Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Diseño del sistema de comunicación de una red inalámbrica de sensores para el estudio de la salud estructural en puentes

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Franz Vargas Acuña

Cartago, 8 de junio de 2016

INSTITUTO TECNOLÓGICO DE COSTA RICA

ESCUELA DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Defensa de Proyecto de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura
Instituto Tecnológico de Costa Rica

El Tribunal Evaluador aprueba la defensa del proyecto de graduación denominado Diseño del sistema de comunicación de una red inalámbrica de sensores para el estudio de la salud estructural en puentes, realizado por el señor Franz Vargas Acuña y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

Ing. Adolfo Chaves Jiménez
Profesor lector

Ing. Julio Stradi Granados
Profesor lector

Ing. Francisco Navarro Henriquez
Profesor asesor

Cartago, 8 de junio de 2016

Declaración de autenticidad

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Cartago, 8 de junio de 2018

Céd: 3-0459-0760

Resumen

A medida que la tecnología avanza y los distintos dispositivos se conectan a internet, se vuelve cada vez más importante tener acceso a la información de una manera eficiente y sencilla. Esto ha hecho que las inalámbricas de sensores cobren una importancia notable en los últimos años. Una red de este tipo permite monitorear y guardar datos sobre condiciones físicas alguna estructura y tener acceso a ellas en tiempo real.

De ahí nace la idea del proyecto e-Bridge de la Escuela de Ingeniería en Construcción del Instituto Tecnológico de Costa Rica. La idea inicial fue tener un sistema automatizado para poder monitorear aspectos estructurales, como desplazamiento y vibración, de los puentes en Costa Rica. Actualmente el Centro de Investigación en Vivienda y Construcción (CIVCO) cuenta con un equipo capaz de tomar las mediciones. Sin embargo, los costos asociados al mantenimiento y calibración de los equipos, así como el tiempo que toma realizar la calibración presentan un problema grande. Esto último se debe a que la calibración de los sensores solamente se puede realizar fuera del país, y el tiempo de viaje (2 meses en total) más el hecho de que se deba realizar anualmente, dan como resultado, dispositivos cuya vida útil anual se ve reducida.

Es una red inalámbrica de sensores que permita monitorear en tiempo real la salud estructural de los puentes en Costa Rica. En 2011 inicia el proyecto en el Centro de Investigación en Vivienda y Construcción (CIVCO) en conjunto con la escuela de Ingeniería Electrónica para desarrollar una red confiable, expandible y de bajo costo.

Este proyecto muestra el proceso de diseño e implementación del sistema de comunicación que se utilizará en esta red de sensores; la cual se realizará en una topología de estrella con un nodo central y dos nodos remotos, los cuales fueron implementados usando tarjetas Raspberry Pi 3. Mientras que la conexión entre los tres se hizo mediante módulos de radio nRF24L01. El sistema es capaz de detectar la cantidad de nodos activos y solicitar los datos de nada uno de estos nodos para luego almacenar los datos en un archivo local.

Palabras Clave: Red, sensores, puentes, comunicación.

Abstract

As technology advances and the various devices connect to the internet, the need to access information easily and efficiently becomes more important. Therefore, wireless sensor networks have become more important in the last years. These networks allow to monitor and store data on physical conditions of structures and have real-time access to it.

From this, comes the idea for the e-Bridge project from on the Construction Engineering department at Instituto Tecnológico de Costa Rica. The initial plan was to have an automated system to monitor structural aspects, like vibration and displacement, of the bridges in Costa Rica. Today, the Centro de Investigación en Vivienda y Construcción (CIVCO) has equipment capable of taken such measurements. But, the cost associated with maintenance and calibration of this equipment, and the time it takes to calibrated them represents a big problem. This is due to the fact that the sensors can only be calibrates outside of the country and the shipping time (2 months total) plus the fact that these calibrations must be done annually, makes the total useable time on the equipment very low.

It is a wireless sensor network that allows the real-time monitoring of the structural health of bridges in Costa Rica. In 2011 this project starts at the Centro de Investigación en Vivienda y Construcción (CIVCO) in conjunction with the Electronics Engineering Department to develop a reliable, expandable and low-cost network

This project shows the design and build of the communication system used in this sensor network which was developed using a star topology with one central node and two remote nodes which were implemented using 3 Raspberry Pi 3 board. Meanwhile, the connection between them it's been handled with the nRF24L01 radio module. The system can detect how many actives nodes there are and request data from each one of them, that later will be stored in a local file.

Keywords: network, sensors, bridges, communication.

Índice General

Índice de figuras

Índice de tablas

1. Introducción

1.1 Antecedentes

Desde el 2010 y hasta la actualidad, la Escuela de Ingeniería en Construcción del Instituto Tecnológico de Costa Rica ha desarrollo el proyecto e-Bridge a cargo de la profesora Gianinna Ortiz. El mismo consiste en la predicción de fallas en puentes a través del monitoreo de variables en estas estructuras (Instituto Tecnológico de Costa Rica, 2017).

El proyecto se divide en 3 etapas. E-Bridge 1.0, desarrollada entre el 2011 y el 2012, tuvo como objetivo la predicción remota de fallas en puentes y logró mejorar la metodología de inspección y medición, así como las variables que influyen en el comportamiento estructural de los puentes. La segunda etapa (e-Bridge 2.0), que inició en 2013, tiene como objetivo desarrollar un prototipo de sistema integrado de información para consultas estratégicas sobre el desempeño de los puentes, basados en datos obtenidos por medio de sistemas de información geográfica, medición cuantitativa del desempeño, modelos de confiabilidad estructural e información técnica de la estructura (Instituto Tecnológico de Costa Rica, 2016). Como etapa final, se desea tener un sistema experto integrando todas las variables que permitan la toma de decisiones en el área de los puentes.

Actualmente el Centro de Investigación en Vivienda y Construcción (CIVCO) de la Escuela de Ingeniería en Construcción cuenta con equipo para medir ciertas variables de salud estructural tales como vibraciones y deformación. Esto se realiza mediante un dispositivo datalogger de Campbell Scientific modelo CR3000. El sistema consiste en un módulo central que se encarga de tomar y registrar todas las mediciones. Los sensores se conectan directamente al él, lo que limita enorme el área de operación del dispositivo ya que los cables de los sensores deben tener una longitud especifica dadas las recomendaciones del fabricante.

Figura 1: Datalogger CR3000 de Campbell Scientific

Recientemente se realizó la compra del sistema de testeo estructural STS4 de la empresa BDI. Este sistema es compatible con los mismos sensores que el CR3000 y tiene la capacidad de comunicar inalámbricamente el nodo con los nodos secundarios, eliminado los problemas que presentaba el *datalogger* de Campbell Scientific. El STS4 consiste en un nodo central, o estación base (Figura 2), con capacidad de comunicación inalámbrica o alambrada hacia los nodos secundarios (Figura 3). Cada uno de estos nodos secundarios pueden acomodar hasta 4 sensores distintos. La comunicación inalámbrica se lleva a cabo mediante conexión Wifi, mientras que la alambrada utiliza ethernet. La alimentación se puede hacer mediante una batería de ion de litio, así como un adaptador de corriente. Además, cuenta con certificaciones a prueba de agua y polvo, para poder ser operados bajo cualquier condición.

Sin embargo, todos estos equipos sufren del mismo problema. Los sensores que utilizan para medir variables estructurales (como la vibración o desplazamiento) requieren de una calibración anual. Esta solo se puede llevar a cabo en los Estados Unidos, por lo que se deben enviar allá todos los daños. De acuerdo con el encargado en el CIVCO, el costo asociado a la calibración y el transporte redondo asciende a los $2000 por sensor. Además, el traslado desde Costa Rica hasta Estados Unidos y viceversa toma alrededor de unos 3 meses. En el caso de los sensores de vibración, estos solo pueden tomar medidas en un eje a la vez.

Figura 2: STS4 estación base

Figura 3: STS4 nodo secundario

Por último, los dispositivos son parte de un sistema cerrado. Y, aunque se utilice un tipo de conexión estandarizada (DIN-9), solamente se puede conectar sensores por este medio. En caso de querer conectar otro tipo de sensores, no es posible hacerlo físicamente y los sistemas no serán capaces de interpretar la entrada de datos.

Dado que la meta final del proyecto es poder dejar los sistemas de monitoreo instalados permanentemente en los puentes, y dado el alto costo que llegan a tener estos equipos, no es viable utilizar estos equipos en esas circunstancias.

Debido a esto, en el 2017 se inició el desarrollo de una red inalámbrica de sensores, en conjunto con la Escuela de Ingeniería Electrónica. El objetivo de realizar el proyecto fue de poder tener una red abierta, que tuviera la capacidad de incorporar otros sensores si se deseara medir otros parámetros, que fuera de bajo costo y que tuviera la capacidad de almacenar los datos en una locación centralizada y remota y que pudieran ser visualizados de distintas maneras.

Para el desarrollo de la primera etapa, que consistía en realizar mediciones de vibración en los tres ejes, se utilizaron tarjetas Waspmote de Libelium para los nodos remotos. Estas cuentan con un pequeño microcontrolador, un RTC y la posibilidad de conectar un módulo XBee. Como nodo central, se utilizó una tarjeta Raspberry Pi. Para comunicar los nodos, se utilizaron módulos XBee mediante el protocolo IEEE 802.15.4. Sin embargo, el sistema presentó una serie de fallas y problemas que dificultaron su desarrollo.

Figura 4: Tarjeta Waspmote de Libelium

Figura 5: Tarjeta Raspberry Pi 3

Una de las fallas más importantes fue la pérdida del enlace entre los módulos. De acuerdo con el encargado del proyecto, eso se debía a que la cantidad de datos que se debían enviar por segundo excedía la capacidad del canal.

Por eso, con el presente proyecto se pretende continuar la investigación, mediante el mejoramiento y desarrollo del sistema de comunicación que utilizarán los nodos.

1.2 Análisis del Problema

La red de sensores presenta problemas en la comunicación entre los nodos. Una vez que se establece el enlace, al cabo de unos pocos segundos se pierde. De acuerdo con el encargado actual del proyecto, esto se debe a la baja taza de bits de transferencias los nodos utilizados (waspmote). Estos tienen una taza de transferencia de 38400 bps. Por otro lado, cada dato enviado por los xbee es de 64 bits y tienen una velocidad de transferencia de 250 kbps. Y es necesario poder realizar mediciones, por lo menos, 30 veces por segundo.

Por lo tanto, el proyecto e-Bridge no cuenta con una red de sensores capaz de tomar medidas a la velocidad de muestreo necesaria y almacenar la información en una base de datos local.

1.3 Solución Implementada

La solución propuesta consiste en una red tipo estrella con un nodo central y varios nodos remotos. Los nodos remotos se comunicarán inalámbricamente por radio frecuencia con el nodo central y le enviarán los datos para que sean recopilados y almacenados. Adicionalmente, cada nodo guardará una copia de los datos localmente para redundancia.

La topología tipo estrella tiene la ventaja de ser más sencilla de implementar. Sin embargo, la comunicación se realiza únicamente entre un nodo remoto y el nodo central, si por alguna razón la comunicación es deficiente, los datos recibidos por el nodo central pueden ser incorrectos. En la Figura 6 se puede observar una red estrella con una línea de comunicación desconectada del nodo central. Cuando esto sucede, el nodo remoto queda completamente aislado de la red y no se recibirá información. Por esta razón es que cada nodo remoto almacenará, adicionalmente, los datos de los sensores conectados localmente.

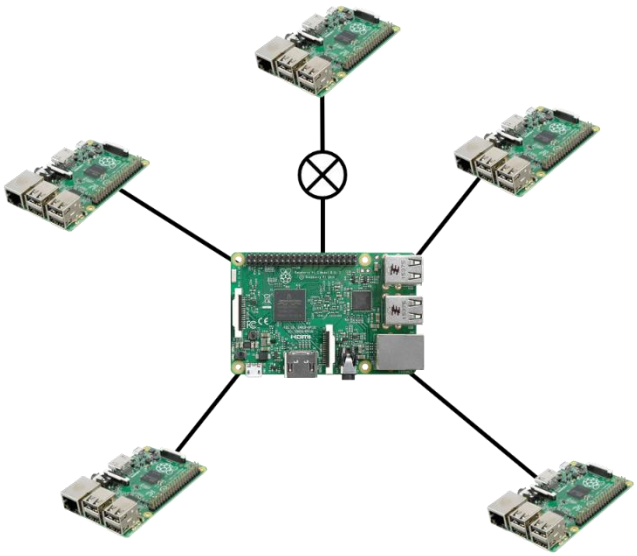


Figura 6: Falla en topología de estrella

Todos los nodos se implementarán usando tarjetas Raspberry Pi ya que permiten conectar una gran cantidad de sensores. Además, al tener un sistema operativo completo, pueden conectarse con pantallas táctiles que sirven para que el usuario interactúe con el sistema. Por otro lado, aceptan medios de almacenamiento masivo USB para guardar los datos en un medio extraíble adicionalmente de en la memoria local. Por último, utilizan una tarjeta microSD como medio principal de almacenamiento el cuál se utilizó para guardar la base de datos.

Para que los nodos se puedan comunicar se utilizaron módulos de radiofrecuencia (RF) nRF24l01 y el protocolo Enhanced Shockburst. Estos permiten un gran rango de distancia además de una taza de bits suficiente para los requerimientos.

2. Meta y Objetivos

2.1 Meta

Desarrollar una red de sensores, que permita obtener los datos para el análisis de vibración, deformación y desplazamiento en puentes y que además tenga facilidad de extensión en un futuro para otro tipo de sensores.

- Indicador: La red almacena datos de las variables sin pérdidas durante la transmisión. Y es posible observar la salud estructural del puente en tiempo real.

2.2 Objetivo General

- Desarrollar el sistema de comunicación para una red inalámbrica de sensores capaz de medir la de deformación en puentes.
    - Indicador: el nodo central recibe datos sin errores, en los tiempos requeridos y los almacena en una base de datos local.

2.3 Objetivos Específicos

1. Verificar los datos recibidos en el nodo central desde el nodo remoto.
    - Indicador: Se tiene un set de datos de deformación para cada intervalo de tiempo definido y estos pasan la prueba de verificación mediante algoritmos de verificación.
2. Almacenar los datos recibidos en una base de datos local y/o en una memoria extraíble.
    - Indicador: La base de datos local tiene mediciones de deformación para cada intervalo de tiempo.
3. Implementar los sensores en varios nodos independientes.
    - Los nodos son capaces de enviar datos cuando el nodo central lo solicita.
4. Establecer un sistema escalable
    - La red puede detectar cuando se introduce un nodo nuevo en el sistema y auto configurarse.

3. Marco Teórico

3.1 Comunicaciones Inalámbricas

Un componente crucial de las redes de sensores es la comunicación entre los nodos. Es una de las partes de una red que consume más energía y si no se dimensiona correctamente, la red podría no trabajar eficientemente o no trabajar del todo. A continuación, se analizan algunos aspectos importantes en el estudio de las comunicaciones.

3.1.1 Propagación de las ondas de radio

Pérdidas por espacio libre (FSL)

Corresponde a las pérdidas de potencia por efectos del aire, sin objetos que puedan causar interferencias o reflexiones. Es un factor muy importante que tomar en cuenta ya que determina la potencia necesaria de transmisión y la sensibilidad del receptor. Es una función de la frecuencia y la distancia y está dada por (Freeman, 2007):

$$FSL_{dB} = 20 \log\left(\frac{4\pi d}{\lambda}\right)$$

Si se utiliza la frecuencia en mega Hertz (MHz) y la distancia en kilómetros (km) se obtiene, entonces:

$$FSL_{dB} = 32.45 + 20 \log D_{km} + 20 \log F_{MHz}$$

Propagación y atenuación

Otro factor importante que tomar en cuenta es el efecto que tienen la atmósfera en la propagación de las ondas. Pequeñas variaciones en el índice de refractividad causan que las ondas no puedan trasladarse en una línea recta infinitamente, sino que se curvara. Las ondas electromagnéticas viajan a distintas velocidades a través de medios con distintas constantes dieléctricas. Factores como la presencia de moléculas de gas y agua en el ambiente aumentan esta constante en comparación al espacio libre, lo que ralentiza la velocidad de propagación de las ondas en ese medio. Al aumentar la altitud con respecto al nivel del mar, la constante dieléctrica disminuye. Así que la parte superior de la onda se propaga con mayor velocidad que la parte inferior causando deflexión del rayo y que ésta, siga la curvatura de la tierra. A partir de esto se puede definir el índice de refractividad como la relación entre la curvatura de la trayectoria de la onda y la curvatura de la Tierra. Se le conoce como factor K (Freeman, 2007).

$$K \approx \frac{r}{r_o}$$

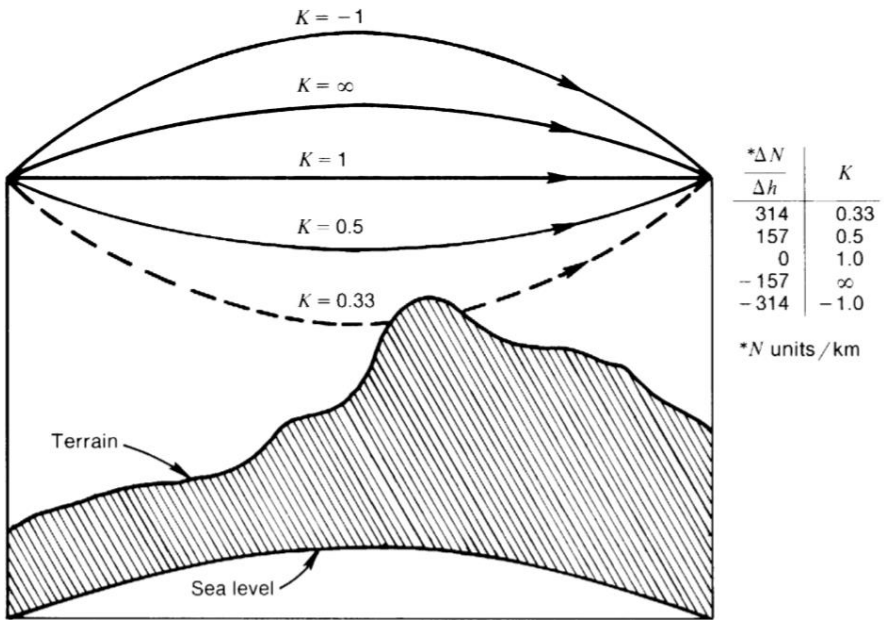


Figura 7: Curvatura del rayo para distintos valores de K (Freeman, 2007)

3.1.2 Radio Enlaces

Cuando se calculan radio enlaces se deben tomar en cuenta varios aspectos para asegurar que la señal que se envía se pueda recibir sin errores al otro lado del canal. A parte de las pérdidas por espacio libre, se debe tener en cuenta lo que se conoce como zona de Fresnel. Aunque las ondas de radio viajan en una línea recta, la energía que emiten se ensancha desde que salen de la antena del emisor hasta que llegan a la antena, formando un ovalo imaginario que corresponde a la zona de Fresnel. Si algún objeto se encuentra dentro de la Zona de Fresnel, parte de la energía de la señal se reflejada y no llegará hasta la antena del receptor, disminuyendo la energía recibida (Young, 2002).

