SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La laguna

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

"DISEÑO LÓGICO DE NODO SENSOR INALÁMBRICO PARA APLICACIONES AGRONÓMICAS BASADO EN FREAKDUINO Y ELABORACIÓN DE PROTOTIPO"

POR

ING. MARÍA FERNANDA ALVAREZ VÉLEZ

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA

DIRECTOR DE TESIS
DR. HÉCTOR AURELIO MORENO CASILLAS

CODIRECTOR DE TESIS
DR. FRANCISCO GERARDO FLORES GARCÍA

ISSN: 0188-9060

TECNOLÓGICO DE LA LAGUNA

RIITEC: (03)-TMCIE-2018
Torreón, Coahuila. México,
Junio 2018

Torreón, Coah., 11/Juniol/2018
Dependencia: DEPI/CPCIE
Oficio: DEPIJ/CPCIE/059/2018
Asunto: Autorización de impresión de tesis.

C. María Fernanda Álvarez Vélez
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELÉCTRICA.
PRESENTE

Después de haber sometido a revisión su trabajo de tesis titulado:

"Diseño lógico de nodo sensor inalámbrico para aplicaciones agronómicas basado en Freakduino y elaboración de prototipo"

Habiendo cumplido con todas las indicaciones que el jurado revisor de tesis hizo, se le comunica que se le concede la autorización con número de registro **RIITEC: (03)-TMCIE-2018**, para que proceda a la impresión del mismo.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

ESTADOS UNIDOS MEXICANOS
SECRETARIA DE EDUCACIÓN PUBLICA
INSTITUTO TECNOLÓGICO de la laguna
División de Estudios de Posgrado e Investigación

P.A.
DR. ARMANDO LONGORIA DE LA TORRE
Jefe de la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de la Laguna

ALT/JIHJ

Torreón, Coah., 31/Mayo/2018

DR. ARMANDO LONGORIA DE LA TORRE
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Por medio de la presente, hacemos de su conocimiento que después de haber sometido a revisión el trabajo de tesis titulado:

"Diseño lógico de nodo sensor inalámbrico para aplicaciones agronómicas basado en Freakduino y elaboración de prototipo"

Desarrollado por el **C. María Fernanda Álvarez Vélez**, con número de control **M1613024** y habiendo cumplido con todas las correcciones que se le indicaron, estamos de acuerdo que se le conceda la autorización de la fecha de examen de grado para que proceda a la impresión de la misma.

ATENTAMENTE
EDUCACIÓN TECNOLÓGICA FUENTE DE INNOVACIÓN

Dr. Héctor Aurelio Moreno Casillas
Asesor/Director de Tesis

Dr. Francisco Gerardo Flores García
Comité Tutorial

Dr. Francisco Valdés Perezgasga
Comité Tutorial

Dr. Emmanuel Gómez Ramírez
Comité Tutorial

Dedicatoria

Dedico este trabajo especialmente a la M.C. Itzelle Johanna Oronoz Ponce por haber sido un gran apoyo incondicional en todo este tiempo, ayudándome a crecer como persona. Ich Liebe dich.

A mis padres que me dieron en la vida lo necesario para salir adelante. Y a Dios por permitirme llegar a este punto de la vida.

Agradecimientos

Expreso mi reconocimiento al Consejo de Ciencia y Tecnología (CONACYT) por su apoyo económico para el desarrollo de la presente investigación durante los estudios de maestría, así como la Dirección General de Educación Superior Tecnológica de la S.E.P.

Agradezco al Dr. Héctor Moreno Casillas por su continuo apoyo e impulsos que me dio durante la elaboración de este proyecto y al personal del Instituto Tecnológico de la Laguna.

También me permito agradecer a todos los desarrolladores de software libre por hacer posible la programación e integración de este proyecto.

Finalmente, doy gracias a mis amigas y futuras Ingenieras en Mecatrónica Elisa, Lorena, Andrea y Mariana por ayudarme a realizar mis pruebas de campo y ser un apoyo incondicional en todo momento.

RESUMEN

Las tecnologías van mejorando y su impacto puede verse reflejado en distintos ámbitos como lo es el sector agrícola. En México, la importancia de la instrumentación electrónica para este sector, reside en la obtención de datos para la medición y control de variables de interés como lo son la temperatura y humedad. Esta información es de gran ayuda para la producción y la investigación La implementación de redes inalámbricas de sensores ha ido en aumento en los últimos años debido a las ventajas que tienen en distintos campos y a su fácil instalación. En este trabajo se presenta un diseño lógico, basado en la tarjeta Freakduino como nodo sensor, cuya finalidad es reducir costos en la elaboración de un nodo; reducir el consumo energético con respecto a un nodo realizado en el ITL que utiliza la tecnología Waspmote Libelium. Su programación es realizada mediante software de acceso libre. Se diseñaron 3 prototipos de nodo cuyas características se adaptan perfectamente para una aplicación en el sector agrícola. El sistema se probó a campo abierto para realizar diferentes pruebas de potencia RSSI y distancia de alcance, haciendo un estudio completo del desempeño de los componentes de la tarjeta. Los resultados mostraron que fue posible optimizar el consumo energético del nodo sensor en un 50% obteniendo un alcance de 200 metros de transmisión de datos de sensores.

ABSTRACT

Technologies are improving and their impact can be reflected in different areas such as the agricultural sector. In Mexico, the importance of electronic instrumentation for this sector lies in obtaining data for the measurement and control of variables of interest such as temperature and humidity. This information is of great help, for production and research applications The implementation of wireless sensor networks has been increasing in recent years due to the advantages they have in different fields and their easy installation. In this paper a logical design is presented, based on the Freakduino card as a sensor node, whose purpose is to reduce costs in the development of a node; reduce the energy consumption with respect to a node made in the ITL that uses Waspmote Libelium technology. Its programming is done through free access software. Three node prototypes whose characteristics are perfectly adapted for an application in the agricultural sector were designed. The system was tested in the open field to perform different tests of RSSI power and range distance, making a complete study of the performance of the components of the card. The results showed that it was possible to optimize the energy consumption of the sensor node by 50%, obtaining a range of 200 meters of sensor data transmission.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

ADC. (Analog to Digital Converter) Convertidor analógico digital, convierte una señal analógica a una señal digital.

BROADCAST. (difusión): Forma de transmisión de la información donde un nodo denominado emisor envía la información a una multitud de nodos receptores de manera simultánea, sin necesidad de reproducir la misma transmisión nodo por nodo.

BPSK. Modulación Binaria por Desplazamiento de Fase.

CANAL. Es una porción especifica de radio frecuencia en la cual se está transmitiendo y recibiendo. Establecer un canal evita la colisión de datos entre diferentes radiotransmisores.

COBERTURA. Área geográfica que cubre una RIS.

CONECTIVIDAD. La capacidad de un nodo de conectarse a una RIS.

COLECTOR DE DATOS (data sink). Punto de colección de información de las fuentes.

GPS. Sistema de Posicionamiento Global.

I2C. Estándar que facilita la comunicación entre microcontroladores memorias y otros dispositivos.

INTERNET DE LAS COSAS. El que las tecnologías, desde las industriales hasta las domesticas, se puedan conectar a internet y subir datos.

IEEE. Instituto de Ingenieros Eléctricos y Electrónicos.

ITL. Instituto Tecnológico de la Laguna.

LQI. (L Quality Intesity) Calidad del Enlace.

MOTE. mota – nodo sensor.

NET. Red**.**

NODO. Es un dispositivo capaz de comunicarse en la red inalámbrica. En este caso la Freakduino es considerada un nodo que contiene un transmisor de radio.

MESH (Malla). Tipo de topología donde cada nodo de una red se conecta a otros nodos de forma descentralizada.

PROTOCOLO. Una serie de reglas que determinan las funciones de red.

PROTOCOLO 802.15.4. Protocolo que especifica la capa física (equipo de transmisión), así como de la capa de acceso (el método por el cual el software puede hablar y escuchar al equipo de transmisión).

RF. (Radio Frequency): Radiofrecuencia.

RED INALÁMBRICA. Interconexión de nodos sin la implementación de cables para su funcionamiento.

RIS. Red Inalámbrica de Sensores.

RSSI. Indicador de fuerza de la señal recibida de un transceptor.

RTC. Reloj de tiempo real.

SO. Sistema operativo.

SPI. Bus Interfaz de Periféricos Seriales.

TIEMPO DE VIDA. Tiempo en que puede permanecer operativa una RIS antes de quedarse sin energía.

TRANSMISOR DE RADIO. Es un dispositivo que puede tanto recibir como transmitir la información en la red inalámbrica.

SRAM. Memoria Estática de Acceso Aleatorio.

WLAN. Red Inalámbrica de Área Local.

WPAN. Red Inalámbrica de Área Personal.

WSN. Redes Inalámbricas de Sensores.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Antecedentes del problema

Hoy en día algunos de los problemas que enfrenta el mundo son la sobreexplotación de recursos naturales en el sector agropecuario. Los suelos constituyen el fundamento de los ecosistemas terrestres y son indispensables para mantener la productividad agrícola [1]. Una de las variables de interés en el monitoreo de cultivos es la humedad, indispensable para que exista un control sobre el insumo del agua y que los alimentos obtengan una calidad aceptable y cuenten con los nutrientes necesarios para el consumo humano.

En México, en 2014 el volumen de agua renovable promedio en el país era de 3,736 metros cúbicos por habitante al año. Sin embargo, existen diferencias sustanciales entre el sureste y el norte del territorio, regiones entre las cuales se observan áreas con gran escasez de agua y zonas con frecuentes eventos hidrometeorológicos. [2].

Una solución para la optimización del uso del agua, es analizar el comportamiento de los cultivos, mediante el monitoreo de la humedad y temperatura para determinar cuánta agua debe suministrarse y por medios automatizados emplear el volumen mínimo necesario y corregir factores cuyo impacto afecten al rendimiento del cultivo. En el Instituto Tecnológico de la Laguna, Flores Medina [3], en su proyecto “Redes Inalámbricas de Sensores en Aplicaciones Agronómicas” propone la implementación de la plataforma Waspmote Libelium, y utiliza para el desarrollo de nodos, los módulos Waspmote Libelium PRO v2.1 (Libelium, Zaragoza, España) como base principal, siendo un sistema para monitoreo de humedad del suelo y otras variables enfocado a ser una herramienta para la comunidad científica agrícola.

En el proyecto de Flores Medina [3], se realizó la estructura y construcción de un sistema basado en RIS con almacenamiento de datos en la base de datos geoespacial PostGIS construido con equipos de mediano costo y software de acceso libre. Se construyeron tres nodos sensores programados con las características necesarias para la aplicación en el sector agrícola. La variable considerada a monitorear por medio de los nodos sensores fue la humedad del suelo. El sistema RIS de Flores Medina se adapta a las necesidades de los investigadores de campo, lo cual quiere decir que es un sistema autónomo capaz de monitorear y registrar de forma automática la mayor información posible de variables agrícolas de interés. Fue probado en campo con obstáculos y en campo abierto en un invernadero en la ciudad de Torreón, Coahuila con ubicación geográfica (Latitud 25º35‘31.7‘‘, Longitud -103º22‘40‘‘), realizando pruebas generales de consumo energético, potencia y distancia con el objetivo de ser una herramienta para la comunidad científica de la Comarca Lagunera para el monitoreo y registro de información en periodos de cultivo, facilitando con ello la toma de decisiones.

El proyecto de esta tesis se va a realizar en el laboratorio de instrumentación (edificio 26) del Instituto Tecnológico de la Laguna (ITL) con domicilio en, Blvd. Revolución y Clzda. Cuauhtémoc CP 27000. El diseño se probará en las instalaciones del ITL para controlar la electrónica y seguridad del nodo, mientras que las pruebas de campo se desarrollarán en el Bosque Venustiano Carranza de Torreón.

1.1 Planteamiento del problema

El principal uso del agua se da en la agricultura con el 91% del agua que se extrae. De los 2,496 millones de m^3 que aprovecha, el 45% proviene de fuentes subterráneas y el 55% restante de las superficiales. En orden de importancia los otros usos son el público – urbano que demanda el 5%, pecuario 2%, e industrial 1%, cuya fuente es exclusivamente subterránea. A nivel municipio el uso del agua dominante es el agropecuario en 10 municipios, público en Gómez Palacio, Lerdo, y San Pedro de las Colonias e industrial en Torreón y Matamoros [5].

Dado que la mayor parte del agua en la Región Lagunera es para la agricultura y a que el Acuífero Principal Región Laguna esta sobreexplotado, es primordial utilizar técnicas agropecuarias que redunden en el ahorro de agua. Una forma es el uso de redes de monitoreo que permitan reducir el consumo de agua y aumenten la productividad.

El problema a resolver es: realizar un diseño lógico de un nodo sensor basado en la tarjeta Freakduino, que integre el uso del software Arduino de acceso libre y la fabricación del prototipo que permita implementar una RIS completamente funcional que ayude a la optimización del agua empleada en los sistemas de riego de cultivos para reducir el impacto ambiental y además ofrecer una mejora a los proyectos previamente mencionados y ofrecer una alternativa que mejore la potencia RSSI y reduzca el consumo energético y costos de diseño de un nodo.

1.2 Objetivos

1.2.1 General

Realizar un diseño lógico de nodo sensor basado en la tarjeta Freakduino, que integre el uso del software Arduino de acceso libre y la fabricación de un prototipo que sea implementado a una RIS enfocada a las aplicaciones agronómicas.

1.2.2 Específicos

- Implementar electrónica y programación de la tarjeta Freakduino-900 v3.0a y 2.1a como parte de la unidad de adquisición de datos, comunicación y monitoreo para que sea un nodo autónomo y funcional para una RIS.
- Diseñar el prototipo para que sea un sistema autónomo de nodo que incluya la implementación de un convertidor analógico digital MCP3208 que incremente la resolución de la tarjeta de 10 a 12 bits.
- Remplazar el uso de Waspmote Libelium con Freakduino para reducir el costo del diseño de un nodo sensor.
- Implementar algoritmos para colocar dispositivos a mayores distancias.

- Reducir el consumo energético en 40%, e incrementar la potencia de la señal en base a las distancias de transmisión y recepción con respecto al prototipo diseñado con Waspmote Libelium.
- Incorporar un diseño con modalidad de adaptable y configurable de otros sensores, que tenga las cualidades necesarias para aplicaciones agronómicas.
- Realizar pruebas con sensores de humedad y temperatura, de comunicación Wireless, consumo energético, potencia y distancia del prototipo en campo abierto, con y sin obstáculos para realizar un estudio de desempeño de Freakduino.
- Facilitar la toma de decisiones del usuario final en base a los resultados del monitoreo obtenido del nodo sensor para mejorar el uso del agua en cultivos.
- Integrar la tarjeta Freakduino con la plataforma mandrágora, mediante la implementación de computadora embebida Raspberry-pi para monitoreo de humedad en el suelo.

1.3 Justificación

La modernización del sector agrícola sigue dependiendo de las tecnologías provenientes de otros países, siendo esto una desventaja competitiva que limita la producción. De esta situación nace la idea de plantear iniciativas que permitan el desarrollo de prototipos de medición y monitoreo que puedan adaptarse a los recursos con los que cuenta el país.

En la actualidad la implementación de nodos en las redes inalámbricas de sensores que cubran con las necesidades requeridas, están en la mira de muchos investigadores y empresas tecnológicas; debido a que, a comparación de una red alámbrica se reducen costos por materiales e instalación. Por ejemplo, para el diseño de nodos sensores alámbricos se puede considerar como una opción el uso de cable de 6 hilos aunque puede variar el número de hilos y su instalación con tubos resistentes a la corrosión. También hay una desventaja del cable, a cierta distancia empieza a comportarse como línea de transmisión pudiendo afectar la medición

directa de los sensores, teniendo que implementar medidas que solucionen esto. Otra desventaja es el hecho de que cada nodo tiene una distancia limitada de transmisión, requiriendo retransmisores cuyos costos pueden ser de la mitad de precio del nodo, con sensores incluidos. El proyecto, por su utilidad en diversos campos de estudio, puede representar una oportunidad para el desarrollo de Redes inalámbricas de sensores.

Se prevé que el diseño lógico de nodos basado en las tarjetas Freakduino pueda reducir costos, presentar un manejo práctico y amigable de redes inalámbricas, que permita el sensado de diferentes parámetros requeridos para aplicaciones agrícolas, e incluso otras aplicaciones como monitoreo ambiental.

1.3.1 Impacto Social

No se prevé un impacto social en el futuro inmediato, debido a que aún siguen los estudios de las RIS para poder tener una tecnología que pueda mostrar resultados que beneficien de forma satisfactoria tanto a los agricultores como a los consumidores finales de los alimentos.

1.3.2 Impacto tecnológico

El impacto tecnológico del proyecto es la mejora de las RIS, en cuanto a consumo energético, potencia y distancia. A futuro, se espera que este estudio sea un apoyo en investigaciones relacionadas con las RIS, que aporte información que sirva de base no solo para el campo de la agronomía sino otros campos en los cuales esta tecnología se ha aplicado.

1.3.3 Impacto económico

Implementar la tarjeta Freakduino en el prototipo es una opción que permite reducir los costos en hasta un 59.53 % de los nodos para redes inalámbricas de sensores.

1.3.4 Impacto ambiental

El impacto ambiental del proyecto a futuro podría ser una reducción del consumo de agua para cultivos.

1.3.5 Viabilidad de la investigación

Se cuentan con los conocimientos necesarios en electrónica y programación para la realización del proyecto, así como de recursos para diseñar un prototipo funcional, autónomo y configurable.

1.4 Metodología

En la figura 1.1**,** se muestra el diagrama de flujo para la realización del proyecto donde se observa cada una de las partes que componen el desarrollo de este proyecto.

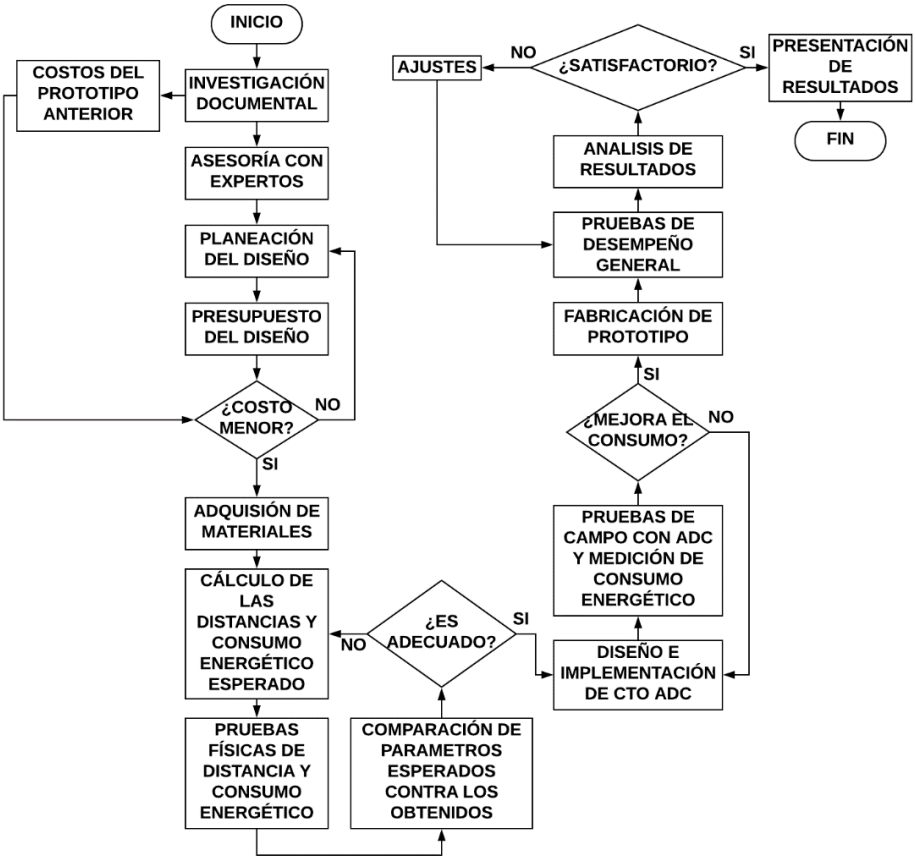


Figura 1.1. Diagrama de flujo de las actividades.

1.5 Cronograma

En la tabla 1.1, se presenta lo que es la programación de actividades y procedimientos a seguir en base al bosquejo del método.

Tabla 1.1. Cronograma.

Año	2017					2018					
MES	Ago	Sept	Oct	Nov	Dic	Ene	Feb	Mar	Abr	May	Jun
ACTIVIDADES											
Investigación documental											
Asesorías con expertos											
Planeación del diseño											
Presupuesto del diseño											
Adquisición de materiales											
Procedimiento de pruebas											
Comparación de resultados											
Diseño e implementación de cto ADC											
Pruebas de campo con ADC											
Medición de consumo energético											
Fabricación de prototipo											
Pruebas de desempeño											
Análisis de resultados											
Redacción de tesis											
Presentación de resultados											

1.6 Presupuesto

El presupuesto se presenta la tabla 1.2, El costo de desarrollar el prototipo de nodos es de $9584.07.

Tabla 1.2. Presupuesto.

No.	Concepto	Cantidad	Costo	Total
1	Freakduino 900 MHz + bottom mounted battery case set + envío	3	$1,580.66	$4,742.07
2	ADC MCP3208 + envío	2	$117.00	$234.00
3	Pilas recargables con cargador + envío	1	$309.00	$309.00
4	Material electrónico general y sensores	-	$1,000.00	$1,000.00
5	VH400/5M Capacitivo	3	$528.00	$1,584.00
6	Raspberry Pi 3 modelo B + envío	1	$1,179.00	$1,179.00
7	Material extra	-	$500.00	$500.00
		Total		**$9,548.07**

El costo del diseño de los nodos del proyecto de Flores Medina [3] se presenta la tabla 1.3. teniendo un total de $32,768.00 para comparación de costos entre proyectos.

Tabla 1.3. Costo del diseño del proyecto RIS.

No.	Concepto	Cantidad	Costo	Total
1	Nodo sensor Libelium Electrónica	3	$2,790.00	$8,370.00
2	Tarjeta de prototipado libelium	3	$720.00	$2,160.00
3	Multiplexor MPC508AP	3	$135.00	$405.00
4	Accesorios en general	-	$1,500.00	$1,500.00
5	Batería y panel solar	3	$1,170.00	$3,510.00
6	VH400/5M Capacitivo	6	$528.00	$1,584.00
7	Sensor TEM-HUME SHT11	1	$459.00	$459.00
8	Mano de obra	-	$500.00	$500.00
9	Agricultura Pro 802.15.4 5 DBI	1	$8,280.00	$8,280.00
10	Servicio de Aduana, IVA y envío	-	$6,000.00	$6,000.00
		Total		**$32,768.00**

CAPÍTULO 2

MARCO TEÓRICO

2.1 Redes Inalámbricas de Sensores (RIS)

Durante años pasados las Redes Inalámbricas de Sensores (RIS) han adquirido una importante relevancia y atención debido a las múltiples aplicaciones en diferentes ámbitos donde estas pueden implementarse. Sin embargo, en la actualidad se sigue vislumbrado un interés creciente por el uso de redes inalámbricas de sensores en numerosas aplicaciones como lo son, el monitoreo forestal, manejo de desastres, exploración espacial, automatización industrial, instalaciones de seguridad, protección de fronteras, y vigilancia en los campos de batalla [6].

La tecnología de las RIS es una de las más prometedoras por que otorgan una respuesta a las necesidades de comunicación y monitoreo. La búsqueda de utilizar nodos que no requieran demasiado mantenimiento, de que su coste sea relativamente bajo y la autoconfiguración de los mismos, es algo que debe tenerse muy presente debido a que las instalaciones de nodos se hacen en áreas de difícil acceso o interacción constante por parte de los técnicos o incluso debido a la exposición a los ambientes extremos.

Típicamente, un nodo sensor es un dispositivo pequeño que incluye tres componentes básicos: un subsistema de sensado para adquisición de datos del entorno físico, un subsistema de procesado para la información y almacenamiento y un subsistema para la transmisión inalámbrica de la información [7]. El nodo requiere de ejecutar múltiples tareas por lo que debe considerarse la adición de una fuente de poder que sea ideal para un tipo de aplicación en específico. El uso de las redes cableadas requiere de la instalación de varios metros de cableado que pueden ser dañados por diversos factores. La implementación de las redes inalámbricas reduce los costos de instalación del cableado y hay existe menor riesgo de daño al equipo.

Una red inalámbrica de sensores, RIS, es una red donde la transferencia de datos se da en forma inalámbrica. Se constituye de nodos sensores autónomos que tienen la capacidad de sensar, procesar, filtrar, almacenar, transmitir los datos recopilados por un sensor o varios, con una especificación en particular. Su estructura básica de muestra en la figura 2.1. Las RIS tienen las características de ser pequeñas, de baja potencia y de bajo costo, sólo pueden monitorear un cierto lugar de interés, ya que los nodos sensores sólo tienen la capacidad de transmitir los datos en pequeñas distancias, por lo que pueden ser usuarios de las Redes Inalámbricas de Área Personal (WPAN) [8].

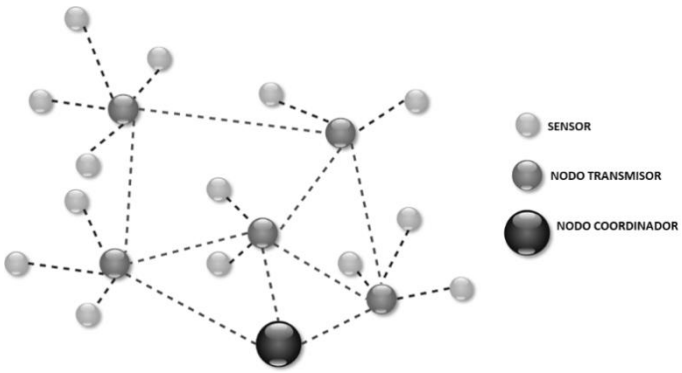


Figura 2.1. Estructura básica de una Red Inalámbrica de Sensores.

Existen proyectos que han logrado llevar a cabo sus desarrollos a un nivel de comercialización como en los casos de Mica2/MicaZ Motes [9], pero en México aún existe una alta dependencia de las compañías extranjeras que otorgan soluciones integradas que emplean una tecnología inalámbrica. Actualmente sólo pocas compañías en el país se dedican a los sistemas RIS y el desarrollo que van generando aún no es suficiente para poder suplir una demanda que a futuro hará un gran uso de tecnologías como lo es el internet de las cosas o su aplicación en conjunto con inteligencia artificial.

