ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"IMPLEMENTACIÓN DEL PROTOTIPO PARA EL CONTROL DE ACCESO DOCENTE MEDIANTE LA INTEROPERABILIDAD DE WSN Y RFID"

TESIS DE GRADO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

Presentado por:

Darío Javier Llanga Herrera

David Antonio Cano Estrella

RIOBAMBA-ECUADOR

-2014-

De corazón agradecemos infinitamente a Dios por bendecirnos y nunca desampararnos en los momentos difíciles.

Nuestra más sincera gratitud a las personas que incondicionalmente tendieron su mano generosa y día a día nos encaminaron e impulsaron para alcanzar esta meta.

A nuestro Director de Tesis Ing. Vinicio Ramos por la orientación, supervisión, y contribuciones con el presente trabajo, de igual manera al Ing. Néiser Ortíz por las sugerencias e interés brindado.

Darío y David

Dedico la culminación de esta Tesis de Grado a mi padre Jorge, a mi madre Mirian y a mis hermanos quienes me supieron dar amor, cariño, sus bendiciones y sus consejos durante todo este tiempo de mi carrera profesional. A mi tío Raúl por siempre estar pendiente y poner un granito de arena para lograr mi meta. A mi amor Mariana por el apoyo que día a día me brinda para poder cumplir mi sueño.

Darío

Dedico este trabajo a mi madre y a mis hermanos que siempre han estado conmigo y me han apoyado.

David

FIRMAS RESPONSABLES Y NOTA

NOMBRE	FIRMA	FECHA
Ing. Gonzalo Nicolay Samaniego **DECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Franklin Moreno **DIRECTOR DE ESCUELA INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Vinicio Ramos **DIRECTOR DE TESIS**	_____	_____
Ing. Néiser Ortíz **MIEMBRO DEL TRIBUNAL**	_____	_____
Director del Centro de Documentación	_____	_____

NOTA DE TESIS ESCRITA:_____

RESPONSABILIDAD DE LOS AUTORES

Nosotros, Darío Javier Llanga Herrera y David Antonio Cano Estrella, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis y el patrimonio intelectual de la misma pertenecen a la Escuela Superior Politécnica de Chimborazo.

_____

Darío Javier Llanga Herrera

_____

David Antonio Cano Estrella

ÍNDICE DE ABREVIATURAS

ACRÓNIMO	DESCRIPCIÓN
ACK	ACUSE DE RECIBO
ADC	CONVERSIÓN ANALÓGICA-DIGITAL
AES	ADVANCED ENCRYPTION STANDARD
ASK	AMPLITUDE SHIFT KEYING
BIESS	BANCO DEL IESS
CTS	CLEAR TO SEND
DARPA	DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
EAN	EUROPE ARTICLE NUMBER
EEPROM	ELECTRICALLY ERASABLE PROGRAMMABLE ROM
ESPOCH	ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FSK	FREQUENCY SHIFT KEYING
GPS	GLOBAL POSITIONING SYSTEM
IDE	INTEGRATED DEVELOPMENT ENVIRONMENT
IEEE	INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS
IP	INTERNET PROTOCOL
JDK	JAVA SE DEVELOPMENT KIT
LCD	LIQUID CRYSTAL DISPLAY
MISO	MASTER INPUT SLAVE OUTPUT

MOSI	MASTER OUTPUT SLAVE INPUT
PC	COMPUTADORA PERSONAL
PMW	PULSE-WIDTH MODULATION
PSK	PHASE SHIFT KEYING
RAM	RANDOM ACCESS MEMORY
ROM	READ ONLY MEMORY
RF	RADIO FRECUENCIA
RFID	RADIO FREQUENCY IDENTIFICATION
RTC	REQUEST TO SEND
SDA	DATA LINE
SQL	STRUCTURED QUERY LANGUAGE
SP	CYCLIC SLEEP PERIOD
SPI	SERIAL PERIPHERAL INTERFACE
TCP	TRANSMISSION CONTROL PROTOCOL
UART	UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER
UPC	UNIVERSAL PRODUCT CODE
WIFI	WIRELESS FIDELITY
WPAN	WIRELESS PERSONAL AREA NETWORK
WWAN	WIRELESS WIDE AREA NETWORK
WSN	WIRELESS SENSOR NETWORK

ÍNDICE GENERAL

2. CAPÍTULO II

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En la actualidad las tecnologías de redes inalámbricas han presentado un acelerado desarrollo, que han logrado una revolución en que las personas y las organizaciones intercambian información y coordinan sus actividades, consiguiendo así una implementación de redes sensoriales inalámbricas (WSN) como también la tecnología de identificación por radio frecuencia (RFID).

Las redes inalámbricas de sensores (WSN) consisten en un gran número de nodos que son ampliamente desplegados en un área predefinida para proporcionar un control de acceso, también se puede monitorear el estado del medio ambiente, incluyendo la presión, la humedad, temperatura y los datos se transfieren a través de nodos a un cierto lugar. La Identificación por Radio Frecuencia (RFID) mediante la lectura de tarjetas pasivas de bajo coste instalados en objetos o llevados por personas, ha sido ampliamente adoptado en la industria de rastreo y seguimiento y puede soportar un posicionamiento preciso dentro de una distancia limitada, también se utiliza para detectar e identificar los objetos etiquetados por señales electromagnéticas.

El objetivo de este proyecto es diseñar, implementar y evaluar un sistema híbrido que combina las tecnologías WSN y RFID para ofrecer un servicio de

control de acceso de asistencia automatizado a los docentes de la academia CISCO-ESPOCH, que en la actualidad se lo viene haciendo de forma manual.

Se utilizó la investigación científica aplicativa ya que por medio de ella se encamina hacia los hechos para obtener información relevante, fidedigna y así poder aplicarlos inmediatamente el sistema propuesto.

Se diseñó e implementó un prototipo empleando un sistema híbrido compuesto por las tecnologías WSN y RFID, para el control de asistencia docente, integradas por módulos de comunicación XBee encargados de la transmisión y recepción de la información, por placas Arduino que sirven para la interoperabilidad de las tecnologías WSN y RFID, por una aplicación diseñada en JAVA para mostrar los registros en tiempo real de la asistencia a los docentes en los laboratorios de la academia CISCO-ESPOCH.

CAPÍTULO I

MARCO REFERENCIAL

1.1. INTRODUCCIÓN

En este capítulo se expondrá la situación actual de las redes de sensores inalámbricas a nivel mundial, y localmente en el país, así como también de la tecnología RFID. Igualmente se detallará su funcionamiento general, sus principales ventajas y sus diversas aplicaciones. Además se propone este trabajo investigativo como solución a la necesidad que tiene la Escuela Superior Politécnica de Chimborazo de implementar un sistema de control de acceso de los docente ya que actualmente no se cuenta con algún mecanismo

automático tecnológico para la localización y el control de acceso a los docentes de cada una de las facultades de la ESPOCH.

Se detalla también los objetivos, métodos y técnicas que nos ayudarán a la investigación del proyecto de tesis.

1.2. ANTECEDENTES

Los sistemas de radio localización permiten conocer datos, ubicaciones, direcciones y velocidades de objetos, al mismo tiempo que permite mantener el control de procesos que se estén realizando en lugares remotos, desde los controladores aéreos hasta el comportamiento de cuerpos celestes que se encuentran próximos al planeta [1].

Las redes de sensores inalámbricas WSN (*Wireless Sensor Networks*) están integradas por unos dispositivos con características de sistemas embebidos. En los últimos años, varios laboratorios de investigación y especialmente multinacionales como Intel Corporation, han apostado fuertemente por esta tecnología [2]. En diversos informes se augura que este tipo de redes conllevará a una revolución tecnológica similar a la que tuvo la aparición de Internet [3]. De hecho, DARPA (*Defense Advanced Research Projects Agency*), institución dependiente del Departamento de Defensa Estadounidense, también se ha involucrado en el desarrollo de este tipo de redes. Ya se habla de redes de vigilancia global del planeta, capaces de registrar los hábitos de la gente, realizar un seguimiento de personas y mercancías concretas, monitorizar el tráfico, etc. Aunque para ello habrá que esperar todavía unos años, sí que han

surgido múltiples iniciativas y proyectos de investigación de enorme interés y aplicabilidad práctica [4].

La red de sensores inalámbricos nos permite registrar y transmitir datos de un dispositivo a otro, y después retransmitir toda la información para almacenarla en una localización central. Estos dispositivos son unidades autónomas que constan de un microprocesador, una fuente de energía, un radio transceptor y un elemento sensor. Las WSN están formadas por un conjunto de pequeños dispositivos denominados nodos, con capacidad limitada de cómputo y comunicación, cuyo tiempo de vida depende de una batería adjunta al dispositivo.

Los sistemas de identificación por radiofrecuencia o RFID (*Radio Frequency Identification*) es una tecnología para la identificación de objetos a distancia sin necesidad de contacto, ni siquiera visual, que consiste en transmitir la identidad de un objeto mediante ondas de radio. Se requiere lo que se conoce como etiqueta o *tag* RFID que consiste en un microchip que va adjunto a una antena de radio y que va a servir para identificar de forma precisa al elemento portador de la etiqueta [5].

Los sensores se comunican mediante la tecnología *ZigBee*. *ZigBee* es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN). Su objetivo son las aplicaciones para redes *Wireless* que requieran comunicaciones seguras y fiables con baja tasa de envío de datos

(20 a 250Kbps) y maximización de la vida útil de sus baterías. *ZigBee* es un sistema ideal para redes domóticas específicamente diseñado para reemplazar la proliferación de sensores/actuadores individuales. *ZigBee* fue creado para cubrir la necesidad del mercado de un sistema de bajo coste, un estándar para redes *Wireless* de pequeños paquetes de información, bajo consumo, seguro y fiable [6].

En el Ecuador se han implementado redes de este tipo para el monitoreo volcánico. El primer trabajo de monitoreo volcánico fue desarrollado en julio del 2004 por un grupo de investigadores conformado por miembros de las Universidades de Harvard, New Hampshire y North Carolina, al volcán Tungurahua ubicado en Ecuador, para lo cual se emplearon sensores acústicos de baja frecuencia (8Hz). Estos sensores estaban provistos de sismómetros y micrófonos para recolectar información sísmica y acústica de la actividad del volcán, información que fue recopilada en una estación maestra para luego ser transmitida a la estación base a unos 10 Km de distancia [7].

Actualmente en la ESPOCH no se cuenta con algún mecanismo automático - tecnológico para la localización y el control de acceso a los docentes tanto a las aulas como en los laboratorios, para lo cual se plantea diseñar e implementar un sistema de localización mediante la interoperabilidad de las tecnologías WSN y RFID. Dicho sistema funcionará como un medio de control para que los docentes cumplan en tiempo y forma con sus horarios de clase, que actualmente registrarán entradas y salidas de los laboratorios de la academia CISCO de forma manual, ya que los sistemas biométricos de registros son

costosos por lo que se dificulta su implementación en cada laboratorio de la academia. Este tipo de control de acceso se ha implementado en entidades financieras como el BIESS (Banco del IESS) en la ciudad de Riobamba.

1.3. JUSTIFICACIÓN DEL PROYECTO DE TESIS

La localización en interiores es una de las áreas más prometedoras en el campo de la computación móvil. Estos sistemas permiten desarrollar innumerables aplicaciones gracias al posicionamiento de objetos o personas en tiempo real. Algunos de los principales servicios están relacionados con el control de accesos, gracias a la identificación de los usuarios, la seguridad en red, basada en la localización física de los usuarios, servicios de emergencia y estadísticas [8].

La problemática de la localización en interiores ha sido objeto de un intenso estudio e investigación durante los años anteriores [9]. Hasta ahora, ninguna de las soluciones propuestas ha conseguido el éxito que han alcanzado los sistemas de localización y navegación empleados en exteriores, como el GPS (*Global Positioning System*). Las razones de este fracaso han sido técnicas, necesidad de una evolución tecnológica, y también económicas, debido a la cantidad de infraestructuras fijas que se precisan como sensores, puntos de control, estaciones base, etc. [10]. Pero el sistema hibrido WSN-RFID será la mejor solución para la localización en interiores.

El sistema WSN-RFID se lo realizará mediante una red de sensores inalámbricos que consiste en dispositivos distribuidos en cada laboratorio. Un sistema WSN incorpora un *Gateway* que provee conectividad inalámbrica de

regreso a los nodos distribuidos. El protocolo inalámbrico es el estándar IEEE 802.15.4 que es la norma de la tecnología *ZigBee* que permite la comunicación inalámbrica a alto nivel entre dispositivos. Los módulos *XBee* utilizan también el protocolo IEEE 802.15.4. El objetivo es crear redes tipo *mesh* que tengan las propiedades de auto-recuperación y bajo consumo de energía. Las placas Arduino *XBee shield* puede ser usada con diferentes módulos *XBee* que permite a una placa Arduino comunicarse de forma inalámbrica usando *ZigBee*. A continuación en la Figura I.1 se muestra la topología propuesta.