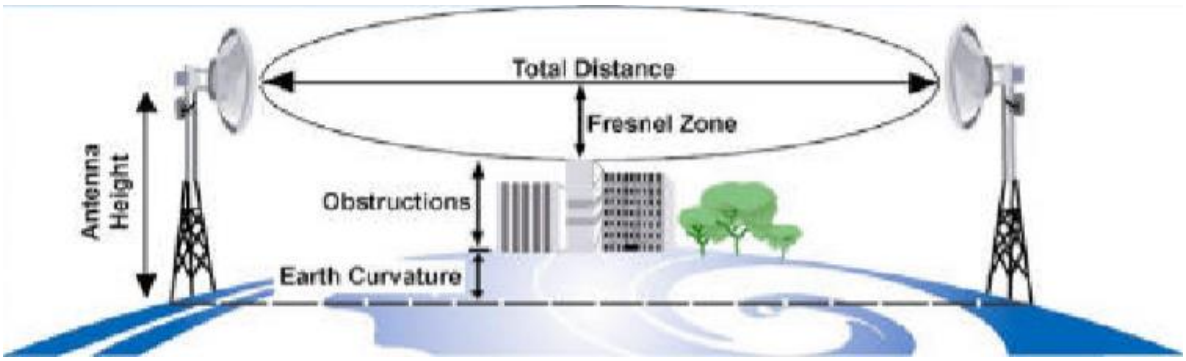


Figura 8: Descripción de la Zona de Fresnel (Young, 2002)

La forma de la Zona de Fresnel depende de la frecuencia de la señal, entre mayor sea la frecuencia, más pequeña será la zona. Y dependiendo de la aplicación que se tenga, es necesaria una mayor o menor Zona de Fresnel.

Otros aspectos importantes que tomar en cuenta son los siguientes, los cuales corresponden directamente al módulo de radio que se esté utilizando.

- Nivel de señal en RX: nivel detectado en la antena de recepción.
- Sensitividad de RX: valor mínimo de señal que es detectable.
- Ganancia de la Antena: capacidad de la antena de enfocar energía en cierta dirección
- Potencia de TX: potencia que coloca el transmisor en la antena TX
- Potencia radiada efectiva (Effective Isotropic radiated power: EIRP): potencia transmitida por la antena.

Una vez que se identifican todos estos parámetros, es necesario realizar un cálculo de margen de operación (SOM). El SOM se define como la diferencia entre el nivel requerido en RX y el recibido. Existen diversos criterios con respecto a cuál sería un SOM adecuado, sin embargo, según (Young, 2002), estos márgenes oscilan entre 10 dB y 20 dB. El problema de bajar más es que se corre el riesgo de que el enlace se pierda si las condiciones atmosféricas cambian. Para calcular el SOM, simplemente se realiza una suma así:

$$Potencia\ de\ salida\ de\ la\ antena = Potencia\ de\ TX - Pérdidas\ en\ el\ cable + EIRP\ de\ la\ antena$$

$$Nivel\ de\ señal\ RX = Potencia\ de\ salida\ de\ la\ antena - FSL + EIRP\ antena\ receptora - Pérdidas\ en\ el\ cable$$

$$SOM = Nivel\ de\ señal\ RX\ (dBm) - Sensitividad\ (dBm)$$

3.1.3 Modulación

Un sistema de comunicación debe ser capaz de transmitir señales que contienen información a través de un canal de comunicación que separa un transmisor de un receptor. Las señales que llevan información se conocen como señales en banda base (Haykin, 2002)

La modulación, entonces, se puede definir como la modificación de algún parámetro de una onda portadora de acuerdo con la señal del mensaje. Los valores que se alteran pueden ser la amplitud, frecuencia o fase de la onda, independientemente del tipo de comunicación (analógica o digital). La onda resulta se conoce como onda modulada.

Modulación Analógica

En el ámbito analógico existen tres tipos distintos de modulación, AM (amplitud modulada), FM (Frecuencia modulada) y PM (fase modulada).

AM

Cuando se modula la amplitud, se varia este parámetro en la señal portadora de acuerdo con la señal de información. Cuando se realiza este proceso, se puede suprimir la componente en frecuencia de la señal portadora o enviar junto a la señal.

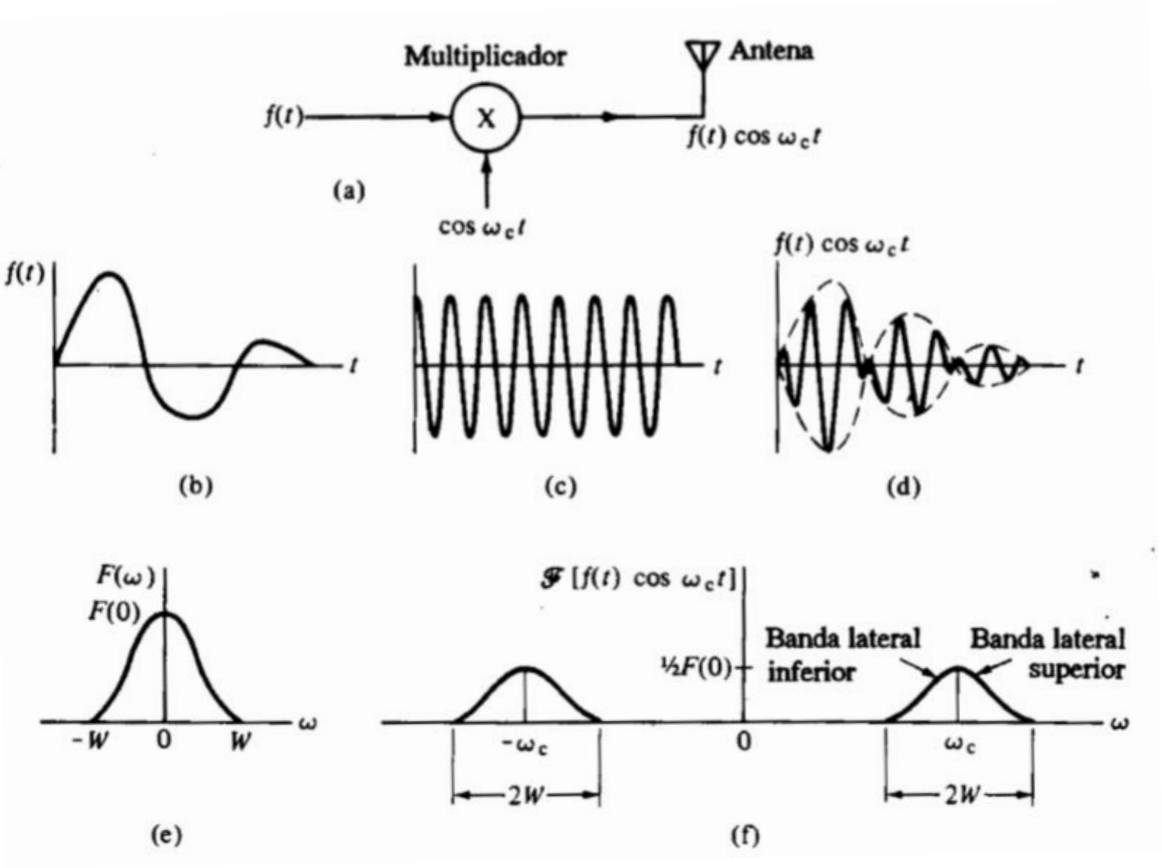


Figura 9: Sistema de transmisión de amplitud modulada de portadora suprimida (Stremler, 1993)

Una señal senoidal se puede expresar de la siguiente manera:

$$\phi(t) = a(t)\cos[\omega_c t + \gamma(t)]$$

Donde $[\omega_c t + \gamma(t)] = \theta(t)$ y corresponde al ángulo de la señal, el término $a(t)$ se le conoce como envolvente de la señal $\phi(t)$ y ω_c es la frecuencia portadora; $\gamma(t)$ es la modulación de fase de la señal. Si se asume que $a(t)$ y $\gamma(t)$ varían lentamente comparados con $\omega_c t$, que el término de fase $\gamma(t)$ es constante (o cero) en el caso de la modulación de amplitud y $a(t)$ se hace proporcional a la señal dada $f(t)$, se obtiene entonces (Stremler, 1993):

$$\phi(t) = f(t)\cos\omega_c t$$

Que corresponde a la señal modulada. Si se transforma al ámbito de la frecuencia se puede observar que la modulación de amplitud traslada el espectro de frecuencia de una señal, pero deja inalterada su forma.

$$\Phi(\omega) = \frac{1}{2}F(\omega + \omega_c) + \frac{1}{2}F(\omega - \omega_c)$$

PM y FM

Una señal modulada en PM o FM tiene una variación en la fase o frecuencia, respectivamente, de acuerdo con la señal de información que se desea enviar. Sin embargo, comprender el concepto de una señal con fase o frecuencia variable no es una tarea trivial. Para comprender esto, (Stremler, 1993) sugiere que es útil ver la representación fasorial de una señal senoidal con su respectiva relación matemática:

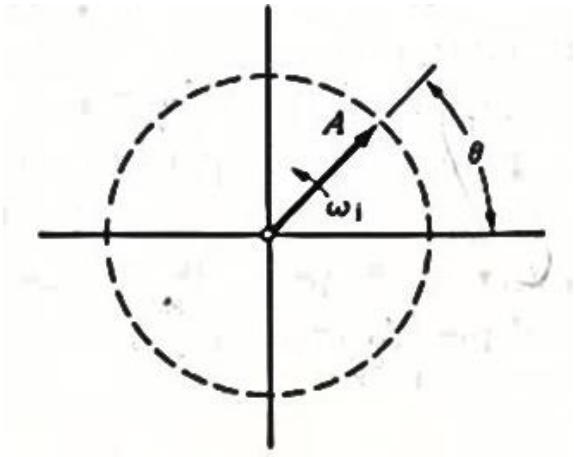


Figura 10: Representación fasorial general (Stremler, 1993)

$$\theta(t) = \int_0^t \omega_i(\tau)d\tau + \theta_0$$

Esto, entonces, ayuda a derivar y comprender el concepto de frecuencia instantánea que se obtiene al derivar la relación entre la fase y la frecuencia, lo que da como resultado:

$$\omega_i(t) = \frac{d\theta}{dt}$$

Una vez que se tiene el concepto de frecuencia instantánea, continua (Stremler, 1993), se pueden comprender como se da el proceso de modulación de fase y de frecuencia. Cuando la fase es la que varía con el tiempo, y es proporcional a la señal de entrada $f(t)$ se describe así:

$$\theta(t) = \omega_c t + k_p f(t) + \theta_0$$

Donde ω_c, k_p y θ_0 son constante. Se le conoce como modulación de fase (PM) y cuenta con una frecuencia instantánea igual a:

$$\omega_i = \frac{d\theta}{dt} = \omega_c + k_p \frac{df(t)}{dt}$$

La segunda opción sería que la frecuencia instantánea sea el parámetro proporcional a la entrada, lo que daría una señal modulada en frecuencia (FM) que se representa así:

$$\theta(t) = \int_0^t \omega_i(\tau) d\tau = \omega_c t + \int_0^t k_i f(\tau) d\tau + \theta_0$$

Cuya frecuencia instantánea seria:

$$\omega_i = \omega_c + k_f f(t)$$

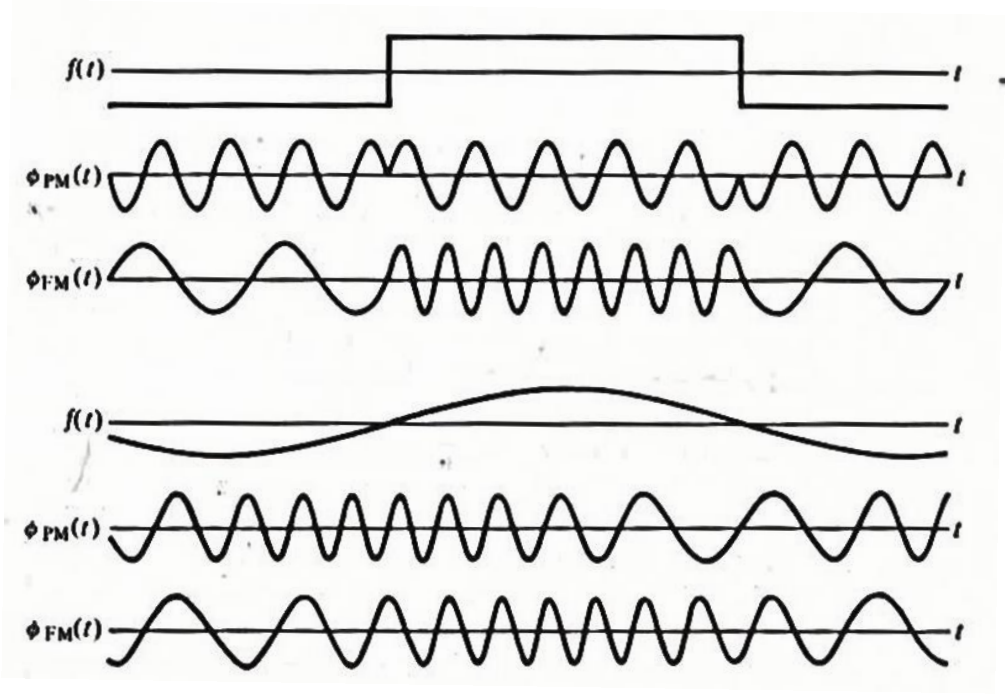


Figura 11: Ejemplos de modulación de frecuencia y de fase

Modulación Digital

En el caso de los sistemas digitales, es conveniente modular la señal portadora con la corriente de datos antes de la transmisión. Existen tres tipos generales de modulación digital ASK (*amplitude-shift keying*), FSK (*frequency-shift keying*) y PSK (*phase-shift keying*) (Stremler, 1993). Un ejemplo gráfico se puede observar en la Figura 13: Formas de onda ilustrativas para las tres formas básicas de información binaria para transmisión de señales . Se puede observar que las señales PSK y FSK tienen una envolvente constante, a diferencia de la señal ASK. Además, es más sencillos distinguir entre las señales PSK y FSK en comparación con sus equivalentes analógicos (FM y PM) (Stremler, 1993).

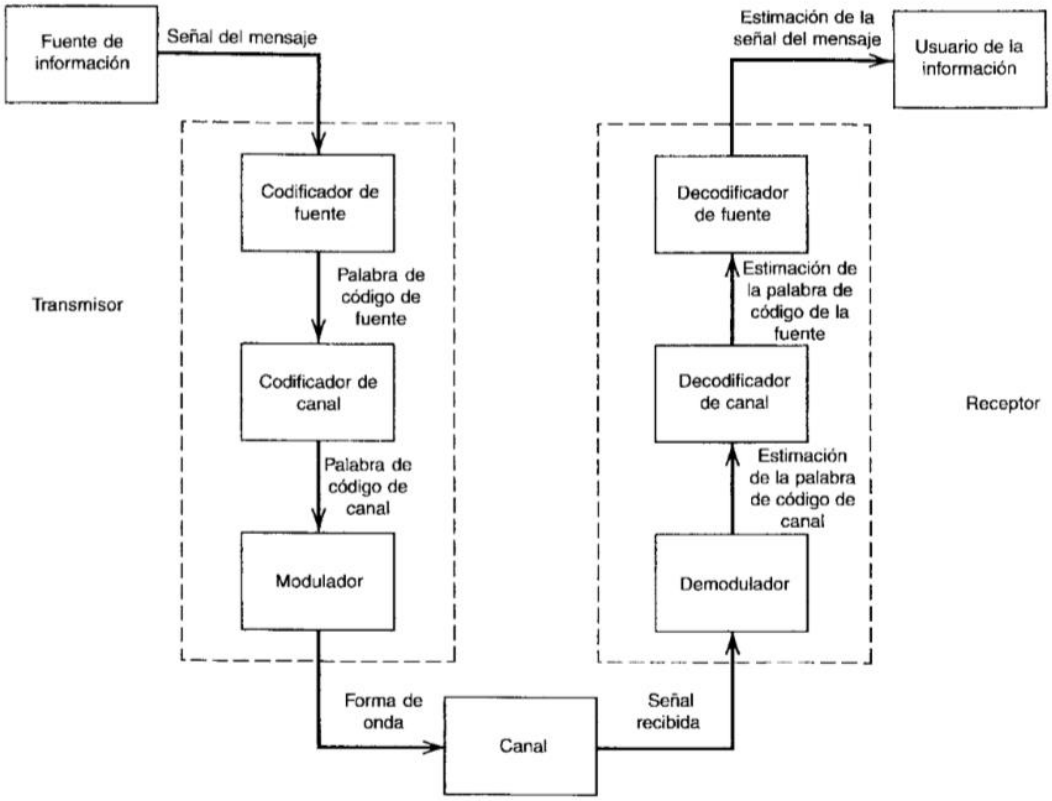


Figura 12: Diagrama de bloques de un sistema de comunicación digital (Haykin, 2002)

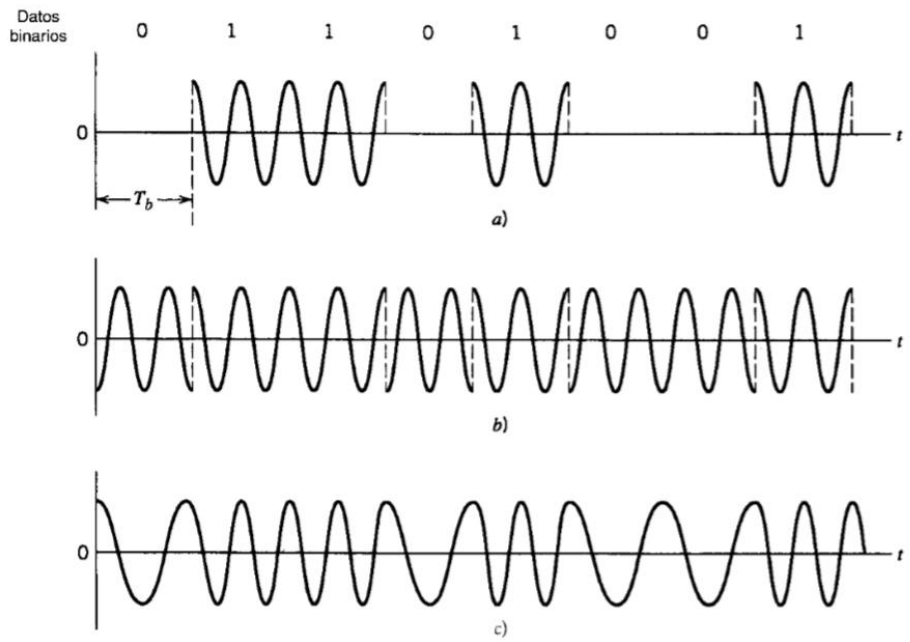


Figura 13: Formas de onda ilustrativas para las tres formas básicas de información binaria para transmisión de señales (Haykin, 2002).