2.2 Elementos de una RIS

Una RIS consiste de varios módulos electrónicos, también conocidos como nodos sensores (o motes), que tienen acoplados diferentes sensores cada uno

teniendo como característica principal el uso del medio inalámbrico para lograr transmitir la información, siendo uno de los medios más utilizados el de radio frecuencia (RF). Los nodos sensores son distribuidos en un área especial estratégica para monitorear las variables de interés. Cada nodo sensor mide de forma puntual las variables utilizando los sensores que le fueran incorporados, para luego procesar esta información. Luego la información en el nodo sensor es transmitida estratégicamente e inalámbricamente entre los demás nodos sensores de la RIS o directamente a un nodo coordinador. El nodo coordinador comúnmente es conectado a un servidor donde se registra toda la información de la RIS permitiendo al usuario tener acceso a los datos de la red para analizarlos según su interés. Las características que pueden tener las RIS según Römer y Mattern [10] se enlistan en el cuadro sinóptico de la figura 2.2, mostrando aquellas que son las más destacadas.

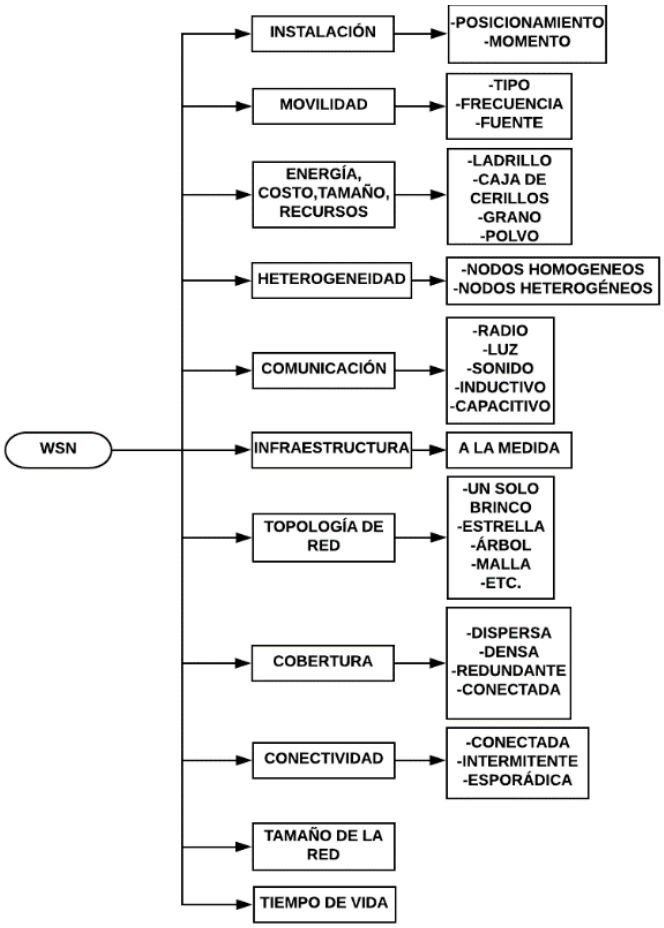


Figura 2.2. Características de una RIS [10].

Los nodos que son implementados en las RIS son usualmente tarjetas de adquisición de datos (DAQ). Las tarjetas de adquisición de datos actúan como la interfaz entre una computadora y señales físicas. La información recaudada por el sensor se pasa al DAQ, el cual se encarga de transformar los códigos del mundo real a los códigos digitales, como si se tratara de un intérprete que traduce de un lenguaje a otro, con el fin de que el sistema digital (es decir, cualquier computadora o dispositivo electrónico) sea capaz de comprender los signos del analógico [11].

2.3 Medio de comunicación y autonomía energética

Las RIS como se menciona anteriormente tienden más al uso del espectro radioeléctrico, y las zonas consideradas ideales para su aplicación en el muestreo de datos tienden ser áreas donde no hay posibilidad de instalar una infraestructura debido a factores diversos como lo puede ser el costo de la instalación o incluso la dificultad para poder tener acceso el instalador. La RF es uno de los medios con mayor complejidad debido a todas las consideraciones que deben tomarse en cuenta. Una de ellas y probablemente la que más afecta al desarrollo de las RIS es todo aquello que concierne a aquello que puede debilitar o disminuir la señal. La propagación de ondas electromagnéticas sigue una ley inversa-cuadrada debido a la dispersión de la energía. En la ecuación (2.1) se presenta el caso de un radiador isotrópico. P_D es la densidad de la potencia en watts por metro cuadrado, P_t es la potencia total del transmisor en watts, y r es la distancia desde la antena medida en metros.

$$P_D = \frac{P_t}{4\pi r^2} \tag{2.1}$$

Mediante esta ecuación se puede observar que es posible demostrar que mientras más lejos se quiera hacer llegar una señal de radiofrecuencia, se requiere una mayor cantidad de energía. Considerando el efecto de la antena transmisora y la receptora, se llega a la ecuación (2.2), que es la *ecuación de transmisión de Friis* [12], representando así la relación de la potencia del receptor contra la del transmisor (P_t y P_R respectivamente), considerando las ganancias de las antenas del transmisor

y del receptor (G_T y G_R respectivamente), la longitud de onda (λ), y la distancia entre las dos antenas en metros (r).

$$\frac{P_R}{P_T} = \frac{\lambda^2 G_T G_R}{16\pi^2 r^2} \tag{2.2}$$

En el campo de las telecomunicaciones, las potencias de transmisión en corto alcance son expresadas en una escala logarítmica en dBmW (usualmente referenciados a una impedancia de 50Ω). La distancia entre las antenas (r) se acostumbra expresarlas en kilómetros, y las ganancias en dBi, que es una relación adimensional, en la que la i indica que se refiere a un radiador isotrópico. En lugar de una longitud de onda λ, se acostumbra utilizar la frecuencia *f* en MHz [13]. Y por ello se utiliza la ecuación (2.3) con más regularidad, siendo esta un despeje de forma logarítmica.

$$\frac{P_R}{P_T} = G_T(dBi) + G_R(dBi) - (32.44 + 20 * \log(d) + 20 * \log(f)) \tag{2.3}$$

Del último término de esta ecuación se deriva la ecuación de *pérdida de espacio libre* (ecuación (2.4)), que es el negativo de la ecuación anterior, y representa la pérdida en dBm que causa la difusión de una señal electromagnética en el aire.

$$L_{fs} = 32.45 + 20 * \log(d) + 20 * \log(f) \tag{2.4}$$

La cuantificación de la potencia que llega al receptor (PR), a través de todas las perdidas (L_{xx}) y ganancias (G_{xx}) causadas por cableado, antenas, la pérdida de espacio libre, etc. se utiliza mediante la ecuación (2.5) que es la ecuación de *presupuesto de enlace,* se considera una antena transmitiendo con una potencia conocida como P_R. Se considera de igual manera una pérdida miscelánea para los fenómenos que tienen poco impacto.

$$P_R = P_T + G_T - L_T - L_{fs} - L_M + G_R - L_R \tag{2.5}$$

Cuando la sensibilidad de un radio (medida en negativo) es mejor que el presupuesto de enlace, representa que los radios pueden comunicarse entre sí. El

margen de atenuación es un límite de potencia que compensa los cambios provocados por el clima, y otros eventos electromagnéticos.

2.4 Autonomía energética y baterías

Otro aspecto que se vuelve una de las metas principales para las RIS es la autonomía energética debido a que el tiempo de vida del nodo se determina por el uso óptimo de la batería que lo alimenta. Para ello es imprescindible que los nodos cumplan ciclos de actividad e inactividad para disminuir el consumo energético apagando funciones que no se estén utilizando como la lectura de sensores o el radio. Por lapsos de tiempo determinados los nodos escuchan el canal RF para detectar si existe algún mensaje para ser entregado y entrar en estado de reposo, aunque también es posible aumentar la latencia aumentando el consumo energético. Los nodos usualmente se alimentan de baterías para su desempeño. Las baterías son acumuladores de energía química que la convierten en corriente eléctrica. Las baterías primarias son aquellas que producen una energía de un proceso, generalmente electroquímico de forma irreversible. Las baterías secundarias son aquellas que absorben la energía eléctrica y la pueden entregar varias veces mediante la restauración de su carga mediante corriente inversa.

Algunas definiciones para la comprensión detallada de una batería se exponen a continuación:

Voltaje: En las terminales de la batería habrá una diferencia de potencial que puede ser utilizada para generar una corriente eléctrica. Este voltaje estará determinado por las celdas químicas que compongan a la batería.

Capacidad: Es la carga eléctrica total que puede almacenar una batería. Por conveniencia, se acostumbra medirla en Amperes-Hora (Ah) en vez de Coloumbs (C). De acuerdo a la definición de Ampere del SI, un ampere equivale a que circule 1 Coulomb durante 1 segundo. Por lo tanto, 1 Ah equivale a una carga de 3600 C.

Energía Específica: La energía se mide en Watts-Hora (Wh) en lugar de Joules. La energía específica habla de la cantidad de energía que tiene la batería con respecto a su masa. Es la energía por unidad de peso (Wh/Kg), y la densidad de energía es

la energía por unidad de volumen (Wh/l). Estas dos medidas de energía sirven para determinar qué tanta energía otorgará una batería de un tamaño dado.

Potencia Específica: La potencia específica determina la cantidad máxima de corriente (en A) que puede entregar la batería en cualquier momento dado. La energía específica no afecta a la potencia específica. Para aumentar la potencia específica se pueden colocar varias celdas electroquímicas en paralelo.

Consumo Energético: El consumo total realizado por un circuito será la integración de la corriente en función del tiempo en un periodo de tiempo dado de t1 a t2 como se muestra en la siguiente fórmula.

$$QT = \int_{t1}^{t2} i(t) * \Delta t$$

En sistemas reales, como las RIS, es difícil obtener o predecir la corriente en función del tiempo. Pero se pueden realizar estimados del consumo utilizando esta simplificación con cada uno de los componentes de corriente del sistema, multiplicar por el periodo durante el que se utilizan, y realizar la sumatoria de todos los componentes. Al realizar esto en Amperes- Hora, se tendrá el consumo de energía que realizó el sistema. Este consumo se resta de la capacidad total de la batería.

El consumo real de un dispositivo eléctrico se mide en Watts, donde 1 Watt equivale a 1 Joule consumido en 1 segundo. De la fórmula de potencia se llega a la ecuación 1.10, que permite convertir los Watts a la unidad que se usa para medir la capacidad de una batería (Ah).

2.5 Potencia de la señal RSSI como indicador para estudio de distancias

Los métodos de localización basados en la potencia de la señal recibida (RSSI) son los más frecuentemente empleados en tecnologías RIS y Wi-Fi [14]. En las tarjetas adquisitoras de datos utilizadas para las RIS se emplea una señal que se manda de nodo emisor a nodo receptor haciendo uso de un indicador RSSI para lograr estimar la distancia entre ellos. Una de las ventajas de usar este método es el hecho de que las medidas de RSSI se obtienen de forma directa por los módulos de

radio sin la necesidad de componentes adicionales. El dBm es una unidad de potencia expresada en decibelios (dB) relativa a un mili vatio (mWatt) utilizada para expresar la medida absoluta de potencia siendo una unidad adimensional cuyo objetivo es la cuantificación de datos [15].

El valor del indicador de fuerza RSSI puede describirse de la siguiente manera:

- Señal débil y con ruido regresará un bajo nivel de RSSI.
- Señal fuerte y sin ruido regresará un alto nivel de RSSI

Para los nodos basados en la tarjeta Freakduino el valor del indicador RSSI será un entero entre 0 a 84 que es el máximo radio que puede manejar según Freaklabs sin llegar a saturarse. Para calcular los dB del valor RSSI se utiliza la siguiente formula:

$$\mathrm{P[dBm]} = (\mathrm{Valor\_base\_RSSI}) + (1.03 * \mathrm{Nivel\_RSSI}) \quad (2.6)$$

Donde:

P[dBm] = Potencia en dBm

Valor_base_RSSI = -98 (modulación OQPSK a 250kbps)

Nivel_RSSI = Nivel obtenido de la tarjeta.

El valor de base se obtiene de la página oficial del fabricante, siendo el parámetro que se utilizará para la configuración del nodo programando el comando correspondiente en la librería y con esto entonces el máximo valor RSSI en dBm sería -11.48 dBm y -96.97 dBm.

La librería creada por Freaklabs para la programación y configuración de la tarjeta Freakduino es un fichero denominado ‘chibiUsrCfg.h’ la cual tiene funciones diversas que permite al usuario realizar cambios a la configuración que logre adaptarse a la aplicación donde se implementa la tarjeta. Freakduino maneja los cambios en su configuración permitida por el protocolo IEEE 802.15.4.

2.6 Estructura de nodos

Típicamente, un nodo sensor es un dispositivo pequeño que incluye tres componentes importantes: un subsistema sensorial para la adquisición de datos del entorno físico, un subsistema de para datos locales y almacenamiento, y por último un subsistema de comunicación inalámbrica para la transmisión de datos [16]. De forma adicional e importante, una fuente de alimentación es requerida para energizar al dispositivo para ejecutar la tarea programada. En la figura 2.3 se muestra la estructura principal de un nodo sensor.

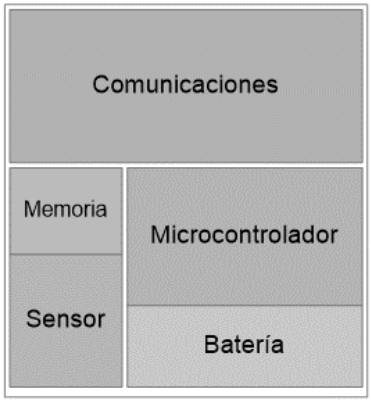


Figura 2.3. Estructura principal de un nodo sensor [16].

Se puede decir en forma general que la estructura de los nodos sensores según Chio Cho Nayibe [17] y Jun Zheng [18] se componen de 4 unidades básicas que permiten su funcionamiento.

- Unidad de sensado. Esta unidad consiste de un sensor que se encarga de medir la variable física, biológica y química del objeto o medio.
- Unidad de procesado. Esta unidad consiste ya sea de un microcontrolador o de un microprocesador según las características de la red, además del control de tiempos y protocolo de comunicación de la red.
- Unidad de comunicación. Es la unidad principal, es la parte en la que se maneja la recepción y transmisión de los datos en la red. La transferencia de datos se da en forma inalámbrica en la red.

- Unidad de alimentación. Consiste de una batería o de un sistema que genera o captura energía para poder alimentar a todo el nodo.

Tipos de nodos

- Nodo sensor: es aquel que contiene la electrónica. La función principal de éste es monitorear por medio de sensores las variables de interés y transmitirla a la red.
- Nodo retransmisor: se encarga de recibir y transmitir datos en la red.
- Nodo coordinador: es el depósito de los datos, es la principal fuente de almacenamiento de datos en red, en ella se pueden visualizar los datos a través de una interfaz gráfica para el usuario.
- Región de interés: es el área en la cual está toda la red y que contiene las variables a sensar.

Para muchas de las aplicaciones de redes inalámbricas es importante conocer la ubicación de los nodos sensores en la red [16]. Esto debido a que esta información proveniente de los sensores es de utilidad, solo si la información de los nodos se encuentra disponible para su transmisión tomando en cuenta el tipo de protocolo de comunicación que le permita las rutas viables posibles.

2.7 Aplicaciones agronómicas de las RIS

Entre diversas aplicaciones de las RIS, en el área de la agricultura, su uso favorece en gran medida al aprovechamiento de recursos como lo son el agua o el uso de pesticidas, para poder así preservar mucho mejor el entorno de los cultivos. La agricultura de precisión cubre múltiples prácticas relativas a la gestión de cultivos y cosechas, árboles, flores y plantas, ganado, etc. [19]. El uso de las RIS como tecnología favorece la optimización de recursos en la agricultura mediante el control de un sistema que monitorice el riego o posibles fallos de inundación por ruptura de tubos e incluso la programación estadística de un servidor de datos. Las RIS son una solución cuya demanda va en aumento en todo el mundo debido a su amplia aplicación con la implementación de sensores que pueden monitorizar el estado del clima, la humedad, la luz e incluso la calidad del aire.

En el monitoreo de amplias áreas de tierras de agricultura o forestales, una estrategia bien desarrollada puede mantener un desempeño y exactitud en el monitoreo utilizando menos nodos en una RIS, que puede reducir el costo de toda una red cuando se utilizan nodos costosos [20]. Por el hecho de que son sistemas que deben tener una durabilidad en el medio externo deben considerarse soluciones energéticas para la alimentación del nodo.

2.8 Nodo Sensor

2.8.1Tarjeta adquisitora Freakduino-900 MHz v3.0a y v2.1a

La tarjeta Freakduino-900 MHz v3.0a de la empresa FreakLabs, está diseñada para un rápido prototipado, experimentado y desarrollo de diseños inalámbricos a bajo costo, además de adquirir los conocimientos básicos de las redes de sensores inalámbricos sin tener que lidiar con herramientas muy complejas, protocolos y software avanzado. Combina la facilidad del entorno de programación Arduino, la compatibilidad con gran cantidad de periféricos, e integra redes de radio para un sistema de prototipado inalámbrico [21].

La Freakduino versión 3.0a mejora el diseño de la versión 2.1a con la adición de nuevas características para una reducción del voltaje de operación, encriptación AES-128, hardware basado en generador de números y modos especiales de transferencia de datos posibles de alcanzar velocidades de hasta 1Mbps. Muchos de los componentes fueron reemplazados para la optimización de consumo eléctrico por lo que ahora puede operar con menos de 200 uA en modo sleep con dos baterías alcalinas comunes en lugar de 300 uA con respecto a la versión 2.1a. En ambos casos utilizando el voltaje de 3.3 volts que provee la tarjeta. También tiene accesorios opcionales como un circuito regulador para baterías, un porta pilas que puede contener dos pilas AA en la base inferior, y una caja resistente. En la figura 2.4 se muestra la tarjeta Freakduino de vista superior e inferior. En el anexo A se muestra el esquemático de la Freakduino.

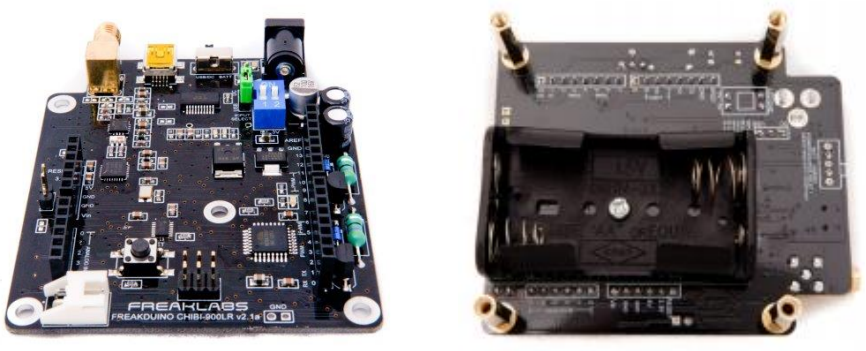


Figura 2.4. Freakduino 900 MHz v3.0a parte superior e inferior.

La tarjeta Freakduino es totalmente compatible con el entorno de programación Arduino, el cual ofrece una interfaz de usuario simple que integra las librerías de Arduino y las librerías para Freakduino.

La integración de un radio inalámbrico basado en el protocolo IEEE 802.15.4 (el mismo protocolo que XBee) permite un control sobre los dispositivos inalámbricos o la colección de datos proveniente de los sensores.

2.8.2 Características de Freakduino-900 v3.0a y 2.1a

A continuación, se muestran algunas de las características especiales que provee esta tarjeta mediante la figura 2.5 y la información de la página oficial en la tabla 2.1.

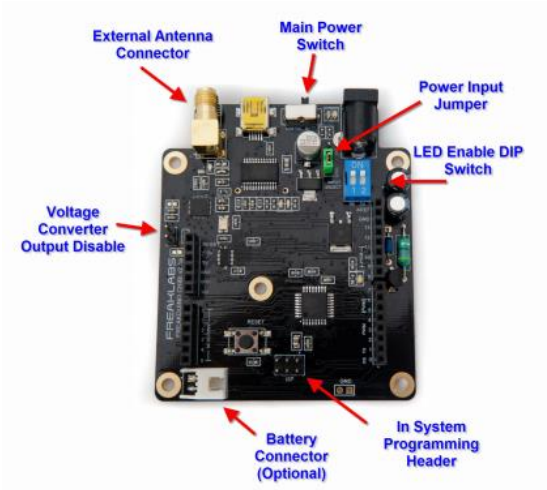


Figura 2.5. Freakduino 900 MHz v3.0a características.

Tabla 2.1. Ficha Técnica de Freakduino-900 v3.0A de la página oficial.

Características	Valores
Microcontrolador	ATmega 328P 32 KB de memoria Flash 2 KB de RAM 1 KB de EEPROM
Velocidad de reloj	16 MHz
Voltaje de operación	5 V
Pines E/S digitales	54
Pines de entrada análogos	16
Voltaje de alimentación	5V (vía USB opcional)
Consumo de corriente	Sleep: 250uA Normal: 40 mA
Dimensiones	68.5x53.4 mm
Comunicación	802.15.4 USB inalámbrico
Tarjeta integrada	Contiene los componentes que integran Arduino con la comunicación inalámbrica
900 MHz +2 dBi de Antena:	Este es el estándar de 900 MHz de cobertura y mejora el rango y la calidad de la señal.
Regulador de batería	: Permite alimentarse por USB o mediante un jack. Viene con los componentes requeridos para implementar una batería reguladora de voltaje que mantiene 5V de salida constante. El voltaje de entrada puede variar entre 0.7V a 6V. La máxima corriente de salida del circuito regulador de corriente es de 200mA.

Compatibilidad SPI y el protocolo Freakduino Chibi

La tarjeta Freakduino cuenta con un protocolo denominado FREAKDUINO-CHIBI, que es esencialmente un sistema basado en Arduino y un circuito integrado denominado *shield* [22]. Lo cual significa que el periférico requerirá solamente usar algunos pines dedicados para la funcionalidad *wireless*. Dos pines en el conector de entrada análogo no están disponibles más que para uso de los pines de entrada/salida digital. Estos son los pines análogos 2 y 3 que sirve para controlar el modo *sleep* y para controlar la selección de comunicación entre el microcontrolador y el radio IC respectivamente. Si la funcionalidad *wireless* no es usada, estos pines son usados como pines digitales estándar de entrada/salida, pero no como entradas análogas. Es recomendable evitar el uso de estos pines.

Cuando la funcionalidad *wireless* se encuentra en uso, el bus del SPI (*Serial peripheral Interface*) es también requerido para comunicación con radio. Lo cual significa que los pines digitales 10-12 (PB3 a PB5 o MOSI, MISO, y SCLK) serán dedicados a ser pines de SPI. Debe tenerse especial cuidado cuando se haga uso del led auxiliar en la tarjeta. Se encuentra localizado en el pin SCLK requerido para el SPI.

2.9 Convertidor Analógico Digital de Freakduino

Las lecturas de las señales provenientes de variables físicas del entorno natural son analógicas y por lo tanto para poder representarlas en un instrumento, es esencial su conversión de analógicas a digitales. Los convertidores analógicos-digitales (ADC) convierten una señal analógica con un tamaño/amplitud que varía continuamente, en una forma digital que es un número discreto de niveles [23].

Un ADC es en esencia un comparador entre el voltaje de la entrada análoga contra otro voltaje que es presentado como una entrada de referencia. La referencia análoga es considerada como el más alto voltaje que una señal presentará en la entrada analógica. Esto no implica que si se recibe un voltaje mayor en la entrada análoga el convertidor se dañará, simplemente aquellos valores fuera del voltaje de referencia se expresarán con el máximo valor límite.

El microcontrolador de la tarjeta Freakduino contiene internamente un conversor analógico digital de 6 canales, el cual otorga una resolución de 10 bits, devolviendo enteros de 0 y 1023 teniendo por cada valor 4.882 mV si se usa el voltaje de referencia de 5 V, pero teniendo un aumento en la resolución. La tarjeta tiene dos voltajes, 5 y 3.3 V con los que opera internamente y a su vez utilizar de forma externa. Si se utiliza 3.3 V, se tiene como resultado 3.3/1024=3.22mV. Para la lectura de señales analógicas provenientes de sensores es necesaria la transcripción de señales analógicas a señales digitales para facilitar su procesamiento, ya sea codificación, compresión, etc. y hacer la señal resultante (digital) más inmune al ruido y otras interferencias a las que son más sensibles las señales analógicas.

En el proyecto de Flores Medina se concluyó que para el análisis de datos en cuanto al monitoreo de humedad se puede mejorar la resolución de la adquisición de datos del ADC de 10 bits que contiene la tarjeta Waspmote mediante la implementación de un ADC cuyo modelo es MCP3208 [24], el cual es un convertidor analógico digital de 8 canales con 12 bits que otorga enteros de 0 a 4096 teniendo como resultado una resolución de 1.22 mV usando el voltaje de 5V, o una resolución de 0.805 mV en el caso de utilizar el voltaje de 3.3V. Su interfaz utiliza el estándar de comunicaciones SPI, el cual es utilizado para la transferencia de información entre circuitos integrados.

2.10 Sensores

2.10.1 Sensor VH400

El sensor VH400 (Figura 2.6) determina el contenido volumétrico del agua en el suelo (VWC o Θv). Sus principales aplicaciones se encuentran en los sistemas de irrigación, monitoreo del medio ambiente, clima y las lluvias.

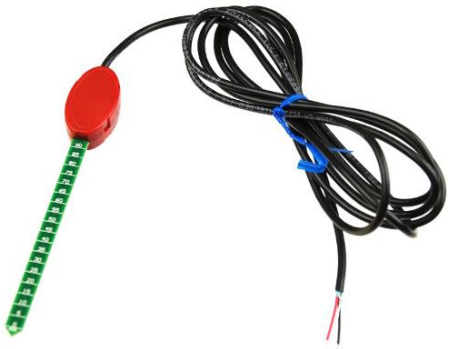


Figura 2.6. Sensor VH400 [25].

El principio con el que trabaja es el de detectar los cambios en la humedad, tomando lecturas rápidas y en tiempo real. Es de fácil instalación, bajo costo y consumo energético además de no afectar el área que muestrea [25]. Está enfocado a aplicaciones del tipo agrícola para la medición de humedad debido a los materiales con los que está diseñado para ser resistente al agua e incluso ser enterrado bajo tierra completamente.

Las especificaciones técnicas del sensor se muestran en la tabla 2.2 y su diagrama se muestra en la figura 2.7.