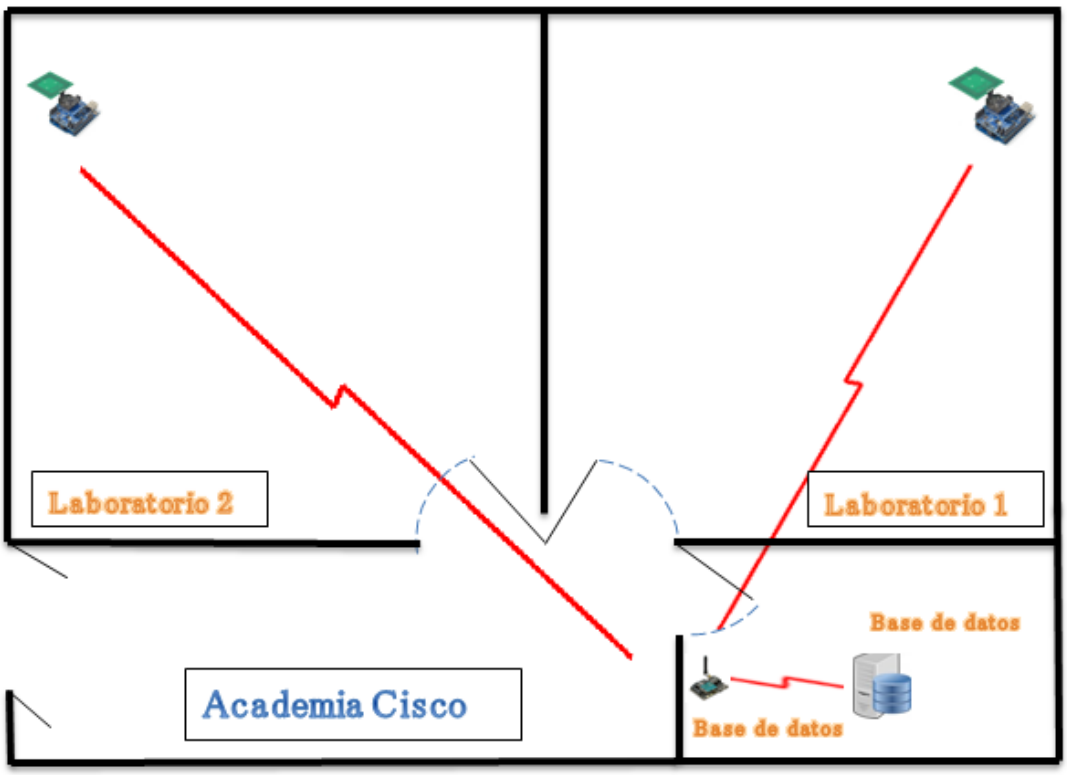


Figura I.1: Topologías propuestas de estudio.

Fuente: Elaboración Propia.

En la actualidad no existe en la academia CISCO-ESPOCH un sistema de registro automático de asistencia de los docentes, si no que el registro se lo realiza de manera manual es decir el docente tiene que acercarse una vez

terminada su hora de clase a firmar, como se puede mostrar en la Figura I.2 y Figura I.3 es el registro manual que realizan los docentes de la EIE-TR.

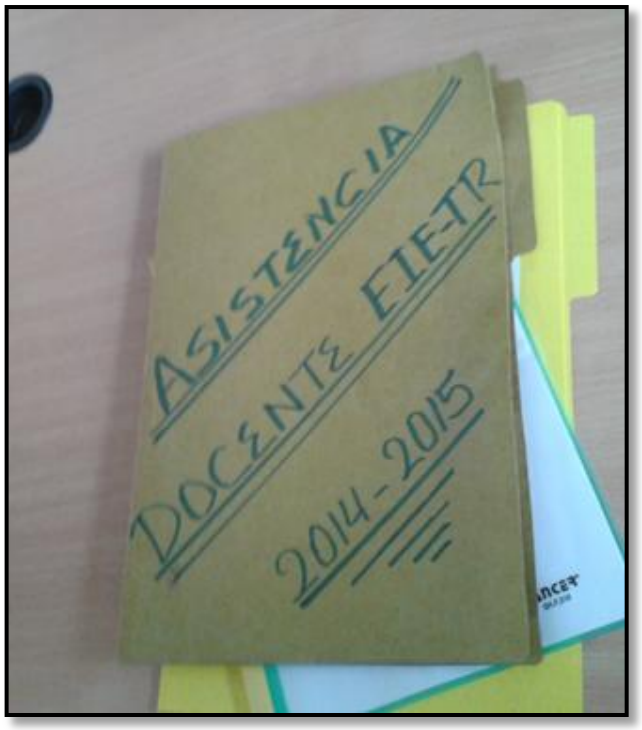


Figura I.2: Registro manual de asistencia docente 1.
Fuente: Elaboración Propia.

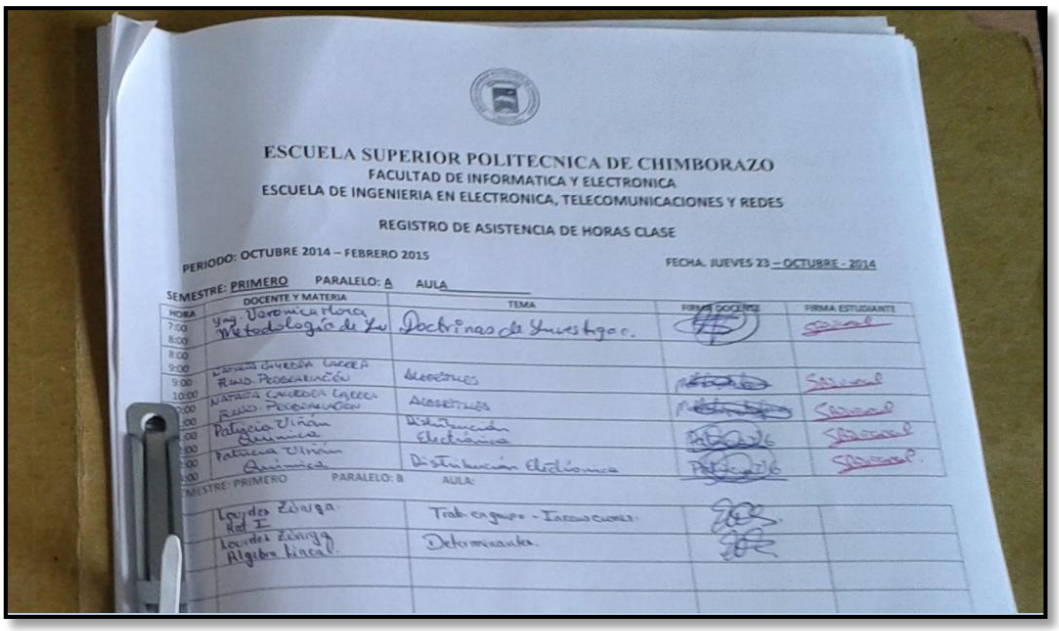


Figura I.3: Registro manual de asistencia docente 2.
Fuente: Elaboración Propia.

El sistema que se propone es basado en la interoperabilidad entre las tecnologías WSN y RFID. Con este sistema se pretende controlar el acceso a los laboratorios para control de asistencia y llevar un registro de los docentes que accedieron a los laboratorios. El docente portará una tarjeta RFID única donde estará registro sus datos, al ingresar al laboratorio el docente deberá acercarle su tarjeta al lector para registrar su entrada la cual se reflejara en la base de datos que será administrada por el encargado de los laboratorios, de igual manera al terminar su hora de clase deberá acercar su tarjeta al lector RFID para registrar su salida de la hora de clase. Los lectores RFID estarán en el interior del laboratorio para una mejor seguridad.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL:

- Implementar el prototipo para el control de acceso Docente mediante la interoperabilidad de WSN y RFID.

1.4.2. OBJETIVOS ESPECÍFICOS:

- Estudiar el estado del arte de las redes de sensores inalámbricos WSN, RFID y determinar los distintos protocolos de comunicación existentes para el desarrollo del sistema.

- Diseñar el sistema de localización para el control de acceso Docente mediante la interoperabilidad de WSN y RFID.
- Implementar un prototipo para el control de acceso Docente de los laboratorios de la academia CISCO que permita la recolección y manipulación de los datos obtenidos.
- Evaluar el desempeño de la Red WSN y RFID realizando pruebas en los laboratorios de la academia CISCO.

1.5. HIPÓTESIS

La implementación del prototipo mediante la interoperabilidad de WSN y RFID, permitirá para el control de acceso de los Docente de la academia CISCO-ESPOCH.

1.6. MÉTODOS Y TÉCNICAS

1.6.1. MÉTODOS

Tipo de investigación científica aplicada descriptiva, no experimental, que aborda el siguiente protocolo de investigación:

1.- Evaluar el desempeño de los sensores inalámbricos (WSN) y las tarjetas RFID.

2.- Implementar el prototipo de sistema de radio localización mediante la interoperabilidad entre una red de sensores inalámbricos (WSN) y RFID.

3.- Comprobar resultados obtenidos del prototipo WSN-RFID vs el sistema anterior mediante la estadística comparativa.

4.- Definir el mejor desempeño para el diseño del prototipo.

El ciclo de vida adoptado para la ejecución del proyecto de tesis es:

- Recopilación de la información.
- Clasificación de la información.
- Formulación de la hipótesis.
- Comprobación Descriptiva.
- Evaluación de Resultados.
- Conclusiones.

1.6.2. TÉCNICAS

Las técnicas a utilizar en la elaboración de este proyecto investigativo, son las siguientes:

- Observación directa: es la inspección que se hace directamente a un fenómeno dentro del medio en que se presenta, a fin de contemplar todos los aspectos inherentes a su comportamiento y características dentro de ese campo [11].

- Método Inductivo*:* es el estudio de las pruebas que permiten medir la probabilidad inductiva de los argumentos así como de las reglas para construir argumentos inductivos fuertes [12].

- Pruebas de software: Su principal característica es que el aprendizaje, el diseño y la ejecución de las pruebas se realizan de forma simultánea [13].

CAPÍTULO II

MARCO TEÓRICO

2.1. INTRODUCCIÓN

Las tecnologías de redes inalámbricas han tenido un rápido desarrollo, hemos empezado con los infrarrojo para comunicaciones punto apunto a las WPAN de corto alcance y multipuntos como es el *BlueTooth* o las redes de rango de alcance medio multisaltos como *ZigBee.* Su desarrollo está orientado a la redes de sensores inalámbricos ya que por medio de estas se puede desarrollar múltiples aplicaciones, lo que le convierte en una de las tecnologías más prometedoras del futuro.

Red Inalámbrica de Sensores es la tecnología más utilizada para la localización en interiores, ya que es barato y fácil de implementar. Sin embargo, la precisión de localización está limitada por la dura propagación de la Radio Frecuencia RF y para superar estos efectos y mejorar el posicionamiento la precisión, la identificación por radiofrecuencia RFID se emplea para ayudar al posicionamiento. En este capítulo proponemos la interoperabilidad de estas dos tecnologías WSN y RFID.

2.2. WIRELESS SENSOR NETWORK (WSN)

Las WSN son un tipo de red inalámbrica ad hoc descentralizada porque no depende de una infraestructura preestablecida ni administración central, pertenece a las Redes Inalámbricas de Área Personal (WPAN). Las WSN están formadas por grandes cantidades de pequeños dispositivos, distribuidos físicamente, llamados nodos o motas, instalados alrededor de un fenómeno para lograr ser monitoreado, con la capacidad de almacenar y comunicar datos en una red en forma inalámbrica un control en tiempo real.

Las redes inalámbricas de sensores se encuadran dentro de la llamada Inteligencia Ambiental que significa un terreno fronterizo entre los últimos avances en computación y los nuevos conceptos de interacción inteligente entre el usuario y máquina. Prácticamente inteligencia ambiental consiste en la capacidad de adquisición de información tanto en un entorno físico como del estado actual del objeto, procesamiento y comunicación, la cual puedan comunicarse entre ellos y ofrecer nuevos servicios a sus usuarios.

