UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

"DESARROLLO DE UN PROTOCOLO DE COMUNICACIÓN PARA UNA RED INALÁMBRICA DE SENSORES EN ÁREAS REMOTAS BASADO EN SOFTWARE DEFINED RADIO"

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

JEAN PIERRE TINCOPA FLORES

ASESOR: ING. JAIME ALBERTO VALLEJOS LAOS

CALLAO, SETIEMBRE, 2017

PERÚ

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO

"DESARROLLO DE UN PROTOCOLO DE COMUNICACIÓN PARA UNA RED INALÁMBRICA DE SENSORES EN ÁREAS REMOTAS BASADO EN SOFTWARE DEFINED RADIO"

PRESENTADO POR:
JEAN PIERRE TINCOPA FLORES

ASESOR:
ING. JAIME ALBERTO VALLEJOS LAOS

CALIFICACION:

(14) CATORCE

Ing. ARMANDO CRUZ RAMIREZ
Presidente de Jurado

Ing. WILBERT CHAVEZ IRAZABAL
Secretario de Jurado

Ing. JACOB ASTOCONDOR VILLAR
Vocal de Jurado

CALLAO – PERÚ
SETIEMBRE 2017

DEDICATORIA

A mi madre por el apoyo y paciencia
ilimitada e incondicional que me ayudó a
obtener todos logros de mi vida.

AGRADECIMIENTO

Agradezco de manera especial al Ing. Oscar Baltuano por haberme aceptado para el desarrollo de esta tesis de pregrado, su confianza en mi trabajo y su capacidad para guiar mis ideas ha sido un aporte totalmente invaluable, tanto para el desarrollo de esta tesis como para mi formación como investigador.

De igual manera agradezco al Ing. Jaime Vallejos por su efectiva colaboración y asesoramiento durante cada una de mis visitas a su laboratorio y, en general, durante toda la realización de esta tesis, la cual se vio reflejada en muchos de los resultados obtenidos.

Así mismo, quiero expresar mi agradecimiento al Dr. Juan Pimentel por su importante aporte y activa participación en el desarrollo de esta tesis, destacando su paciencia en cada etapa del desarrollo de este proyecto.

Finalmente, debo agradecer al Laboratorio de Desarrollo Electrónico del Área de Investigación y Desarrollo en el Instituto Peruano de Energía Nuclear, por haberme permitido desarrollar esta tesis en sus instalaciones

Este trabajo de investigación ha sido posible gracias al financiamiento otorgado por el Programa Nacional de Innovación para la Competitividad y Productividad (InnóvatePerú) del Ministerio de la Producción a través del Contrato N° 198-FINCyT-IA-2013 para el desarrollo del Proyecto de Investigación Aplicada "Desarrollo de sistemas telemétricos avanzados para sensores medio ambientales remotos" ejecutado por el IPEN en asociación con LeSoft LLC.

ÍNDICE

FIGURAS DE CONTENIDO

TABLAS DE CONTENIDO

RESUMEN

Actualmente las redes inalámbricas de sensores tienen un gran auge en diversas aplicaciones, dado que otorgan una excelente respuesta a las necesidades actuales de tener redes flexibles y de bajo costo, lo cual permite obtener datos de monitoreo desde entornos poco accesibles para que estos sean enviados a una estación base donde pueda procesarse dicha información.

De igual manera, en el área de las telecomunicaciones existen actualmente módulos basados en Software Defined Radio (SDR), esta tecnología pretende reemplazar el hardware que habitualmente se usa para realizar tareas como modulación o demodulación por algoritmos ejecutados por un procesador de propósito general, logrando con esto adaptabilidad a muchos sistemas de comunicación.

La presente tesis plantea el desarrollo de un protocolo de comunicación para una red inalámbrica de sensores mediante el uso de SDR, la cual permitirá conectar múltiples dispositivos inalámbricos para transmitir datos de distintas variables ambientales o atmosféricas, este prototipo será escalable dado que podrá expandir la cantidad de nodos en la red para poder abarcar un área extensa y además de ser compatible con varios tipos de sensores mientras estos puedan comunicarse con un microcontrolador.

ABSTRACT

Nowadays, wireless sensor networks are in vogue because they provide a solution to the current demand of flexible and low-cost networks that allow the obtainment of monitoring data of inaccessible locations and to send that data to a base station for information to be processed.

Currently, in telecommunications, hardware is being replaced by modules based in Software Defined Radio (SDR) for tasks such as algorithm modulation or demodulation executed by a sole purpose processor and thus achieving the adaptability of multiple communication systems.

This thesis presents the development of a communication protocol for a wireless sensor network through SDR which will connect multiple wireless devices to transmit data from several environmental or atmospheric variables. This prototype will be scalable since the quantity of nodes present in the network will be expanded in order to cover wider areas. The prototype will also be compatible to several types of sensors that communicate with a microcontroller.

CAPITULO I:

PLANTEAMIENTO DEL PROBLEMA

1.1. Determinación del problema

Existe la creciente necesidad de monitorear parámetros ambientales y atmosféricos en áreas remotas con condiciones agrestes para lo cual es necesario el uso de sensores, habitualmente se implementan sistemas de medición mediante cableado el cual resulta demasiado costoso y poco escalable, es por ello que la comunicación inalámbrica es una alternativa a esta problemática además de ello se conoce que en algunos entornos rurales no existen redes de telecomunicaciones como internet, ni servicio de telefonía celular GSM, siendo estas las más usadas típicamente en redes inalámbricas, por lo tanto resulta imperativo el establecer una comunicación a larga distancia para redes de sensores que funcione bajo estas condiciones.

1.2. Formulación del problema

¿Es posible establecer una comunicación mayor a 1 kilómetro en redes de sensores inalámbricas en áreas remotas?

1.3. Objetivos de la Investigación

1.3.1. Objetivo General

- Crear un protocolo de comunicación para una red inalámbrica de sensores mediante el uso de SDR.

1.3.2. Objetivos Específicos

- Integrar los módulos de SDR con microcontroladores para crear los nodos que conforman la red.

- Diseñar algoritmos de programación para la transmisión y recepción de datos para todos los nodos que conforman la red.

- Diseñar algoritmos de programación que logren establecer la lectura de datos de sensores usando microcontroladores.

1.4. Justificación

El presente trabajo está justificado dado que aporta al desarrollo de las redes de comunicación inalámbricas, dando paso a la creación a redes de telemetría en áreas remotas, además dada la exigencia de proponer nuevos métodos que contribuyan a la reducción de costos y la escalabilidad de las redes esta resulta ser una alternativa adecuada. La obtención de datos de este sistema tiene una amplia área de aplicaciones complejas como prevención de desastres naturales, monitoreo del ambiente, etc.

1.5. Importancia

La importancia del desarrollo de este trabajo está reflejado en las múltiples aplicaciones del mismo ya que su contribución al monitoreo de parámetros ambientales tiene influencia en sectores productivos del país tales como agricultura y ganadería así como la prevención de desastres con análisis de microclima.

CAPITULO II:

MARCO TEÓRICO

2.1. Antecedentes del estudio

En primer lugar se tiene como antecedente el trabajo de Suárez Barón, Juan Carlos **"Diseño Y Construcción De Un Sistema De Monitoreo Para Invernaderos Apoyado Con Tecnología Zigbee"**, presentado en la Escuela De Ciencias Básicas Tecnología E Ingeniería de la Universidad Nacional Abierta Y A Distancia (UNAD) en Colombia como parte de los requerimientos para obtener el título profesional de Ingeniero Electrónico en el año 2013.

La investigación trata sobre el diseño y desarrollo de un prototipo de una red inalámbrica de sensores (WSN) basada en el estándar Zigbee. En el presenta la comunicación entre los sensores y módulos de comunicación Zigbee con microcontroladores en cada dispositivo, además se describe el diseño de una interfaz gráfica de usuario (GUI) desarrollada por medio de instrumentos virtuales que visualizan la lectura de las variables. El análisis de los resultados permitió determinar los aspectos positivos y negativos del hardware utilizado y los programas de visualización.

De esta misma manera también se tiene como antecedente el trabajo de Villón Valdiviezo, Daniel **"Diseño De Una Red De Sensores Inalámbrica Para Agricultura De Precisión"**, presentado en la Facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú (PUCP) en Perú, como parte de los requerimientos para obtener el título profesional de Ingeniero Electrónico en el año 2009 en la cual se desarrolla el diseño y la implementación de una red de sensores inalámbrica, tomando en cuenta la topología de la red y el protocolo a diseñar, en el podemos encontrar que el proyecto fue desarrollado dentro de un sistema embebido, además de implementar aplicaciones basada en un lenguaje de programación. Muchos otros tópicos son de mucha ayuda para el desarrollo de este trabajo.

2.2. Marco conceptual

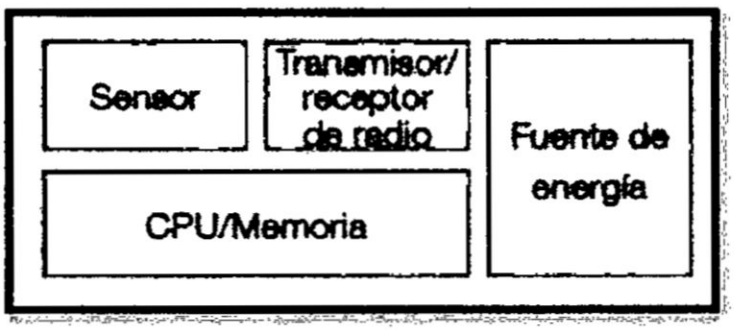
Wireless Sensor Network

Una WSN (Wireless Sensor Network) o red inalámbrica de sensores, es un tipo de red con una gran cantidad de dispositivos distribuidos en un área, que hacen uso de sensores para el monitoreo de diversos parámetros en distintos puntos del área. Dentro de ella existen dispositivos, a los que se le denomina nodos, los cuales son unidades que constan de un microcontrolador o un sistema embebido, una fuente de energía, un transmisor y un sensor.

Las WSN conforman un sistema de comunicación que permite reemplazar las redes cableadas. En este tipo de redes los datos se propagan a través de un medio inalámbrico mediante ondas electromagnéticas a través de antenas.

Por lo general cada uno de elementos de la red está conformado por dispositivos autónomos que constan de un microcontrolador (CPU), un sensor, un transceiver (Transmisor/Receptor) y una fuente de energía.

FIGURA 2.1. PARTES QUE CONFORMAN UN DISPOSITIVO DENTRO UNA WSN



Fuente: [5]

Elementos de una WSN

Por lo general las WSN están conformadas por 3 elementos principales detallados a continuación:

- **Base Station (Estación Base):**
  Recolecta toda la información de los nodos de la red de sensores y al mismo tiempo hace trabajo de interfaz con la PC para administrar los datos recibidos.

- **Cluster Head (Cabeza del grupo):**
  Son elementos que se extienden por toda el área que conforma la red, hacen trabajo de Router ya que reciben información de los nodos sensores y lo retransmiten hacia otro CH o hacia la Base Station.

- **Sensor Node (Nodo Sensor):**
  Son los elementos finales de la red, quienes se encargan de adquirir datos de los sensores para luego transmitírselos a los CH.

Topología de una Red

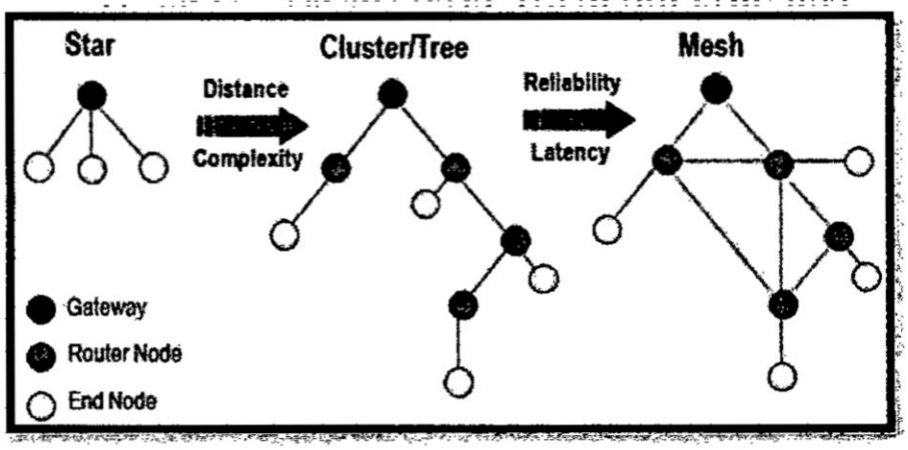
Es la forma en la que esta interconectada la red para realizar la transferencia de información, por lo general existen varios tipos de topologías dependiendo la aplicación para la cual va ser utilizada ya que algunas son más favorables para alcanzar mayores distancias y otras para tener una mayor velocidad de transmisión de datos.

- **Estrella:**
  En esta red todos los elementos de la red se conectan directamente a la estación base.

- **Cluster/Árbol:**
  Esta topología está basada en jerarquías en forma de árbol ya que la estación base es el enlace troncal de la red y los demás elementos son sus ramificaciones, se puede entender esta red también como un conjunto de topologías en estrella.

- **Malla:**
  En este caso todos los nodos que conforman la red están conectados entre sí, haciendo que sea posible llevar información por distintos caminos.

FIGURA 2.2. EJEMPLOS DE TOPOLOGÍAS DE WSN



Fuente: [6]

Enrutamiento de una Red

Se refiere al camino por el cual van a ser transmitidos los datos, se busca que el enrutamiento sea lo más eficaz posible para ahorrar tiempos de transmisión y optimizar la energía administrada a los dispositivos

1. **Modelo de un salto:**
   En este modelo se establece una comunicación directa, todos los elementos de la red se comunicación directamente con la estación base, en este modelo el consumo energético es alto y la distancia está limitada por la capacidad de transmisión de los nodos.