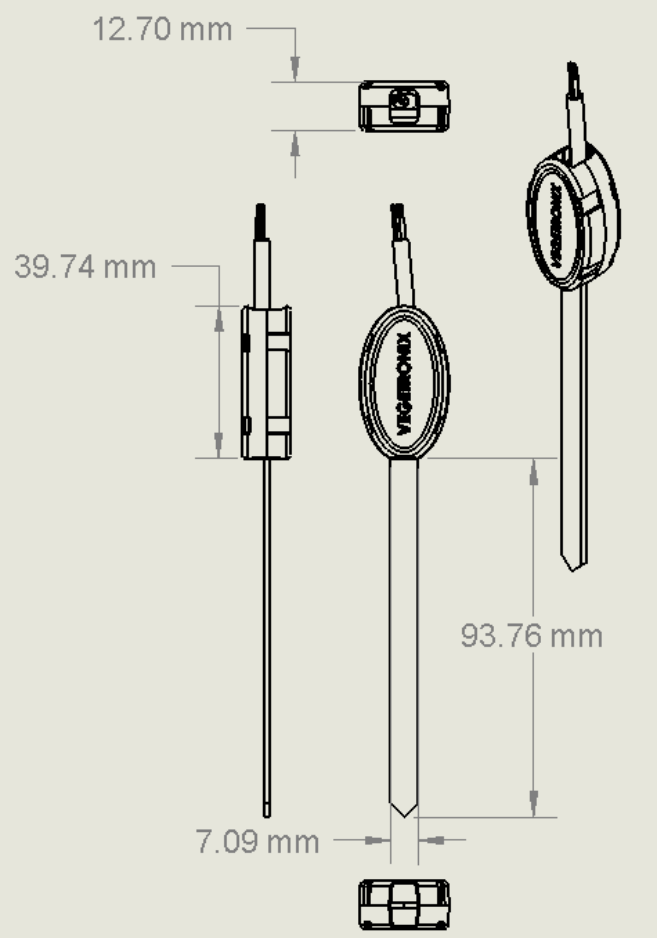


Figura 2.7. Esquema del Sensor VH400 [24].

Tabla 2.2. Especificaciones del sensor VH400.

Sensor de humedad VH400	
Consumo de potencia	7mA
Voltaje de alimentación	3.3 V a 20 VDC
Consumo de salida estable	400 ms
Impedancia de salida	10K ohm
Temperatura de operación	-40°C a 85°C
Precisión a 25°C	2%
Salida	0 - 3V relacionado al VWC

El fabricante provee al usuario de 5 ecuaciones para lograr linealizar la respuesta de salida, estas ecuaciones están contenidas en la tabla 2.3. Aunque la mayoría de los suelos tienen una capacidad de retención de menos del 50%, el fabricante otorga una ecuación que es una aproximación para esos casos, la cual es la quinta ecuación de la tabla de ecuaciones.

Tabla 2.3. Ecuaciones del fabricante del sensor VH400.

Rangos de voltaje de entrada	Ecuación	VWC
0 V a 1.1 V	VWC= 10*V-1	0-10
1.1 V a 1.3 V	VWC= 25*V- 17.5	10-15
1.3 V a 1.82 V	VWC= 48.08*V- 47.5	15-40
1.82 V a 2.2 V	VWC= 26.32*V- 7.89	40-50
2.2 V a 3.0 V	VWC= 62.5*V - 87.5	50-100

2.10.2 Sensor Lm35dz

El LM35 es un sensor de temperatura analógico, con una precisión calibrada de 1ºC. No necesita circuitería externa ni ajustes de calibración para funcionar, su rango de medición abarca desde -55°C hasta 150°C. La salida es lineal y cada grado centígrado equivale a 10mV, el empaquetado de este sensor es *through-hole* TO-92 [26]. En la figura 2.8, se muestra el sensor.

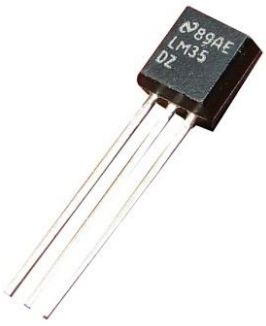


Figura 2.8. Sensor lm35dz.

Es ideal para aplicaciones remotas ya que consume menos de 60 mA de corriente. Es compatible con Arduino, PIC, EMBED, AVR o cualquier microcontrolador. Algunas características principales del modelo lm35dz son:

Especificaciones:
-Calibrado en centígrados
-Factor de escala lineal 10.0 mV/°C
-Rango de medición de ?55° a +150°C
-Ideal para aplicaciones remotas
-Bajo costo
-Funciona de 4-30 V
-Consumo menor a 60 uA
-Baja impedancia

2.10.3 Sensor DHT11

El sensor DHT11 (figura 2.9) es capaz de realizar mediciones simultáneas de humedad relativa y temperatura**,** entregando su lectura de forma digital. Es posible encontrar aplicaciones para el DHT11 en el control de invernaderos, monitoreo de centros de datos, climatización de casas y edificios, etc.

Comunicación del DHT22 o DHT11 con Arduino

El DHT11 no utiliza una interfaz serial estándar como I2C, SPI o 1Wire (es similar a este último). En cambio, requiere su propio protocolo para comunicarse a través de un solo hilo. El protocolo de comunicación del DHT11 es simple y puede implementarse usando los pines de I/O en un Arduino [27].

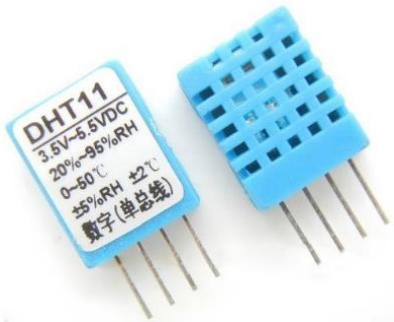


Figura 2.9. Sensor DHT11 [27].

El Arduino inicia la comunicación con el DHT11 manteniendo la línea de datos en estado bajo durante al menos 18 ms. Luego el DHT11 envía una respuesta con un pulso a nivel bajo (para indicar su presencia) de 80 uS y luego deja “flotar” la línea de datos por otros 80 uS. En la figura 2.10, el pulso de inicio enviado por el microcontrolador esta coloreado en rojo, mientras que la respuesta desde el sensor esta coloreada en azul.

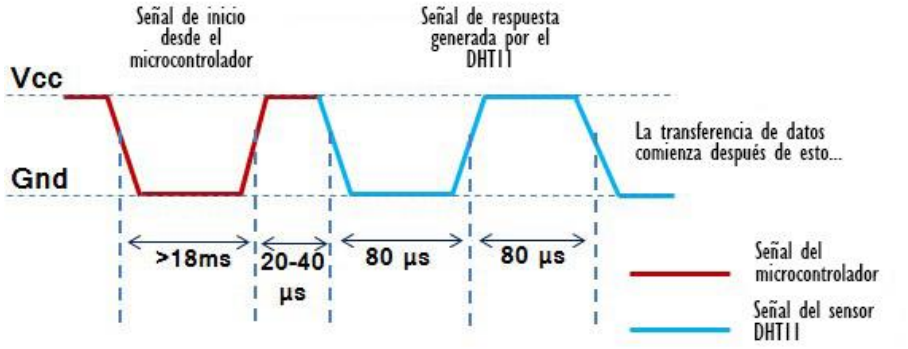


Figura 2.10. Señales del DHT11 [27].

Codificación de bits

La codificación de datos está basada en un esquema de ancho de pulso (se toma en cuenta el ancho del estado alto): Un pulso ancho representa un 1 lógico, un pulso corto representa un 0 lógico.

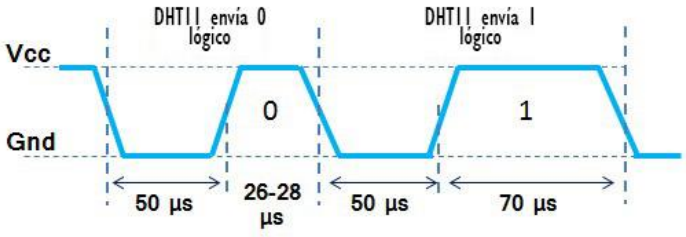


Figura 2.11. Codificación de bits del sensor DHT11 [27].

Todos los bits comienzan con un pulso bajo de 50 uS. Las librerías de comunicación con el DHT11 aprovechan este pulso para la sincronización. Luego

viene un pulso alto que varía según el estado lógico o el valor del bit que el DHT11 desea transmitir:

- Se utilizan pulsos de 26-28 microsegundos para un “0”.
- Se utilizan pulsos de 70 microsegundos para un “1”.

Significado de los bits transmitidos por DHT11

En la figura 2.12, se observa el inicio de una comunicación con el DHT11.

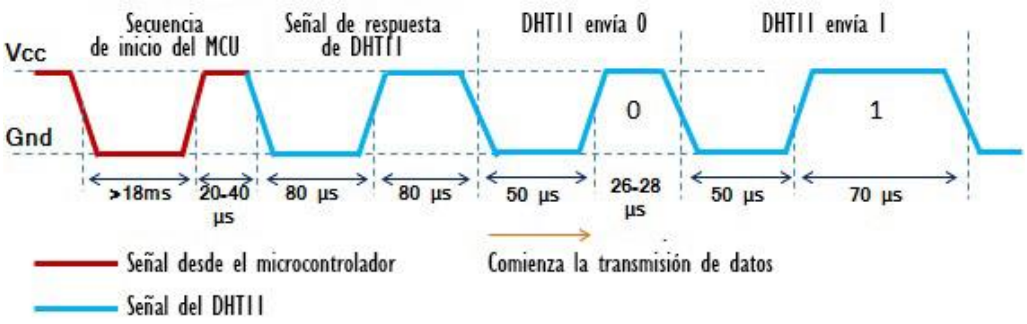


Figura 2.12. Transmisión de bits del sensor DHT11 [27].

Una transmisión completa se compone de 40 bits (5 bytes) que incluyen todos los datos que el sensor puede proporcionar. En cuanto a los datos que se transmiten, su interpretación es como sigue:

- El primer byte que recibimos es la parte entera de la humedad relativa (RH)
- El segundo byte es la parte decimal de la humedad relativa (no se utiliza en el DHT11, siempre es 0).
- El tercer byte es la parte entera de la temperatura.
- El cuarto byte es la parte decimal de la temperatura (no se utiliza en el DHT11, siempre es 0).
- El último byte es la suma de comprobación (*checksum*), resultante de sumar todos los bytes anteriores.

2.10.4 Sensor FC-28

El sensor de humedad FC-28 se basa en un divisor de voltaje ajustable mediante un potenciómetro [28]. El sensor para detectar la humedad en el suelo que se muestra en la figura 2.13, funciona de la siguiente forma: Cuando hay deficiencia de agua en el suelo, la salida del módulo emite un nivel alto, mientras que si el nivel está bajo emite la salida en bajo.

Figura 2.13. Sensor FC-28.

Según el fabricante [29]:

- El uso de este sensor para regar plantas lo hace un dispositivo autónomo que avisa al usuario cuando debe administrar el uso del agua en ellas.
- La sensibilidad se puede ajustar (con el potenciómetro digital que se ve en la figura 2.13).
- Se arregló el agujero del tornillo, para una instalación conveniente.
- Comparador LM393 chip, estable.
- Especificaciones de interfaz de placa (3 cables).
- VCC externo 3.3~5 V.
- GND externo GND.
- DO Salida digital (0 y 1).
- La salida digital DO puede ser conectada directamente con un microcontrolador con el cual detecte los niveles altos y bajos para la monitorear la humedad del suelo.

2.10.5 Sensor FZ-0430

El FZ0430 es un módulo comercial que permite medir tensiones de hasta 25V de forma sencilla con un procesador como Arduino [30]. En la tabla 2.4 se muestran las características del sensor FZ0430.

Tabla 2.4. Características del sensor FZ0430.

Sensor de Tensión FZ0430	
Voltaje de entrada	0 – 25VDC
Detección de voltaje	0.02445 – 25 VCD
Voltaje de salida	0 – 5 VDC
Dimensiones	25 mm x 13 mm
Resolución normal	4.88 mV

Es un divisor de tensión con resistencias de 30 kohm y 7.5k Ohm, lo que supone que la tensión percibida tras el módulo sea de divida por un factor de 5 (7.5/ (30+7.5)). La tensión máxima que puede medir es 25V para un procesador de tensión de alimentación Vcc 5V, y 16.5V para un procesador de Vcc 3.3V. Superar esta tensión en el input del FZ0430 dañará el pin analógico de Arduino. Esta ampliación del rango de medición tiene una consecuencia negativa en la precisión de la medición. En los modelos de Arduino que incorporan un ADC de 10 bits alimentados a 5V, la resolución normal es de 4.88mV. Tras el FZ0430 la resolución de la medición es de 24.41mV. En caso de emplear una carga de 25V, la corriente que atraviesa el divisor es de 0.7mA, y las pérdidas del divisor 16.67mW. La figura 2.14 muestra su modo de conexión.

Figura 2.14. Sensor de voltaje FZ0430.

Este módulo incorpora clemas de conexión y terminales para conectar de forma sencilla y rápida a Arduino.

2.10.6 Sensor ACS712 [31].

Es una solución económica para medir corriente, internamente trabaja con un sensor de efecto Hall que detecta el campo magnético que se produce por inducción de la corriente que circula por la línea que se está midiendo.

El ACS712 (figura 2.15), se encuentra implementado en un módulo que facilita la conexión del mismo, teniendo un bloque terminal para conectar la línea que se quiere medir y 3 pines, dos para conectar la alimentación y un pin para la salida analógica.

Figura 2.15. Sensor de corriente ACS712.

La corriente aplicada que fluye a través de esta pista de conducción de cobre genera un campo magnético que es detectado por el IC integrado *Hall* y convertida en una tensión proporcional. La exactitud de los dispositivos se optimiza a través de la proximidad de la señal magnética al transductor *Hall*. Una tensión precisa, proporcional es proporcionada por el bajo offset, chopper-estabilizado IC BiCMOS *Hall*, que está programado para la exactitud después del encapsulado.

Este sensor se encuentra en una pequeña placa, que soporta un sensor de corriente de *Allegro*, está basado en el efecto *Hall* lineal, dicha relación es una línea recta en una gráfica Voltaje y Corriente donde la pendiente es la sensibilidad y la intersección en el eje Y es 2.5 voltios. La ecuación de la recta seria la siguiente:

$$V = mI + 2.5 \tag{2.8}$$

Donde la pendiente es *m* y equivale a la sensibilidad.

Despejando tendremos la ecuación para hallar la corriente a partir de la lectura del sensor:

$$I = (V - 2.5)/\text{sensibilidad} \tag{2.9}$$

2.11 Pila de protocolo de comunicación inalámbrica de Freakduino

La pila de protocolos de chibiArduino es un puerto de la pila inalámbrica de la plataforma Arduino Chibi 802.15.4. y depende principalmente de tres funciones principales: inicialización, enviar, y recibir. Otras funciones disponibles para gestión o configuración, sin embargo, solo con las funciones básicas de inicialización y las configuraciones por default pueden permitir a las personas a empezar la comunicación inalámbrica. El *hardware* del radio se encuentra ya integrado en diversos circuitos integrados inalámbricos incluyendo la funcionalidad de manejar características como reintentos y tiempos muertos, simplificando las comunicaciones además de ejecutarse sin la necesidad de un sistema operativo o una compleja máquina de estados. Simplifica de gran manera la implementación de la pila de protocolos disminuyendo un cumulo de recursos del sistema como la memoria Flash y la memoria RAM. El *software* de también integra un comando de línea el cual se puede ser usado dentro del sketch de Arduino. Los comandos de usuario pueden ser integrados en el comando de línea para llamar funciones de usuario interactivamente desde una terminal. Esto resulta útil para configurar la dirección o el canal del nodo. Sirve también como una herramienta para controlar los datos que están siendo enviados y ver los datos recibidos desde el nodo.

2.12 Topologías

Para la comunicación en las RIS se distribuyen los nodos de diferentes formas en busca de la mejor comunicación del envío y recepción de información. Estas

agrupaciones se denominan topologías de la red y sus diagramas se muestran en la figura 2.16.

Las principales topologías para la RIS son [16]:

- Topología Estrella. Se da cuando todos los nodos sensores dirigen directamente los datos sensados al depósito de datos.
- Topología Árbol. Consiste en realizar varios grupos constituidos por nodos sensores llamándose así nodos miembros, cada grupo tiene un nodo maestro que es el que transfiere los datos hacia el depósito de datos.
- Topología Malla. Todos los nodos sensores comunican los datos a sus nodos sensores vecinos y sólo los nodos sensores más cercanos al depósito de datos entregan los datos sensados de la red, lo que hace que la red tenga un menor consumo de energía.

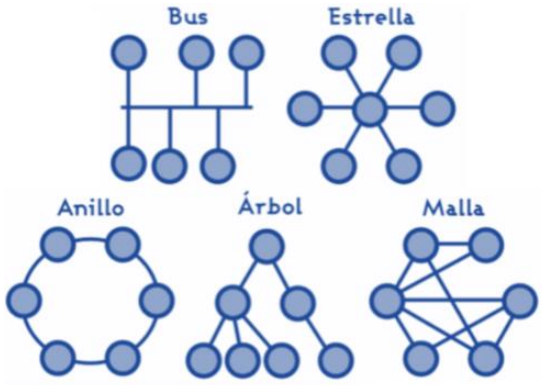


Figura 2.16. Topologías de red [16].

2.13 Protocolos

Existen varios protocolos comerciales que utilizan el protocolo IEEE 802.15.4 como parte de su pila de protocolos, alguno de estos son ZigBee, *WirelessHART, 6LowPan*, cada uno con diferentes características según la aplicación requerida.

Las principales características del protocolo IEEE 802.15.4 son [32]:

- Tasa de transferencia de 250 kbps, 40 kbps y 20 kbps.

- Dos modos de direccionamiento: la corta de 16 bits (usuario) y la larga de 64 bits (dispositivo).
- Opera en tres diferentes bandas de frecuencias libres, acorde al área geográfica donde el sistema es implementado: 915 MHz utilizada en el Norte América y área del pacifico, con una tasa de transferencia de 40 kbps; y por último la de 2.4GHz (ISM *industrial scientific medical*) de uso mundial. Ésta última con 16 canales con una taza de transferencia de 250kbps, un mínimo de 85dBm y un rango de transmisión ideal de 220 metros.
- El estándar IEEE 802.15.4 maneja dos topologías de red: punto a punto y estrella.
- La transmisión de datos es organizada en *frames*, cada capa añade un encabezado y un pie con especificaciones acorde a la capa, los cuales pueden ser diferentes dependiendo del propósito.

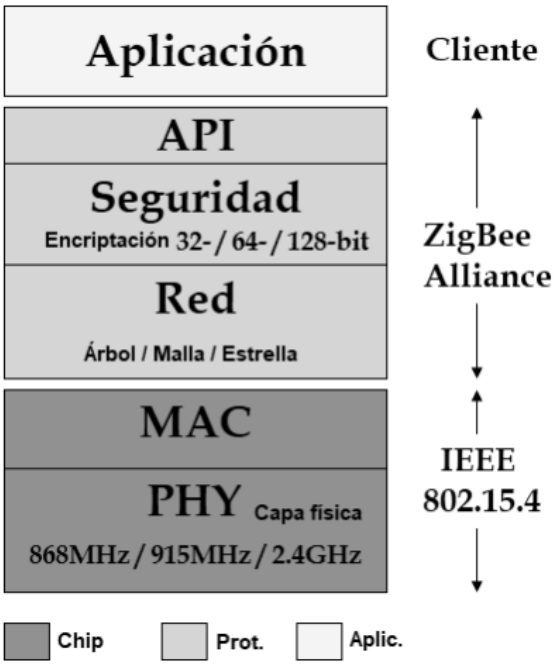


Figura 2.17. Protocolo IEEE 802.15.4 [32].

2.14 Interfaz visual para pruebas con Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible [33]. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Posee una licencia de código abierto, denominada *Python Software Foundation License*,1 que es compatible con la licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

2.15 Computadora embebida Raspberry Pi 3

Raspberry Pi es una computadora del tamaño de una tarjeta de crédito de bajo costo que puede conectarse con un monitor o televisión, y utiliza un teclado estándar y ratón [34]. Diseñada con el último procesador Broadcom 2837 ARMv8 de 64 bits de nueva generación, el modelo B de Raspberry Pi permite a los usuarios aprender lenguajes de programación como Python y realizar diversos proyectos que requieren más capacidad de la que puede otorgar un solo microcontrolador. En la tabla 2.5 se muestran las características de la tarjeta Raspberry Pi 3 modelo B utilizada para este proyecto.

Tabla 2.5. Características de Raspberry Pi 3 Modelo B.

Raspberry Pi 3 Modelo B	
CPU	Quad-Core Cortex A7 a 900MHZ
GPU	VideoCore IV de doble núcleo
RAM	1GB DDR2
Puertos	• 4 x USB 2.0 • 1 x 40 GPIO pin • 1 x HDMI 1.4 • 1 x Ethernet • 1 x Combo audio/mic • 1 x Interfaz de cámara (CSI) • 1 x Interfaz de Pantalla (DSI) • 1 x Micro SD • 1 x Núcleo Grafico 3D • Módulo Bluetooth • Módulo de Wi-Fi b/g/n en la banda de 2.4GHz

CAPÍTULO 3

METODOLOGIA

La realización del proyecto aborda el seguimiento de diferentes fases para lograr completar los objetivos planteados de cada etapa. Se contempla la investigación documental y la elaboración del protocolo de pruebas. En esta fase se determinan los procedimientos y experimentos de la fase de pruebas. La elaboración del presupuesto para el prototipo del diseño lógico no tomo mucho tiempo, debido a que todo se previó en la fase de investigación documental y entrevista con expertos del tema. Una vez adquiridos los materiales, se procede a realizar pruebas de distancias, consumo energético y potencia. El análisis de resultados permite compararlos con los nodos de Waspmote Libelium y Xbee. En la siguiente fase se implementa el circuito ADC, y se analizan entre otros factores, la variación del consumo energético. Luego se procede a la fabricación del prototipo, y se pone a prueba determinar su rendimiento en el campo. Finalmente se hace la presentación de los resultados obtenidos. Para el protocolo de pruebas de la potencia RSSI contra la distancia se siguió un método el cual se expone en la figura 3.1.

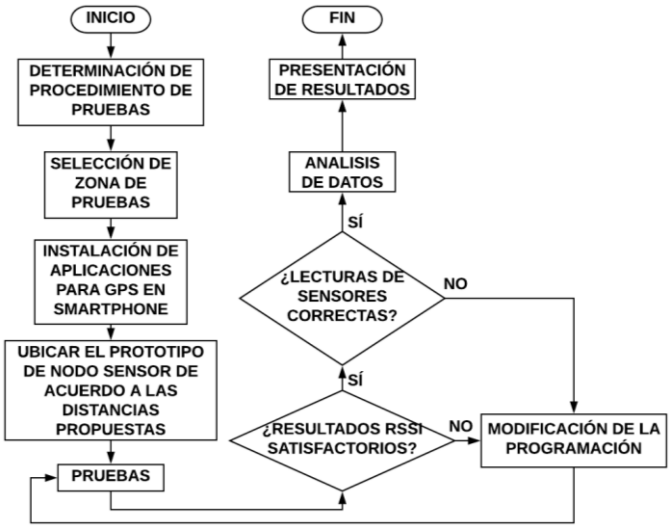


Figura 3.1. Metodología de pruebas de potencia RSSI vs distancia.

3.1 Desarrollo del sistema

Las pruebas de desempeño del nodo basado en la tarjeta Freakduino se realizaron en tres lugares distintos de la ciudad de Torreón, Coahuila, México siendo dos lugares en el Tecnológico Nacional de México, campus Instituto Tecnológico de la Laguna, con domicilio en Blvd. Revolución y Av. Instituto Tecnológico de la laguna s/n, Col. Centro CP 27000, cuyas coordenadas son 25°32′07″N 103°26′06″O, en el Laboratorio de Instrumentación Electrónico (edificio 26) y el área de la cancha de atletismo. El tercer lugar es el bosque Venustiano Carranza, ubicado en las coordenadas 25°32.472'N 103°25.980'O para las pruebas con obstáculos siendo una zona boscosa.

La programación de la tarjeta se hizo por medio del *software* de acceso libre Arduino, y se inició el desarrollo de la interfaz en Python con la implementación de la tarjeta Raspberry Pi. La tarjeta Freakduino está contenida en una caja de uso rudo para que se pueda transportar de forma segura o para soportarla en el medio en donde se requiera instalar sin la necesidad de preocuparse por la circuitería. La forma en comunicar a los nodos será de punto a punto para las diferentes pruebas. Se contempla que la RIS donde se implementen los nodos será con la topología de malla.

3.1.2 Características de la plataforma RIS del proyecto

Teniendo en cuenta las clasificaciones de las RIS, la red actual contará con las siguientes características: Una instalación manual, movilidad nula, su medio de comunicación será por Radio, nodos heterogéneos con topología de malla, cobertura dispersa, conectividad intermitente, un tamaño de red menor a 1024 nodos y un tiempo de vida mínimo de 6 meses.

3.2 Nodo Sensor basado en Freakduino

3.2.1 Electrónica

Implementación de ADC MCP3208 al nodo

La implementación del ADC con la tarjeta Freakduino se configuró como se muestra en la tabla:

Tabla 3.1: Conexión de pines y sus referencias.

No. de pin MCP3208	Referencia	No. de pin digital Freakduino
0-7	Canales del ADC	-
9	DGND	GND
10	Selección de Chip (CS)	10
11	Din MOSI	11
12	Dout MISO	12
13	Reloj (CLK)	13
14	AGND	GND
15	Voltaje de referencia (máx. lectura del ADC	-
16	VDD voltaje de alimentación máximo de 5.5V	-

En la figura 3.2, se presenta el diagrama funcional del MCP3208. En el Anexo C se pueden ver más detalles del ADC MCP3208.

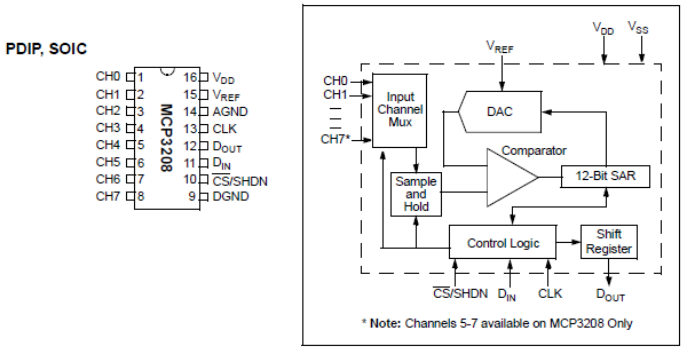


Figura 3.2. Diagrama funcional del ADC MCP3208.

Para realizar la interfaz del ADC con Freakduino se realizaron pruebas para comunicarlos mediante el estándar SPI, una librería propia del MCP y algunos comandos como se muestra en la figura 3.3.

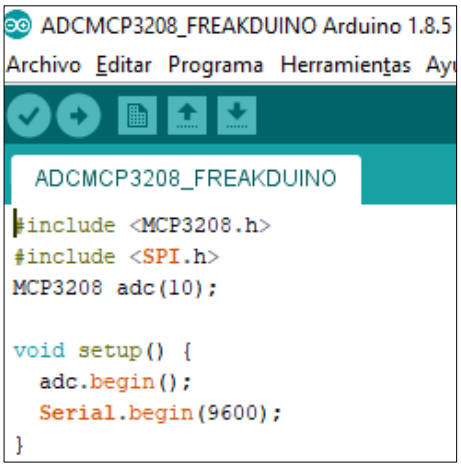


Figura 3.3. Implementación del ADC con la interfaz Arduino.