ASK

En la conmutación de amplitud, la amplitud de la señal portadora varía entre dos o más valores. Para el caso binario, seria entre dos, ON y OFF. Lo que resulta en una señal de pulsos, llamados marcas, que representan unos y espacios que representan ceros, como se aprecia en la Figura 14.

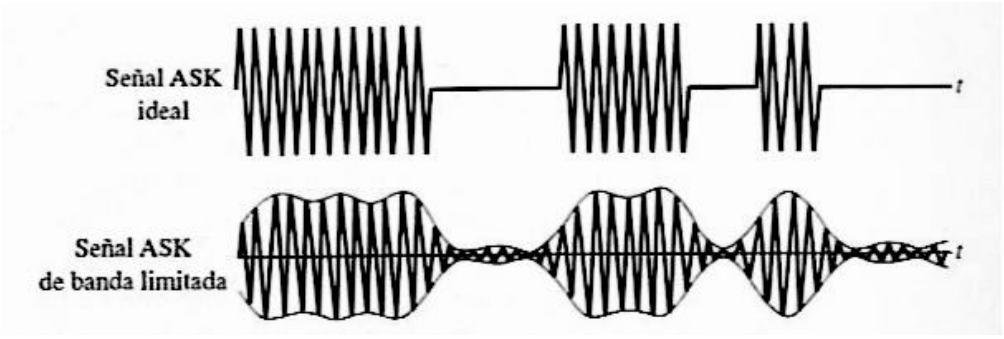


Figura 14: Señales binarias ASK (Stremler, 1993)

La representación matemáticamente para un uno binario está dada por:

$$\phi(t) = \begin{cases} A \sin \omega_o t & 0 < t \leq T \\ 0 & en\ otro\ caso \end{cases}$$

FSK

Traducida literalmente como conmutación de frecuencia. La variación se da en la frecuencia instantánea y conmuta entre dos o más valores en respuesta al código inicial. La Figura 15 muestra una señal FSK ideal y se puede observar que una señal FSK consiste en la suma de dos señales ASK con frecuencias portadoras distintas, las cuales se representan así:

$$\phi_1(t) = \begin{cases} A \sin m\omega_o t & 0 < t \leq T \\ 0 & en\ otro\ caso \end{cases}$$

$$\phi_2(t) = \begin{cases} A \sin m\omega_o t & 0 < t \leq T \\ 0 & en\ otro\ caso \end{cases}$$

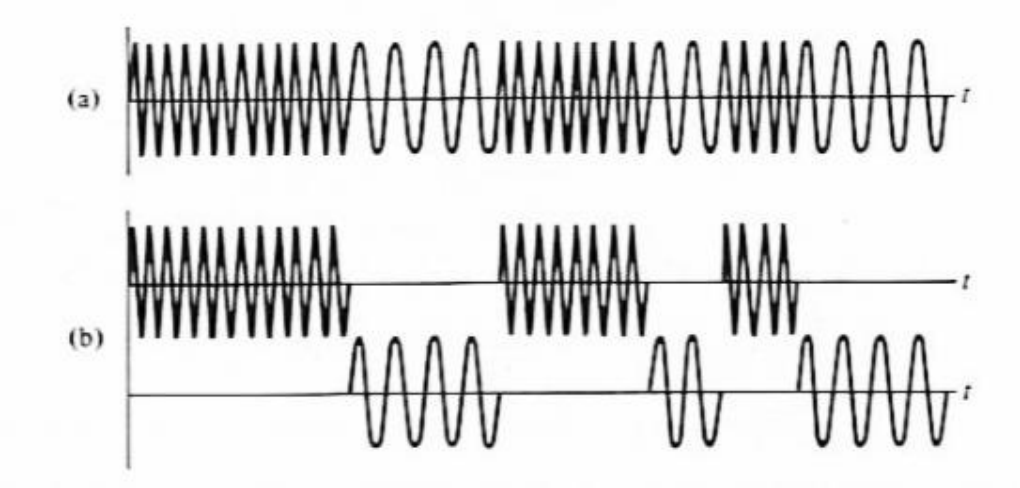


Figura 15: (a) Señal FSK ideal y (b) su descomposición en dos señales ASK (Stremler, 1993)

3.2 Redes Inalámbricas de Sensores

Una red de sensores inalámbricos (WSN) es una red inalámbrica que consiste en dispositivos distribuidos espaciados autónomos utilizando sensores para monitorear condiciones físicas o ambientales. Un sistema WSN incorpora un gateway que provee conectividad inalámbrica de regreso al mundo de cables y nodos distribuidos (National Instruments, 2009). Dependiendo de la manera en la que se comuniques y estructuren sus nodos, estos tendrán una función diferente. Esta manera de acomodarlos se denomina topología. Existen varias topologías que se utilizan en las redes de sensores, cada una con sus ventajas y desventajas. Algunas de estas son: bus, estrella, anillo y mesh.

3.2.1 Topologías de red

Topología de bus

En este tipo de topología, todos los nodos están conectados a un bus central así que todos los mensajes que envía un nodo en particular son recibidos por el resto de los nodos. Sin embargo, solo el nodo para el cual el mensaje iba dirigido lo acepta y procesa. El resto de los nodos descartan el mensaje.

Esto genera mucho ruido en el canal de comunicación, se desperdicia ancho de banda y puede generar colisión de paquetes y, por ende, perdida de información.

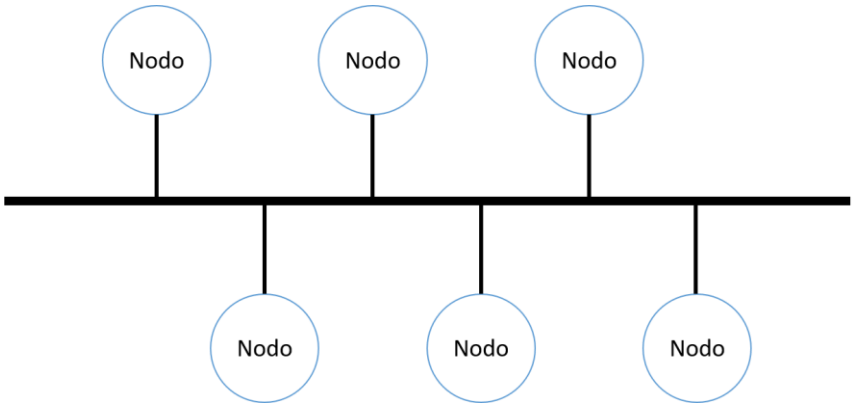


Figura 16: Topología tipo bus

Topología de estrella

En esta topología todos los nodos sensores están conectados a un nodo central independientemente de los demás. Esta normalmente se usa en aplicaciones de red en hogares, todos los equipos se conectan al router, ya sea por cable o inalámbricamente, y dicha comunicación es independiente de los demás equipos. Si un paquete necesita ir de un equipo a otro, este debe viajar a través del router.

En el caso de las redes inalámbricas de sensores sucede de manera similar. Cada nodo sensor (conocidos como dispositivos finales) se comunica con el nodo central (conocido como coordinador) mediante un canal bidireccional. Si un dispositivo final necesita comunicarse con otro, debe hacerlo mediante el coordinador únicamente, no pueden hacerlo directamente (Faludi, 2010).

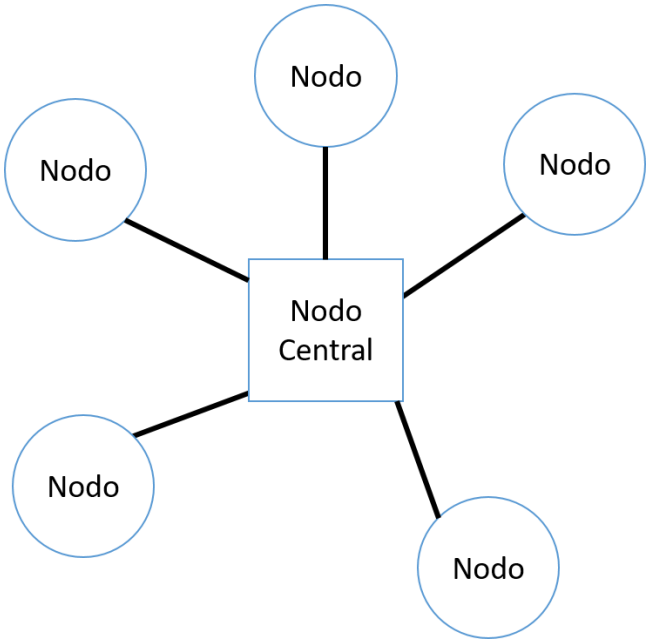


Figura 17: Topología tipo estrella

Al tener el coordinador comunicación directa con todos los nodos y ser el único medio de comunicación entre dispositivos finales, el poder de procesamiento para manejar los mensajes y tomar decisiones debe ser mayor que el de los demás nodos. Además, si un nodo sensor pierde comunicación con el nodo central, solamente ese nodo quedaría aislado de la red. Sin embargo, si el coordinador falla, toda la red se destruye (Cook & Das, 2004).

Topología de anillo

Los nodos, en este caso, se encuentran conectados a otros dos nodos para formar un anillo (de ahí su nombre). Los datos viajan de nodo a nodo en el sentido horario o antihorario.

No tiene dispositivos finales, a diferencia de la topología de estrella, todos los nodos son considerados repetidores. Si alguno de los nodos falla en su función de repetidor, toda la red podría caerse y perder toda la comunicación. Sin embargo, en ciertas aplicaciones, los dispositivos tienen la capacidad de detectar un fallo en el nodo y sacarlo del anillo para que no se pierda en enlace (Castelli, 2002).

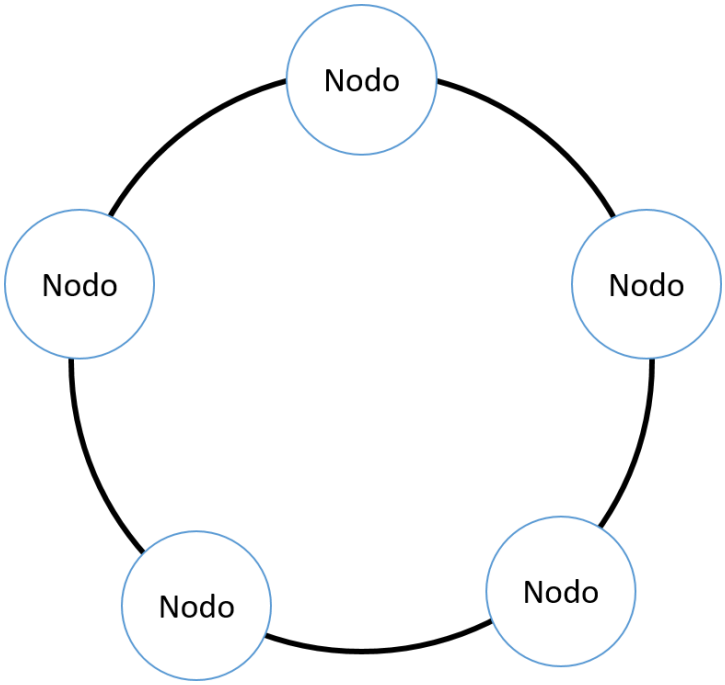


Figura 18: Topología tipo anillo

Topología Mesh

Es la más utilizada en redes de sensores ya que permite que los nodos sensores se comuniquen entre sí. Si un nodo desea comunicarse con otro nodo que se encuentra fuera de su rango, puede utilizar un nodo intermedio para retransmitir el mensaje. Una ventaja de esta topología es que, si un nodo falla, lo mensajes se pueden retransmitir a través de los que aun funcionan y se encuentran dentro del rango. Sin embargo, el consumo energético es mayor debido a las transmisiones redundantes (Selmic, Phoha, & Serwadda, 2016).

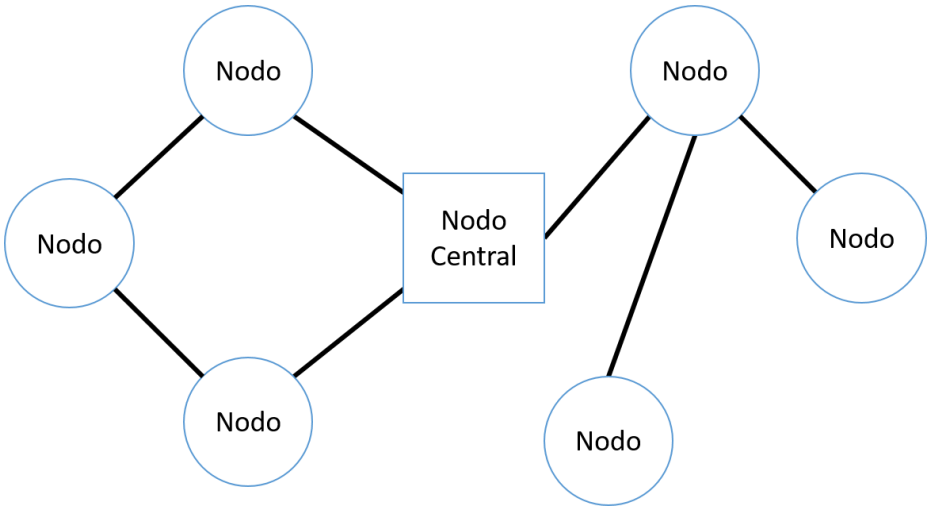


Figura 19: Topología tipo mesh

Hay una variación de esta topología que se conoce como estrella-mesh y combina los atributos de ambas. Se puede plantear como una red similar a la mesh, en la cual los dispositivos más lejanos son nodos simples de bajo consumo (similares a los que usa la red de estrella). Mientras tanto, los nodos más cercanos al centro son de alto consumo (y en muchas ocasiones suelen estar conectados a la red eléctrica. Esto se debe a que estos dispositivos deben poder reenviar mensajes entre una gran cantidad de nodos y funcionar como puertas de enlace (Selmic, Phoha, & Serwadda, 2016).

3.3 Protocolos de Comunicación.

3.3.1 SPI

Es el bus de interfaz serial periférico (Serial Peripheral Interface) y se realiza mediante 4 líneas de comunicación MOSI, MISO, SCLK y SS. Permite comunicación *dúplex*, sincrónica y serial entre un microcontrolador y un dispositivo periférico y se maneja bajo el esquema maestro/esclavo, es decir, un dispositivo maestro solicita datos al dispositivo esclavo. Este protocolo de comunicación permite velocidades de transferencia de 0 – 10 Mbps.

El dispositivo maestro envía los datos mediante el pin MOSI (Máster Output – Slave Input) hasta el esclavo, Mientras que el esclavo responde y envía los datos mediante el pin MISO (Máster Input – Slave Output). Ya que es una conexión síncrona, existe una línea dedicada a enviar la señal de reloj del maestro hasta el esclavo, esta se llama SCK. El último pin (SS) se conoce como Slave Select y permite habilitar el dispositivo con el que se desea establecer la comunicación. Por lo general, este pin se encuentra negado en la mayoría de los dispositivos por lo que es necesario enviar un cero lógico para habilitar el dispositivo.

Éste protocolo de comunicación permite conectar una gran cantidad de dispositivos al mismo bus y habilitar solamente el que se desea. En la Figura 20 se puede observar un esquema de cómo se conectarían los dispositivos. Cuando el máster quiere hablar con un dispositivo en particular solamente se envía un cero lógico al pin SS del dispositivo y se inicia el proceso de comunicación.

