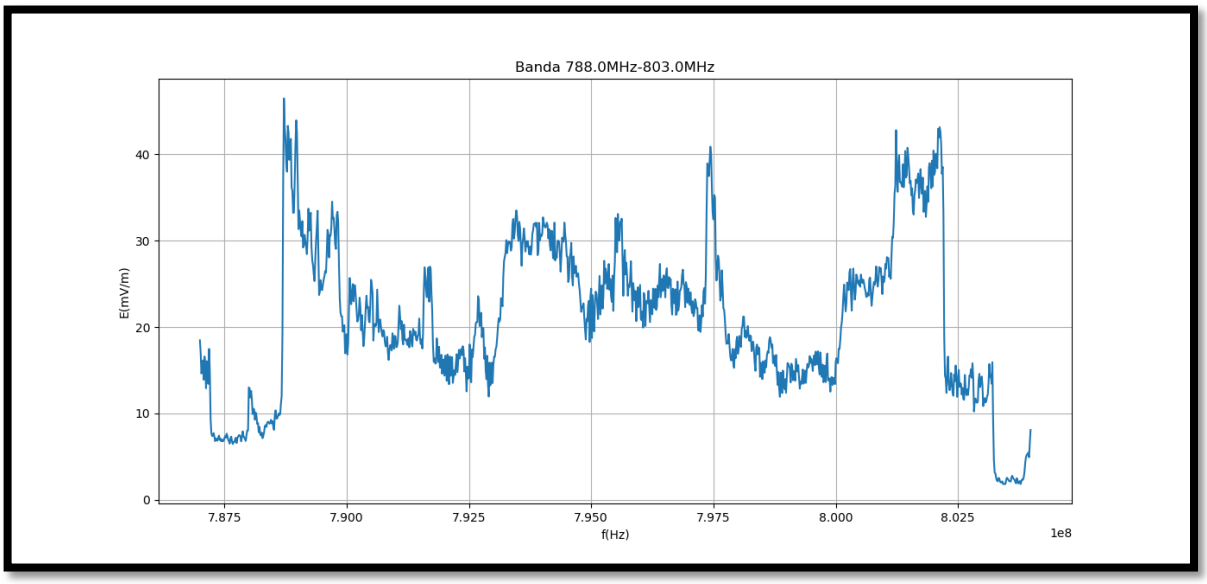


FIGURA Nº5. 4
GRÁFICA DE LA BANDA DE 869-880MHZ

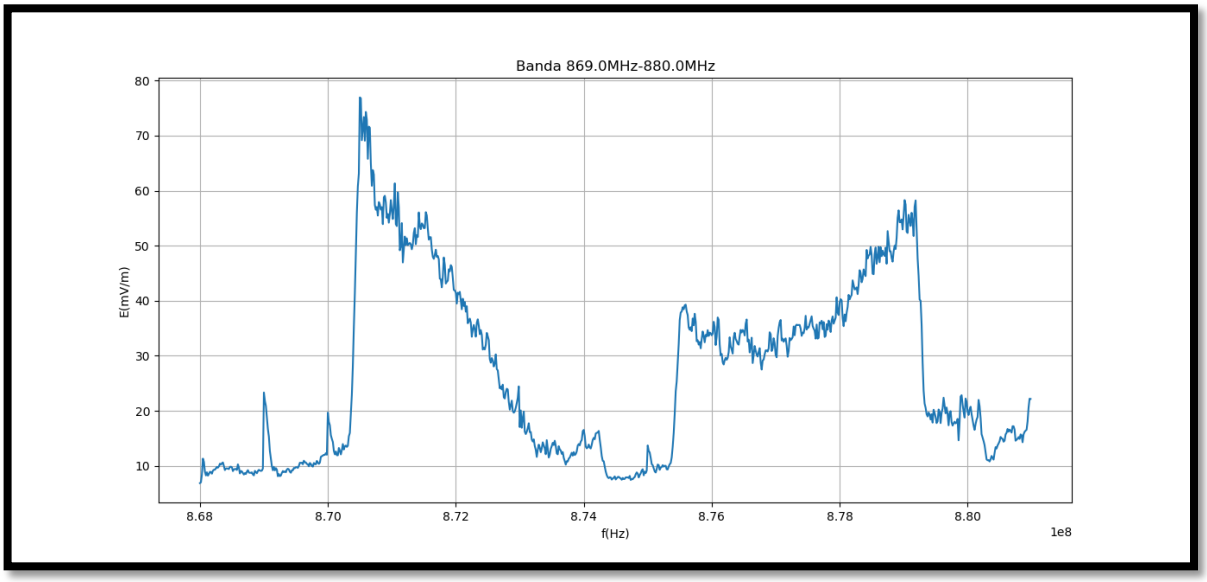


FIGURA Nº5. 5
GRÁFICA DE LA BANDA DE 880-890MHZ

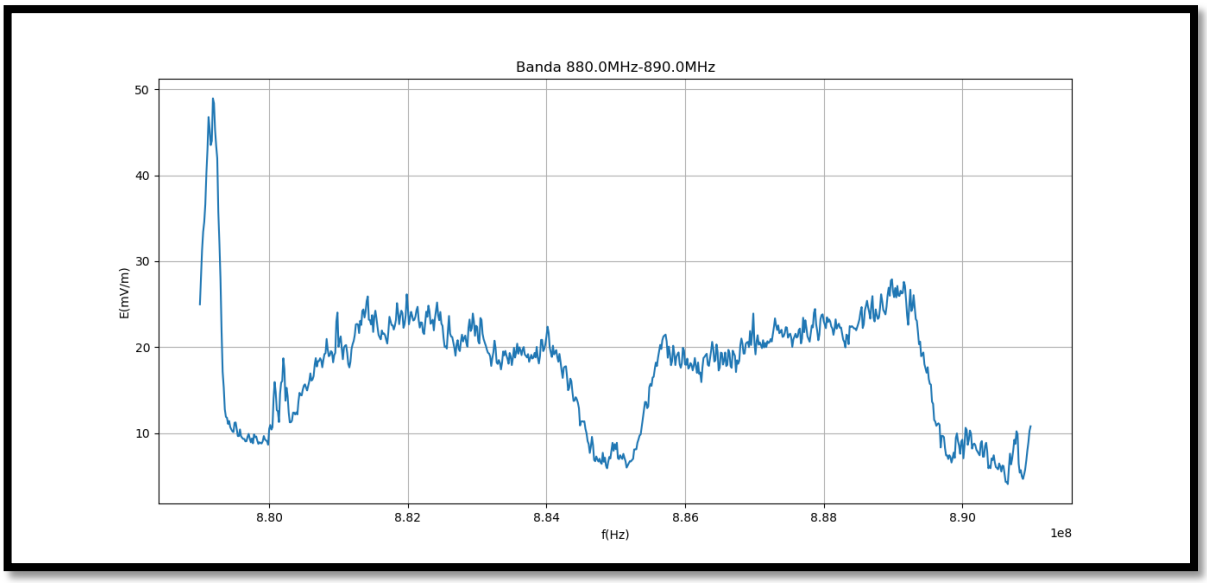


FIGURA Nº5. 6
GRÁFICA DE LA BANDA DE 944-960MHZ

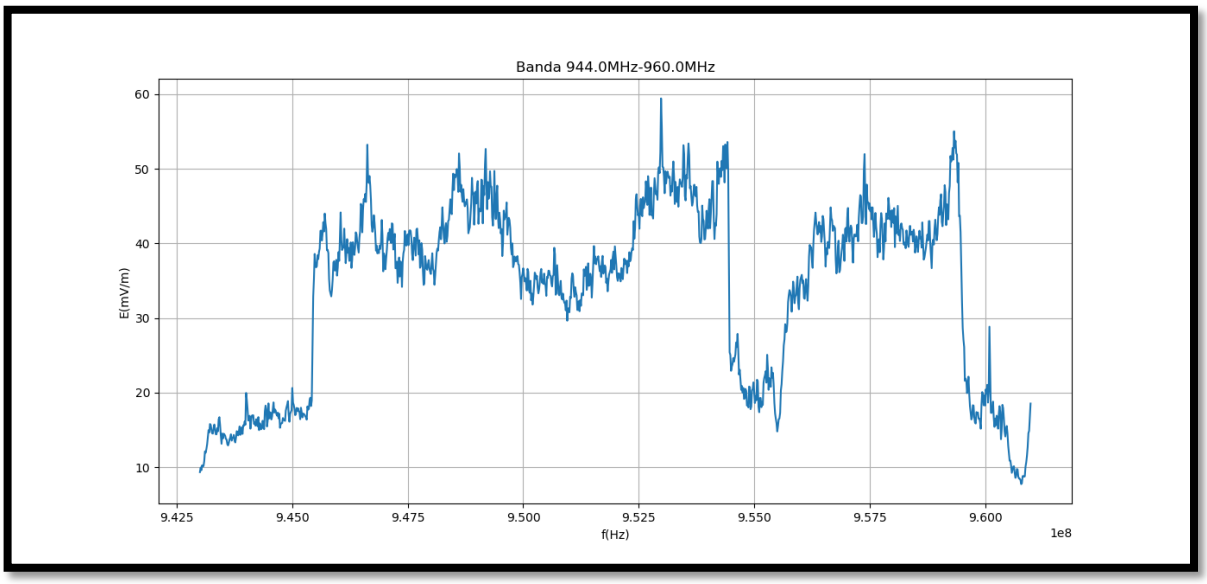
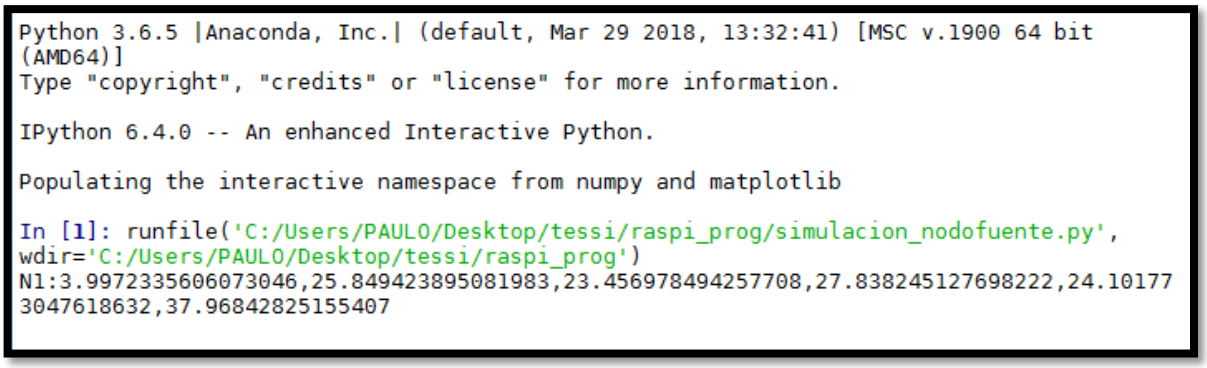


FIGURA Nº5. 7
RESULTADOS DE LA SIMULACIÓN DE MEDICIONES DEL NODO FUENTE



2.2 Simulación del nodo sumidero e interfaz de visualización

Para simular el funcionamiento del nodo sumidero, también se usó el entorno de desarrollo integrado Spyder. Se simuló el funcionamiento del servidor de web sockets que brinda la información adquirida por los nodos fuente. Para visualizar esta información se ejecutó la interfaz de visualización de datos diseñada en el capítulo 4.

Para este caso, el servidor se creó en la dirección local de la computadora usada, y la interfaz se conectó a la misma dirección local. En esta simulación, los datos que normalmente son obtenidos de los nodos fuente fueron reemplazados por valores aleatorios.