Estas redes están diseñadas para aplicaciones donde no se tenga una infraestructura cableada o en lugares en las cuales no se cuente con suministro de energía eléctrica tal como se puede observar en la figura II.4 por esta razón es necesario que las motas funciones con la mínima cantidad de energía de sus fuentes como pueden ser las pilas, baterías o paneles solares y lograr comunicarse por medio de canales inalámbricos.

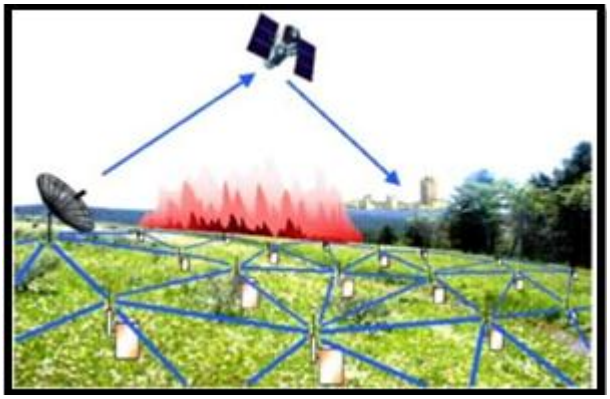


Figura II.4: Redes WSN.

Fuente:
http://3.bp.blogspot.com/_rU0Xp1joQeI/TDKVP0DLOgI/AAAAAAAAAEk/vIdNV3JD6lg/s1600/Redes%2520de%2520sensores%2520sin%2520cable.jpg

La red de sensores puede ser definida como un grupo de motas o nodos que se coordinan para llevar a cabo una aplicación específica, a diferencia de las redes tradicionales.

2.2.1. Descripción De Una WSN

Las redes inalámbricas de sensores se basan en dispositivos de bajo coste que son capaces de obtener información, procesarla localmente, y comunicarla a través de enlaces inalámbricos hasta un nodo central de coordinación. Los nodos actúan como elementos de la infraestructura de comunicaciones al reenviar los mensajes transmitidos por nodos más lejanos hacia al centro de

coordinación. La red de sensores inalámbricos está formada por numerosos dispositivos distribuidos espacialmente, que utilizan sensores para controlar diversas condiciones en distintos puntos, entre ellas la temperatura, el sonido, la vibración, la presión y movimiento o los contaminantes. Los sensores pueden ser fijos o móviles.

Los dispositivos son unidades autónomas que constan de un microprocesador, una fuente de energía casi siempre una batería, un radio transceptor y un elemento sensor tal como observamos en la figura II.5.

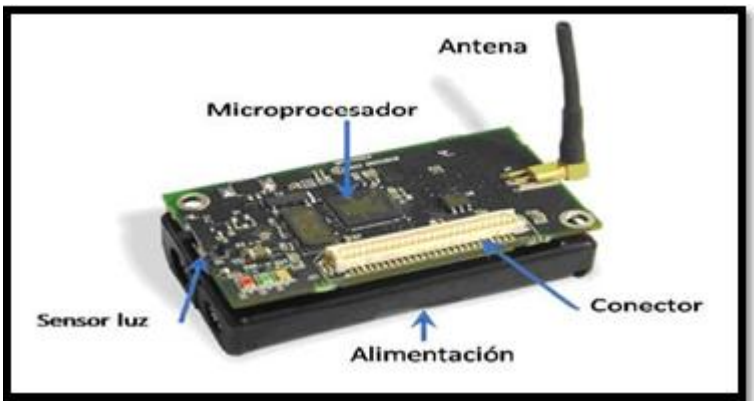


Figura II.5: Mota o nodo.

Fuente: http://lasredesconsensores.blogspot.com/2012_07_01_archive.html

Con respecto a las limitaciones de la vida de la batería, los nodos se construyen tomando en cuenta la conservación de la energía, ya que pasan mucho tiempo en modo durmiente o modo *Sleep* de bajo consumo de potencia. Las WSN tienen capacidad de auto restauración, es decir, si deja de funcionar un nodo, la red encontrará nuevas vías para encaminar los paquetes de datos. De esta forma, la red sobrevivirá en su conjunto, aunque existan nodos individuales que pierdan potencia o se destruyan.

Las redes de sensores se caracterizan por ser redes desatendidas es decir sin la intervención humana, con alta probabilidad de fallo en los nodos o en su topología. La red Ad-Hoc significa una red en la que no hay un nodo central, sino que todos los dispositivos están en igualdad de condiciones, es el modo más sencillo para crear una red, es un tipo de red formada por un grupo de nodos que forman una red temporal sin la ayuda de ninguna infraestructura externa. Para llevar a la práctica es necesario que los nodos se puedan ayudar mutuamente para conseguir un objetivo común, para que cualquier paquete llegue a su destino aunque el destinatario no sea accesible directamente desde el origen.

El responsable de descubrir las rutas entre los nodos para hacer posible la comunicación es el protocolo de encaminamientos. Las redes de sensores son un concepto relativamente nuevo en adquisición y tratamiento de datos con múltiples aplicaciones en distintos campos, tales como entornos industriales, Domótica, entornos militares, detección ambiental etc[14].

2.2.2. Características

Las redes de sensores tienen una serie de características propias y otras adaptadas de las redes Ad-Hoc:

- **Topología Dinámica:** En una red de sensores, la topología siempre es cambiante y éstos tienen que adaptarse para poder comunicar nuevos datos adquiridos.
- **Variabilidad del canal:** El canal radio es un canal muy variable en el que existen una serie de fenómenos como pueden ser la atenuación,

desvanecimientos rápidos, desvanecimientos lentos e interferencias que puede producir errores en los datos.

- **Infraestructura de red:** Una red de este tipo que utiliza sensores no tiene la necesidad de una infraestructura para poder operar, ya que sus nodos o motas pueden actuar de emisores, receptores o enrutadores de la información, el nodo recolector también denominados *sinknode*, que es el nodo que recolecta la información y por el cual se recoge la información generada normalmente en tiempo discreto.
- **Tolerancia a errores:** esta característica hace referencia a la capacidad que tiene los nodos para seguir funcionando, a pesar de presentarse fallos en la red o sistema.
- **Consumo energético:** Este factor es el más sensible ya que deben combinar autonomía energética con capacidad de proceso, ya que cuentan actualmente cuentan con una energía limitada.
- **Limitaciones hardware:** Para poder conseguir un consumo ajustado, se hace indispensable que el hardware sea lo más sencillo posible, así como su transceptor radio, esto nos deja una capacidad de proceso limitada.

- **Costes de producción:** Dada que la naturaleza de una red de sensores tiene que ser en número muy elevada, para poder obtener datos con fiabilidad, los nodos sensores una vez definida su aplicación, son económicos de hacer si son fabricados en grandes cantidades [15].

2.2.3. Elementos De Una WSN

Las redes de sensores inalámbricos están formadas por un conjunto de elementos que se distribuidos en una área determinada para su monitorización. Uno de estos elementos denominado nodos sensores tiene una capacidad limitada ya sean de almacenamiento o de procesamiento de información y comunicación, cuyo tiempo de vida depende de la fuente de energía instalada en el dispositivo. En las redes de comunicación ad-hoc no existe un nodo central sino que todos los nodos trabajaran de la misma manera.

Por su facilidad de despliegue y por ser auto configurables, pudiendo convertirse en todo momento en emisor, receptor, ofrecer servicios de encaminamiento entre nodos sin visión directa, así como registrar datos referentes a los sensores locales de cada nodo. En la Figura II.6 podemos apreciar los elementos de una Red de Sensores Inalámbricos en la cual realiza mediciones, aquella información le transforma en digital en el propio nodo y transmitirla fuera de la red de sensores a través de un elemento llamado *Gateway* hacia una estación base, donde la información pueda ser almacenada temporalmente para que finalmente en un servidor con una gran capacidad nos permita realizar un análisis de datos.

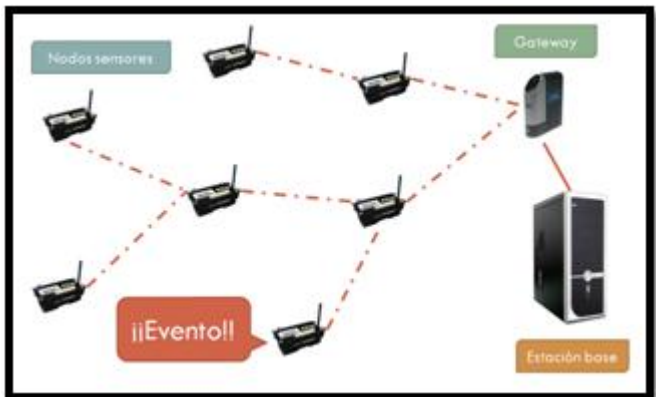


Figura II.6: Elementos de WSN.

Fuente: http://2.bp.blogspot.com/-IZxuE1VMSJc/UAXVb7reUQI/AAAAAAAAABM/7yAvXSkPCLY/s1600/WSN.png

A continuación se explicara cada elemento que conforma la estructura de una red WSN.

2.2.3.1. Nodos Inalámbricos

Los nodos inalámbricos se llaman motas o motas por su reducido tamaño y ligereza. Éstos son dispositivos electrónicos capaces de captar información del medio, procesarla y transmitirla inalámbricamente hacia otro destinatario convirtiéndolos en señales eléctricas.

Al diseñar una mota hay que tener en cuenta que se quiere un espacio reducido, un consumo muy bajo de energía y un coste de los dispositivos reducido. Una mota aislada tiene muy poca utilidad ya que son diseñadas y programadas para formar parte de una red con un objetivo particular. El hardware de cada uno de estas motas tiene varias partes que se diferencian claramente.

2.2.3.2. Nodos Sensores

Estos nodos están formados por una mota y una placa de sensores, siendo la mota el conjunto formado por un procesador y los dispositivos de radio, para el procesamiento de procesamiento y de comunicación al nodo sensor. Los procesadores de radio, toman los datos del nodo sensor a través de sus puertas de datos, y envían la información a la estación base.

En el mercado existen placas con sensores de medida de muy diversos parámetros, como sensores de presión barométrica, GPS, luz, medida de radiación solar, humedad en suelo, humedad aire, temperatura, sonido, velocidad del viento y un largo etc.

2.2.3.3. Gateway

Es el que nos permite la interconexión entre la red de sensores y una red de datos como por ejemplo puede ser TCP/IP. Gateway de la red que sirve a su vez como programador con conexión *Ethernet* al que nos podemos conectar desde un PC.

También se los llama puerta de enlace ya que interconectar dos redes de diferente naturaleza, pero el término más conocido en el ambiente de las redes es *Gateway.*

2.2.3.4. Estación Base

Recolector de datos basado en un ordenador común o sistema embebido es aquí donde se procesa e interpreta la información.

2.2.3.5. Red Inalámbrica

Es el medio por el cual permite enviar y recibir datos para la comunicación entre los nodos o motas. Los medios a elegir para realizar una comunicación inalámbrica son varios, radio frecuencia, comunicación óptica mediante láser e infrarrojo, la radio frecuencia (RF) es la más adecuada para usar en aplicaciones inalámbricas. Las WSN usan las frecuencias de comunicación que anda entre 433 MHz y 2.4Ghz. El sistema más popular dentro de los sistemas de comunicación de radio para nodos de redes inalámbricos está típicamente basado en el estándar 802.15.4ZigBee [16].

2.2.4. Componentes De Un Nodo WSN

Los componentes de un nodo WSN es decir su estructura como se muestra en la Figura II.7.

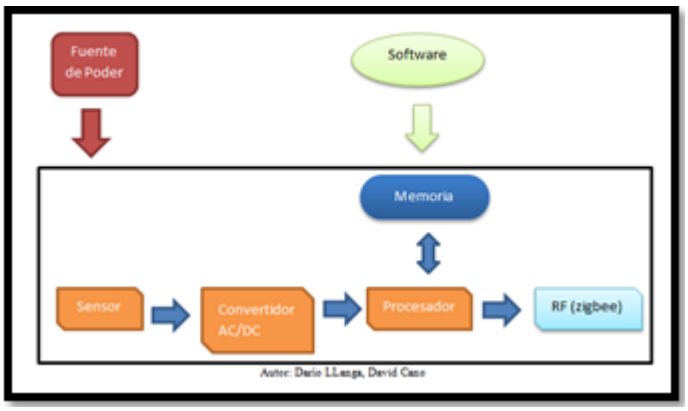


Figura II.7: Componentes de WSN.

Fuente: Elaboración propia

El nodo de una WSN integra detección, transformación, procesamiento, almacenamiento y transmisión inalámbrica con la ayuda de una fuente de poder. A continuación se explica brevemente alguno de estos elementos.