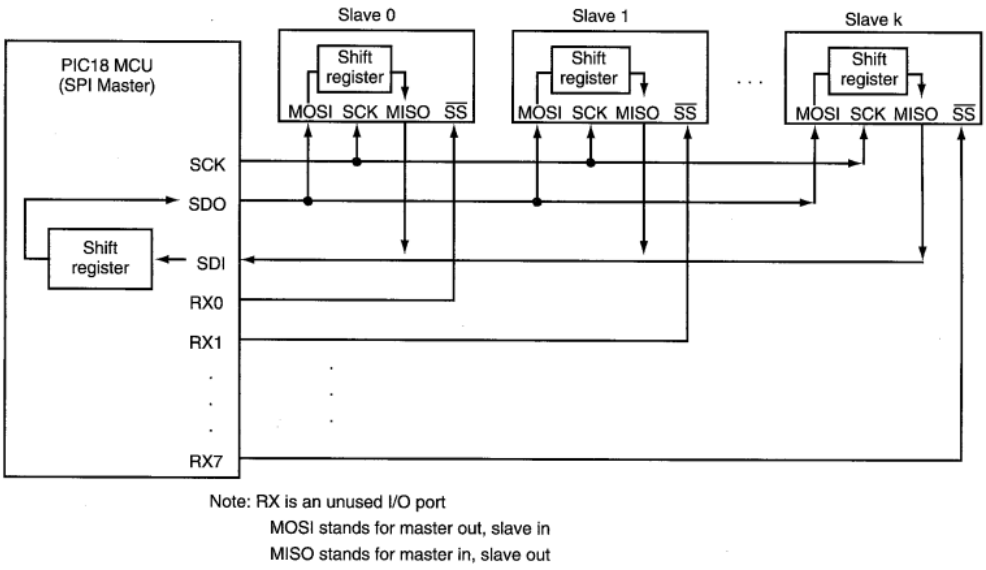
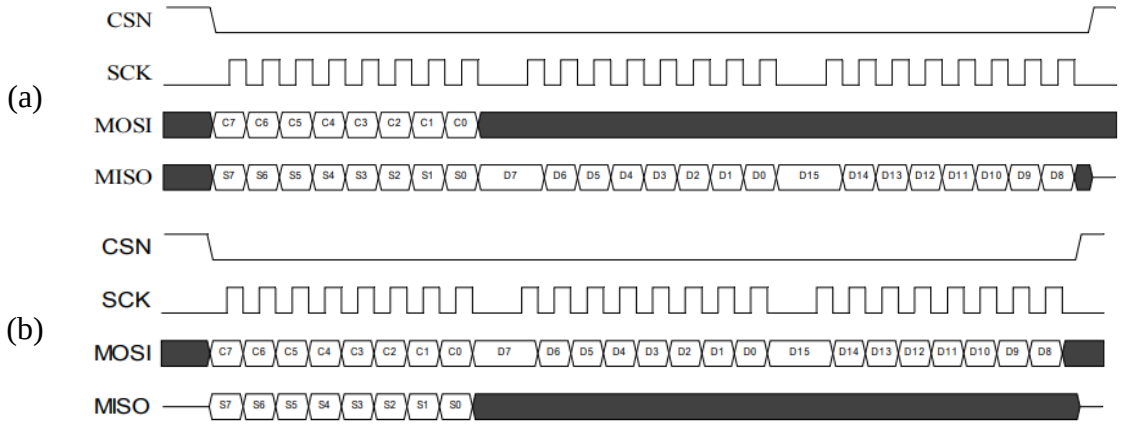


Figura 20: Esquema de conexión SPI con un máster y múltiples esclavos (Huang, 2005).

Existe otra forma de comunicar utilizando únicamente 3 cables, uno para el reloj y otro para habilitar el dispositivo. La diferencia se encuentra en la línea de datos, mientras que en la implementación habitual se usa una línea dedicada para la salida de datos y otra para la entrada de datos, en esta modalidad se tiene una única línea de datos bidireccional. Sin embargo, no todos los módulos no permiten esta modalidad, y por esto no es tan común como la de 4 líneas.

El proceso de comunicación SPI se lleva acabo de la siguiente manera. Primero, es necesario habilitar la línea SS con un cero lógico. Una vez que se habilita el esclavo, se debe iniciar la transmisión del reloj compartido. Si se está realizando una operación de lectura entonces se envía el comando de lectura y se reciben los datos solicitados, si se realiza una operación de escritura, sucede de manera contraria.

Figura 21: (a) Operación de lectura SPI y (b) de escritura SPI (Nordic Semiconductor, 2008)



3.3.2 Enhanced Shockburst

De acuerdo con la hoja de datos de los módulos NRF24L01+, se define Enhanced Shockburst como una capa de enlace de datos (capa dos del modelo OSI) basada en paquetes. Cuenta con auto ensamblaje de los paquetes y con acuso de recibo automático (ACK) y retransmisión de paquetes (Nordic Semiconductor, 2008).

Aparte de esto, el protocolo de comunicación cuenta con cargas dinámicas, es decir, existe un máximo tamaño de las cargas (32 bytes) pero el protocolo es capaz de ajustar el tamaño del paquete al tamaño de la carga que se esté empaquetando. La transacción de paquetes se maneja de la siguiente manera:

1. Se transmite un paquete de datos desde el transmisor primario (PTX) al receptor primario (PRX), luego el protocolo cambia automática al PTX a modo receptor para esperar el mensaje ACK.
2. Si el paquete se recibe en el PRX, el protocolo arma automáticamente un paquete ACK que envía al PTX justo antes de volver al modo de recepción.

3. Si el PTX no recibe el paquete ACK desde el PRX, Enhanced Shockburst retransmite el paquete original después un tiempo de retardo programable y, una vez más, se cambia al PTX a modo recepción para esperar el paquete de ACK.

La retransmisión puede ajustarse, se puede configurar la cantidad máxima de retransmisiones y el tiempo de retardo que el módulo de radio debe esperar para retransmitir. Todo el manejo de paquete descrito anteriormente se lleva acabo a nivel de hardware, localmente, en el módulo de radio sin que el microprocesador se involucre (Nordic Semiconductor, 2008)

Los paquetes que utiliza este protocolo siguen la estructura de la Figura 22. El preámbulo es una secuencia de bits que se alterna entre 1 y 0 para poder sincronizar el demodulador del receptor. En caso de que la dirección inicie con un 1, entonces el protocolo se encarga de que el ultimo bit del preámbulo sea un cero. De igual manera, si el primero bit de la dirección es un 0 entonces el ultimo bit del preámbulo será un 1. Esto se realiza para que existan la cantidad suficiente de transmisiones para estabilizar al receptor. La dirección corresponde a la del receptor y se utiliza para asegurar que los paquetes sean detectados y recibidos correctamente.

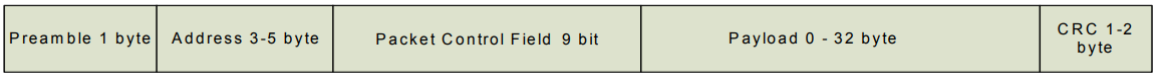


Figura 22: Un paquete Enhanced Shockburst con carga (0-32 bytes) (Nordic Semiconductor, 2008)

La tercera sección, *Packet Control Field (PID)* o campo de control de paquetes, contiene información referente a las características de la transmisión y se puede observar el Figura 23. Este indica el tamaño que tendrá la carga de acuerdo con la siguiente codificación:

- 000000 = 0 byte (cuando se envían paquetes ACK vacíos)
- 100000 = 32 bytes
- 100001 = no es relevante el tamaño

El espacio de PID se utiliza, también, para identificar si el paquete es nuevo o retransmitido. Este permite duplicidad de datos ya que si se recibe un paquete que tenga el mismo ID que el anterior y no es una retransmisión entonces se desecha el paquete. Esto se utiliza en conjunto con la sección CRC para poder determinar la naturaleza del paquete. Y, por último, se tiene la bandera de NO_ACK, si se encuentra un 1 en este espacio, el PRX no enviara el paquete ACK automático (el paso 2 presentado anteriormente).

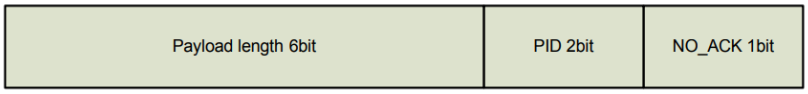


Figura 23: Espacio de control de paquetes

El ultimo espacio es el del CRC (cyclic redudancy check) y corresponde a un valor resultante de una operación matemática y utiliza como entradas la dirección, el PID y la carga. Puede usar CRC de 1 o 2 bytes y sigue las siguientes expresiones.

$$Para\ 1\ byte\colon\ X^8 + X^2 + X + 1, \qquad con\ valor\ inicial\ 0xFF$$

$$Para\ 2\ bytes\colon\ X^{16} + X^{12} + X^5 + 1, \qquad con\ valor\ inicial\ 0xFFFF$$

Para realizar la validación de los paquetes los módulos están constantemente revisando las direcciones validas, cuando se encuentra una el protocolo inicia el proceso de validación. Cuando se utiliza carga dinámica, se captura la carga dependiendo de la información que se encuentra en el espacio de control de paquetes. Luego se realiza el CRC, si este es válido entonces se revisa el PID y se compara con el PID del último paquete, si son diferentes entonces el paquete se etiqueta como nuevo. Si el PID coincide, se revisa el CRC. Si este último es el mismo que el del mensaje anterior, entonces se etiqueta como paquete repetido y se descarta.

Otra característica que tiene el protocolo es el modo MultiCeiver, que permite al PRX recibir datos y comunicarse paralelamente con 6 PTX mediante 6 medios llamados *data pipes.* El direccionamiento se realiza con valores de 3 a 5 bytes (dependiendo de lo que se configure). Y se puede observar un esquema en la Figura 25, el *pipe* 0 tiene una dirección única mientras que el resto, comparte los primeros 4 bytes y variar el último.

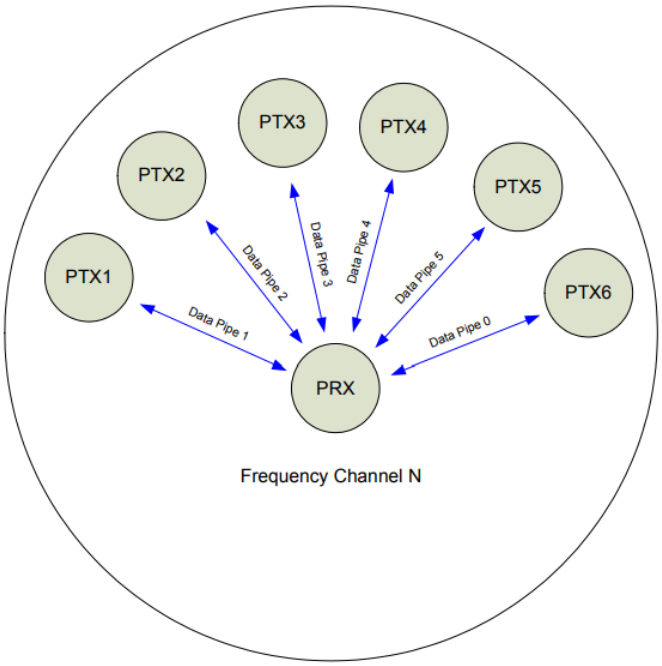


Figura 24: PRX usando MultiCeiver (Nordic Semiconductor, 2008)

	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Data pipe 0 (RX_ADDR_P0)	0xE7	0xD3	0xF0	0x35	0x77
Data pipe 1 (RX_ADDR_P1)	0xC2	0xC2	0xC2	0xC2	**0xC2**
Data pipe 2 (RX_ADDR_P2)	0xC2	0xC2	0xC2	0xC2	**0xC3**
Data pipe 3 (RX_ADDR_P3)	0xC2	0xC2	0xC2	0xC2	**0xC4**
Data pipe 4 (RX_ADDR_P4)	0xC2	0xC2	0xC2	0xC2	**0xC5**
Data pipe 5 (RX_ADDR_P5)	0xC2	0xC2	0xC2	0xC2	**0xC6**

Figura 25: Direccionamiento de los pipes 0-5 (Nordic Semiconductor, 2008)

3.4 Dispositivos embebidos.

3.4.1 Raspberry Pi 3

Raspberry pi es una pequeña computadora, del tamaño de una tarjeta de crédito, con casi todas las capacidades de un equipo personal. A diferencia de un microcontrolador, Raspberry Pi necesita de un sistema operativo para trabajar. Actualmente soporta varias distribuciones de Linux e inclusive, una versión especial de Windows 10 para dispositivos de Internet de las Cosas (IoT) (Raspberry Pi Foundation, s.f.). El sistema operativo recomendado por el fabricante es Raspbian, una versión personalizada para esta tarjeta y basada en Debian. Es una plataforma Open Source, o de código abierto, utilizada en centros educativos (desde escuela hasta universidades) que permite desarrollar sistema embebidos y aprender distintos lenguajes de programación. Su versatilidad permite que sea utilizada en sistema sencillos (como proyectos académicos) hasta proyectos formales y de gran complejidad. Actualmente, la versión más reciente del hardware es la 3 versión B. En la **Tabla 1** se pueden observar las especificaciones técnicas del dispositivo.

Tabla 1: *Especificaciones Técnicas de Raspberry Pi 3*

ESPECIFICACIÓN	
PROCESADOR	Broadcom BCM2837
NÚCLEOS	4
FRECUENCIA	1.2 GHz
MEMORÍA RAM	1 GB
ALMACENAMIENTO	microSD
ALIMENTACIÓN	5V / 2.5A vía Micro USB

Además, la tarjeta cuenta con 4 puertos USB 2.0 para conectar una gran variedad de periféricos, un puerto Ethernet que soporta velocidades de transferencia de hasta 1 Gbps. Para la salida de audio y video, cuenta con un puerto HDMI y un puerto de 3.5mm con salida de audio estéreo y video compuesto. Adicionalmente cuenta con interfaces CSI y DSI para conectar cámaras y pantallas táctiles directamente. La conectividad WiFi y Bluetooth LE se obtienen gracias al integrado BCM43438 que se encuentra en la tarjeta. Por otro lado, el procesador, cuenta con buses I2C, SPI y UART, así como pines de entrada/salida de uso general (GPIO: General Purpose Input/Output) (Raspberry Pi Foundation, s.f.).

3.4.2 nRF24L01+

Los módulos nRF24L01 son transceptores de radio que funcionan a una frecuencia de 2.4 GHz. Son manufacturados por la empresa Nordic Semiconductor. Pueden transmitir datos inalámbricamente a velocidades de 250 kbps, 1 Mbps y 2 Mbps y utilizan modulación GFSK. Permite comunicar dos o más dispositivos hasta un máximo de 1km dependiendo de la antena y potencia que se escoja. Existen distintas presentaciones del módulo, algunas traen la antena integrada en el PCB como una pista mientras que otros cuentan con un conector SMA que permite instalar una antena dipolo externa. Al contar con una antena externa, el rango máximo aumenta y permite posicionar los dispositivos más distanciados y no tener que tener unidades repetidoras para hacer llegar la señal desde el punto A al B.

(a) (b)

Figura 26 (a) nRF24L01+ con antena en PCB y (b) con antena SMA externa

Además de poder realizar una comunicación de punto a punto, estos módulos permiten hacer conexiones de punto a multipunto mediante el protocolo Enhanced ShockBurst. Éste permite que un módulo se pueda comunicar con un máximo de 6 nodos independientes mediante distintos canales llamados *pipes*. Cada uno de estos es completamente independiente del otro.

Tabla 2: *Características de TX/RX del módulo nRF24L01 (Nordic Semiconductor, 2008)*

CARACTERÍSTICA	VALOR
FRECUENCIA DE TX	2.4 GHz
POTENCIA DE TX	0, -6, -12, -18 dBm
SENSIBILIDAD	-82 dBm @ 2 Mbps -85 dBm @ 1 Mbps -96 dBm @ 250 kbps
VELOCIDAD DE INTERFAZ SPI	Máx. 10 Mbps

4. Procedimiento Metodológico

4.1 Obtención y análisis de la información

El proyecto fue desarrollado en el Centro de Investigación en Vivienda y Construcción (CIVCO), adscrito la Escuela de Ingeniería en Construcción del Instituto Tecnológico de Costa Rica y forma parte del proyecto e-Bridge. Una vez que se tenían claros los requerimientos por parte del equipo, se inició el proceso de recolección de información y consulta de material bibliográfico referente a las necesidades específicas que tienen las redes inalámbricas de sensores en cuanto al proceso de comunicación entre nodos.

Además, se estudiaron casos en donde se aplicaron las distintas tecnologías de comunicación que se encontraron, sus aspectos positivos, así como los posibles puntos de falla en dichos sistemas. Luego, se compararon las similitudes de esos casos al sistema requerido por el CIVCO. Estos casos de estudio sirvieron como base para el desarrollo del sistema.

Una vez establecido esto, se realizó un análisis comparativo entre las tecnologías, sus alcances y limitaciones, sus facilidades de implementación y adaptación a los cambios tecnológicos en materia de telecomunicaciones y su capacidad de crecer y agregar más elementos al sistema.

4.2 Alternativas de Solución

Dentro de los requerimientos que se tienen para el proyecto e-Bridge, una de las más importantes es el alcance de la comunicación. Era necesario que los módulos pudieran comunicarse a una distancia de por lo menos 100 metros con una meta máxima ideal de 1 kilómetro. Además, la velocidad de transferencia debía ser mayor a la que tienen los módulos actuales, que es de 250 kbps. Otro requisito importante es el costo y la disponibilidad de las tecnologías en el país. Debido al proceso de compra para el Instituto Tecnológico de Costa Rica, se dificulta realizar compras al exterior a través de tiendas como Amazon.com o Sparkfun.com. Por otro lado, el tener que pedir componentes al exterior anula uno de los objetivos del CIVCO en desarrollar la red desde cero.

Las 4 tecnologías de comunicación más prominentes son WiFi, ZigBee/XBee, Bluetooth y RF. A continuación, se encuentra un breve análisis acerca de las 4 opciones.

4.2.1 Bluetooth

Bluetooth se encuentra actualmente en la versión 5. Esta, agrega mayor velocidad y rango al estándar 4.2 que se utilizaba anteriormente. Distintas pruebas realizadas con esta versión han alcanzado hasta 350 metros de distancia entre los módulos. Inclusive hay módulos comerciales en el mercado que pueden llegar hasta 500 metros. Adicionalmente, la versión 5 aumenta la velocidad de transmisión de 1 Mbps a 2 Mbps (Woolley, 2017). Adicionalmente, el estándar cuenta con un algoritmo de detección y corrección de errores.

Uno de los módulos disponibles en el mercado local es el HC-05. Se basa en la versión 2.0+EDR de bluetooth la cual permite una velocidad de transferencia mayor que la versión 5 (hasta 3 Mbps). Además, cuenta con una sensibilidad de recepción que puede llegar hasta -80dBm, una potencia de salida de -6dBm a +4dB y una antena integrada en el PCB como una pista. Además, utiliza una frecuencia de 2.4GHz. De acuerdo con el fabricante estos módulos tienen un rango aproximado de unos 10m.

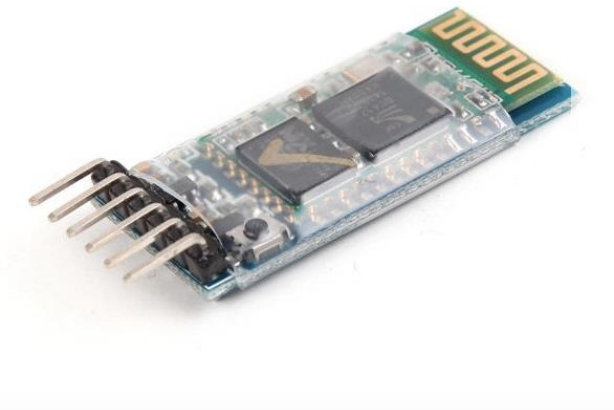


Figura 27: Módulo HC-05 Bluetooth

Por otro lado, la versión de bluetooth que utilizan estos módulos no es compatible con las tecnologías de redes de sensores que las versiones actuales si tienen. Estos son diseñados para realizar conexiones de punto a punto. Podría desarrollarse una red, pero implicaría que todo el manejo de paquetes y administración debería manejarse desde el nodo central, no se realizaría nada en hardware.

