INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS" ZACATENCO

"DISEÑO DE UNA RED DOMOTICA UTILIZANDO MÓDULOS XBEE"

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

PRESENTAN:

MIGUEL ANGEL JUAREZ MALDONADO
EDUARDO ALEJANDRO SANCHEZ CANCHOLA
ARMANDO TENORIO ARCHUNDIA

ASESORES:

ING. ISMAEL GABRIEL COSME CISNEROS
ING. RICARDO LÓPEZ MACEDO

CDMX, Junio 2019

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

T E M A D E T E S I S

QUE PARA OBTENER EL TÍTULO DE	**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**
POR LA OPCIÓN DE TITULACIÓN	**TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL**
DEBERA (N) DESARROLLAR	**C. MIGUEL ANGEL JUAREZ MALDONADO** **C. EDUARDO ALEJANDRO SANCHEZ CANCHOLA** **C. ARMANDO TENORIO ARCHUNDIA**

"DISEÑO DE UNA RED DOMOTICA UTILIZANDO MÓDULOS XBEE"

IMPLEMENTAR UNA RED TIPO ESTRELLA PARA EL MONITOREO DEL NIVEL Y CONTROL DE LLENADO DE DEPÓSITOS DE AGUA UTILIZANDO MÓDULOS XBEE.

- **MARCO TEÓRICO**
- **MICROCONTROLADORES**
- **DESARROLLO**

CIUDAD DE MÉXICO, A 10 DE JUNIO DEL 2019.

A S E S O R E S

ING. ISMAEL GABRIEL COSME CISNEROS **ING. RICARDO LÓPEZ MACEDO**

ING. GABRIEL VEGA REYES
JEFE DEL DEPARTAMENTO DE
INGENIERÍA EN COMUNICACIONES Y ELECTRÓNICA

Instituto Politécnico Nacional

P r e s e n t e

Bajo protesta de decir verdad los que suscriben **Miguel Angel Juarez Maldonado, Eduardo Alejandro Sanchez Canchola y Armando Tenorio Archundia**, manifestamos ser autores y titulares de los derechos morales y patrimoniales de la obra titulada **"DISEÑO DE UNA RED DOMOTICA UTILIZANDO MÓDULOS XBEE"**, en adelante **"La Tesis"** y de la cual se adjunta copia de **un impreso y un cd**, por lo que por medio del presente y con fundamento del artículo 27 fracción II, inciso b) de la Ley Federal del Derecho de Autor, otorgamos al **Instituto Politécnico Nacional,** en adelante **EL IPN**, autorización no exclusiva para comunicar y exhibir públicamente total o parcialmente en medios digitales o en cualquier otro medio; para apoyar a trabajos futuros relacionados con el tema de **"La Tesis"** por un periodo de **2 años** contando a partir de la fecha de la presente autorización, dicho periodo se renovará automáticamente en caso de no dar aviso expreso a **EL IPN** de su terminación.

En virtud de lo anterior, **El IPN** deberá reconocer en todo momento nuestra calidad de autores de la **"La Tesis".**

Adicionalmente y en nuestra calidad de autores y titulares de los derechos morales y patrimoniales de **"La Tesis"**, manifestamos que la misma es original y que la presente autorización no contraviene ninguna otorgada por los suscritos respecto de **"La Tesis"**, por lo que deslindamos de toda responsabilidad a **EL IPN** en caso de que el contenido de **"La Tesis"** o la autorización concedida afecte o viole derechos autorales, industriales, secretos industriales, convenios o contratos de confidencialidad o en general cualquier derecho de propiedad intelectual de terceros y asumimos las consecuencias legales y económicas de cualquier demanda o reclamación que puedan derivarse del caso.

Ciudad de México, a 19 de agosto de 2019

Atentamente

Miguel Angel Juarez Maldonado

Eduardo Alejandro Sanchez Canchola

Armando Tenorio Archundia

Agradecimientos

Mi primera y mayor deuda de gratitud es para mis padres, los que me motivaron a seguir adelante, los que me dieron las herramientas para no desistir, los que me dieron consejos y lecciones para cada día ser mejor. A quienes la ilusión de su existencia ha sido convertirme en una persona de provecho. A quienes nunca podré pagar todos los desvelos ni con las riquezas más grandes del mundo. A quienes estaban ahí cuando más los necesitaba, motivarme, ser mi motor, guiarme por un camino donde cada vereda lleva a lugares distintos. Deseo expresarles, que mis ideales, esfuerzos y logros han sido también suyos e inspirados en ustedes y constituyen en el legado más grande que pudiera recibir, Con amor, admiración y respeto. Hoy y siempre gracias Mamá y Papá por lo que juntos hemos logrado.

Agradezco a Dios y a la vida por haberme guiado durante el transcurso de mi carrera, por tomar buenas decisiones e incluso por haber aprendido de mis errores, siendo así mi mejor maestro de vida para alcanzar mis metas.

No dejo atrás esa gran familia, mis amigos, esos con los que formé grandes amistades, me llevo sus pláticas, risas, triunfos, derrotas, momentos que vivimos juntos, me llevo lo que me compartieron y compartí, esa familia que elegí, gracias por todo.

Al Instituto Politécnico Nacional y a la Escuela Superior de Ingeniería Mecánica y Eléctrica, mi segundo hogar que me permito estar en cada una de sus instalaciones sacare provecho de cada uno de los conocimientos que adquirí en mi instancia para poner "La Técnica al Servicio de la Patria", no te defraudaré...

Miguel Ángel Juárez Maldonado

Agradecimientos

A mis padres por ser los principales promotores de mis sueños, por cada consejo y por cada una de sus palabras que me guiaron durante mi vida, gracias a ellos por cada día confiar y creer en mí y en mis expectativas, gracias a mi madre por siempre alentarme cuando más no podía, gracias a mi padre por siempre desear y anhelar lo mejor para mi vida.

Al Instituto Politécnico Nacional que me abrió las puertas y me dio la oportunidad de pertenecer a sus filas y así poder realizar mi superación académica.

A la Escuela Superior de Ingeniería Mecánica y Eléctrica, que se convirtió durante el tiempo de mi trayecto en mi segundo hogar, gracias por haberme permitido fórmame en ella y por brindarme el conocimiento para poder concluir este logro en mi vida.

A mis amigos, compañeros y a todas las personas que fueron participes de este proceso, ya sea de manera directa o indirecta, gracias a todos ustedes.

A Dios y a la vida por permitirme llegar a donde me encuentro, por darme salud y mantenerme fuerte, pero sobre todo por permitir que mis padres y seres queridos formaran parte de este logro.

Eduardo Alejandro Sanchez Canchola

Agradecimientos

Primeramente, a **Dios**, porque gracias a él, he cumplido una de las etapas más importantes de vida, por darme salud y por permitirme seguir adelante.

A mis padres, **Rocio Archundia** y **Francisco Tenorio** por todo el apoyo que me han brindado para ser de mí una gran persona, por todos los consejos que me dieron para formarme por el buen camino y por ser un pilar fundamental en mi formación profesional y personal. Los amo.

A mi hermano, **Francisco Salvador Tenorio** por alentarme a cumplir mi sueño, por compartir las desveladas y todo lo hecho en esta etapa. Te amo.

A mi novia, **Xóchitl Hernández** por estar ahí, en todo momento, por apoyarme para cumplir todas mis metas. Te amo.

Al **Instituto Politécnico Nacional** y a la **Escuela Superior de Ingeniería Mecánica y Eléctrica** que durante estos 7 años y medio me dieron mi segundo hogar, por darme la oportunidad de pertenecer a esta gran casa de estudios.

Y a toda mi familia, en especial a mi primo **Alonso Archundia**, a mis amigos de banca y mis amigos de toda la vida, **Alan**, **David**, **Ricardo**, **Ulises**, **Eduardo**, **Miguel**, **Francisco**, **Jorge**. Gracias a todos ustedes.

Armando Tenorio Archundia

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

La creación de sistemas domóticos en México es un tema que actualmente está en desarrollo, la inquietud de realizar este tipo de sistemas en nuevas generaciones ha hecho posible que se propongan prototipos automatizados que realicen tareas comunes permitiendo el mínimo esfuerzo al usuario ya sea en el hogar, en la industria o en algunos otros lugares donde implique la solución a un proceso simple. En este proyecto se muestra un sistema de comunicación domótico que controla el llenado de un tinaco, verifica el nivel de agua de una cisterna y visualiza el estado de ambos por medio de una pantalla LCD. Este sistema incluye interruptores de nivel de agua que son necesarios en los depósitos antes mencionados y que, al accionarse enviarán una interrupción por medio de un microcontrolador programado ejecutando tareas ya establecidas, estos a su vez serán enviados mediante radiofrecuencia por medio de módulos XBee.

El escrito se compone de tres capítulos que desglosa investigación relevante para la ejecución del sistema. En primera instancia se muestra una introducción breve sobre domótica, protocolos, el tipo de módulo de radiofrecuencia utilizado (XBee), entre otra información. Posteriormente describiremos el microcontrolador que hará posible la comunicación entre los circuitos electrónicos. En este apartado se muestran referencias que identifican al microcontrolador. Teniendo estos antecedentes finalizaremos con información que compone el proyecto titulado "Diseño de una red domótica utilizando módulos XBee", el cual está basado en la implementación de una red tipo estrella, la cual está compuesta por tres módulos de radiofrecuencia XBee, los cuales se encargan de establecer la comunicación de forma inalámbrica entre ellos y que, a su vez son controlados por microcontroladores atmega48 que por medio de su programación permita la ejecución de una tarea establecida, la cual es el llenado automático de un tinaco que, dependiendo de su nivel de agua, es decir, al estar el tinaco en un nivel bajo, enviara la señal de RF al módulo asignado en la bomba de agua para el encendido automático de la bomba, pero antes se corrobora la capacidad de agua disponible en la cisterna como protección para que la bomba no funcione sin agua y se pueda dañar, dicho lo anterior podrá ser visualizado por medio de una pantalla LCD que constantemente enviara texto referente al estado del sistema descrito. Cabe mencionar que las señales enviadas por los módulos XBee depende de los interruptores de nivel de agua los cuales están conectados a los microcontroladores y estos microcontroladores a su vez están conectados a los módulos para poder establecer la comunicación correctamente.

Justificación

En la actualidad, México se enfrenta a procesos de desarrollo en materia tecnológica, donde es posible la adquisición de tecnología en sistemas domóticos, pero solo una parte del sector de la población tienen la posibilidad de adquirirla, esto se debe a que este tipo de sistemas son costosos y particularmente son importados de otros países más desarrollados. Sabemos que existen en el mercado una variedad de sistema y aplicaciones que realizan tareas automatizadas por lo que México en materia de desarrollo de estos sistemas aún no están familiarizados.

Una de las razones para trabajar en sistemas domóticos es implementar un sistema de comunicación inalámbrica en una casa habitación para el control de diferentes parámetros con la mínima intervención humana y con disponibilidad de adquirir dicho sistema, por lo que se busca que México este a la altura de países con desarrollo tecnológico en casas inteligentes. El encender o apagar luces, abrir o cerrar una ventana, llenar un tinaco de agua o regar el césped, son actividades que pueden convertirse en un problema para aquellas personas que cuentan con alguna discapacidad, que viven solos, adultos mayores o personas que tengan el gusto por la modernidad en este tipo de sistemas.

Objetivo general

Implementar una red tipo estrella para el monitoreo del nivel y control de llenado de depósitos de agua utilizando módulos XBee.

Objetivos particulares

- Diseñar un sistema para detectar el nivel de agua en depósitos.
- Utilizar microcontroladores para controlar la comunicación de circuitos transmisores y receptores en una red domótica.
- Configurar módulos XBee en una red tipo estrella.
- Diseñar un circuito electrónico de potencia para el encendido de una bomba de agua.

Capítulo 1 Marco teórico

1.1 Domótica

1.1.1 ¿Qué es domótica?

El término "domótica" se refiere a plataformas que incluyen la creación de controles automatizados para los hogares, por lo que al término "domótica" también se le conoce como los sistemas de casas inteligentes [Figura 1].

Figura 1 Aplicaciones de una red domótica, tomado de (1).

Para ello, la domótica incluye elementos de software y de hardware, que dan lugar al desarrollo de plataformas con ciertos aspectos de diseño, incluyendo las necesidades puntuales de los usuarios que van a utilizar dicho sistema. La domótica posee grandes ventajas, ya que, al tratarse de un conjunto de tecnologías aplicadas al control y la automatización inteligente del hogar, permite entre otras cosas lograr un real ahorro energético y mejorar el acceso a elementos por parte de personas con discapacidades [2].

La domótica es la automatización de sistemas o equipos del hogar por medio de una red de comunicación; la domótica trata de brindar confort y seguridad en una casa habitación economizando gastos energéticos por medio de redes de comunicación implementadas para este uso específico. La domótica al utilizar sistemas de control trabaja con ciertos protocolos a través redes de comunicación.

1.1.2 Protocolos de comunicación utilizados en domótica

Un protocolo de comunicación es un conjunto de normas que permiten que dos o más dispositivos de un sistema de comunicación se comuniquen entre sí para transmitir información, sin un protocolo de comunicación resultaría caótica y por tanto imposible el intercambio de comunicación entre los dispositivos unidos a la red [3].

Un protocolo define únicamente cómo se deben comunicar los equipos, es decir, el formato y la secuencia de datos que van a intercambiar. Por el contrario, un protocolo no

define cómo se programa el software para que sea compatible con el protocolo. Esto se denomina implementación o la conversión de un protocolo a un lenguaje de programación [4].

Algunos de los principales protocolos de comunicación utilizados en domótica se muestran a continuación que, posteriormente se contrastaran estos protocolos para determinar el protocolo a fin de utilizar en la red.

X10

Es uno de los protocolos más antiguos que aún se emplean en aplicaciones domóticas. La tecnología X10 de corrientes portadoras fue desarrollada entre 1976 y 1978 por ingenieros de Pico Electronics Ltd, en Glenrothes.

El sistema X10 se caracteriza principalmente por:

- Tener una instalación sencilla.
- Fácil manejo por el usuario.
- Flexible y ampliable.
- Ser un sistema descentralizado; configurable; no programable.
- Compatibilidad casi absoluta con los productos de la misma gama.

El proceso de comunicación del sistema se basa en la capacidad del transmisor X10 de enviar un código de baja tensión superpuesta a la señal eléctrica. Los receptores X10 conectados a la red eléctrica tienen la capacidad de leer este código, pero solo responde el receptor al que va dirigido el mensaje. Cada equipo tiene asignada una dirección que lo identificará.

KNX

El Bus de Instalación Europeo (EIB o EIBus) actualmente llamado KNX es un sistema de domótica e inmótica basado en un Bus de datos. A diferencia de X10, que utiliza la red eléctrica, y otros sistemas actuales por RF, el KNX utiliza su propio cableado, con lo cual se ha de proceder a instalar las conducciones adecuadas en el hogar y para el sistema.

El KNX, a través de pasarelas, puede ser utilizado en sistemas inalámbricos como los infrarrojos, radiofrecuencia o incluso empaquetado para enviar información por internet u otra red TCP/IP. El antiguamente llamado EIB, desde 1999 la KNX Association ha

fusionado este bus con otros dos existentes en el mercado europeo (BatiBUS y EHS), dando lugar a KNX que se establece como una alternativa de automatización.

LONWorks

LONWorks es una plataforma tecnológica basada en el protocolo abierto llamado LONTalk, para aplicaciones de control y automatización, que permite distribuir inteligencia descentralizadamente a pequeños nodos o dispositivos dentro de un sistema más grande y en los que éstos pueden intercambiar información para la ejecución de diferentes funciones: medir, procesar información, conmutar, regular dentro de las instalaciones e infraestructuras.

1.2 ZigBee

Es un conjunto de protocolos de comunicación inalámbrica, para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (Wireless Personal Area Network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías [5].

ZigBee utiliza la banda ISM (Industrial, Scientific and Medical) para usos industriales, científicos y médicos; en concreto, 868 MHz en Europa, 915 en Estados Unidos y 2,4 GHz en todo el mundo. Sin embargo, a la hora de diseñar dispositivos, las empresas optarán prácticamente siempre por la banda de 2,4 GHz, por ser libre en todo el mundo.

Características

- ZigBee, también conocido como "HomeRF Lite", es una tecnología inalámbrica con velocidades comprendidas entre 20 kB/s y 250 kB/s.
- Los rangos de alcance son de 10 m a 75 m.
- Puede usar las bandas libres ISM de 2,4 GHz (Mundial), 868 MHz Europa y 915 MHz EE. UU.
- Diferentes tipos de topologías como estrella, punto a punto, malla, árbol.
- Acceso de canal mediante CSMA/CA (acceso múltiple por detección de portadora con evasión de colisiones).

1.2.1 Comparación de protocolos ZigBee/X10/LONWorks/KNX

La siguiente tabla muestra la comparación entre protocolos utilizados en redes de comunicación utilizadas en domótica, los parámetros que se presentan detallan información que sirve para determinar el tipo de protocolo que se desea utilizar [Tabla 1].

Tabla 1 Comparación entre protocolos ZigBee/X10/LONWorks/KNX.

Protocolo / Características	ZigBee	X10	LONWorks	KNX
Medio de Transmisión	Inalámbrica	Cable eléctrico	TP Cable eléctrico Radio Coaxial FO	TP0 TP1 PL100 PL132 Ethernet Radio
Velocidad de transmisión	20 Kbps-250Kbps	60 bps EE. UU. 50 bps Europa	78 Kbps-1.28Mbps 5.4 Kbps	9600 bps 1200/ 2400 bps 2.4 Kbps
Distancia Máxima del Dispositivo	10 – 75 m	185 m^2	500 – 2700 m	1000 m 600m
Tipo de Estándar	Abierto	Abierto	Bajo licencia	Bajo licencia
Dispositivos en la Red	65535	256	Casi ilimitado	10000

1.3 Estándar IEEE 802.15.4

IEEE 802.15.4 es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos. Las características más importantes en este estándar son su flexibilidad de red, bajos costos, bajo consumo de energía; este estándar se puede utilizar para muchas aplicaciones en el hogar que requieren una tasa baja en la transmisión de datos. Las redes inalámbricas implican un gran intercambio de información con un mínimo de esfuerzo de instalación. Esta tendencia es impulsada por la gran capacidad de integrar componentes inalámbricos de una forma más barata y el éxito que tienen otros sistemas de comunicación inalámbrica

como los celulares. Con el gran crecimiento de Internet, las mayores preocupaciones de los diseñadores es el satisfacer la necesidad de compartir conexiones de alta velocidad.

En el otro lado del espectro, las aplicaciones como la automatización del hogar y aplicaciones de seguridad han relajado dichas necesidades. Estas aplicaciones no pueden manejar protocolos muy pesados ya que afectarían seriamente en el consumo de energía y requerirían de mayor poder de procesamiento [6].

1.3.1 Características del estándar 802.15.4

A continuación, se enlistan algunas de las características del estándar 802.15.4 [6].

- Rango de transmisión de datos: 868 MHz: 20Kb/s; 915 MHz: 40Kb/s; 2.4 GHz: 250Kb/s.
- Alcance: 10 – 30 m.
- Tiempo de respuesta: Abajo de los 15 ms.
- Canales: 868/915 MHz: 11 canales; 2.4 GHz: 16 canales.
- Bandas de frecuencia: Dos PHY: 868/915 MHz y 2.4 GHz.
- Direccionamiento: Cortos de 8 bits o 64 bits IEEE.
- Canal de acceso: CSMA-CA y rasurado CSMA-CA.
- Temperatura: El rango de temperatura industrial: -40° a +85° C.

1.3.2 Capas de modelo IEEE 802.15.4

El estándar ZigBee define solo las capas de red, aplicación y seguridad, y adopta del estándar IEEE 802.15.4 la capa física y MAC. El estándar fue desarrollado por el comité 802 del IEEE y fue inicialmente lanzado en 2003.

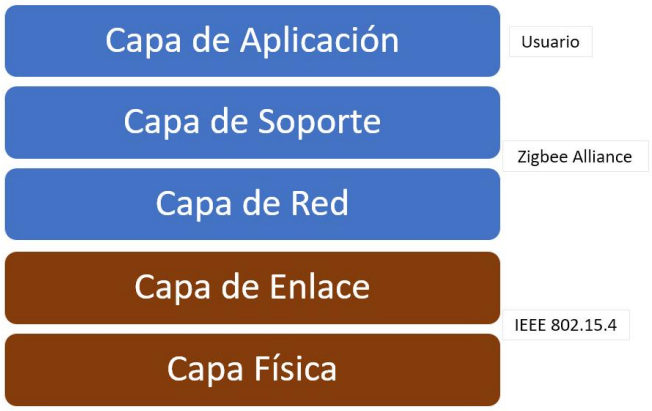

Figura 2 Capas del modelo ZigBee y estándar IEEE 802.15.4

1.3.2.1 Capa física

Es la capa más cercana al hardware y es donde tiene lugar el control y las comunicaciones del transceptor. Esta capa es la responsable de activar la transmisión y recepción de paquetes del sistema. Además, también selecciona el canal de frecuencia y se asegura que este no es usado por otros dispositivos de la red.