2. **Modelo Multi-Salto:**
   En este modelo los nodos finales transmiten hacia los nodos intermedios para que estos retransmitan la información al nodo más cercano a la estación base, de esta manera el dato va saltando hasta llegar al punto central de la red.

Sensor

Es un dispositivo con capacidad de transformar una magnitud física en otra, generalmente en una señal eléctrica para que pueda ser interpretada por otro dispositivo.

Para el desarrollo de esta tesis se ha hecho uso de múltiples sensores ambientales tales como sensor de temperatura, humedad relativa, presión atmosférica y geo localización.

Protocolo de Comunicación

Un protocolo de comunicación es un conjunto de normas que permiten que dos o más dispositivos de un sistema de comunicación se puedan comunicar de forma eficiente para transmitir información por algún medio físico o inalámbrico.
Dentro del protocolo está definida la sintaxis y sincronización de la comunicación, así como los métodos de corrección de errores en la transmisión y recepción de datos.

Codificación y Decodificación

Son tipos de técnicas de comunicación en las que el emisor convierte un mensaje en signos que puedan ser interpretados por el receptor y viceversa.

Modulación

Son los tipos de técnicas utilizadas para transferir información desde un origen a un destino. Un modulador traduce un mensaje de señal a una señal de portador que opera dentro de la banda de frecuencia de los medios de comunicación. Un demodulador realiza el proceso inverso.

Tipos de Modulación:

Modulación Analógica

- Modulación de Amplitud (AM):

Esta modulación consiste en aumentar o disminuir la amplitud de la onda potadora en base a la señal moduladora.

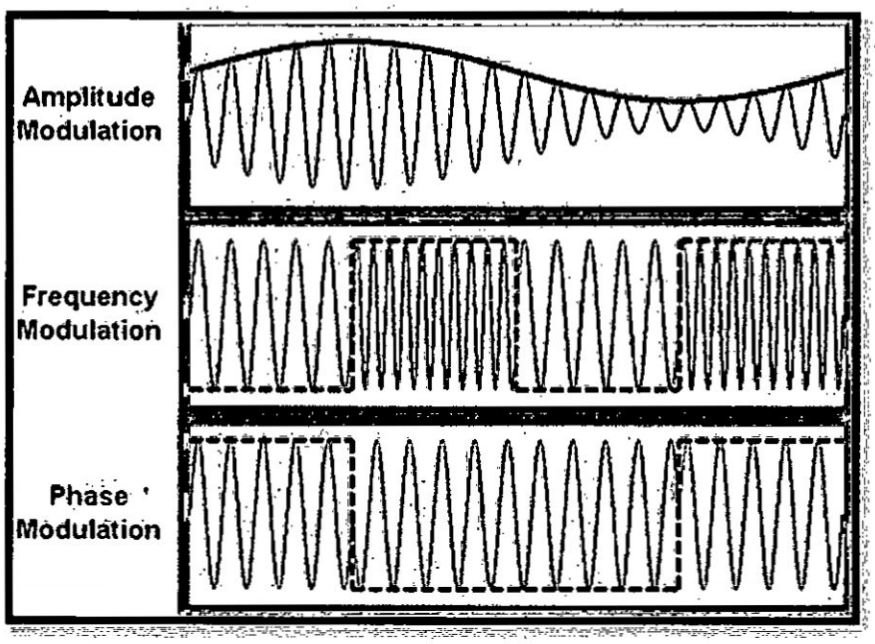
- Modulación de Frecuencia (FM):

Es la modulación de una señal mediante la variación de su frecuencia de acuerdo a la señal a transmitir.

- Modulación de Fase (PM):

En este caso la señal que variara de acuerdo a la señal a transmitir será la fase.

FIGURA 2.3. TIPOS DE MODULACIÓN ANALÓGICA



Fuente: [7]

Modulación Digital

- Modulación por Desplazamiento de Amplitud (ASK):

Esta modulación representa de forma digital la amplitud de la onda portadora en función a la señal a transmitir.

- Modulación por Desplazamiento de Frecuencia (FSK):

En esta modulación se usa 2 o más frecuencias para representar los bits, la señal a transmitir se forma como un tren de pulsos en base a esas frecuencias.

- Modulación por Desplazamiento de Fase (PSK):

Esta modulación se caracteriza porque la señal portadora está representada por la fase de la señal moduladora.

C++ de esta manera se puede implementar sistemas de radio de alto rendimiento.

SISTEMA EMBEBIDO

Un sistema embebido es un sistema de computación diseñado para realizar una o muchas funciones dedicadas, habitualmente es un sistema en tiempo real.

2.3. Definición de Términos

C++

Es un lenguaje de programación sucesor al exitoso lenguaje C que permite la manipulación de objetos. Por lo cual el C++ es llamado un lenguaje híbrido.

GNU

GNU es un sistema operativo basado en el núcleo Unix el cual está formado en su totalidad por software libre.

GPL

Es una Licencia Pública General, la más ampliamente usada en el mundo del software y garantiza a los usuarios tengan la libertad de usar, estudiar, compartir y modificar el software.

GSM

Es un estándar de comunicaciones orientado a telefonía de segunda generación, porque a diferencia de la primera generación de teléfonos portátiles las comunicaciones se producen de forma completamente digital.

GUI

Grafic Interface User o interfaz gráfica de usuario en español, es un tipo de programa informático que actúa como interfaz para usuario, el cual utiliza un conjunto de imágenes y gráficos para representar información y acciones del programa.

IEEE

IEEE, es una asociación mundial de técnicos e ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas.

MICROCONTROLADOR (uC)

Es un circuito integrado que en su interior contiene una unidad central de procesamiento, unidades de memoria volátil y estática además de contar con puertos de entrada y salida.

PYTHON

Es un lenguaje de programación denominado multiparadigma porque soporta programación orientada a objetos, imperativa y funcional. Además de usar tipado dinámico y es multiplataforma, es decir, funciona en múltiples sistemas operativos.

RF

Radio Frecuency o radiofrecuencia, se refiere a una porción del espectro electromagnético, ubicada entre los 3 Hz y los 300 GHz.

ZIGBEE

ZigBee es el nombre de un conjunto de protocolos de alto nivel orientado a la comunicación inalámbrica en radiodifusión digital, basada en el estándar IEEE 802.15.4.

CAPITULO III:

VARIABLES E HIPÓTESIS

3.1. Variables de la investigación

3.1.1. Variable Independiente

Estructura del Frame

Un frame es una unidad de envío de datos. Es una serie sucesiva de bits, organizados en forma cíclica, que transportan información y que permiten en la recepción extraer esta información.

3.1.2. Variables Dependientes

Asignación del Dispositivo

Es la identificación que tendrá cada dispositivo dentro de una red.

Dato a transmitir

Es resultado de la lectura de los sensores y que será transmitido.

Cantidad de Dispositivos

En el desarrollo de este protocolo se permitirá que la cantidad de dispositivos que contenga sea dinámico, es decir que esta cantidad pueda aumentar de acuerdo a la distancia que se quiera alcanzar.

Cantidad de sensores

Cada dispositivo tendrá la capacidad de manejar más de un sensor a través de su microcontrolador por lo que esta cantidad será variable.

3.2. Operacionalización de variables

TABLA 3.1. OPERACIONALIZACIÓN DE VARIABLES

Variable	Tipo	Definición	Indicador
Estructura del Frame.	Independiente	Serie sucesiva de bits organizados en forma cíclica.	Frame de datos
Asignación del dispositivo.	Dependiente	Es la identificación que tendrá cada dispositivo en la red.	Número de Identificación
Dato a transmitir.	Dependiente	Resultado de la lectura de los sensores.	Lectura de los sensores
Cantidad de dispositivos	Dependiente	Cantidad de dispositivos con los que contara la red	Número de dispositivos
Cantidad de sensores	Dependiente	Cantidad de sensores con los que contara el dispositivo.	Número de sensores

Fuente: Elaboración propia.

3.3. Hipótesis general

El desarrollo de un protocolo de comunicación haciendo uso de Radio Definido por Software contribuye a la creación de redes de sensores más flexibles en cuanto a su frecuencia, cantidad de sensores y cantidad de dispositivos para que estos puedan ser implementados en áreas mayores a 1 kilómetro.

CAPITULO IV:

METODOLOGÍA

4.1. Tipo de investigación

La metodología es tipo analítico y experimental, analítico porque se realizó un análisis de los elementos que son parte de este trabajo los cuales son: topología de la red inalámbrica de sensores, hardware de cada dispositivo y software para cada nodo dentro de la red. Y experimental porque se inducirán cambios en las variables dependientes con la finalidad de observar el rendimiento del protocolo propuesto para la red.

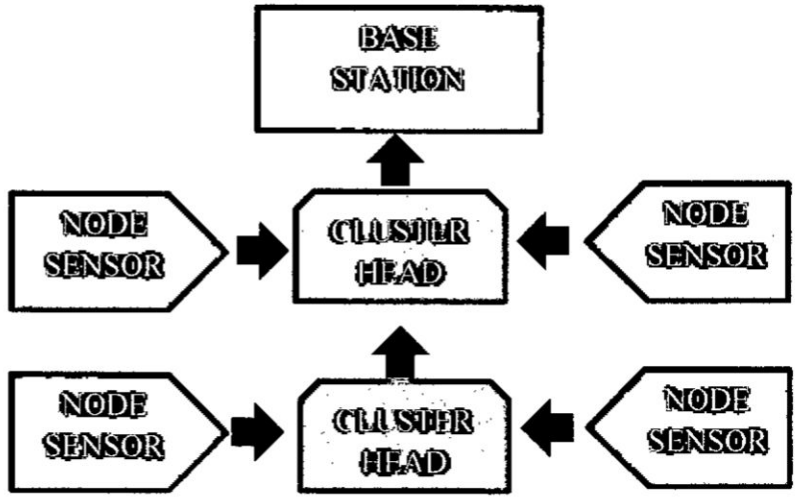
4.2. Diseño de la investigación

Se hizo uso del estado del arte de las redes de comunicación inalámbrica para diseñar de forma teórica cada nodo que conforma una red. De igual manera se diseñó de forma experimental los elementos que conforman la red para la implementación de los mismos como veremos a continuación:

4.2.1. Descripción General del Sistema

Esta red de sensores tiene 3 elementos principales dentro de su estructura: Base Station, Cluster Head y Node Sensors. Estos a su vez están distribuidos mediante una topología tipo árbol (jerárquica), en la figura 4.1 podemos ver cómo están ubicados.

FIGURA 4.1. JERARQUÍA DE LA RED



Fuente: Elaboración propia.

4.2.1.1. **Estación Base (Base Station)**

Este elemento almacena la información de todos los nodos de la red luego para luego ser visualizada mediante una GUI en la PC para administrar los datos recibidos.

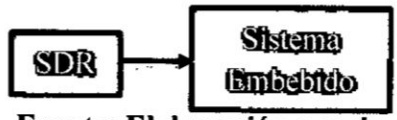
FIGURA 4.2. ELEMENTOS DE LA BASE STATION

Fuente: Elaboración propia.

4.2.1.2. **Cabecera de Grupo (Cluster Head)**

Este elemento hace de repetidor y envía la información de los nodos sensores hacia los cluster head o de ser el caso a la estación base.

FIGURA 4.3. ELEMENTOS DE LOS CLUSTER HEAD

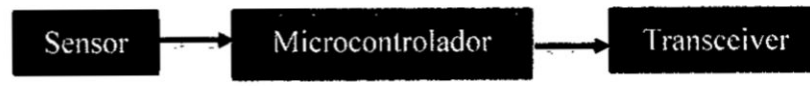


Fuente: Elaboración propia.

4.2.1.3. Nodos Sensores (Sensor Nodes)

Este elemento se encarga de recopilar la información de los sensores y mediante transceivers enviárselas a los cluster heads.

FIGURA 4.4. ELEMENTOS DE LOS NODE SENSOR



Fuente: Elaboración propia.

4.2.2. Desarrollo del Protocolo de Comunicación

El protocolo de comunicación a desarrollar constara de 2 partes, una diseñada para la comunicación entre el nodo sensor y los cluster head y otro diseñado para comunicar los cluster head entre sí, la estación base será la encargada de recepcionar y decodificar la última etapa del protocolo.