Analizando los requerimientos tenemos lo siguiente:

- ✓ Costo: $8.95
- ✓ Disponibilidad en el país: sí
- ✓ Velocidad de transferencia mayor a 250kbps: si, es de hasta 3Mbps
- ✗ Rango es por lo menos 100m: no, es alrededor de 10.

Por otro lado, se podría utilizar un adaptador bluetooth USB o la unidad integrada en Raspberry Pi. Pero sucede algo similar, el rango no llega a ser suficiente ya que los módulos del mercado (adaptadores USB) suelen ser de pequeño tamaño. Están diseñados para conectar periféricos (como mouse o teclado) a las computadoras por lo que la potencia de salida es baja y no alcanza el rango necesario.

4.2.2 ZigBee/Xbee

Una tecnología muy utilizada en aplicaciones de este tipo es ZigBee/Xbee. Estos módulos utilizan el protocolo IEE 802.15.4 y se comunican a una frecuencia de 2.4 GHz. Estos son los que se utilizaron anteriormente en el proyecto. Sin embargo, los módulos xbee soportan una variedad de protocolos dependiendo del modelo. Los que se encontraban en el proyecto anteriormente eran los Serie 1. Estos módulos tienen una velocidad de transferencia máxima de 250 kbps. Adicionalmente, el alcance máximo con línea de vista es de 100 metros (Digi International, n.d.). Existe una variación de mayor potencia para estos módulos que logra tener un alcance mucho mayor, hasta 1.6km con línea de vista. Sin embargo, ambas variaciones están limitadas a 250 kbps.

Una segunda generación de xbee (S2C) aumenta el rango a 1.2km para el módulo normal y 3.2km para el de mayor potencia. E inclusive existe una variación (S3B) que trabaja a 900 MHz lo cual disminuye la interferencia y aumenta el rango gasta 6km. Pero, una vez más, estos tienen un tope de 250 kbps y 200 kbps respectivamente.

Estos módulos se pueden conectar al bus I2C directamente en los pines GPIO o mediante USB utilizando un adaptador como el de la Figura 28: Adaptador USB Explorer, lo que aumenta el costo del sistema. Por otro lado, los módulos que se pueden conseguir en el mercado local no cuentan con una antena integrada, solamente tiene un conector tipo U.FL. Por lo que es necesario conseguir, no solo la antena si no también el convertidor de U. FL a SMA.

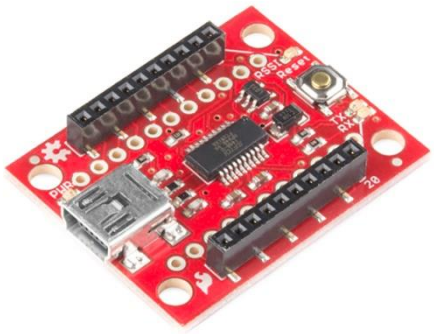


Figura 28: Adaptador USB Explorer

Tabla 3: *Especificaciones técnicas para varios tipos de módulos xbee*

MÓDULO	FREC. DE TX	P. DE TX	SENSIBILIDAD	VELOCIDAD DE TX	VELOCIDAD INTERFAZ
XBEE S1	2.4 GHz	0 dBm	-92 dBm	250kbps	1200 bps – 250 kbps
XBEE S1 PRO	2.4 GHz	+18 dBm	-100 dBm	250kbps	1200 bps – 250 kbps
XBEE S2C	2.4 GHz	+5 dBm -8 dBm	-100 dBm -102 dBm	250kbps	-
XBEE S2C PRO	2.4 GHz	+18 dBm	-101 dBm	250kbps	-
XBEE S3B	900 MHz	+24 dBm	-101 dBm -110 dBm	10 kbps – 200 kbps	9600 baud – 230400 baud

El mayor problema que presenta esta tecnología es la velocidad máxima a la que pueden enviar los datos. Si bien es cierto en las etapas tempranas del desarrollo del sistema la cantidad de sensores y variables es limitada, uno de los objetivos del proyecto es tener una red escalable, que permita manejar más sensores, más nodos y más variables. Al tener una velocidad limitada a 250 kbps, se corre el riesgo de que no den abasto los nodos para manejar todo el tráfico que se requiere.

4.2.3 WiFi

Por otro, está la tecnología Wifi, basada en el protocolo IEEE 802.11, se presenta como una opción tentadora. La velocidad de transferencia teórica puede alcanzar hasta 150 Mbps y un alcance que, utilizando las antenas correctas, puede alcanzar grandes distancias.

El mercado local ofrece los módulos ESP8266 los cuales cuentan con una antena en el PCB. La ventaja que presenta la tecnología Wifi es la gran cantidad de revisiones del protocolo y que sean compatibles entre sí. Las revisiones más recientes (801.11n) han logrado aumentar la velocidad de transmisión sin que se el consumo energético se vea afectado. La tecnología utiliza el stack TCP/IP el cual cuenta una gran cantidad de protocolos para manejar los datos, enrutarlos y evitar colisiones.

El alcance máximo anda cerca de los 300 metros lo cual lo sitúa como un candidato atractivo para desarrollar la red. Sin embargo, el consumo energético sigue siendo alto. Aunque los avances en el protocolo han logrado disminuir en consumo de energía considerablemente, el mínimo consumo en la transmisión es de 135mA por cada paquete que se envíe, mientras que por cada paquete que se recibe se consumen 62 mA. Esto generaría un consumo de 197 mA por cada comunicación exitosa. Se necesitan 30 mediciones por segundo y que cada prueba debe durar alrededor de 40 min a 1 h. Si se realiza el cálculo matemático, la capacidad energética sería muy grande. Adicional a eso, se debe aún alimentar los nodos y los sensores que así lo requieran.

Por lo tanto, la tecnología Wifi no es energéticamente viable para el proyecto.

4.2.4 nRF24L01+

Por último, se consideró el módulo nRF24L01 el cuál trabaja a 2.4 GHz y tiene una velocidad de transferencia de hasta 2 Mbps y logra un alcance de hasta 1km. Además, pueden trabajar con una comunicación de punto a punto (en caso de solo tener dos nodos) o de punto a multipunto (en caso de tener 3 o más nodos). Permite que un módulo se comunique con hasta 6 otros módulos independientemente. Y, por último, tienen un costo bajo (USD 14,95) y el kit incluye la antena y todo lo necesario para realizar el enlace.

En la sección 3.4.2 se explicó con detalle las características de este módulo. Sin lugar a duda, una de las características más llamativas que tiene esta tecnología es su relativo bajo costo, y su largo alcance. Tiene la capacidad de trabajar a 250, 1 o 2 Mbps por lo que la red no se va a ver limitada por la velocidad de transmisión. Además, puede manejar tareas de retransmisión, identificación y verificación de paquetes directamente en el hardware, lo que libera al procesador del nodo central para realizar otras tareas criticas como la de ordenamiento y almacenaje de los datos. Por otro lado, cuando la red crezca y pueda ser instalada indefinidamente en las estructuras, puede manejar tareas de comunicación hacia la base de datos central. Dadas estas especificaciones esta fue la tecnología que se eligió para desarrollar la red.

En la Tabla 4 se puede observar una comparación entre todas las tecnologías que se analizaron.

Tabla 4: *Comparativa de tecnologías de comunicación inalámbrica.*

MÓDULO	ALCANCE	VELOCIDAD	PRECIO
XBEE S1	100 m	250 kbps	N/A
XBEE S1 PRO	1,6 km	250 kbps	USD 54,95
XBEE S2C	1,2 km	250 kbps	USD 24,95
XBEE S2C PRO	3,2 km	250 kbps	USD 38,95
XBEE S3B PRO	6,5 km	200 kbps	USD 49,95
BLUETOOTH HC-06	40 m	2 Mbps	USD 7,95
WIFI ESP8266	300 m	150 Mbps	USD 7,95
NRF24L01	1 km	2 Mbps	USD 14,95

5. Descripción de la Solución

5.1 Proceso de Diseño

Dentro de los requisitos que debía cumplir el diseño, el sistema debía ser escalable, auto configurable y redundante. El proceso de comunicación se desarrolla en las siguientes etapas primero, se realiza una rutina de configuración de la red para habilitar y contabilizar los nodos activos. Seguidamente, el nodo central realiza la solicitud de datos a los nodos secundarios, estos toman los datos de los sensores que tengan conectados y envían la información de vuelta al nodo central. Por último, el nodo central almacena la información en un archivo CSV.

Para realizar la comunicación del nodo central al secundario y viceversa, se utiliza el módulo nRF24L01 conectado a los puertos GPIO de la Raspberry Pi y que se comunican mediante el estándar SPI. Los datos luego se empaquetan en una trama de acuerdo con el protocolo *Enhanced Shockburst*, y son enviados hasta el otro nodo. El nodo receptor recibe los datos, los desempaqueta y los almacena en el archivo CSV junto con la fecha y hora de la medida. Y, por último, se guarda el archivo en la memoria local, es decir, en la tarjeta microSD. Los datos desde los sensores vienen codificados de tal manera que permite al nodo central identificar su origen y poder acomodarlos correctamente dentro de la base de datos.

A continuación, se puede ver la rutina general del sistema completo:

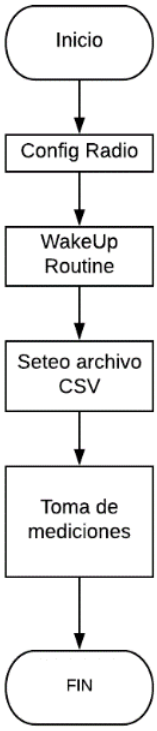


Figura 29: Diagrama de flujo general del sistema

La rutina de reconocimiento (llamada WakeUp) realiza un recorrido por todas las direcciones de pipe abierta. Abre el canal de TX para esa respectiva dirección y envía la señal de reconocimiento, si no recibe respuesta después de 5 intentos, se abre el pipe de TX para la siguiente dirección y reinicia la secuencia. Si, por el contrario, se recibe una respuesta, se lleva la cuenta de los nodos activos y se genera una lista aparte con las direcciones de dichos nodos.

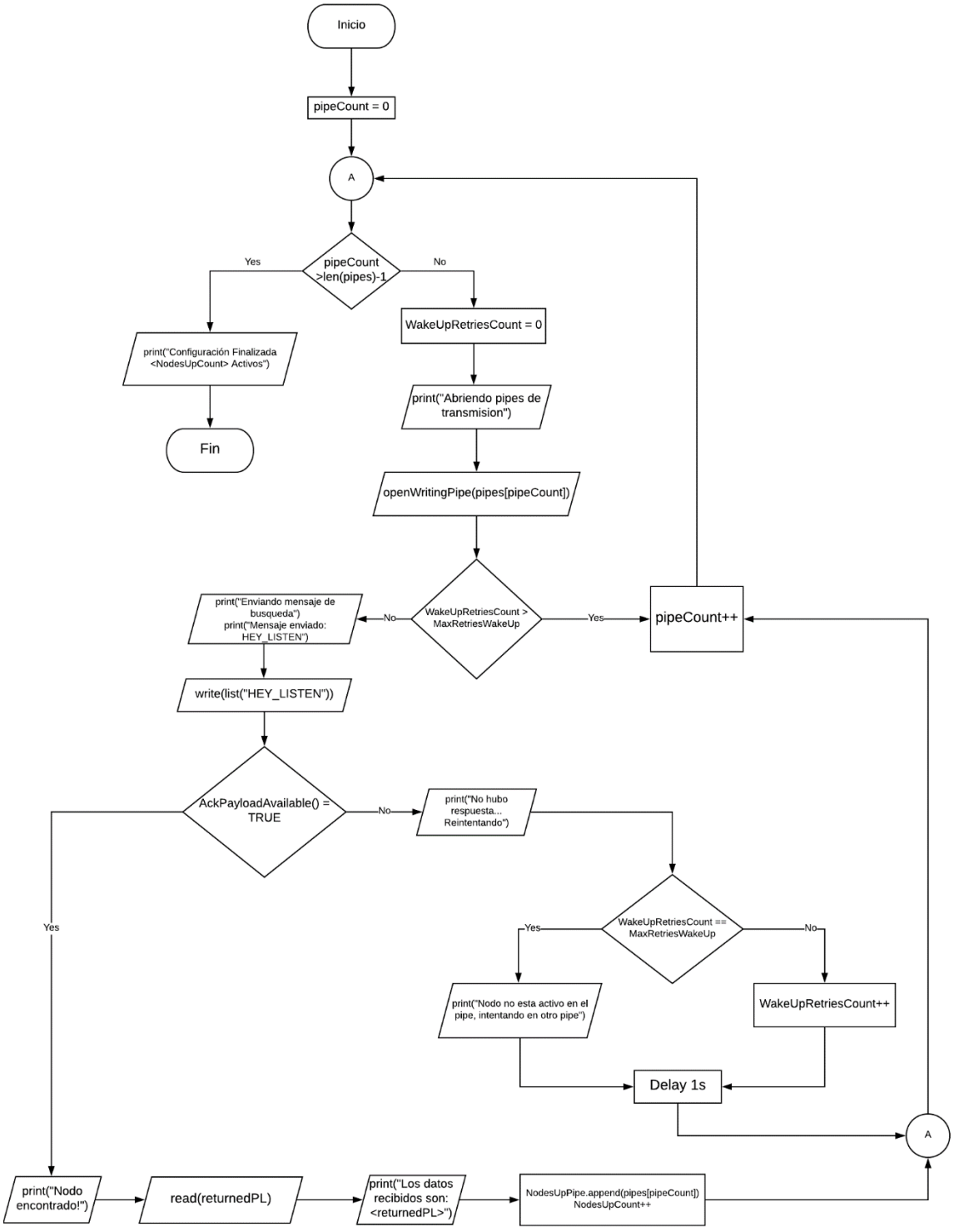


Figura 30: Diagrama de flujo de la rutina WakeUp

Para la configuración del archivo CSV, simplemente se toma la cantidad de nodos activos y se genera una línea de texto con la cantidad de nodos activos. Adicionalmente, se realiza una revisión de la existencia del archivo.

Por último, la rutina de toma de mediciones utiliza el mismo principio de abrir el canal de TX, dirección por dirección, y envía un mensaje de solicitud de datos, una vez que recibe la respuesta, genera un *stream* de texto en donde se van anexando los valores obtenidos desde los nodos para ser guardados en el archivo CSV (Figura 32)

Mientras tanto, desde el punto de vista del nodo secundario, el programa es más sencillo, no consta de tantas partes, ya que solo se dedica a escuchar el mensaje proveniente del nodo central y enviar la información que se solicite, como se puede observar en la Figura 31

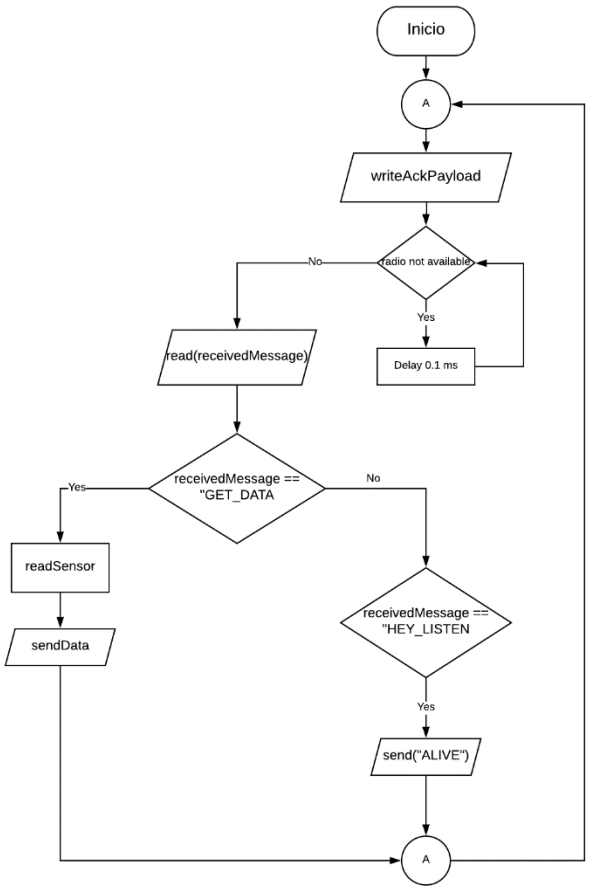


Figura 31: Diagrama de flujo para el programa del nodo secundario

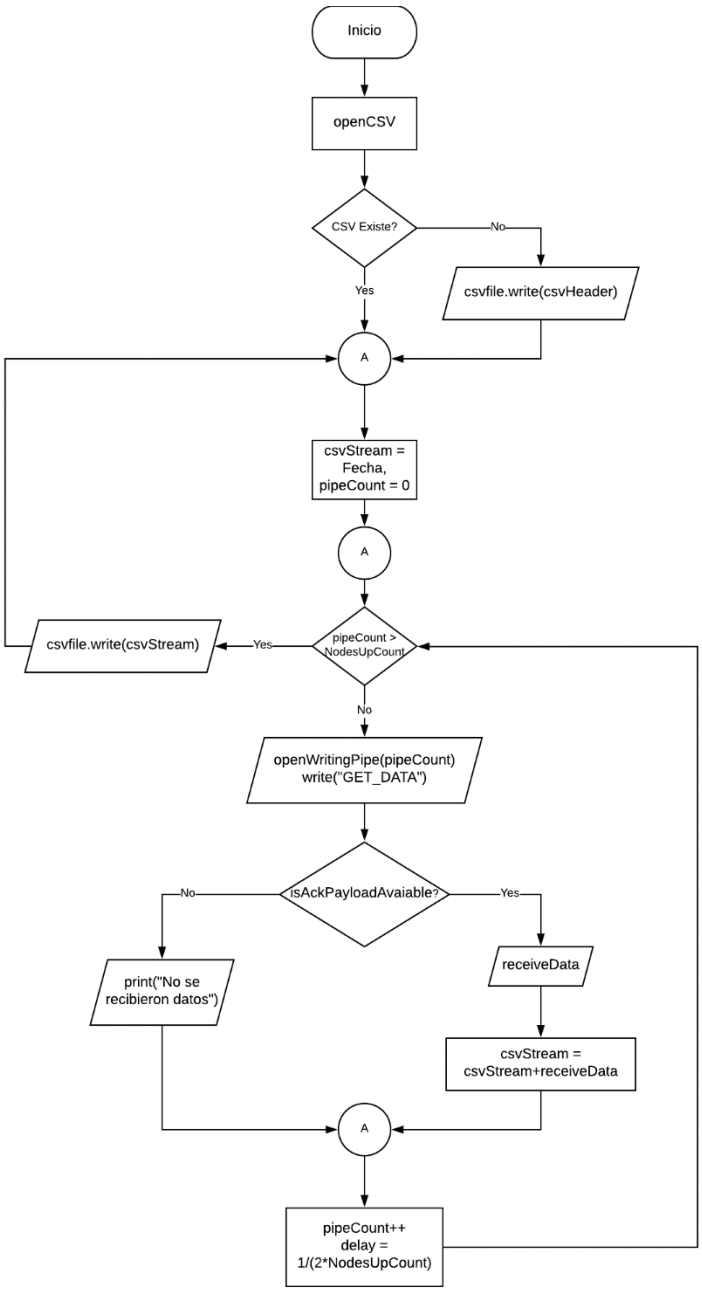


Figura 32: Diagrama de flujo de la toma de mediciones para el nodo central

Ambos nodos realizan todas estas acciones mediante un programa escrito en Python 2.7 (Apéndice A.1). Se escogió este lenguaje por su facilidad a la hora de leer, es decir, una persona puede leer el código y comprender con relativa facilidad como funciona.