2.2.4.1. Fuente De Poder

Las fuentes de poder es el que otorga la electricidad imprescindible para alimentar al nodo o mota. La fuente de poder debe ser eficiente y ser capaz de proporcionar energía al nodo un largo periodo de tiempo, dependiendo de la aplicación que vaya a tener el nodo. En la Figura II.8 se puede observar que la mota está funcionando con 4 pilas AA.

Figura II.8: Fuente de Poder.

Fuente: http://t2.gstatic.com/images?q=tbn:ANd9GcRSuU2ye5vH8BOL_xFpYqZPOp_G9wQaOEuBF6cMehoi1Rma_JLIkA

Además de los requerimientos de larga duración, debe considerar el tamaño, peso y disponibilidad de las baterías, así como estándares internacionales para envíos de baterías. El bajo costo y la amplia disponibilidad de las baterías de zinc-carbono y alcalinas las hace una opción muy común. Las técnicas para recolección de energía también se están volviendo más frecuentes en redes de sensores inalámbricas. Con dispositivos que utilizan celdas solares o colectan calor del ambiente, se puede reducir o hasta eliminar la necesidad de usar baterías. Los estados de un nodo son los siguientes:

- *Sleep* (durmiendo): la mayor parte del tiempo pasa en modo Sleep para ahorrar energía.

- *Wakeup* (despertando): minimizar este tiempo para pasar rápidamente al estado de trabajo.
- *Active* (activo): trabajando el mínimo tiempo posible y retorno inmediato al estado Sleep.

2.2.4.2. Sensores

Los sensores son dispositivos diseñado para recibir información de una magnitud del exterior y transformarla en otra magnitud, eléctrica normalmente, para tener la capacidad de cuantificar y manipular dicha información.

Un tipo de clasificación muy básico, es diferenciar a los sensores atendiendo a la naturaleza de la señal eléctrica generada, así se tiene a los sensores de tipo Pasivos o Activos; los sensores activos generan la señal de salida sin la necesidad de una fuente de alimentación externa, mientras que los pasivos si requieren de esta alimentación para poder efectuar su función [17].En la Figura II.9 se muestra un sensor para placas Arduino.

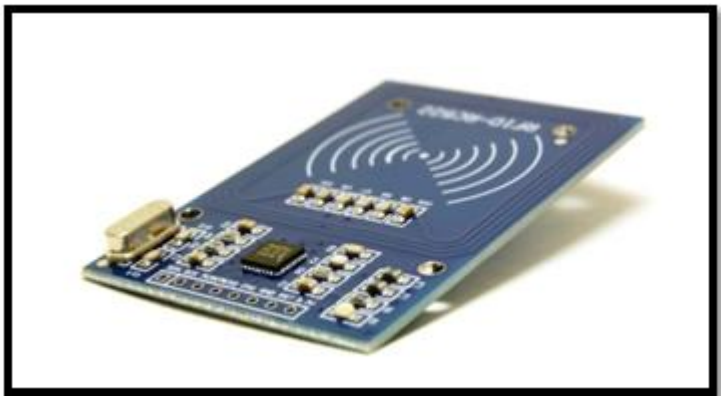


Figura II.9: sensor.

Fuente: http://mlm-s1-p.mlstatic.com/modulo-lector-rfid-rc522-con-tarjeta-y-llavero-arduino-pic-5953-MLM5021588939_092013-F.jpg

2.2.4.3. Procesador

El procesador es un circuito integrado central y más complejo de un sistema informático. Es el encargado de ejecutar las instrucciones programadas en lenguaje de bajo nivel, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir, las lógicas binarias y accesos a memoria.

Los requisitos cómputo y almacenamiento para una WSN dependen de la aplicación ya que se puede utilizar un micro controlador de 8 bits hasta 64 bits y almacenamiento pueden oscilar entre 0,01 hasta 100 gigabytes GB. Además de la memoria proporcionada por el micro controlador se lo puede encontrar modelos que incluyan memoria externa adicional, como por ejemplo en forma de memoria flash. En la Figura II.10 se observa un Arduino Uno que el cual cumplirá con las tareas del procesador en los nodos o motas.

Figura II.10: Microprocesador Arduino.

Fuente: http://www.arduino.cc/es/

2.2.4.4. Radio

El Radio es un dispositivo de comunicación que permite enviar y recibir datos inalámbricamente entre todos los nodos dentro de su rango de transmisión que

forma una WSN. Las WSN utilizan frecuencias entre 433MHz y 2.4GHz para su comunicación.

El transceptor es el encargado de la emisión y recepción de la información. Los estados de operación son: emitir, recibir, dormir e inactividad, el modo recepción consume casi igual que el modo inactivo, ya que es recomendable apagar las comunicaciones radio, en el modo inactivo, cuando no se está emitiendo ni recibiendo información. Existe mayor cantidad de energía consumida cuando cambia de modo durmiente a transmisión de datos.

En la Figura II.11 se muestra una antena *XBee* que es un módulo que permite crear redes *mesh* muy complejas basadas en el *firmware* de *XBee ZB ZigBee*. Estos módulos son bastante confiables para comunicación sencilla entre microcontroladores, computadoras, sistemas y cualquier dispositivo con puerto serial. Soporta comunicación punto-punto y multipunto, están diseñados para aplicaciones de alto rendimiento que requieren baja latencia y tiempo de comunicación predecibles [18].

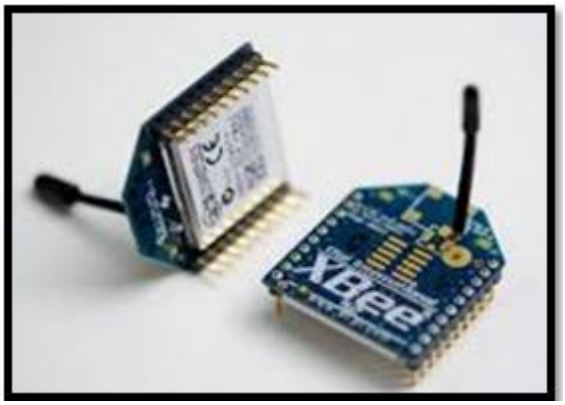


Figura II.11: Antena Wireless XBee.

Fuente: https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQ_cb34finKGcTj-45XQDaotvfjGgUhkmU14DYYSAwyy_HL-orvFg

2.2.5. Aplicaciones de las Redes de Sensores Inalámbricos

Las aplicaciones de las WSN son diversas, debido a que su estructura no tiene cables, el acceso a lugares remotos su bajo consumo de energía, hace posible que se utilice en todas las áreas en las que se requiera monitorear, estudiar y almacenar información que receptan los sensores de los nodos. A continuación se presenta diferentes áreas de aplicación de las WSN con las características más común, su desarrollo y sus usos.

Aplicaciones Militares

Fue uno de los escenarios que primero se dieron la aplicación de esta tecnología WSN, como por ejemplo son los vehículos robóticos no tripulados, tanques aviones de guerra, submarinos, misiles y torpedos para ser guiado de obstáculos hacia una posición exacta y así lograr ataques o defensas más efectivas, también la detección remota de armas nucleares, biológica y químicas. Las WSN tienen mayor importancia en las tareas militares ya que logran ataques y defensas más inteligentes y con menor involucramientos del personal humano.

Aplicaciones en la Agricultura

En esta área de la agricultura es donde se prevé que pueda implantarse con mayor rapidez este tipo de tecnología. Por ejemplo, las redes de sensores favorecen una reducción en el consumo de agua y pesticidas, contribuyendo a la preservación del entorno también pueden alertar sobre la llegada de heladas, así como ayudar en el trabajo de las cosechadoras. Gracias a los desarrollos de esta tecnología en los últimos años, especialmente la miniaturización de los

dispositivos, han surgido nuevas tendencias en el sector agrícola como la llamada agricultura de precisión.

Aplicaciones en el Medio Ambiente

En el medio Ambiente la aplicación de WSN nos permite poder distribuir una gran cantidad de sensores en un espacio natural y obtener una gran cantidad de datos que de otra forma es imposible de obtener con la instrumentación tradicional. Nos ayuda a la detección de incendios forestales, la detección de inundaciones, entre otras.

Aplicaciones en el Monitoreo de Estructuras

Una gran aplicación de las WSN es el monitoreo de estructuras físicas como puentes, edificios y construcciones en general experimentan vibraciones, por actividades normales para las que fueron construidas o por fenómenos naturales. En Estados Unidos y Canadá se estima que tienen unos 25 trillones de dólares invertidos en estructuras civiles y ante tanta inversión se desea tener un control de las estructuras realizadas como la identificación y el monitoreo de comportamientos extraños.

Aplicaciones en la Salud

En la salud las investigaciones y desarrollos de redes de sensores inalámbricos han aumentado tanto en el nivel académico para aplicaciones médicas, que pueden llevar a cabo la monitorización de pacientes, los diagnósticos de enfermedades, administración de la medicina, monitoreo de movimiento de pacientes dentro del hospital y demás funciones la cual se envían la

información hasta las oficinas de los médicos o simplemente avisando al paciente si alguna de sus señales cae drásticamente.

Aplicaciones en la Domótica

En este tipo de aplicaciones como por ejemplo tenemos el control de una casa lo que hará más cómoda la permanecía de las personas, como puede ser en aplicaciones de seguridad y protección de las personas o bienes patrimoniales.

2.3. RADIO FREQUENCY IDENTIFICATION (RFID)

2.3.1. Descripción De Una RFID

La tecnología RFID permite la identificación de objetos de forma inalámbrica vía radio, sin la necesidad de que exista contacto o línea de visión directa entre el lector y el objeto, requisito imprescindible para el funcionamiento de otras tecnologías como la lectura láser de códigos de barras. La identificación se realiza mediante la integración o incorporación de un transponder o tag al objeto, el cual es el encargado de transmitir los datos que contiene cuando detecta que está siendo interrogado por un lector RFID.

Aunque el desarrollo de esta tecnología no es nuevo, los avances técnicos en aspectos tales como alcance, seguridad, capacidad de almacenamiento o velocidad de lectura entre otros, han suscitado el interés de la industria por este tipo de tecnología llegando incluso a considerarla como el sustituto natural del código de barras debido a la importante oportunidad que la tecnología RFID ofrece para conseguir una importante reducción de costes y tiempo en las cadenas de producción y logística. En varias empresas internacionales con una

importante carga logística o de producción se ha comenzado a utilizar la tecnología o se ha exigido a sus proveedores que la incorporen, motivadas por las notables mejoras que supone su introducción para sus procesos productivos. Algunos casos ampliamente documentados son los de grandes empresas como Wal-Mart, Nokia, Michelin, Departamento de Defensa de Estados Unidos, BMW, Volvo, Hewlett-Packard o Tesco.

Debido a que existen muchas áreas en las cuales se puede utilizar la tecnología RFID en todos los sectores de actividad se la considera uno de los pilares básicos de la siguiente evolución de las redes de comunicación como por ejemplo en el denominado “internet de las cosas”, el cual se basa en la interacción automática e inteligente entre dispositivos en cualquier circunstancia o ubicación, y su comunicación con sistemas remotos de datos a través de las redes de telecomunicación.

El sistema RFID utiliza un lector con un micro controlador incrustado y una antena que opera a 13,56 MHz . El lector mantiene a su alrededor un campo electromagnético de modo que al acercarse una tarjeta al campo, ésta se alimenta eléctricamente de esta energía inducida y puede establecerse la comunicación lector-tarjeta.

2.3.2. Características

La identificación por radiofrecuencia es una tecnología que permite la captura e identificación automática de información, almacenamiento y recuperación remota de datos previamente almacenados en las etiquetas en las que reside la información. Cuando estas etiquetas entran en el área de cobertura de un

lector este se encarga de enviar una señal para que la etiqueta le responda enviando la información contenida en su memoria. Una de las ventajas de la identificación por radiofrecuencia es que el intercambio de información entre la etiqueta y el lector se realiza vía radiofrecuencia, sin la necesidad de que exista contacto físico o visual (línea de vista) aunque en determinados casos es necesaria cierta proximidad entre los elementos.