1.3.2.2 Capa MAC

La capa MAC tiene la misión de proveer servicios a las capas superiores para que estas se encarguen tanto del manejo de los datos que son transferidos en una red WSN como una de las primitivas para que un sistema operativo administre estas dos capas (Capa física y de enlace de datos).

1.4 Topologías de red

1.4.1 Topología de red tipo estrella

Como se muestra en la figura siguiente [Figura 3], el coordinador se sitúa en el centro, la topología de red tipo estrella establece la comunicación entre los dispositivos y un controlador central único. El coordinador puede ser conectado a la corriente eléctrica, mientras que los otros dispositivos pueden ser alimentados por baterías. Las aplicaciones que se benefician de esta topología incluyen domótica, computadoras personales, juguetes y juegos. Después de que un dispositivo se activa por primera vez, puede establecer su propia red y convertirse en el coordinador. Cada red de inicio elige un identificador, que no está utilizado actualmente por cualquier otra red dentro de la esfera de radio de influencia. Esto permite que cada red en estrella pueda operar de forma independiente [7].

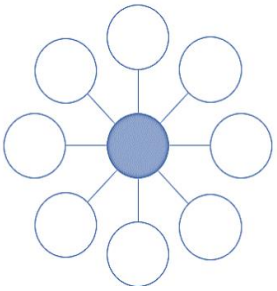

Figura 3 Topología de red tipo estrella.

1.4.2 Topología de red tipo árbol

La topología de red tipo árbol corresponde a un híbrido que se forma aplicando una estructura de árbol con varios nodos, mostrado en la figura siguiente [Figura 4]. Cada nodo es un grupo, y el conjunto será un árbol de grupo. La red de árbol de grupo es un caso especial de una red punto a punto en la que la mayoría de los dispositivos son dispositivos de funcionalidad completa y un dispositivo de funcionalidad reducida pueden conectarse a una red de árbol de grupos como un nodo único al final de cada rama. Cualquiera de los dispositivos de funcionalidad completa puede actuar como coordinador y proporcionar servicios de sincronización con otros dispositivos y coordinadores. Sin embargo, sólo uno de estos coordinadores es el coordinador principal. El coordinador principal se forma con el primer grupo mediante el establecimiento de sí mismo como la cabeza de grupo con un identificador de grupo de cero, elegir un identificador sin usar y transmitir tramas de señalización a los dispositivos vecinos. Un dispositivo candidato que reciba una trama de datos puede solicitar unirse a la red. Si el coordinador permite unirse al dispositivo, añadirá este nuevo dispositivo como un dispositivo secundario en su lista de vecinos. Una vez que se cumplen los requisitos de aplicación o de la red, el coordinador puede instruir a un dispositivo para convertirse en un nuevo grupo adyacente a la primera. La ventaja de esta estructura en clúster es el aumento del área de cobertura a costa de una mayor latencia de mensajes [7].

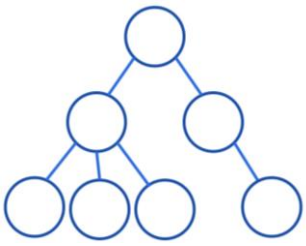

Figura 4 Topología de red tipo árbol.

1.4.3 Topología de red tipo malla o red mesh

Por su parte, la estructura de la topología de red tipo malla es similar a la de la topología de red tipo árbol, con el coordinador en la parte superior de una estructura en forma de árbol como se muestra en la figura siguiente [Figura 5]. El coordinador está vinculado a un conjunto de routers y dispositivos finales. Un router puede entonces estar vinculado a más routers y dispositivos finales. Esto puede continuar a un número de niveles. Sin embargo, las reglas de comunicación son más flexibles que en los nodos de router dentro

del alcance del otro puede comunicarse directamente. La topología de malla da lugar a la propagación de mensajes más eficiente, y significa que posee rutas alternativas en donde se pueden encontrar si un enlace falla o hay congestión. Una función de "descubrimiento de ruta" es siempre lo que permite a la red para encontrar la mejor ruta disponible para un mensaje [7].

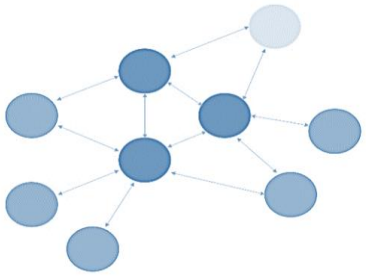

Figura 5 Topología de red tipo malla.

1.5 Alianza ZigBee (ZigBee Alliance)

La alianza ZigBee es un ecosistema internacional de compañías cuyo objetivo es habilitar redes inalámbricas con capacidades de control y monitoreo que sean confiables, de bajo consumo de energía y de bajo costo; todo basado en un estándar público global que permita a los fabricantes alrededor del mundo crear productos que sean compatibles entre ellos. Una de las tareas más importantes de esta alianza es definir el conjunto de protocolos que habilitarán la comunicación entre los dispositivos y es a esta definición de protocolos a la que se le conoce como ZigBee [8]. En otras palabras: ZigBee es un protocolo de comunicaciones inalámbricas para redes de área personal (WPANs) IEEE 802.15.4.

La Alianza ZigBee ha atraído a algunas de las organizaciones líderes más innovadoras y conocidas del mundo. En conjunto, estás empresas están colaborando para mejorar la forma en que vivimos con el desarrollo de estándares inteligentes y fáciles de usar que ayudan a ganar más control sobre el mundo. La Alianza tiene tres niveles de membresía: Promotor, Participante y Adopter, lo muestra en la tabla siguiente [Tabla 2].

Tabla 2 Niveles de membresía de Alianza ZigBee.

Promotores	Participantes	Adopter
• **Silicon Labs** • **Huawei** • **Comcast** • **NXP** • **MMP Networks** • **Schneider Electric** • **Texas Instrument**	• Broadcom • Analog Devices • Cómodo Computime • DIGI • Dolphin • Intel • Microchip • Nortek • Samsung • Siemens • Toshiba • Universal Electronics	• Calix • Deutsche Telekom AG • Electro Cirkel Retail BV • Energate, Inc. • Energate, Inc. • Ibis Networks, Inc. • Panasonic Corporation • Powerley • Toshiba Visual Solutions • Unitech Electronics

1.6 Módulo XBee

Los módulos XBee son pequeños dispositivos que pueden comunicarse entre sí de manera inalámbrica [Figura 6]. Son fabricados por DIGI International, los cuales ofrecen una variedad de combinaciones de hardware, protocolos, antenas y potencias de transmisión. De acuerdo con DIGI, los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos dispositivos utilizan un protocolo de comunicación para crear redes multipunto o punto a punto. Fueron diseñados para aplicaciones que requieren un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible [9].

Figura 6 Modulo XBee convencional, tomado de (10).

1.6.1 Elementos y tipos de módulos XBee

Elementos de una Red XBee

Los elementos básicos de una red XBee se muestran en el siguiente esquema, [Figura 7].

- **Coordinador:** Es el responsable de mantener la red. Solo puede haber uno por red.
- **Router:** Actúa como un mensajero entre otros dispositivos que están demasiado separados para transmitir la información por su cuenta. Son generalmente conectados a una toma eléctrica, ya que deben estar encendidos todo el tiempo.
- **Dispositivo terminal:** Pueden enviar y recibir información, pero no actúan como mensajeros entre cualquier otro dispositivo. Para ahorrar energía pueden entrar temporalmente en un modo de espera [9].

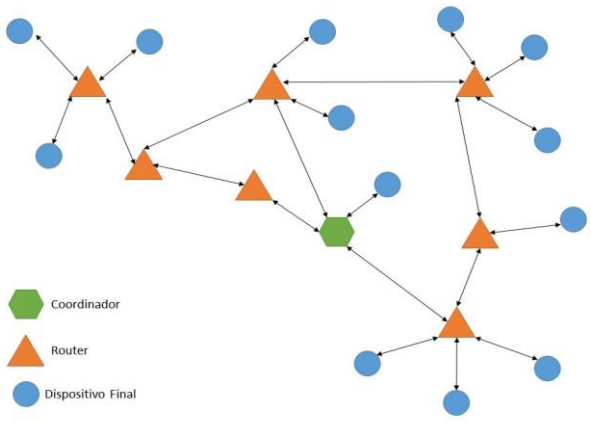

Figura 7 Elementos dentro de una red XBee, tomado de (11).

Tipos de módulos XBee.

Existen dos tipos básicos de variedades de hardware, como lo muestra la siguiente tabla. [Tabla 3 Tipos de módulos XBee]

Tabla 3 Tipos de módulos XBee

XBee Serie 1	XBee Serie 2
	
• No necesitan ser configurados, por lo tanto, fáciles para trabajar e ideales para realizar una comunicación básica. • Comunicación punto a punto, punto-multipunto. • Los módulos XBee serie 1 no pueden comunicarse con los módulos XBee serie 2.	• Comunicación punto a punto, punto-multipunto y Red Mesh. • Necesitan ser configurados

A continuación, [Tabla 4], se muestran los distintos tipos de antenas que pueden utilizar los módulos XBee.

Tabla 4 Módulos XBee comerciales con distintas antenas.

Tipo de Antena	Imagen	Características o Descripción
Antena de Cable		Como su nombre lo indica, es un cable que sobresale del XBee. En él la distancia máxima de transmisión es más o menos la misma en todas las direcciones.
Antena de Chip		La antena es un chip de cerámica plana que está al ras del cuerpo de la XBee. Eso hace que sea más pequeño y robusto, sin embargo, la señal se atenúa en muchas direcciones. Se utilizan cuando existe el riesgo que la antena de cable se rompa o se tiene poco espacio para colocarlo.
Antena PCB		La antena se imprime directamente en la placa de circuito del XBee. La antena PCB ofrece la mayoría de las ventajas (y desventajas) de la antena de chip con un costo menor.
Conector U. FL y conector RPMSA		Necesitan una antena externa. Se utilizan cuando se desea orientar una antena en diversas posiciones o se requiere utilizar un tipo especial de antena.

1.6.2 Circuito básico de un módulo XBee

La tabla muestra las terminales con las que cuenta el módulo XBee, se muestran 20 terminales que, dependiendo de la ocupación del módulo determinara el modo de operación que se desea, las terminales sombreadas son las mínimas necesarias para tener una transmisión y recepción simple de comunicación. El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TX y RX) para comunicarse con un microcontrolador, o directamente a un puerto serial utilizando algún conversor adecuado para los niveles de voltaje. Esta configuración, no permite el uso de Control de Flujo (RTS & CTS), por lo

que esta opción debe estar desactivada en el terminal y en el módulo XBee. En caso de que se envíe una gran cantidad de información, el buffer del módulo se puede sobrepasar. Para evitarlo existen dos alternativas: bajar la tasa de transmisión y activar el control de flujo [12].

Cada módulo XBee, al igual que ocurre con las direcciones MAC de los dispositivos ethernet, tiene una dirección única. En el caso de los módulos XBee cada uno de ellos tiene una dirección única de 64 bits que viene grabada de fábrica. Por otro lado, la red ZigBee, utiliza para sus algoritmos de ruteo direcciones de 16 bits. Cada vez que un dispositivo se asocia a una red ZigBee, el Coordinador al cual se asocia le asigna una dirección única en toda la red de 16 bits. Por eso el número máximo teórico de elementos que puede haber en una red ZigBee es de 2^16 = 65535, que es el número máximo de direcciones de red que se pueden asignar. Estos módulos XBee, pueden ser ajustados para usarse en redes de configuración punto a punto, punto a multipunto o peer-to-peer.

Tabla 5 Asignación de pines de módulos XBee

Pin No.	Nombre	Dirección	Descripción
1	VCC	-	Fuente de Alimentación.
2	DOUT	Salida	UART-Salida de datos
3	DIN	Entrada	UART-Entrada de datos
4	DO8	Salida	Salida digital 8
5	RESET	Entrada	Módulo reset
6	PWM0/RSSI	Salida	Salida PWM0/Indicador de intensidad de señal
7	PWM1	Salida	Salida PWM1
8	[Reservado]	-	No conectar
9	DTR/Sleep_RQ/DI8	Entrada	Línea de control sleep/Entrada digital 8
10	GND	-	Tierra
11	AD4/DIO4	Ambos	Entrada analógica 4/Entrada digital 4
12	CTS/DIO7	Ambos	Control de flujo de limpieza de envió/entrada analógica 7
13	ON/SLEEP	Entrada	Módulo de indicador de estado
14	VREF	Ambos	Voltaje de referencia para entradas A/D
15	Associate/AD5/DOI5	Ambos	Indicador asociado/Entrada analógica 5/ I/O digital 7
16	RTS/AD6/DIO6	Ambos	
17	AD3/DIO3	Ambos	Entrada analógica 3/ I/O digital 3
18	AD2/DIO2	Ambos	Entrada analógica 2/ I/O digital 2
19	AD1/DIO1	Ambos	Entrada analógica 1/ I/O digital 1
20	AD0/DIO0	Ambos	Entrada analógica 0/ I/O digital 0

Capítulo 2. Microcontroladores

2.1 ¿Qué son los microcontroladores?

Los microcontroladores están presentes en muchos de los equipos electrónicos que empleamos en nuestra vida cotidiana. Existen en el mercado una gran variedad de modelos y una gran cantidad de aplicaciones posibles. Sin embargo, a pesar de su diversidad, hay coincidencia en los principios de funcionamiento y en las arquitecturas de muchos microcontroladores [13].

Los microcontroladores se han desarrollado para cubrir las más diversas aplicaciones. Se emplean en la industria automotriz, en equipos de comunicaciones y de telefonía, en instrumentos electrónicos, en equipos médicos e industriales de todo tipo, en electrodomésticos, en juguetes, etc.

Los microcontroladores están concebidos para ser empleados en aplicaciones puntuales, es decir, aplicaciones donde los microcontroladores deben realizar un pequeño número de tareas, al menor costo posible. En estas aplicaciones el microcontrolador ejecuta un programa almacenado permanentemente en su memoria, el cual trabaja con algunos datos almacenados temporalmente e interactúa con el exterior a través de las líneas de entrada salida de que dispone. El microcontrolador es parte de la aplicación: es un controlador incrustado o embebido en la aplicación (*embedded controller)*. En aplicaciones de cierta envergadura se utilizan varios microcontroladores, cada uno de los cuales se ocupa de un pequeño grupo de tareas.

Hay varias características que son deseables en un microcontrolador:

- Recursos de entrada salida
- Espacio optimizado
- El microcontrolador idóneo para la aplicación
- Seguridad en el funcionamiento del microcontrolador (watch-dog)
- Bajo consumo
- Protección de los programas frente a copias

2.1.1 Microcontrolador Atmega48a

El microcontrolador Atmega48A es un dispositivo construido con tecnología CMOS, para tener menor consumo de energía, de 8 bits basados en arquitectura RISC, capaz de ejecutar instrucciones en cada ciclo de reloj y una estructura definida de entradas/salidas (I/O) que limitan el uso de dispositivos externos. Posee osciladores internos, timers, USART, SPI, PWM, ADC, watch-dog timer, comparadores analógicos entre otras cosas [13].

- Soporta programación en ensamblador y en lenguaje C
- Programación ISP (In system Programming).
- Alto desempeño y bajo consumo de energía (<1 µA en estado apagado, 1.1 mA en activo).
- Cuenta con una gama de instrucciones sencillas que operan con 32 registros de propósito general.

Arquitectura de los microcontroladores AVR de Atmel

ALU – Unidad Aritmética Lógica

La ALU opera en conexión directa con los 32 registros de propósito general del AVR. Está dividida en tres categorías: aritmética, lógica y funciones de bits. Soportando inclusive operaciones de multiplicación en algunos dispositivos [13].

Status Register (registro de estatus).

Contiene información acerca de las operaciones aritméticas realizadas más recientemente. Esta información puede ser utilizada para alterar el flujo del programa o realizar operaciones condicionales [13].

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 8 Trama correspondiente a los bits del estado de registro, tomado de (13).

- Bit 7 - I: habilitación global de interrupciones.
- Bit 6 - T: almacenamiento de bit de copia.
- Bit 5 – H: bandera de medio acarreo.
- Bit 4 – S: bandera de signo, S = N V.
- Bit 3 – V: bandera de sobre flujo de complemento a 2.
- Bit 2 – N: bandera negativa.
- Bit 1 – Z: bandera de cero.
- Bit 0 – C: bandera de acarreo.

Hablando de arquitectura, el microcontrolador Atmega48A cuenta con arquitectura Harvard, a continuación, se enlistan las características de dicho microcontrolador:

- Alto desempeño
- Arquitectura RISC avanzada.

  -130 instrucciones.

  -32 registros de trabajo de propósito general.

  -Operación estática.

  -Hasta 16 MIPS a 16 MHz

  -Multiplicador de dos ciclos en el chip.

- Alto desempeño en segmentos de memoria no volátil

  -8 K bytes de memoria flash.

  -512 bytes de EEPROM.

  -1 K byte de SRAM.

  -Ciclos de borrado escritura 10,000 flash/ 100,000 EEPROM.

  -Retención de datos 20 años a 85°C / 100 años a 25 °C.

- Características de los periféricos.

  -Dos temporizadores/ contadores de 8 bits.

  -Un temporizador / contador de 16 bits.

-Un contador de tiempo real.

-Tres canales de PWM.

-Ocho canales de CAD de 10 bits.

-Seis canales de CAD de 10 bits en PDIP.

-USART serie programable.

-Interface serial de dos alambres TWI.

-Watch-dog timer programable.

- Características especiales del microcontrolador.

  -Reset en power on y detección brown out.

  -Oscilador interno.

  -Fuentes de interrupción interna y externa.

  -Cinco modos de dormido

- Entrada salida.

  -23 líneas de i/o programables.

- Voltajes de operación

  -2.7 a 5.5 volts atmega8L

  -4.7 a 5.5 volts atmega8

- Velocidad de operación.

  -0 a 8 MHz atmega8L.

  -0 a 16 MHz atmega8.

- Consumo de energía a 4 MHz 3 volts, 25 °C.

  -Activo 3 ma.

  -Modo de espera 1.0 ma.

  -Modo dormido 0.5 µA.

Entradas y Salidas

Los microcontroladores disponen de un oscilador que genera los pulsos que sincronizan todas las operaciones internas. El oscilador puede ser del tipo RC, aunque generalmente se prefiere que esté controlado por un cristal de cuarzo (XTL) debido a su gran estabilidad de frecuencia. La velocidad de ejecución de las instrucciones del programa está en relación directa con la frecuencia del oscilador del microcontrolador.

La CPU de un microcontrolador dispone de diferentes registros, algunos de propósito general y otros de propósito específico. Entre estos últimos están el registro de instrucción, el registro de estado, el contador de programa, el registro de direcciones de datos y el puntero de pila [13].

- El registro de instrucción almacena la instrucción que está siendo ejecutada por la CPU. Este registro de instrucción es invisible para el programador.
- El registro de estado (status) agrupa los bits indicadores de las características del resultado de las operaciones aritméticas y lógicas realizadas en la ALU. Entre estos indicadores están el signo del resultado (si es positivo o negativo), si el resultado es cero, si hay acarreo o préstamo, el tipo de paridad (par o impar) del resultado, etc.
- El contador de programa (PC: program counter) es el registro de la CPU donde se almacena direcciones de instrucciones. Cada vez que la CPU busca una instrucción en la memoria, el PC se incrementa, apuntando así a la dirección de la instrucción que será ejecutada a continuación de la que se está ejecutando en el momento. Las instrucciones de transferencia de control modifican el valor del PC.
- El puntero de pila (SP: Stack Pointer) es el registro que almacena direcciones de datos en la pila.

La memoria del microcontrolador es el lugar donde las instrucciones son almacenadas del programa y los datos que manipula. En un microcontrolador siempre hay dos tipos de memoria: la memoria RAM (Random Access Memory) y la memoria ROM (Read Only Memory).

- La memoria RAM es una memoria de lectura escritura, que además es volátil, es decir, pierde la información almacenada cuando falla la energía que alimenta la memoria.

- La memora ROM es una memoria de sólo lectura y no es volátil.

Tanto la memoria RAM como las memorias ROM son de acceso aleatorio, pero la costumbre ha dejado el nombre de RAM para las memorias de lectura y escritura. El término "acceso aleatorio" se refiere a que el tiempo necesario para localizar un dato no depende del lugar de la memoria donde este almacenado. En las memorias de acceso secuencial, en cambio, cuando más alejado esté un dato de la posición a la que se ha accedido por última vez, más se tarda en localizarlo.

La memoria ROM se emplea para almacenar permanentemente el programa que debe de ejecutar el microcontrolador. En la memora RAM se almacenan temporalmente los datos con los que se trabaja el programa. Un número creciente de microcontroladores dispone de alguna memoria no volátil de tipo EEPROM para almacenar datos fijos o que sólo sean cambiados esporádicamente.