5.2 Implementación del Diseño

El módulo nRF24L01 se comunica con la tarjeta Raspberry Pi mediante el puerto SPI, por lo tanto, es necesario primera habilitar este puerto. Para lograr esto es necesario ingresar a la configuración del hardware que se encuentra en las preferencias y se denomina "Configuración de Raspberry Pi". Una vez ingresado a este menú, es necesario ir a la sección de Interfaces (1), cambiar la sección de SPI de *Desactivado* a *Activado* (2), guardar la configuración (3) y reiniciar.

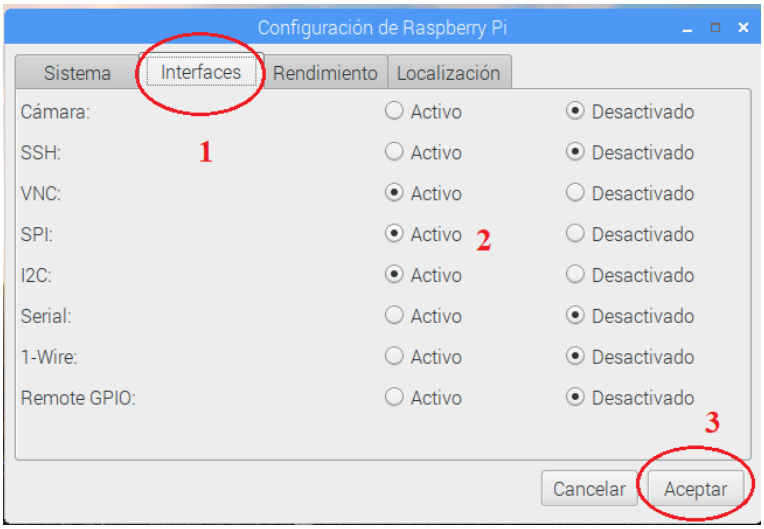


Figura 33: Menú de Configuración de Raspberry Pi.

Una vez que se encuentra habilitado el puerto, es necesario instalar la librería que permite que Python envié información por el puerto. La misma se encuentra alojada en el sitio GitHub, solamente es necesario clonar el repositorio y ejecutar el *script* de instalación. El nombre de la librería es "*SpiDev*".

La conexión física se realiza a través de los pines GPIO. En la Tabla 5 se puede observar la asignación de pines y donde se deben conectar cada uno. Se incluye ambas notaciones para tener facilidad de conexión ya que existen gran variedad de accesorios para Raspberry Pi que cambian la asignación de pines entre el chip y la tarjeta. Además, la Figura 34 muestra la distribución de pines de la tarjeta y su respectiva asignación a los pines del chip. La conexión se lleva a cabo mediante cables o *jumpers* hembra-hembra como se muestra en la

Tabla 5: *Conexión física de Raspberry Pi y nRF24L01*

NRF24L01+	PI (CHIP)	PI (TARJETA)
VCC	3.3v	Pin 1
GND	Ground	Pin 6
CSN	GPIO08	Pin 24
CE	GPIO17	Pin 11
MOSI	GPIO10	Pin 19
MISO	GPIO09	Pin 21
SCK	GPIO11	Pin 23

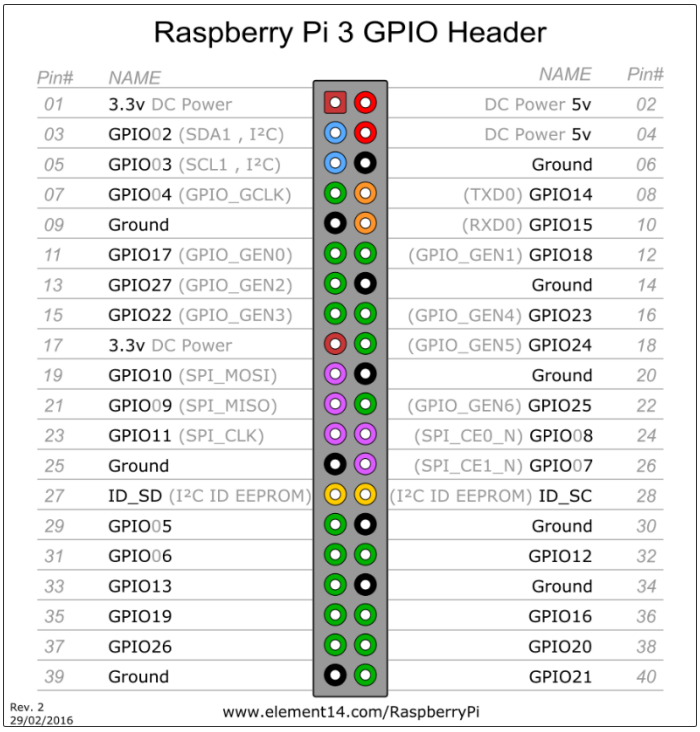


Figura 34: Asignación de pines GPIO de Raspberry Pi

Para facilitar la comunicación entre los nodos, existe una librería que permite enviar los comandos de configuración y operación a los módulos inalámbricos. La misma se encuentra alojada en GitHub y sirve como base para el desarrollo del sistema de comunicación. Ésta se llama “lib_nrf24” y consiste en un archivo “.py” con una serie de definiciones y funciones.

Como primer paso es necesario establecer que la comunicación entre la tarjeta Raspberry Pi y el radio nRF24L01 es correcta. Para ello, se debe abrir el puerto SPI y especificar el bus y el pin CE que se va a utilizar para ese dispositivo. Algo importante que se debe rescatar es que se debe configurar la velocidad de transmisión entre estos dispositivos. De no hacerlo así, los datos que reciba la tarjeta serán erróneos. Esto se realiza mediante la siguiente rutina:

```
1. #Inicialización del puerto y configuración de velocidad
2. spi = spidev.SpiDev()
3. spi.max_speed_hz = 7529
4.
5. # Creación del objeto y apertura del puerto
6. radio = NRF24(GPIO, spi)
7. radio.begin(0, 17)
```

Inicialmente se configuro una velocidad de 7259 Hz para tener una configuración estable y de ahí aumentar hasta llegar a la velocidad requerida por el sistema. Seguidamente, es necesario realizar la configuración de los radios. Ambos deben tener la misma configuración. La misma se encuentra en la **Tabla 6**. Y para lograrlo se utiliza la siguiente rutina:

```
1. radio.setPayloadSize(32)
2. radio.setChannel(0x60)
3. radio.setDataRate(NRF24.BR_2MBPS)
4. radio.setPALevel(NRF24.PA_MIN)
5. radio.setAutoAck(True)
6. radio.enableDynamicPayloads()
7. radio.enableAckPayload()
8. unique_ID = 1
```

Tabla 6: *Configuración de los radios*

	NODO CENTRAL	**NODO SENSOR 1**	**NODO SENSOR 2**
TAMAÑO MÁXIMO DE CARGA.	32 bytes	32 bytes	32 bytes
CANAL	0x60	0x60	0x60
TAZA DE BITS	1MBPS	1MBPS	1MBPS
POTENCIA DEL AMP	PA_MAX	PA_MAX	PA_MAX
ACK DE CARGA	Activo	Activo	Activo
TAMAÑO DE CARGA DINÁMICO	Activo	Activo	Activo
UNIQUE_ID	0	1	2

Estos módulos tienen la capacidad de especificar un tamaño máximo para la carga 32 bytes. Sin embargo, no todos los paquetes que se envían tendrán una carga de este tamaño. Esto se puede lograr gracias a que se activa la opción de carga dinámica, así se pueden ahorrar recursos y aumentar la velocidad al no tener que enviar basura para llenar la carga completa.

Una vez finalizada toda la configuración de los radios, se deben abrir los distintos medios de comunicación entre los nodos. Para el caso del nodo central, se habilitan 6 medios de recepción, uno para cada uno de los posibles nodos secundarios. Como parte de los procedimientos de inicialización se encuentra la creación de una lista o *pool* de direcciones. En este paso, es cuando se asigna una dirección a cada pipe y se habilita la recepción. Si se trata del nodo secundario, solamente se abre un *pipe* de lectura y otro de escritura, ambos con la misma dirección. Esto se logra así

Para el nodo central:

```
pipes = [
    [0xe7, 0xe7, 0xe7, 0xe7, 0xe7],
    [0xc2, 0xc2, 0xc2, 0xc2, 0xc2],
    [0xC3],
    [0xC4],
    [0xC5],
    [0xC6]
]
radio.openReadingPipe(0, pipes[0])
radio.openReadingPipe(1, pipes[1])
radio.openReadingPipe(2, pipes[2])
radio.openReadingPipe(3, pipes[3])
radio.openReadingPipe(4, pipes[4])
radio.openReadingPipe(5, pipes[5])
```

Y para el nodo secundario:

```
radio.openWritingPipe(pipes[0])
radio.openReadingPipe(0, pipes[0])
```

Una vez que se tiene la red lista con los nodos activos, se abre el archivo CSV, se envía la solicitud de datos y se captura lo que se recibe. AL finalizar la prueba se genera dentro de la carpeta de reportes un archivo CSV para el día de la prueba. La estructura de las carpetas es de la siguiente manera:

```
bridge_monitor_software:
.
├── __init__.py
├── lib
│   ├── __init__.py
│   ├── __init__.pyc
│   ├── lib_nrf24.py
│   └── lib_nrf24.pyc
├── logs
│   └── dummy.log
├── mainCentral
│   ├── __init__.py
│   └── main.py
├── mainSensor
│   ├── __init__.py
│   └── main.py
└── reportes
    ├── 2018-06-05.csv
    └── dummy.rpt
```

6. Resultados y Análisis

6.1 Conexión entre el nodo y el módulo de radio

Primero se comprueba la comunicación entre el nodo y el módulo de radio. Esto se debe llevar acabo en todos los dispositivos de la red, sea que son nodos secundarios o nodos centrales. En una de las primeras pruebas no se especificaba la velocidad (en Hz) a la que se debían comunicar estos módulos y generaba errores en los datos que se recibían luego de la configuración. La Figura 35 muestra la información recibida mientras que la Tabla 7 muestra las diferencias entre los datos que se envían al módulo de radio y los que se recibe.

```
STATUS   = 0x01 RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=0 TX_FULL=1
RX_ADDR_P0-1     = 0xffffffffff 0xfcfcfcfcfc
RX_ADDR_P2-5     = 0xfc 0xfd 0xfc 0xfc
TX_ADDR          = 0xffffffffff
RX_PW_P0-6       = 0x06 0x06 0x00 0x00 0x00 0x00
EN_AA            = 0x07
EN_RXADDR        = 0x00
RF_CH            = 0x1e
RF_SETUP         = 0x03
CONFIG           = 0x41
DYNPD/FEATURE    = 0x00 0x00
Data Rate        = 1MBPS
Model            = nRF24L01
CRC Length       = Disabled
```

Figura 35: Configuración errónea recibida desde el módulo de radio

Para solucionar esto, se configuró la velocidad de transmisión del puerto SPI. La librería SpiDev cuenta con un comando que permite configurar el reloj basado en un divisor del reloj, por lo que los valores son fijos. Se inició con un valor base de 7629Hz, esa es la velocidad más lenta que se puede configurar y por eso se utiliza. Una vez configurada, la comunicación se realiza con éxito como se observa en la Figura 36.

Tabla 7: *Comparación entre datos enviados y recibidos*

REGISTRO/PARÁMETRO POR CONFIGURAR	DATO CONFIGURADO	DATO RECIBIDO
RX_ADDR_P0	0xe7e7e7e7e7	0xffffffffff
RX_ADDR_P1	0xc2c2c2c2c2	0xfcfcfcfcfc
TX_ADDR	0xe7e7e7e7e7	0xffffffffff
EN_RXADDR	0x07	0x00
RF_CH	0x60	0x1e
RF_SETUP	0x08	0x03
FEATURE	0x06	0x00
POTENCIA PA	PA_MIN	PA_MIN
TAZA DE BITS	2MBPS	1MBPS
CRC LENGTH	16 bits	Disabled

```
STATUS   = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1     = 0xe7e7e7e7e7 0xc2c2c2c2c2
RX_ADDR_P2-5     = 0xc3 0xc4 0xc5 0xc6
TX_ADDR          = 0xe7e7e7e7e7
RX_PW_P0-6       = 0x20 0x20 0x00 0x00 0x00 0x00
EN_AA            = 0x3f
EN_RXADDR        = 0x07
RF_CH            = 0x60
RF_SETUP         = 0x08
CONFIG           = 0x5c
DYNPD/FEATURE    = 0x3f 0x06
Data Rate        = 2MBPS
Model            = nRF24L01
CRC Length       = 16 bits
PA Power         = PA_MIN
```

Figura 36: Configuración correcta recibida desde el módulo de radio

El problema de comunicación surge en el momento en el que no se especifica la velocidad que se va a utilizar. El integrado central de Raspberry Pi (BCM2837) cuenta con un reloj interno que controla todas las interfaces. Para poder controlar la velocidad con la que los dispositivos se comunican se dispone de un divisor de reloj y este utiliza potencias de dos. Sin embargo, cuando no se utiliza ningún valor explicito, la librería configura la velocidad de transmisión a la máxima posible del reloj que en este caso es 250 MHz (o 400 MHz si la tarjeta entra en modo turbo).

Aunque la velocidad de transmisión permita que el módulo de radio y la tarjeta se puedan comunicar, también deben tomarse en cuenta que debe ser lo suficientemente rápida para poder manejar las cantidades de datos que se deben muestrear por segundo. Para poder encontrar la velocidad adecuada se realizaron distintas pruebas

6.2 Comunicación entre los radios

El siguiente paso establecer la comunicación entre dos módulos de radio. Para lograr que ambos módulos se puedan comunicar es necesario que tengan la misma configuración. Parámetros básicos como la velocidad de transferencia, el canal de RF, y las direcciones de los pipes son las opciones principales, sin embargo, hay otros parámetros que deben ser configurados también para tener una comunicación exitosa.

Como primera prueba se realizó la comunicación básica entre los dos módulos. Valiéndose de los ejemplos que acompañan a la librería lib_nrf24.py (Anexo B.1), se estableció el enlace inicial. Este ejemplo consiste en una solicitud simple al nodo secundario en el que se pide la temperatura, el nodo secundario recibe el mensaje y reacciona ante esa solicitud especifica. Luego, realiza la medición de la temperatura (que en este caso es una variable dentro del programa) y le envía de vuelta al nodo central.