RFID emplea señales de radiofrecuencia (en diferentes bandas dependiendo del tipo de sistema, típicamente 125 KHz, 13,56 MHz, 433-860-960 MHz y 2,45 GHz)[19].Existe una gran diversidad de sistemas RFID, siendo posible desarrollar un gran número de aplicaciones para los que pueden ser utilizados. Todos estos sistemas se basan en el mismo principio de funcionamiento, que se describe a continuación:

1. Se equipa a todos los objetos a identificar, controlar o seguir, con una etiqueta RFID.
2. La antena del lector o interrogador emite un campo de radiofrecuencia que activa las etiquetas.
3. Cuando una etiqueta ingresa en dicho campo utiliza la energía y la referencia temporal recibidas para realizar la transmisión de los datos almacenados en su memoria. En el caso de etiquetas activas la energía necesaria para la transmisión proviene de la batería de la propia etiqueta.
4. El lector recibe los datos y los envía al ordenador de control para su procesamiento [20].

2.3.3. Componentes

Los sistemas RFID se componen principalmente de los siguientes elementos:

- Tarjeta RFID, también conocida como tag. Esta etiqueta se inserta o adhiere en un objeto, animal o persona, portando información sobre el mismo. Se compone de un microchip donde se almacena los datos y una antena que se encarga de la comunicación por radiofrecuencia con el lector.
- Un lector, es el que está encargado de transmitir la energía suficiente hacia la tarjeta RFID activándola y esta enviará los datos contenidos en la tarjeta hacia el lector. El lector está formado por un módulo de radiofrecuencia que trabaja como transmisor y receptor, también cuenta con un microcontrolador y una antena.

2.3.3.1. Transponder O Tag

La tarjeta RFID contiene un microchip en el cual están grabados los datos de cada tarjeta, la cual será transmitida al lector cuando este la solicite. Está compuesta por un microchip, una antena y en las tarjetas activas una batería.

El microchip incluye:

- Una circuitería analógica que se encarga de realizar la transferencia de datos y de proporcionar la alimentación.
- Una circuitería digital que incluye:

  · La lógica de control.

  · La lógica de seguridad.

  · La lógica interna o microprocesador.

  · Una memoria para almacenar los datos. Esta memoria suele contener:

Una ROM (Read Only Memory) para alojar los datos de seguridad y las instrucciones de funcionamiento del sistema.

Una RAM (Random Access Memory) utilizada para facilitar el almacenamiento temporal de datos durante el proceso de interrogación y respuesta.

Una memoria de programación no volátil, utilizada para asegurar que los datos están almacenados aunque el dispositivo esté inactivo. Típicamente suele tratarse de una EEPROM (Electrically Erasable Programmable ROM).Este tipo de memorias permite almacenar desde 16 bytes hasta 1 Mbyte, posee un consumo elevado, un tiempo de vida (número de ciclos de escritura) limitado de entre 10.000 y 100.000 en un tiempo de escritura de entre 5 y 10 ms[18].

- Registros de datos (buffers) que soportan de forma temporal, tanto los datos entrantes después de la demodulación como los salientes antes de la modulación. Además actúa de interfaz con la antena.

La información contenida en la etiqueta es transmitida modulándola en amplitud ASK (Amplitude Shift Keying), frecuencia FSK (Frequency Shift Keying) o fase PSK (Phase Shift Keying).

Para realizar la transmisión se modifica la amplitud, frecuencia o fase de la señal del lector. La modulación más utilizada es la ASK debido a su mayor facilidad a la hora de realizar la demodulación.

La frecuencia utilizada por el transponder, en la gran mayoría de los casos, es la misma que emite el lector. Sin embargo, en ocasiones se trata de una frecuencia subarmónica (submúltiplo de la del lector) o incluso de una frecuencia totalmente diferente de la del lector.

2.3.3.1.1. Tarjetas Activas

Las tarjetas funcionan con una batería, porque les permite guardar mayor información y alcanzar mayores distancias de transmisión, el uso de una batería aumenta el tiempo de vida del dispositivo, al acoplar una batería de forma apropiada a la circuitería de baja potenciase puede obtener un tiempo de vida de algo más de 10 años, esto dependerá de las condiciones de trabajo en las que se encuentre como la temperatura, ciclos de lectura/escritura y su utilización.

Las etiquetas RFID activas tienen un mayor radio de cobertura, mayor inmunidad al ruido y ofrece una tasa de transmisión más alta cuando trabaja con altas frecuencias. Todas estas ventajas hacen más costosas a este tipo de etiquetas, por lo que se las utiliza cuando los bienes a identificar lo justifican.

Existen dos tipos de etiquetas activas:

- Las que normalmente se encuentran desactivadas (modo reposo) y se activan (despiertan) cuando un lector las interroga. Ahorrando batería.
- Las que envían señales periódicamente, aunque no exista un lector que las interrogue.

Trabajan con frecuencias más bajas, ofreciendo menores tasas de transferencias con el fin de ahorrar batería. En la Figura II.12 se muestra una tarjeta RFID activa.

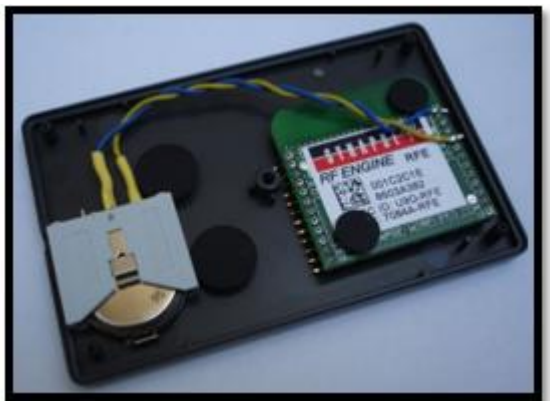


Figura II.12: Tarjeta RFID activa.

Fuente: http://www.ns-tech.co.uk/blog/wp-content/uploads/2010/01/tag_nocover.jpg

2.3.3.1.2. Tarjetas Pasivas

Este tipo de etiquetas funcionan sin una batería interna, estas etiquetas obtienen la potencia necesaria para funcionar del campo generado por el lector. Al no poseer una batería estas etiquetas son más delgadas, pequeñas, flexibles y baratas que los activas, lo que hace posible diseñarlas en una amplia gama de formas, una de sus principales ventajas es que ofrecen un tiempo de vida prácticamente ilimitado.

Entre sus desventajas se destaca su menor radio de cobertura y requieren más cantidad de energía procedente del lector para poder iniciar la transmisión de datos, poseen una limitada memoria para el almacenamiento de datos y no funcionan correctamente en ambientes con interferencias electromagnéticas. La sensibilidad está limitada por la potencia disponible por parte del lector.

A pesar de estas limitaciones, las etiquetas pasivas ofrecen mejores ventajas en términos de coste y longevidad. En la Figura II.13 se muestra una tarjeta RFID pasiva.

Figura II.13: Tarjeta RFID pasiva.

Fuente: http://www.ns-tech.co.uk/blog/wp-content/uploads/2010/01/tag_nocover.jpg

2.3.3.2. Lector

Un lector o interrogador es el dispositivo que se encarga de enviar una señal a las etiquetas, esperando a que estos les respondan con la información contenida en las tarjetas. Para cumplir estas funciones, está equipado con un módulo de radiofrecuencia, una unidad de control y una antena.

El lector puede actuar de tres modos:

1. Interrogando continuamente su zona de cobertura, esperando la presencia de etiquetas de forma continua.
2. Interrogando periódicamente con el fin de detectar nuevas etiquetas en su zona de cobertura.
3. Interrogando de forma puntual, por ejemplo cuando la presencia de una nueva etiqueta sea detectada por un sensor.

El lector está compuesto por un módulo de radiofrecuencia (formado por receptor y transmisor), la unidad de control y la antena. El módulo de radiofrecuencia consta básicamente de un transmisor que genera y envía una señal de radiofrecuencia y un receptor que es el encargado de recibirla los datos enviados por las etiquetas.

Sus funciones son:

- Generar la señal de radiofrecuencia para proporcionar energía y activar la etiqueta.
- Modular la señal transmitida para enviar los datos a la etiqueta.
- Recibir y demodular las señales enviadas por la etiqueta.
- La unidad de control está básicamente constituida por un microprocesador, adaptado a los requerimientos deseados para la aplicación.

La unidad de control se encarga de realizar las siguientes funciones:

- Codificar y decodificar los datos recibidos de las etiquetas.
- Verificar la integridad de los datos y almacenarlos.
- Gestionar el acceso al medio: activar las etiquetas, inicializar la sesión, autenticar y autorizar la transmisión, detectar y corregir errores, gestionar el proceso de multilectura, cifrar y descifrar los datos, etc.

Establecer comunicación con el sistema de información, realizando las órdenes recibidas y enviando la información obtenida de las etiquetas. Una de las funciones más críticas que debe realizar la unidad de control es gestionar el acceso al medio. Cuando se realiza la transmisión de información utilizando una tecnología en la que no es requerido el contacto físico pueden aparecer interferencias que provoquen cambios indeseados en la información transmitida provocando errores durante la transmisión, para evitar este tipo de problemas se utilizan métodos de comprobación. En la siguiente Figura II.14 se muestra el lector RFID.

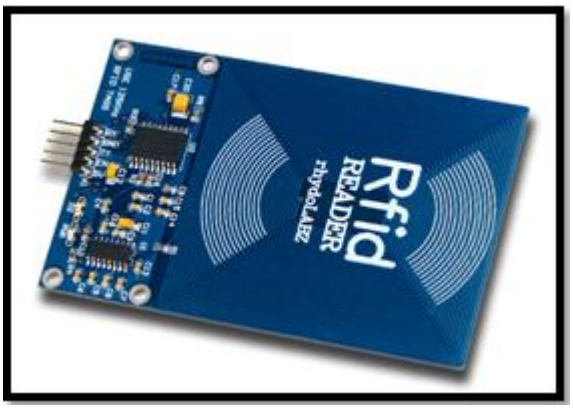


Figura II.14: Lector RFID.
Fuente: http://www.rhydolabz.com/blog/wp-content/uploads/2012/12/RFID-Serial-TTL.jpg

Los parámetros principales de un lector RFID son:

Frecuencia de operación: El lector puede funcionar a baja frecuencia, alta frecuencia, ultra alta frecuencia y frecuencia de microondas.

Protocolo de funcionamiento: Muchos lectores ofrecen soporte multiprotocolo, pero no admiten todos los protocolos existentes.

Tipo de regulación que siguen: Existen distintas regulaciones de frecuencia y de potencia en Estados Unidos y en Europa. La banda de UHF funciona a 902–930 MHz en Estados Unidos y a 869 MHz en Europa. En Estados Unidos la potencia máxima permitida es de 2 Watts y en Europa 0,5 Watts.

2.3.4. Frecuencia

Los rangos de frecuencia utilizados por los sistemas RFID son:

Baja Frecuencia (BF): rangos de frecuencia inferiores a 135 KHz.

Alta Frecuencia (AF): la frecuencia de funcionamiento es de13,56 MHz.

Ultra Alta Frecuencia (UHF): cuando las frecuencias de funcionamiento están en las bandas de 433 MHz, 860 MHz, 928 MHz.

Frecuencia de Microondas: cuando las frecuencias de funcionamiento están en las bandas de 2,45 GHz y 5,8 GHz.

2.3.5. Aplicaciones en RFID

Esta tecnología es utilizada en aplicaciones que requieran leer poca cantidad de datos y para pequeñas distancias. Por ejemplo: control de accesos, identificación de animales, gestión de bienes, identificación de vehículos y contenedores, y como soporte a la producción.

Aplicación en el Comercio

Las etiquetas RFID son la alternativa que reemplazará a los códigos de barras UPC (Universal Product Code) o EAN (Europe Article Number), debido al gran número de ventajas frente a la tecnología de código de barras, como una mayor facilidad para llevar a cabo inventarios.

Es mucho más fácil realizar el control de la mercadería nivel de pallet usando etiquetas RFID, y a nivel de artículo con etiqueta única, en lugar de códigos de barras únicos por artículo.

Aplicación en el Transporte y Logístico

Actualmente, la aplicación más importante de RFID es la logística. El uso de esta tecnología permitiría tener localizado cualquier producto dentro de la cadena de suministro. En lo relacionado a la trazabilidad, las etiquetas podrían

tener gran aplicación ya que las mismas pueden grabarse, con lo que se podría conocer el tiempo que el producto estuvo almacenado.

Aplicación en Identificación de los Animales

Las etiquetas de baja frecuencia también aparecen en la identificación animal con el fin de gestionar el ganado, identificar y controlar las especies protegidas o identificar animales domésticos.

Aplicación en los Deportes

Una de las mejores manera de cronometrar con precisión los tiempos de los deportistas es usando la tecnología RFID, para lo cual se utiliza etiquetas de identificación por radiofrecuencia que pueden ir adheridos a los zapatos deportivos y que envían una señal al pasar por determinados lugares de control. Con otros métodos es difícil obtener precisión en tiempos individuales, sobre todo cuando hay cientos o miles de personas corriendo.