La cantidad de memoria ROM disponible es normalmente muy superior a la cantidad de memoria RAM. Esto obedece a dos razones. La primera es que la mayoría de las aplicaciones requieren programas que manejan pocos datos. La segunda razón es que la memoria RAM ocupa mucho más espacio en el circuito integrado que la memoria ROM, de modo que es mucho más costosa que está.

2.1.2 Características de los microcontroladores Atmel

Los AVR son una familia de microcontroladores RISC de Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip [13].

El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento.

El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador son utilizados como punteros de 16 bits al espacio de memoria externa, bajo los nombres X, Y y Z. Esto

es un compromiso que se hace en arquitecturas de ocho bits desde los tiempos de Intel 8008, ya que su tamaño de palabra nativo de 8 bit (256 localidades accedidas) es pobre para direccionar. Por otro lado, hacer que todo el banco superior de 16 registros de 8 bit tenga un comportamiento alterno como un banco de 8 registros de 16 bit, complicaría mucho el diseño, violando la premisa original de su simplicidad. Además, algunas instrucciones tales como 'suma inmediata' faltan; ya que la instrucción 'resta inmediata' con el complemento a dos puede ser utilizada como alternativa.

Los microcontroladores AVR tienen un segmentado ('pipeline') con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.

El set de instrucciones de los AVR es más regular que la de la mayoría de los microcontroladores de 8-bit (por ejemplo, los PIC). Sin embargo, no es completamente ortogonal:

- Los registros punteros X, Y y Z tienen capacidades de direccionamiento diferentes entre sí.
- Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31.
- Los registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.
- La instrucción CLR afecta los 'flag', mientras que la instrucción SER no lo hace, a pesar de que parecen ser instrucciones complementarias (dejar todos los bits en 1, y dejar todos los bits en 0 respectivamente).
- Los códigos de operación 0x95C8 y 0x9004 hacen exactamente lo mismo (LPM).

2.1.3 Componentes del microcontrolador

El microcontrolador AVR requiere de pocos componentes externos para comenzar a utilizarlo. Estos componentes son el circuito de reset y el circuito de reloj. Inclusive pueden llegar a ser opcionales en algunos microcontroladores [13].

- **Reloj:**

Para el funcionamiento del AVR, se requiere una fuente de pulsos de reloj, la cual se encargue de suministrar al AVR con una frecuencia de trabajo al reloj del CPU del microcontrolador. Este reloj del CPU está ligado a los módulos de los registros de

propósito general, registro de estado, registros de memoria de datos entre otros. Al detener el reloj del CPU, se inhibe al núcleo para realizar operaciones o cálculos.

Una fuente de reloj externa confiable, es un cristal o un oscilador. El microcontrolador atmega48, tiene la característica de que puede utilizar una fuente de reloj interna, pre calibrado para una frecuencia de 8 MHZ, y la posibilidad de utilizar un divisor de reloj entre 8, lo que resulta en una frecuencia de trabajo de 1 MHz.

- **Reset:**

El reset es una acción con la cual se "inicia" el trabajo de los microcontroladores. Esta acción se ejecuta cuando se aplica una señal (denominada reset) a una terminal, designado también como reset. El efecto práctico de la señal es poner el contador de programa (PC) en un valor predeterminado (por ejemplo, PC = 0), haciendo así que el microprocesador o microcontrolador comience a ejecutar las instrucciones que están a partir de esa posición de memoria apuntada por el AVR.

El circuito de reset es aquel que permite regresar todos los registros de entradas y salidas a sus valores iníciales y empezar a ejecutar el programa en el vector del reset.

Cuando una fuente de reset se activa, todos los puertos de entradas y salidas regresan inmediatamente a sus estados iníciales; sin requerir ningún ciclo de reloj.

Una vez que todas las fuentes de reset son desactivadas, transcurre un ciclo de espera (retardo), que amplía la duración del reset interno, permitiendo que las fuentes de poder almacenen un nivel estable antes de comenzar con las operaciones regulares. Este tiempo de espera puede ser seleccionado por el usuario a través de los bits fusibles CKSEL.

Las fuentes de reset del microcontrolador atmega8 son las siguientes.

- Reset de energizado: cuando el voltaje es aplicado por primera vez.
- Reset externo: cuando se aplica un nivel lógico al pin de reset.
- Reset de watchdog: cuando expira el contador del watchdog (si es que este está habilitado).
- Reset de brown-out: reset de protección ante caídas de tensión (si es que está habilitado).

Puertos de entrada salida:

El AVR Atmega48 consiste en tres puertos de entrada y salida (I/O). Cada puerto de entrada y salida consiste en tres registros: DDRx, PINx y PORTx.

- Registro DDRx:

El registro DDRx configura la dirección que tienen los datos en el puerto, si será una entrada o una salida. Escribir un uno a un bit de este registró, configura el pin correspondiente al bit como salida. Escribir un cero lo hace entrada.

- Registro PINx:

Lee el estado de PORTx, independientemente del estado de DDRx. Básicamente sirve para leer el estado del pin del puerto cuando este se ha configurado como entrada.

- Registro PORTx:

Si el pin está configurado como salida escribir un uno o un cero en el bit correspondiente de este registro ocasiona que la salida en este pin sea uno o cero. Si el pin está configurado como entrada, escribir un uno en el bit correspondiente de este registro, habilita la resistencia de pull up. Escribir un cero, estando configurado como salida, deshabilita la resistencia de pull up.

2.1.4 USART AVR (comunicación serial)

La comunicación serial es un proceso de envío de múltiples bits de datos sobre un solo alambre. Los bits de un byte serial están separados en tiempo tal que el dispositivo receptor puede determinar los niveles lógicos de cada bit [13].

El termino USART, viene de receptor transmisor síncrono asíncrono universal, es una forma de comunicación entre dispositivos que tengan esta capacidad, donde los datos pueden ser enviados en grupos de 5, 6, 7, 8 o de 9 bits pero bit por bit, esto es en serie, por eso se dice que esta es una comunicación serial, en esta sección se comentará sobre la comunicación serial asíncrona utilizando el módulo USART del microcontrolador AVR, con el módulo USART AVR el microcontrolador puede comunicarse e intercambiar datos con el ordenador, con otros microcontroladores, etc.

Para la comunicación serial asíncrona entre microcontroladores y para la comunicación entre el microcontrolador y el ordenador, se necesitan 2 hilos de conducción para la

transmisión y recepción de datos, y un hilo de conducción para la conexión de los comunes o GND que tienen que ser los mismos, para la comunicación serial entre el microcontrolador y el ordenador se seguirá la norma RS232.

En la comunicación USART AVR asíncrona, uno de los hilos será para la transmisión de los datos de un dispositivo a otro y el otro hilo será para la recepción de datos entre un dispositivo a otro, la transmisión y la recepción pueden ocurrir en forma simultánea, lo que si se tiene que cumplir es que la frecuencia de trabajo de ambos dispositivos tiene que ser la misma, a esto se le conoce como los baudios que viene a ser la cantidad de bits por segundo que se transmitirán entre ambos dispositivos. En la Figura 9, se muestran marcados los pines de Rx y Tx del microcontrolador ATmega48A

- El pin RXD es el pin para la recepción de datos.
- El pin TXD es el pin para la transmisión de datos.
- El pin RXD del AVR tiene que ser conectado al pin TX o TXD del otro dispositivo.
- El pin TXD del AVR tiene que ser conectado al pin RX o RXD del otro dispositivo
- Los comunes GND de ambos dispositivos también tienen que estar conectados entre sí.

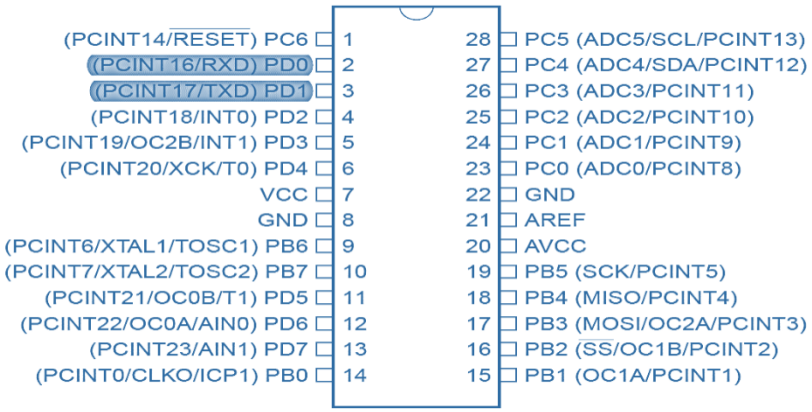

Figura 9 Terminales del microcontrolador ATmega48A, tomado de (13).

Los niveles de tensión con los que trabajan los pines del módulo USART AVR son de 0V y 5V un bajo será 0V mientras que un alto será 5V, por eso cuando la comunicación es entre microcontroladores la conexión entre pines se puede hacer directamente, pero cuando la comunicación es entre el microcontrolador y un ordenador la conexión entre pines tiene que hacerse a través de un conversor de nivel como el MAX232, ya que los niveles de tensión para la comunicación serial del ordenador son mayores que para el

AVR típicamente entre -12V y 12V, además de trabajar con lógica negativa, esto es para el ordenador un bajo será 12V mientras un alto será -12V.

El socket donde se conecta el AVR con el ordenador para la comunicación serial es del tipo DB9 y se conoce como puerto serie, pero resulta que este tipo de puerto ya no viene en los ordenadores portátiles que son los que hoy en día la mayoría utiliza, lo que traen ahora son puertos USB, por lo que para realizar la comunicación serial con el módulo USART AVR será necesario la utilización de un conversor SERIE-USB.

El conector DB9 consta de 9 pines, hay conectores hembra y conectores macho, de este conector solo se utilizarán 3 pines, uno para el pin RX, otro para el pin TX y el otro para tierra, estos pines tienen una numeración que hay que respetar.

Numero	Nombre	Función
1	CD	Carrier detect
2	RXD	Recepción de dato
3	TXD	Transmisión de dato
4	DTR	Data terminal ready
5	GND	Tierra del sistema
6	DSR	Data set ready
7	RST	Request to send
8	CTS	Clear to send
9	RI	Ring indicator

Figura 10 Elementos del byte de la comunicación serial, tomado de (13).

En la imagen anterior [Figura 10] se muestra los elementos del byte de la comunicación serial asíncrona. Esta figura muestra la definición de cada bit de la palabra serial. El mensaje inicia con una espera estando en alto y pasando a bajo para iniciar el mensaje. El bit de arranque o inicio toma el valor de un bit completo y es seguido por los ocho bits de él byte de dato que se muestra sobre una línea serial en un orden inverso, esto es, el bit menos significativo aparece primero y el bit más significativo aparece al último. El bit de fin de mensaje sigue al bit más significativo corresponde a un uno lógico, el mismo valor para el bit de espera [Figura 11].

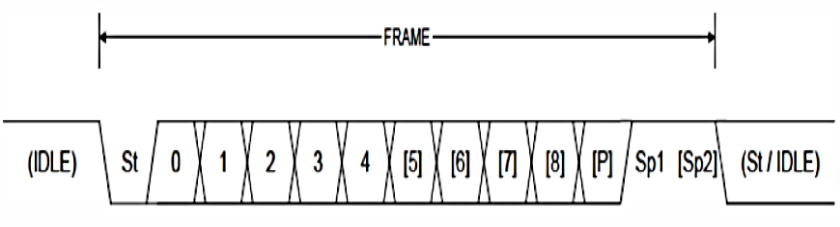

Figura 11 Trama de bits, tomado de (13).

- St: Bit de inicio siempre en cero lógico.
- (n): Bits de datos (0 a 8).
- P: Bit de paridad, puede ser par o impar.
- Sp: Bit o bits de fin de mensaje, siempre en uno lógico.
- Idle: Bit de espera. No existe transferencia de dato sobre la línea de comunicación.

El flanco de bajada inicia la secuencia de transmisión en el receptor serial. Iniciando desde el flanco de bajada del bit de inicio, el receptor espera 1.5 bits antes de muestrear la línea receptora. Después de eso, el receptor espera 1 bit por cada bit, por lo tanto, se muestrea cada bit de dato sucesivo en el centro del periodo para máxima eficiencia.

Afortunadamente, todo el tiempo del formateo del byte serial, el muestreo de los bits seriales, y la suma del bit de inicio y el bit de fin de transmisión son manejados automáticamente por el Universal Synchronous Asynchronous Receiver Transmitter (USART).

El transmisor receptor serial asíncrono y síncrono universal (USART por sus siglas en inglés) es una unidad de comunicación periférica muy flexible, que en el microcontrolador ATmega48 permite entre otras funciones.

- Operación full dúplex (se puede enviar y recibir datos simultáneamente).
- Operación síncrona y asíncrona.
- Operación en modo maestro esclavo con reloj síncrono.
- Soporta frames de 5, 6, 7, 8 y 9 datos y 1 o 2 de parada.
- Generador de paridad par o impar.
- Detección de errores (sobre flujo de datos, error en el frame).
- Filtrado de ruido (inicio falso, filtro digital).
- Generación de interrupciones por transmisión completa, por recepción completa o por registro de datos de transmisión vacía.
- Comunicación entre multiprocesadores.
- Doblador de velocidad modo de comunicación asíncrona.

2.1.5 Sistema de interrupciones

Una interrupción es la ocurrencia de un evento producido por algún recurso del microcontrolador, que ocasiona la suspensión temporal del programa principal. La CPU atiende al evento con una función conocida como rutina de servicio a la interrupción (ISR,

Interrupt Service Routine). Una vez que la CPU concluye con las instrucciones de la ISR, continúa con la ejecución del programa principal, regresando al punto en donde fue suspendida su ejecución.

Manejo de las interrupciones.

Si en un programa se utilizan las interrupciones, se requiere:

- Configurar el recurso o recursos para monitorear el evento o eventos.
- Habilitar a la interrupción o interrupciones (individual y global, en cada caso).
- Continuar con la ejecución normal de la aplicación.

En aplicaciones que utilizan interrupciones, es frecuente ciclar al programa principal en un lazo infinito sin realizar alguna actividad, con ello, el MCU permanece ocioso, dejando la funcionalidad del sistema a las ISRs.

Cuando ocurre una interrupción, el microcontrolador automáticamente realiza lo siguiente:

- Concluye con la instrucción bajo ejecución.
- Desactiva al habilitador global de interrupciones, para que no pueda recibir una nueva interrupción mientras atiende a la actual.
- Respalda en la pila al PC (previamente incrementado).
- Asigna al PC una dirección de los vectores de interrupciones, para dar paso a la ISR.
- Atiende al evento con la ISR.

Cuando una ISR termina (con la instrucción RETI, si se programó en ensamblador), en el MCU ocurre lo siguiente:

- Se limpia la bandera del evento que generó la interrupción.
- El habilitador global se activa.

Capítulo 3. Desarrollo

3.1 Descripción del proyecto

En este apartado se muestra de forma detallada los componentes y funcionamiento del sistema realizado, obteniendo así, un prototipo satisfactorio como proyecto. En la siguiente imagen se muestra de forma general los diagramas del sistema [Figura 12].

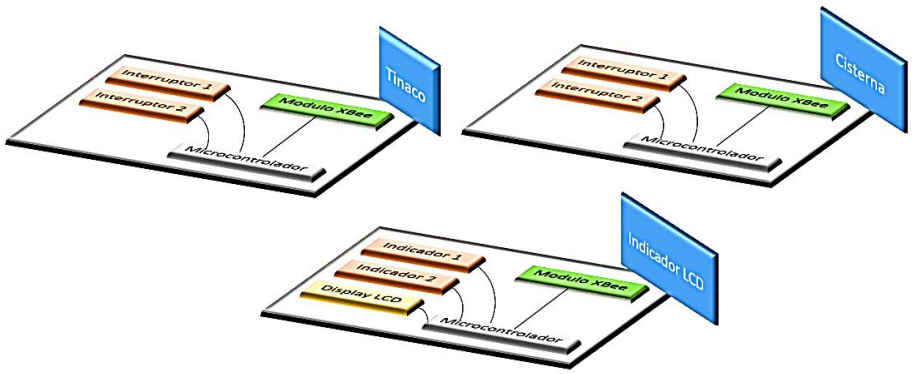

Figura 12 Diagrama a bloques general del sistema.

En el hogar habitualmente se tiene un sistema simple de llenado de tinaco, en el que, solo es utilizado un flotador que al alcanzar el nivel de agua máximo impide que se esparza el agua y se tire, como se muestra en la siguiente figura [Figura 13].

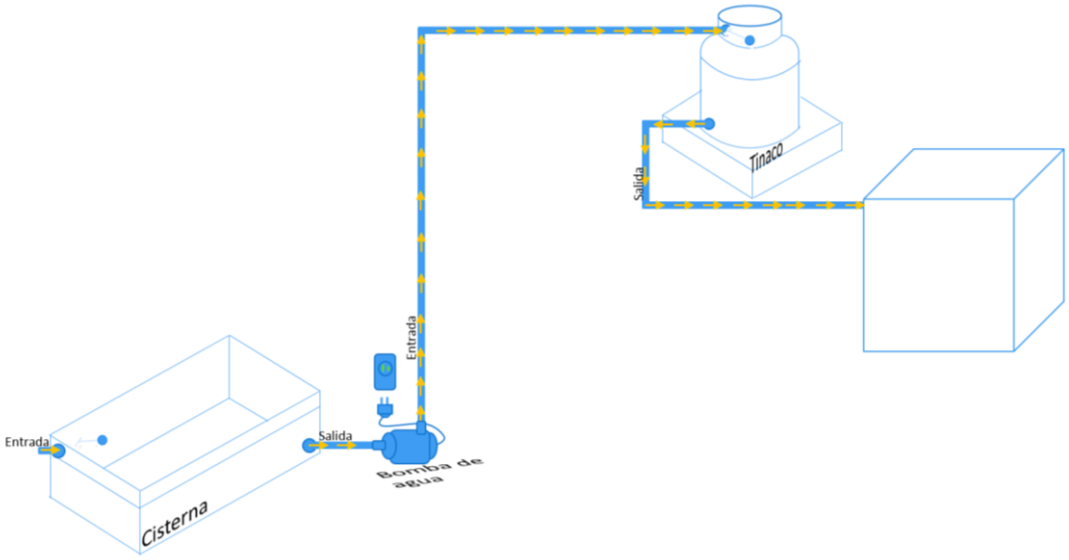

Figura 13 Diagrama general, sistema de llenado de tinaco simple.

Como se observa en la siguiente imagen, el sistema se conforma con 2 interruptores de nivel de agua que accionan circuitos individuales y procesos conectados entre sí [Figura 14]. Cada circuito está conectado a un sistema colocado en ciertas partes del hogar y estos

accionan tareas específicas, por ejemplo, llenar un tinaco, verificar una cisterna, poder visualizar este sistema. Para poder tener una comunicación entre los circuitos segura y satisfactoria entre los módulos XBee, contienen instrucciones de seguridad que hacen que la comunicación no sea invadida por intrusos que decidan cambiar alguna configuración en el sistema creado, además de que este tipo de módulos pueden comunicarse por medio de un vínculo transparente entre ellos.

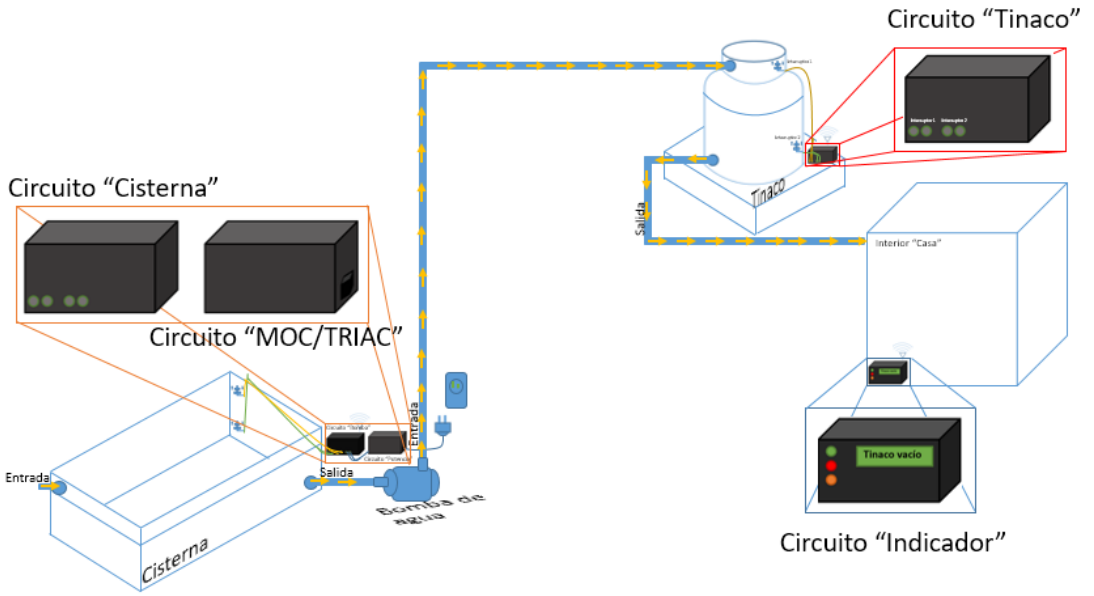

Figura 14 Diagrama general de sistema a implementar.