FIGURA Nº5. 8
RESULTADOS DE LA SIMULACIÓN DE DATA SERVIDA POR EL NODO SUMIDERO

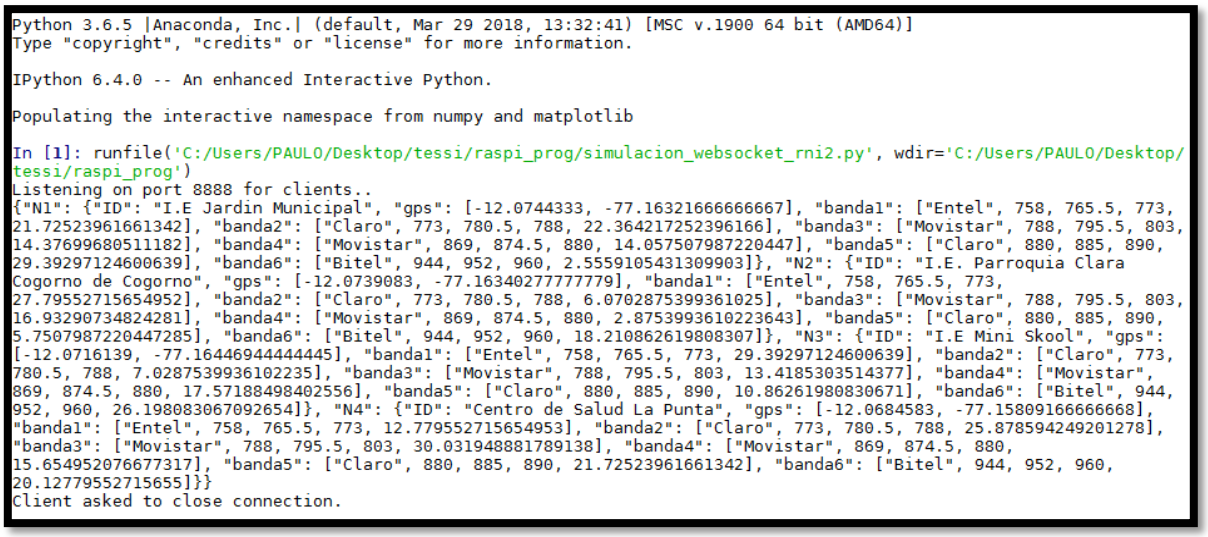


FIGURA Nº5. 9
DATA VISUALIZADA EN LA INTERFAZ OBTENIDA DE LA SIMULACIÓN

Data adquirida por la red de sensores

Referencia	I.E Mini Skool			
Coordenadas	-12.0716139,-77.16446944444445			
Banda	**F.Central**	**Operador**	**Intensidad de Campo (E)**	**Comparación**
758-773MHz	765.5MHz	Entel	16.61341853035144 mv/m	0.061585968444433988%
773-788MHz	780.5MHz	Claro	7.0287539936102235 mv/m	0.025804018407474812%
788-803MHz	795.5MHz	Movistar	8.626198083067093 mv/m	0.031368574773812 51%
869-880MHz	874.5MHz	Movistar	25.559105431309906 mv/m	0.0886463971619324%
880-890MHz	885MHz	Claro	18.210862619808307 mv/m	0.0627847550935066%
944-960MHz	952MHz	Bitel	26.198083067092654 mv/m	0.0870856292782455%

3. Presupuesto

TABLA Nº5. 1
PRESUPESTO DE MATERIALES

Componente	Descripción	Cantidad	Costo Unitario	Costo Total
Módulo RTL-SDR	Sensor de Intensidad de Campo.	1	S/.40	S/.40
Antena y conectores	Para el funcionamiento del sensor.	1	S/.12	S/.12
Módulo NEO-M8M	Módulo de localización.	1	S/.70	S/.70
Raspberry Pi	Procesador principal.	1	S/.200	S/.200
Módulo Xbee Zigbee	Interfaz de enlace inalámbrico.	1	S/.100	S/.100

Regulador TPS61030	Regulador de voltaje.	1	S/.4	S/.4
Resistencias y Capacitores	Componentes pasivos para el circuito regulador.	10	S/.1	S/.10
Microcontrolador AVR	Microcontrolador para control de potencia.	2	S/.8	S/.16
Transistor IRF740	Switch para controlar los componentes del sistema.	1	S/.4	S/.4
Batería	Fuente de energía.	1	S/.200	S/.200
TOTAL				S/.656

TABLA Nº5. 2
PRESUPUESTO TOTAL POR ALUMNO

Categoría	**Salario Mensual**	**% de Jornada**	**Tiempo**	**Costo Total**
Alumno Tesista 1	1500	100	4	6000
Alumno Tesista 2	1500	100	4	6000
Alumno Tesista 3	1500	100	4	6000
TOTAL				18 000

CAPITULO VI

DISCUSIÓN DE RESULTADOS

1. Contrastación de hipótesis con los resultados

1.1 Contrastación de hipótesis con los resultados

En esta investigación, llamada Diseño de red de sensores para monitoreo de Radiación No Ionizante de los servicios de telefonía móvil en colegios y hospitales del distrito de La Punta en la región Callao, se ha comprobado que los componentes que forman esta red funcionan de manera correcta.