El almacenamiento del protocolo dentro del microcontrolador será a través de cadenas de caracteres (Strings), previamente se leen los datos de los sensores y se almacena en variables de formato punto flotante (float) que luego se transformaran a caracteres (FloatToString) y estos se concatenaran con los siguientes elementos del protocolo:

Formato del Frame de Node Sensor a Cluster Head

El formato que se usara para transmitir los datos del nodo sensor a los cluster head será el siguiente:

$$\{NID, N, [SID, SD]_N\}$$

Los elementos del Frame serán descritos de la siguiente manera:

- NID: Identificación del Nodo sensor (1char)
- N: Numero de sensores (1char)
- SID: Identificación del sensor (1char)
- SD: Valor del Sensor (7 char)

Formato del Frame de los Cluster Head a Base Station

El formato que se usara para transmitir los datos desde los cluster head a la estación base será el siguiente:

$$\{CD, CP, C, [NID, N, [SID, SD]_N]_C\}$$

Los nuevos elementos del Frame serán descritos de la siguiente manera:

- CD: Cluster destino (1char)
- CP: Cluster Procedencia (1char)
- C: Cantidad de Cluster (1char)

Finalmente para poder conocer el tamaño del frame del protocolo cuantificaremos la cantidad de caracteres del mismo en un ejemplo:

De NS a CH:
En este caso se envía desde el Node Sensor 2 a un Cluster Head la lectura de un único sensor de presión atmosférica

[NID2N1SID1SD1001.34]

De CH a BS:
En este caso se tiene un total de 2 Cluster Head y se envía desde el Cluster Head 2 al Cluster Head 1 el valor previamente enviado por el Node Sensor 2

[CD1CP2C2NID2N1SID1SD1001.34]

De este ejemplo podemos observar que existe un total de 27 caracteres en el protocolo, por lo tanto:

27 char → **27 bytes**

Dado que 1 bytes es equivalente a 8 bits:

27 char → 27 x 8bits → 216 bits

Entonces el tamaño total del frame sería de **216 bits.**

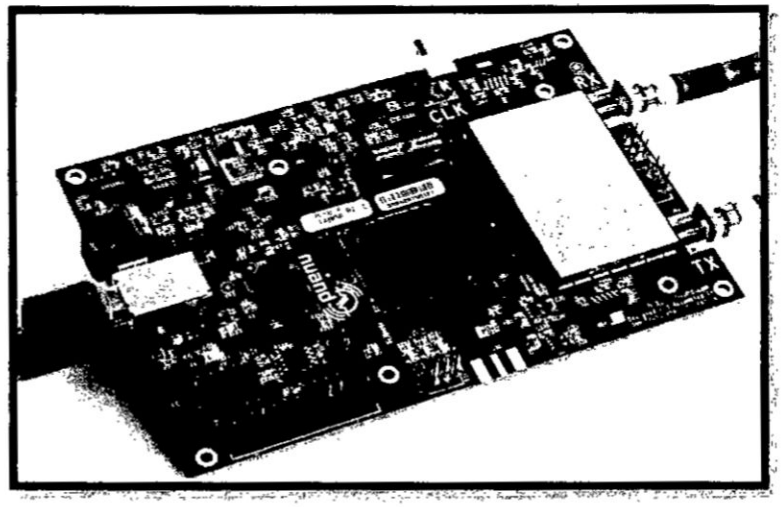
4.2.3. Elección de Dispositivos

4.2.3.1. Módulo SDR

Este es el elemento principal de toda la red por las diversas características programables que posee entre ellas modulación, frecuencia, codificación. Para tener un gran rango de frecuencias para probar la red es necesario usar un módulo de Radio Definido por Software (SDR) de última generación, además que estos tienen un gran ancho de banda y alta potencia de transmisión, entre ellos el módulo BladeRF x115 es la mejor opción, ya que posee las siguientes características: **[8]**

- Rango de Frecuencia: 300MHz - 3.8GHz
- Independiente RX/TX a 12-bit 40MSPS
- 16-bit DAC
- Alta eficiencia, bajo ruido
- Estable en Linux, Windows y Mac además soporte para GNURadio.
- Capacidad para funcionar como analizador de espectro, analizador de vector de señal, y generador de señal del vector.
- Potencia de transmision de +6dBm.

FIGURA 4.5. BLADERF X115



Fuente: [8]

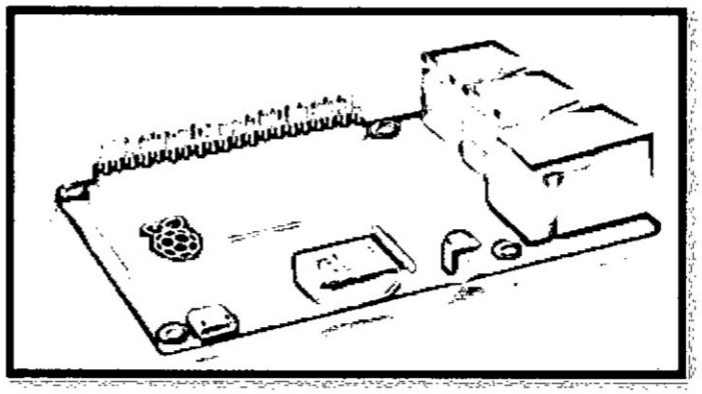
Dado que es compatible con Linux este módulo puede estar conectado directamente a una PC o a un sistema embebido que posea un sistema operativo basado en Linux, ya que esto generaría un ahorro de costos en los nodos de la red

4.2.3.2. Sistema Embebido

Actualmente en el mercado existen muchos sistemas embebidos basados en ARM con sistemas operativos basados en Linux tales como BeagleBone o Banana Pi, pero a diferencia de ellos Raspberry Pi es el que posee mayor documentación y una comunidad que está en constante crecimiento, para el desarrollo de esta tesis se hizo uso de la versión 2 del Raspberry Pi modelo B, entre sus características tenemos: **[9]**

- Unidad central basada en ARM Cortex-A7 a 900MHz
- 1GB de memoria RAM
- 4 puertos USB
- 40 pines de GPIO

FIGURA 4.6. RASPBERRY PI 2 MODELO B



Fuente: [9]

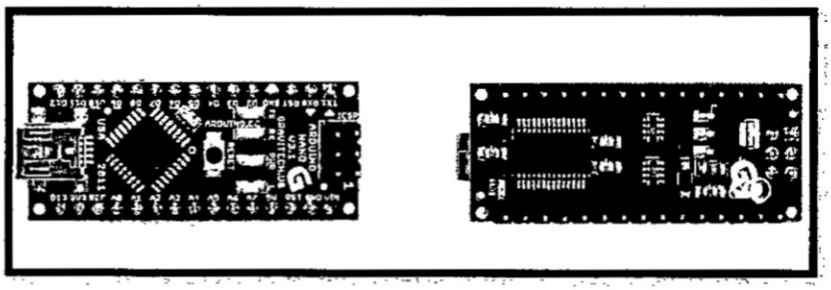
4.2.3.3. Microcontrolador

Para poder adquirir los datos de los sensores ambientales es necesario usar un microcontrolador, actualmente existen muchos módulos con microcontroladores integrados que

facilitan el trabajo de tener que diseñar una PCB dedicada, la plataforma Arduino Nano es uno de los más versátiles por su tamaño y su conexión directa a USB para programar y adquirir datos. Sus especificaciones son las siguientes: **[10]**

- Microcontrolador: ATmega328
- Entrada de Voltaje (limites): 6-20 V
- Pines Digitales(D0-D13): 14
- Pines Analógicos (A0-A7): 8
- Memoria Flash: 32 KB
- Velocidad de Reloj: 16 MHz
- Soporta comunicación USART, SPI y I2C.

FIGURA 4.7. ARDUINO NANO V3



Fuente: [10]

4.2.3.4. **Sensores**

Dado que una de las aplicaciones de la red será el monitorio de parámetros ambientales, la referencia ideal para la elección de sensores es la guía para la medición de variables meteorológicas de la organización mundial de meteorología (WMO) **[11]**. Para el desarrollo de esta tesis se observaran 3 parámetros principales y las recomendaciones de la WMO para cada uno son:

Temperatura

- Rango de operación: -40 a 100°C
- Resolución: 0,1 °C
- Exactitud: ± 0,4°C

Humedad

- Tipo de sensor: Capacitivo
- Rango de operación: 5 a 100%
- Resolución: 1%
- Exactitud: ± 3%

Presión Atmosférica

- Rango de operación: 500 – 1 080 hPa
- Resolución: 0.1 hPa
- Exactitud: ± 0.1hPa

Por lo tanto, dadas las existentes recomendaciones del WMO los sensores digitales elegidos que cumplen estos requisitos son los siguientes:

DHT22

Sensor de temperatura y humedad relativa con salida digital (También llamado AM2302) **[12]**

Alimentación: 3.3-6V DC

Elemento de sensado: Polímero capacitivo

Rango de Operación:

- Humedad: 0-100%RH
- Temperatura: -40~80Celsius

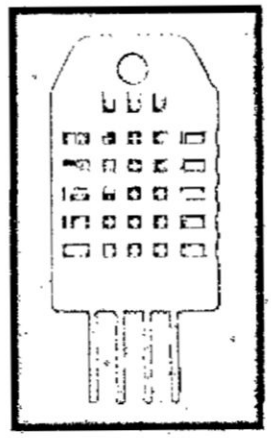
Exactitud:

- Humedad: +-2%RH(Max +-5%RH)
- Temperatura: <+-0.5Celsius

Resolución:

- Humedad: 0.1%RH
- Temperatura: 0.1Celsius

FIGURA 4.8. SENSOR DHT22



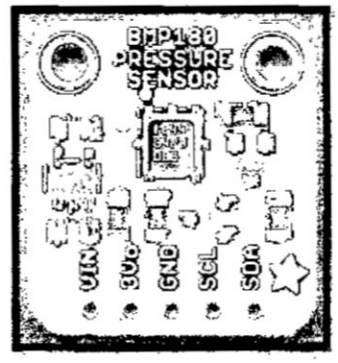
Fuente: [12]

BMP180

Sensor digital de presión atmosférica **[13]**

- Alimentación: 1.8 - 3.6V DC
- Rango de Operación: 300 – 1100 hPa
- Resolución: 0.01 hPa
- Exactitud: ±0.12

FIGURA 4.9. SENSOR BMP180



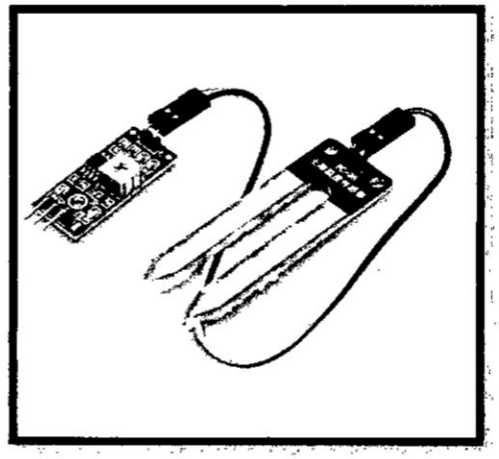
Fuente: [12]

FC-28

Sensor de humedad del suelo **[17]**

- Alimentación: 3.3 - 5V DC
- Rango de Operación: 0 – 100%
- Resolución: 0.02

FIGURA 4.10. SENSOR FC-28



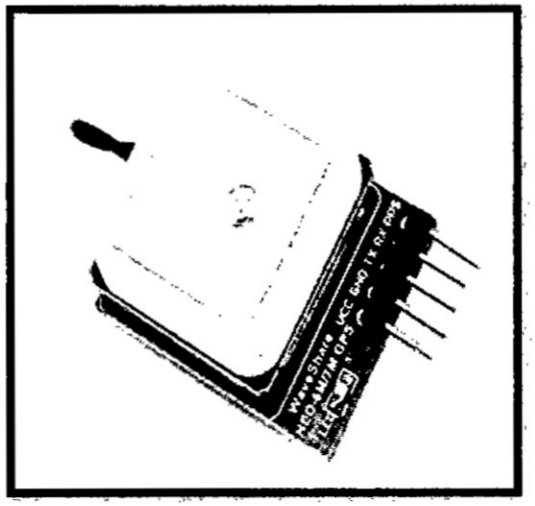
Fuente: [17]

GPS NEO-6M

Módulo de Global Positioning System (GPS) **[18]**

- Alimentación: 3.3 - 5V DC
- Cantidad de canales : 50
- Exactitud: 2.5m
- Sensibilidad de Navegación: -161dBm

FIGURA 4.11. MODULO GPS NEO-6M



Fuente: [18]

4.2.3.5. **Transceiver**

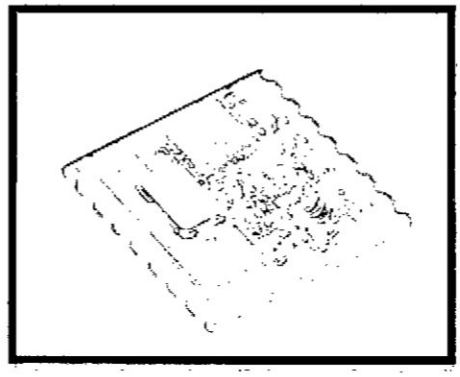
Para abaratar los costos de los nodos sensores se usaran transceivers de frecuencia variable que puedan ser manejados por los microcontroladores y además estos se comunicaran con los módulos de SDR.

RFM22B

Módulo transceiver de frecuencia programable **[15]**

- Alimentación: 1.8 – 3.6 v
- Rango de frecuencias: 240 – 960 MHz
- Sensibilidad: -121 dBm
- Máxima Potencia de Transmision: +20dBm
- Modulaciones: FSK, GFK y OOK
- RSSI Digital
- Comunicación SPI

FIGURA 4.12. TRANSCEIVER RFM22B



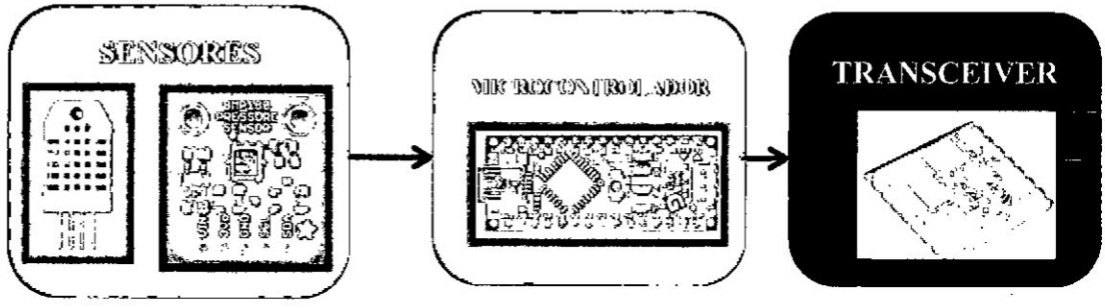
Fuente: [16]

4.2.4. Diseño de los Node Sensors

4.2.4.1. Partes

El nodo sensor constara en 3 partes mostradas en los siguientes bloques:

Fig. 4.13. Bloques del Sensor Node

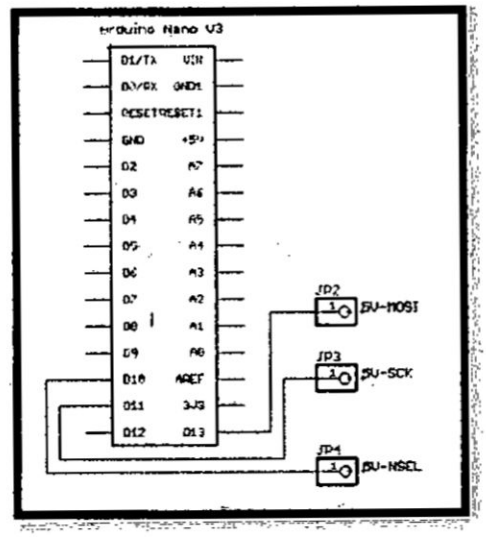


Fuente: Elaboración Propia

4.2.4.2. Diseño de Circuitos

Para lograr una comunicación entre el microcontrolador y el transceiver es necesario usar un convertidor de nivel, ya que poseen diferentes niveles de voltaje en sus entradas y salidas, es por ello que se usaran los pines MOSI, SCK y un pin arbitrario, en este caso el D10 para lograr establecer una comunicación mediante el protocolo SPI.