Siendo más concretos, el sistema realiza las siguientes acciones: se inicia con la verificación de agua en una cisterna, donde tendrá dos estados disponibles expresados en dos interruptores de nivel de agua que interpretaran las opciones LLENO o VACIO, si la cisterna está en modo VACIO el sistema no se accionará, por el contrario, si la cisterna está en modo LLENO accionará una bomba de agua instalada en la cisterna, iniciará el transportado de agua de la cisterna a un tinaco, si y solo si, el estado del tinaco es VACIO. En el tinaco se encuentran 2 interruptores de nivel de agua al igual que en la cisterna que interpretan las opciones LLENO o VACIO, dependiendo de su estado de estos interruptores, por ejemplo, si el estado se encuentra en VACIO, enviará una señal al aire que alcanzará hasta el circuito que acciona la bomba y este la encenderá hasta que el estado del tinaco cambie a LLENO, volverá a enviar una señal indicando que el tinaco está lleno, por lo que desactivara la bomba, dicho lo anterior podemos visualizarlo en el siguiente diagrama de flujo [Figura 15].

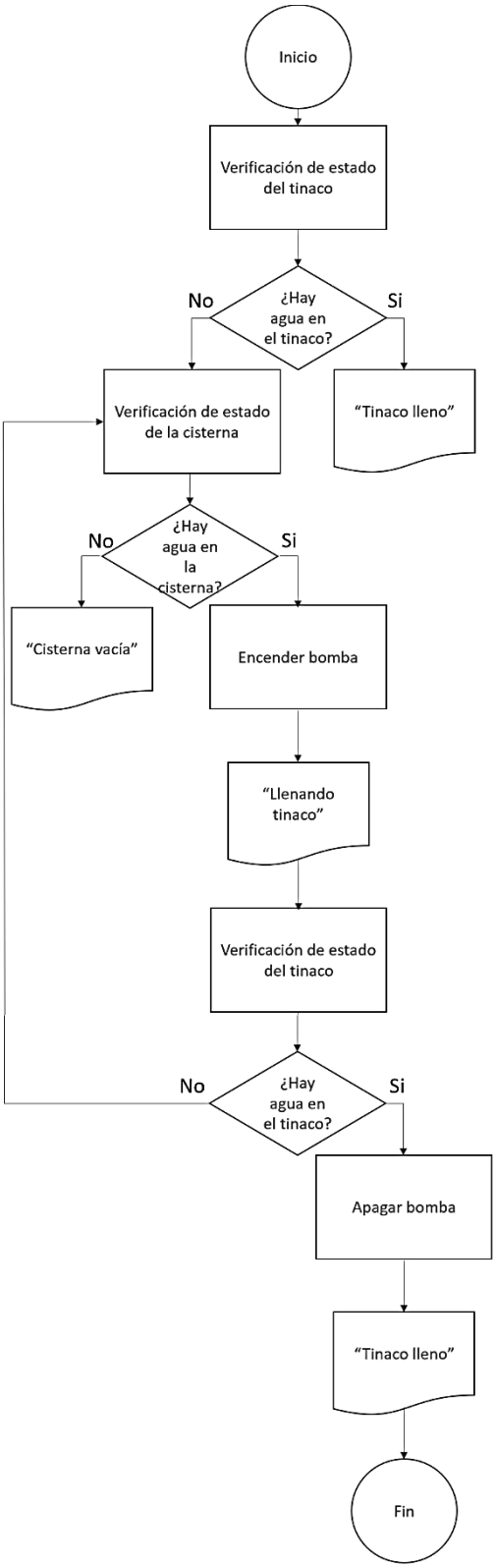

Figura 15 Diagrama de flujo del sistema.

Este proceso podrá ser visualizado en otro circuito que recibirá las mismas instrucciones y que por medio de una pantalla LCD el usuario verificará el estado del sistema.

3.2 Etapa de censado

3.2.1 Módulos RF XBee

Los dispositivos principales para el desarrollo del proyecto son los módulos XBee [Figura 16], los cuales nos permiten realizar la comunicación de manera inalámbrica entre cada uno de los diferentes circuitos que integran el sistema; este sistema se conforma por tres módulos XBee, de los cuales dos se comportan como dispositivos finales y uno como coordinador, esto debido a las características de la serie que utilizamos ya que la serie 1 de estos módulos no nos permite que en la red tengamos un módulo configurado como router. Cabe mencionar que estos módulos se encuentran en cada uno de los circuitos que conforman al sistema (tinaco, cisterna e indicador).

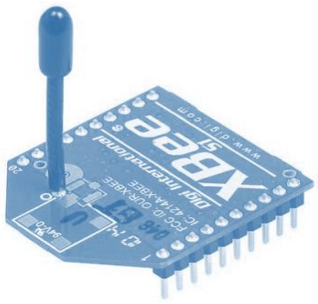

Figura 16 Modulo XBee, tomado de (14).

3.2.2 Microcontrolador ATMEGA48A

El siguiente elemento que se utilizó en el sistema es el microcontrolador [Figura 17], este dispositivo nos permite tener el manejo de los módulos XBee por medio la programación previamente realizada y de esta manera determinar cuándo envían y reciben las señales los módulos. Este dispositivo también se encuentra en los tres circuitos del sistema (tinaco, cisterna e indicador).

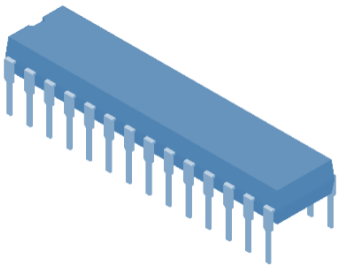

Figura 17 Microcontrolador ATMEGA48A, tomado de (13).

3.2.3 Interruptor de nivel de agua

Este elemento utilizado para poder detectar el nivel de un líquido o fluido es parte importante del sistema, este dispositivo como se muestra en los diagramas está conectado al microcontrolador, el interruptor al percibir un cambio de nivel por parte del líquido el cual hace que el flotador suba o baje y que funciona por medio de magnetismo que al activarse envía la señal al microcontrolador para que este a su vez indique al módulo XBee si envía o no información a otro de los módulos para realizar el proceso pertinente. En el proyecto se utilizaron 4 de estos interruptores [Figura 18], dos están colocados en el tinaco, indicándonos un nivel alto y un nivel bajo; los otros 2 interruptores están instalados en la cisterna, de esta manera se puede saber si es pertinente el encendido de la bomba de agua dependiendo el nivel de agua de la cisterna.

Figura 18 Interruptor de nivel de agua, tomado de (15).

La siguiente imagen [Figura 19] muestra el diagrama eléctrico del circuito utilizado en la cisterna, en el, se visualiza el microcontrolador con sus respectivos puertos de entrada, así como de salida, además de sus fuentes de alimentación individuales. En este circuito podemos observar en el Puerto D los pines que conectan al módulo XBee (Tx y Rx), además de los interruptores de nivel de agua.

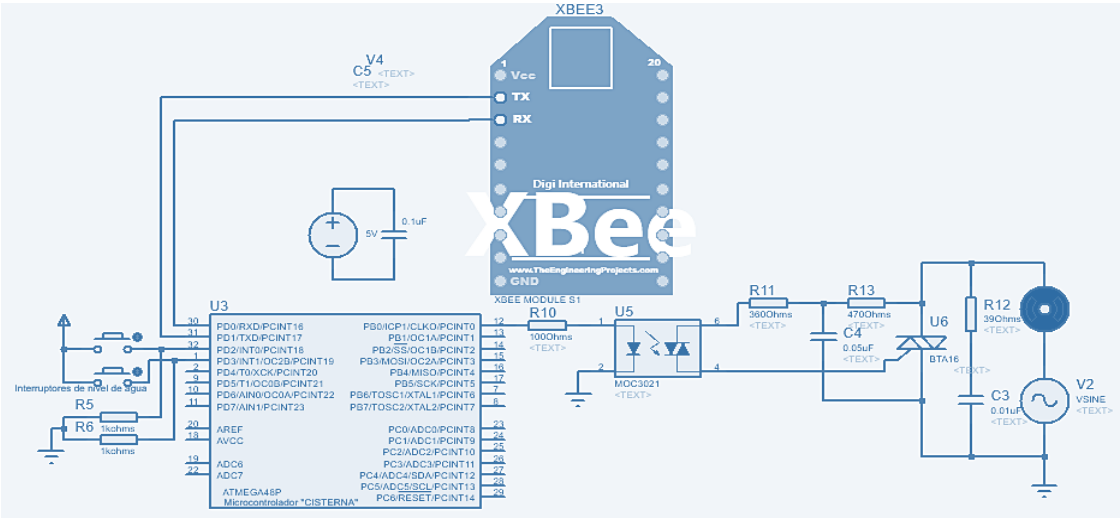

Figura 19 Diagrama electrónico "Cisterna".

A continuación, tenemos la placa electrónica realizada para la cisterna, [Figura 20].

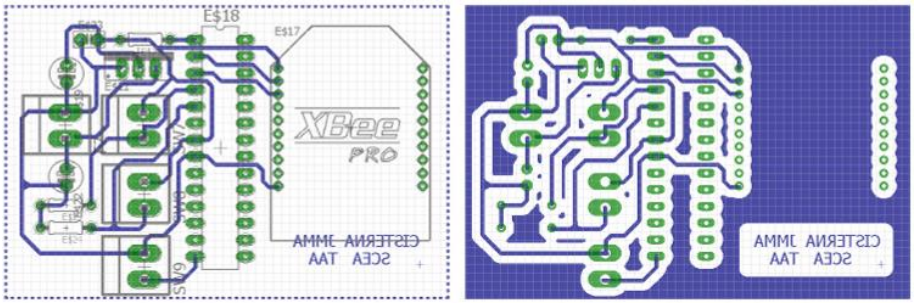

Figura 20 Placa electrónica "Cisterna".

El siguiente circuito corresponde al utilizado en un tinaco [Figura 21], podemos ver que cuenta con sus respectivos interruptores de nivel de agua, así como su fuente de alimentación. Como en el circuito anterior, podemos observar de igual forma los pines que conectan al módulo XBee y los dos interruptores de nivel de agua.

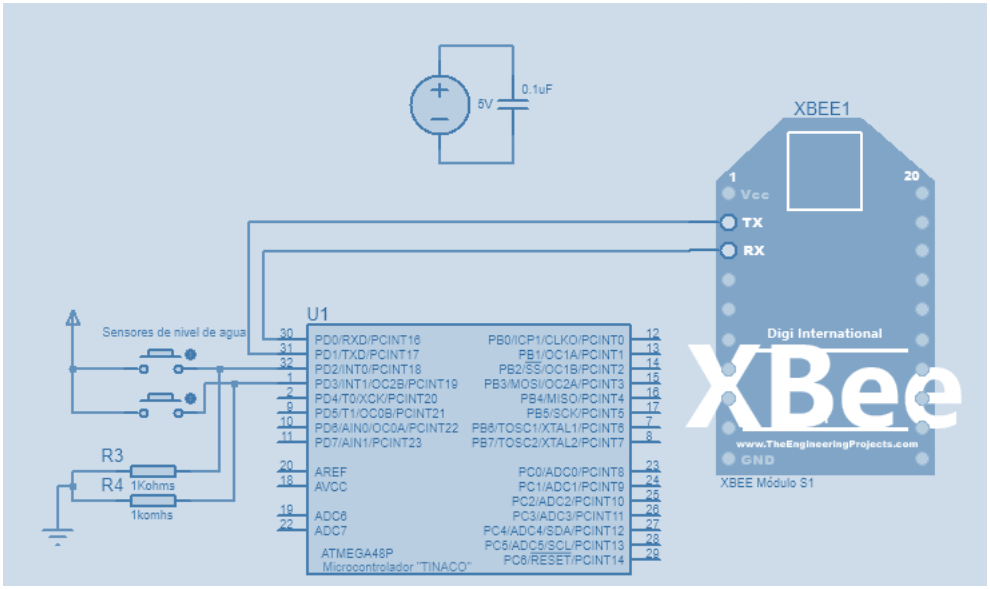

Figura 21 Diagrama electrónico "Tinaco".

A continuación, se muestra la placa realizada para el tinaco [Figura 22].

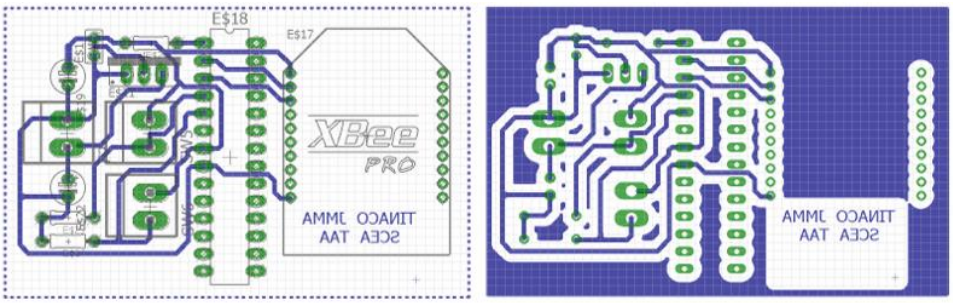

Figura 22 Placa electrónica "Tinaco".

Etapa de potencia.

3.2.4 MOC 3021

El Optodiac MOC3021 consta de un diodo emisor de infrarrojos, acoplado ópticamente a un interruptor de Silicio y está diseñado para aplicaciones que requieren disparo aislado de Triac [16][17], bajo corriente de conmutación, un alto aislamiento eléctrico (a 7500 Vpico), pequeño tamaño y de bajo costo [Figura 23].

Este dispositivo nos permite aislar la parte de control con la parte de potencia en el circuito de la cisterna; básicamente el circuito de la cisterna se conforma por dos circuitos, el de control que esta conformador por los interruptores de nivel, el microcontrolador y el módulo XBee; el segundo circuito es el de la parte de potencia conformado por el MOC3021 (Optodiac), el Triac BTA16 600B, la bomba de agua, como se trata de un circuito de potencia estos elementos son manejados con corriente alterna (AC), debido a esto tanto el Triac como el MOC3021 llevan redes de amortiguamiento para transitorios de voltaje y de corriente (red snubber RC), cabe mencionar que dichas redes snubber fueron obtenidas de la hoja de especificaciones del MOC3021

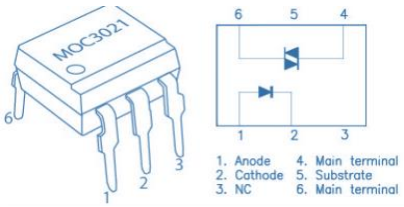

Figura 23 MOC 3021, tomado de (18).

3.2.5 Triac BTA16 600B

Este dispositivo al igual que el MOC3021 mencionado anteriormente también forma parte del circuito de potencia que conforma al circuito de la cisterna, el Triac nos permite encender la bomba de agua cuando el nivel del tinaco este bajo, pero siempre y cuando el nivel de agua en la cisterna este en alto, el accionamiento del Triac depende del MOC3021 y obviamente de la parte de control del circuito de la cisterna [Figura 24].

Figura 24 Triac BTA16 600B, tomado de (19).

Protección contra dv/dt

Las subidas repentinas de voltaje aplicadas al Triac, pueden llegar a provocar activaciones imprevistas del mismo. Una conexión no adecuada a redes de circuitos próximos con cargas inductivas puede dar lugar a variaciones bruscas de la tensión de alimentación del Triac.

El método comúnmente más empleado para limitar el tiempo de subida de la tensión consiste en colocar un capacitor en paralelo con el Triac, si este capacitor se descargara de manera brusca durante la activación del Triac, introduciría una sobreintensidad y una di/dt excesivas que podrían dañar el Triac, por eso es necesario limitar la corriente de descarga colocando una resistencia (de 20Ω a 100 Ω) en serie con el capacitor. El circuito RC resultante se debe de colocar en paralelo con el Triac y lo más cerca posible de él.

Cálculos

- Para R_1

Por Ley de Ohm

$$V = RI \rightarrow R = \frac{V}{I} \qquad \text{- Ecuación 1}$$

$$\therefore R_1 = \frac{3.3\ V}{50\ mA} = 66\Omega \simeq \mathbf{100\Omega}\ \ Valor\ comercial$$

- Para R_2

$$R_2 = \frac{V_{TM} - V_F}{I_{FT}} \qquad \text{- Ecuación 2}$$

$$R_2 = \frac{3\ V - 1.15\ V}{7\ mA} = 264.28\Omega \simeq \mathbf{330\Omega}\ Valor\ comercial$$

Donde:

$V_{TM} = Voltaje\ maximo\ en\ estado\ activo.$

$V_F = Voltaje\ de\ Foward.$

$I_{FT} = Corrinete\ de\ activacion\ necesaria\ para\ salida.$

*Valores tomados de la hoja de especificaciones del MOC3021

- Para R_3

Por Ley de Ohm

$$R_3 = \frac{V_{AC}}{I_{Bomb}} \qquad \text{- Ecuación 3}$$

$$R_3 = \frac{120\ V}{3.3\ A} = 36.36\Omega \simeq \mathbf{39\Omega}\ Valor\ comercial$$

- Para R_4

$$R_4 = \frac{V - V_{gt}}{i_{gt}} \qquad \text{- Ecuación 4}$$

$$R_4 = \frac{3.3\ V - 1.7\ V}{350x10^{-5}\ Amp} = 457.14\Omega\ \simeq \mathbf{470\Omega}\ Valor\ comercial$$

Donde:

$V = Voltaje\ en\ ese\ nodo$

$V_{gt} = Voltaje\ de\ disparo\ de\ compuerta$

$i_{gt} = Corriente\ de\ disparo\ de\ compuerta$

*Valores tomados de la hoja de especificaciones del BTA16

- Para C_1

Primero obtenemos el Voltaje de Rizo

$$V_r = V_2 - V_1 \qquad \text{- Ecuación 5}$$

Donde:

$$V_2 = 63\%\ de\ 120\ V = 75.6\ V$$

$$V_1 = 10\%\ de\ 120\ V = 12\ V$$

$$\therefore\ V_r = 75.6\ V - 12\ V = 63.6\ V$$

Ahora obtenemos el Factor de Rizo

$$F_r = \frac{V_r}{V_{max}} x100\% \qquad \text{- Ecuación 6}$$

$$F_r = \frac{63.6\ V}{120\ V} = 53\%\ \text{ó}\ 0.53$$

$$C_1 = \frac{2 + F_r}{2fR_{Bomb}F_r} = \frac{2 + 0.53}{(2)(120\ Hz)(18x10^6\ \Omega)(0.53)} = \mathbf{0.011}\ \mu\boldsymbol{F}$$

$$*\ f = Frecuencia\ de\ onda\ completa\ en\ AC$$

- Para C_2

$$C_2 = \frac{\tau}{R_4} \qquad \text{- Ecuación 7}$$

El valor típico de τ como constante que está en la hoja de especificaciones del MOC3021.

$$C_2 = \frac{2.14\ \mu seg}{470\ \Omega} = 0.0455\ \mu F\ \simeq \mathbf{0.05}\ \mu \boldsymbol{F}\ Valor\ comercial$$

El diagrama electrónico se muestra a continuación con los valores correspondientes a los calculados.

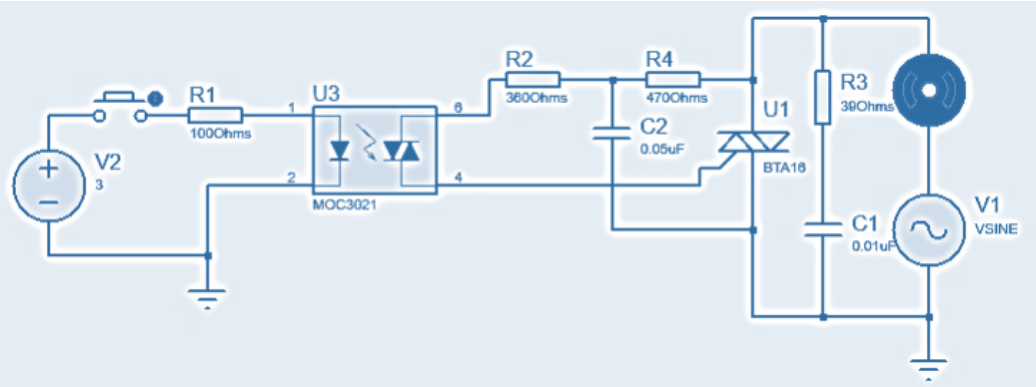

Figura 25 Diagrama electrónico "Bomba de agua".

Se realizó la placa para el encendido de la bomba de agua [Figura 26].