Se corroboró que el uso de un sensor sintonizador de RF, como el módulo RTL-SDR, y un computador capaz de usar el lenguaje de programación Python, como el Raspberry Pi Zero, pueden ser usados para adquirir,

procesar y medir los niveles de radiación no ionizante emitidos por los servicios de telefonía móvil.

Se verificó que la distribución de los nodos en el distrito de La Punta permite un enlace adecuado entre los dispositivos y que la topología de red estrella es suficiente para la transmisión y recepción de datos.

También se comprobó que la data obtenida de estas mediciones puede ser observada por un usuario con acceso al nodo sumidero, usando una interfaz de visualización web.

2. Contrastación de resultados con otros estudios similares

Se verificó los resultados de la tesis Escuela Superior Politécnica del Litoral de la Facultad de Ingeniería en Electricidad y Computación. “Análisis de Mediciones de Radiaciones No Ionizantes en ambientes interiores y exteriores en predios de la ESPOL” de Guayaquil- Ecuador, Wilson Alejandro Díaz García y Felipe Walkir Proaño Salvatierra, con el prototipo de la red de sensores, tomando como punto de referencia su desarrollo que es afín a nuestra investigación, comprobando que el prototipo funciona adecuadamente.

CAPÍTULO VII

CONCLUSIONES

- Se concluyó que se puede usar componentes de bajo costo como el sensor RTL-SDR, un computador de tarjeta reducida, un transceptor de radio Zigbee y un banco de baterías para formar los nodos de la red de sensores.
- Debido a las limitaciones del módulo RTL-SDR, sólo fue posible realizar mediciones de 6 bandas de frecuencias usadas en los servicios de telefonía móvil.
- La interfaz de visualización de las mediciones realizadas por los nodos en la red es compatible con cualquier dispositivo que soporte el lenguaje HTML5 y tenga acceso a la red externa del nodo sumidero.

- Los nodos fuente de la red de sensores pueden permanecer sin supervisión externa durante aproximadamente 596 días debido a su operación intermitente en la medición de RNI, que dura 6 minutos, y se realiza cada 24 horas.
- El servidor de websockets en el nodo sumidero sirve las lecturas más recientes obtenidas de los nodos fuente y no tiene la capacidad de almacenar datos históricos.

CAPÍTULO VIII

RECOMENDACIONES

- Para obtener mediciones más precisas, se recomienda usar una antena isotrópica con el módulo RTL-SDR, esta se debe acoplar con un conector MCX macho.
- Para reducir el consumo de potencia del Raspberry Pi durante la secuencia de encendido y operación, se recomienda deshabilitar el puerto HDMI de la tarjeta modificando los archivos de configuración en la tarjeta microSD.

- Para evitar interferencia y colisiones de datos entre la transmisión de los nodos fuente en la red de sensores, se recomienda iniciar el ciclo de trabajo de cada nodo en diferentes horarios.
- Para evitar ataques de denegación de servicio (DoS), de desbordamiento de buffer u otros al servidor implementado en el nodo sumidero, se recomienda incluir firewalls, o routers con reglas de control de acceso en la red externa.
- Para observar y estudiar las mediciones de los nodos a lo largo del tiempo de vida de la red, se recomienda incluir un sistema de base datos en la red externa del nodo sumidero, que permita almacenar dicha información de manera periódica.

CAPITULO IX

REFERENCIAS BIBLIOGRÁFICAS

[1]. Kraus J.D. *Electromagnetismo.*

[2]. Terman F.E. *Ingeniería Electrónica y de Radio.*

[3]. Oficina Internacional del Trabajo. *Protection of Workers from Power Frecuency Electric and Magnetic Fields.*

[4]. Andrieu J.M. (agosto 2000-setiembre 2000) *Apunte Radiofísica Sanitaria*. IEEE Spectrum.

[5]. Mag. Ing. Víctor Cruz Ornetta (2005). *La telefonía Móvil y su Salud*. Inictel.

[6]. Bengt Knave. *Radiaciones no Ionizantes*.