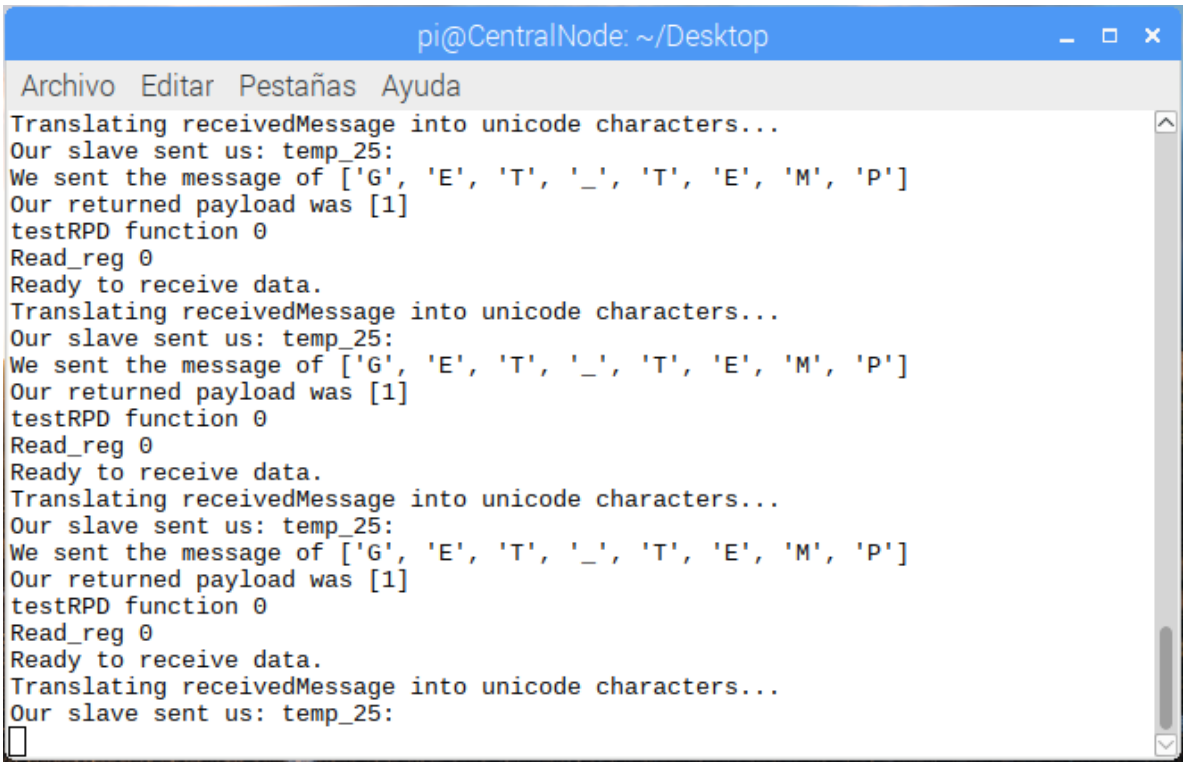


Figura 37: Resultado del enlace exitoso visto desde el nodo central

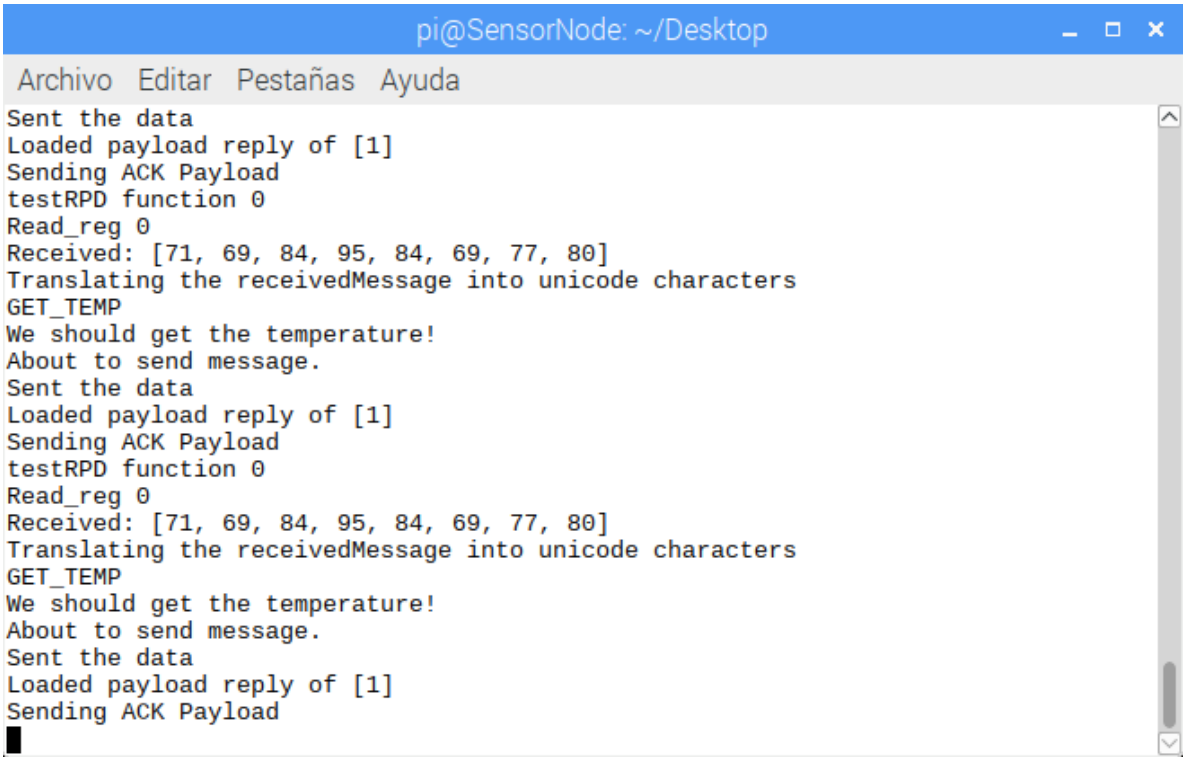


Figura 38: Resultado del enlace exitoso visto desde el nodo secundario

Otro aspecto importante de esta sección fue el alcance que pudieran tener los nodos. Como requisito por parte de la empresa, se necesitaba que la distancia mínima fuera de 100m, con una distancia ideal de hasta un 1km. Esta prueba se realizó utilizando un nodo fijo en una posición, el nodo central, y otro nodo móvil, el secundario. Se procedió a correr el programa y monitorear que la comunicación fuera estable a medida de la distancia entre los nodos aumentara. Para media la distancia entre los nodos eficazmente, se utilizaron herramientas de mapas satelitales que permiten tomar medidas desde un punto A hasta un punto B. De esta prueba se logró determinar que a 300 metros la conexión se mantiene estable, sin pérdida de paquetes ni perdida del radioenlace.

El módulo nRF24L01 cuenta con un registro que permite medir la potencia recibida, el bit menos significativo del registro RPD se activa a 1 cuando la potencia recibida es mayor a -64 dBm y se mantiene en 0 si está por debajo de este nivel. Sin embargo, durante las pruebas realizadas no se logró tener un 1 en el bit menos significativo. Aun cuando los nodos se encontraban en cercanía, siempre se mantenía en 0. Adicionalmente, la información que brinda no es de particular importancia, ya que no permite saber cuánta es la potencia exacta que se está recibiendo, simplemente reporta si se encuentra por encima o por debajo de un valor fijo.

6.3 Almacenamiento de la información en los nodos

Una vez que el nodo central puede tener un radio enlace estable y recibir los datos correctos, se procede a almacenar los datos en el nodo central. Para esto se genera un archivo tipo csv en la carpeta "reportes" y que siguen el siguiente formato: "<*Año*>_<*Mes*>_<*Dia*>.csv". Cuando el programa inicia se realiza un chequeo que determina si el archivo existe o no. Si al realizar la prueba se encuentra que el archivo existe, se genera un mensaje de alerta indicando que ya existe un archivo para el día, y procede a abrir en archivo y agregar la información a partir de la última línea.

En el siguiente caso, se corrió utilizando un único sensor, esta representa la primera prueba de almacenaje de datos, por lo que el valor identificador del sensor aún se encuentra como prefijo del dato.

Tabla 8: *Primera prueba de almacenamiento de datos*

TIMESTAMP	SENSOR1
2018-05-23 11:28:16.166055	1_13229
2018-05-23 11:28:16.474306	1_13436
2018-05-23 11:28:16.783661	1_13514
2018-05-18 11:28:17.092944	1_13513
2018-05-18 11:28:17.401072	1_13468
2018-05-18 11:28:17.709366	1_13470
2018-05-18 11:28:18.017990	1_13480
2018-05-18 11:28:18.325837	1_13455
2018-05-18 11:28:18.633940	1_13442
2018-05-18 11:28:18.942915	1_13483
2018-05-18 11:28:19.250737	1_13460
2018-05-18 11:28:19.558837	1_13465
2018-05-18 11:28:19.867946	1_13517
2018-05-18 11:28:20.175867	1_13394
2018-05-18 11:28:20.483623	1_13450
2018-05-18 11:28:20.792929	1_13486
2018-05-18 11:28:21.102253	1_13488
2018-05-18 11:28:21.411689	1_13499
2018-05-18 11:28:21.721084	1_13496
2018-05-18 11:28:22.030336	1_13489
2018-05-18 11:28:22.338174	1_13441
2018-05-18 11:28:22.646205	1_13466
2018-05-18 11:28:22.955351	1_13514
2018-05-18 11:28:23.263191	1_13446
2018-05-18 11:28:23.571018	1_13463
2018-05-18 11:28:23.880514	1_13519
2018-05-18 11:28:24.189962	1_13509
2018-05-18 11:28:24.499832	1_13335
2018-05-18 11:28:24.809172	1_13493
2018-05-18 11:28:25.118579	1_13492
2018-05-18 11:28:25.427684	1_13486
2018-05-18 11:28:25.735448	1_13458
2018-05-18 11:28:26.043266	1_13460
2018-05-18 11:28:26.352487	1_13484
2018-05-18 11:28:26.661890	1_13487
2018-05-18 11:28:26.970647	1_13204
2018-05-18 11:28:27.278543	1_13450
2018-05-18 11:28:27.586801	1_13401
2018-05-18 11:28:27.896027	1_13496
2018-05-18 11:28:28.205446	1_13485

2018-05-18 11:28:28.514820	1_13454
2018-05-18 11:28:28.824066	1_13494
2018-05-18 11:28:29.133449	1_13515
2018-05-18 11:28:29.442750	1_13510
2018-05-18 11:28:29.752067	1_13514
2018-05-18 11:28:30.061451	1_13503
2018-05-18 11:28:30.370893	1_13481
2018-05-18 11:28:30.680868	1_13454
2018-05-18 11:28:30.991644	1_13379
2018-05-18 11:28:31.300087	1_13442
2018-05-18 11:28:31.609091	1_13462
2018-05-18 11:28:31.918189	1_13497
2018-05-18 11:28:32.226071	1_13441
2018-05-18 11:28:32.534100	1_13451
2018-05-18 11:28:32.843436	1_13502

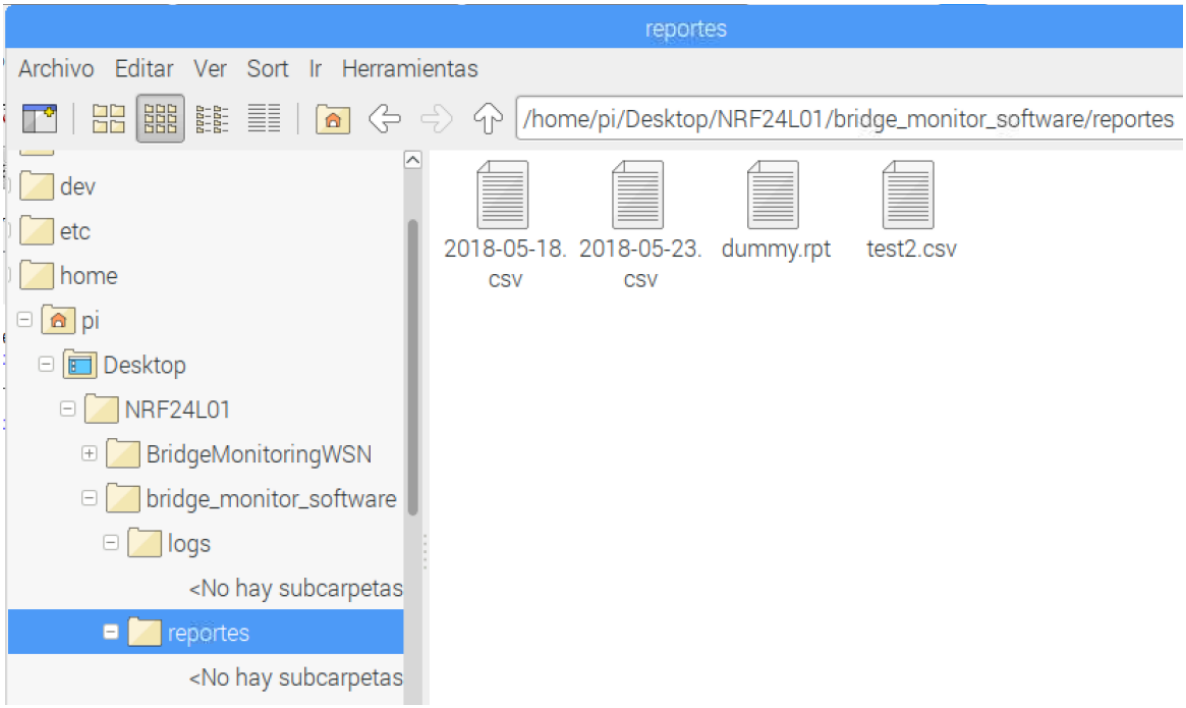


Figura 39: Captura de los archivos CSV creados

6.4 Rutina de autoconfiguración

Para la rutina de autoconfiguración, el nodo central envía un mensaje a través de cada *pipe* y espera la respuesta de cada nodo. Debido a la manera en el que se diseñó el protocolo Enhanced Shockburst, es necesario hacerlo así. Cuando el programa se inicia y se configuran los parámetros de los módulos de radio y los archivos de "log", se procede a la rutina de autoconfiguración.

En la Figura 40 se puede observar el resultado parcial de la búsqueda de nodos. En el caso del primer pipe (dirección 0xe7e7e7e7e7) se envía hasta un máximo de 5 veces el mensaje de activación, sin embargo, no se recibe respuesta por parte de este nodo por lo que el programa emite una alerta diciendo que no se encuentra un nodo activo en la dirección y procede a abrir otro pipe de comunicación e intentar comunicarse con un nodo. En la segunda dirección (0xc2c2c2c2c2) sucede lo contrario. Se encuentra un nodo activo, se documenta que se encuentra activo, se procede a abrir el tercer pipe y la configuración continua hasta que todos los pipes fueron escaneados. Cuando finaliza la configuración, se indica que así sucedió y se documenta la cantidad de nodos activos. En el caso de la Figura 40, por ejemplo, solo existe un nodo activo aparte del central por lo que así lo indica el programa.

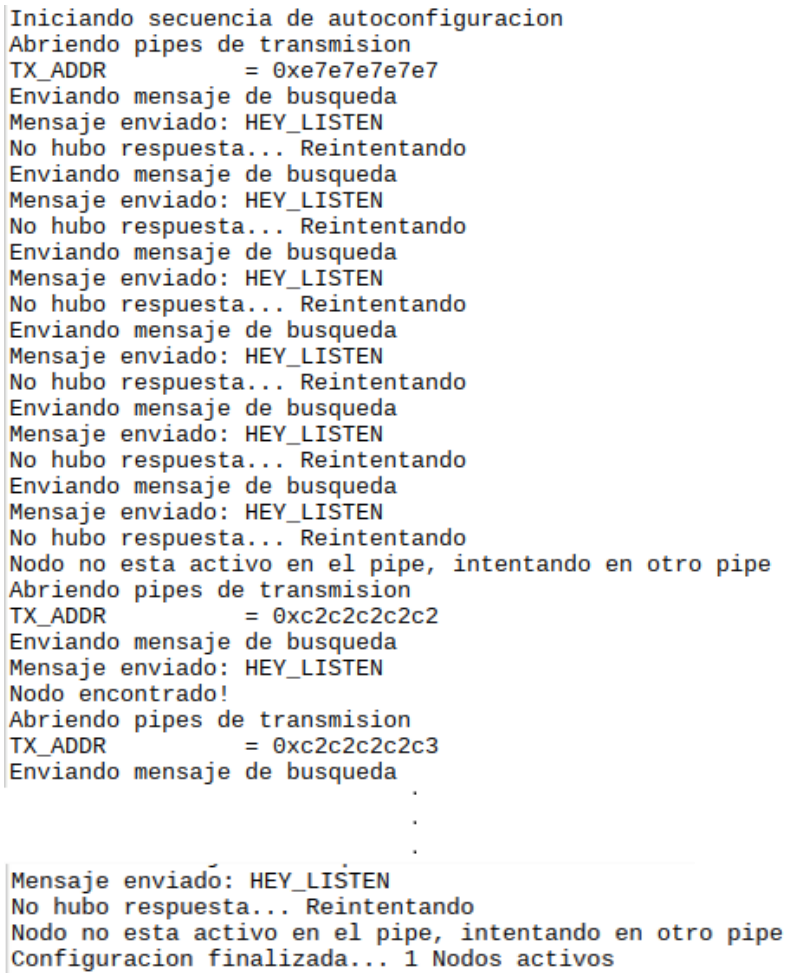


Figura 40: Resultado de la búsqueda de nodos

6.5 Verificación de los datos

La verificación de los datos se puede dividir en dos partes, la primera de ellas tiene que ver con el algoritmo de verificación por redundancia cíclica (CRC). El cual se aplica directamente en el hardware de los módulos NRF24L01. No es necesario realizarla en el lado de software ya que el módulo de radio descarta los paquetes que fallen la prueba de verificación.

La idea inicial era poder realizar mediciones con los sensores de desplazamiento ni vibración en los 3 ejes. Los nodos secundarios se conectarían a los mismo y podrían tomar medidas reales de los sensores. Sin embargo, ambos sensores se encuentran en desarrollo paralelamente a este en particular. Por lo tanto, las mediciones que se obtienen corresponden a simulaciones de datos. El sensor 1 está tomando una señal diferencial desde un convertidor analógico a digital, mientras que el sensor 2 está enviando siempre un valor contante.

Para comprobar que los valores que se están leyendo en el nodo secundario y los que se están recibiendo en el nodo central, podemos comparar los datos de la **Tabla 9** y

Tabla 10. Se puede observar, entonces, que los datos que se obtienen en el nodo central son correctos.

Tabla 9*: Resultado parcial de la toma de datos en el nodo central*

TIMESTAMP	SENSOR1	SENSOR2
2018-06-06 04:34:14.508154	1_14442	2_25
2018-06-06 04:34:37.438395	1_14446	2_25
2018-06-06 04:35:33.565084	1_14447	2_25
2018-06-06 04:36:45.417432	1_14437	2_25
2018-06-06 04:38:10.685724	1_14440	2_25
2018-06-06 04:38:45.216305	1_14440	2_25
2018-06-06 04:40:07.459485	1_14441	2_25

Tabla 10: *Resultado parcial de la toma de datos en un nodo secundario*

TIMESTAMP	SENSOR
2018-06-06 04:34:14.541242	14442
2018-06-06 04:34:37.470232	14446
2018-06-06 04:35:33.591343	14447
2018-06-06 04:36:45.451084	14437
2018-06-06 04:38:10.708949	14440
2018-06-06 04:38:45.248381	14440
2018-06-06 04:40:07.488682	14441

La segunda parte entra en juego cuando se descarta un dato. En caso de que un dato se corrompa y sea rechazado por el nRF24L01, la medición no se pierde ya que hay una copia

almacenada en el nodo sensor. Esta, se encuentra en un archivo CSV con el mismo formato que tiene el archivo del nodo central.

7. Conclusiones

- Se logró implementar una red de sensores escalable bajo las condiciones y restricciones del proyecto e-Bridge del CIVCO. El sistema envía una fila de caracteres por lo que el mensaje enviado puede incluir más de un dato dentro del mismo paquete hasta llegar a un máximo de 32 bytes. Además, el sistema puede escalarse en la cantidad de nodos desde 1 hasta 6 nodos secundarios.

- Se logra obtener un enlace estable a 300 metros de distancia entre los nodos, sin pérdida de paquetes a una frecuencia de muestreo de 30Hz. Con posibilidad de aumentar la frecuencia de ser necesario.