El programa para probar la conexión con el MCP3208 fue el siguiente:

```
#include <MCP3208.h>
#include <SPI.h>
MCP3208 adc(10);
void setup() {
        adc.begin();
        Serial.begin(9600);}
void loop() {
        char temp[10];
        static int ctr = 0;
        if (ctr == 0) {
                Serial.println();
                Serial.println("              Final único              |    Diferencial");
                Serial.println("    0    1    2    3    4    5    6    7 |    0    1    2    3");
                Serial.println("-----------------------------------------------------------------");}
        ctr++;
        if (ctr == 10) {
                ctr = 0;
        }
        for (int i = 0; i < 8; i++) {
                sprintf(temp, "%5d", adc.analogRead(i))
                Serial.print(temp);
        }
        Serial.print(" |");
        for (int i = 0; i < 4; i++) {
                sprintf(temp, "%5d", adc.analogReadDif(i));
                Serial.print(temp);}
        Serial.println();
delay(100);}
```

El diagrama de conexión con Freakduino se muestra en la figura 3.3. Para más información y los resultados de las pruebas del programa base del ADC se puede consultar el anexo C

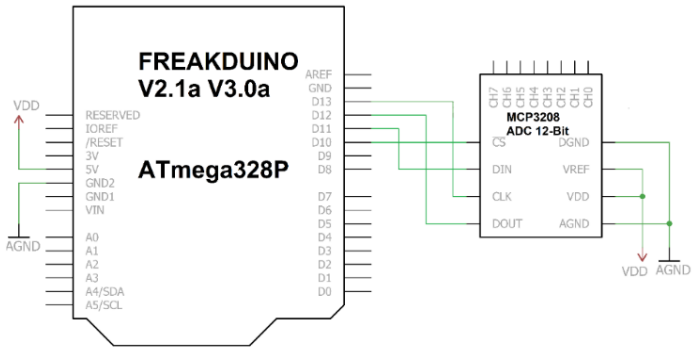


Figura 3.4. Esquema de conexión de Freakduino con ADC MCP3208.

En la figura 3.5 se muestra el MCP3208 conectado a la tarjeta Freakduino para pruebas de comunicación y toma de lecturas.

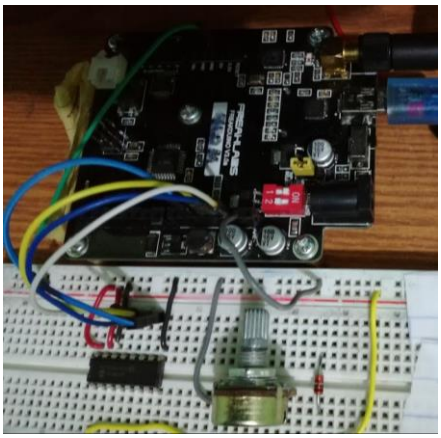


Figura 3.5. ADC MCP3208 con Freakduino.

3.2.2 Sensores para realizar las pruebas de transmisión de datos

Para realizar las primeras pruebas de transmisión de datos vía inalámbrica de nodo transmisor a nodo receptor, se utilizaron diferentes tipos de sensores relacionados con el monitoreo ambiental, siendo estos el lm35dz para medir la

temperatura ambiente, el DHT11, que igual que el anterior mide la temperatura ambiente y la humedad. Finalmente, el FC-28 es un sensor para medir la humedad del suelo, las especificaciones de los sensores fueron consultados en [29] y [35-36].

3.3 Módulo de tiempo real para activar y dormir al nodo

Un módulo RTC (*Real Time Clock*) o "Reloj de tiempo real" fue incorporado al prototipo del nodo sensor y consiste en un circuito integrado alimentado por una batería el cual, en todo momento, registra la fecha, día de la semana y hora al igual que un reloj digital convencional. Estos podrán ser consultados mediante comunicación I2C [37]. Las funciones que sirven para dormir al nodo se explican a detalle en el manual de operación en el anexo B. En la figura 3.6 se aprecia el módulo RTC y en la figura 3.7 se muestra acoplado a la tarjeta Freakduino receptora.

Figura 3.6. Módulo de tiempo Real (RTC 1302).

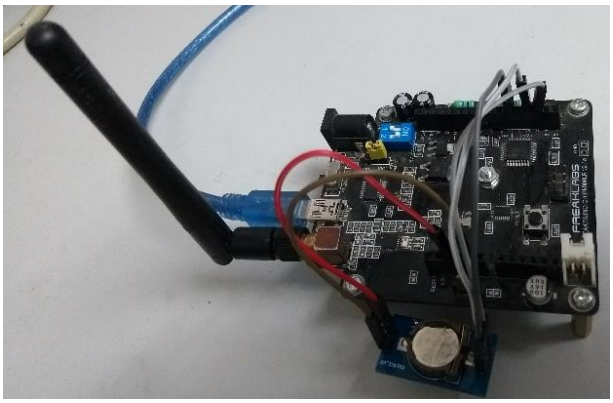


Figura 3.7. RTC acoplado a tarjeta Freakduino.

3.4 Diseño de placa electrónica para nodos

El diseño de la placa del circuito eléctrico para integrar sensores y el ADC, se realizó mediante un *software* de acceso libre denominado Fritzing [38], el cual permitió incorporar en la placa en modo shield, el MCP3208, dos bases para sensores y un módulo RTC. El circuito impreso para la tarjeta Freakduino, se muestra en la figura 3.4.

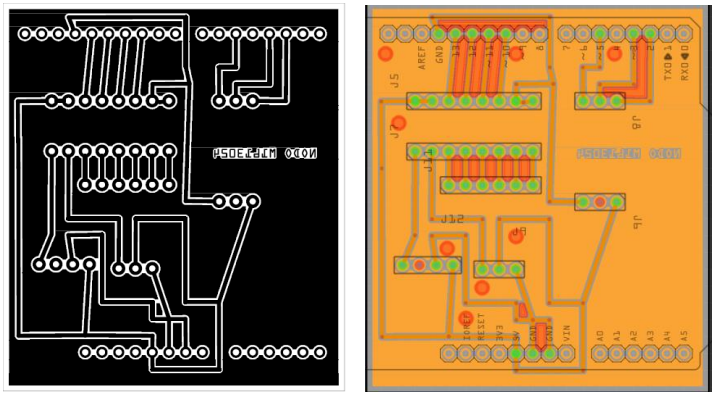


Figura 3.8. Circuito impreso del shield para nodo.

El desarrollo del circuito impreso para el prototipo se muestra en el anexo D.

3.5 Programación de nodos sensores

Para la programación del nodo sensor, se adquirieron conocimientos avanzados de programación del entorno Arduino, con comandos especiales del protocolo usado por la tarjeta Freakduino. Se hizo una investigación sobre el manejo de librerías para sensores, el manejo de la Raspberry pi 3 en el entorno Linux y diseño de interfaces en Python. También, se desarrollaron los programas para realizar las pruebas.

3.5.1 Configuración de la tarjeta Freakduino

La librería creada por Freaklabs para la programación y configuración de la tarjeta Freakduino es un fichero denominado ‘chibiUsrCfg.h’ la cual tiene funciones diversas que permite al usuario realizar cambios a la configuración que logre

adaptarse a la aplicación donde se implementa la tarjeta. Freakduino maneja los cambios en su configuración permitida por el protocolo IEEE 802.15.4.

La tarjeta Freakduino cuenta con las siguientes características:

Frecuencia: 900 MHz
Potencia: 0-84 RSSI o -91 dB a-7dB
Canales: 10

Para la configuración de los nodos sensores se programaron los siguientes parámetros:

Dirección de red:

- Nodo coordinador se estableció 0x0003
- Nodo sensor 1 se estableció 0x0004
- Nodo sensor 2 se estableció 0x0005

Identificador de PAN: Identificador de 16 bits para identificar la red de área personal (PAN) en la que se está trabajando. Todos los nodos deben configurarse con el mismo identificador PAN para que puedan comunicarse entre sí.

- Se estableció 0x1234

Identificador de nodo: Cadena de 20 caracteres para identificar el nodo en la red.

- No se implementó

Canal: Este parámetro establece la frecuencia en la que transmite la red.

- Configurado en el canal 10 en la frecuencia de 868 -915MHz

Máximo de carga útil de datos: Determina el tamaño más largo que puede ser transmitido en un cuadro único. El rango va de 1 a 116 bits.

- Maxpayload: 100 bits

Modulación para tarjeta de 900 MHz

- Se estableció: CHB_900MHZ_INIT_MODE x_SIN_1000

En la figura 3.5, se muestra el diseño lógico del funcionamiento del nodo.

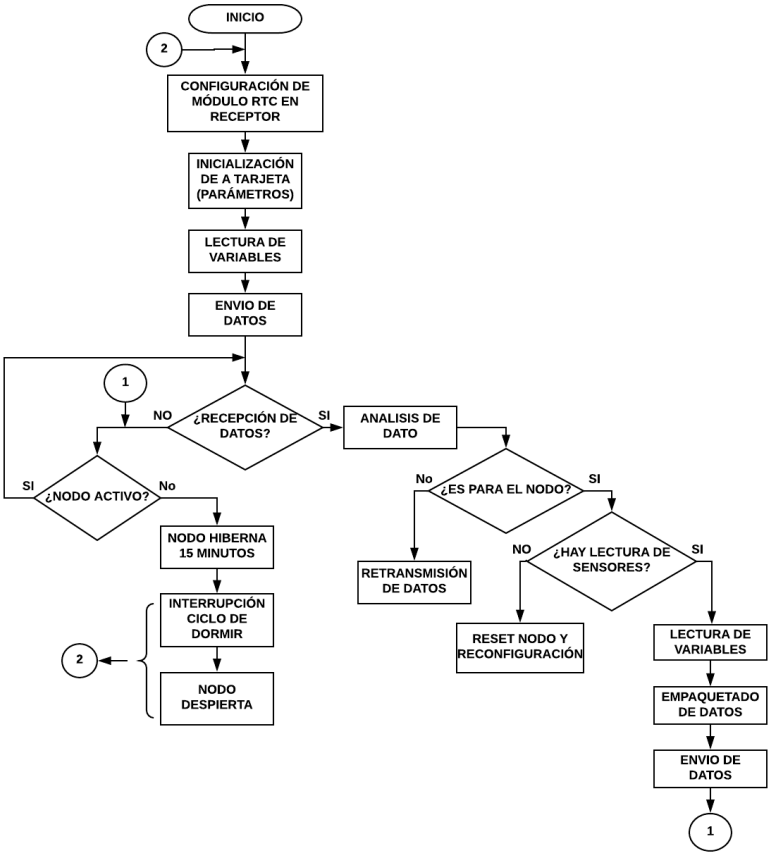


Figura 3.9 Diagrama a bloques del programa del nodo sensor.

3.5.2 Ciclo de Despertar - Dormir

Para lograr una eficiencia en cuanto al consumo energético, se programó una interrupción a través de un ciclo en función de la librería de Arduino implementada en la Freakduino para despertar-dormir al nodo.

El ciclo programado consiste en habilitar (despertar) al nodo sensor para transmitir los datos monitoreados de la variable de interés. Después del minuto el nodo sensor entra en hibernación por 15 minutos, apagando de forma total sus funciones, esperando según la programación de la librería para despertar al nodo nuevamente. Este ciclo puede ser configurado dependiendo de la aplicación o necesidades requeridas del sistema RIS. Cada nodo tiene configurado su tiempo, el cual puede diferir los otros nodos, teniendo en cuenta que los nodos son arrancados o encendidos en diferentes tiempos y, por tanto, los nodos sensores pueden o no ejecutar el ciclo de despertar-dormir al mismo tiempo que otros. El procedimiento

que se realiza para el ciclo de dormitar al nodo se hace mediante dos procesos principales. Primero el comando de Freakduino denominado *chibiSleepRadio* que se encarga de poner el radio inactivo. Se programa un 1 como verdadero para dormir al nodo y 0 para falso y despertar al nodo. Seguido de esto es necesario que los pines de SPI y radio control se inhabiliten mediante la manipulación de los registros del microcontrolador. El consumo de la tarjeta dormida debe ser en promedio de 22mA. Aun sin la integración del segundo proceso como se muestra en la figura 3.10.

Figura 3.10 Consumo energético.

El segundo procedimiento consiste en hacer uso de las librerías de Arduino avr/sleep.h y avr/power.h que permiten poner al microcontrolador a dormir. Para prevenir que el microcontrolador este mandando valores se establece el registro UART en cero mediante el siguiente comando:

```
UCSR0B=0x00;
```

Y después los puertos de los pines de uso general se establecen en cero y como entradas:

```
PORTC =0x00;
PORTD= 0x00;
DDRC= 0x00;
DDRD= 0x00;
```

El ADC se inhabilita ya que no se realizarán funciones de lecturas:

```
ADCSRA &= ~(1<<ADEN);
```

Al final se establece al microcontrolador en el modo del menor consumo energético posible. Cuando se hace esto las funciones del microcontrolador no serán accesibles hasta que se despierte, los comandos son los siguientes:

```
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
sleep_enable();
sleep_mode();
```

La tarjeta Freakduino que se utiliza en este proyecto se diseñó para que su consumo energético según Freaklabs sea de 180uA cuando este dormido el nodo lo que la hace capaz de sobrevivir durante un año con dos baterías AA estándar gracias a sus dos convertidores DC a DC (2x~60 uA) y la corriente de reposo del regulador de voltaje (~60 uA). Como se muestra en la figura 3.11 existe una optimización en el consumo energético.

Figura 3.11 Consumo energético reducido.

La única forma de despertar al microcontrolador es por medio de un reset externo. El consumo energético del nodo estando activo es de en promedio 140 ase muestra en la figura 3.12.

Figura 3.12 Consumo energético en activo.

Las pruebas de su rendimiento se muestran en el capítulo 4 en la sección de pruebas de consumo energético y su programación en la sección 3.10 de este mismo capítulo.

3.6 Fase de pruebas de diseño lógico de nodo sensor para intensidad RSSI

Para la realizar de las pruebas de intensidad con las tarjetas Freakduino se utilizó un sensor de temperatura para transmitir de nodo a nodo la información del sensor. El rango de la potencia de la señal es 84 (-7 dB) a 0 (-91 dB) para el AT86RF230 con una precisión de +/- 5 dB según Freaklabs, previamente en base a fórmulas expuestas en la sección 2.5 de esta tesis se determinaron los valores que maneja la tarjeta Freakduino.

El código programado para la tarjeta receptora es el siguiente, siendo utilizada la tarjeta Freakduino v2.1 LR, este código muestra la base principal para los futuros programas de pruebas.

```
#include <chibi.h> //Librería de la tarjeta Freakduino
byte buf[CHB_MAX_PAYLOAD]; //carga útil máx. de 116 bytes, default 100 para evitar sobrecargas
int val=0; //para conversión de unidades
//-----Variables para sensor de temperatura lm35-----
//int TempOffset = 500; // valor en mV cuando el ambiente es 0 grados Celsius
//int TempCoef = 10; // Coeficiente de temperatura en mV por grados Celsius
float ADCmV = 4.8828; // mV por incremento del ADC(5 volts / 1024 incrementos)
float Temp = 0; // calcular temperatura en Celsius
void setup(){
  Serial.begin(9600); //Serial.begin(57600);
//chibiSetShortAddr(3); //dirección Nodo receptor
  chibiInit();} //Inicializar el NODO
void loop(){
// Código principal donde se visualizan datos recibidos, si existen, procesarlos.
if (chibiDataRcvd() == true){
int x; //variable en la que se almacenara dato del sensor
uint8_t len, buf[20]; //entero sin signo de 8 bits
//------RSSI Y DIRECCIÓN DE NODO-----------------
int rssi, src_addr; //variables para RSSI y dirección del nodo proveniente
// obtener la fuerza se la señal y la dirección de donde proviene
  rssi = chibiGetRSSI();
  src_addr = chibiGetSrcAddr();
  val=rssi*-1; //Conversión
//--------datos del sensor y su interpretación------
  x=atoi((char*)buf); //cadena de caracteres a numero
  Temp = ((x*ADCmV*100)/1024);
```

```
//Temp = ((x * ADCmV) - TempOffset) / -TempCoef;
delay(200);
unsigned int rcv_data; //para usarse...
chibiGetData(buf);
 //------------Impresión de datos----------------
      Serial.print("Temperatura: ");
      Serial.print(Temp);
      //Serial.println((char *)buf);
      Serial.print(", Mensaje recibido del nodo 0x");
      Serial.print(src_addr, HEX);
      Serial.print(", RSSI = 0x");
      Serial.println(rssi,HEX);
      Serial.print("RSSI dB =");
      Serial.println(val,DEC); }}}
```

El código programado para la tarjeta transmisora es el siguiente, siendo utilizada la tarjeta Freakduino v2.1 estándar:

```
#include <Wire.h>
//#include <Sleep.h> //libreía para poner al nodo a dormir
#include <chibi.h> //Librería de la tarjeta Freakduino
// Variables del proyecto
int A0raw; //Pin analógico 0 para el sensor
//int A1raw;
const int bufSize = 10; //tamaño de buffer
byte XmitBuf[bufSize]; // transmisión de bufSize
void setup() {
Serial.begin(9600); //Serial.begin(57600);
chibiInit(); //Inicialización del NODO}
void loop() {
char temp [bufSize]; //almacenar el buffer para carácter temp
A0raw = analogRead(A0);//Lectura del sensor en A0
sprintf(temp,"%i",A0raw); //números a cadenas, formato núm. decimal, valor de lectura del sensor en A0
//A1raw = analogRead(A1);
//memcpy(XmitBuf, &A0raw, 2);// otro sensor extra
memcpy(XmitBuf, temp, bufSize); //copia datos en memoria
//chibiSleepRadio(0); // despertar el radio
//delay(100);
chibiTx(BROADCAST_ADDR, XmitBuf, bufSize);//chibiTx(nodo anfitrión (3),XmitBuf,bufSize)
Serial.println(temp);
//chibiSleepRadio(1); // volver a dormir // Tiempo de dormir de Freakduino.
//No poner nada extra en el código principal después de esto.
delay(200); //SleepClass::powerDownAndWakeupExternalEvent(0); }
```

Los códigos finales programados para la tarjeta receptora y transmisora se exponen en el Anexo E, donde para cada prueba se presentan las variaciones de la

recepción y transmisión de datos dependiendo de los sensores y formato de la información.

3.7 Consumo energético de las baterías

Para la prueba de consumo energético se usará MATLAB® para poder graficar el consumo de las baterías del nodo teniendo una transmisión de las lecturas cada 5 minutos. En la figura 3.13, se muestra el diagrama de conexión.

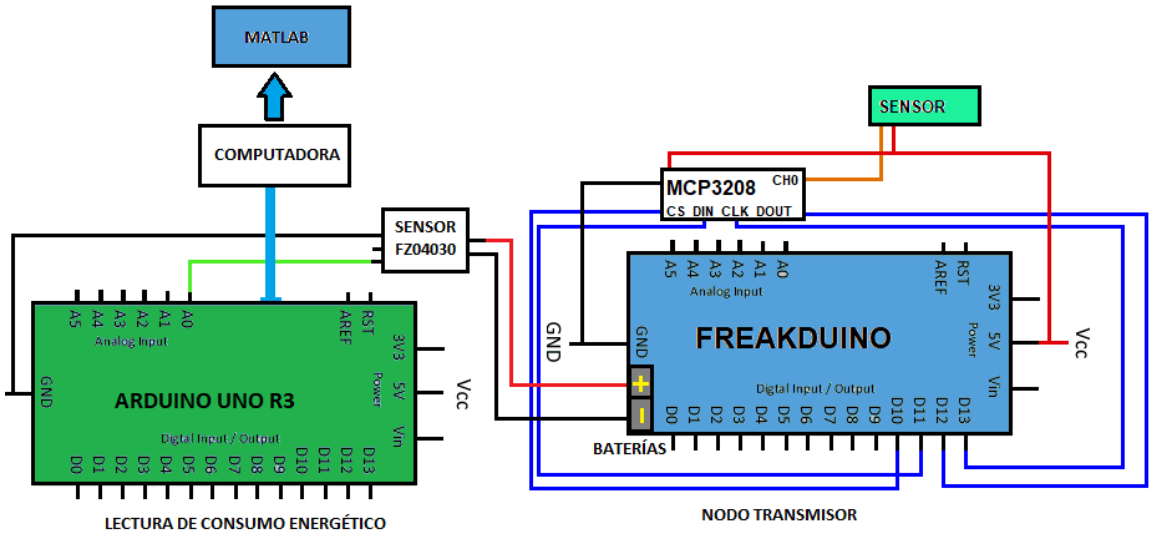


Figura 3.13. Esquema de conexión para pruebas.

Para la toma de lecturas del nodo ya sea en estado activo o inactivo se utilizaron equipos externos al mismo para no influir en el rendimiento del nodo. Se utilizó un multímetro STEREN modelo UT55 para verificar el voltaje al igual que un osciloscopio Tektronix TDS2002B y el sensor ACS712 para la corriente. Las baterías empleadas son AA recargables de marca MITZU cuyas características son: 2500mAh y voltaje de 2.5 V.

El nodo para el análisis del consumo energético con sensores implementados se muestra en la figura 3.14, donde ya se encuentra con los sensores implementados y una caja de protección para su estudio en campo exterior. En el anexo F se muestra el programa de MATLAB diseñado para el monitoreo del estado del voltaje.

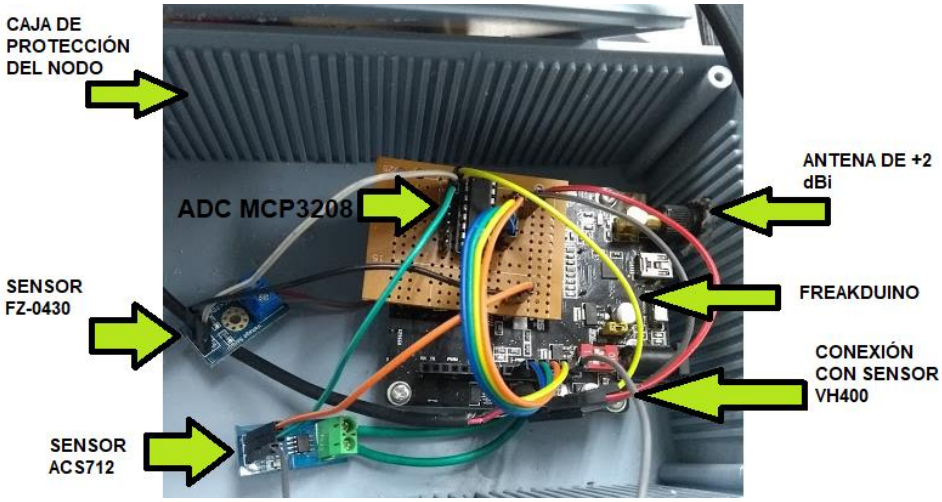


Figura 3.14. Conexión de sensores de consumo energético.

3.8 Diseño de prototipos de Nodos Sensores

Se hicieron 3 prototipos de nodos sensores con la siguiente integración:

Nodo 1: Implementado con Módulo RTC para verificar la recepción de datos con fecha y hora del nodo transmisor.

Nodo 2: Implementado con sensor VH400 y módulos FZ0430 y ACS712 para el monitoreo del estado de la batería del nodo para la prueba de consumo energético.

Nodo 3: Implementado para sensores de todo tipo, teniendo un sensor lm35dz y DHT11 para la prueba de transmisión de datos con diferentes formatos.

En la figura 3.15 se presenta el interior del nodo 2.

Figura 3.15. Conexión nodo 2.

En las figuras 3.16, 3.17 y 3.18 se presentan los diagramas de conexión para cada uno de los nodos.

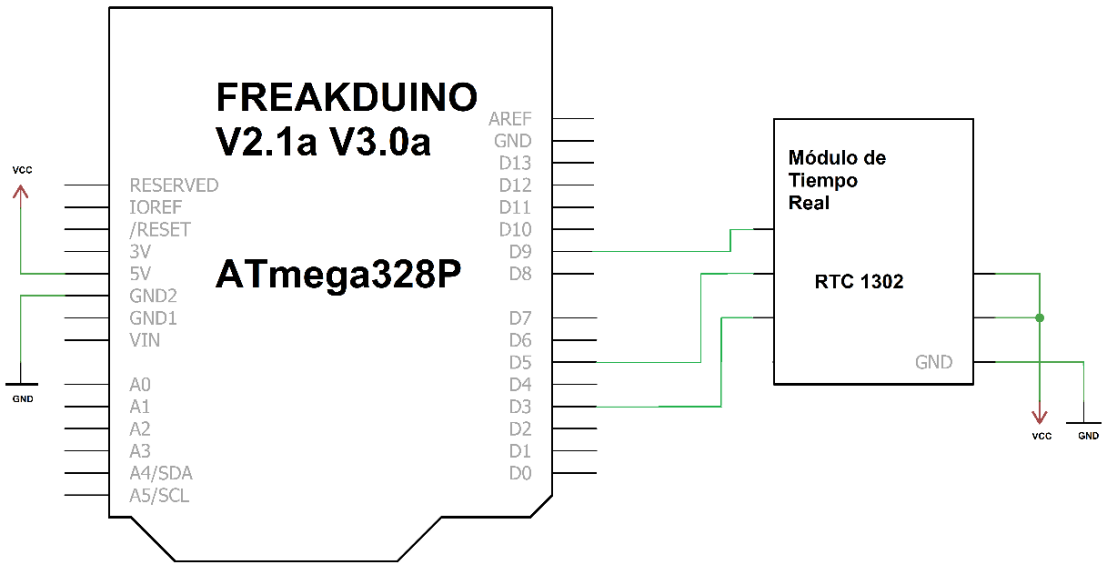


Figura 3.16. Esquema de conexión del nodo 1.

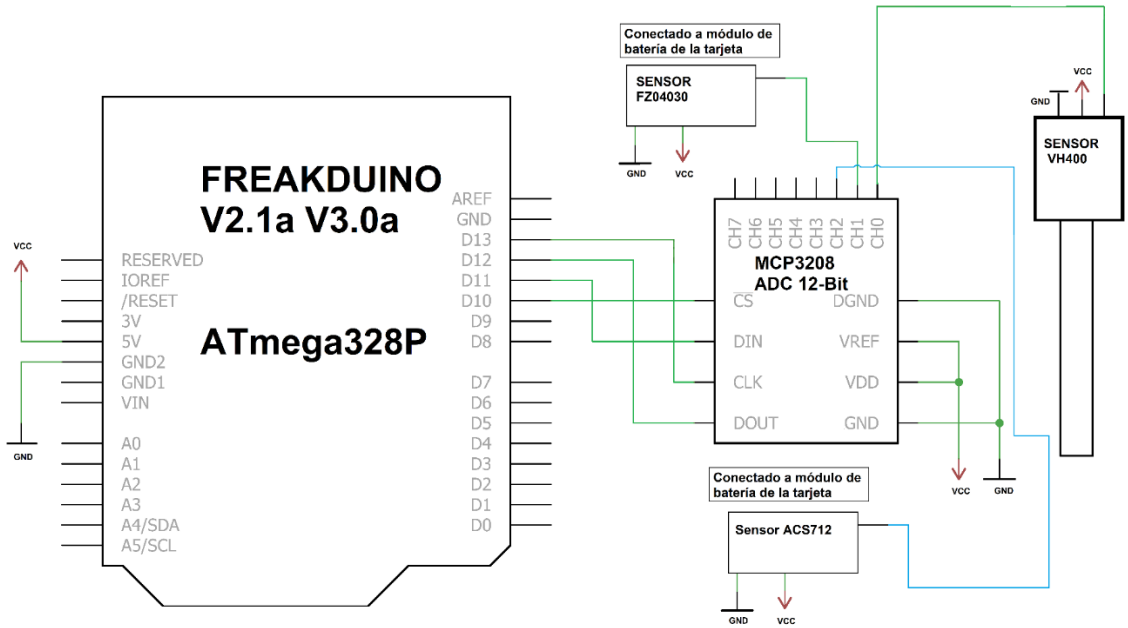


Figura 3.17. Esquema de conexión del nodo 2.