[7]. Wavecontrol, *Campos Electromagnéticos y Telefonía Móvil, Mapas de Radiaciones Municipales*.

[8]. Mobile Manufactures Fórum, *La Salud y los Campos Electromagnéticos de teléfonos celulares*, Boletín

[9]. Antonio Pérez Yuste (2002). *La percepción social de los campos electromagnéticos*, Revista Mundo Electrónico.

[10]. Instituto de Investigaciones científicas y técnicas de las fuerzas armadas, *Radiación No Ionizante de sistemas de telefonía celular móvil: la percepción de la población, la disparidad de los estándares y el monitoreo a gran escala*, www.citefa.gov.ar/soluciones_tecno/Antenas.

[11]. Presidencia del Consejo de Ministros. PCM*, Aprueban Estándares de Calidad Ambiental (ECAs) para Radiaciones No Ionizantes Decreto Supremo N°10-2005-PCM*.

[12]. Ministerio de Transporte y Comunicaciones, *establecen los Límites Máximos Permisibles de Radiaciones No Ionizantes en Telecomunicaciones,* Decreto Supremo N°038-2003-MTC.

[13]. Ministerio de Transportes y Comunicaciones, *Norma Técnica de Lineamientos para el Desarrollo de los Estudios Teóricos de Radiaciones no Ionizantes*, Resolución Ministerial N°612-2004-MTC-03.

[14]. Ministerio de Transportes y Comunicaciones, *Aprueban norma técnica sobre Restricciones Radioeléctricas en Áreas de Uso Público*, Resolución Ministerial N °120-2005-MTC/03.

[15]. Sohraby, M. (2007), *Wireless Sensor Networks Technology, Protocols and Applications, John Wiley & Sons*, Inc Publications.

[16]. SCHILLER, J. *Mobile Communications*. 2ª ed. Gran Bretaña: Pearson Education Limited, 2003. ISBN 0321123816.

[17]. ELAHI, A., GSCHWENDER, A. *ZigBee Wireless Sensor and Control Network*. 1ª ed. Estados Unidos de América: Prentice Hall, 2009. ISBN-10: 9780137134854.

[18]. MONK, S. *Raspberry Pi Cookbook*. 1ª ed. California: O'Reilly Media, 2014. ISBN-10: 9781449365226

[19]. Rhode & Schwarz. (2012), Generation of IEEE 802.15.4 Signals. (Recuperado el 19 de Agosto de 2018) https://www.rohde-schwarz.com/hu/applications/generation-of-ieee-802.15.4-signals-application-note_56280-95360.html

[20]. JADAGERIMATH, A.N. (2014), Battery Capacity Management in Wireless Sensor Network Rechargeable Sensor Nodes. (Recuperado el 19 de Agosto de 2018) https://www.ijecs.in/index.php/ijecs/article/download/1500/1382/

[21]. Digi. (2018), *XBee®/XBee-PRO S2C Zigbee®.*

[22]. RF Module User Guide. (Recuperado el 22 de agosto de 2018) https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf

[23]. FAHMY, H.M.A. *Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis*. 1ª ed. Singapur: Springer Singapore, 2016. ISBN 9789811004124.

[24]. MeshNetics. (2008). ZigBeeNet™ Software 1.0 Application Note: Battery Lifetime Estimation. Recuperado el 22 de agosto de 2018 de

http://www2.ee.ic.ac.uk/t.clarke/projects/Resources/ZDK_v2.0_Complete/Documentation/Application%20Notes/AN-481~12-(Battery%20Lifetime%20Estimation).pdf

[25]. Microchip. (2015). ATtiny25/V / ATtiny45/V / ATtiny85/V. Recuperado el 25 de Agosto de 2018 de http://ww1.microchip.com/downloads/en/devicedoc/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf

[26]. ASIX. (2008). AX88772: USB to 10/100 Fast Ethernet/HomePNA Controller. Recuperado el 3 de setiembre de 2018 de https://www.framboise314.fr/wp-content/uploads/2016/08/AX88772.pdf

[27]. Broadcom. (2012). BCM2835 ARM Peripherals. Recuperado el 28 de agosto de 2018 de https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf

[28]. MeanWell. (2009). DR-15 series. Recuperado el 1 de setiembre de 2018 de https://www.mouser.com/ds/2/260/DR-15-spec-1108947.pdf