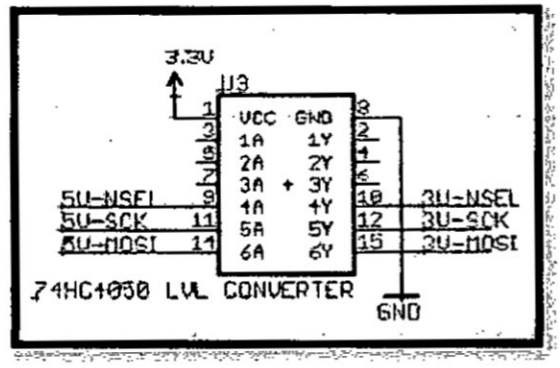
FIGURA 4.14. SALIDAS USADAS EN ARDUINO NANO



Fuente: Elaboración propia.

Dado que usaremos el protocolo SPI entonces convertiremos esas salidas en niveles de 5v a 3.3v mediante un buffer. Para alimentar el buffer se usan los pines 1 y 8 en el caso del 74HC4050, este esquema variara dependiendo si se usa tecnología TTL (74HC450) o CMOS (CD4050).

FIGURA 4.15. CONEXIÓN DE BUFFER 74HC4050



Fuente: Elaboración propia.

Cuando tenemos todas las salidas y entradas al mismo nivel podemos realizar la conexión al RFM22B sin correr el riesgo de dañar el transceiver.

Para el esquemático final se agregaron pines de entrada para conexión directa con sensores, los pines usados son los siguientes:

A4, A5 → Bus I2C

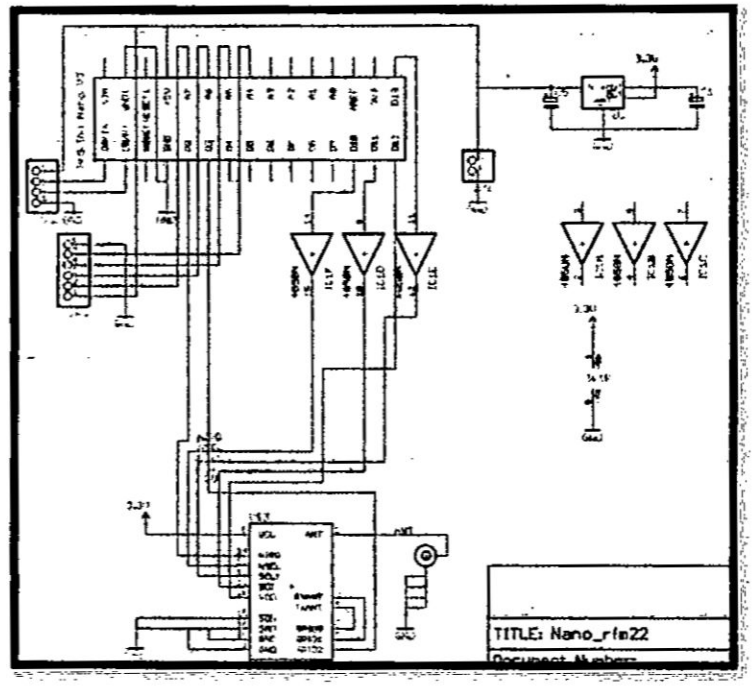
A6, A7 → ADC

D0, D1 → USART

Para que el transceiver y el microcontrolador puedan usar una misma alimentación se optó por usar un regulador de 5 a 3.3 voltios LD33 en empaquetadura superficial para optimizar el tamaño de la placa.

En la siguiente imagen podemos observar el diseño completo con las entradas de sensores colocadas en los pines JP2 y JP3.

FIGURA 4.16. ESQUEMÁTICO FINAL DEL NODO SENSOR



Fuente: Elaboracion propia.

4.2.4.3. Diseño PCB

Para un mejor desempeño de las pruebas finales se diseñó una placa de circuito impreso del nodo sensor.
El diseño del PCB final sería el siguiente, dado que tanto el transceiver, el regulador y los condensadores son de formato smd, es decir usan componentes superficiales.
El diseño de la placa fue de una sola capa.

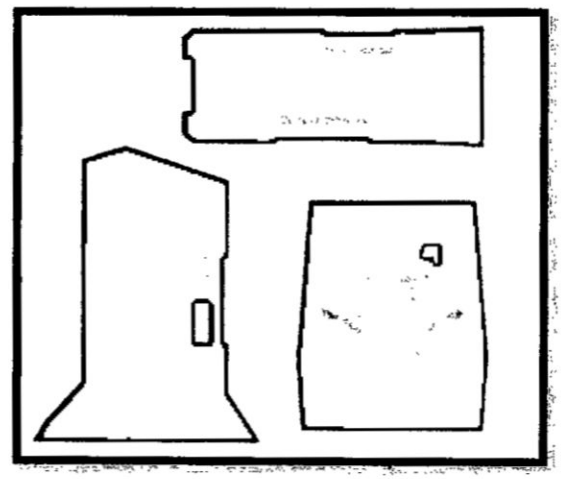
FIGURA 4.17. DISEÑO FINAL DEL PCB DEL NODO SENSOR

Fuente: Elaboracion propia.

4.2.4.4. Diseño de estructura de soporte

Dado que el nodo sensor está expuesto a condiciones agrestes, es necesario diseñar un case o estructura de soporte que pueda mantener en buenas condiciones la placa de circuito impreso además de la batería de alimentación.

FIGURA 4.18. DISEÑO 3D DE LA ESTRUCTURA



Fuente: Elaboración propia.

Para la alimentación del nodo sensor se está haciendo uso de una batería de Iones de Litio (Li-Ion) de la marca Ultrafire, sus capacidades son las siguientes:

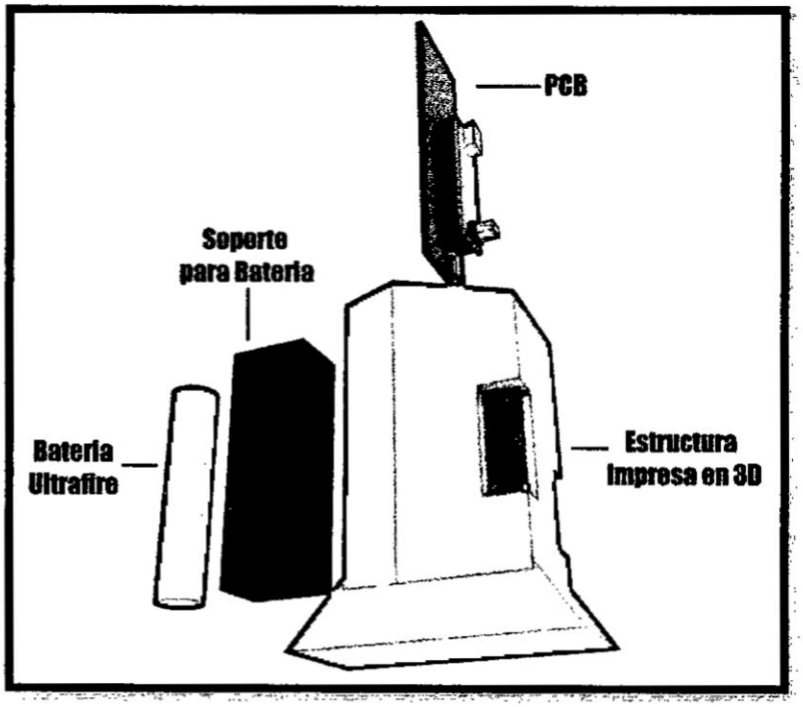
Voltaje: 3.7v
Capacidad: 9800 mAh
Modelo 18650

FIGURA 4.19. BATERÍA ULTRAFIRE

Fuente: Elaboración propia.

En la siguiente imagen podemos observar todos los componentes que componen los node sensor y el modo en que ingresan a su estructura impresa en 3D.

Fig. 4.20. Armado del nodo sensor y sus componentes

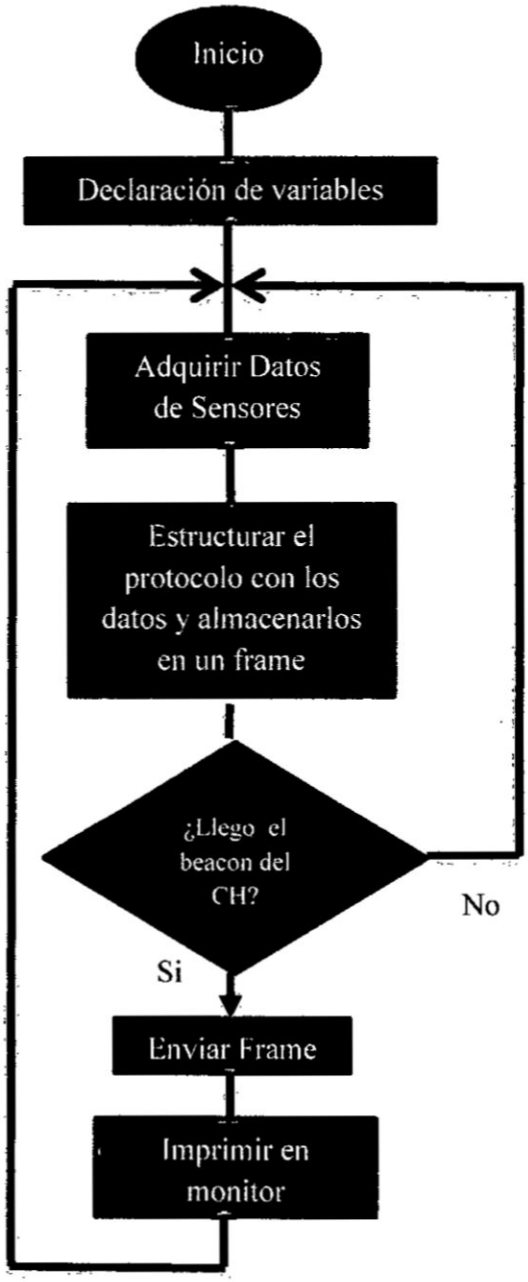


Fuente:Elaboracion propia.

4.2.4.5. Diagrama de Flujo

El diagrama de flujo que describe el comportamiento del nodo sensor en la ejecución de la red será el siguiente:

FIGURA 4.21. DIAGRAMA DE FLUJO DEL NODO SENSOR



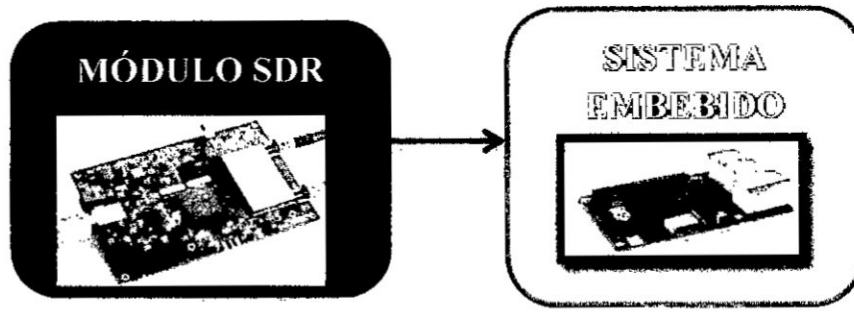
Fuente: Elaboración propia.

4.2.5. Diseño de los Cluster Head

4.2.5.1. Partes

Los cluster head contaran con 2 partes mostradas en los siguientes bloques:

FIGURA 4.22. BLOQUES DEL CLUSTER HEAD

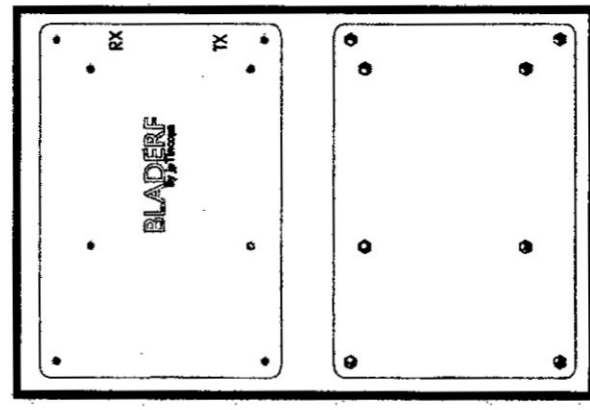


Fuente: Elaboración propia.

4.2.5.2. Diseño de estructura de soporte

Para poder proteger los módulos de SDR se diseñó un case en 3D a la medida del modelo BladeRF:

FIGURA 4.23. DISEÑO 3D PARA ESTRUCTURA



Fuente: Elaboración propia.