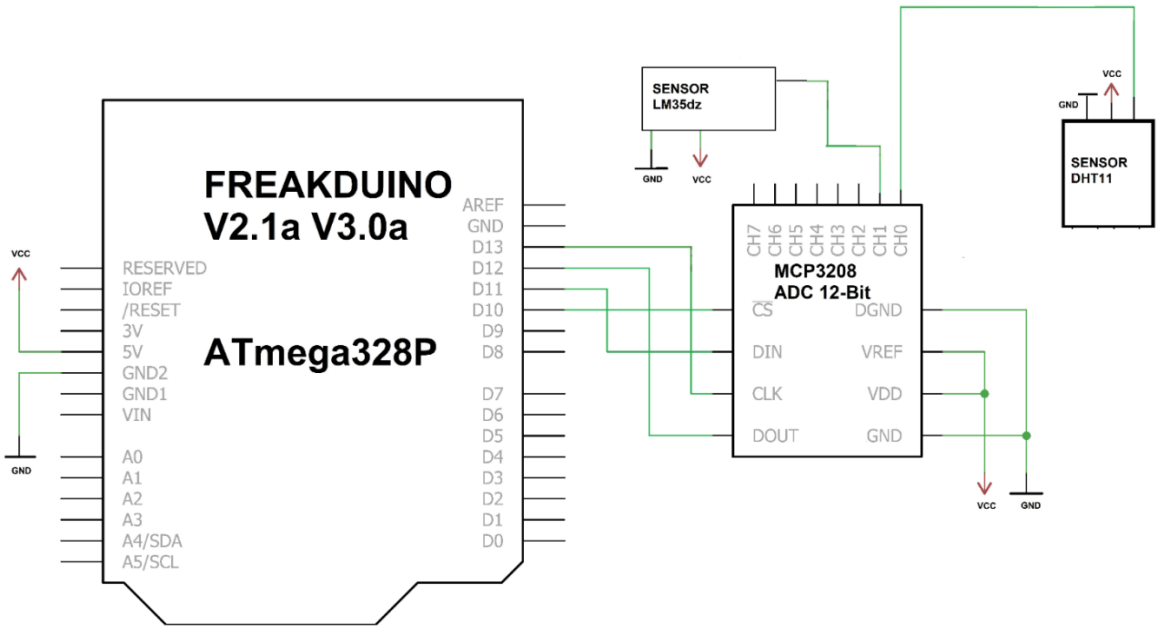


Figura 3.18. Esquema de conexión del nodo 3.

3.9 Pruebas de distancia con respecto a la potencia de la señal RSSI

Para la medición de la propagación de la señal, se seleccionó como zona de estudio un escenario externo con obstáculos; el Bosque Venustiano Carranza. En la figura 3.19, se muestra la ubicación de la distancia de los nodos con respecto al nodo receptor.

Figura 3.19. Ubicación de nodos.

Las pruebas se realizaron con dos tipos de tarjetas: Freakduino Long Range 900 MHz v2.1a como nodo receptor y Freakduino Estándar 900 MHz v3.0a como transmisora, cuyos microcontroladores son el Atmega328P y el circuito integrado inalámbrico que permite la radio comunicación para ambas es el AT86RF230 cuya potencia de señal es de 84 (-11.48dBm) a 0 (-96.97dBm) con una precisión de +/-5 dB. También se utilizó una computadora Dell cuyo procesador es AMD A10-8700P Radeon R6 de 4 núcleos y sistema operativo Windows 10 de 64 bits para el desarrollo de la programación y visualización de los resultados en el monitor serial.

Para realizar la medición de la potencia RSSI con respecto a la distancia, se utilizaron dos aplicaciones de geo posicionamiento con smartphones de Sistema operativo (SO) Android las cuales son GPS Status de la compañía *MobiWIA-EclipSim y* la aplicación llamada Medición de áreas y distancias de la compañía *Studio Noframe* siendo estas aplicaciones de libre acceso mediante Google Play store y sus interfaces se muestran en la figura 3.20, que permitieron mediante coordenadas, tomar una lectura en metros del nodo receptor al nodo transmisor en diferentes puntos de ubicación. La primera aplicación maneja un error de 2 a 5 metros y la segunda un error de 2 a 4 metros en las mediciones.

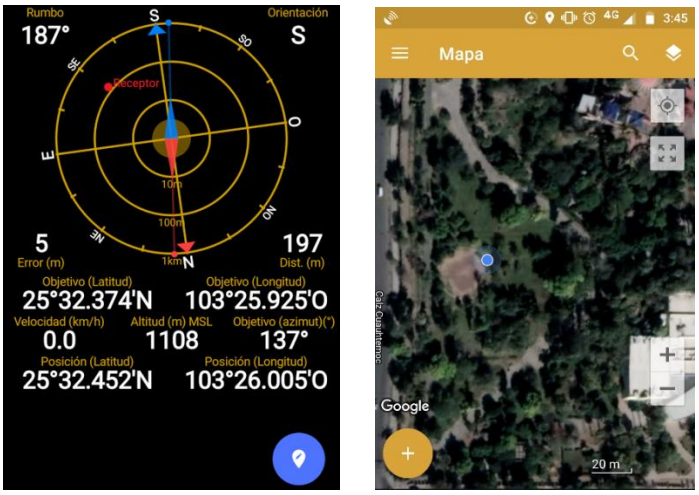


Figura 3.20. Aplicaciones GPS.

Por medio del nodo transmisor, se realiza la adquisición de datos de los sensores, se programa la dirección del nodo receptor y la dirección del nodo transmisor a la qué se va enviar el paquete de información. Las distancias seleccionadas para las pruebas de las tres fases son de 0, 30, 50, 100, 120, 150, 180 y 200 metros.

3.10 Programación de pruebas de consumo energético.

En las pruebas de consumo energético se utilizaron diferentes configuraciones para un solo nodo transmisor y el uso de MATLAB® para mostrar los resultados. En el anexo C se muestra el programa diseñado para monitorear el consumo. Se programó al nodo para estar mandando la información cada 5 minutos por medio de hasta el final de la carga de las baterías.

Los programas realizados para el monitoreo del consumo energético de forma independiente a la Freakduino fueron los siguientes:

Para sensor de corriente ACS712:

```
float Sensibilidad=0.185; //sensibilidad en Voltios/Amperio para sensor de 5A
void setup() {
  Serial.begin(9600);
}
void loop() {
  float I=get_corriente(200);//obtenemos la corriente promedio de 500 muestras
  Serial.print("Corriente: ");
  Serial.println(I,3);
  delay(100);
}
float get_corriente(int n_muestras){
  float voltajeSensor;
  float corriente=0;
  for(int i=0;i<n_muestras;i++)
  {
    voltajeSensor = analogRead(A0) * (5.0 / 1023.0);////lectura del sensor
    corriente=corriente+(voltajeSensor-2.5)/Sensibilidad; //Ecuación  para obtener la corriente
  }
  corriente=corriente/n_muestras;
  return(corriente);}
```

Para sensor de corriente de voltaje FZ-0430:

```
const int sensor = A0; // seleccionar la entrada para el sensor
int ValorSensor;       // variable que almacena el valor raw (0 a 1023)
float valor=0;         // variable que almacena el voltaje (0.0 a 25.0)

void setup(){
 Serial.begin(9600);
}
void loop(){
   ValorSensor = analogRead(sensor);            // realizar la lectura
   valor = fmap(ValorSensor, 0, 1024, 0.0, 25.0);// cambiar escala a 0.0 - 25.0

   Serial.println(ValorSensor);
   //Serial.println(valor);                     // mostrar el valor por serial
   delay(500);
}
// cambio de escala entre floats
float fmap(float x, float in_min, float in_max, float out_min, float out_max)
{return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;}
```

CAPÍTULO 4

RESULTADOS

4.1 Resultados de la prueba de distancia con respecto a la potencia de la señal RSSI con sensor de temperatura.

En la primera prueba se utilizaron los parámetros de la librería estándar de la tarjeta y un sensor lm35dz para medir la temperatura ambiente. Las características de la prueba fueron las siguientes:

Zona: Bosque Venustiano Carranza **Hora:** 12:00 pm Fecha:30/09/17
Tarjera receptora: Freakduino Long Range 900MHz
Tarjeta Transmisora: Freakduino Estándar 900MHz
Distancia Total: 200.00 m.
Clima: 27°C **Humedad** 18% **Viento:** 0km/h. Nublado **Precipitación** 0%

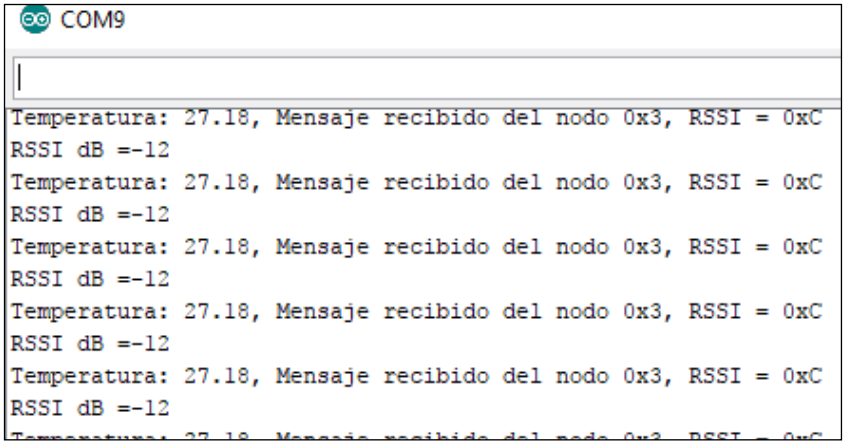


Figura 4.1. Lecturas del sensor lm35dz en monitor serial.

Los resultados de la prueba 1, se presentan en la tabla 4.1 mostrando variaciones en las lecturas de temperatura debido a que ese día se encontraba en riego zonas diferentes del bosque. También otro factor influyente fue la sombra de los arboles donde se ubicó al nodo.

Tabla 4.1. Resultados de la prueba con sensor de temperatura lm35dz.

FECHA Y HORA	DIR. NODO	SENSOR DE T.	DISTANCIA	RSSI
vie_30/09/2017 HORA:12:00:00	0X3	25.27	0	84
vie_30/09/2017 HORA:13:07:01	0x3	26.7	30	26
vie_30/09/2017 HORA:12:50:02	0X3	27.18	50	7
vie_30/09/2017 HORA:12:52:00	0X3	27.18	100	12
vie_30/09/2017 HORA:12:53:00	0X3	27.17	120	11
vie_30/09/2017 HORA:12:56:04	0X3	27.66	150	4
vie_30/09/2017 HORA:12:58:25	0x3	27.71	180	3
vie_30/09/2017 HORA:13:00:01	0X3	25.75	200	1

En la figura 4.2, se presenta la gráfica de RSSI vs distancia El valor máximo fue de 84, a una distancia de 0 metros entre nodos y la mínima fue de 1, a 200 m de distancia. Los valores intermedios presentan variaciones, debido a los obstáculos, sin embargo; se considera que las lecturas obtenidas con el sensor de temperatura son confiables hasta los 200 metros, pero empezando a tener pérdidas de la información y retrasos en la recepción de las lecturas.

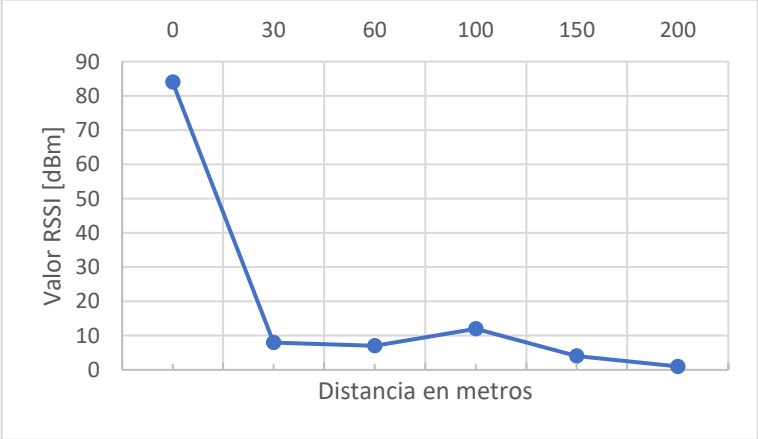


Figura 4.2. RSSI vs distancia (sensor lm35dz).

En las pruebas siguientes se hicieron modificaciones a la librería, en específico en el tipo de modulación propia de la tarjeta Freakduino en el parámetro CHB_900_MHz_INIT_MODE_OQPSK_SIN_250 para tratar de aumentar la potencia RSSI.

4.2 Resultados de la prueba de distancia con respecto a la potencia de la señal RSSI con sensor de temperatura y humedad.

En la segunda prueba se toma como referencia la temperatura máxima de 33°C pronosticada. Se utilizó el sensor DHT11 como se muestra en la figura 4.3. Las características de la prueba fueron las siguientes:

Zona: Bosque Venustiano Carranza **Hora:** 11:19 am Fecha:07/11/17
Tarjera receptora: Freakduino Long Range 900MHz
Tarjeta Transmisora: Freakduino Estándar 900MHz
Distancia Total: 200.00 m.
Clima: 33°C **Humedad** 14% Viento: 11.1km/h. **Precipitación** 0%

Figura 4.3. Programación de sensor DHT11.

Los resultados de la prueba 2 que se muestran en la tabla 4.2, muestran un comportamiento similar la primera prueba realizada con el sensor lm35dz. El sensor de temperatura y humedad también mantuvo lecturas confiables en todo momento hasta el límite de 200 metros, pero teniendo retrasos en la recepción. Nuevamente, las variaciones de temperatura son por la misma causa mencionada anteriormente y el factor de humedad debido a que se habían regado distintas áreas del bosque. El resultado logra mostrar un aumento en la potencia RSSI.

Tabla 4.2. Prueba con sensor DHT11.

FECHA Y HORA	DIR. NODO	SENSOR DE H.	SENSOR DE T.	DISTANCIA	RSSI
mar_7/11/2017 HORA:11:19:00	0x3	20%	27.00	0	84
mar_7/11/2017 HORA:11:28:00	0x3	14%	27.00	30	22
mar_7/11/2017 HORA:11:43:00	0x3	8%	33.00	50	15
mar_7/11/2017 HORA:11:52:00	0x3	11%	30.00	100	19
mar_7/11/2017 HORA:11:52:00	0x3	11%	30.00	120	17
mar_7/11/2017 HORA:13:19:00	0x3	13%	30.00	150	12
mar_7/11/2017 HORA:13:26:00	0x3	10%	30.00	180	7
mar_7/11/2017 HORA:13:27:00	0x3	10%	30.00	200	6

La figura 4.4, presenta la gráfica de RSSI vs distancia. El valor máximo fue de 84, a una distancia de 0 metros entre nodos y la mínima fue de 6, a 200 metros de distancia. Entre 30 y 100 metros se puede observar que se mantuvo la potencia RSSI casi sin una variación muy diferente a cuando aumenta la distancia de 150 metros en delante.

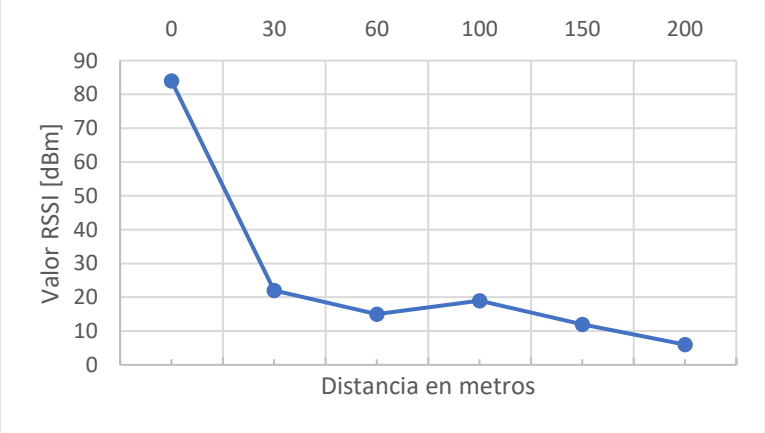


Figura 4.4. RSSI vs distancia (DHT11).

Esto significa un avance en cuanto al comportamiento de las tarjetas en la zona del bosque además de ser otro modo de transmitir los datos de un sensor cuya programación es diferente a la del sensor usado en la primera prueba.

4.3 Prueba de distancia con respecto a la potencia de la señal RSSI con sensor de humedad.

En la tercera prueba se usó un sensor de humedad FC-28. Además, se implementó un aumento en la resolución; de 10 bits de la tarjeta Freakduino a 12 bits mediante el uso de un ADC MCP3208. En la figura 4.5 se muestra el ADC MCP3208 con dos sensores FC-28 acoplado con Freakduino al momento de estar siendo programado para realizar las pruebas en campo con obstáculos.

Zona: Bosque Venustiano Carranza **Hora:** 1:05 pm Fecha:24/01/18
Tarjera receptora: Freakduino Long Range 900MHz
Tarjeta Transmisora: Freakduino Estándar 900MHz
Distancia Total: 180.00 m.
Clima: 18°C **Humedad** 43.6% Viento:13 km/h. **Precipitación** 0%

Figura 4.5. Freakduino con sensor FC-28 y ADC MCP328.

Se modificaron parámetros de la modulación mediante la librería de la tarjeta Freakduino, definiendo el comando directo en el programa del nodo: CHB_900_MHz_INIT_MODE_OQPSK_SIN_1000, para ver si existía algún aumento en la distancia de cobertura. Se investigó en la página oficial el uso de una modulación por desplazamiento de fase de dos símbolos (BPSK) para lograr incrementar el rango de la distancia, teniendo como único inconveniente la velocidad con que se transfieren los datos, por lo cual se optó por seguir utilizando las modulaciones de desplazamiento de fase en cuadratura escalonada (OQSPK).

Tabla 4.3. Prueba con sensor de humedad FC-28.

FECHA Y HORA	DIR. NODO	SENSOR DE H.	DISTANCIA	RSSI
mie_24/1/2018 HORA:13:5:52	0X5	39%	0	84
mie_24/1/2018 HORA:13:19:34	0X5	39%	30	36
mie_24/1/2018 HORA:13:21:59	0X5	39%	50	31
mie_24/1/2018 HORA:13:25:37	0X5	40%	100	27
mie_24/1/2018 HORA:13:28:27	0X5	40%	120	28
mie_24/1/2018 HORA:13:58:43	0X5	39%	150	29
mie_24/1/2018 HORA:13:39:26	0X5	40%	180	27

Los resultados de la tabla 4.3, muestran que el RSSI aumenta de forma notable, con respecto a las dos primeras pruebas. Las lecturas del sensor se mantuvó estable en todo momento excepto cuando la distancia entre nodos superaba los 180 metros. Dejo de existir recepción de datos cuando se dejaba al nodo ubicado en 200 metros. Con la modificación a las configuraciones de la librería, se logró un aumento en el valor RSSI aumentando casi el doble su potencia. No se logró obtener un rango de distancia que sobrepasara los 180 metros como se ve en la figura 4.6.

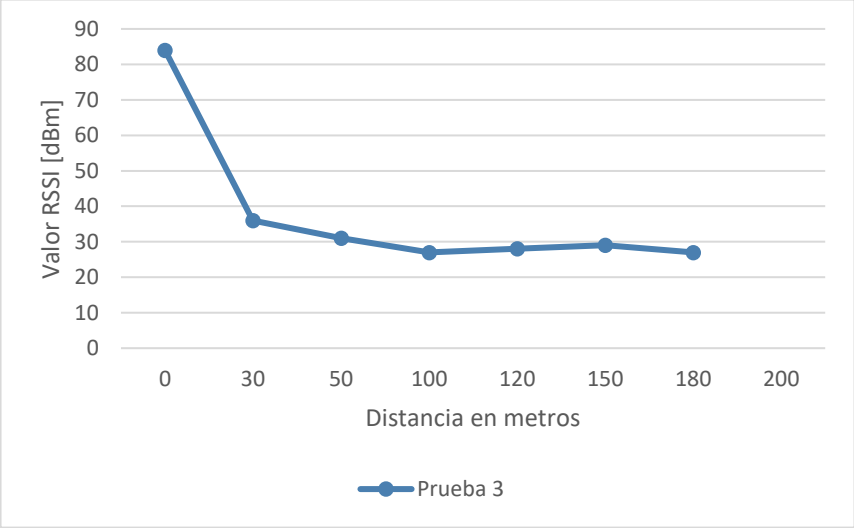


Figura 4.6. RSSI vs distancia con sensor FC-28.

4.4 Comparación de las pruebas de potencia RSSI

En la figura 4.7, se presenta la comparativa de la potencia RSSI vs distancia seleccionada para las 3 pruebas. Como se puede observar el resultado de la prueba 3 muestra un aumento en la potencia RSSI a comparación de las primeras dos pruebas donde las configuraciones en la primera no sufrieron modificación a como viene por default la librería programada. Esto sin duda puede significar que la tarjeta Freakduino es capaz de penetrar obstáculos como lo son árboles y estructuras de concreto sin mayor problema. La tarjeta Freakduino se puede implementar en un campo de cultivo ya sea a campo abierto o cerrado como en invernadero y mantener lecturas confiables a distancias desde 1 hasta 200 metros considerando que aun puede haber incremento en la distancia si se utiliza una antena direccional que se acople a este propósito.

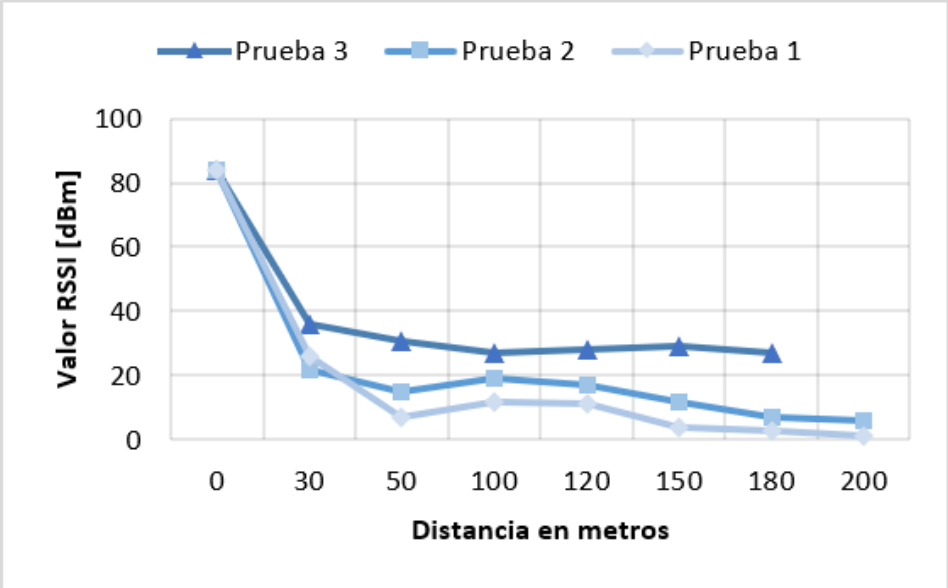


Figura 4.7. RSSI vs distancia de las tres pruebas.

Otro detalle a considerar es que la zona del bosque Venustiano Carranza tiene muchas señales Wi-Fi que muy probablemente hayan causado interferencia en cada una de las pruebas. En una zona libre de interferencias es probable que la distancia y la potencia RSSI logren tener un alcance mayor.

4.5 Pruebas de consumo energético en campo abierto del ITL

El consumo requerido de energía por nodo sensor varía dependiendo de los elementos que componen al nodo en cuestión. Las primeras dos pruebas se basan en el nodo 2. Cuando el nodo se encuentra activo y transmitiendo los datos de un sensor, consume un promedio de 99.2 mA. En el estado inactivo del nodo durante la primera prueba en el laboratorio consumía en promedio 26.3 mA, obteniendo una lectura como se aprecia en la figura 4.8 que es diferente a lo que se especifica por Freaklabs. El voltaje de operación que se seleccionó para que trabajara la tarjeta fue de 3.3 volts.

Figura 4.8. Medición de consumo energético con nodo activo.

Después de realizar algunos ajustes para el ciclo de dormitar para optimizar y reducir el consumo energético, consumió en promedio 200uA con voltaje de operación de 3.0 V siendo suministrados por las baterías AA recargables Mitzu. Estas pruebas y la programación se hicieron primero en el laboratorio.

Figura 4.9. Medición de consumo energético con nodo inactivo.

Especificaciones de las condiciones de las pruebas:

Zona: Gradas del ITL Hora: 13:20 Fecha:28/08/17
Tarjera receptora: Freakduino Long Range 900MHz
Tarjeta Transmisora: Freakduino Estándar 900MHz
Distancia Total: 207.35 m.
Clima: 27°C Humedad 51% Viento: 5km/h. Nublado Precipitación 0%

En la primera de consumo, se encontró que el sistema consumía 330uA con las baterías a 2.45V en modo inactivo, y un promedio de 82mA en modo activo.

Zona: Cancha de atletismo del ITL Hora: 2:15 pm Fecha: 05/09/17
Tarjera receptora: Freakduino Long Range 900MHz
Tarjeta Transmisora: Freakduino Estándar 900 MHz
Prueba con antena de +8dbi en el receptor (2.1a), con antena en el transmisor 2dbi, **Bytes de información enviados:** 10,
Distancia Total: 158 m.
Clima; 27° C **Humedad** 51% Viento: 5km/h. Nublado Precipitación 0%

Para ambas pruebas se muestran los resultados en la figura 4.10 y 4.11.

Figura 4.10. Medición de consumo promedio inactivo.

Figura 4.11. Medición de consumo promedio activo.

El estado de las baterías se mantuvo constante durante 3 días con 11 horas de pruebas teniendo programado un muestreo cada 5 minutos del nodo teniendo 288 muestras por día manteniendo las lecturas del sensor de forma regular sin perdidas

de la información. En la figura 4.12 se muestra la gráfica del voltaje de las baterías hasta 1000 muestras.