[29]. Rafael Micro. (2011). R820T High Performance Low Power Advanced Digital TV Silicon Tuner. Recuperado el 11 de Agosto de 2018 de https://rtl-sdr.com/wp-content/uploads/2013/04/R820T_datasheet-Non_R-20111130_unlocked.pdf

[30]. Semtech. (2007). Calculating Radiated Power and Field Strength for Conducted Power Measurements. Recuperado el 22 de Agosto de 2018 de https://www.semtech.com/uploads/documents/semtech_acs_rad_pwr_field_strength.pdf

[31]. Giangrandi. (2000). Field strength and received power. Recuperado el 29 de Agosto de 2018 de http://www.giangrandi.ch/electronics/anttool/rx-field.shtml

ANEXOS

1. Matriz de Consistencia

"RED DE SENSORES PARA MONITOREO DE RADIACIÓN NO IONIZANTE DE LOS SERVICIOS DE TELEFONÍA MÓVIL EN COLEGIOS Y HOSPITALES DEL DISTRITO DE LA PUNTA EN LA REGIÓN CALLAO"

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	METODOLOGÍA
Problema General: ¿Es posible que el diseño de una red de sensores permita monitorear la radiación no ionizante de los servicios de telefonía móvil en áreas de uso público del distrito de La Punta en la Región Callao? **Problemas Específicos:** P.E.1 ¿Qué componentes son necesarios para realizar las medición, procesamiento y transmisión de los niveles de radiación no ionizante en el distrito de La Punta? P.E.2 ¿Cuál es la manera más óptima de transmitir las mediciones de radiación no ionizante en la red de sensores? P.E.3 ¿Dónde se deben ubicar los sensores en el distrito de La Punta para obtener la medición de los niveles de radiación no ionizante en las áreas de uso público? P.E.4 ¿Cómo se puede visualizar los resultados de las mediciones obtenidas por la red de sensores en el distrito de la Punta?	**Objetivo General:** Diseñar una red de sensores para el monitoreo de radiación no ionizante de los servicios de telefonía móvil en el distrito de La Punta en la región Callao. **Objetivos Específicos:** O.E.1 Diseñar el hardware y software de los nodos que componen la red de sensores. O.E.2 Diseñar la topología usada en la red de sensores. O.E.3 Planificar la distribución de nodos en el distrito de La Punta. O.E.4 Programar la interfaz de visualización de las mediciones de radiación no ionizante.	**Hipótesis General:** El diseño de una red de sensores de radiación no ionizante emitida por los servicios de telecomunicaciones permitirá el monitoreo en tiempo real de los niveles de intensidad de campo en el distrito de la Punta en la región Callao que pueden ser comparados con los límites máximos permisibles establecidos en el Decreto Supremo N°38-2003-MTC y Resolución Ministerial N°120-2005-MTC para garantizar su cumplimiento, sin tener que depender únicamente en estudios teóricos. **Hipótesis Específicos:** H.E.1 El hardware y software que conforman los nodos de la red de sensores será necesario para la adquisición, procesamiento y transmisión de las mediciones de los niveles de radiación no ionizante en el distrito de La Punta. H.E.2 La topología de red del sistema optimizará la transmisión y recepción de datos correspondientes a la medición de los nivele s de radiación no ionizante en el distrito de La Punta. H.E.3 La distribución adecuada de los nodos en las áreas de uso público en el distrito de La Punta permitirá la medición de los niveles de radiación no ionizante. H.E.4 La interfaz web para visualizar las mediciones de la red de sensores facilitará el monitoreo en tiempo real de los niveles de radiación no ionizante en el distrito de La Punta.	**Variables Independientes:** X= Nivel de radiación no ionizante emitida por las estaciones radioeléctricas de servicios de telecomunicaciones. **Variables Dependientes:** Y1= Diseño de nodos sensores en la red. Y2= Topología de la red de sensores. Y3= Distribución geográfica de los nodos de la red. Y4= Interfaz web para visualizar los datos procesados.	La investigación que se realizó para hacer este estudio se determina como un proyecto factible en el cual se demostrará los beneficios de este tipo de tecnología, este estudio se fundamenta en una investigación aplicada.