De igual manera se procedio a emsamblar el modulo con la estructura impresa en 3D como se puede apreciar en la siguiente imagen:

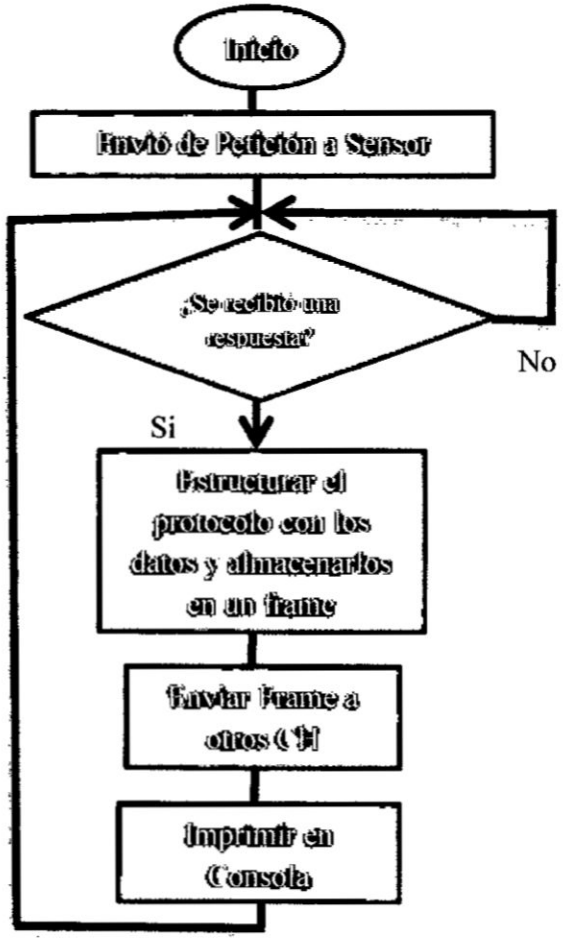
FIGURA 4.24. ESTRUCTURA 3D PARA BLADERF

Fuente: Elaboración propia.

4.2.5.3. Diagrama de Flujo

El diagrama de flujo del cluster head será de la siguiente manera

FIGURA 4.25. DIAGRAMA DE FLUJO DEL CLUSTER HEAD



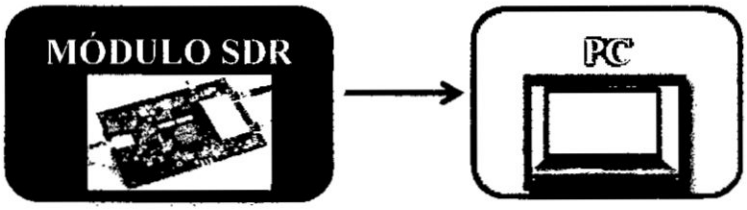
Fuente: Elaboración propia.

4.2.6. Diseño de la Base Station

4.2.6.1. Partes

La estación base constara de 2 partes mostradas en los siguientes bloques:

FIGURA 4.26. BLOQUES DEL BASE STATION

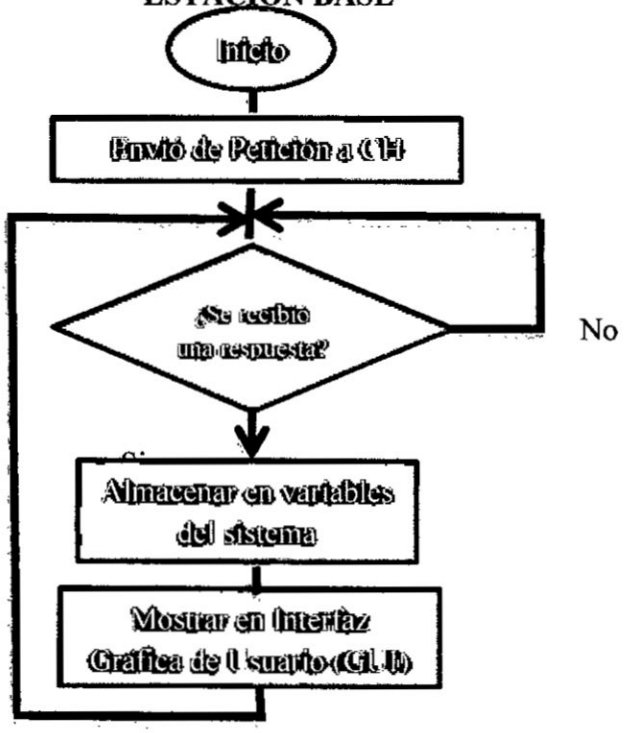


Fuente: Elaboración propia.

4.2.6.2. Diagrama de Flujo

El diagrama de flujo de la estación base será de la siguiente manera

FIGURA 4.27. DIAGRAMA DE FLUJO DE LA ESTACIÓN BASE



Fuente: Elaboración propia.

4.2.7. Diseño de Algoritmos y software

4.2.7.1. Comunicación entre Nodo Sensor y Cluster Head

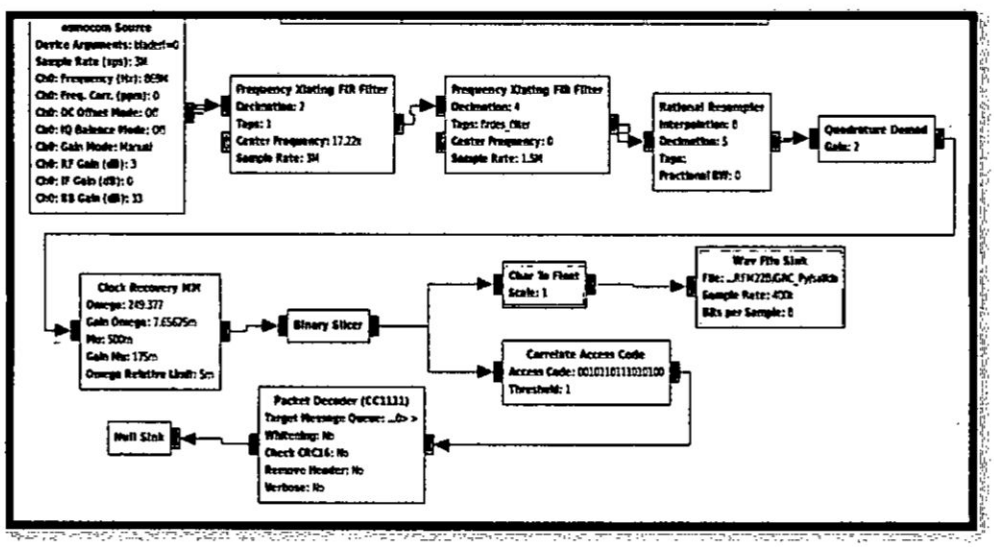
Para poder adquirir los datos obtenidos por el nodo sensor transmitidos por el RFM22B mediante el módulo de SDR (BladeRF) se hará uso del programa GNU Radio Companion (GRC) para decodificar toda la trama y luego trabajar directamente con lenguaje Python a nivel de consola:

Gnu Radio Companion

Los Bloques de GRC que se usaran para decodificar la trama del RFM22B serán los siguientes:

- Osmocom Source
- Frequency Xlating FIR Filter
- Rational Resampler
- Quadrature Demod
- Clock Recovery MM
- Binary Slicer

FIGURA 4.28. DIAGRAMA DE BLOQUES GRC PARA DECODIFICACIÓN DE RFM22B.



Fuente: Elaboración propia

FIGURA 4.29. FRAME DE DATOS DE RFM22B

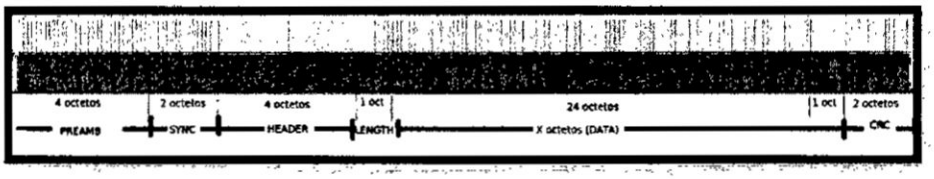
Fuente: Elaboración propia

Con estos Bloques se logra Decodificar la modulación FSK para obtener un tren de pulso que veremos a continuación:

Analizando de forma visual interpretaremos la trama de datos de acuerdo a la hoja de datos del RFM22B **[15]**

Observamos los siguientes bloques dentro de la trama:

FIGURA 4.30. PARTES DEL FRAME DE RFM22B



Fuente: Elaboración propia.

- Preamble
- Sync
- Header
- Length
- Data
- CRC

Para que solo trabajemos el bloque Data es necesario usar otros bloques de GRC dentro del programa principal, esos son:

- Corelate Access Code
- Packet Decoder

Una vez completados todos los bloques de GRC se genera un archivo de extensión .py el cual puede trabajarse directamente en lenguaje Python.

Para decodificar y almacenar correctamente el dato transmitido se hizo uso de una librería para CRC el cual confirma el dato obtenido con el cálculo del mismo, de ser cierto, se transmite. El Script será anexado a esta tesis.

4.2.7.2. Comunicación entre Cluster Head y Base Station

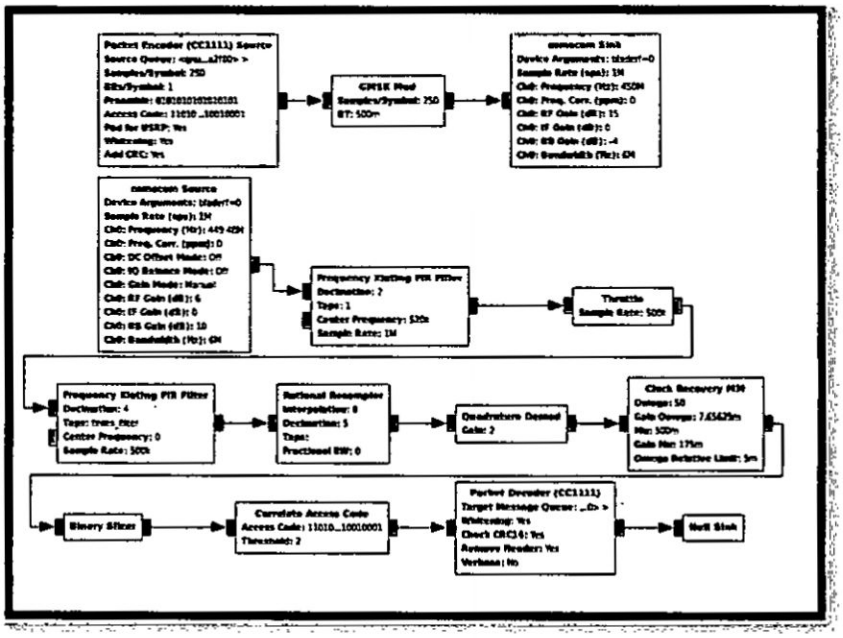
Para poder recibir los datos obtenidos por el cluster y retransmitirlos hacia la estación base mediante el módulo de SDR (BladeRF) se hará uso del programa GNU Radio Companion (GRC) para decodificar toda la trama y luego trabajar directamente con lenguaje Python a nivel de consola:

Los Bloques de GRC que se usaran serán los siguientes:

Para recepción de datos:

- Packet Encoder
- GMSK Mod
- Osmocom Sink

FIGURA 4.31. DIAGRAMA DE BLOQUES GRC PARA COMUNICACIÓN ENTRE CH Y BS



Fuente: Elaboración propia.

Para transmisión de datos:

- Osmocom Source
- Frequency Xlating FIR Filter
- Rational Resampler
- Quadrature Demod
- Clock Recovery MM
- Binary Slicer
- Corelate Access Code
- Packet Decoder

Una vez decodificado el dato a la salida del packet decoder es almacenada en una variable del programa para luego ser retransmitida por el osmocom sink. El Script será anexado a esta tesis.

4.3. Diseño de la Interfaz Gráfica de Usuario (GUI)

Para una correcta administración de los datos recibidos por la estación base se ha diseñado una interfaz gráfica de usuario (GUI) en lenguaje de alto nivel Python 2.7, para la generación de ventanas se hizo uso de la librería GTK+ 3.0

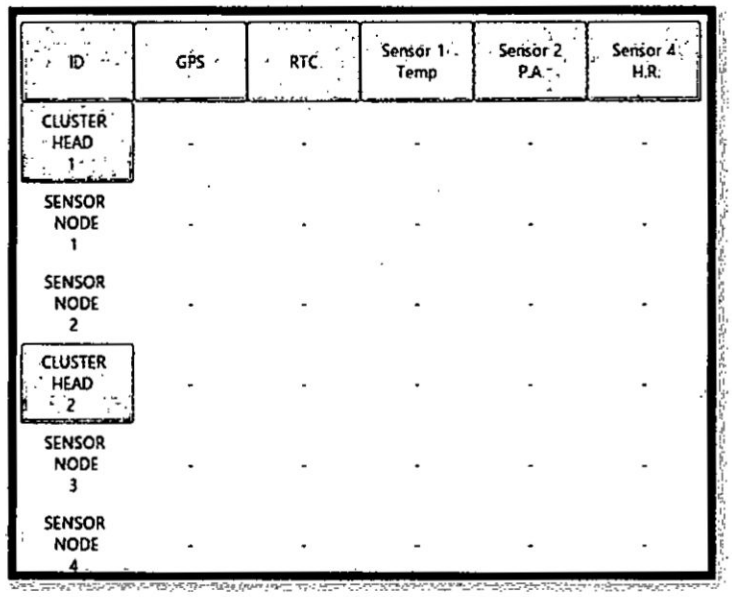
FIGURA 4.32. LOGO GTK

Fuente: [20]

En esta interfaz se visualizaran los siguientes parámetros:

- RTC(Hora)
- GPS (Sensor 1)
- Temperatura (Sensor 2 y 3)
- Humedad Relativa (Sensor 2)
- Presión Atmosférica (Sensor 3)
- Humedad del suelo (Sensor 4)