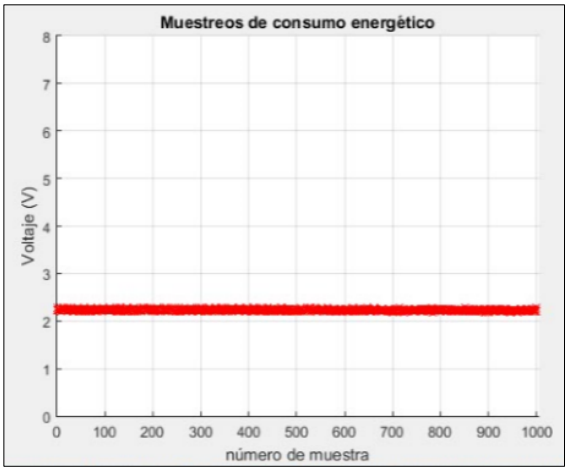


Figura 4.12. Gráfica del estado de las baterías en MATLAB®.

Este resultado del consumo se obtuvo leyendo el voltaje de forma no invasiva a la Freakduino haciendo uso de una tarjera Arduino UNO. Para la obtención de los resultados del consumo de corriente, se decidió usar de un instrumento de medición externo. A comparación del nodo sensor de Waspmote Libelium, en el cual, sin tener un sistema de recarga de baterías, el rendimiento de la tarjeta Freakduino muestra una disminución de apenas 50% en las baterías y tan solo un 15% El porcentaje para la tarjeta Waspmote disminuye del 90% a un 15% en cuatro días teniendo una pérdida del 75%. La tabla 4.4 y 4.5 muestran una comparativa básica del comportamiento en ambos nodos.

Tabla 4.4. Batería de nodo Waspmote.

Nodo basado en Wapmote Libelium	
Día	batería
1	90%
2	50%
3	42%
4	15%

Tabla 4.5. Batería de nodo Freakduino.

Nodo basado en Freakduino	
Día	batería
1	92%
2	89%
3	85%
4	50%

El porcentaje disminuye del 92% a un 50% en cuatro días teniendo una pérdida del 42%. El nodo sensor de Freakduino empieza a tener perdidas de la información además de lecturas erróneas cuando el porcentaje de la batería restante es de menos de 15%. Después de analizar los resultados se puede concluir que además de mostrar de que el sistema puede permanecer teóricamente 6 meses de forma autónoma y que la reducción de costos en la implementación de nodos diseñados mediante Freakduino representan una mejora para el desarrollo de una RIS esperando que en caso de falla en alguna parte de la red gran parte de la información se logre respaldar. Si al nodo además se le agrega un sistema de recarga como lo puede ser una celda solar, su vida en el campo puede alargarse de tal modo que pueda estar activo durante un tiempo aproximado de más de 6 meses como es lo esperado. Otro detalle importante es que el diseño lógico del nodo sensor de este proyecto no solo puede ser aplicado con un enfoque agronómico, sino que puede ser adaptado a múltiples ámbitos debido a que su base de programación es completamente adaptable a diversos tipos de sensores. El implementar nodos basados en Freakduino para una RIS teniendo en cuenta el tipo de clima de la comarca lagunera mediante el uso celdas fotovoltaicas como fuentes de alimentación para los nodos, favorecería en gran medida el prolongar el tiempo de vida de la red para un monitoreo de cultivos en zona abierta.

CAPÍTULO 5

CONCLUSIONES

- El diseño lógico de nodo sensor basado en la tarjeta Freakduino logró integrar el uso del software Arduino de acceso libre y se desarrolló la fabricación del prototipo.
- Se Implementó la electrónica y programación de la tarjeta Freakduino-900 v3.0a y 2.1a como parte de la unidad de adquisición de datos, comunicación y monitoreo.
- Se encuentra en desarrollo la Integración de la tarjeta Freakduino con el protocolo mandrágora, mediante la implementación de computadora embebida Raspberry-pi.
- Se diseñó el prototipo para que sea un sistema autónomo de nodo que incluya la implementación de un convertidor analógico digital MCP3208 que incrementó la resolución de 10 a 12 bits.
- Se redujeron los costos en 59.53% remplazando el uso de Waspmote Libelium con Freakduino, como se presentó en el presupuesto del capítulo 1 teniendo un total de $32,768.00 contra $7964.07.
- Se logró implementar sensores que midan diferentes variables agrícolas, las cuales fueron temperatura, humedad del ambiente y humedad del suelo.
- Se utilizaron algoritmos y estructuras de topologías RIS para colocar dispositivos a diferentes distancias cubriendo distancias mayores con respecto a los nodos de Waspmote Libelium.
- El consumo energético se mejoró al lograr optimizar en más de 50% de perdida de energía en las baterías.
- Se incorporó un diseño con modalidad de adaptable y configurable de otros sensores, que tenga las cualidades necesarias para aplicaciones agronómicas.

- Se realizaron pruebas con sensores de humedad y temperatura, de comunicación Wireless, consumo energético, potencia y distancia del prototipo en campo abierto, con y sin obstáculos.
- Como resultado del proyecto se sometió un artículo científico denominado Estudio de potencia del Indicador de Fuerza de la Señal Recibida (RSSI) para la distancia entre nodos, basado en Freakduino para Redes Inalámbricas de Sensores (RIS) en la revista Ingeniería e Investigación de Colombia, ISSN: 01205609.
- Se redactó un manual de operación de la tarjeta Freakduino como resultado del desarrollo del prototipo para la RIS el cual se presenta en el anexo A.
- Las aportaciones más importantes de este proyecto son el uso e implementación de las tarjetas Freakduino cuyo uso sirve para conocer su funcionamiento en la integración de nodos, así como de implementar la electrónica necesaria como elemento vital para tener un nodo completamente funcional y autónomo capaz de realizar sus funciones de manera eficiente.
- A pesar de obtener resultados positivos en diversas pruebas, queda pendiente el probar nuevas configuraciones que otorguen mejor resultados. El modo de transmisión de datos fue de punto a punto siendo eficiente para su estudio y aunque a los nodos se les puede programar la opción de retransmisión entre nodos, se sigue buscando un mejor método para el enrutamiento por si esta RIS se implementa a distancias mayores a las estudiadas.
- Se trabajo en la interfaz visual con la computadora embebida sin tener aun algunos de las bases para integrar el protocolo mandrágora, y actualmente se encuentra en desarrollo.
- En el diseño electrónico se dejó lista la conexión para implementarle un módulo GPRS para saber exactamente la ubicación del nodo.

RECOMENDACIONES PARA TRABAJOS FUTUROS

Es recomendable trabajar en un diseño electrónico que pueda fabricarse en base al prototipo desarrollado, que actualmente a pesar de ser funcional no es un sistema completamente integrado. Es importante considerar que para futuras pruebas en las que se estudien las capacidades máximas de las tarjetas orientadas a las RIS como lo es la Freakduino, se seleccione un campo abierto en el cual la existencia de señales sea nula para evitar posibles interferencias como las encontradas durante las pruebas en el bosque Venustiano Carranza y el ITL que son actualmente zonas donde las señales Wi-Fi se encuentran en abundancia interfiriendo con la transmisión de la información. Se puede implementar la energía solar por medio de paneles solares para prolongar la vida útil del nodo.

Fuentes de Información

[1] S. Naturales, "Participa SEMARNAT en la Alianza Nacional por el Suelo", gob.mx, 2018. [Online]. Disponible en: https://www.gob.mx/semarnat/prensa/participa-semarnat-en-la-alianza-nacional-por-el-suelo. [Consultado el: 15- Dic- 2018].

[2] "Situación y contexto de la problemática del agua en México", Aguas.org.mx, 2018. [Online]. Disponible en: http://www.aguas.org.mx/sitio/index.php/panorama-del-agua/diagnosticos-del-agua. [Consultado el: 02- Dic- 2017].

[3] M. Flores, Redes Inalámbricas de Sensores en Aplicaciones Agronómicas. TECNOLÓGICO NACIONAL DE MÉXICO Instituto Tecnológico de La Laguna, 2015.

[4] V. Velasco, Diseño de Nodos y de una Red Inalámbrica de Sensores para Mediciones Agropecuarias. México: Instituto Tecnológico de la Laguna, 2009.

[5] M. Cervantes and A. Franco, "Diagnóstico Ambiental de la Comarca Lagunera", Observatoriogeograficoamericalatina.org.mx, 2006. [Online]. Disponible en: http://observatoriogeograficoamericalatina.org.mx/egal11/Procesosambientales/Impactoambiental/22.pdf. [Consultado el: 25- Feb- 2018].

[6] A. Nasir, X. Zhou, S. Durrani and R. Kennedy, "Relaying Protocols for Wireless Energy Harvesting and Information Processing", IEEE Transactions on Wireless Communications, vol. 12, no. 7, pp. 3622-3636, 2013.

[7] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, "Energy conservation in wireless sensor networks: A survey", Ad Hoc Networks, vol. 7, no. 3, pp. 537-568, 2009.

[8] J. A. L. Riquelme, F. Soto, J. Suardíaz, and A. Iborra, “Red de Sensores Inalámbrica para Agricultura de Precisión”, en *Telecoforum*, 2008, pp. 3–4.

[9] Narayanan, R., Sarath, T. and Vineeth, V. (2016). Survey on Motes Used in Wireless Sensor Networks: Performance & Parametric Analysis. Wireless Sensor Network, 08(04), pp.51-60.

[10] *K. Romer and F. Mattern, "The design space of wireless sensor networks", IEEE Wireless Communications, vol. 11, no. 6, pp. 54-61, 2004.*

[11] "Tarjetas de adquisición de datos", Jmi.com.mx, 2017. [Online]. Disponible en: https://www.jmi.com.mx/tarjetas-de-adquisicion-de-datos.html. [Consultado el: 10- Nov- 2017].

[12] H. Friis, "A Note on a Simple Transmission Formula", Proceedings of the IRE, vol. 34, no. 5, pp. 254-256, 1946.

[13] *B. Roy. “Sistemas electrónicos de comunicaciones”, Thomson Learning, 2da ed., Mayo 2004.*

[14] Mesa López-Colmenar, R. Castañeda Barrena and L. Villar Angulo, La innovación en la enseñanza superior. Sevilla: Universidad de Sevilla, Instituto de Ciencias de la Educación, 2006.

[15] S. Bigelow, Understanding telephone electronics. Indianapolis, IN: SAMS, 1991.

[16] J. Zheng and A. Jamalipour, Eds., “*Wireless Sensor Networks: A Networking Persepective*”. United States of America”. Wiley, 2009.

[17] N. Chio Cho, D. A. Tibaduiza Burgos, A. Z. L. Cristina, C. O. L. Miguel, —Redes de sensores inalámbricos, ‖ in *2do. Congreso Internacional de Ingeniería Mecatrónica UNAB*, 2009.

[18] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Computer Networks (Elsevier) Journal 38 (4) (2002) 393–422.

[19] "Agricultura de Precisión para Cultivos de Café en Colombia", Docplayer.es, 2018. [Online]. Disponible en: http://docplayer.es/12721892-Agricultura-de-precision-para-cultivos-de-cafe-en-colombia.html. [Consultado el: 03- Feb- 2018].

[20] G. Zhou, S. Tang, D. Lu, L. Liu, J. Han and W. Dong, "Wireless Sensor Networks for Agriculture and Forestry", International Journal of Distributed Sensor Networks, vol. 11, no. 6, p. 845364, 2015.

[21] Freaklabs Store, "FreakLabs Store, Open Source Wireless Sensor Networks", Freaklabsstore.com, 2017. [Online]. Disponible en: http://www.freaklabsstore.com. [Consultado el: 06- May- 2017].

[22] Freaklabs Store, "Freakduino 2.4 GHz Wireless Arduino Compatible Board, v1.1a [FREAKDUINO-CHIBI]: FreakLabs Store, Open Source Wireless Sensor Networks", Freaklabsstore.com, 2017. [Online]. Disponible: http://www.freaklabsstore.com/index.php?main_page=product_info&products_id=187. [Consultado el: 06- May- 2017].

[23] B. Bolton, Mediciones y pruebas eléctricas y electrónicas. México: Alfaomega, 1996.

[24] MCP3208 DataSheet disponible en: www.alldatasheet.com/Mcp3208 [Consultado el 01 ene. 2018]

[25] "Soil Moisture Sensor - VH400", Vegetronix.com, 2008. [Online].
Disponible en: http://www.vegetronix.com/Products/VH400/. [Consultado el: 08- Ene- 2018].

[26] LM35 Sensor de temperatura. [online] Teslabem.com. Disponible en: http://teslabem.com/lm35-sensor-de-temperatura.html [Consultado el 27 Sept. 2017].

[27] J. Zamudio, "DHT11 con Arduino: Sensor Temperatura y Humedad - Geek Factory", Geek Factory, 2018. [Online]. Disponible en: https://www.geekfactory.mx/tutoriales/tutoriales-arduino/dht11-con-arduino/. [Consultado el: 27- Sep- 2018].

[28] "PFG – Sensor de humedad FC-28", Eodos, 2018. [Online]. Disponible en: https://eodos.net/proyectos/sensor-de-humedad#.WtfJ-IjwbIU. [Consultado el: 27 - Sept- 2018].

[29] FC-28 DataSheet disponible en: http://www.alldatasheet.es/view.jsp?Searchword=FC-28 [Consultado el 02 ene. 2018].

[30] "Medir voltajes de hasta 25V con Arduino y FZ0430 | tecno4", Scoop.it, 2017. [Online]. Disponible en: https://www.scoop.it/t/tecno4/p/4073391858/2016/12/29/medir-voltajes-de-hasta-25v-con-arduino-y-fz0430. [Consultado el: 27- Sep- 2017].

[31] "Sensor de corriente ACS712", Hispavila.com, 2017. [Online]. Disponible en: https://www.hispavila.com/sensor-de-corriente-asc712/. [Consultado el: 03- Dic- 2017].

[32] "IEEE 802.2-1989 - IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 15.4: Wireless Medium Acces Control (MAC)", Standards.ieee.org, 2003. [Online]. Disponible en: https://standards.ieee.org/findstds/standard/802.2-1989.html. [Consultado el: 25- May- 2018].

[33] "Welcome to Python.org", Python.org, 2017. [Online]. Disponible en: https://www.python.org/. [Consultado el: 15- May- 2017].

[34] "What is a Raspberry Pi?", Opensource.com, 2018. [Online]. Disponible en: https://opensource.com/resources/raspberry-pi. [Consultado: 15- May- 2018].

[35] lm35dz DataSheet disponible en: http://pdf1.alldatasheet.com/datasheetpdf/view/520583/TI1/LM35DZ/LFT1.html [Consultado el 06 may. 2017]

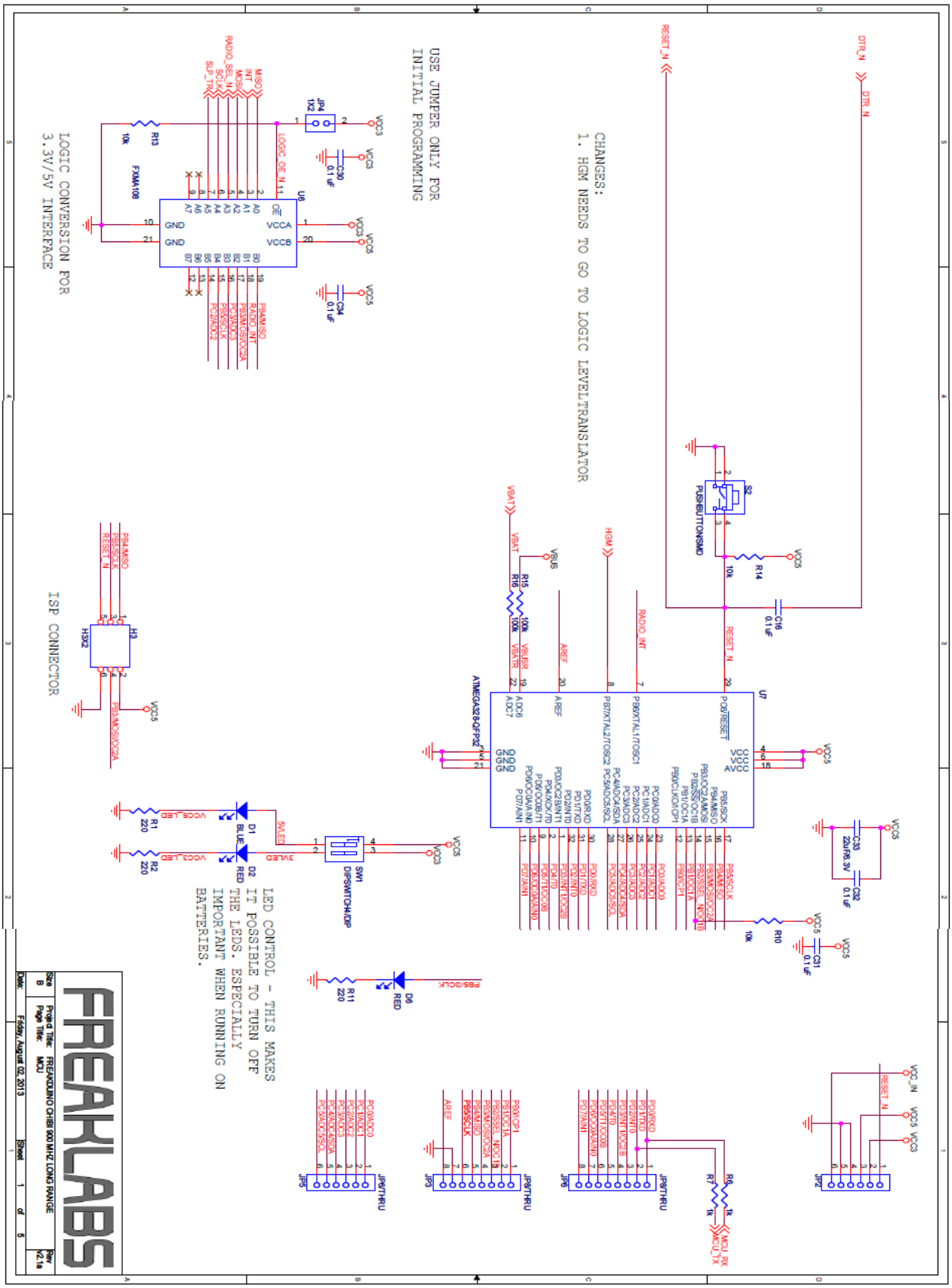
[36] DHT11 DataSheet disponible en: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf [Consultado el 07 nov. 2017]

[37] Módulo RTC DS1302. Reloj de tiempo real. [online] Librearduino.blogspot.mx. Disponible en: http://librearduino.blogspot.mx/2014/01/rtc-arduino-modulo-reloj-tiempo-real-tutorial.html [Consultado el 04 Feb. 2018].

[38] "Fritzing", Fritzing.org. [Online]. Disponible en: http://fritzing.org/home/. [Consultado el: 10- Feb- 2007].

ANEXO(S)

ANEXO A: Esquemáticos de Freakduino

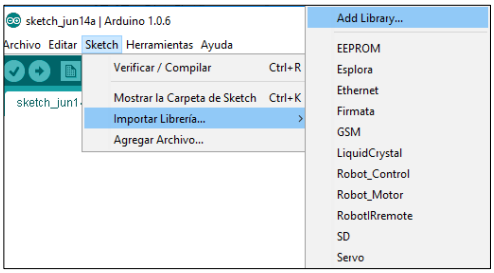


ANEXO B: Manual de operación de la tarjeta Freakduino

Instalación de librerías de Freakduino al entorno Arduino.

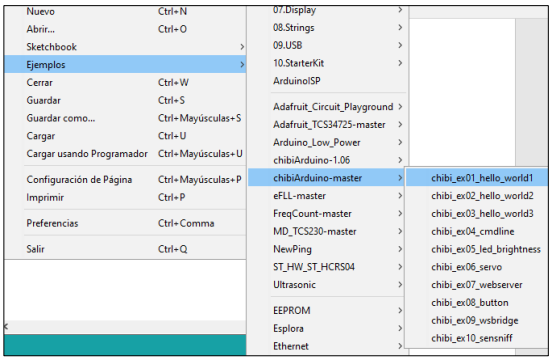
Paso 1: Descargar la librería en formato .zip del siguiente enlace: https://github.com/freaklabs/chibiArduino

Paso 2: En la ventana principal de Arduino seleccionar la pestaña de sketch y hacer clic en importar librería>add Library. Se procede a buscar el archivo .zip y seleccionarlo para que se realice la instalación.



Instalación de librería

Paso 3: Para verificar la instalación de la librería, en la ventana principal de Arduino seleccionar skectch>importar librería y deberá aparecer como chibi Arduino en el menú inferior. O de igual manera al seleccionar los ejemplos deberá aparecer también los ejemplos para chibi Arduino.

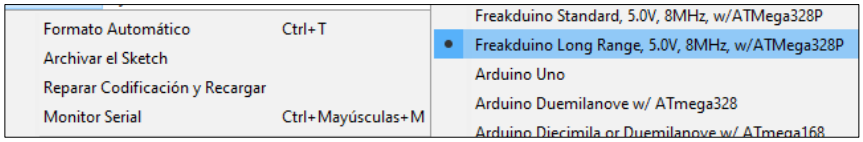


Ejemplos de la librería

Instalación del driver para el funcionamiento de la tarjeta Freakduino

Se procederá a instalar el driver requerido para que la tarjeta sea reconocida. Para ello debe descargarse del siguiente enlace: http://www.freaklabsstore.com/pub/freaklabs_hw.zip

En Windows, ubicar el directorio de Arduino, para que la tarjeta sea compatible con la más reciente versión de Arduino se debe ubicar en documentos>Arduino> y crear una nueva carpeta bajo el nombre de “Hardware”, y dentro de esa carpeta descomprimir el archivo.



Instalación terminada.

Comandos básicos para programación de tarjeta Freakduino

Para una mejor comprensión se explica a continuación los comandos más importantes que utiliza la programación de la tarjeta Freakduino:

- **chibiSetShortAddr(0x1234):** establece la dirección del nodo.
- **chibiGetShortAddr():** obtiene la dirección del nodo.
- **chibiSetChannel(canal de byte):** establece el canal de radio de 11–26.
- **chibiTx(unsigned int address, byte data[], byte len):** transmite.
- **chibiDataRcvd():** regresa verdadero si el radio tiene datos esperando.
- **chibiGetData(destination):** no solo regresa el dato desde el radio y lo arroja en el destinatario, sino que también retorna el largo del dato.
- **chibiSleepRadio():** ningún cero despertata el radio;cero lo mandara a dormir. Esta función en particular es muy importante para sensores alimentados con baterías.

Establecimiento de dirección de nodo y uso del comando de línea

En una red inalámbrica de sensores es importante que cada nodo tenga su identificador, evitando con ello algunos problemas relacionados con la transferencia de datos y que la información a su vez se pierda intentando tomar un canal un solo canal en el aire.

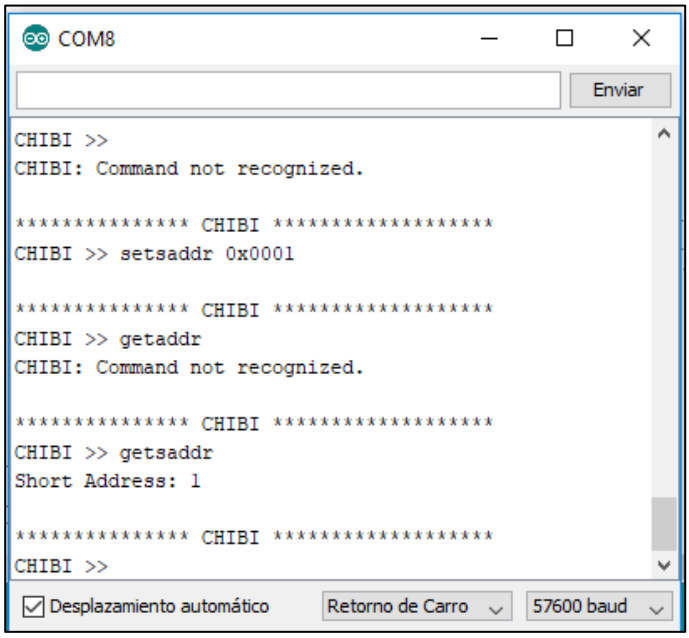
Para el establecimiento de un número de identificador para la tarjeta Freakduino se hace un procedimiento simple en el que se debe cargar el programa de la librería llamado Freakduino-Chibi cmdline, en el cual se le otorga a la tarjeta un comando terminal de línea. Existen tres comandos para obtener, establecer la dirección y mandar información:

- **getsaddr:** obtiene la dirección corta del nodo desde el radio de la eeprom.
- **setsaddr:** establece la dirección corta; por ejemplo: “setsaddr 0x200”, debería establecerlo a hexadecimal 200.
- **send:** manda un mensaje a otro nodo; “send 200 Aquí está mi mensaje” debe mandar “Aquí está mi mensaje” al nodo 200.

Como nota adicional la dirección 0xFFFF es la dirección de emisión por lo que no se debe intentar establecer el identificador del nodo a 0xFFFF y también evitar la dirección 0x0000.

Para acceder al comando de línea se debe usar el programa terminal apropiado. El monitor serial no maneja muy bien las entradas del usuario. Al cargar la aplicación en el monitor serial se establece lo siguiente:

- Data/Baud rate = 9600 (Velocidad de comunicación,)
- Data Bits:8, Parity:none, Stop Bits:1 (Paridad: ninguna, Bits de parada:1)
- Append CR, Local echo OFF (Retorno de carro, eco local desactivado)



Este nodo se estableció con la dirección 0x0001.

Configuración de parámetros mediante la librería de configuración de usuario

Mediante la edición de la librería de configuración del usuario que otorga Freaklabs, es posible optimizar ciertos parámetros correspondientes a ciertas características que permiten a la tarjeta adaptarse a las necesidades del usuario. A continuación, se explican en qué consisten tales parámetros correspondientes al proyecto de tesis.

```
Max payload determina el máximo tamaño de largo que puede transmitirse por trama.
Entero, byte(s); Rango: 1 - 116; Default: 100
Nota: valores arriba de 100 no funcionaran.

#define CHB_MAX_PAYLOAD   100

Esta es la modulación por default para las tarjetas de 900 MHz que usan el
AT86RF231
#define CHB_900MHZ_INIT_MODE OQPSK_SIN_500

Identificador de PAN es el identificador de 16 bits para identificar la red de área
personal(PAN). Solo los dispositivos que comparten el mismo identificador se pueden
comunicar entre sí. Se define el siguiente identificador.

#define CHB_PAN_ID  0x1234

Canal por default con el que el Radio se inicializa. Para la banda de 802.15.4
868/915 MHz, los canales van de 0 a 10. Los canales del 1 hasta el 10 están en la
banda de 915 MHz y pueden usarse bajo licencia libre en la mayor parte de norte
américa.

#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10

El número de veces que el radio reintentara una transmisión. El reintento se activa
cuando un ACK no es recibido en el periodo de tiempo fuera del ACK. Este tiempo es
típicamente de aproximadamente 1 ms.

#define CHB_MAX_FRAME_RETRIES   3 //intentos de transmisión cuando el canal está ocupado

Valor escrito en en el registro de radio. Cada valor corresponde a diferentes potencias
de transmisión en dBm. El valor de 10 dBm equivale a 0XE1.

#define CHB_900MHZ_TX_PWR  0xE1
```

Modo de operación de la tarjeta Freakduino

El manejo de la tarjeta consiste en la inicialización/configuración del nodo y el establecimiento como y cuando la transmisión y recepción de datos tiene lugar.