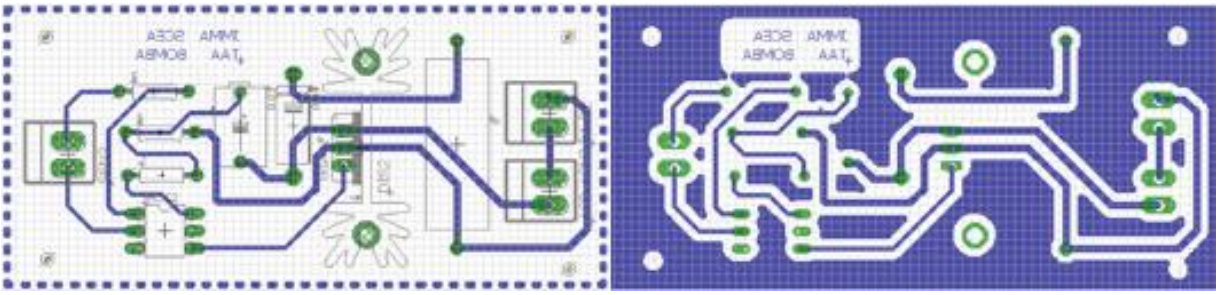

Figura 26 Placa electrónica "Bomba de agua".

3.2.6 Display LCD

Una LCD es un dispositivo que es programable, el display utilizado es del tipo alfanumérico de 2 líneas y 16 caracteres por línea, cada modelo varía el color del display y cuentan con retro iluminador incorporado [Figura 27].

Este elemento se encuentra en el circuito del indicador, el cual nos despliega los estados de los procesos que se están realizando como, por ejemplo, llenando tinaco, tinaco lleno o cisterna vacía, el LCD va conectado a su vez al microcontrolador para poder realizar las tareas previamente programadas el LCD utilizada cuenta con 14 pines dispuestos en las dos líneas que contiene.

Los pines de conexión de los displays LCD incluye un bus de datos completo de 8 bits; un pin de habilitación E (Enable); uno de selección llamado RS (Register Select), que dependiendo de su estado indicara si es una instrucción o si se trata de un carácter a escribir, además, uno denominado R/W (Read/Write) que indica si se va a leer o escribir.

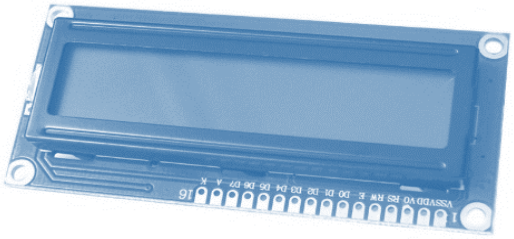

Figura 27 Display de 7 segmentos 2x16, tomado de (20).

Finalmente, un circuito para visualizar la información que arroja los circuitos anteriores y que servirá como medio al usuario para verificar que el sistema está funcionando adecuadamente, esto por medio de un LCD de 16x2 [Figura 28].

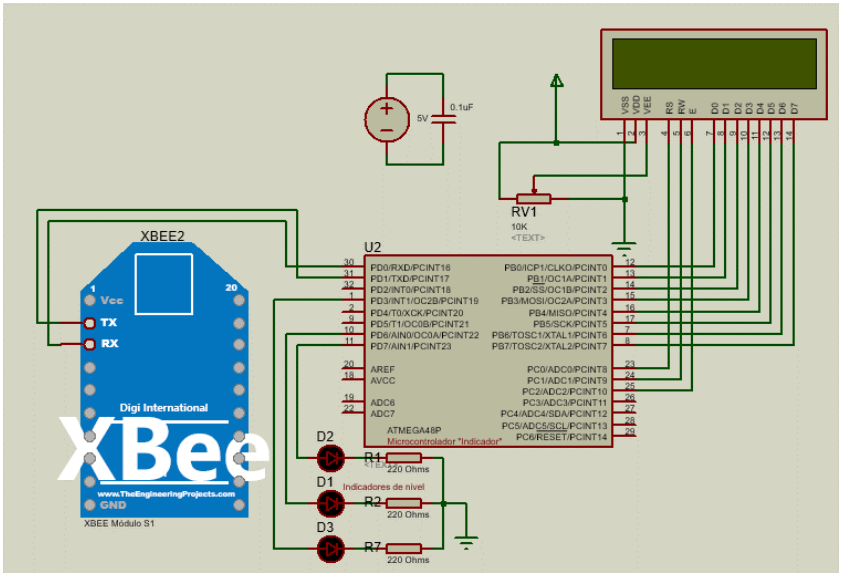

Figura 28 Diagrama electrónico "Indicador LCD".

Se diseñó la placa electrónica del indicador y a continuación se muestra [Figura 29].

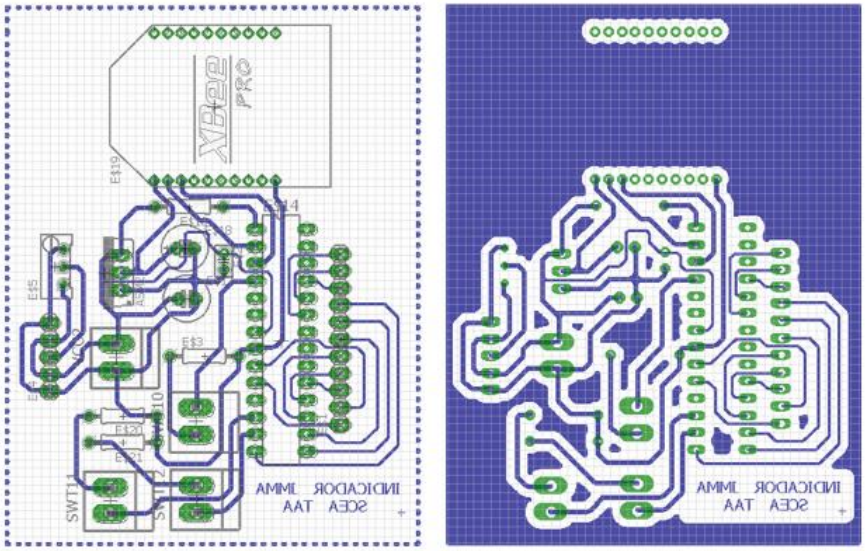

Figura 29 Placa Electrónica "Indicador LCD".

Para evitar la descarga rápida de las pilas al momento del funcionamiento del indicador, se diseñó una fuente de alimentación de 5 V a 500 mA [Figura 30].

Los valores utilizados para el puente de diodos se determinaron a partir del transformador, el cual nos entregan una corriente de 1 Ampere por lo que, la matricula utilizada de diodos comerciales son:

1N4001

Para determinar el capacitor que se utilizara en la fuente de alimentación es necesario determinar el voltaje mínimo que gasta el circuito, por lo que se toma el voltaje mínimo del regulador de voltaje para funcionar adicionado el voltaje necesario del puente de diodos.

$$Voltaje\ de\ entrada = 7\ V_{min} + 1.4\ V = 8.4\ V$$

Determinamos la corriente mínima de entrada del transformador.

$$Corriente\ de\ entrada = \left(\frac{v_2}{v_1}\right) I_2 = \left(\frac{12\ V}{120\ V}\right) 1\ A = 100\ mA$$

$$T = 10\ ms$$

Sabemos que la corriente máxima que entregará el transformador será de:

$$I_{max} = 1\ A$$

El transformador a su vez, específicamente se adquirió para dar en el voltaje secundario un voltaje de 12v

$$V_{max} = V_{pico} = (12\ V)\sqrt{2} = 16.970\ V$$

Teniendo estos valores se procede a determinar el valor del capacitor.

$$C = \frac{(I_{max})\ (T)}{V_{max} - V_{min}} = \frac{(1\ A)(10\ ms)}{16.970\ V - 8.4\ V} = 1166.86\mu F \approx 1100\mu F$$

Dentro de la datasheet de regulador de voltaje LM7805 se muestran los capacitores utilizados para evitar transitorios dentro de la fuente de alimentación.

$C_3 = 0.33\mu F$

$C_2 = 0.1\ \mu F$

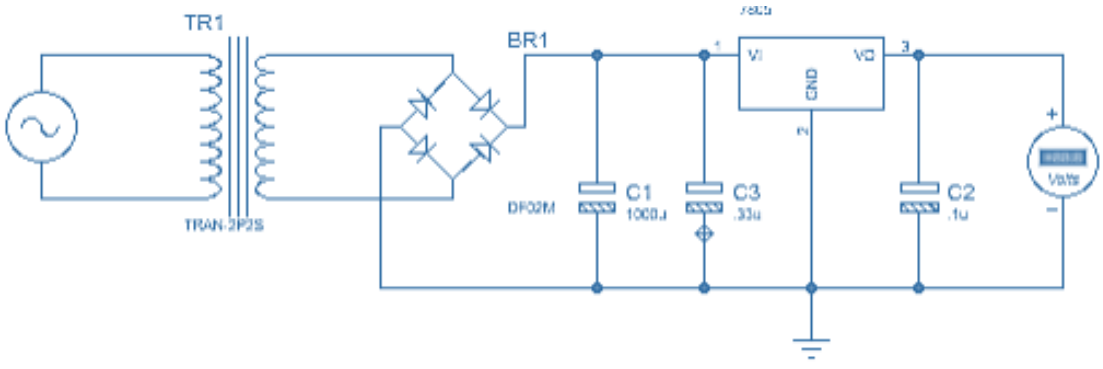

Figura 30 Diagrama electrónico "Fuente de alimentación".

Y también se diseñó la placa electrónica del circuito [Figura 31].

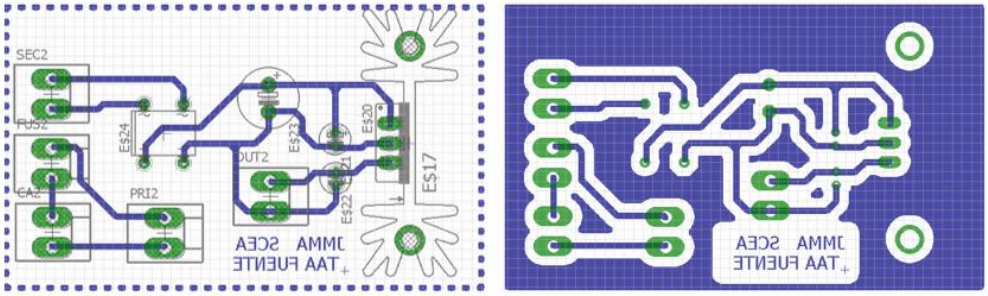

Figura 31 Placa electrónica "Fuente de alimentación".

3.3 Programación de módulos XBee

3.3.1 Comunicación punto a punto

Para la programación de módulos XBee es necesario utilizar una aplicación gratuita llamada XCTU que permite a los desarrolladores interactuar con módulos DIGI RF a través de una interfaz gráfica fácil de usar. En este programa se configura los distintos dispositivos para determinar la posición en el que se encontrará dentro de la red, así como otras especificaciones que se deben de configurar en cada módulo para su interconexión dentro de la red mesh. Algunas de las características que son mencionadas en la página de "DIGI" donde se descargó este software, se enuncian a continuación:

- Puede administrar y configurar varios dispositivos de RF, incluso dispositivos conectados remotamente (por aire).
- La actualización del firmware procesa sin problemas, restaura la configuración del módulo, la manipulación automática de velocidad de transmisión de modo y cambios.
- Dos consolas API y AT específicas, se han diseñado desde cero para comunicarse con sus dispositivos de radio.
- Ahora puede guardar sus sesiones de consola y cargarlas en una PC diferente ejecutando XCTU.
- XCTU incluye un conjunto de herramientas integradas que se pueden ejecutar sin tener conectado ningún módulo de RF:
    - Generador de marcos: genere fácilmente cualquier tipo de marco API para guardar su valor.

- Intérprete de marcos: decodifica un marco API y ve sus valores de marco específicos.
- Recuperación: recupera los módulos de radio que tienen firmware dañado o están en modo de programación.
- Cargar sesión de consola: carga una sesión de consola guardada en cualquier PC con XCTU.
- Prueba de rango: realice una prueba de rango entre 2 módulos de radio de la misma red.
- Explorador de firmware: Navegue a través de la biblioteca de firmware de XCTU.

- Un proceso de actualización le permite actualizar automáticamente la aplicación y la biblioteca de firmware de la radio sin necesidad de descargar ningún archivo adicional.
- XCTU contiene documentación completa y completa a la que se puede acceder en cualquier momento.

En la experimentación se configuraron dos módulos XBee para realizar una comunicación coordinador-dispositivo final y posteriormente se añadió un módulo más para realizar una red tipo estrella.

3.3.2 Configuración de los módulos XBee con X-CTU

En la siguiente figura [Figura 32] se muestra la interfaz del software X-CTU.

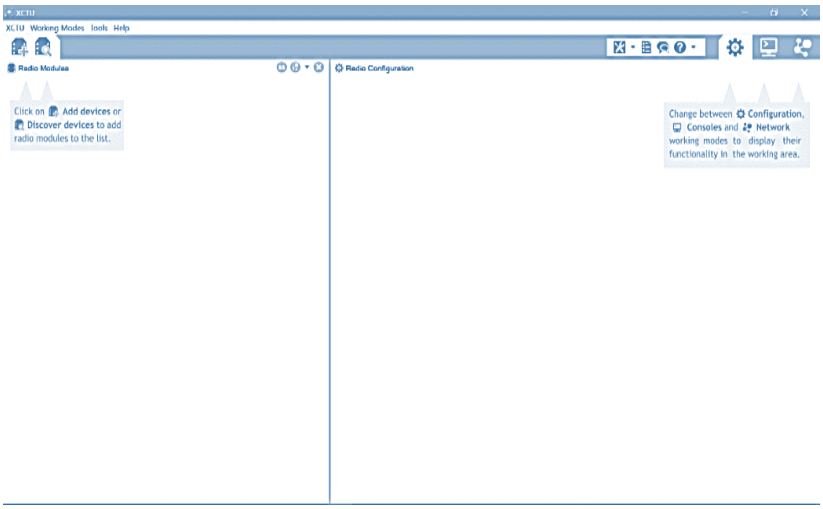

Figura 32 Interfaz software X-CTU.

Se da clic en la opción de buscar algún modulo [Figura 33] que esté conectado a la PC (algún módulo debe estar conectado anteriormente para que sea encontrado).

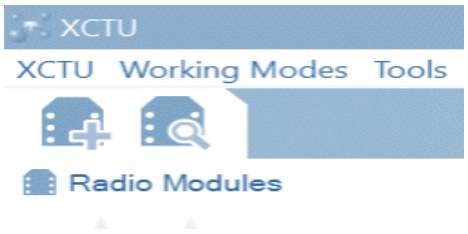

Figura 33 Opción para buscar módulo XBee.

Una vez que se dio clic en la opción anterior, aparecerá la siguiente ventana.

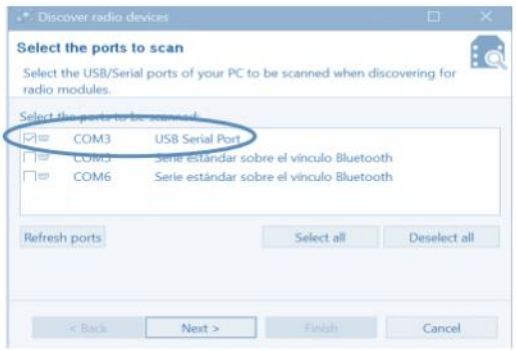

Figura 34 Ventana de selección de puertos para escanear.

En esta ocasión, el módulo XBee nos aparece conectado al puerto COM3 como se muestra en la figura anterior [Figura 34]. En el caso en el cual no se conozca cual es el puerto al cual está conectado nuestro modulo, se puede ingresar a la opción de administrador de dispositivos y verificar en que puerto se encuentra [Figura 35].

Figura 35 Administrador de dispositivos de la PC.

Marcamos la casilla en la cual está el módulo XBee conectado (por lo regular esta se selecciona por default) y posteriormente damos clic en "Next", [Figura 36].

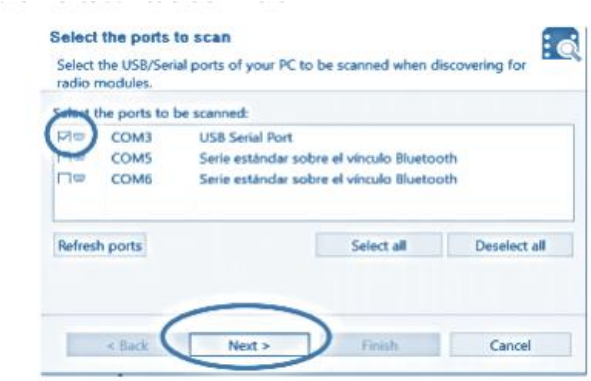

Figura 36 Selección de puertos COM para escanear.

Después de seleccionar el puerto COM, aparcera la siguiente venta en la cual se pueden configurar varios parámetros del puerto donde está el módulo XBee.

En esta ocasión, dejamos los valores que viene por default, ya que sólo es demostrativo para la configuración de los módulos. Damos clic en la opción de "Finish", [Figura 37].

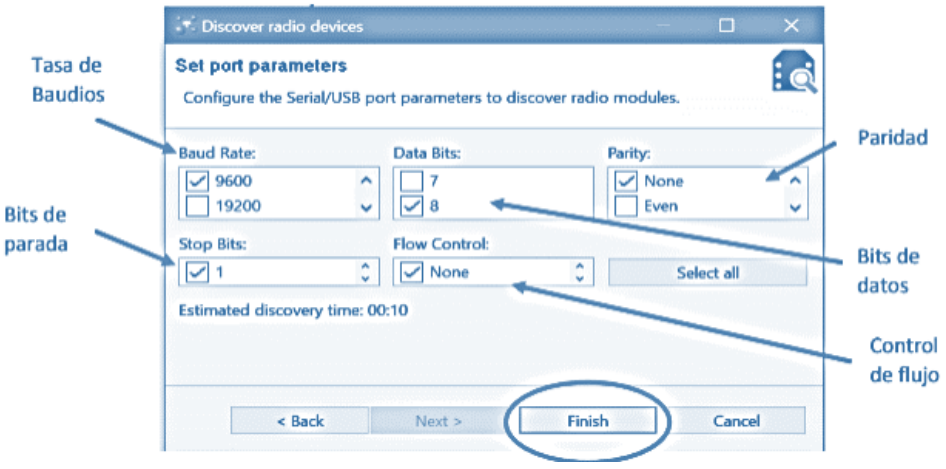

Figura 37 Ventana de selección de parámetros de puerto.

Por default en la opción de tasa de baudios, vienen seleccionados las cantidades de 9600 y de 115200, así también lo que se muestra con las casillas marcadas igualmente viene marcado de forma predeterminada.

Después de seleccionar los parámetros el software empezara a buscar los módulos de radio, [Figura 38].

Figura 38 Ventana de búsqueda de módulos de radio.

La búsqueda de los módulos de radio es muy rápida, ya que encontró el o los módulos aparecerá la siguiente ventana en donde mostrara los módulos encontrados y algunas de sus características, como por ejemplo la dirección MAC, el puerto COM (sabemos que es el COM3) y el nombre del módulo, en esta ocasión el nombre aparece en blanco ya que es la primera vez que se conecta. Damos clic en la opción de "Add selected devices", [Figura 39].

Figura 39 Ventana de módulos encontrados.

Ya que agregamos el dispositivo seleccionado este aparecerá agregado en la parte izquierda de la interfaz del software X-CTU, como se muestra a continuación [Figura 40].

Figura 40 Modulo XBee agregado.

Al poner el cursor sobre el módulo agregado este se pone de color anaranjado, damos clic para que se desplieguen las opciones de configuración del módulo, [Figura 41].

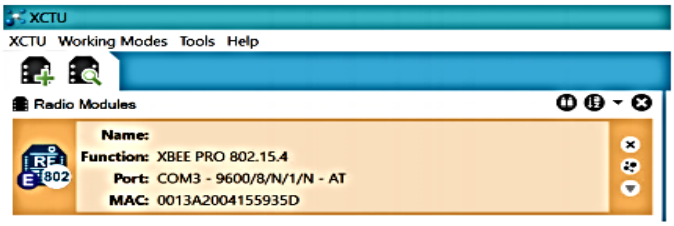

Figura 41 Clic en módulo agregado.

Ya que se da clic en el módulo agregado aparecerán las características a configurar en el módulo XBee [Figura 42].