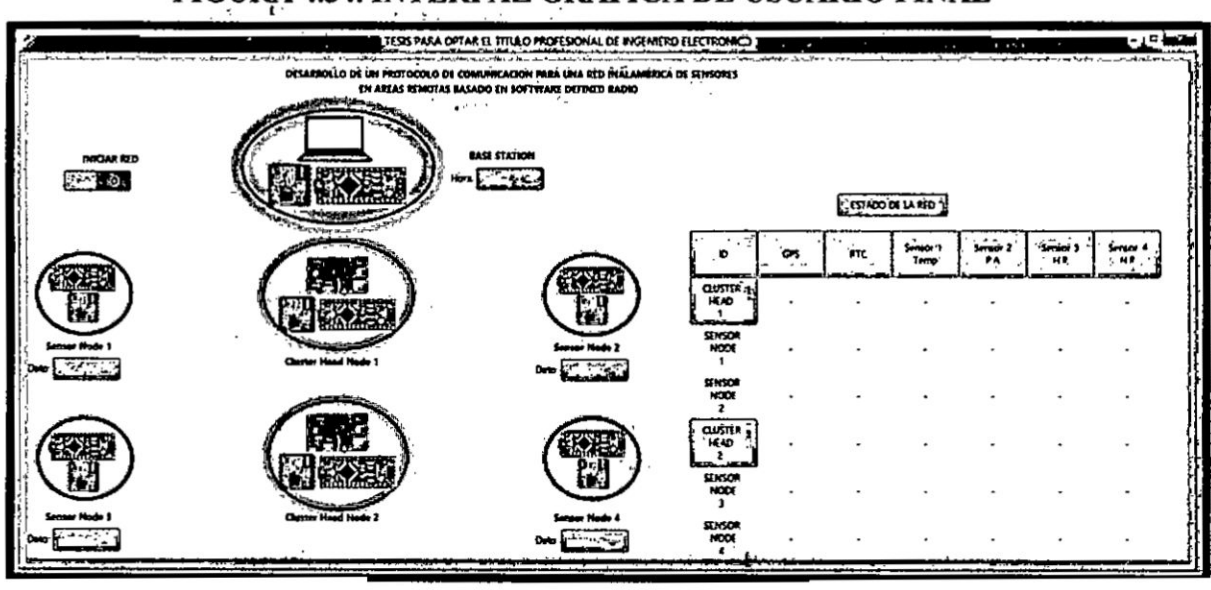
FIGURA 4.33. PARTE DE INTERFAZ GRÁFICA DE USUARIO (GUI)



Fuente: Elaboración propia.

El diseño final de la interfaz es mostrado en la figura 4.34.

FIGURA 4.34. INTERFAZ GRÁFICA DE USUARIO FINAL



Fuente: Elaboración propia.

Pérdida de cable Rx (LfRx): 8.71 dB

Por lo tanto, se usaron los datos que se conocen para calcular la máxima distancia teórica del enlace:

$$PTx - LfTx + GTx - L + GRx - LfRx = PRx$$
$$6 - 8.3 + 5 - (32.4 + 20\log(D) + 20\log(460)) + 4.5 - 8.71 = -82$$
$$80.49 = 32.4 + 20log(D) + 20x(2.662)$$
$$-5.15 = 20log(D)$$
$$D = 589m$$

4.4.2. Máxima distancia entre Cluster Head y Node Sensor

En este enlace los parámetros serán dados por los RFM22B los cuales son los siguientes:

- Potencia de Transmisión: 13dBm
- Frecuencia de trabajo: 460MHz
- Sensibilidad del Receptor: -121 dBm
- Ganancia de la antena Transmisión: 0dB
- Ganancia de la antena Recepción: 0dB
- Pérdida en los cables: En este caso no existe perdida en los cables ya que la antena está directamente conectada al circuito, si se considera una pérdida de los conectores por lo que esta seria 2.5dB para recepción y 2.5dB para transmisión

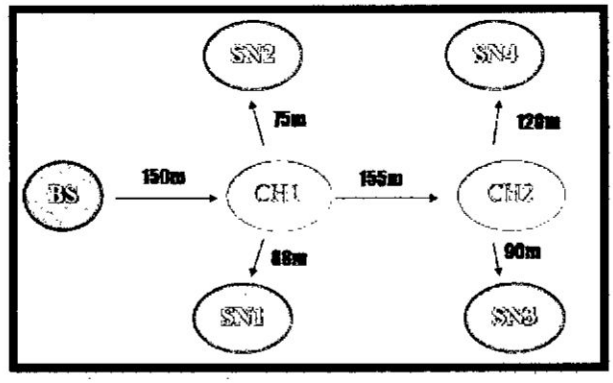
Por lo tanto, se usaron los datos que se conocen para calcular la máxima distancia teórica del enlace:

$$PTx - LfTx + GTx - L + GRx - LfRx = PRx$$
$$13 - 2.5 + 0 - (32.4 + 20\log(D) + 20\log(460)) + 0 - 2.5 = -121$$
$$129 = 32.4 + 20log(D) + 20x(2.662)$$
$$43.36 = 20log(D)$$
$$D = 322m$$

4.5. Pruebas Finales en Campo

Debido a la extensión máxima de las instalaciones del IPEN, los nodos quedaron distribuidos de la siguiente manera:

FIGURA 4.35. DISTANCIA ENTRE NODOS DEL SISTEMA



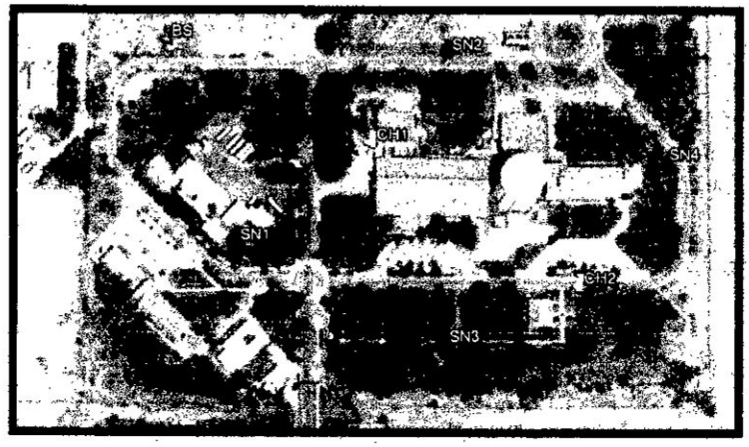
Fuente: Elaboración propia.

La localización de los nodos del sistema son los siguientes:

BS → **Lat.:** 11°48'2.67"S, **Long:** 77° 0'49.74"O
CH1 → **Lat.:** 11°48'0.21"S, **Long:** 77° 0'45.44"O
SN1 → **Lat.:** 11°48'3.73"S, **Long:** 77° 0'45.23"O
SN2 → **Lat.:** 11°47'57.87"S, **Long:** 77° 0'46.21"O
CH2 → **Lat.:** 11°47'58.19"S, **Long:** 77° 0'40.54"O
SN3 → **Lat.:** 11°48'1.18"S, **Long:** 77° 0'41.04"O
SN4 → **Lat.:** 11°47'54.61"S, **Long:** 77° 0'41.91"O

Por lo que la máxima distancia del sistema será 425m por las condiciones en las que se llevaron las pruebas, podemos observar una imagen satelital de la ubicación de cada uno de los puntos dentro de las instalaciones del IPEN.

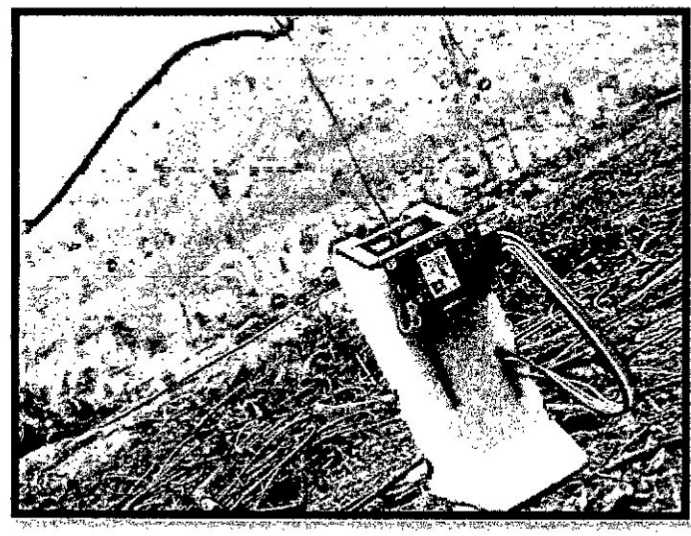
FIGURA 4.36. UBICACIÓN DE LOS NODOS DEL SISTEMA



Fuente: Elaboracion propia.

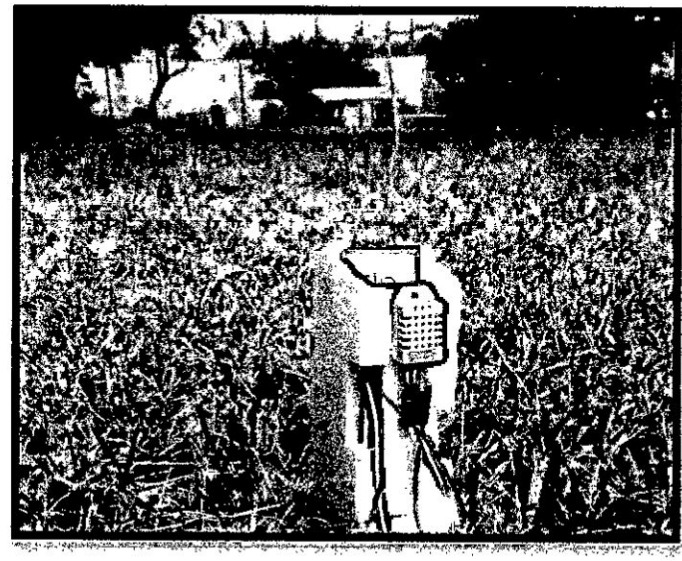
Adicionalmente se muestran los 4 nodos sensores en funcionamiento:

FIG. 4.37. SENSOR NODE 1 EN FUNCIONAMIENTO



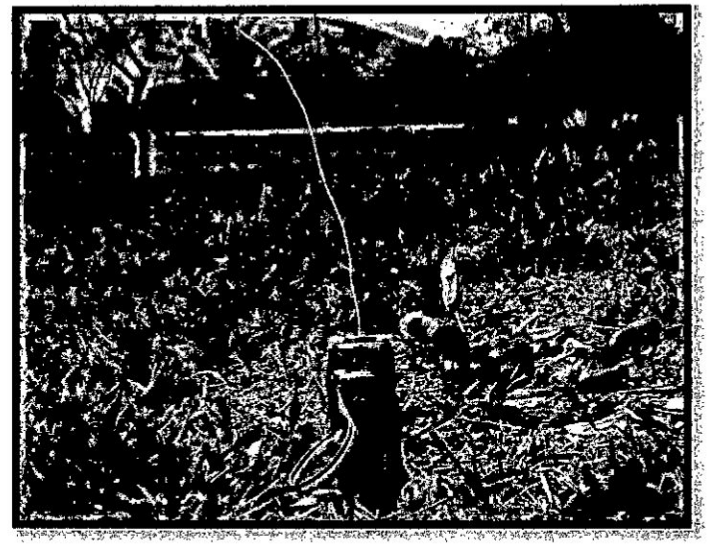
Fuente: Elaboracion propia.

FIG. 4.38. SENSOR NODE 2 EN FUNCIONAMIENTO



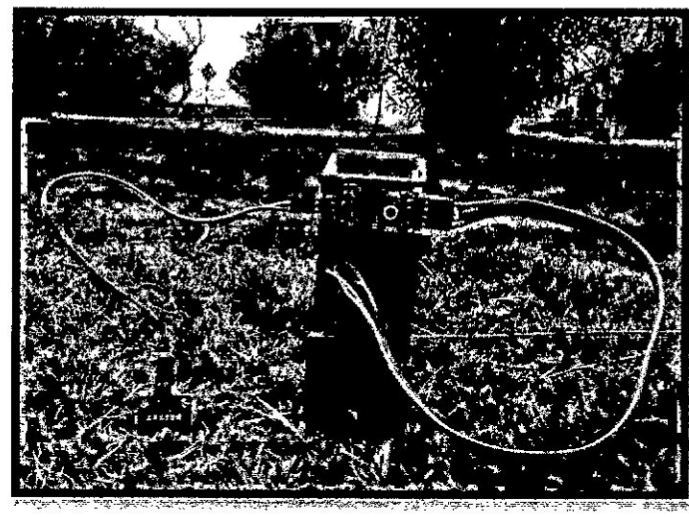
Fuente: Elaboracion propia.

FIG. 4.39. SENSOR NODE 3 EN FUNCIONAMIENTO



Fuente: Elaboracion propia.

FIG. 4.40. SENSOR NODE4 EN FUNCIONAMIENTO



Fuente: Elaboracion propia.

4.6. Técnicas e instrumentos de recolección de datos.

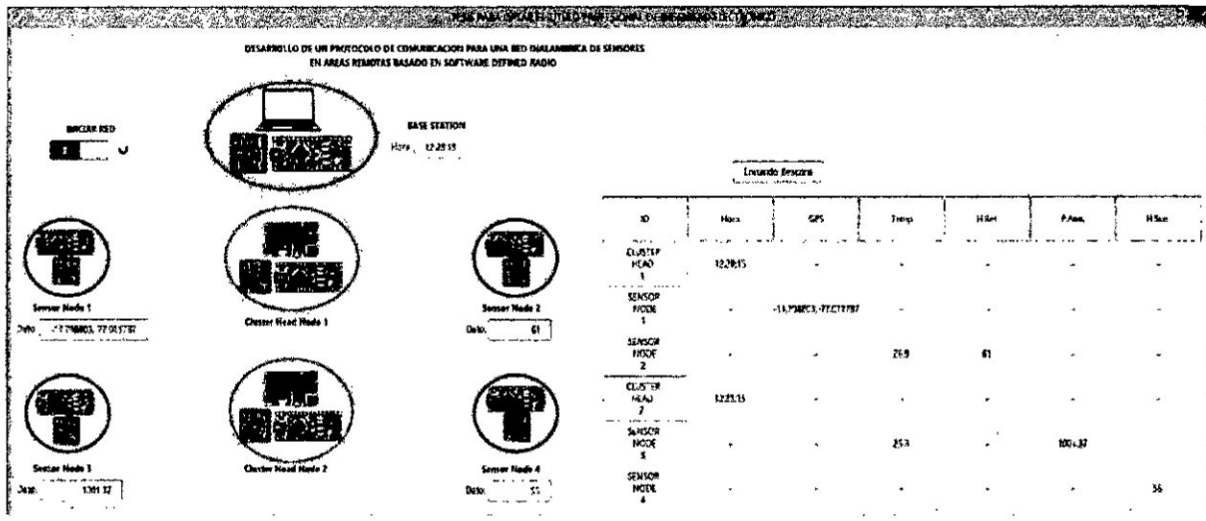
Para la recolección de datos se hará uso de algoritmos diseñados en el lenguaje de programación Python en su versión 2.7 que permitan visualizar el histórico de los datos y guardado en formato de tablas para ser visualizado en hojas de datos, estas llevaran una estampa de tiempo en cada hora, así mismo esto será visualizado mediante la GUI.