Inicialización

Lo primero es realizar es la inicialización del nodo. Lo cual queda de la siguiente manera:

```
void setup() {
   chbInit();
   chibiSetShortAddr(0xAAAA);
   chibiSetChannel(15); }
```

De esta forma se configura la dirección del nodo, la cual solo se usa únicamente para la identificación del mismo, y se establece al canal 15. Freakduino utiliza la especificación IEEE 802.15.4 que permite asignar 16 canales en el espectro de los 2.4 GHz de los 11 canales en el espectro de 868/900 MHz. Los canales son definidos como canales de 11 a 26 en 2.4 GHz y 0 a 10 en 868/900 MHz con el canal 0 siendo 868 MHz. Los canales necesitan estar dentro de los rangos propios para el dispositivo seleccionado.

Transmisión

La transmisión de datos requiere de tres cosas: una dirección a transmitir los datos, la información, y el largo de la información. Por ejemplo, ejecutar un simple "Hola Mundo" se escribiría de la siguiente forma:

```
void loop() {
   byte msg[] = "Hola Mundo";
   // 1 byte added to length for terminating character
   chibiTx(0xAAAA, msg, 12); }
```

Como se puede ver el largo del mensaje de Hola Mundo es de 10 bytes pero con un byte extra añadido al largo de la transmisión. Esto es debido a que las cadenas (strings) contienen un carácter nulo (null) arrastrando para la terminación de la cadena. Tiende esto a ser una de las cosas peculiares de la programación

En el ejemplo mostrado anteriormente, el nodo debe transmitir un "Hola Mundo" al nodo cuya dirección es 0xAAAA. El largo informa el final de cuanta información se espera en la recepción. Es también importante el mantener los mensajes por debajo del tamaño máximo de carga para una trama del 802.15.4. Para la pila de chibiArduino el máximo tamaño de carga es de 116 bytes pero se encuentra establecido para 100 bytes en los parámetros de configuración del usuario para mantener un margen.

Otro ejemplo de transmisión Wifi seria como el que se presenta a continuación:

```
   void loop()
{
   byte data[6];
   readAccelerometer(data);
   chibiTx(CHIBI_BROADCAST_ADDR, data, 6);
   delay(500);
}
```

Este código lee información desde un acelerómetro, lo transmite a todos los nodos en la misma red, y luego espera por 500 milisegundos antes de repetir el proceso. CHIBI_BROADCAST_ADDR se define dentro de la pila de chibiArduino y contiene el valor de 0xFFFF. Esta es la dirección reservada por la especificación 802.15.4 para la emisión de la transmisión y todos los nodos en la misma red recibirán la transmisión en esta dirección.

Recepción

La recepción de información es un poco más complicada que la transmisión de la misma y eso es debido a que existe un pequeño control sobre cuando es recibida. La información puede venir en cualquier momento y la función loop necesita estar checando si existe nueva información llegando y manejarla. Este es un ejemplo de cómo hacerlo.

```
   void loop() {
   byte data[100];
   if (chibiDataRcvd() == true) {
   chibiGetData(data);}}
```

Comandos para el mandar a la tarjeta Freakduino al modo reposo (sleep mode) con programación de Arduino.

El modo reposo se utiliza para el ahorro de energía en las tarjetas Arduino. Aunque para algunas tarjetas en realidad no haya mucho beneficio. Como principio global el modo reposo es asistido mediante interrupciones, ya que sin ellas solo un reseteo pudiera despertar al Arduino de nuevo. Afortunadamente las interrupciones son incorporadas desde la versión 0007 del IDE de Arduino

Para esta tarjeta existen 5 modos de reposo disponibles mediante el uso de la librería sleep.h cuyo archivo ya se encuentra disponible para su uso en el directorio de las librerías de Arduino.

- SLEEP_MODE_IDLE
- SLEEP_MODE_ADC
- SLEEP_MODE_PWR_SAVE
- SLEEP_MODE_STANDBY
- SLEEP_MODE_PWR_DOWN

El modo PWR_DOWN es el que otorga el mayor ahorro de energía.
Adicionalmente al modo de reposo, el archivo power.h provee una forma de desconectar cada periférico conectado al CPU de ATMEL.
Eventos sucedidos en el puerto serial también despertarían al Arduino. Para que esto funcione, el Arduino debe estar en modo POWER_MODLE_IDLE, el único modo de energía que no deshabilita el puerto serie. Este modo en realidad no ayuda mucho al ahorro de energía.

Interrupciones de nivel

Cuando el Arduino se encuentra en SLEEP_MODE_PWR_DOWN las únicas formas que existen para despertarlo es con la interrupción del temporizador del perro guardián del microcontrolador, una interrupción en los pines 2 y 3, o un cambio de pin (ver en ATmega328 datasheet tabla 10-1, incluyendo la nota 3 en la página 38).
Una interrupción de nivel significa que el pin debe ser mantenido en un estado por una cierta cantidad de tiempo antes de que la interrupción sea activada. En la rutina de interrupción del servicio (ISR por sus siglas en inglés) para una interrupción de nivel, la interrupción debe ser separada, de otra forma la interrupción seguirá pasando y la ISR será llamada repetidamente hasta que el pin cambie de estado.

```
void pin2_isr(){
 detachInterrupt(0);
 pin2_interrupt_flag = 1;}
```

Como consecuencia, la interrupción debe ser nuevamente habilitada en el código una vez que el pin ha vuelto al estado normal como, por ejemplo, para una interrupción de bajo nivel, hay que checar que el pin ha sido puesto el alto antes de vincular la interrupción de nuevo.

Precaución: El siguiente código es un típico ejemplo de activación del modo reposo (sleep), pudiera causar problema si se confía en la interrupción para despertarlo del modo reposo.

```
attachInterrupt(0, pin2_isr, LOW);
/* 0, 1, o muchas líneas de código aquí */
set_sleep_mode(SLEEP_MODE_PWR_DOWN); //establecer el modo sleep
cli();
sleep_enable(); //habilitando
sleep_bod_disable();
sei();
sleep_cpu();
/* despertar aquí*/
sleep_disable();
```

El problema es que si la interrupción ocurre después de *attachInterrupt* pero antes de sleep_cpu().El ISR se ejecutará, la interrupción será separada , y el CPU entrara en modo reposo con interrupción no habilitada. El MCU nunca se despertará de esta forma. Afortunadamente existe una solución. El bit que habilita el modo sleep puede ser borrado durante el ISR y entonces el MCU no se ira a modo reposo. El comando sleep_enable() también va arriba de la función *attachInterrupts*.

Ejemplo:

```
sleep_enable();
attachInterrupt(0, pin2_isr, LOW);
/* 0, 1, or many lines of code here */
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
cli();
sleep_bod_disable();
sei();
sleep_cpu();
/* Despertar aquí */
sleep_disable();

void pin2_isr(){
  sleep_disable();
  detachInterrupt(0);
  pin2_interrupt_flag = 1;
}
```

ANEXO C: Especificaciones del ADC MCP3208 y resultado de programa de base

Descripción

La tecnología Microchip Inc. de los dispositivos MCP3204/3208 son convertidores analógicos – digitales(A/D) con una consecutiva aproximación de 12-bit. El MCP3208 está programado para proveer cuatro seudo entradas pares diferenciales u ocho entradas únicas. El diferencial no lineal (DNL) esta especificado a ±1 LSB, mientras que el integrador no lineal (INL) es ofrecido en las versiones ±1 LSB (MCP3204/3208-B) y ±2 LSB (MCP3204/3208-C). La comunicación con los dispositivos es lograda usando una simple interfaz compatible con el protocolo SPI. Los dispositivos son capaces de convertir rangos de hasta 100 ksps. Los dispositivos MCP3204/3208 operan sobre un rango de voltaje de (2.7V – 5.5V). El diseño de baja corriente permite la operación con un típico modo espera y corrientes activas de solo 500nA y 320nA respectivamente. El MCP3208 es ofrecido en un encapsulado de 16-pin PDIP y SOIC

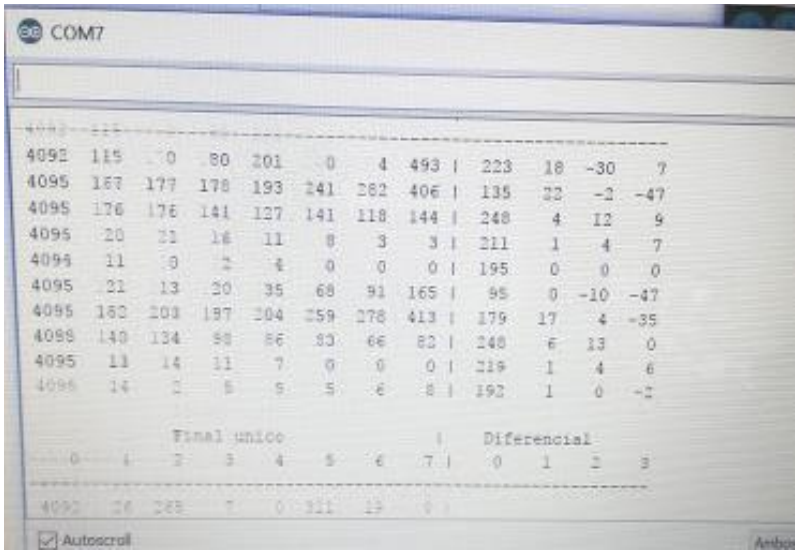
Nombre	**Función**
VDD	+2.7V a 5.5V Voltaje de alimentación
DGND	Tierra digital
AGND	Tierra analógica
CH0-CH7	Entradas analógicas
CLK	Reloj serial
DIN	Datos de entrada serial
DOUT	Datos de salida serial
CS/SHDN Chip	Entrada de Selección/Apagado
VREF	Voltaje de referencia

Características

• 12-bits de resolución
• ± 1 LSB max DNL
• ± 1 LSB max INL (MCP3204/3208-B)
• ± 2 LSB max INL (MCP3204/3208-C)
• 4 (MCP3204) u 8 (MCP3208) canales de entrada
• Entradas analógicas programables como finales únicos o pares seudo diferenciales.
• Muestreo y retención de muestras
• SPI interfaz serial (modos 0,0 and 1,1)
• Unico voltaje de operación: 2.7V - 5.5V
• 100 ksps max. Rango de muestreo a VDD = 5V

• 50 ksps max. Rango de muestreo a VDD = 2.7V
• Tecnología de bajo consumo CMOS:
- Típica corriente de consumo 500 nA, máximo 2 µA.
- Corriente máxima activa 400 µA max. a 5V
• Rango de temperatura industrial: -40°C hasta +85°C
• Disponible en encapsulado PDIP, SOIC y TSSOP

Mediante la implementación del programa de interfaz entre el MCP3208 y el Arduino se obtuvieron los siguientes resultados:

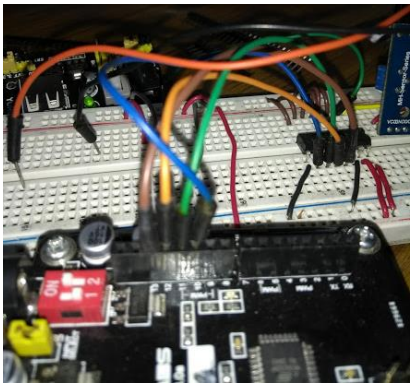


Monitoreo de canales del ADC.

ANEXO D: DISEÑO DE PCB SHIELD PARA FREAKDUINO PARA IMPLEMENTAR MCP3208, BASE DE SENSORES Y GPS.

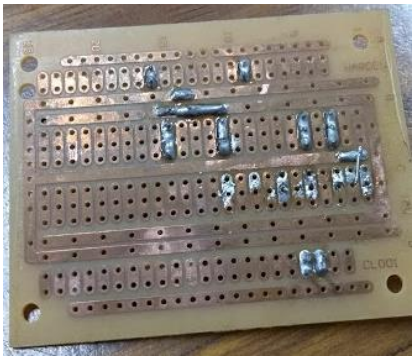
Para el desarrollo del pcb se utilizó un software de acceso libre que permitió de forma muy intuitiva desarrollar un shield que pudiera acoplarse a la Freakduino.

Primero se hicieron las pruebas mediante una tarjeta para prototipos y sus respectivas conexiones con el MCP3208.



Prueba en tarjeta de prototipos.

Antes de poder llegar a ese diseño se procedió a diseñar de forma manual por medio de pcbs ya perforados los primeros prototipos para las pruebas como se muestra en la siguiente figura.

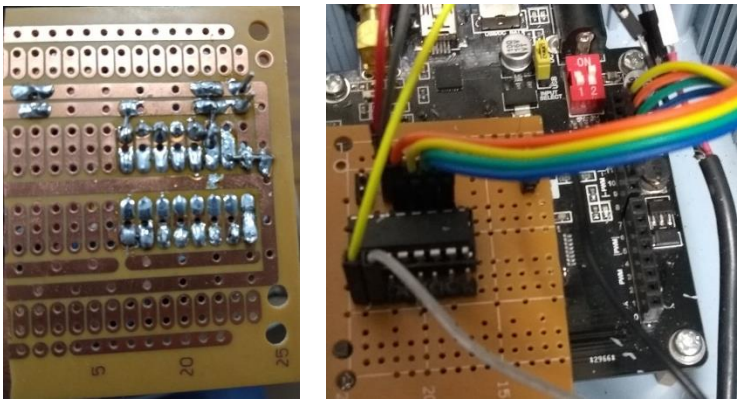


Vista trasera del PCB perforado.

El primer prototipo se probó con la Freakduino, integrando el sensor de humedad FC-28 teniendo como objetivo principal la transmisión de datos mediante la integración del MCP3208 y probar la potencia de la señal RSSI entre nodos. En la siguiente figura se muestra el PCB acoplado.

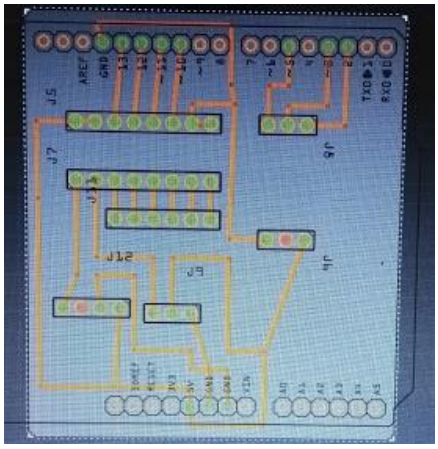
Freakduino con MCP3208 y sensores.

Se hizo un nuevo diseño más reducido para el nodo 2 y realizar nuevas pruebas quedando como se muestra en la siguiente figura.

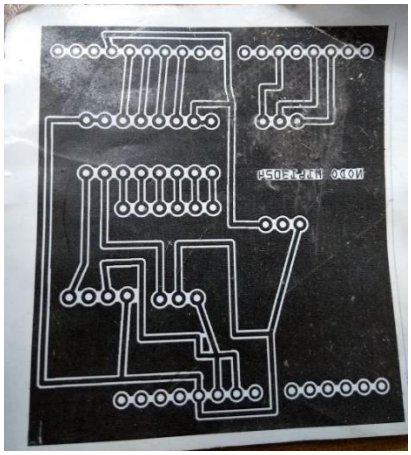


Diseño de PCB reducido.

El diseño PCB que se realizó mediante Fritzing ya incluye más características para más sensores y el GPS, primero se muestra el desarrollo en software y luego en físico quedando como se ve en las siguientes figuras:



Diseño en Fritzing.



Diseño en papel de transferencia.

Elaboración de prototipos.

Prototipo PCB sin conectar.

ANEXO E: PROGRAMAS DE LAS PRUEBAS DE POTENCIA RSSI

Programa Receptor para la prueba con lm35dz.

```
#include <chibi.h> //Librería de la tarjeta Freakduino
byte buf[CHB_MAX_PAYLOAD]; //Tamaño de carga útil máxima de 116 bytes por default 100 para evitar sobrecargas
int val=0; //para conversión de unidades
//-----Variables para sensor de temperatura lm35-----
//int TempOffset = 500; // valor en mV cuando el ambiente es 0 grados Celcius
//int TempCoef = 10; // Coeficiente de temperatura en mV por grados Celcius
float ADCmV = 4.8828; // mV por incremento del ADC(5 volts / 1024 incrementos)
float Temp = 0; // calcular temperatura en Celcius
void setup()
{
  Serial.begin(9600);
  chibiSetShortAddr(3); //dirección Nodo receptor
  chibiInit(); //Inicializar el NODO
}
void loop()
{
// Código principal donde se visualizan datos recibidos,
// si existen, procesarlos.
if (chibiDataRcvd() == true)
{
int x; //variable en la que se almacenara dato del sensor
uint8_t len, buf[20]; //entero sin signo de 8 bits
//------RSSI Y DIRECCIÓN DE NODO----------------
int rssi, src_addr; //variables para RSSI y dirección del nodo proveniente
// obtener la fuerza se la señal y la dirección de donde proviene
  rssi = chibiGetRSSI();
```

```
  src_addr = chibiGetSrcAddr();
//-------------------------------------------------
  val=rssi*-1; //Conversión
//--------datos del sensor y su interpretación------
  x=atoi((char*)buf); //cadena de caracteres a numero
  Temp = ((x*ADCmV*100)/1024);
//Temp = ((x * ADCmV) - TempOffset) / -TempCoef;
delay(200);
unsigned int rcv_data; //para usarse...
chibiGetData(buf);
 //------------Impresión de datos----------------
      Serial.print("Temperatura: ");
      Serial.print(Temp);
      //Serial.println((char *)buf);
      Serial.print(", Mensaje recibido del nodo 0x");
      Serial.print(src_addr, HEX);
      Serial.print(", RSSI = 0x");
      Serial.println(rssi,HEX);
      Serial.print("RSSI dB =");
      Serial.println(val,DEC);
   }}}
```

Programa Transmisor para la prueba conlm35dz.

```
#include <Wire.h>
//#include <Sleep.h> //libreía para poner al nodo a dormir
#include <chibi.h> //Librería de la tarjeta Freakduino
// Variables del proyecto
int A0raw; //Pin analogico 0 para el sensor
const int bufSize = 10; //tamaño de buffer
byte XmitBuf[bufSize]; // transmisión de bufSize
//--------------------------
void setup() {
Serial.begin(9600);
//Serial.begin(57600);
chibiInit(); //Inicialización del NODO
}
void loop() {
char temp [bufSize]; //almacenar el buffer para caracter temp
A0raw = analogRead(A0);//Lectura del sensor en A0
sprintf(temp,"%i",A0raw); //numeros a cadenas, formato número decimal, valor de lectura del sensor en
A0
memcpy(XmitBuf, temp, bufSize); //copia datos en memoria
//delay(100);
chibiTx(BROADCAST_ADDR, XmitBuf, bufSize);//chibiTx(nodo anfitrion (3),XmitBuf,bufSize)
Serial.println(temp);
}
```

Programa Receptor para la prueba con DHT11.

```
#include <chibi.h>//Librería de la tarjeta Freakduino
#include <chibiUsrCfg.h>//Librería de configuración de usuario

/*RSSI; Indicador de la fuerza de la señal recibida, una escala de referencia (en relación a 1 mW)
para medir el nivel de potencia de las señales recibidas por un dispositivo en las redes inalámbricas
(típicamente WIFI)
RSSI indica intensidad y no calidad de la señal.
->dBm decibeliomilivatio es una unidad de potencia expresada en decibelios (dB) relativa a un milivatio
(mW)
Muy usada para expresar la medida absoulta de la potencia. (dB-> es una unidad adimensional y usada para
cuantificación de datos)
->DBi. Son los Decibeles de ganancia sobre un radiador isotrópico o una
Relación logarítmica entre la potencia de emisión de una antena en relación a un radiador isotrópico. Es la
directividad de la antena.
Utilizando uint8_t chibiGetRSSI(): Obtiene la potencia de la señal desde el ultimo marco de datos
recibidos.
 El rango de la potencia de la señal es:
 84 (-7 dB ) a 0 (-91 dB) para el AT86RF230
 con una precisión de +/- 5 dB.
*/
//Código para nodo receptor
//COM8
//Configuración de parametros para la tarjeta Freakduino--------//
//#ifndef CHB_USR_CFG_H
//#define CHB_USR_CFG_H
//#include <src/chb_drvr.h> // necesaria para modulación OQPSK_SIN
//#include <chibiUsrCfg.h> //necesario para la confoguración del usuario
//#define  CHB_900MHZ_INIT_MODE OQPSK_SIN_1000 //modulacion
//#define  CHB_900MHZ_INIT_MODE BPSK_40
//#define CHB_PAN_ID  0x1234 //Identificador de red, dispositivos que deben comunicarse deben tener
mismo pan id
//#define CHB_900MHZ_TX_PWR  0xE1
//#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10
//#define CHB_MAX_PAYLOAD   100
//#endif
//------------termina configuración de parametros-------------//
//Distancia maxima que alcanza a recibir datos 200 metros al 30/11/17
byte buf[CHB_MAX_PAYLOAD]; //Tamaño de carga útil máxima de 116 bytes pero por default 100 para
evitar sobrecargas
int val=0; //para conversión de unidades
void setup()
{
Serial.begin(9600);
chibiInit(); //Inicializar el NODO
```

```
//Serial.print("...");
}
void loop()
{
// Código principal donde se visualizan datos recibidos,
// si existen, procesarlos.
if (chibiDataRcvd() == true)
{
//int x; //variable en la que se almacenara dato del sensor
uint8_t len, buf[12]; //entero sin signo de 8 bits
//------POTENCIA RSSI Y DIRECCIÓN DE NODO----------------------------------------------------------------//
int rssi, src_addr; //variables para RSSI y dirección del nodo proveniente

// obtener los datos y el largo de la información recibida
  len = chibiGetData(buf);
  if (len == 0) return;
// Obtención de la fuerza de la señal y la dirección del nodo provienente //
  rssi = chibiGetRSSI();
  src_addr = chibiGetSrcAddr();
//---------Conversión de valor RSSI en dB---------------------------------//
  val=rssi*+1; //Conversión en dBm
//--------Datos del sensor y su interpretación (solo lm35dz)---------//
//  x=atoi((char*)buf); //cadena de caracteres a numero
// Temp = ((x*ADCmV*100)/1024);
//(no)Temp = ((x * ADCmV) - TempOffset) / -TempCoef;
//-----------------------------------------------------------------//
delay(100);
unsigned int rcv_data; //para usarse...
chibiGetData(buf);
//------------Impresión de datos-----------------------------------//
//Serial.print("Temperatura: ");
//Serial.print(Temp);
Serial.print((char *)buf);
Serial.print(", Mensaje recibido del nodo 0x");
Serial.print(src_addr, HEX);
Serial.print(", RSSI en HEX = 0x");
Serial.println(rssi,HEX);
Serial.print("RSSI =");
Serial.println(val,DEC);
//-----------------------------------------------------------------//
}}
```

Programa Transmisor para la prueba con DHT11.

```
//Librería de la tarjeta Freakduino//
#include <chibi.h>
```

```
#include <chibiUsrCfg.h>
//---------------------------------//
//Librería de ADC MCP3208----------//
#include <MCP3208.h>
#include <SPI.h>
MCP3208 adc(10);
//---------------------------------//
#include <Wire.h> //Comunicación I2C
const int bufSize = 10; //tamaño de buffer
byte XmitBuf[bufSize]; // transmisión de bufSize
//Sensor de corriente ACS712----------------------------------//
//float Sensibilidad=0.185;//sensibilidad para modelo de 5A
////Libería RTC Reloj de tiempo real y configuración---------// //*Se comentara el RTC para pasarlo al
código del receptor
//#include <DS1302.h>
//// Inicializacion del modulo.
//DS1302 rtc(2, 3, 5);
//Time t;
//-----------------------------------------------------------//
void setup() {
Serial.begin(9600);//Velocidad de comunicación Serial
chibiInit(); //Inicialización del NODO
adc.begin();//Ininicalización del ADC
}
void loop() {
////-----RTC--------//
// t = rtc.getTime();
////----------------//
//-----SENSOR HUMEDAD FREAKDUINO-------------------------------------------------
char humedad [bufSize];
int valor_humedaduno = map(adc.analogRead(0),0,4095,100,0); //recordando que se esta utilizando un
ADC de 12 bits por lo que en lugar de 1023 steps son 4095
int valor_humedaddos = map(adc.analogRead(1),0,4095,100,0);
sprintf(humedad,"%i",valor_humedaduno);

memcpy(XmitBuf, humedad, bufSize);
chibiTx(BROADCAST_ADDR, XmitBuf,bufSize);
//----------------------------------------------------------------
//--------------Imprimir variables----------------------------//
//if (t.dow == 1) Serial.print("lun_");  // La variable t.dow (dia de la semana) tedra valor de 1 para dia
lunes y 7 para domingo.
//  if (t.dow == 2) Serial.print("mar_");
//  if (t.dow == 3) Serial.print("mie_");
//  if (t.dow == 4) Serial.print("jue_");
//  if (t.dow == 5) Serial.print("vie_");
//  if (t.dow == 6) Serial.print("sab_");
```

```
  //  if (t.dow == 7) Serial.print("dom_");
  //  Serial.print(t.date, DEC);  // Dia del mes
  //  Serial.print("/");
  //  Serial.print(t.mon);
  //  Serial.print("/");
  //  Serial.print(t.year, DEC);
  //  Serial.print(" HORA:");  // Hora en formato 0-23.
  //  Serial.print(t.hour, DEC);
  //  Serial.print(":");  // Minutos.
  //  Serial.print(t.min, DEC);
  //  Serial.print(":");  // Segundos.
  //  Serial.print(t.sec, DEC);
  //  Serial.println();
  //Serial.print("Humedad sensor uno: ");
  //Serial.print(valor_humedaduno);
  //Serial.println("%");
  //Serial.print("Humedad sensor dos: ");
  //Serial.print(valor_humedaddos);
  //Serial.println("%");
  delay(3000);}
```