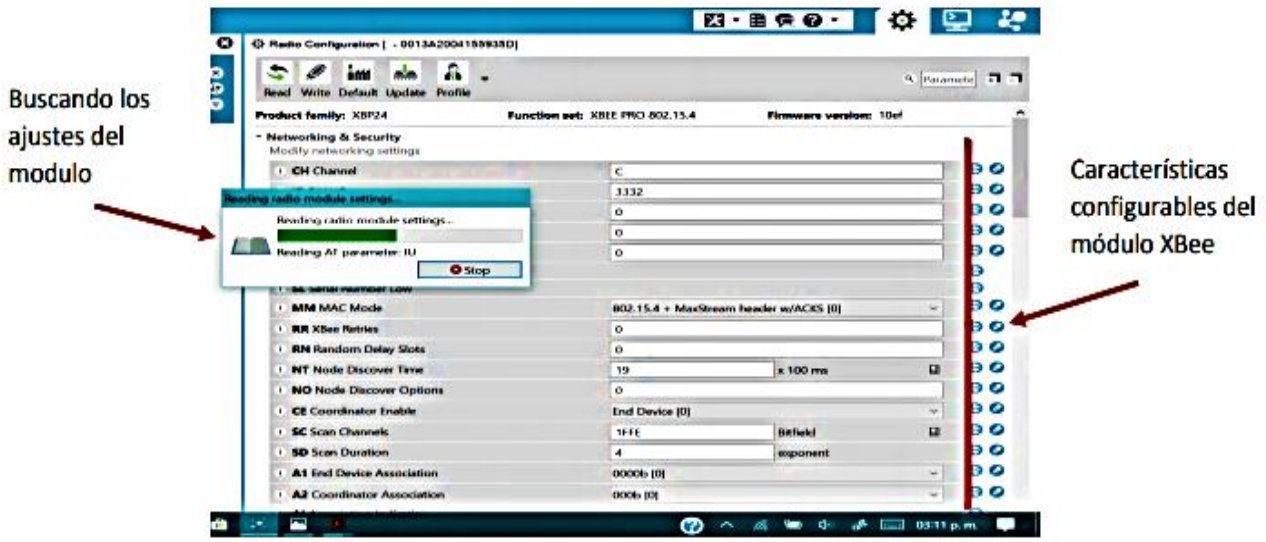

Figura 42 Leyendo ajustes del radio del módulo.

En esta ocasión se configurarán solo algunos parámetros de los módulos XBee, así también se establecerá que módulo es el "Coordinador" y cual el "Dispositivo Final" para poder realizar una conexión punto a punto. Empezamos configurando el módulo que será el "Coordinador" de la red punto a punto. A continuación, se muestran los parámetros a configurar, [Figura 43].

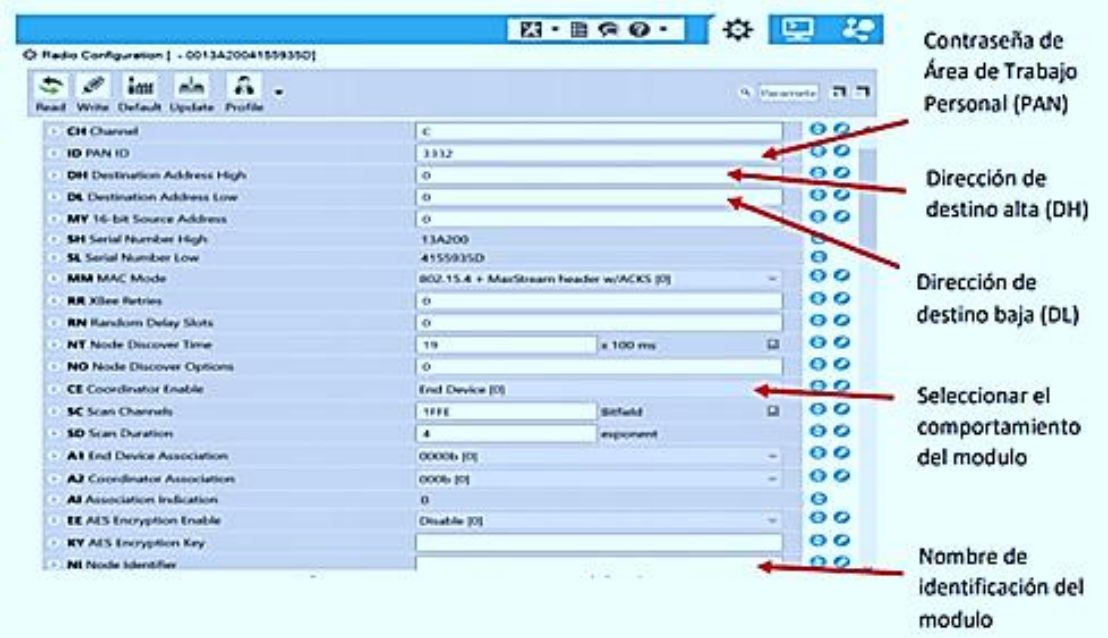

Figura 43 Parámetros a configurar del módulo XBee.

Para la contraseña de Área de Trabajo Personal, se colocará una sencilla para fines demostrativos, la cual será **"1234".**

En dirección de estado alta (DH), se coloca la dirección que viene al reverso de los módulos XBee, por lo general esta dirección es siempre la misma; hay que tener muy en cuenta que la dirección que se ingresa es la del módulo XBee con el cual se comunicara nuestro "Coordinador" (el que se está configurando actualmente) y viceversa, es decir, que cuando se configure el dispositivo final, en estos parámetros se meterá la dirección de destino alta y baja, pero del "Coordinador". Posteriormente, se muestra con imágenes lo mencionado anteriormente [Figura 44], [Figura 45].

Figura 44 Modulo coordinador visto de ambos lados.

Figura 45 Modulo "Dispositivo final" visto desde ambos lados.

Se procede a la configuración de los parámetros mencionados anteriormente.

Una vez realizado los cambios, damos clic en la opción de "Write" para que se guarden los cambios en el módulo, [Figura 46].

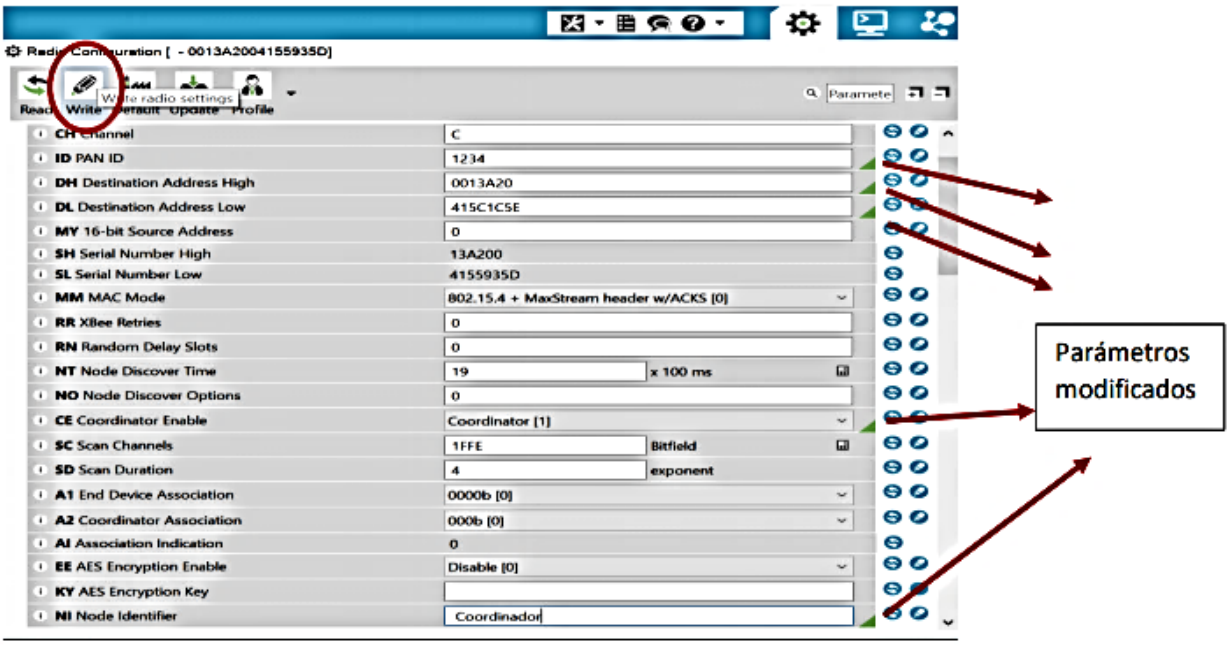

Figura 46 Parámetros modificados.

Nota: Cabe mencionar que los parámetros que se modifican estarán marcados con una "banderas de color verde". Cuando se da clic en la opción de "write", aparece una ventana la cual nos indica que se están escribiendo en el módulo los parámetros cambiados, [Figura 47].

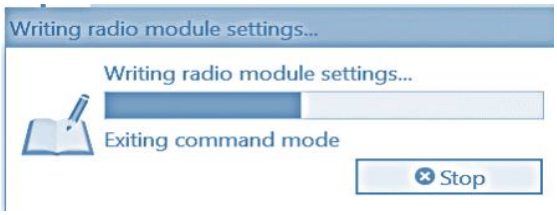

Figura 47 Ventana de escritura de parámetros.

Cuando termina el proceso de escritura en el módulo, se puede apreciar el cambio en el recuadro del módulo agregado.

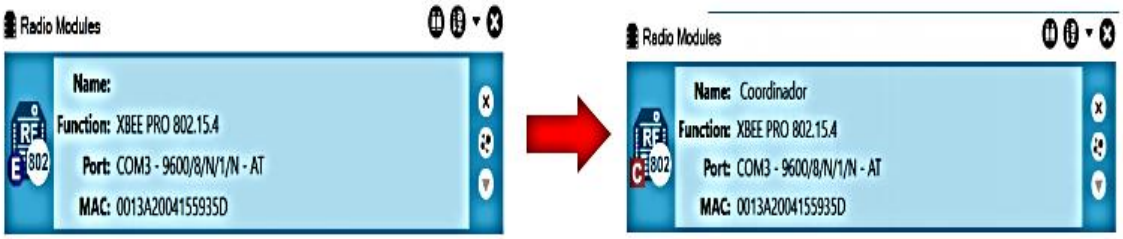

Figura 48 Módulo coordinador antes y después de ser configurado.

De esta manera el primer módulo queda configurado como "Coordinador", [Figura 48]. Para retirarlo solo basta con dar clic en el símbolo de "x" que está en la parte superior del recuadro del módulo agregado.

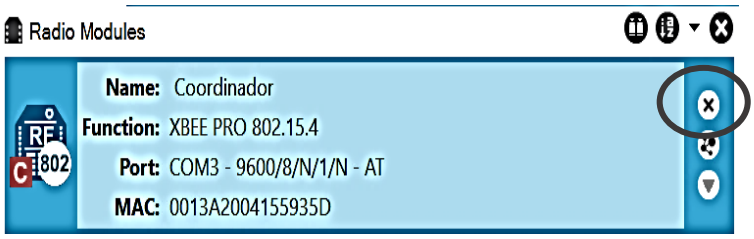

Figura 49 Desconexión de módulo.

Cuando se da clic en la opción de la "x", se puede desconectar el módulo XBee son problemas de la PC, [Figura 49]. Ahora para el módulo que será configurado como dispositivo final (End Device), son los mismos pasos que se explicaron anteriormente, solo cambiara la dirección de destino baja (Destination Address Low), es decir, para la configuración del "End Device" se introducirá la dirección de destino baja del "Coordinador" previamente configurado.

A continuación, se muestran los parámetros ya configurados del módulo [Figura 50] que se usara como dispositivo final (End Device).

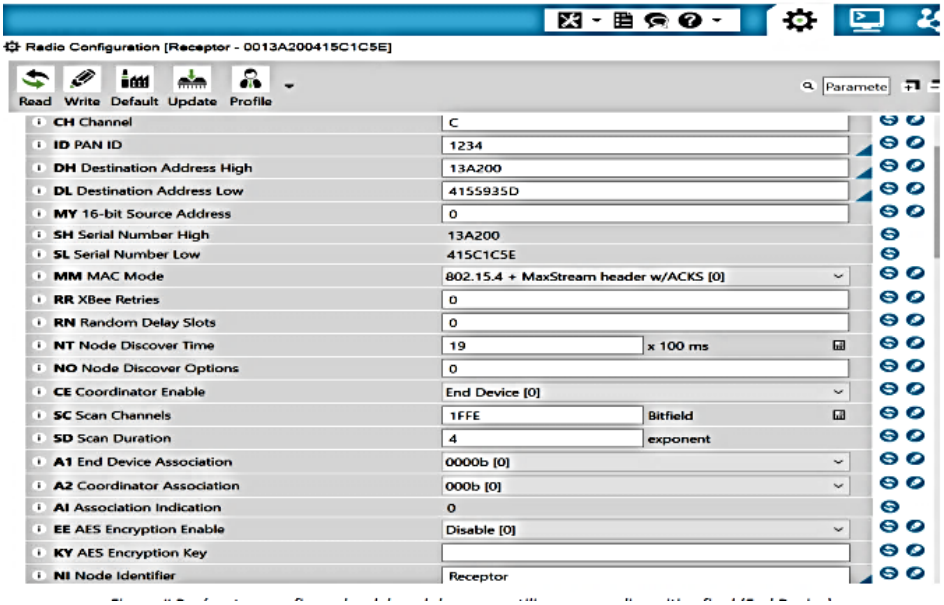

Figura 50 Parámetros configurados del módulo que se utilizará como dispositivo final (End Device).

En las siguientes imágenes se muestra el estado del módulo agregado antes y después de ser configurado, [Figura 51].

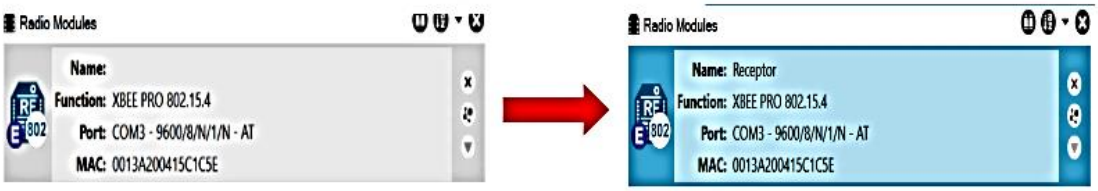

Figura 51 Modulo "Dispositivo final" antes y después de ser configurado.

De esta manera se configuran los módulos XBee para poder realizar la comunicación entre dos dispositivos con ayuda del microcontrolador Atmega48A, cabe mencionar que del módulo XBee solo conectamos cuatro de sus terminales para poder realizar la comunicación [Figura 52], las terminales son las siguientes:

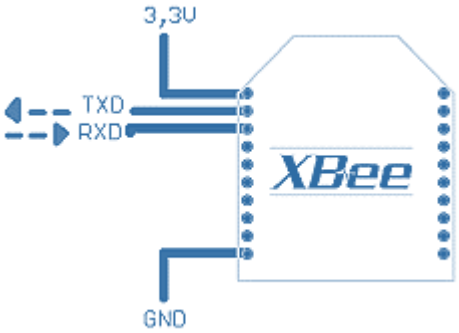

Figura 52 Terminales para la conexión con el microcontrolador, tomado de (21).

Posteriormente se realizará la conexión entre microcontrolador y módulos para lograr la comunicación de radio frecuencia. El programador utilizado para la configuración de los módulos XBee por medio del software X-CTU se puede apreciar en la siguiente imagen [Figura 53].

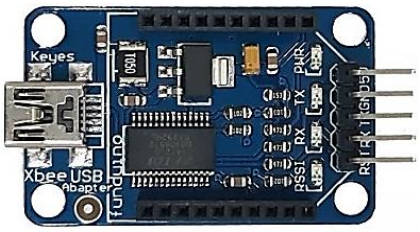

Figura 53 Programador utilizado para la configuración de módulos XBee, tomado de página del fabricante.

Posteriormente a la configuración anterior, nos vimos en la necesidad de agregar un módulo más ya que la red paso de ser una red punto a punto a una red multipunto.

Siguiendo los pasos de las figuras 32 a 51 que se encuentran anteriormente, una vez realizados estos pasos procedemos realizar la configuración para que los tres módulos XBee se puedan comunicar entre ellos sin llegar a tener coaliciones de datos y/o conexiones erróneas [Figura 54].

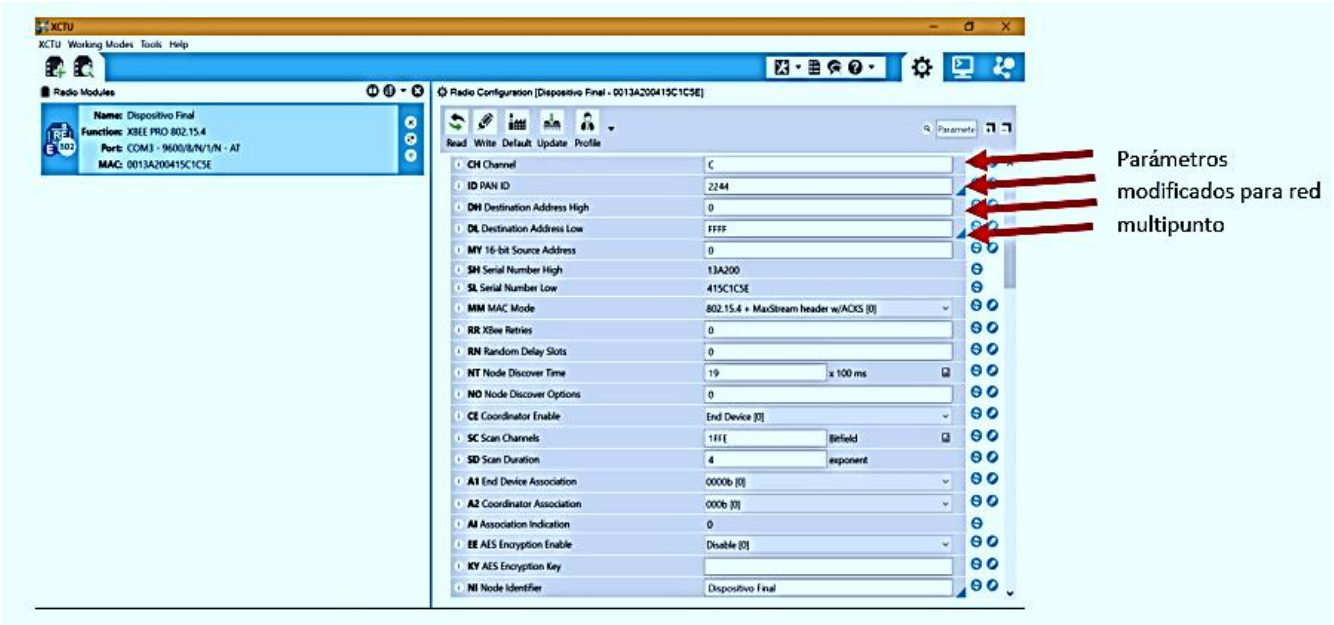

Figura 54 Configuración de parámetros para red multipunto.

En esta configuración y como se puede apreciar en la figura anterior, las características que se modificaron en los módulos son las marcadas con las flechas; Contraseña de

Trabajo de Área Personal (PAN), Dirección de Destino Alta (DH), Dirección de Destino Baja [Figura 55, Figura 56].

Nota: La configuración de los parámetros para una red multipunto de tres o más XBee S1 es la misma.

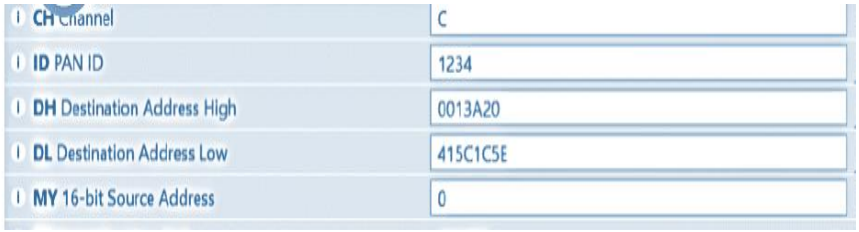

Figura 55 Configuración punto a punto.

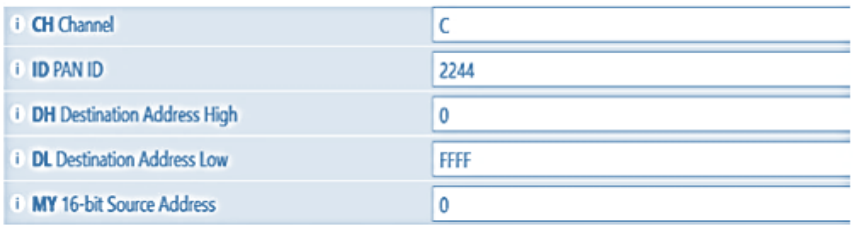

Figura 56 Configuración de red multipunto.

3.4 Desarrollo de los códigos de programación

Cisterna.

Comenzamos el código de programación incluyendo las librerías. En este caso la principal y la librería para las interrupciones. Definimos la frecuencia de trabajo del microcontrolador y definimos dos variables tipo char para poder utilizarlas en la trasmisión y recepción de los datos.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000UL
unsigned char dato3 = 'c';
unsigned char dato6 = 'f';
```

Esta es la función de trasmisión de los datos que deseamos enviar, en la segunda línea tenemos un ciclo while, lo cual nos indica que mientras este vacío el buffer de transmisión será posible enviar el dato.