- Se obtienen datos comprobados mediante algoritmos de verificación, desde cada nodo sensor. La verificación la lleva acabo automáticamente el protocolo de comunicación en el módulo de radio, lo que garantiza que los datos almacenados ya han sido verificados.

- Se tiene una red compuesta de tres nodos, uno central y dos secundarios. Estos últimos envían información, ya sea de medición o de información, cuando el nodo central lo solicita.

- La red de sensores es capaz de almacenar los datos de todos los nodos conectados en el nodo central, así como una copia de los datos de cada nodo en cada uno respectivamente.

8. Recomendaciones

- En caso de necesitar más de 6 nodos, se puede agregar una rutina de configuración para generar nodos secundarios que funcionen como enrutadores y nodos central de una nueva subred. Además, serían el punto de contado entre el nodo central y la nueva subred que administra este nodo secundario.

- Se puede implementar una función que permita desarmar los mensajes con longitud mayor a 32 bytes en secciones más pequeñas y ensamblar esas secciones en el receptor.

Bibliografía

Castelli, M. (2002). *Network Consultants Handbook.* Cisco Press.

Cook, D., & Das, S. (2004). *Smart Environments: Technology, Protocols and Applications (Wiley Series on Parallel and Distributed Computing).* Nueva York: Wiley-Interscience.

Digi International. (n.d.). *Digi XBee® 802.15.4*. Retrieved from Digi International.

Faludi, R. (2010). *Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing.* O'Reilly Media, Inc.

Freeman, R. L. (2007). *Radio system design for telecommunications.* Hoboken: Wiley-Interscience.

Haykin, S. S. (2002). *Sistemas de comunicación.* Mexico: Limusa Wiley.

Huang, H.-W. (2005). *PIC Microcontroller: An Introduction to Software and Hardware Interfacing.* Cengage Learning.

Instituto Tecnológico de Costa Rica. (2016, Abril). *E-Bridge 2.0*. Retrieved from Tecnológico de Costa Rica: https://www.tec.ac.cr/proyectos/bridge-20

Instituto Tecnológico de Costa Rica. (2017). *Proyecto EBridge*. Retrieved from Tecnológico de Costa Rica: https://www.tec.ac.cr/proyectos/proyecto-ebridge

MaxElectrónica. (n.d.). *MaxElectrónica*. Retrieved from Módulo Transceptor Emisor Receptor NRF24L01 2.4GHz SPI: http://www.maxelectronica.cl/wifi-24ghz/273-modulo-transceptor-emisor-receptor-nrf24l01-24ghz-spi.html

National Instruments. (2009, Abril 22). *¿Qué es una Red de Sensores Inalámbricos?* Retrieved from National Instruments Web siite: http://www.ni.com/white-paper/7142/es/

Nordic Semiconductor. (2008). *nRF24L01+ Single Chip 2.4GHz Transceiver Preliminary Product Specification.* Retrieved from https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

Raspberry Pi Foundation. (n.d.). *Raspberry Pi 3 Model B - Raspberry Pi*. Retrieved from Raspberry Pi Foundation: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberry Pi Foundation. (n.d.). *Raspberry Pi Downloads - Software for the Raspberry Pi*. Retrieved from Raspberry Pi Foundation: https://www.raspberrypi.org/downloads/

Selmic, R. R., Phoha, V. V., & Serwadda, A. (2016). *Wireless Sensor Networks: Security, Coverage, and Localization.* Springer International Publishing.

Stremler, F. (1993). *Introducción a los sistemas de comunicación.* Addison-Wesley Iberoamericana.

Woolley, M. (Febrero de 2017). *Exploring Bluetooth 5 - Going the Distance*. Obtenido de Bluetooth Technology Website: https://blog.bluetooth.com/exploring-bluetooth-5-going-the-distance

Young, M. F. (2002). *Planning a Microwave Radio Link.* Retrieved from YDI Wireless: http://wireless.fcc.gov/outreach/2004broadbandforum/comments/YDI_microwavelink.pdf

Apéndice

A.1 Código del Programa

Nodo Central

```
from __future__ import division
import os,sys,inspect
current_dir =
os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe())))
parent_dir = os.path.dirname(current_dir)
sys.path.insert(0, parent_dir)

import RPi.GPIO as GPIO
from lib.lib_nrf24 import NRF24
import spidev
import logging
import csv
from datetime import datetime
from time import sleep, strftime, time

GPIO.setmode(GPIO.BCM)
pipes = [[0x78, 0x78, 0x78, 0x78, 0x78], [0xb3, 0xb4, 0xb5, 0xb6, 0xF1],
[0xb3, 0xb4, 0xb5, 0xb6, 0xcd], [0xb3, 0xb4, 0xb5, 0xb6, 0xa3], [0xb3,
0xb4, 0xb5, 0xb6, 0x0f], [0xb3, 0xb4, 0xb5, 0xb6, 0x05]]

spi = spidev.SpiDev()

radio = NRF24(GPIO, spi)
radio.begin(0, 17)
spi.max_speed_hz = 1953000
radio.setPayloadSize(32)
radio.setChannel(0x60)

radio.setDataRate(NRF24.BR_1MBPS)
radio.setPALevel(NRF24.PA_MAX)
radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

#radio.openWritingPipe(pipes[0])
radio.openReadingPipe(0, pipes[0])
radio.openReadingPipe(1, pipes[1])
radio.openReadingPipe(2, pipes[2])
radio.openReadingPipe(3, pipes[3])
radio.openReadingPipe(4, pipes[4])
radio.openReadingPipe(5, pipes[5])

radio.printDetails()

unique_ID = "0_"

refreshRate = 30
WakeUpRetriesCount = 0
MaxRetriesWakeUp = 5
```

```
NodesUpCount = 0
NodesUpPipe = []
NodeCount = 1
exists_flag = 0
csvHeading = "Timestamp,"
reportes_path = 'reportes/'
csvfile_path = reportes_path + str(datetime.now().date()) + '.csv'


def receiveData():
    print("Listo para recibir los datos.")
    radio.startListening()
    print("Escuchando")
    while not radio.available(0):
        sleep(1 / 100)

    receivedMessage = []
    radio.read(receivedMessage, radio.getDynamicPayloadSize())

    print("Traduciendo el mensaje...")
    string = ""
    for n in receivedMessage:
        # Decode into standard unicode set
        if (n >= 32 and n <= 126):
            string += chr(n)
    print("El mensaje recibido fue: {}".format(string))
    radio.stopListening()
    return string

print ("Iniciando secuencia de autoconfiguracion")
for pipeCount in range(0, len(pipes)-1):
    WakeUpRetriesCount = 0
    radio.openWritingPipe(pipes[pipeCount])
    print("Abriendo pipes de transmision")
    radio.print_address_register("TX_ADDR", NRF24.TX_ADDR)
    while (WakeUpRetriesCount <= MaxRetriesWakeUp):
        print("Enviando mensaje de busqueda")
        print("Mensaje enviado: HEY_LISTEN")
        radio.write(list("HEY_LISTEN"))
        radio.print_address_register("TX_ADDR", NRF24.TX_ADDR)
        if radio.isAckPayloadAvailable():
            print("Nodo encontrado!")
            returnedPL = []
            radio.read(returnedPL, radio.getDynamicPayloadSize())
            print("Los datos recibidos son: {} ".format(returnedPL))
            print(receiveData())
            NodesUpPipe.append(pipes[pipeCount])
            NodesUpCount += 1
            break
        else:
            print("No hubo respuesta... Reintentando")
            if WakeUpRetriesCount == MaxRetriesWakeUp:
                print("Nodo no esta activo en el pipe, intentando en otro
pipe")
            WakeUpRetriesCount += 1
            sleep(1)
delay = 1/(NodesUpCount*refreshRate)
```

```
print("Configuracion finalizada... {0} Nodos
activos".format(str(NodesUpCount)))
print("Tiempo de retardo: {}".format(str(delay)))

if (os.path.isfile(str(csvfile_path))):
    exists_flag = 1
    print("El archivo ya existe!")
else:
    exists_flag= 0
    print("Archivo inexistente!")
    print("Creando archivo nuevo")

while (NodeCount <= NodesUpCount):
    if NodeCount == NodesUpCount:
        csvHeading = csvHeading+"Sensor"+str(NodeCount)+"\n"
    else:
        csvHeading = csvHeading+"Sensor"+str(NodeCount)+","
    NodeCount += 1

with open(csvfile_path, 'a') as csvfile:
    if (exists_flag == 0):
        print(csvHeading)
        csvfile.write(str(csvHeading))
    print("Iniciando Toma de Datos")
    csvStream = str(datetime.now())+","
    while(1):
        command = "GET_DATA"
        for pipeCount in range(0, NodesUpCount):
            print(pipeCount)
            radio.openWritingPipe(NodesUpPipe[pipeCount])
            radio.print_address_register("TX_ADDR", NRF24.TX_ADDR)
            radio.write(list(command))
            print("Mensaje enviado: {}".format(list(command)))
            if radio.isAckPayloadAvailable():
                returnedPL = []
                radio.read(returnedPL, radio.getDynamicPayloadSize)
                message = receiveData()
                #print("Recibido: {}".format(message))
                csvStream = csvStream+str(message)+","
            else:
                print("No se recibieron datos")
            sleep(delay)
        csvStream = csvStream+"\n"
        print(csvStream)
        csvfile.write(str(csvStream))
        print("DEBUG: final del ciclo")
```

Nodo Secundario

```
from __future__ import division

import os,sys,inspect
current_dir =
os.path.dirname(os.path.abspath(inspect.getfile(inspect.currentframe())))
parent_dir = os.path.dirname(current_dir)
sys.path.insert(0, parent_dir)

import RPi.GPIO as GPIO
from lib.lib_nrf24 import NRF24
import spidev
import logging
import csv
import Adafruit_ADS1x15
from datetime import datetime
from time import sleep, strftime, time

GPIO.setmode(GPIO.BCM)
pipes = [[0x78, 0x78, 0x78, 0x78, 0x78], [0xb3, 0xb4, 0xb5, 0xb6, 0xF1],
[0xcd], [0xa3], [0x0f], [0x05]]

spi = spidev.SpiDev()

radio = NRF24(GPIO, spi)
radio.begin(0, 17)
spi.max_speed_hz = 1953000
radio.setPayloadSize(32)
radio.setChannel(0x60)

radio.setDataRate(NRF24.BR_1MBPS)
radio.setPALevel(NRF24.PA_MAX)
radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openWritingPipe(pipes[1])
radio.openReadingPipe(1, pipes[1])

radio.printDetails()
radio.startListening()

unique_ID = "1_"

reportes_path = '../reportes/'
csvfile_path = reportes_path + str(datetime.now().date()) + '.csv'

################# ADC Setup #################
adc_input = Adafruit_ADS1x15.ADS1115()
GAIN = 1

def readSensor():
    flex = adc_input.read_adc_difference(0, gain=GAIN)
    return str(flex)
```

```
def sendData(ID, value):
    radio.stopListening()
    sleep(1/60)
    message = list(ID) + list(value)
    print("Iniciando envio de datos.")
    radio.write(message)
    print("Datos enviados")
    radio.startListening()

while(1):
    ackPL = [1]
    radio.writeAckPayload(1, ackPL, len(ackPL))
    print("Enviando ACK")
    while not radio.available(0):
        sleep(1 / 100)
    receivedMessage = []
    radio.read(receivedMessage, radio.getDynamicPayloadSize())
    print("Recibido: {}".format(receivedMessage))

    print("Traduciendo el mensaje recibido...")
    string = ""
    for n in receivedMessage:
        # Decode into standard unicode set
        if (n >= 32 and n <= 126):
            string += chr(n)
    print(string)

    # We want to react to the command from the master.
    command = string
    if command == "GET_DATA":
        print("Solicitud de datos recibida")
        flex = readSensor()
        sendData(unique_ID, flex)
    elif command == "HEY_LISTEN":
        print("Secuencia de autoconfiguracion")
        sendData(unique_ID,"ALIVE")
    command = ""
    radio.writeAckPayload(1, ackPL, len(ackPL))
    print("Cargando respuesta de carga {}".format(ackPL))
```

A.2 Glosario

ACK	Acknowledge
AM	Amplitude Modulation
ASK	Amplitude Shift Keying
CRC	Cyclic Redundancy Check
CSV	Comma Separated Values
EIRP	Effective Isotropic radiated power
FM	Frequency Modulation
FSK	Frequency Shift Keying
FSL	Free Space Loss
MISO	Master Input- Slave Output
MOSI	Master Output-Slave Input
OSI	Open System Interconnection
PM	Phase Modulation
PRX	Primary Receiver
PTX	Primary Transmitter
RX	Recepción, Receptor, Recibir
SCK	Serial Clock
SOM	System Operating Margin
SPI	Serial Peripheral Interface
SS	Slave Select
TX	Transmisión, Transmisor, Transmitir

A.3 Hoja de información del proyecto

Información del Estudiante:

Nombre: Franz Vargas Acuña

Cédula; 3-0459-0760 **Carné ITCR:** 200932862

Dirección de su residencia en época lectiva: Los Sauces, San Francisco de Dos Ríos, San José

Dirección de su residencia en época no lectiva: Los Sauces, San Francisco de Dos Ríos, San José

Teléfono en época lectiva: 88393752 **Teléfono época no lectiva:** 88393752

Email: franzvargas91@gmail.com **Fax:** 22263068

Información del Proyecto

Nombre del Proyecto: Diseño del sistema de comunicación de una red inalámbrica de sensores para el estudio de la salud estructural en puentes

Área del Proyecto: Telecomunicaciones, programación

Información de la Empresa

Nombre: Centro de Investigación en Vivienda y Construcción

Zona: Los Ángeles, Oriental, Cartago, Cartago.

Dirección: Instituto Tecnológico de Costa Rica

Teléfono: 2550-2309 **Fax**: No **Apartado Postal**: 30101

Actividad Principal: Investigación y extensión de construcción

Información del encargado de la empresa:

Nombre: Giannina Ortiz Quesada

Puesto: Coordinadora proyecto e-Bridge

Departamento: CIVCO, Escuela Ingeniería en Construcción

Profesión: Ingeniera en Construcción **Grado Académico**: Maestría

Teléfono: 2550-2423 **Ext**.: 2423

Email: gortiz@tec.ac.cr

Información del asesor de la empresa:

Nombre: Ángel Navarro Mora

Puesto: Encargado de pruebas en estructuras

Departamento: Evaluación de puentes y estructuras

Profesión: Ingeniero en Construcción **Grado Académico**: Licenciatura

Teléfono: 2550-2246 **Ext**.: 2246

Email: ahnavarro@tec.ac.cr

Anexos

B.1 Código de ejemplo de lib_nrf24.py:

Master.py

```
import RPi.GPIO as GPIO
from lib_nrf24 import NRF24
import time
import spidev

GPIO.setmode(GPIO.BCM)

pipes = [[0xe7, 0xe7, 0xe7, 0xe7, 0xe7], [0xc2, 0xc2, 0xc2, 0xc2,
0xc2]]

radio = NRF24(GPIO, spidev.SpiDev())
radio.begin(0, 17)
radio.setPayloadSize(32)
radio.setChannel(0x60)

radio.setDataRate(NRF24.BR_2MBPS)
radio.setPALevel(NRF24.PA_MIN)
radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openReadingPipe(1, pipes[0])
radio.openWritingPipe(pipes[1])
radio.printDetails()

# radio.startListening()


def receiveData():
    print("Ready to receive data.")
    radio.startListening()

    while not radio.available(0):
        time.sleep(1 / 100)

    receivedMessage = []
    radio.read(receivedMessage, radio.getDynamicPayloadSize())

    print("Translating receivedMessage into unicode characters...")
    string = ""
    for n in receivedMessage:
        # Decode into standard unicode set
        if (n >= 32 and n <= 126):
            string += chr(n)
    print("Our slave sent us: {}:".format(string))
    radio.stopListening()
```

```
while(1):
    command = "GET_TEMP"
    message = list(command)
    # message = list("Hello World")
    radio.write(message)
    print("We sent the message of {}".format(message))

    # Check if it returned ackPL
    if radio.isAckPayloadAvailable():
        returnedPL = []
        radio.read(returnedPL, radio.getDynamicPayloadSize())
        print("Our returned payload was {}".format(returnedPL))
        receiveData()
    else:
        print("No payload received")
    time.sleep(1)
```

Slave.py

```
import RPi.GPIO as GPIO
from lib_nrf24 import NRF24
import time
import spidev

GPIO.setmode(GPIO.BCM)

pipes = [[0xe7, 0xe7, 0xe7, 0xe7, 0xe7], [0xc2, 0xc2, 0xc2, 0xc2,
0xc2]]

radio = NRF24(GPIO, spidev.SpiDev())
radio.begin(0, 17)
radio.setPayloadSize(32)
radio.setChannel(0x60)

radio.setDataRate(NRF24.BR_2MBPS)
radio.setPALevel(NRF24.PA_MIN)
radio.setAutoAck(True)
radio.enableDynamicPayloads()
radio.enableAckPayload()

radio.openWritingPipe(pipes[0])
radio.openReadingPipe(1, pipes[1])
radio.printDetails()

radio.startListening()


def getTemp():
    temp = 25
    return str(temp)
```

```
def sendData(ID, value):
    radio.stopListening()
    time.sleep(0.25)
    message = list(ID) + list(value)
    print("About to send message.")
    radio.write(message)
    print("Sent the data")
    radio.startListening()

while(1):
    ackPL = [1]
    radio.writeAckPayload(1, ackPL, len(ackPL))
    while not radio.available(0):
        time.sleep(1 / 100)
    receivedMessage = []
    radio.read(receivedMessage, radio.getDynamicPayloadSize())
    print("Received: {}".format(receivedMessage))

    print("Translating the receivedMessage into unicode characters")
    string = ""
    for n in receivedMessage:
        # Decode into standard unicode set
        if (n >= 32 and n <= 126):
            string += chr(n)
    print(string)

    # We want tp react to the command from the master.
    command = string
    if command == "GET_TEMP":
        print("We should get the temperature!")
        tempID = "temp_"
        temp = getTemp()
        sendData(tempID, temp)
    command = ""

    radio.writeAckPayload(1, ackPL, len(ackPL))
    print("Loaded payload reply of {}".format(ackPL))
```