Programa Receptor para la prueba con FC-28.

```
//Diseño: Ing. Fer Alvarez M1613024
#include "chibi.h"//Librería de la tarjeta Freakduino
#include "chibiUsrCfg.h"//Librería de configuración de usuario
#define PAYLOAD_SIZE 100
//#define  CHB_900MHZ_INIT_MODE BPSK_40
#define  CHB_900MHZ_INIT_MODE OQPSK_SIN_250 //modulación
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR 0xE1
#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10
#define CHB_MAX_FRAME_RETRIES   3 //intentos de transmisión cuando el canal esta ocupado
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR  0xE1
//Libería RTC Reloj de tiempo real y configuración---------//
#include <DS1302.h>
// Inicializacion del modulo RTC
DS1302 rtc(2, 3, 5);
Time t;
//-------------------------------------------------------//
//int b; //variable de alternación... (prueba)
//-----------------------------------//
/*RSSI; Indicador de la fuerza de la señal recibida, una escala de referencia (en relación a 1 mW)
para medir el nivel de potencia de las señales recibidas por un dispositivo en las redes inalámbricas (típicamente
WIFI)
RSSI indica intensidad y no calidad de la señal.
```

->dBm decibeliomilivatio es una unidad de potencia expresada en decibelios (dB) relativa a un milivatio (mW) Muy usada para expresar la medida absoulta de la potencia. (dB-> es una unidad adimensional y usada para cuantificación de datos)
->DBi. Son los Decibeles de ganancia sobre un radiador isotrópico o una
Relación logarítmica entre la potencia de emisión de una antena en relación a un radiador isotrópico. Es la directividad de la antena.
Utilizando uint8_t chibiGetRSSI(): Obtiene la potencia de la señal desde el ultimo marco de datos recibidos.
El rango de la potencia de la señal es:
84 (-11.48 dBm) a 0 (-96.97 dBm) para el AT86RF212
con una precisión de +/- 5 dB.
--> Utilizando el radio para 869.4 MHz hay que establecer dos registros:
CC_CTRL_1(address 0x14). Los 3 bits de fondo establecen la banda de radio frecuencia.
Para la banda de 857 a 882.5 MHz se necesita establecer a 2.
Esto se pone en la función setup()
*/
//Código para nodo receptor
//COMX
//------------termina configuración de parametros-------------//
//Distancia maxima que alcanza a recibir datos 200 metros al 30/11/17
byte buf[CHB_MAX_PAYLOAD]; //Tamaño de carga útil máxima de 116 bytes pero por default 100 para evitar sobrecargas
int val=0; //para conversión de unidades
void setup()
{
Serial.begin(57600);
chibiInit(); //Inicializar el NODO
chibiSetShortAddr(0x0003); //Establecer dirección 3
chibiRegWrite(0x14,0x02);//Establecer la banda de frecuencia
chibiRegWrite(0x13,0xff);//Establecer los registros para sintonizar
//Serial.print("...");
//----------------Configurar FECHA Y HORA con RTC MANUAL-------------------------//
  rtc.halt(false);
  rtc.writeProtect(false);
  rtc.setDOW(THURSDAY); // Configurar dia de la semana: MARTES.
  rtc.setTime(18, 20, 00); // Configurar hora en formato 24hs con min y seg: 17:00:00 HORAS.
  rtc.setDate(22, 2, 2018);
}
void loop()
{
//-----RTC--------//
 t = rtc.getTime();
//----------------//
// Código principal donde se visualizan datos recibidos,
// si existen, procesarlos.
if (chibiDataRcvd() == true)
{

```
//uint8_t len, buf[12]; //entero sin signo de 8 bits
//------POTENCIA RSSI Y DIRECCIÓN DE NODO----------------------------------------------------------------//
int rssi, src_addr; //variables para RSSI y dirección del nodo proveniente
// obtener los datos y el largo de la información recibida
 byte len = chibiGetData(buf);
  if (len == 0) return;

// Obtención de la fuerza de la señal y la dirección del nodo provienente //
  rssi = chibiGetRSSI();
  src_addr = chibiGetSrcAddr();
//---------Conversión de valor RSSI en dB---------------------------------//
  val=rssi*+1; //Conversión en dBm
delay(100);
unsigned int rcv_data; //para usarse...
//probando aun//
chibiGetData(buf);
//if (Serial.read()=='b'){
//      chibiTx(0x0005, buf, b);
//      delay(500);
//    }
//------------Impresión de datos-----------------------------------//
if (t.dow == 1) Serial.print("lun_");  // La variable t.dow (dia de la semana) tedra valor de 1 para dia lunes y 7
para domingo.
  if (t.dow == 2) Serial.print("mar_");
  if (t.dow == 3) Serial.print("mie_");
  if (t.dow == 4) Serial.print("jue_");
  if (t.dow == 5) Serial.print("vie_");
  if (t.dow == 6) Serial.print("sab_");
  if (t.dow == 7) Serial.print("dom_");
  Serial.print(t.date, DEC);  // Dia del mes
  Serial.print("/");
  Serial.print(t.mon);
  Serial.print("/");
  Serial.print(t.year, DEC);
  Serial.print(" HORA:");  // Hora en formato 0-23.
  Serial.print(t.hour, DEC);
  Serial.print(":");  // Minutos.
  Serial.print(t.min, DEC);
  Serial.print(":");  // Segundos.
  Serial.print(t.sec, DEC);
  Serial.println();
  Serial.print("humedad: ");
  Serial.print((char *)buf);
  Serial.print(" %");
//Serial.print(", Mensaje recibido del nodo 0x");
Serial.print(", Mensaje del nodo 0x");
```

```
Serial.print(src_addr, HEX);
Serial.print(", RSSI en HEX = 0x");
Serial.println(rssi,HEX);
Serial.print("RSSI =");
Serial.println(val,DEC);
//-----------------------------------------------------------------//
}}
```

Programa Transmisor para la prueba con FC-28.

```
//Diseño: Ing. Fer Alvarez M1613024
//Librería de la tarjeta Freakduino//
#include "chibi.h"
#include "chibiUsrCfg.h"
//prueba de modificación de parametros según lo observado en el blog de freakduino
#define PAYLOAD_SIZE 100
#define  CHB_900MHZ_INIT_MODE OQPSK_SIN_250
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10
#define CHB_MAX_FRAME_RETRIES   3 //intentos de transmisión cuando el canal esta ocupado
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR  0xE1
//--------------------------------//
//Librería de ADC MCP3208----------//
#include <MCP3208.h>
#include <SPI.h>
MCP3208 adc(10);
//--------------------------------//
byte buf[PAYLOAD_SIZE];
#include <Wire.h> //Comunicación I2C
const int bufSize = 10; //tamaño de buffer
byte XmitBuf[bufSize]; // transmisión de bufSize
//--------------------------------//
//float valor;
////Libería RTC Reloj de tiempo real y configuración---------// //*Se comentará el RTC para pasarlo al código
del receptor
//#include <DS1302.h>
//// Inicializacion del módulo.
//DS1302 rtc(2, 3, 5);
//Time t;
//-----------------------------------------------------------//
void setup() {
Serial.begin(57600);//Velocidad de comunicación Serial
chibiInit(); //Inicialización del NODO
chibiSetShortAddr(5); //Establecer dirección 3
//chibiRegWrite(0x14,0x02);//Establecer la banda de frecuencia
//chibiRegWrite(0x13,0xff);//Establecer los registros para sintonizar
```

```
adc.begin();//Ininicalización del ADC
//chibiRegWrite(0x14, 0x02);
//chibiRegWrite(0x13, 0x7C);
}
void loop() {
//if(chibiDataRcvd() == true)
//{
// int len = chibiGetData(buf);
//    if (len == 0) return;
//    TXnodo;
//    }
//if (Serial.read()=='b'){
//
//bateria();}
//
//else
//{char b;
//  int len = chibiGetData(buf);
//   if (len== b)
//    { bateria();
//    }
////-----RTC--------//
// t = rtc.getTime();
////----------------//
//-----SENSOR HUMEDAD FREAKDUINO-------------------------------------------------
char humedad [bufSize];
//char humedad_dos[bufSizeh];
//char humedad_dos [bufSizeh];
int valor_humedaduno = map(adc.analogRead(0),0,4095,100,0); //recordando que se esta utilizando un ADC de 12 bits por lo que en lugar de 1023 steps son 4095
//int valor_humedaddos = map(adc.analogRead(1),0,4095,100,0);
//sprintf(humedad,"%i,%i",valor_humedaduno,valor_humedaddos); //para mandar datos de mas sensores en el mismo paquete... (nombre del char, "%i,%x,%d....", lectura_sensor,lectura,sectura_sensor);
sprintf(humedad,"%i",valor_humedaduno);// ultimo usado un solo valor
//sprintf(humedad,"%i,%i",valor_humedaduno,dtostrf(ValorSensor,5,2,humedad));
//sprintf(humedad,"%i,%i",valor_humedaduno,ValorSensor);
//dtostrf(ValorSensor,5,2,humedad);
memcpy(XmitBuf, humedad, bufSize);
chibiTx(3, XmitBuf,bufSize); //transmitir al nodo 3
//memcpy(XmitBufh, humedad_dos, bufSizeh);
//chibiTx(BROADCAST_ADDR, XmitBufh,bufSizeh);
//-----------------------------------------------------------------
//--------------Imprimir variables----------------------------//
//if (t.dow == 1) Serial.print("lun_");  // La variable t.dow (dia de la semana) tedra valor de 1 para dia lunes y 7 para domingo.
//  if (t.dow == 2) Serial.print("mar_");
```

```
//  if (t.dow == 3) Serial.print("mie_");
//  if (t.dow == 4) Serial.print("jue_");
//  if (t.dow == 5) Serial.print("vie_");
//  if (t.dow == 6) Serial.print("sab_");
//  if (t.dow == 7) Serial.print("dom_");
//  Serial.print(t.date, DEC);  // Dia del mes
//  Serial.print("/");
//  Serial.print(t.mon);
//  Serial.print("/");
//  Serial.print(t.year, DEC);
//  Serial.print(" HORA:");  // Hora en formato 0-23.
//  Serial.print(t.hour, DEC);
//  Serial.print(":");  // Minutos.
//  Serial.print(t.min, DEC);
//  Serial.print(":");  // Segundos.
//  Serial.print(t.sec, DEC);
//  Serial.println();
Serial.print("Humedad: ");
Serial.print(valor_humedaduno);
Serial.println("%");
//Serial.print("Humedad sensor dos: ");
//Serial.print(valor_humedaddos);
//Serial.println("%");
//delay(3000);
//Serial.print("batería: ");
//Serial.println(ValorSensor);
delay(5000); //mandara dato cada 5 segundos
//delay(300000); //cada 5 minutos
}
//}
//Voltaje actual de la batería-----------------------------//
//void bateria(){
//char humedad [bufSize];
//float ValorSensor = fmap(adc.analogRead(2),0,4095,0.0,25.0);
//dtostrf(ValorSensor,5,2,humedad);
//memcpy(XmitBuf, humedad, bufSize);
//chibiTx(0x0003, XmitBuf,bufSize);
//Serial.print("batería: ");
//Serial.println(ValorSensor);
//delay(5000); //mandara dato cada 5 segundos
//
//}
// cambio de escala entre floats para sensor de voltaje
//float fmap(float x, float in_min, float in_max, float out_min, float out_max)
//{
//   return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
```

```
//}
//void TXnodo(int arg_cnt,char **args){
//  byte data[100];
//   int addr, len;
//    // convert cmd line string to integer with specified base
//    addr = chibiCmdStr2Num(args[1], 16);
//
//    // concatenate strings typed into the command line and send it to
//    // the specified address
//    len = strCat((char *)data, 2, arg_cnt, args);
//    chibiTx(0x0003, data,len);
//}
//int strCat(char *buf, unsigned char index, char arg_cnt, char **args)
//{
//    uint8_t i, len;
//    char *data_ptr;
//    data_ptr = buf;
//    for (i=0; i<arg_cnt - index; i++){
//        len = strlen(args[i+index]);
//        strcpy((char *)data_ptr, (char *)args[i+index]);
//        data_ptr += len;
//        *data_ptr++ = ' ';
//    }
//    *data_ptr++ = '\0';
//    return data_ptr - buf;}
```

Programa Receptor para la prueba con VH400.

```
//Diseño: Ing. Fer Alvarez M1613024
#include "chibi.h"//Librería de la tarjeta Freakduino
#include "chibiUsrCfg.h"//Librería de configuración de usuario
#define PAYLOAD_SIZE 100
//#define  CHB_900MHZ_INIT_MODE BPSK_40
#define  CHB_900MHZ_INIT_MODE OQPSK_SIN_250 //modulación
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR 0xE1
#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10
#define CHB_MAX_FRAME_RETRIES   3 //intentos de transmisión cuando el canal esta ocupado
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR  0xE1
//Libería RTC Reloj de tiempo real y configuración---------//
#include <DS1302.h>
// Inicializacion del modulo RTC
DS1302 rtc(2, 3, 5);
Time t;
//-------------------------------------------------------//
```

```
//int b; //variable de alternación... (prueba)
//Código para nodo receptor
//COMX
//------------termina configuración de parametros-------------//
//Distancia maxima que alcanza a recibir datos 200 metros al 30/11/17
byte buf[CHB_MAX_PAYLOAD]; //Tamaño de carga útil máxima de 116 bytes pero por default 100 para evitar
sobrecargas
int val=0; //para conversión de unidades
void setup()
{
Serial.begin(57600);
chibiInit(); //Inicializar el NODO
chibiSetShortAddr(0x0003); //Establecer dirección 3
chibiRegWrite(0x14,0x02);//Establecer la banda de frecuencia
chibiRegWrite(0x13,0xff);//Establecer los registros para sintonizar
//Serial.print("...");
//----------------Configurar FECHA Y HORA con RTC MANUAL-------------------------//
  rtc.halt(false);
  rtc.writeProtect(false);
  rtc.setDOW(THURSDAY);  // Configurar dia de la semana: MARTES.
  rtc.setTime(18, 20, 00);  // Configurar hora en formato 24hs con min y seg: 17:00:00 HORAS.
  rtc.setDate(22, 2, 2018);
}
void loop(){
//-----RTC--------//
 t = rtc.getTime();
//----------------//
// Código principal donde se visualizan datos recibidos,
// si existen, procesarlos.
if (chibiDataRcvd() == true)
{
//uint8_t len, buf[12]; //entero sin signo de 8 bits
//------POTENCIA RSSI Y DIRECCIÓN DE NODO-----------------------------------------------------------------//

int rssi, src_addr; //variables para RSSI y dirección del nodo proveniente

// obtener los datos y el largo de la información recibida
 byte len = chibiGetData(buf);
  if (len == 0) return;

// Obtención de la fuerza de la señal y la dirección del nodo provienente //
  rssi = chibiGetRSSI();
  src_addr = chibiGetSrcAddr();
//---------Conversión de valor RSSI en dB---------------------------------//
  val=rssi*+1; //Conversión en dBm
delay(100);
```

```
unsigned int rcv_data; //para usarse...
//probando aun//
chibiGetData(buf);
//if (Serial.read()=='b'){
//
//      chibiTx(0x0005, buf, b);
//      delay(500);
//    }
//
//------------Impresión de datos----------------------------------//
if (t.dow == 1) Serial.print("lun_");  // La variable t.dow (dia de la semana) tedra valor de 1 para dia lunes y 7
para domingo.
  if (t.dow == 2) Serial.print("mar_");
  if (t.dow == 3) Serial.print("mie_");
  if (t.dow == 4) Serial.print("jue_");
  if (t.dow == 5) Serial.print("vie_");
  if (t.dow == 6) Serial.print("sab_");
  if (t.dow == 7) Serial.print("dom_");
  Serial.print(t.date, DEC);  // Dia del mes
  Serial.print("/");
  Serial.print(t.mon);
  Serial.print("/");
  Serial.print(t.year, DEC);
  Serial.print(" HORA:");  // Hora en formato 0-23.
  Serial.print(t.hour, DEC);
  Serial.print(":");  // Minutos.
  Serial.print(t.min, DEC);
  Serial.print(":");  // Segundos.
  Serial.print(t.sec, DEC);
  Serial.println();
  Serial.print("humedad: ");
  Serial.print((char *)buf);
  Serial.print(" %");
//Serial.print(", Mensaje recibido del nodo 0x");
Serial.print(", Mensaje del nodo 0x");
Serial.print(src_addr, HEX);
Serial.print(", RSSI en HEX = 0x");
Serial.println(rssi,HEX);
Serial.print("RSSI =");
Serial.println(val,DEC);
//-----------------------------------------------------------------//
}}
```

Programa Transmisor para la prueba con VH400.

```
//Diseño: Ing. Fer Alvarez M1613024
//Librería de la tarjeta Freakduino----------//
#include "chibi.h"
#include "chibiUsrCfg.h"
//Establecimiento de parametros según lo permitido en la configuración
#define PAYLOAD_SIZE 100
#define CHB_900MHZ_INIT_MODE OQPSK_SIN_250 //250kbps (recomendado)
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_DEFAULT_CHANNEL 10 //Utilizando canal 10
#define CHB_MAX_FRAME_RETRIES   3 //intentos de transmisión cuando el canal esta ocupado
#define CHB_PAN_ID  0x1234
#define CHB_900MHZ_TX_PWR  0xE1
//----------//
//Librería de ADC MCP3208---------//
#include <MCP3208.h>
#include <SPI.h>
MCP3208 adc(10);
//--------------------------------//
byte buf[PAYLOAD_SIZE];
#include <Wire.h> //Comunicación I2C
const int bufSize = 10; //tamaño de buffer
byte XmitBuf[bufSize]; // transmisión de bufSize

//Librerías para dormitar al nodo----//
#include <avr/interrupt.h>
#include <avr/sleep.h>
//----------------------------------//

////Libería RTC Reloj de tiempo real y configuración---------// //*Se comentará el RTC para pasarlo al código
del receptor
//#include <DS1302.h>
//// Inicializacion del modulo RTC
//DS1302 rtc(2, 3, 5);
//Time t;
//----------//
void setup() {
Serial.begin(57600);//Velocidad de comunicación Serial
chibiInit(); //Inicialización del NODO
chibiSetShortAddr(0x0003); //Establecer dirección 3
//chibiRegWrite(0x14,0x02);//Establecer la banda de frecuencia
//chibiRegWrite(0x13,0xff);//Establecer los registros para sintonizar
adc.begin();//Ininicalización del ADC
}
void loop() {
```

```
//Funciones a ejecutar del nodo
//1)Lectura 2)Despertar al nodo 3)Transmitir 4)dormir nodo 5)...
lectura_sensor();
despertarNodo();
transmitir();
dormirNodo();
//-------------------------------------------------------------//
//si existen datos de otro nodo retransmitir
if(chibiDataRcvd() == true)
{
 int len = chibiGetData(buf);
    if (len == 0) return;
    TXnodo;
}
//---Alternando los datos a transmitir entre humedad y batería//
//Prueba.....................
if (Serial.read()=='b'){
bateria();
}
else
{ char b;
  int len = chibiGetData(buf);
   if (len== b)
    { bateria();
    }
}}
////-----RTC--------//
// t = rtc.getTime();
////----------------//
//-----SENSOR HUMEDAD FREAKDUINO-----------------------------------------------
void lectura_sensor(){
char humedad [bufSize];
int sensorVoltaje= adc.analogRead(0); //leyendo voltaje de entrada
float valor = sensorVoltaje*(5 / 4095.0);
float VWC;
//
//  // Calcular el  VWC (contenido volumétrico del agua
  if(valor <= 1.1) {
    VWC = 10*valor-1;
  } else if(valor > 1.1 && valor <= 1.3) {
    VWC = 25*valor-17.5;
  } else if(valor > 1.3 && valor <= 1.82) {
    VWC = 48.08*valor-47.5;
  } else if(valor > 1.82 && valor <= 2.2) {
    VWC = 26.32*valor-7.89;
  } else if(valor > 2.2 && valor <= 3.0) {
```

```
    VWC = 62.5*valor-87.5;
  }
  Serial.println(VWC);
  //return(VWC);
  sprintf(humedad,"%i",valor);// ultimo usado un solo valor
  memcpy(XmitBuf, humedad, bufSize);
  delay(5000);
Serial.print("Humedad sensor uno: ");
Serial.print(valor);
Serial.println("%");
}
void transmitir(){
  chibiTx(0x0003, XmitBuf,bufSize); //transmitir al nodo 3  }
//---------------------------------------------------------------
//--------------Imprimir variables----------------------------//
//if (t.dow == 1) Serial.print("lun_");  // La variable t.dow (dia de la semana) tedra valor de 1 para dia lunes y
7 para domingo.
//  if (t.dow == 2) Serial.print("mar_");
//  if (t.dow == 3) Serial.print("mie_");
//  if (t.dow == 4) Serial.print("jue_");
//  if (t.dow == 5) Serial.print("vie_");
//  if (t.dow == 6) Serial.print("sab_");
//  if (t.dow == 7) Serial.print("dom_");
//  Serial.print(t.date, DEC);  // Dia del mes
//  Serial.print("/");
//  Serial.print(t.mon);
//  Serial.print("/");
//  Serial.print(t.year, DEC);
//  Serial.print(" HORA:");  // Hora en formato 0-23.
//  Serial.print(t.hour, DEC);
//  Serial.print(":");  // Minutos.
//  Serial.print(t.min, DEC);
//  Serial.print(":");  // Segundos.
//  Serial.print(t.sec, DEC);
//  Serial.println();
//delay(300000); //cada 5 minutos
//Voltaje actual de la batería-------------------------------------------------//
void bateria(){
char humedad [bufSize];
float ValorSensor = fmap(adc.analogRead(2),0,4095,0.0,25.0);
dtostrf(ValorSensor,5,2,humedad);
memcpy(XmitBuf, humedad, bufSize);
chibiTx(0x0003, XmitBuf,bufSize);
Serial.print("batería: ");
Serial.println(ValorSensor);
delay(5000); //mandara dato cada 5 segundos
```

```
}
//cambio de escala entre floats para sensor de voltaje
float fmap(float x, float in_min, float in_max, float out_min, float out_max)
{
 return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
//-------------------------------------------------------------------------//
//-Si el nodo transmisor recibiera datos de otro nodo mandarlo al nodo
//receptor)
void TXnodo(int arg_cnt,char **args){
  byte data[100];
   int addr, len;
    // convert cmd line string to integer with specified base
    addr = chibiCmdStr2Num(args[1], 16);

    // concatenate strings typed into the command line and send it to
    // the specified address
    len = strCat((char *)data, 2, arg_cnt, args);
    chibiTx(0x0003, data,len);
}
int strCat(char *buf, unsigned char index, char arg_cnt, char **args)
{
    uint8_t i, len;
    char *data_ptr;
    data_ptr = buf;
    for (i=0; i<arg_cnt - index; i++){
        len = strlen(args[i+index]);
        strcpy((char *)data_ptr, (char *)args[i+index]);
        data_ptr += len;
        *data_ptr++ = ' ';
    }
    *data_ptr++ = '\0';
    return data_ptr - buf;
}
//--Funciones de dormir y despertar al nodo----------//
void dormirNodo(){
  chibiSleepRadio(1);//Dormir radio (activar=1)
  delay(100);
  }
void despertarNodo(){
  chibiSleepRadio(0);//Despertar radio (desactivar=0)
  delay(100);
  //SleepClass::powerDownAndWakeupExternalEvent(0);
}
```

ANEXO F: Función de MATLAB para el consumo energético

A continuación, se presenta la siguiente función para el monitoreo

```
%Diseño de Ing. Fer Alvarez.
%Función que sirve para graficar el consumo energetico de la Freakduino.
%Comunicación serial Matlab con Arduino.
function Monitoreo_ArduinoML(n_muestras)
close all;
clc;
y=zeros(1,1000);%Vector que almacena los datos
%Inicialización del puerto serial a utilizar
delete(instrfind({'Port'},{'COM5'}));
puerto_serial=serial('COM5');
puerto_serial.Baudrate=9600;
%warning('off','MATLAB:serial:fscanf:unsuccesfulRead');
%Abrir puerto serial
fopen(puerto_serial);
%contador del número de muestras
contador_muestras=1;
%Gráfica
figure('Name','Monitoreo de consumo para Freakduino RIS. Ing. Fer
Alvarez')
title('Muestreos de consumo energético');
xlabel('número de muestra');
ylabel('Voltaje (V)');
grid on;
hold on;
%Lecturas de las muestras por medio de un ciclo while
while contador_muestras<=n_muestras
    ylim([0 8]); %las baterias no exceden de 3 volts
    %ylim auto;
    xlim([0 contador_muestras+5]); %xlim([contador_muestras-5
contador_muestras+5])
    ValorSensor=fscanf(puerto_serial,'%d')';
    y(contador_muestras)=(ValorSensor(1)*25/1024);
    plot(contador_muestras,y(contador_muestras),'X-r');
    drawnow
    contador_muestras=contador_muestras+1;
end
%Cerrar la conexión con el puerto y eliminación de las variables
fclose(puerto_serial);
delete(puerto_serial);
clear all;
end
```

ANEXO G: Participaciones y logros académicos

MÉXICO
GOBIERNO DE LA REPÚBLICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Otorgan la presente CONSTANCIA a

Maria Fernanda Alvarez Vélez

por haber concluido el curso masivo abierto en línea

Desarrollo Sustentable, nuestro futuro compartido

Impartido por Tecnológico Nacional de México, a través de la plataforma MéxicoX.

Dr. Francisco Valdés Perezgasga
Profesor Investigador
Instituto Tecnológico de La Laguna

Dr. Hesner Coto Fuentes
Profesor Investigador
Instituto Tecnológico Superior de Lerdo

Este curso no acredita al participante como alumno oficial de la institución que le imparte. No contiene créditos académicos ni revalidación académica en ninguno de los programas de estudios formales o de extensión.

MÉXICO DIGITAL Televisión Educativa méxicoX

Constancia número: 0c00976c746f47c3b48cf3488842f355

31/03/2017

MÉXICO
GOBIERNO DE LA REPÚBLICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Otorgan la presente CONSTANCIA a

Maria Fernanda Alvarez Vélez

por haber concluido el curso masivo abierto en línea

Ahorro de energía

Impartido por Tecnológico de Monterrey, a través de la plataforma MéxicoX.

Dr. Jonathan Carlos Mayo Maldonado	Dr. Enrique Ortiz Nadal	Dr. Jesús Elías Valdez Reséndiz
Profesor Titular Tecnológico de Monterrey	Profesor Titular Tecnológico de Monterrey	Profesor Titular Tecnológico de Monterrey

Este curso no acredita al participante como alumno oficial de la institución que lo imparte. No contiene créditos académicos ni revalidación académica en ninguno de los programas de estudios formales o de extensión.

MX MÉXICO DIGITAL | Televisión Educativa | méxicoX

Constancia número: a9e773aa15884da38c5b65c85135e249

6/11/2017

LUZAC

Escuela Preparatoria Federal por Cooperación
Clave EMS-2/164

OTORGA EL PRESENTE AGRADECIMIENTO AL :

ING. MARIA FERNANDA ÁLVAREZ VÉLEZ

Por su excelente colaboración como Jurado en la FERIA CIENTÍFICA DE FÍSICA 2017 de los alumnos de 5to Semestre

11 de Noviembre de 2017

DRA. ÁNGELICA ROCÍO MOTA BARRAGÁN
Directora