```
void USARTTx(unsigned char data){      //funcion de transmision
       while (!(UCSR0A)&&(1<<UDRE0)); //
       UDR0 = data;
}
```

En esta parte del código son las interrupciones INT 0 e INT 1. Cuando se llegue a generar la interrupción INT 0, está nos va a poner en un nivel alto a PC0 el cual nos sirve para el buen funcionamiento del programa en la parte de la interrupción por recepción, también se transmite el dato 6.

```
ISR(INT0_vect){
        PORTC = 0x01;
        USARTTx(dato6);
}

ISR(INT1_vect){
        PORTC = 0x00;
        PORTB = 0x00;
        USARTTx(dato3);
}
```

Por lo tanto, cuando se llegue a generar la INT 1, nos pondrá todo el puerto C y todo el puerto B en un nivel lógico bajo, esta parte también es importante en la parte de la interrupción por recepción y a su vez nos transmite el dato 3.

En esta parte de la interrupción por recepción, definimos la variable dato tipo char y tenemos varias condicionales if / else if, para que, dependiendo del dato que recibamos en este caso "a" o "b", el programa realice las funciones que están dentro de su condicional. Además, agregamos un segundo condicional después de recibir el dato para el correcto funcionamiento de la bomba de agua. Con el condicional else if logramos que, en dado caso que el tinaco requiera agua enviándonos el dato "a" y la cisterna no cuente con la suficiente agua para llenarlo, por ningún motivo se prenda la bomba de agua ya que se corre el riesgo de que la bomba se queme si no tiene agua.

Por su parte, cuando se recibe el dato "b", se pone en un nivel lógico bajo todo el puerto B y esto físicamente nos representa el apagado de la bomba de agua.

```
ISR (USART_RX_vect){
        unsigned char dato;
        dato = UDR0;
        if (dato == 'a'){
                if (PORTC == 0x01){
                        PORTB = 0x01;
                }
                else if(PORTC == 0x00) {
                        PORTB = 0x00;
                }
        }
         if (dato == 'b'){
                PORTB = 0x00;
        }
}
```

Esta es la función de configuración del USART para el microcontrolador. UBRR0H = 0 y UBRR0L = 12 es para una velocidad de 9600 bps.

En el registro UCSR0A habilitamos el modo asíncrono a doble velocidad (U2X), en el registro UCSR0B habilitamos receptor y transmisor por interrupción por recepción (RXCIE, RXEN y TXEN). Y, por último, en el registro UCSR0BC habilitamos 1 bit de paro, sin paridad y datos de 8 bits.

```
void conf_USART(){
        UBRR0H = 0;
        UBRR0L = 12;
        UCSR0A = 0x02;
        UCSR0B = 0x98;
        UCSR0C = 0x86;
}
```

Es la función principal del programa, configuramos los puertos C, B y D como salidas, con el registro EIMSK habilitamos la INT 0 e INT 1 y con el registro EICRA habilitamos el flanco de subida.

```
int main(void)
{
        DDRC = 0xFF;
        DDRB = 0xFF;
        DDRD = 0xFF;
        EIMSK = 0x03;        //habiltamos la int 0, int 1
        EICRA = 0x05;        //hanilitacion del flanco de subida
        conf_USART();
        sei ();
        while (1){
        asm ("nop");
                }
}
```

Tinaco.

En este programa incluimos las librerías, definimos la frecuencia de trabajo del microcontrolador y definimos 4 variables tipo char para la función de transmisión y recepción.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000UL
unsigned char dato1 = 'a';
unsigned char dato2 = 'b';
unsigned char dato5 = 'e';
unsigned char dato4 = 'd';
```

Esta parte del código es la función de configuración del USART, se define la velocidad de transmisión, habilitamos la transmisión y recepción, se define el tamaño de los datos que es de 8 bits, un bit de paro y sin paridad.

```
void conf_usart(){                  //configurar usart
      UBRR0H = 0;
      UBRR0L = 12;                  //definir 9600 baud
      UCSR0A = 0x02;                //dobla la velocidad de transmision
      UCSR0B = 0x98;                //definimos interrupcion por recepcion,
habilitamos transmision y recepcion
      UCSR0C = 0x86;                //definimos el tamaño de los datos 8 bits
}
```

Esta es la función de trasmisión y como ya se había comentado, el ciclo while, nos indica que mientras este vacío el buffer de transmisión será posible enviar el dato. Esta función de transmisión se coloca en los tres códigos realizados para el desarrollo del proyecto.

```
void USARTTx(unsigned char data){ //funcion de transmision
      while (!(UCSR0A)&&(1<<UDRE0));
      UDR0 = data;
}
```

En esta parte tenemos las interrupciones INT 0 e INT 1. Al momento que se genere la interrupción INT 0 esta enviará el dato 1, por lo tanto, cuando se genere la interrupción INT 1 esta enviará el dato 2. Estos datos enviados los recibirá el microcontrolador con el programa llamado cisterna dado que estos datos son "a" y "b".

```
ISR(INT0_vect){                     //funcion de interrupcion 1
      USARTTx(dato1);               //transmite el valor de dato 1
}

ISR (INT1_vect){                    //funcion de interrupcion 2
      USARTTx(dato2);
}
```

En la parte de la función de interrupción por recepción, definimos nuevamente la variable dato tipo char y esa variable al momento que se genera la interrupción se guarda en el buffer de recepción (UDR0), también, tenemos los condicionales en los cuales, si se cumple la condición, enviarán el dato que se encuentra dentro de ellos y estos datos serán enviados al microcontrolador con el programa del indicador.

```
ISR (USART_RX_vect){
        unsigned char dato;
        dato = UDR0;
        if (dato == 'c'){
                USARTTx(dato4);
        }
        if (dato == 'f'){
                USARTTx(dato5);
        }
}
```

Por último, tenemos la función principal en donde mandamos a llamar la función de configuración del USART, configuramos el puerto D como entrada y la activación de los resistores de pull up.

También habilitamos las interrupciones INT 0 e INT 1 y a su vez, habilitamos el flanco de subida.

```
int main(void){
        conf_usart();
        DDRD = 0x00;          //puerto D es entrada
        PORTD = 0xFF;         //activacion de resistores de pull up
        EIMSK = 0x03;         //habiltamos la int 0, int 1
        EICRA = 0x05;         //hanilitacion del flanco de subida
        sei();
        while (1){
                asm("nop");
        }
}
```

Indicador.

```
#include <avr/io.h>              //librería de entrada salida.
#include <util/delay.h>          //librería de retardos.
#include <avr/interrupt.h>       //libreria de interrupciones
#define F_CPU 1000000UL
void conf_USART();
//*****************************************************//Configuración de LCD
#define Enable_On PORTC|=_BV(PC2) //habilita el encendido en el Puerto C.
#define Enable_Off PORTC&=~_BV(PC2)//habilita el apagado en el puerto C.
#define RS_On PORTC|=_BV(PC0)     //el encendido de la señal de control RS.
#define RS_Off PORTC&=~_BV(PC0)   //el apagado de la señal de control RS.
#define RW_On PORTC|=_BV(PC1)     //el encendido de la señal de control RW.
#define RW_Off PORTC&=~_BV(PC1)   //el apagado de la señal de control RW.
#define Data PORTB                //define Data para el puerto B
#define DelayL _delay_ms(5);      //establece DelayL como un retardo de 5
milisegundos.

int i=0;                          //declara la variable entera i.
```

En este código incluimos una nueva librería la cual, es la librería de los delay o retardos del microcontrolador, también, habilitamos el encendido y apagado del puerto C, de la

señal de control RS y de la señal de control RW, declaramos una variable de tipo entero "i".

```
void PORT_init (void){              //función para inicializar puertos.
      DDRC=0x07;                    //salidas de los bits 0,1 y 2 del puerto C.
      DDRB=0xFF;                    //salidas de todas las terminales del puerto
B.
      PORTC=0x00;                   //valor de las terminales del puerto C.
      PORTB=0x00;                   //valor de las terminales del puerto B
      DDRD = 0xCC;
}                                   //fin de la función
```

Esta es la función para inicializar los puertos, aquí, también configuramos los registros de los puertos.

```
  void LCD_init (void){             //función para inicializar el LCD.
        Data=0x0F;                  //iguala el valor de Data con 0x0F.
        Enable_On;                  //llama a Enable_On.
        DelayL;                     //llama a DelayL.
        Enable_Off;                 //llama a Enable_Off.
        Data=0x00;                  //iguala a Data con 0x00.
        RS_On;                      //llama a RS_ON.
        DelayL;                     // llama a DelayL.
  }                                 //fin de la función
```

Esta es la función para inicializar la LCD, cabe resaltar que se tiene que seguir una serie de instrucciones para poder inicializarla y después de eso un puntero debe de aparecer en la misma y esto indica que está lista para recibir los caracteres.

```
void WriteLCD(char text[15]){       //función para escribir en el LCD.
      RS_On;                        //llama a RS_On.
      for (i=0; i<16; i++){         //inicio del for la pantalla es de 2x16.
            Data=text[i];           //se iguala Data a vector de longitud 15.
            Enable_On;              //llama a Enable_On.
            DelayL;                 //llama a DelayL.
            Enable_Off;             //llama a Enable_Off.
            DelayL;                 //llama a DelayL.
      }                             //fin de la función
      i=0;                          //variable entera igual a 0.
      Data=0x00;                    //Data se igual a 0x00.
      RS_Off;                       //llama a RS_Off.
}                                   //fin de la función.
```

Aquí tenemos la función para poder escribir en la LCD, la primera línea es para llamar el RS_On y después le agregamos un ciclo for, esto se agrega porque se escribe carácter por carácter, esto es logrado mediante una entrada de tipo char pero en este caso es un vector, es decir, es una serie de caracteres, por lo tanto, cuando se llame a la función y a la entrada

se podrá el texto y este se guardará en un vector de caracteres. Al final del ciclo for, la variable entera "i" es igual a cero, data es igual a 0x00 y se llama a RS_Off.

```
void ClearLCD(void){              //función para limpiar la pantalla del LCD.
      Data=0x01;                  //iguala a Data con 0x01.
      Enable_On;                  //llama a Enable_On.
      DelayL;                     //llama a DelayL.
      Enable_Off;                 //llama a Enable_Off.
}                                 //fin de la función.
```

Esta función es para limpiar el texto de la LCD, se borran todos los caracteres escritos en ella.

```
ISR (USART_RX_vect){
      unsigned char dato;
      dato = UDR0;
 if (PORTC == 0x08){
      if (dato == 'a'){
            PORTD = 0x40;
            WriteLCD("Llenando  Tanque");
          ClearLCD();
            WriteLCD("Llenando  Tanque");
      }
      if (dato == 'b'){

            PORTD = 0x80;
            WriteLCD("  Tanque Lleno  ");
            ClearLCD();
            WriteLCD("  Tanque Lleno  ");
       }
       }

      if (dato == 'd'){
            PORTD = 0x04;
            PORTC = 0x00;
            WriteLCD("Cisterna   Vacia");
            ClearLCD();
            WriteLCD("Cisterna   Vacia");
      }
      }
```

En la interrupción por recepción definimos nuevamente la variable dato tipo char y esa variable se va a guardar en el buffer de recepción. Pusimos el condicional if para que, dependiendo del dato recibido, se realizará lo que nosotros necesitamos físicamente, por ejemplo, si recibimos el dato "a", físicamente el indicador nos mostrará con un LED de color ámbar o anaranjado y en la pantalla que se está llenado el tinaco, por su parte, si recibimos el dato "b", el indicador nos mostrará en la pantalla y con un LED verde que el tinaco está lleno y por último, si recibimos el dato "d", el indicador nos mostrará en la pantalla y con un LED rojo que la cisterna está vacía y que hay que revisar la cisterna. En el dato "d" tenemos que al momento que nos llegue, el puerto C se pondrá en un nivel bajo. Al principio de la interrupción por recepción tenemos un condicional que nos ayudará a que funcione bien la escritura y el encendido correcto de los LEDs.

```
void conf_USART(){          //Configurar USART
      UBRR0H = 0;
      UBRR0L = 12;          //Definir 9600 baudios
      UCSR0A = 0X02;        //Dobla la velocidad de transmision
      UCSR0B = 0x98;        //definimos interrupcion por recepcion, habilitamos
transmision y recepcion
      UCSR0C = 0x86;        //definimos el tamaño de los datos 8 bits
}
```

Aquí tenemos la función de configuración del USART y habilitamos las mismas cosas que los códigos anteriores.

```
int main (void){            //inicio del programa principal.
      PORT_init();          //llama a la función PORT_init().
      LCD_init();           //llama a la función LCD_init().
      conf_USART();
      sei ();
      while(1){                     //inicia ciclo infinito.
            unsigned char dato;
          dato = UDR0;

            if(dato=='e')
            {
                  PORTC = 0x08;
            }
      }
}
```

Esta es la última parte del código, mandamos a llamar las funciones del USART y del LCD en la función principal, y dentro del ciclo while, definimos la variable dato como tipo char y la guardamos en el buffer de recepción, además, tenemos un condicional que nos ayudará a que el puerto C se iguale a 0x08 y con esto nuestra interrupción por recepción funcione adecuadamente.

3.5 Análisis de resultados

A continuación, se muestra el funcionamiento físico del sistema realizado de la comunicación inalámbrica entre una cisterna, un tinaco y un indicador LCD por medio de los microcontroladores Atmega48A y los módulos XBee bajo el estándar IEEE 802.15.4, usando los programas descritos en los anexos para el coordinador y los dispositivos finales.

En este apartado se muestra el proceso que lleva nuestro sistema en cada sitio donde se encuentra el módulo y su respectivo microcontrolador, se desglosa paso a paso el recorrido que logra el sistema de dispositivo en dispositivo.

Circuito "Tinaco"

En la figura se muestra el dispositivo coordinador, que se encarga de enviar el estado que se encuentra el tinaco, se visualizan dos interruptores de nivel que enviaran señales al microcontrolador y este a su vez a la red tipo estrella creada [Figura 57].

Figura 57 Implementación de interruptores de nivel de agua a un recipiente "Tinaco".

En la siguiente imagen se muestra la placa electrónica del tinaco [Figura 58], el circuito es alimentado por una pila cuadrada de 9 volts. Una vez alimentado el circuito procederá a verificar el estado de los interruptores de nivel de agua, dependiendo del estado en el que se encuentre enviará un mensaje a los demás dispositivos para que interactúen con este.

Si el agua ingresada al tinaco sobresale del interruptor de nivel de agua 1, tomará como valor un "Tinaco lleno" por lo contrario si el agua desactiva el interruptor de nivel de agua 1 y llega al interruptor de nivel de agua 2, tomará como valor un "Tinaco vacío".

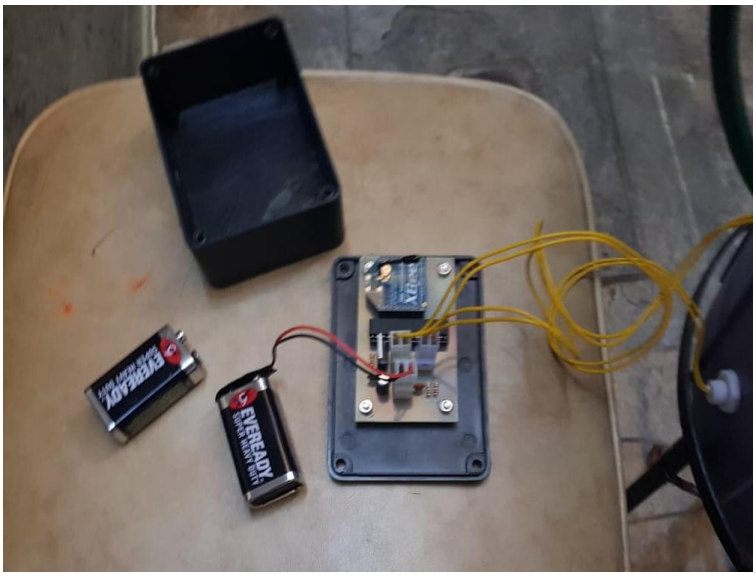

Figura 58 Montaje de placa electrónica "Tinaco".

Circuito "Cisterna"

En la siguiente imagen se muestra uno de los dispositivos finales, tomaremos como referencia que este circuito es encontrado en la cisterna, se muestran al igual que en el circuito anterior dos interruptores de nivel de agua [Figura 59].

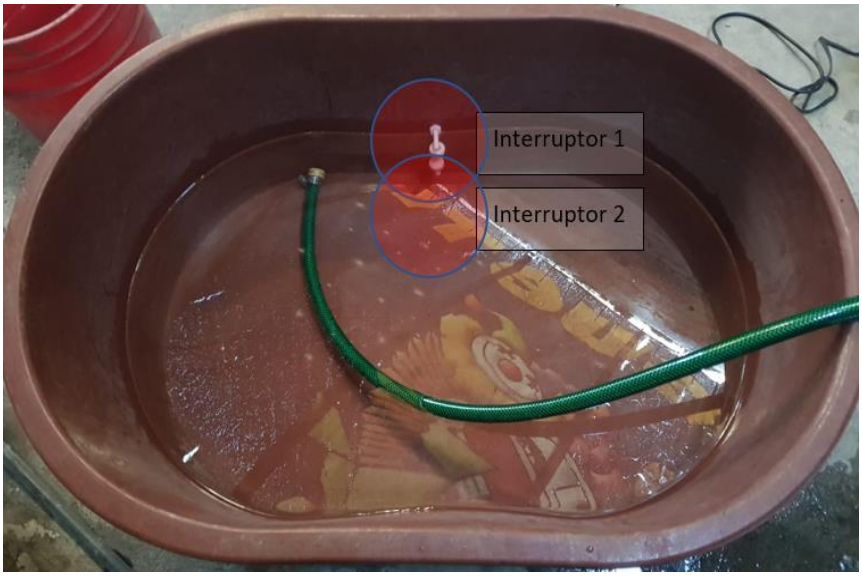

Figura 59 Implementación de interruptores de nivel de agua a un recipiente "Cisterna".

Los interruptores son conectados a un microcontrolador con su respectivo módulo XBee [Figura 60], que además, será conectado a un circuito externo que accionará una bomba, este circuito cuenta con un arreglo de un Triac y un Optoacoplador [Figura 61] que a su vez tendrá una red snubber en la parte donde se conecta la bomba de agua y realizará la tarea de no permitir que los transitorios de corriente producidos por la activación de la bomba sobrepasen y dañen este circuito, en la siguiente imagen se muestra estos dos circuitos.

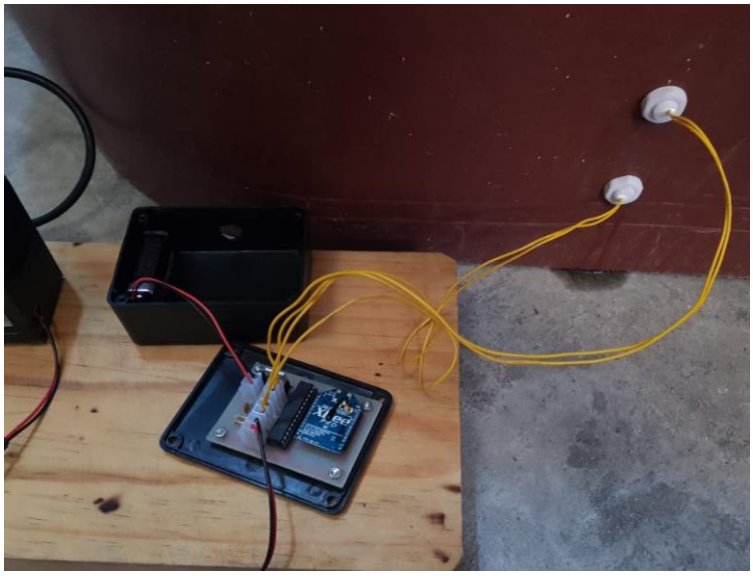

Figura 60 Montaje de placa electrónica "Cisterna".

Figura 61 Montaje de placa electrónica Circuito Triac/MOC

Al igual que en el circuito "Tinaco" es alimentado por medio de una pila de 9 volts que al alimentar al circuito conformado por un microcontrolador y el módulo XBee ejecutará sus tareas respectivas.

Si el nivel de agua ingresada a la "Cisterna "sobresale del interruptor de nivel de agua 1, tomara como dato "Cisterna llena", que servirá como llave para poder accionar la bomba de agua si el circuito del tinaco ha llegado al interruptor de nivel de agua 2 y es necesario llenar nuevamente el tinaco, por el contrario, si el agua ingresada a la "Cisterna "desactiva el interruptor de nivel de agua 1 y llega al interruptor de nivel de agua 2, tomará como valor un "Cisterna vacía ", este dato apagará la bomba y no dejará que continúe llenando hasta que el nivel de agua sobresalga el interruptor de nivel de agua 1 de la cisterna.

Circuito "Indicador"

En la siguiente figura se muestra el dispositivo final que ayudará a visualizar el estado de los dispositivos que están interactuando, por medio de un display LCD se visualizará al usuario las acciones que está realizando el sistema [Figura 62].

Este dispositivo cuenta con un microcontrolador conectado a su respectivo módulo XBee y su indicador LCD, además, de 3 indicadores LED (Verde, Naranja y Rojo) que servirán como medio visual para el usuario.