Aplicación en la Identificación Humana

Existen chips RFID que pueden ser implantados bajo la piel, diseñados originalmente para el etiquetado de animales, se los está utilizando también para los seres humanos.

2.3.6. NFC

NFC (Near field communication) es una tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de

comunicación y formatos de intercambio de datos, y están basados en ISO 14443 es decir en la tecnología RFID.

Figura II.15: NFC.

Fuente: http://img.xataka.com/2011/01/nfc-1.jpg

NFC es una plataforma abierta pensada desde el inicio para teléfonos y dispositivos móviles. Su tasa de transferencia puede alcanzar los 424 kbit/s por lo que su enfoque más que para la transmisión de grandes cantidades de datos es para comunicación instantánea, es decir, identificación y validación de equipos/personas [21].

NFC se comunica mediante inducción en un campo magnético, en donde dos antenas de espira son colocadas dentro de sus respectivos campos cercanos. Trabaja en la banda de los 13,56 MHz, esto hace que no se aplique ninguna restricción y no requiera ninguna licencia para su uso.

2.3.6.1. Aplicación

Algunos teléfonos de gama alta y gama media incluyen NFC y cada vez comienza a ser más habitual verlos en el resto de móviles nuevos, por lo que

para el proyecto de tesis es una alternativa y poder sustituir con las tarjetas RFID.

Figura II.16: Aplicación NFC.

Fuente: http://img.xataka.com/2011/01/nfc-1.jpg

Se utilizaría para el control de acceso a las aulas de clases donde es precisa una identificación podría hacerse simplemente acercando el teléfono móvil o tarjeta con chip NFC al dispositivo de lectura.

2.4. MÓDULOS DE COMUNICACIÓN XBEE

Los módulos de comunicación *XBee* son módulos de radio frecuencia que trabajan en la banda de 2.4 GHz, son capaces de comunicarse de forma inalámbrica unos con otros, son utilizados para realizar como por ejemplo en automatización de casas, sistemas de seguridad, monitoreo de sistemas remotos, aparatos domésticos, alarmas contra incendio entre otras.

Los módulos *XBee* se basan en una plataforma compatible con *ZigBee* es decir con protocolo de comunicación 802.15.4. En el mercado existen diferentes tipos de módulos *XBee*, pero la ventajas de estos módulos es que independiente del modelo o serie, tienen los pines similares. Alimentación, tierra y los pines de comunicación TX/RX se encuentran en el mismo lugar, la cual hace que los chip sean totalmente intercambiables, para la mayoría de las aplicaciones más simples, ya que algunas de las características más avanzadas no son siempre compatibles.

Existen bastantes módulos inalámbricos, pero debido a que estos módulos *XBee* son los que tienen un bajo precio y una alta calidad, y su pequeño tamaño y fácil programación ya que se requiere una conexión serial, su ultra bajo consumo de potencia, uso de bandas de radio libres y sin necesidad de licencias, instalación barata y simple, redes flexibles y extensibles. La cual es ideal para el proyecto de tesis. A continuación en la Figura II.17 se muestran los modelos de módulos.

Figura II.17: Módulos de comunicación XBee.

Fuente: http://diegoromanoubalde.files.wordpress.com/2012/08/xbeezb.jpg

Los módulos *XBee* proveen 2 formas de comunicación, la primera es la transmisión serial transparente modo AT y la segunda es el modo API que provee muchas ventajas.

Modo AT: Esta el modo de transmisión serial transparente, en el cual la comunicación se asemeja a lo que sería una transmisión a través de un puerto serial, ya que el dispositivo se encarga de crear la trama y el dato que llegue al pin Tx que será enviado de forma inalámbrica, por lo cual se considera como el modo más sencillo para utilizar estos nodos, su principal desventaja es que para enviar información a distintos nodos es necesario entrar constantemente al modo configuración para cambiar la dirección de destino.

Modo API: En este modo de comunicación, un micro controlador externo se debe encargar de crear una trama especifica al tipo de información que se va a enviar, este modo es recomendado para redes muy grandes donde no se puede perder tiempo entrando y saliendo del modo configuración de los dispositivos. Para redes con topología en Malla este es el modo a utilizar [22].

2.4.1. Circuito básico para XBee

Las Conexión mínimas que necesita el módulo *XBee* para poder ser utilizado como se muestra en la Figura II.18 se puede observar los pines necesarios para conectar y configurar un módulo *XBee*. Luego se debe configurar según el modo de operación ya estudiado.

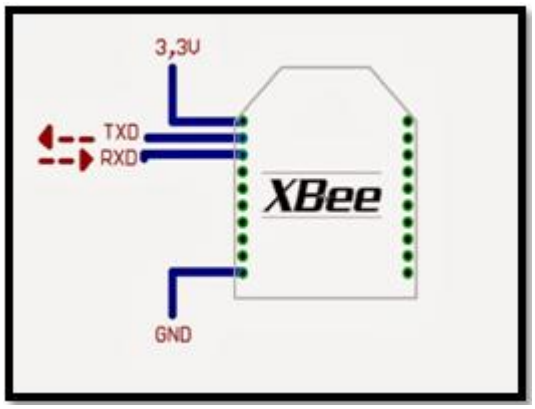


Figura II.18: Circuito básico de XBee.
Fuente: http://2.bp.blogspot.com/-pvl4OVz2GLo/UspHEQolWwI/AAAAAAAAAQw/3Z1T2XFlP-c/s1600/xc.jpg

El circuito requiere una alimentación desde 2.8 a 3.4 V, una conexión a tierra y las líneas de transmisión de datos por medio del UART(Universal Asynchronous Receiver-Transmitter) TXD y RXD para comunicarse con un micro controlador, 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 (Request to Send) y CTS (Clear to Send), por lo que ésta 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 evitar existen dos alternativas:

- Bajar la tasa de transmisión
- Activar el control de flujo.

2.4.2. Arquitectura Básica de una Red XBee

En una red *XBee* la forman básicamente 3 tipos de elementos, un único dispositivo llamado Coordinador, dispositivos Routers y dispositivos finales. Los módulos *XBee* son versátiles a la hora de establecer diversas topologías de red, dependiendo la serie de *XBee* que escojamos pueden crearse redes tipo punto a punto, estrella, malla, árbol y mixtas.

- **El Coordinador**

Es el nodo de la red cuya función es construir una red, a él van conectados el dispositivo router y los dispositivos finales cuando ya está formada la red el coordinador se comporta como un router participa en el enrutamiento de paquetes y es el origen y destino de información.

- **Los Routers**

Es un nodo que crea y mantiene información sobre la red para determinar la mejor ruta para enrutar un paquete de información. Un router se debe unir a una red *ZigBee* para retransmitir paquetes de otros Routers o dispositivos finales.

- **Dispositivos finales**

Son dispositivos que tienen la capacidad de enrutar paquetes es decir que solo interactúan con su nodo padre y su función es adquirir información que es recibida por el nodo al cual están conectados, estos nodos finales están alimentados por baterías de 9 voltios.

2.4.3. Modos de Operación

Los módulos *XBee* pueden operar en los siguientes modos como se muestra en la Figura II.19. Así los modos de operación son:

- Modo de Transmisión
- Modo de Recepción
- Modo de comando
- Modo de bajo consumo (*Sleep*)

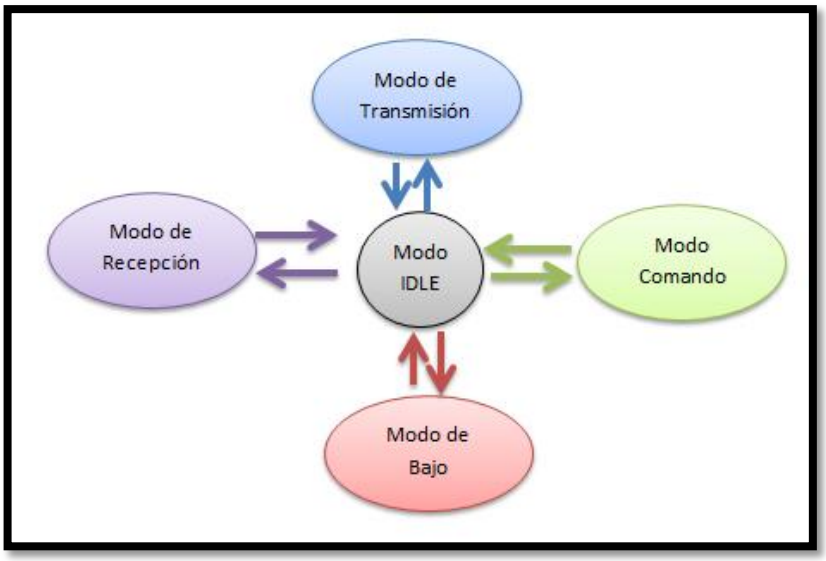


Figura II.19: Modos de Operación de XBee.
Fuente: Elaboración propia.

2.4.3.1. Modo RECIBIR/TRANSMITIR

En este modo recibir/transmitir ocurre cuando llega algún paquete RF a través de la antena, estará en modo *Recieve* o cuando se envía información a través del serial buffer que generalmente es el pin 3 del módulo UART Data in y posteriormente se transmite y ahí ya pasa a modo *Transmit.*

La información transmitida se puede enviar de forma directa o indirecta, de forma directa los datos se transmiten en forma inmediata a la dirección de

destino. La forma indirecta los datos se almacenan temporalmente mientras que la dirección de destino solicita la información, esta información se envía por *unicast* o *broadcast*, en una comunicación punto a punto, el nodo que recibe la información, envía un mensaje de confirmación ACK al nodo emisor y en una comunicación *broadcast* no existe un mensaje de confirmación ASK.

2.4.3.2. Modo de Bajo Consumo

Llamado también *Sleep Mode*, para obtener un sistema con bajo consumo de los módulos se ponen en consumo bajo se energía cuando no se está utilizando es decir el módulo RF se puede colocar en modo sueño, sin embargo debe cumplir con las siguientes características:

- Sleep_RQ(pin 9) está en alto y el módulo está en pin Sleep Mode (SM= 1,2 o 5)
- El módulo está en reposo (no hay transmisión ni recepción de datos) por la cantidad de tiempo definido por ST (Time before Sleep). [ST sólo está activado cuando SM=4,5] , por defecto SM está programado en 0 [SM=0]

La configuración de los ciclos Sleep se realiza principalmente con el comando SM. Por defecto, los modos de Sleep están deshabilitados (SM=0), permaneciendo el módulo en estado de reposo/recepción. En este estado el módulo está siempre preparado para responder a un comando, ya sea, por el puerto serial o la interfaz RF.

En la Tabla II.1se muestra los modos de operación, consumos, voltajes y las condiciones para un modo de bajo consumo.

Modo	Consumo alimentación			Modo Sleep	Modo Wake -up
	2.8 – 3 V	3.2 V	3.4V		
SM=1	<3 uA	32 uA	255 uA	Sleep_RQ	Sleep_RQ
SM=2	<35 uA	48 uA	170 uA	Sleep_RQ	Sleep_RQ
SM=3		(reservado)		(reservado)	(reservado)
SM=4	<34 uA	49 uA	240 uA	Comando ST	Comando SP
SM=5	<34 uA	49 uA	240 uA	Comando ST	Sleep RQ

Tabla II.1: Modo Sleep y consumos de corriente.
Fuente: http://www.olimex.cl/pdf/Wireless/ZigBee/XBee-Guia_Usuario

2.4.3.3. Modo de Comando

Para realizar la configuración mediante comandos AT, se deberá ingresar a este modo que es el que nos permite realizar los ajustes o modificaciones de parámetros como la dirección propia o la de destino, así como su modo de operación entre otras cosas. Para poder ingresar los comandos AT se necesita el software Hyperterminal.

La herramienta X-CTU no es más que una interfaz gráfica que nos permite modificar el conjunto de parámetros del *XBee* de una manera amigable y sencilla.

2.4.3.4. Modo Idle

Cuando el módulo no se está en ninguno de los otros modos, se encuentra en éste. Es decir, si no está ni transmitiendo ni recibiendo, ni ahorrando energía ni en el modo de comandos, entonces se dice que se encuentra en un estado al que se le llama IDLE.

2.4.4. Modelos de módulos XBEE

La empresa MaxStream es la fábrica que construye más de 70 tipos de módulos *XBee* con diferentes antenas, potencia y capacidades. Muchas de las características de los módulos *XBee* tales como velocidad de transmisión y canales por ejemplo pueden ser configurados utilizando el software X-CTU o directamente desde el microcontrolador.