CAPITULO V:

RESULTADOS

Los resultados de este desarrollo se muestran en tiempo real mediante la GUI donde se visualizan los 4 parámetros de los sensores usados en cada nodo.

FIGURA 5.1. GUI EN FUNCIONAMIENTO



Fuente: Elaboración propia.

Este script realizado en Python 2.7 permite poder generar un conjunto de datos en formato de tablas (csv) para luego pueda ser visualizado en un programa de hojas de datos como Microsoft Excel.

En la tabla 5.1 se muestran los datos obtenidos del sistema en funcionamiento donde:

A: Hora
B: Temperatura 1
C: Humedad Relativa
D: Temperatura 2
E: Presión Atmosférica
F: Humedad del Suelo

TABLA 5.1. DATOS ADQUIRIDOS POR LA RED

	A	B	C	D	E	F
1	12:28:15	26.9	61	25.3	1001.37	56
2	13:28:15	26.2	61	25.1	1001.36	55
3	14:28:15	26.2	61	25.1	1001.36	55
4	15:28:15	26.2	61	25.1	1001.38	55
5	16:28:15	25.3	63	25.3	1001.4	55
6	17:28:15	24.4	63	25.4	1001.4	56
7	18:28:15	24.4	61	25.4	1001.4	56
8	19:28:15	24.5	62	25.4	1001.4	56
9	20:28:15	24.4	62	26.1	1001.4	56
10	21:28:15	25.2	62	26.1	1001.4	54
11	22:28:15	25.1	61	25.3	1001.4	54
12	23:28:15	25.2	60	25.3	1001.39	54
13	00:28:15	25.2	61	25.2	1001.39	54
14	01:28:15	25.5	61	25.2	1001.39	56
15	02:28:15	25.4	61	25.2	1001.39	56
16	03:28:15	25.4	61	25.2	1001.39	55
17	04:28:15	25.4	63	25.3	1001.39	55
18	05:28:15	26.1	63	26	1001.38	56
19	06:28:15	26.1	62	26	1001.37	56
20	07:28:15	26.4	62	26	1001.37	56
21	08:28:15	263	62	25.7	1001.38	56

Fuente: Elaboración propia.

CAPITULO VI:

DISCUSION DE RESULTADOS

6.1 Contrastación de hipótesis con los resultados

- Se demostró que el desarrollo de este sistema contribuye a la creación de nuevas redes de sensores inalámbricos flexibles ya que puede agregar de forma eficiente nodos centrales (cluster head) como nodos sensores (sensor nodes) para lograr obtener mayores distancias de acuerdo a las necesidades de cada sistema de telemetría.

- Se buscaba establecer una red de alcance mayor a un kilómetro, se comprobó de manera teórica que las capacidades de los nodos que conforman el sistema tienen un alcance de 1500 metros, por las condiciones de las pruebas solo se pudo comprobar el funcionamiento con una distancia de 425m de distancia entre la estación base y el ultimo nodo sensor.

6.2 Contrastación de resultados con otros estudios similares

- A diferencia de los estudios similares de redes de sensores, este sistema al hacer uso de SDR tiene la capacidad de usar diferentes frecuencias de transmisión lo cual significa una ventaja sustancial ya que puede usar un rango de canales no comerciales como el 460MHz en el cual se hicieron las pruebas finales, se espera que estos canales sean de libre uso en el momento que el país pase por un apagón analógico.

CAPITULO VII:

CONCLUSIONES

- Se comprobó que mediante las herramientas de SDR se pueden decodificar diferentes Frames de transmisores, en este caso la decodificación del RFM22B fue exitosa

- Se comprobó también que un sistema embebido como el Raspberry Pi puede soportar el trabajo de administrar un módulo de SDR de forma eficiente.

- El desarrollo del protocolo fue exitoso ya que este pudo transmitir de forma ordenada y eficiente todos los datos obtenidos por los nodos sensores así como separar cada parámetro ambiental medido.

CAPITULO VIII:

RECOMENDACIONES

- Se recomienda para próximos trabajos aprovechar la capacidad de establecer parámetros de forma automática en los módulos de SDR mediante algoritmos.
- Se recomienda también exigir al máximo la capacidad de transmisión de los módulos transceiver ya que suman en conjunto grandes distancia para establecimiento de enlaces.

CAPITULO IX:

REFERENCIAS BIBLIOGRÁFICAS

[1] TOMASI, Wayne. **Sistemas de Comunicaciones Electrónicas.** México. Editorial Pearson Educación. 4ta edición. 2003.

[2] WYGLINSKI, Alexander. **Cognitive Radio Communications and Networks.** USA. Editorial Elsevier. 1ª Edición. 2010.

[3] SUÁREZ BARÓN, Juan Carlos. **Diseño Y Construcción De Un Sistema De Monitoreo Para Invernaderos Apoyado Con Tecnología Zigbee.** Tesis para optar el título profesional de Ingeniero Electrónico. Boyacá. Universidad Nacional Abierta Y A Distancia (UNAD). 2013.

[4] VILLÓN VALDIVIEZO, Daniel. **Diseño De Una Red De Sensores Inalámbrica Para Agricultura De Precisión.** Tesis para optar el título profesional de Ingeniero Electrónico. Lima. Pontificia Universidad Católica del Perú (PUCP). 2009.

[5] MANUEL FERNANDEZ BARCELL. **Introducción a las redes inalámbricas de sensores (WSN)** Disponible en: http://www.mfbarcell.es/conferencias/wsn.pdf . Articulo web. Consultada el 20 de agosto del 2016.

[6] ANDRÉS PÉREZ. **Topologías de Red WSN** Disponible en: http://wirelesssensornetworkproyecto.blogspot.pe/2010/07/topologias-de-red-wsn.html . Articulo web. Consultada el 30 de julio del 2016.

[7] 2EECINFORMATICAAP. **Conceptos basicos de Comunicaciones** Disponible en: http://2eecinformaticaap.blogspot.pe/2012/01/telecomunicacion-comunicacion-distancia.html . Articulo web. Consultada el 30 de julio del 2016.

[8] Nuand. **BladeRF x115.** Disponible en: https://www.nuand.com/blog/product/bladerf-x115/. Articulo web. Consultado el 25 de julio del 2016.

[9] Raspberry Pi Foundation. **RASPBERRY PI 2 MODEL B.** Disponible en: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/ . Articulo Web. Consultado el 2 de Agosto del 2016.

[10] Genuino. **Arduino Nano.** Disponible en: https://www.arduino.cc/en/Main/ArduinoBoardNano . Articulo web. Consultado el 25 de julio del 2016.

[11] World Meteorological Organization. **Procedure for Updating the Guide to Meteorological Instruments and Methods of Observation (wmo-no. 8, cimo guide).** Disponible en: http://www.wmo.int/pages/prog/www/IMOP/CIMO-Guide.html . Articulo web. Consultado el 4 de Agosto del 2016.

[12] Aosong Electronics Co.,Ltd . **DHT22.** Disponible en: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf . Articulo web. Consultado el 4 de Agosto de 2016.

[13] Bosch Sensortec. **BMP180 Digital pressure sensor.** Disponible en: https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf . Articulo web. Consultado el 10 de setiembre de 2016.

[14] Adafruit. **BMP180 Barometric Pressure / Temperature / Altitude Sensor- 5V ready.** Disponible en: https://www.adafruit.com/product/1603 . Articulo web. Consultado 10 de setiembre de 2016.

[15] HopeRF. **RFM22B Transceiver Module.** Disponible en: https://www.sparkfun.com/datasheets/Wireless/General/RFM22B.pdf . Articulo web. Consultado el 20 de octubre de 2016.

[16] Sparkfun. **RFM22B-S2 SMD Wireless Transceiver.** Disponible en: https://www.sparkfun.com/products/10153 . Articulo web. Consultado el 20 de octubre de 2016.

[17] Micropik. FC-28 Soil Mesure Humidity. Disponible en: http://www.micropik.com/PDF/FC_28.pdf . Articulo web. Consultado el 29 de Enero de 2017.

[18] u-Blox. NEO-6 GPS Modules. Disponible en: https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/GY-NEO6MV2-GPS-Module-Datasheet.pdf . Articulo web. Consultado el 29 de Enero de 2017.

[19] **Seguridad Wireless.** Conectores y cables de antena utilizados en el universo wireless. Disponible en: http://www.seguridadwireless.net/hwagm/conectores-cables.html Artículo web. Consultado el 29 de Enero de 2017.

[20] **Geekland.** Qué son las librerías GTK y que versión usamos. Disponible en: https://geekland.eu/que-son-librerias-gtk-version-usamos/ Artículo web. Consultado el 29 de Enero de 2017.

CAPITULO X: ANEXOS

ANEXO N°1: Matriz de Consistencia

DESARROLLO DE UN PROTOCOLO DE COMUNICACIÓN PARA UNA RED INALAMBRICA DE SENSORES EN ÁREAS REMOTAS BASADO EN SOFTWARE DEFINED RADIO

DEFINICION DEL PROBLEMA	OBJETIVOS	HIPOTESIS	VARIABLES	METODOLOGIA
¿Es posible establecer una comunicación mayor a 1 kilómetro en redes de sensores inalámbricas en áreas remotas?	**Objetivo General** Crear un protocolo de comunicación para una red inalámbrica de sensores mediante el uso de SDR. **Objetivos Específicos** Integrar los módulos de SDR con microcontroladores para crear los nodos que conforman la red. Diseñar algoritmos de programación para la transmisión y recepción de datos para todos los nodos que conforman la red. Diseñar algoritmos de programación que logren establecer la lectura de datos de sensores usando microcontroladores.	El desarrollo de un protocolo de comunicación haciendo uso de Radio Definido por Software contribuye a la creación de redes de sensores más flexibles en cuanto a su frecuencia, número de sensores y número de dispositivos para que estos puedan ser implementados en áreas mayores a 1 kilómetro.	Estructura del Frame Asignación del dispositivo Dato a transmitir. Número de dispositivos. Número de sensores.	**Tipo de Investigación** Analítico Experimental **Diseño de la Investigación** Diseño teórico Diseño experimental

ANEXO N°2: Programación de Node Sensor 1 en Arduino

```
/*
Nodo Sensor 1
Sensor: GPS
Potencia de Tx: 13dbm
Autor: Jean Pierre Tincopa
Marzo 2017
*/
#include <TinyGPS.h>//Librerias para GPS

#include <RHRouter.h> //Librerias de Radiohead (RFM22B)
#include <RH_RF22.h>
#include <SPI.h>      //Librerias para comunicacion SPI

#define CLIENT_ADDRESS 1 //Declaracion de elementos de la red
#define SERVER1_ADDRESS 2
#define SERVER2_ADDRESS 3

RH_RF22 driver;  //Asignacion de funcion dentro de la red
RHRouter manager(driver, SERVER1_ADDRESS); //Server1 --> Nodo Sensor 1

char stuff[30]; //Variable para almacenar  el frame total
char senString[30];  //tamaño referencial del frame

TinyGPS gps;

void setup()
{
 Serial.begin(9600);
 if (!manager.init()) //Inicializacion del transceiver
  Serial.println("init failed");
 manager.addRouteTo(CLIENT_ADDRESS, CLIENT_ADDRESS);  //Rutas de la
red
 manager.addRouteTo(SERVER2_ADDRESS, SERVER2_ADDRESS);

}

uint8_t buf[RH_ROUTER_MAX_MESSAGE_LEN]; //Tamaño del beacon

void loop()
{
 //Variables del sensor
 float dataSensor1;  //latitud
 float dataSensor2;  //longitud
```

```
  if (gps.encode(c)) // Nueva secuencia recibida
    newData = true;
  if (newData)
    { unsigned long age;
    gps.f_get_position(&dataSensor1, &dataSensor2, &age); //se almacena en
variables
    }

  uint8_t len = sizeof(buf); //almacenamos tamaño del beacon en "len"
  uint8_t from;
  if (manager.recvfromAck(buf, &len, &from)) //Verificamos llegada del beacon
  {
    Serial.print("Recibiendo beacon de CH: "); //Imprimir beacon
    Serial.print(from, HEX);
    Serial.print(": ");
    Serial.println((char*)buf);

    //Almacenamiento del dato del sensor
    dtostrf(dataSensor1,5,2,senString1); //Cambiamos de tipo de variable float a
string
    dtostrf(dataSensor2,5,2,senString2); //Cambiamos de tipo de variable float a
string
    senString=senString1+senString2 //Se concatenan ambos datos en una
variable

    //Declaracion del protocolo
    //NID:1 Identificación del Nodo sensor
    //N:1 Numero de sensores
    //SID:1 Identificación del sensor
    //SD: Valor del Sensor

    sprintf(stuff,"NID1N1SID1SD:%s",senString); //concatenamos string creado
con otro string y almacenamos en  stuff

    //Envio de Frame final
    Serial.println(stuff); //imprimimos string final
    Serial.println(sizeof(stuff)); //imprimimos tamaño del string
    Serial.println();

    if (manager.sendtoWait((uint8_t *)stuff, sizeof(stuff), from) !=
RH_ROUTER_ERROR_NONE) //Enviamos el string
      Serial.println("Envio fallido"); //imprimmir si falla
  }
}
```