Figura 62 Montaje circuito "indicador".

SIGNIFICADOS DEL INDICADOR

El indicador LED de color verde, acompañado del display LCD [Figura 63] desplegará un texto "Tanque Lleno" cuando el interruptor de nivel de agua 1 del tinaco sobresalga o tenga un nivel adecuado de agua sin llegar al interruptor de nivel de agua 2, además de que es similar en la cisterna (el nivel de agua sobresale del interruptor de nivel de agua 1 o tiene un nivel adecuado sin llegar al interruptor de nivel de agua 2).

Figura 63 Indicador "Tanque lleno".

El segundo indicador LED color naranja, precedido acompañado de un texto en el display LCD "Llenando tanque" [Figura 64], mostrará que el nivel de agua en el tinaco ha bajado al interruptor de nivel de agua 2 y al tener este estado, ha enviado una señal que ha llegado al dispositivo de la cisterna que verificará si el contenedor tiene un nivel de agua adecuado (el nivel de agua sobresale del interruptor de nivel de agua 1 o tiene nivel adecuado mayor al interruptor de nivel de agua 2) y en caso que la cisterna tenga el nivel adecuado de agua se accionará la bomba de agua para satisfacer nuevamente el nivel que necesita el tinaco para volver a desplegar un "Tinaco lleno" y encender el primer indicador LED de color verde.

Figura 64 Indicador "Llenando tanque".

El tercer y último indicador LED (Color rojo), acompañado de un texto "Cisterna vacía" en el display LCD [Figura 65], indicaran que no es posible realizar la ejecución del sistema debido a la falta de agua en la cisterna (el interruptor de nivel de agua 2 es igual o menor) por lo que, hasta que el nivel de agua sea adecuado en la cisterna podrá ejecutar de nuevo el proceso antes mencionado.

Figura 65 Indicador "Cisterna vacía".

Conclusiones

El trabajo terminal que se ha presentado mejoró nuestra la habilidad para darle una solución a una problemática, en el que, existen tareas que son difíciles de realizar para personas con alguna discapacidad, mayores de edad o personas que adquieren tecnología en este tipo de sistemas, el uso de programación de los microcontroladores en su totalidad funcionó para poder ejercer la comunicación entre los dispositivos conectados y que, al utilizar tecnología ajena a nuestro aprendizaje como son los módulos XBee, motiva a conocer y utilizar una de las herramientas de comunicación de las cuales podemos hacer posible una buena implementación en cualquier lugar donde sea posible la interconexión de dispositivos.

Los resultados obtenidos al hacer actuar el proyecto mostraron características que fueron comparadas teóricamente y vistas prácticamente al medir en ciertas partes de los circuitos electrónicos. Dentro de las cuales cabe destacar las señales presentadas en los módulos y los interruptores de nivel de agua.

El alcance de las antenas al realizar las mediciones correspondientes estuvo en el rango establecido por el fabricante, salvo las posibles modificaciones que se debieron de realizar en el gabinete que contenía el circuito indicador, que, al ser de metal, impedía que la señal de radiofrecuencia del módulo XBee se propagara, Se modificó una de las paredes de dicho gabinete por acrílico para así, propagar la onda radiada y poder comunicar con los otros dispositivos conectados.

Recomendaciones

En el proyecto que se ha presentado se proponen los siguientes puntos que consideramos como una recomendación o mejora a futuro.

En los circuitos que se encuentran a la intemperie (cisterna y tinaco) utilizan baterías cuadradas de 9V y al estar siempre conectadas alimentando a su respectivo microcontrolador y módulo XBee se desgastan, proponemos como mejora, implementar celdas solares capaces de guardar energía eléctrica en una pila y como fin alimentar al circuito en el que se encuentre.

Además, proponemos implementar más dispositivos a la red de comunicación que se propuso a tal grado de obtener una casa inteligente capaz de satisfacer tareas simples al usuario, como, por ejemplo: regar un césped, abrir o cerrar cortinas, cambiar la intensidad de la luz en un cuarto habitación, entre otros.

Anexos

Sistema terminado con sus respectivos gabinetes

Figura 66 Instalación de gabinete "Tinaco".

Figura 67 Instalación de gabinetes utilizados en "Cisterna".

Figura 68 Gabinete "Indicador".

CÓDIGO FUENTE "CISTERNA"

```
/*
 * Programa de tesis receptor.c
 *
 * Created: 23/08/2018 07:46:39 p. m.
 * Author: Armando Tenorio, Eduardo Sanchez y Miguel Juarez
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000UL
unsigned char dato3 = 'c';
unsigned char dato6 = 'f';

void USARTTx(unsigned char data){     //funcion de transmision
       while (!(UCSR0A)&&(1<<UDRE0)); //
       UDR0 = data;
}

ISR(INT0_vect){
       PORTC = 0x01;
       USARTTx(dato6);
}

ISR(INT1_vect){
       PORTC = 0x00;
       PORTB = 0x00;
       USARTTx(dato3);
}

ISR (USART_RX_vect){
       unsigned char dato;
       dato = UDR0;
       if (dato == 'a'){
              if (PORTC == 0x01){
                     PORTB = 0x01;
              }
              else if(PORTC == 0x00) {
                     PORTB = 0x00;
              }
       }
        if (dato == 'b'){
              PORTB = 0x00;
       }
}
void conf_USART(){
       UBRR0H = 0;
       UBRR0L = 12;
       UCSR0A = 0x02;
       UCSR0B = 0x98;
       UCSR0C = 0x86;
}
int main(void)
{
       DDRC = 0xFF;
       DDRB = 0xFF;
       DDRD = 0xFF;
       EIMSK = 0x03;         //habilitamos la int 0, int 1
       EICRA = 0x05;         //habilitación del flanco de subida
       conf_USART();
       sei ();
       while (1){
       asm ("nop");
              }
```

CÓDIGO FUENTE "TINACO"

```
/*
 * Programa de Tesis 1.c
 *
 * Created: 20/08/2018 07:29:05 p. m.
 * Author : Armando Tenorio, Eduardo Sanchez y Miguel Juarez
 */
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 1000000UL
unsigned char dato1 = 'a';          //el valor de dato 1 es 0x01
unsigned char dato2 = 'b';          //el valor de dato 2 es 0x02
unsigned char dato5 = 'e';
unsigned char dato4 = 'd';
void conf_usart(){                  //configurar usart
       UBRR0H = 0;
       UBRR0L = 12;                 //definir 9600 baud
       UCSR0A = 0x02;       //dobla la velocidad de transmision
       UCSR0B = 0x98;       //definimos interrupción por recepción, habilitamos
transmisión y recepción
       UCSR0C = 0x86;       //definimos el tamaño de los datos 8 bits
}

void USARTTx(unsigned char data){   //funcion de transmision
       while (!(UCSR0A)&&(1<<UDRE0));
       UDR0 = data;
}

ISR(INT0_vect){                     //función de interrupción 1
       USARTTx(dato1);              //transmite el valor de dato 1
}

ISR (INT1_vect){                    //función de interrupción 2
       USARTTx(dato2);
}

ISR (USART_RX_vect){
       unsigned char dato;
       dato = UDR0;
       if (dato == 'c'){
              USARTTx(dato4);
       }
       if (dato == 'f'){
              USARTTx(dato5);
       }
}
int main(void){
       conf_usart();
       DDRD = 0x00;          //puerto D es entrada
       PORTD = 0xFF;         //activación de resistores de pull up
       EIMSK = 0x03;         //habilitamos la int 0, int 1
       EICRA = 0x05;         //habilitación del flanco de subida
       sei();
       while (1){
              asm("nop");    // nop - consume un ciclo maquina,
       }
}
```

CÓDIGO FUENTE "INDICADOR LCD"

```
/*
 * Indicador bueno.c
 *
 * Created: 08/10/2018 08:29:58 p. m.
 * Author : arman
 */
#include <avr/io.h>                 //librería de entrada salida.
#include <util/delay.h>             //librería de retardos.
#include <avr/interrupt.h>          //libreria de interrupciones
#define F_CPU 1000000UL
void conf_USART();
//*****************************************************//Configuración de LCD
#define Enable_On PORTC|=_BV(PC2)   //habilita el encendido en el Puerto C.
#define Enable_Off PORTC&=~_BV(PC2)//habilita el apagado en el puerto C.
#define RS_On PORTC|=_BV(PC0)       //el encendido de la señal de control RS.
#define RS_Off PORTC&=~_BV(PC0)     //el apagado de la señal de control RS.
#define RW_On PORTC|=_BV(PC1)       //el encendido de la señal de control RW.
#define RW_Off PORTC&=~_BV(PC1)     //el apagado de la señal de control RW.
#define Data PORTB                  //define Data para el puerto B
#define DelayL _delay_ms(5); //establece DelayL como un retardo de 5 milisegundos.

int i=0;                            //declara la variable entera i.

void PORT_init (void){              //función para inicializar puertos.
       DDRC=0x07;                   //salidas de los bits 0,1 y 2 del puerto C.
       DDRB=0xFF;                   //salidas de todas las terminales del puerto B.
       PORTC=0x00;                  //valor de las terminales del puerto C.
       PORTB=0x00;                  //valor de las terminales del puerto B
       DDRD = 0xCC;
}                                   //fin de la función

void LCD_init (void){       //función para inicializar el LCD.
       Data=0x0F;                   //iguala el valor de Data con 0x0F.
       Enable_On;                   //llama a Enable_On.
       DelayL;              //llama a DelayL.
       Enable_Off;                  //llama a Enable_Off.
       Data=0x00;                   //iguala a Data con 0x00.
       RS_On;                       //llama a RS_ON.
       DelayL;              // llama a DelayL.
}                                   //fin de la función

void WriteLCD(char text[15]){       //función para escribir en el LCD.
       RS_On;                       //llama a RS_On.
       for (i=0; i<16; i++){ //inicio del for la pantalla es de 2x16.
              Data=text[i];         //se iguala Data a vector de longitud 15.
              Enable_On;            //llama a Enable_On.
              DelayL;       //llama a DelayL.
              Enable_Off;           //llama a Enable_Off.
              DelayL;       //llama a DelayL.
       }                            //fin de la función
       i=0;                         //variable entera igual a 0.
       Data=0x00;                   //Data se igual a 0x00.
       RS_Off;              //llama a RS_Off.
}                                   //fin de la función.

void ClearLCD(void){                //función para limpiar la pantalla del LCD.
       Data=0x01;                   //iguala a Data con 0x01.
       Enable_On;                   //llama a Enable_On.
       DelayL;              //llama a DelayL.
       Enable_Off;                  //llama a Enable_Off.
}                                   //fin de la función.
//*****************************************************//Configuracion de LCD
//*****************************************************//Configuracion de USART
```

```
ISR (USART_RX_vect){
       unsigned char dato;
       dato = UDR0;
 if (PORTC == 0x08){
       if (dato == 'a'){
              PORTD = 0x40;
              WriteLCD("Llenando  Tanque");
           ClearLCD();
              WriteLCD("Llenando  Tanque");
       }                                          //Enciende bomba
       if (dato == 'b'){                          //Si hay agua

              PORTD = 0x80;
              WriteLCD("  Tanque Lleno  ");
              ClearLCD();
              WriteLCD("  Tanque Lleno  ");
        }
        }

       if (dato == 'd'){
              PORTD = 0x04;
              PORTC = 0x00;
              WriteLCD("Cisterna   Vacia");
              ClearLCD();
              WriteLCD("Cisterna   Vacia");
       }
       }
void conf_USART(){           //Configurar USART
       UBRR0H = 0;
       UBRR0L = 12;          //Definir 9600 baudios
       UCSR0A = 0X02;//Dobla la velocidad de transmision
       UCSR0B = 0x98;//definimos interrupción por recepción, habilitamos
transmisión y recepción
       UCSR0C = 0x86;//definimos el tamaño de los datos 8 bits
}
int main (void){             //inicio del programa principal.
       PORT_init();          //llama a la función PORT_init().
       LCD_init();           //llama a la función LCD_init().
       conf_USART();
       sei ();
       while(1){                    //inicia ciclo infinito.
              unsigned char dato;
           dato = UDR0;

              if(dato=='e')
              {
                     PORTC = 0x08;
              }
       }
}
```

CÁLCULO PARA LA FUENTE DE ALIMENTACIÓN.

Características: Lineal 5V a 1A

Regulador de voltaje: LM7805

	Temp.	Min.	Typ.	Máx.
Voltaje de salida	25 ºC	4.8v	5v	5.2v
	0ºC - 125ºC	4.75v		5.25v
Corriente de salida	1.5A			

Puente rectificador: BD2

Capacitores de acoplo: $0.1\mu F, 0.33\mu F$

Transformador: 12V 1A

Filtrado

Sabemos que por cada 1A se utiliza un capacitor electrolítico con valor de 1100µF, entonces:

$$c = 1A * 1100\mu F = 1100\mu F \text{ a } 25V$$

Rectificación

$1A = 1N4001$

$Vpsec = 12Vrms\sqrt{2} = 16.970Vp$

$Vsal = Vpsec - 2Vf = 16.970Vp - 2(1.1v) = 14.707Vp$

$$PIV = \left(\frac{Vsal}{2} - Vf\right) + 20\%$$

$$PIV = \left(\frac{14.707}{2} - (1.1)\right) + 20\% = 6.2535 + 1.2507 = 7.5042Vp$$

Glosario

Banda ISM (Federal Communications Comission): Banda de frecuencia de uso industrial, científica y médica. Es una banda de frecuencias de radio y microondas agrupadas alrededor de los 2.4GHz, reservadas y destinadas para equipo industrial, científico y médico que use radiofrecuencia.

WPAN (Red inalámbrica de área personal): Red inalámbrica de corto alcance que abarca un área de decenas de metros, se usa generalmente para conectar dispositivos periféricos, como impresoras, teléfonos móviles, electrodomésticos o asistentes personales digitales a un ordenador sin utilizar cables.

Rx (Receptor): Agente que recibe el mensaje, señal o código emitido por un emisor o transmisor; es el destinatario que recibe la información.

Tx (Transmisor): Agente que capta la variable de proceso y la transmite a distancia a un agente receptor el mensaje, señal o código.

UART (Universal Asynchronous Receiver-Transmitter): Dispositivo que controla los puertos y dispositivos serie.

MAC (Media Access Control): Identificador único asignado por el fabricante a una pieza de hardware de red.

RTS (Request to send): Señal de salida que envía para condicionar que hay una transmisión de datos.

CTS (Clear to send): Señal que indica que el receptor está preparado para recibir datos.

Peer-to-peer: establece una conexión directa entre dispositivos, sin necesidad de un servicio intermedio.

CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance): El acceso múltiple por detección de portadora y prevención de colisiones es un protocolo de control de acceso a redes de bajo nivel que permite que múltiples estaciones utilicen un mismo medio de transmisión.

Topología: Es el arreglo físico o lógico definido como la forma en como está diseñada una red.

Payload (Carga util): Es el conjunto de datos transmitidos (enviados).

CMOS (Complementary metal-oxide-semiconductor): Semiconductor complementario de oxido metálico, familia lógica empleada en la fabricación de circuitos integrados.

RISC (Complex Instruction Set Computer): Tipo de arquitectura que dispone de un repertorio corto de instrucciones. Cada instrucción puede realizar una operación muy simple, por lo que la complejidad de la CPU disminuye, además de logar que las instrucciones tengan la misma longitud.

Bibliografía

[1] Publireportaje. *Domótica Y Seguridad: Hacia Dónde va El Mercado Este 2018 - Libertad Digital*. 2018, https://www.libertaddigital.com/ciencia-tecnologia/tecnologia/2018-01-25/domotica-y-seguridad-hacia-donde-va-el-mercado-este-2018-1276612720/. Accessed 4 de agosto. 2018.

[2] Tecnología fácil. "¿Qué Es Domótica? ¿Para Qué Sirve?." *19 de marzo*, 2017, https://tecnologia-facil.com/que-es/que-es-domotica-para-que-sirve/. Accessed 20 febrero. 2018.

[3] Suyama, Maria. "Protocolos de Comunicaciones." *30 de agosto*, 2004, https://desarrolloweb.com/articulos/1617.php. Accessed 25 febrero. 2018.

[4] Villagómez, Carlos. "Protocolo de Comunicación." *17 de enero*, 2018, https://es.ccm.net/contents/275-protocolo-de-comunicacion. Accessed 20 febrero. 2018.

[5] *Capítulo 3: Estándar IEEE 802.15.4 "Redes ZigBee"* http://www.ptolomeo.unam.mx:8080/jspui/bitstream/132.248.52.100/229/6/A6.pdf . Accessed 6 de abril. 2018.

[6] Archundia, P. *El Estándar IEEE 802.15.4*. http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/archundia_p_fm/capitulo4.pdf. Accessed 15 de abril. 2018.

[7] *Topologías de Red ZigBee - EcuRed*. https://www.ecured.cu/Topologías_de_red_ZigBee. Accessed 20 de abril. 2018.

[8] Software Guru. *Zigbee, Comunicación Para Dispositivos | SG Buzz*. 2007, https://sg.com.mx/content/view/310. Accessed 17 marzo. 2018.

[9] Coronado, Enrique. "Tutorial Xbee Parte 1: ¿Qué Es Un Xbee Y Qué Es Necesario? – Mecatrónica UASLP." *4 de Julio*, 2013, https://mecatronicauaslp.wordpress.com/2013/07/04/xbee-parte-1-que-es-un-xbee-y-que-es-necesario/. Accessed 6 de abril. 2018.

[10] Robert Faludi, "Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing", O'Reilly, 2010, Accessed 4 de agosto. 2018.

[11] INCIBE. "Seguridad En Comunicaciones ZigBee | INCIBE-CERT." *26 de abril*, 2016, https://www.incibe-cert.es/blog/seguridad-comunicaciones-zigbee. Accessed 4 de agosto. 2018.

[12] Aguayo Eduardo Martin, Ingeniería MCI LTDA, Paul. *Guía Del Usuario XBEE Series 1*. 2010, http://www.mcielectronics.cl/website_MCI/static/documents/XBee-Guia_Usuario.pdf, Accessed 6 de abril. 2018.

[13] Curso Atmega48, Microcontroladores- Balderas Eduardo-ESIME ZACATENCO, Accessed 4 de agosto. 2018.

[14] Ojeda, Luis. *¿Qué Es XBee? ~ XBee.cl*. 2015, http://xbee.cl/que-es-xbee/. Accessed 4 de agosto. 2018.

[15] *Interruptor de flotador de ángulo recto sensor de nivel de agua en estado líquido de montaje lateral, bang Good.com*, https://www.banggood.com/es/Side-mounted-Liquid-Water-Level-Sensor-Right-Angle-Float-Switch-p-945298.html?cur_warehouse=CN, Accessed 11 de noviembre, 2018.

[16] Área tecnológica. *Optoacoplador*. http://www.areatecnologia.com/electronica/optoacoplador.html. Accessed 25 de agosto. 2018.

[17] ELECTROALL. *OPTOACOPLADOR TRIAC MOC 3021: ELECTROALL*. 1999, https://che-charls-electroall.webnode.es/optoacoplador-triac-moc-3021/. Accessed 11 de noviembre. 2018.

[18]Op*toacopladores.*http://www.itlalaguna.edu.mx/Academico/Carreras/electronica/opteca/OPTOPDF3_archivos/UNIDAD3TEMA1.PDF. Accessed 11 de noviembre. 2018.

[19] TRIAC BTA16-600B 600V 16A, Electrónica embajadores, https://www.electronicaembajadores.com/es/Productos/Detalle/SMTHBTA16B/semiconductores/tiristores-triacs-diacs/triac-bta16-600b-600v-16a, Accessed 11 de noviembre, 2018

[20] *LCD 16x2 Blanca - JHD-162ASTNGLED | HeTPro*. https://hetpro-store.com/lcd-16x2-luz-de-fondo-blanca/. Accessed 11de noviembre. 2018.

[21] Aguayo, Paul. *Guía Del Usuario XBee Series 1 DOCUMENTO PRELIMINAR Revisión agosto 2008 Desarrollada Por: Andrés Oyarce Revisada Por*. https://www.mcielectronics.cl/website_MCI/static/documents/XBee_Guia_Usuario.pdf. Accessed 4 de agosto. 2018.

[22] OMEGA. *Sensores de Nivel*. 2010, https://es.omega.com/prodinfo/sondas-de-nivel-medicion.html. Accessed 19 de agosto. 2018.

[23] ElectronicsTutorials. *Triac Tutorial and Triac Switching Circuits*. 2018, https://www.electronics-tutorials.ws/power/triac.html. Accessed 11 de noviembre. 2018.

[24] Espinosa, Felipe, *"Los microcontroladores de atmel"*, 2012, mayo, file:///C:/Users/arman/Desktop/Memoria%20Iron%20Man/DPP/Los%20Microcontroladores%20AVR%20de%20ATMEL/02_Indice%20y%20Prologo.pdf, Accessed 17 de noviembre. 2018.