2.4.4.1. Series Módulos XBEE

Existen 2 series de estos módulos la serie 1 y la serie 2 o también conocida como 2.5. Los módulos de la Serie 1 y la Serie 2 tienen el mismo pin-out, sin embargo, no son compatibles entre sí ya que utilizan distintos chipset y trabajan con protocolos diferentes.

La serie 1 está basada en el chipset *Freescale* y está pensado para ser utilizado en redes punto a punto y punto a multipunto. Los módulos de la serie 2 están basados en el chipset de *Ember* y están diseñados para ser utilizados en aplicaciones que requieren repetidores o una red mesh. Ambos módulos pueden ser utilizados en los modos AT y API.

Serie 1 o XBee 802.15.4

La Serie 1 viene con firmware 802.15.4 que permite redes punto a punto o de topología en estrella. Este firmware ofrece una conversión analógica-digital, además posee entradas y salidas digitales, y tiene un cifrado de 128-bit AES. Los *XBee* Serie 2 no ofrece ninguna clase de firmware 802.15.4, sino que siempre se está ejecutando el firmware de *ZigBee mesh.*

XBee Serie 2

La familia *XBee* Serie 2 que proporciona interoperabilidad con dispositivos *ZigBee* de otros fabricantes. Este módulo sólo es compatible con otros módulos *XBee* Serie 2, no es compatible con la serie *XBee* 1.Debido a que los módulos *XBee* tienen 2 mm de distancia entre pines será necesario el uso de un adaptador con pines de salida de 0.1 pulgadas de espacio para la conexión a una placa estándar o protoboard.

Algunas características principales son:

- Antena de cable
- Compatible con otros módulos de la Serie 2
- Bajo consumo de energía en modo de suspensión (*Sleep*)
- 133 pies (40 m) para ambientes interiores/urbanos y 400 pies (120 m) de alcance en exteriores, línea de vista
- Configuración con comandos de la API o AT, localmente o por aire
- 10 E / S digitales y 4 entradas ADC de 10 bits

XBee 900

La familia *XBee* 900 lleva su nombre por la banda ISM de 900 MHz que ocupa. Los dispositivos de esta familia conservan el factor de forma y presentan un alcance mayor que el del resto de los dispositivos *XBee*, llegando hasta los 24 kilómetros de alcance. Así como los dispositivos de la Serie 1, no soportan el protocolo *ZigBee* ni topologías de red enmalladas. Estos dispositivos fueron concebidos para satisfacer la necesidad de redes inalámbricas confiables de

dispositivos remotos operando con bajo consumo y de bajo costo. También funciona bajo el estándar 802.15.4 [23]

En la siguiente Tabla II.2 se muestra un resumen de las diferentes series de módulos *XBee*.

	XBee Series 1	XBee Series 2	XBee 900
Indoor/Urban range	up to 100 ft. (30m)	up to 133 ft. (40m)	up to 1200 ft. (370m)
Outdoor RF line-of-sight range	up to 300 ft. (100 m)	up to 400 ft. (120 m)	up to 15 miles. (24 km)
Transmit Power Output	1 mW (0dbm)	2 mW (+3dbm)	up to 693 mW (+28.4dbm)
RF Data Rate	250 Kbps	250 Kbps	230 Kbps
Receiver Sensitivity	-92dbm (1% PER)	-98dbm (1% PER)	-106dbm (1% PER)
Supply Voltage	2.8 - 3.4 V	2.8 - 3.6 V	3.0 - 3.6 V
Transmit Current (typical)	45 mA (@ 3.3 V)	40 mA (@ 3.3 V)	265 mA (@ 3.3 V)
Idle/Receive Current (typical)	50 mA (@ 3.3 V)	40 mA (@ 3.3 V)	80 mA (@ 3.3 V)
Power-down Current	10 uA	1 uA	60 uA
Frequency	ISM 2.4 GHz	ISM 2.4 GHz	ISM 900 MHz
Dimensions	0.0960" x 1.087"	0.0960" x 1.087"	0.0960" x 1.087"
Operating Temperature	-40 to 85 C	-40 to 85 C	-40 to 85 C
Antenna Options	Chip, Integrated Whip, U.FL	Chip, Integrated Whip, U.FL, RPSMA	Chip, Integrated Whip, U.FL, RPSMA

Tabla II.2 Series de los módulos XBee.
Fuente: http://www.xbee.cl/diferencias.html

2.4.4.2. Tipo de Antenas módulos XBEE

Los *XBee* necesitan antenas para recibir y transmitir señales, y existen varios tipos de antenas que componen un módulo *XBee* las cuales se pueden visualizar en la Figura II.20, entre las cuales se tienen:

- **Antena Chip:** Esta antena básicamente es un pequeño chip es rápido, sencillo y barato. La desventaja es que tiene una radiación cardiode es decir en forma de corazón lo que significa que la señal se atenúa en muchas direcciones.
- **Antena Whip:** Es un pequeño cable que sobresale del módulo su ventaja es que ofrece una radiación omnidireccional y su desventaja es que es frágil.
- **Antena U.FL:** Tiene un conector pequeño para conectar tu propia antena. Esto es perfecto si tienes tu equipo en una caja y deseas la antena afuera de ésta.
- **Antena RPSMA:** Este conector es una variación del conector U.FL ya que es más grande para conectar una antena externa.
- **ANTENA PCB:** Este tipo de antena fue introducido con el modelo *XBee-PROS2B,* la antena PCB es impresa directamente en el circuito de la tarjeta del módulo XBee.

Figura II.20: Tipo de Antenas XBee.

Fuente: http://1.bp.blogspot.com/-xi5iEYpaFGc/T521PZmYumI/AAAAAAAAApA/Zhxb_JBIHtc/s320/antenas.png

2.4.5. Componentes Hardware Adicionales

Es necesario conocer algunos hardware adicionales que nos ayudará en el proyecto como son: *XBee Explorer, XBee Explorer Regulado, XBee shield, XBee breakouts.*

- ***XBee Explorer***

El *XBee Explorer* nos permite conectar y utilizar cualquier módulo *XBee* directamente mediante un puerto USB ya que es muy sencillo como conectar un módulo *XBee,* es decir hay que pinchar un cable mini USB al PC y tendremos acceso a los pines TX/RX del *XBee* y estará listo para funcionar.

Es ideal para establecer una base inalámbrica desde un ordenador y así poder conectar sin cables a una placa que utilice un módulo *XBee.* En la Figura II.21 podemos observar el *XBee Explorer.*

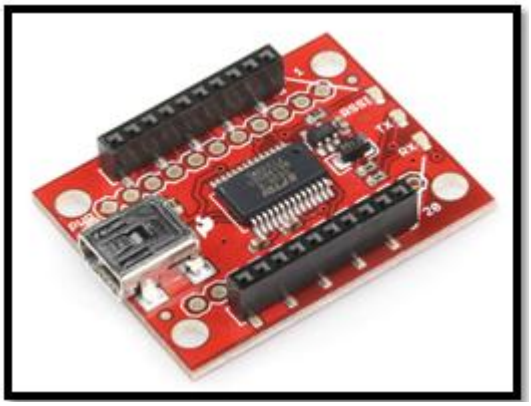


Figura II.21: XBee Explorer.

Fuente: https://www.sparkfun.com/products/8687

- ***XBee Explorer Regulado***

El *Explorador de XBee Regulado* es una placa que nos permite utilizar cualquier módulo XBee sin preocuparse por su regulación, porque dichos módulos funcionan a 3.3V. Tienen incorporado un regulador de tensión por lo que podemos conectar cualquier micro controlador a 5V.

Convierte las señales de 5V a 3.3V y está diseñada para ser conectada directamente con los Arduino. El regulador integrado proporciona un máximo de 150mA en la Figura II.22 se muestra la placa *XBee Explorer Regulado*.

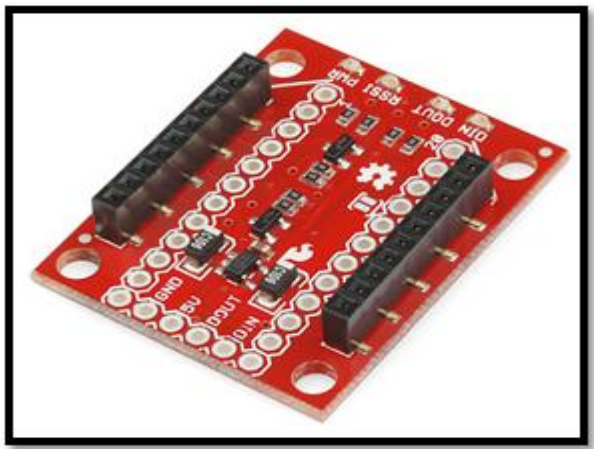


Figura II.22: XBee Explorer Regulado.
Fuente: http://www.bricogeek.com/shop/155-placa-xbee-explorer-regulada.html

- ***XBee shield***

La tarjeta *shield* para *XBee* simplifica la tarea de interconectar entre un *XBee* y un Arduino. Esta tarjeta se monta directamente sobre un Arduino y le agrega las características de comunicación inalámbrica del popular módulo *XBee*. Esta tarjeta funciona con todas las series de XBee incluyendo Serie 1 y Series 2.5, en versiones estándar y pro[24].

La tarjeta también se encarga del cambio de voltaje en el pin DIN del XBee e incluye leds para indicar que está encendida y El Botón de reinicio del Arduino se conecta con el shield. En la Figura II.23 se muestra la tarjeta *shield* para *XBee.*

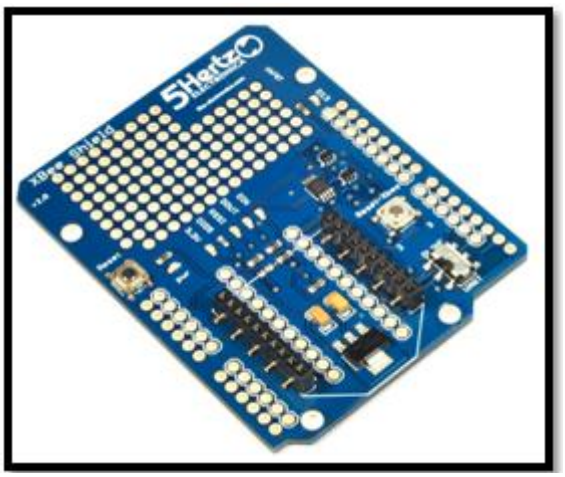


Figura II.23: XBee Shield.

Fuente:http://5hertz.com/index.php?main_page=product_info&products_id=496

- ***XBee Breakout***

Esto es ideal para conectar al protoboard para evitar el molestoso espaciado de 2mm del *XBee* al espaciado adecuado del protoboard. En la Figura II.24 se muestra el XBee Breakouts.

Figura II.24: XBee Breakout.
Fuente: http://www.olimex.cl/images/tutorial-xbee/header2mm.jpg

2.4.6. Software X-CTU

Este es el programa con el que se configuran los módulos *XBee*, también contiene un terminal con el cual poder mandar y recibir datos mediante el puerto que está conectado el *XBee y* sólo está disponible en Windows. En la Figura II.25 se observa la ventana del software X-CTU.

Figura II.25: Software X-CTU.

Fuente:http://cs.smith.edu/dftwiki/index.php/Tutorial:_Arduino_and_XBee_Communication

2.4.7. Tipos de Redes XBee

Los módulos *XBee*, pueden ser configurados para poder utilizarse en redes de configuración punto-a-punto o punto-a-multipunto. Teniendo en cuenta que los módulos *XBee* Serie 2 permiten hacer redes *mesh*, y la serie 1 no.

2.4.7.1. Conexión Punto a Punto

La conexión punto a punto se debe configurar la dirección. Para ello se utilizan los comandos MY y el DL. La idea, es que se define arbitrariamente una dirección para un módulo, usando el comando MY, el cual se va a comunicar con otro que tiene la dirección DL, también definida arbitrariamente. En la siguiente Figura II.26 se muestra una red punto a punto.

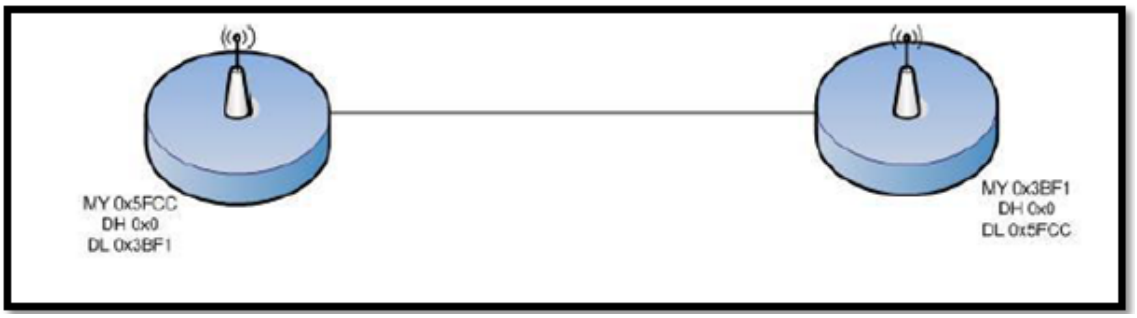


Figura II.26: Red punto a punto.