ANEXO N°3: Programación de Node Sensor 2 en Arduino

```
/*
Nodo Sensor 2
Sensor: DHT22
Potencia de Tx: 13dbm
Autor: Jean Pierre Tincopa
Marzo 2017
*/
#include "DHT.h" //Librería DHT22
#define DHTPIN 2 //pines del sensor
#define DHTTYPE DHT22 //Se elige el sensor DHT22
DHT dht(DHTPIN, DHTTYPE); //variable a usar

#include <RHRouter.h> //Librerias de Radiohead (RFM22B)
#include <RH_RF22.h>
#include <SPI.h>      //Librerias para comunicacion SPI

#define CLIENT_ADDRESS 1 //Declaracion de elementos de la red
#define SERVER1_ADDRESS 2
#define SERVER2_ADDRESS 3

RH_RF22 driver;  //Asignacion de funcion dentro de la red
RHRouter manager(driver, SERVER1_ADDRESS); //Server1 --> Nodo Sensor 1

char stuff[30]; //Variable para almacenar  el frame total
char senString[30];  //tamaño referencial del frame

void setup()
{
 Serial.begin(9600);
 if (!manager.init()) //Inicializacion del transceiver
   Serial.println("init failed");
 manager.addRouteTo(CLIENT_ADDRESS, CLIENT_ADDRESS);  //Rutas de la
red
 manager.addRouteTo(SERVER2_ADDRESS, SERVER2_ADDRESS);
 dht.begin(); //Se inicializa el sensor
}

uint8_t buf[RH_ROUTER_MAX_MESSAGE_LEN]; //Tamaño del beacon

void loop()
{
 //Variables del sensor
 float dataSensor1;  //Temperatura
 float dataSensor2;  //Humedad Relativa
```

```
  dataSensor1 = dht.readTemperature(); //Se almacena los datos en las variables
 dataSensor2 = dht.readHumidity()

 uint8_t len = sizeof(buf); //almacenamos tamaño del beacon en "len"
 uint8_t from;
 if (manager.recvfromAck(buf, &len, &from)) //Verificamos llegada del beacon
 {
  Serial.print("Recibiendo beacon de CH: "); //Imprimir beacon
  Serial.print(from, HEX);
  Serial.print(": ");
  Serial.println((char*)buf);

  //Almacenamiento del dato del sensor
  dtostrf(dataSensor1,5,2,senString1); //Cambiamos de tipo de variable float a
string
  dtostrf(dataSensor2,5,2,senString2); //Cambiamos de tipo de variable float a
string

  //Declaracion del protocolo
  //NID:2 Identificación del Nodo sensor
  //N:2 Numero de sensores
  //SID: Identificación del sensor
  //SD: Valor del Sensor

  sprintf(stuff1,"NID2N2SID1SD:%s",senString1); //concatenamos string creado
con otro string y almacenamos en  stuff
  sprintf(stuff2,"NID2N2SID2SD:%s",senString2); //concatenamos string creado
con otro string y almacenamos en  stuff

  //Envio de Frame final
  Serial.println(stuff); //imprimimos string final
  Serial.println(sizeof(stuff)); //imprimimos tamaño del string
  Serial.println();

  Serial.println(stuff2); //imprimimos string final
  Serial.println(sizeof(stuff2)); //imprimimos tamaño del string
  Serial.println();

  if (manager.sendtoWait((uint8_t *)stuff, sizeof(stuff), from) !=
RH_ROUTER_ERROR_NONE) //Enviamos el string 1
    Serial.println("Envio fallido"); //imprimmir si falla
  if (manager.sendtoWait((uint8_t *)stuff2, sizeof(stuff2), from) !=
RH_ROUTER_ERROR_NONE) //Enviamos el string 2
    Serial.println("Envio fallido"); //imprimmir si falla
 }
}
```

ANEXO N°4: Programación de Node Sensor 3 en Arduino

```
/*
Nodo Sensor 3
Sensor: BMP180
Potencia de Tx: 13dbm
Autor: Jean Pierre Tincopa
Marzo 2017
*/
#include <SFE_BMP180.h> //Librerias para el sensor BMP180
#include <Wire.h>

SFE_BMP180 bmp180;

#include <RHRouter.h> //Librerias de Radiohead (RFM22B)
#include <RH_RF22.h>
#include <SPI.h>      //Librerias para comunicacion SPI

#define CLIENT_ADDRESS 1 //Declaracion de elementos de la red
#define SERVER1_ADDRESS 2
#define SERVER2_ADDRESS 3

RH_RF22 driver;  //Asignacion de funcion dentro de la red
RHRouter manager(driver, SERVER1_ADDRESS); //Server1 --> Nodo Sensor 1

char stuff[30]; //Variable para almacenar  el frame total
char senString[30];  //tamaño referencial del frame

void setup()
{
 Serial.begin(9600);
 if (!manager.init()) //Inicializacion del transceiver
   Serial.println("init failed");
 manager.addRouteTo(CLIENT_ADDRESS, CLIENT_ADDRESS);  //Rutas de la
red
 manager.addRouteTo(SERVER2_ADDRESS, SERVER2_ADDRESS);
 bmp180.begin(); //Se inicializa el sensor
}
uint8_t buf[RH_ROUTER_MAX_MESSAGE_LEN]; //Tamaño del beacon

void loop()
{
 //Variables del sensor
 float dataSensor1;  //Temperatura
 float dataSensor2;  //Presion Atmosferica
```

ANEXO N°5: Programación de Node Sensor 4 en Arduino

```
/*
Nodo Sensor 4
Sensor: FC-28
Potencia de Tx: 13dbm
Autor: Jean Pierre Tincopa
Marzo 2017
*/

#include <RHRouter.h> //Librerias de Radiohead (RFM22B)
#include <RH_RF22.h>
#include <SPI.h>      //Librerias para comunicacion SPI

#define CLIENT_ADDRESS 1 //Declaracion de elementos de la red
#define SERVER1_ADDRESS 2
#define SERVER2_ADDRESS 3

RH_RF22 driver;  //Asignacion de funcion dentro de la red
RHRouter manager(driver, SERVER1_ADDRESS); //Server1 --> Nodo Sensor 1

char stuff[30]; //Variable para almacenar  el frame total
char senString[30];  //tamaño referencial del frame
const int sensorPin = A0; //Pin para leer el sensor

void setup()
{
 Serial.begin(9600);
 if (!manager.init()) //Inicializacion del transceiver
  Serial.println("init failed");
 manager.addRouteTo(CLIENT_ADDRESS, CLIENT_ADDRESS);  //Rutas de la
red
 manager.addRouteTo(SERVER2_ADDRESS, SERVER2_ADDRESS);

}

uint8_t buf[RH_ROUTER_MAX_MESSAGE_LEN]; //Tamaño del beacon

void loop()
{
 //Variables del sensor
 float dataSensor;

 dataSensor= analogRead(sensorPin); //Lectura del sensor
 dataSensor=(dataSensor/1024)*100; //Convertimos el dato para que este entre 0
y 100%
```

```
  uint8_t len = sizeof(buf); //almacenamos tamaño del beacon en "len"
  uint8_t from;
  if (manager.recvfromAck(buf, &len, &from)) //Verificamos llegada del beacon
  {
   Serial.print("Recibiendo beacon de CH: "); //Imprimir beacon
   Serial.print(from, HEX);
   Serial.print(": ");
   Serial.println((char*)buf);

   //Almacenamiento del dato del sensor
   dtostrf(dataSensor,5,2,senString); //Cambiamos de tipo de variable float a
string

   //Declaracion del protocolo
   //NID:4 Identificación del Nodo sensor
   //N:1 Numero de sensores
   //SID:1 Identificación del sensor
   //SD: Valor del Sensor

   sprintf(stuff,"NID4N1SID1SD:%s",senString); //concatenamos string creado
con otro string y almacenamos en  stuff

   //Envio de Frame final
   Serial.println(stuff); //imprimimos string final
   Serial.println(sizeof(stuff)); //imprimimos tamaño del string
   Serial.println();

   if (manager.sendtoWait((uint8_t *)stuff, sizeof(stuff), from) !=
RH_ROUTER_ERROR_NONE) //Enviamos el string
    Serial.println("Envio fallido"); //imprimmir si falla
  }
}
```

```
BaseS=Gtk.Label()
BaseS.set_markup("<b>BASE STATION</b>")
fix.put(BaseS,600,120)
BaseS1=Gtk.Label("Hora:                    ")
fix.put(BaseS1,575,150)
self.bot1=Gtk.Button("-")
fix.put(self.bot1,610,145)

SensN1=Gtk.Label()
SensN1.set_markup("<b>Sensor Node 1</b>")
fix.put(SensN1,25,380)
SensN12=Gtk.Label("Dato:                    ")
fix.put(SensN12,0,410)
self.bot2=Gtk.Button(S1)
fix.put(self.bot2,35,405)

SensN3=Gtk.Label()
SensN3.set_markup("<b>Sensor Node 3</b>")
fix.put(SensN3,25,610)
SensN32=Gtk.Label("Dato:                    ")
fix.put(SensN32,0,640)
self.bot3=Gtk.Button(S5)
fix.put(self.bot3,35,635)

SensN2=Gtk.Label()
SensN2.set_markup("<b>Sensor Node 2</b>")
fix.put(SensN2,715,380)
SensN22=Gtk.Label("Dato:                    ")
fix.put(SensN22,690,410)
self.bot4=Gtk.Button(S2)
fix.put(self.bot4,725,405)

SensN4=Gtk.Label()
SensN4.set_markup("<b>Sensor Node 4</b>")
fix.put(SensN4,715,610)
SensN42=Gtk.Label("Dato:                    ")
fix.put(SensN42,690,640)
self.bot5=Gtk.Button(S7)
fix.put(self.bot5,725,635)

ClusH1=Gtk.Label()
ClusH1.set_markup("<b>Cluster Head Node 1</b>")
fix.put(ClusH1,350,400)

ClusH2=Gtk.Label()
ClusH2.set_markup("<b>Cluster Head Node 2</b>")
```

```
        try:
            self.ejecutar()
        except:
            print "No hay datos recibidos"



    def ejecutar(self):
        #Datos obtenidos de los sensores
        S1 = float(self.spin1.get_text())
        S2 = float(self.spin2.get_text())
        S3 = float(self.spin3.get_text())
        S4 = float(self.spin4.get_text())
        S5 = float(self.spin5.get_text())

        #Poniendo los resultados en formato de tablas
        cols = [Hora, S1, S2, S3, S4, S5] #Columnas
        txt = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]
#Cantidad de datos almacenados

        for num in range(5):                                    #iniciamos el
guardado de los datos
                        row = sheet1.row(num)
                        for index, col in enumerate(cols):
                                value = txt[index] + num
                                row.write(index, value)

                # Save the result
                book.save("sensores.xls")                       #Se guarda la
hoja final

print "\f"
win=MiVentana()
win.connect("delete-event", Gtk.main_quit)
win.show_all()
Gtk.main()
```

```
        tabla.attach(Gtk.Label("-"), 5, 6, 3, 4)
        tabla.attach(Gtk.Label("-"), 6, 7, 3, 4)

        tabla.attach(Gtk.Button("CLUSTER \n  HEAD\n     2"), 0, 1, 4, 5)
        tabla.attach(Gtk.Label("-"), 1, 2, 4, 5)
        tabla.attach(Gtk.Label("-"), 2, 3, 4, 5)
        tabla.attach(Gtk.Label("-"), 3, 4, 4, 5)
        tabla.attach(Gtk.Label("-"), 4, 5, 4, 5)
        tabla.attach(Gtk.Label("-"), 5, 6, 4, 5)
        tabla.attach(Gtk.Label("-"), 6, 7, 4, 5)

        tabla.attach(Gtk.Label("SENSOR \n  NODE\n     3"), 0, 1, 5, 6)
        tabla.attach(Gtk.Label("-"), 1, 2, 5, 6)
        tabla.attach(Gtk.Label("-"), 2, 3, 5, 6)
        tabla.attach(Gtk.Label(S3), 3, 4, 5, 6)  #Sensor 3
        tabla.attach(Gtk.Label("-"), 4, 5, 5, 6) #Sensor 3
        tabla.attach(Gtk.Label(S4), 5, 6, 5, 6)
        tabla.attach(Gtk.Label("-"), 6, 7, 6, 7)

        tabla.attach(Gtk.Label("SENSOR \n  NODE\n     4"), 0, 1, 6, 7)
        tabla.attach(Gtk.Label("-"), 1, 2, 6, 7)
        tabla.attach(Gtk.Label("-"), 2, 3, 6, 7)
        tabla.attach(Gtk.Label("-"), 3, 4, 6, 7)
        tabla.attach(Gtk.Label("-"), 4, 5, 6, 7)
        tabla.attach(Gtk.Label("-"), 5, 6, 6, 7)
        tabla.attach(Gtk.Label(S5), 6, 7, 6, 7)  #Sensor 4

        fix.put(tabla,900,230)

        #Empaquetar todo en la ventana
        vbox.pack_start(tabla,False,False,0)
        vbox.pack_end(fix,True,True,0)
        self.add(vbox)

    #Funciones para el switch de inicio
    def on_switch1_actived(self,switch1,gparam):
        if switch1.get_active():
            #state="on"
            self.spinner.start()
            self.boton5.set_label("Enviando Beacons")
        else:
            #state="off"
            self.spinner.stop()
            self.boton5.set_label("Conexion Cerrada")
```