Fuente:http://rinconelectronico.bligoo.com.co/redes-xbee#.U6nRutLuL1Y

2.4.7.2. Conexión Punto a Multipunto

La conexión multipunto nos permite transmitir información desde la entrada serie de un módulo a uno o varios módulos conectados a la misma red de manera más controlada ya que se necesitan las direcciones de los otros módulos, por lo que existe mayor seguridad. Se necesitan dos comandos más aparte de MY y DL.

El primer comando es el ID de la PAN (Red de Área Personal). Todos los módulos que tengan idéntico PAN ID, pertenecerán a la misma red. El comando para configurar este valor es ID, es decir, ATID, y su rango va entre 0x0 y 0xFFFF. En la Figura II.27 representa un tipo de red punto a multipunto y como se debe configurar.

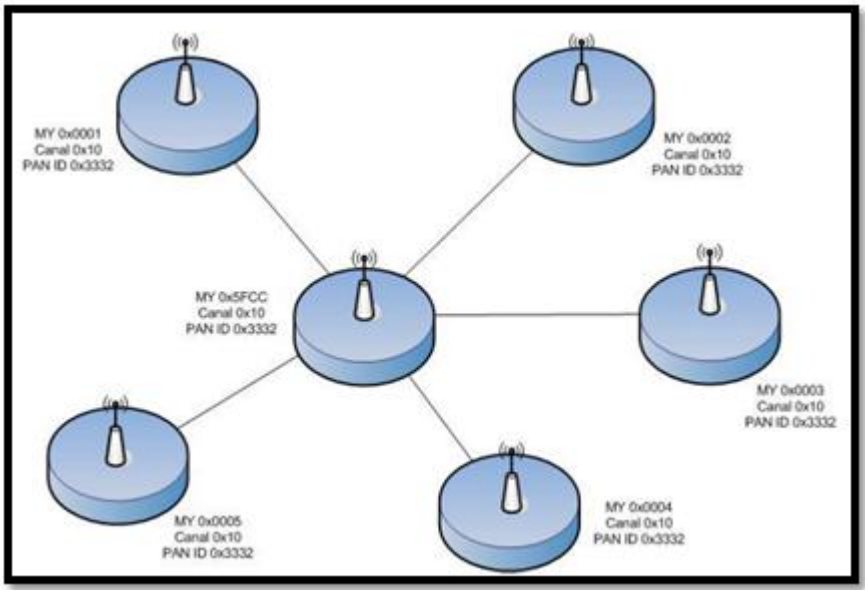


Figura II.27: Red punto a multipunto.

Fuente:http://rinconelectronico.bligoo.com.co/redes-xbee#.U6nRutLuL1Y

2.4.7.3. Conexión NonBeacon Peer to Peer

Una red *peer-to-peer* permite que todos los módulos, se conecten con todos, es decir, se crea una conexión de par en par con cada uno de los módulos de la red. El modo de conexión *NonBeacon* es la configuración por defecto y permite establecer una red *peer-to-peer* donde cada módulo puede hacer las funciones de maestro o esclavo.

La configuración de red *Non-Beacon*, se refiere a que cada nodo se mantiene despierto siempre. Por lo que los demás dispositivos que se conectan a él, pueden entrar en modo *Sleep* es decir ahorro de energía, y sólo se despertará cuando sea necesario para enviar datos. En una red *Beacon*, los dispositivos enrutadores están siempre en modo *Sleep*, y envían señales llamadas Beacon cada ciertos intervalos de tiempo al resto de los nodos de red. Así para poder comunicarse, deben estar totalmente organizados todos los dispositivos, ya

que de no ser así, existe la posibilidad de perderse la señal *Beacon* y no poder enviar hasta la próxima entrega.

Las redes *Beacon*, ahorran energía, mientras que las redes *Non-Beacon* están pensadas para dispositivos que posean una alimentación segura. Los módulos *XBee* Series 1, sólo soportan redes *Non-Beacon,* cada módulo se debe configurar como dispositivo terminal (*End Device*) y todos deben tener el mismo canal (ATCH) y la misma PAN (ATID). Para configurar los módulos como dispositivos terminales, se debe ingresar el comando CE=0 (ATCE0).

2.4.7.4. Conexión Non-Beacon con coordinador

Es básicamente lo mismo que una red punto-multipunto, con la diferencia de que existe un módulo central que posee ciertas propiedades y características que le permiten administrar mejor la red. En esta red, el módulo central es llamado Coordinador, mientras que el resto de módulos son llamados Dispositivos Terminales**.** Un mismo módulo *XBee* puede ser configurado para funcionar como Coordinador o como Dispositivo Terminal.

Para configurar esta red, todos los módulos deben tener el mismo canal ATCH y la misma PAN (ATID). El módulo Coordinador se configura como ATCE=1 (ATCE1), mientras que todos los demás, los cuales serán llamados Dispositivos Terminales, se configuran como ATCE=0 (ATCE0).

En este tipo de red, los Coordinadores pueden ser usados para usar transmisiones directas o indirectas. En las directas, la información es enviada de inmediato, mientras que en la indirecta, la información es guardada un tiempo dado por el parámetro SP (Cyclic Sleep Period). Este tipo de

configuración con un Coordinador se utiliza cuando se requiere una unidad central para enviar mensajes a varios módulos, o juntar información proveniente de varios Dispositivos Terminales, asignar canales o ID de redes PAN.

Una red de datos RF consistente de un Coordinador y uno o varios dispositivos terminales, forman lo que se llama una PAN es decir cada dispositivo en una PAN tiene un identificador llamado ID (ATID), el cual debe ser el mismo para todos los módulos de la misma PAN. En la siguiente Figura II.28 representa varias redes PAN NonBeacon con Coordinador.

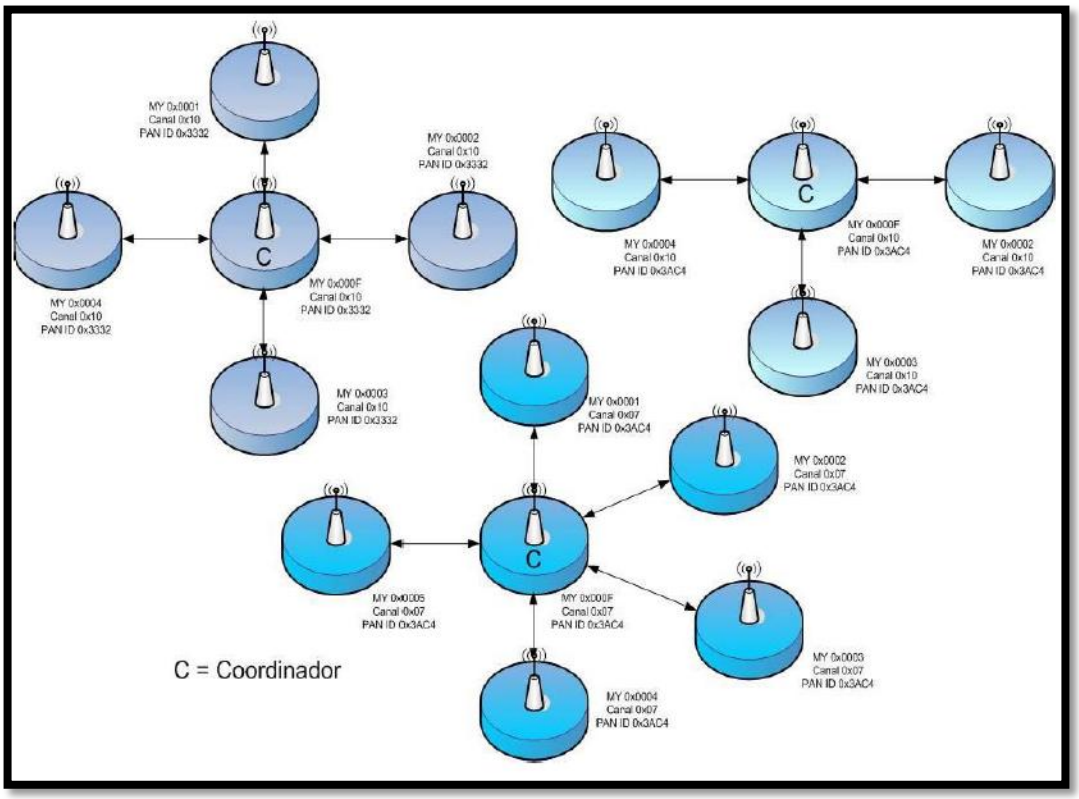


Figura II.28: Ejemplo varias redes PAN NonBeacon con Coordinador.

Fuente:http://rinconelectronico.bligoo.com.co/redes-xbee#.U6nRutLuL1Y

2.5. MÓDULOS OPEN HARDWARE ARDUINO

Arduino es una plataforma de electrónica abierta, es decir que utiliza hardware libre para la creación de prototipos basada en software y hardware flexibles y fáciles de usar, su funcionamiento se basa en una placa con un micro controlador y un entorno de desarrollo.

Utilizando Arduino se puede obtener información del medio a través de sus pines de entrada ya que estos son compatibles con una gran gama de sensores, permitiendo controlar multitud de tipos de luces, motores y otros actuadores físicos. Para ejecutar los proyectos hechos con Arduino no es necesario conectar a un ordenador, aunque existe la posibilidad de comunicar con diferentes tipos de software (Flash, Processing, MaxMSP).

Las placas se las puede armar o comprar montadas de fábrica; el software está disponible para ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta. Para programar el micro controlador de la placa Arduino se usa el lenguaje de programación Arduino (basado en *Wiring*) y el entorno de desarrollo Arduino. El lenguaje de programación de Arduino es una implementación de *Wiring*, una plataforma de computación física parecida, que a su vez se basa en *Processing*, un entorno de programación multimedia [25].

2.5.1. Antecedentes

El proyecto Arduino se inició en el año 2005 para los estudiantes del Instituto IVREA, en Ivrea (Italia) los cuales usaban el micro controlador BASIC Stamp,

cuyo valor no superaba los 100 dólares estadounidenses, lo que se consideraba un precio muy alto para los estudiantes. Uno de los fundadores de Arduino, MassimoBanzi, daba clases en Ivrea.

Una vez concluida dicha plataforma, sus investigadores trabajaron con el fin de hacerlo más ligero, más económico y disponible para la comunidad de código abierto (hardware y código abierto).En la primera producción en serie se tomó en cuenta que el coste no fuera mayor de 30 euros, que fuera ensamblado en una placa de color azul como se ve en la figura x, debía ser Plug and Play y que trabajara con todas las plataformas informáticas tales como MacOSX, Windows y GNU/Linux. Las primeras 300 unidades se las dieron a los alumnos del Instituto IVRAE, con el fin de que las probaran y empezaran a diseñar sus primeros prototipos[26].En la Figura II.29 se muestra un Arduino.

Figura II.29: Arduino.

Fuente: http://arduino.cc/es/uploads/Main/arduino_uno_test.jpg

2.5.2. Arduino

Además de simplificar el proceso de trabajar con micro controladores, Arduino ofrece algunas ventajas extra respecto a otros sistemas orientados a profesores, estudiantes y demás usuarios:

Accesible - Arduino tiene un menor costo comparado con otras plataformas de micro controladores, la versión más cara de un módulo de Arduino puede ser armada a mano y su precio sigue siendo bajo.

Multi-Plataforma - El software de Arduino funciona con los principales sistemas operativos como Windows, Macintosh OSX y Linux, los entornos de los demás microcontroladores están limitados solamente a Windows.

Entorno de programación simple y directo - El entorno de programación de es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados.

Software ampliable y de código abierto- El software utilizado por Arduino es libre y puede ser ampliado por programadores avanzados a través de librerías de C++.

Hardware ampliable y de Código abierto - Los planos de los módulos están publicados bajo licencia *Creative Commons*, por lo que usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo y así entender su funcionamiento.

2.5.3. Tipos De Placas Arduino

A continuación se presentara los distintos tipos de placas Arduino existentes como se muestra en la Figura II.30.

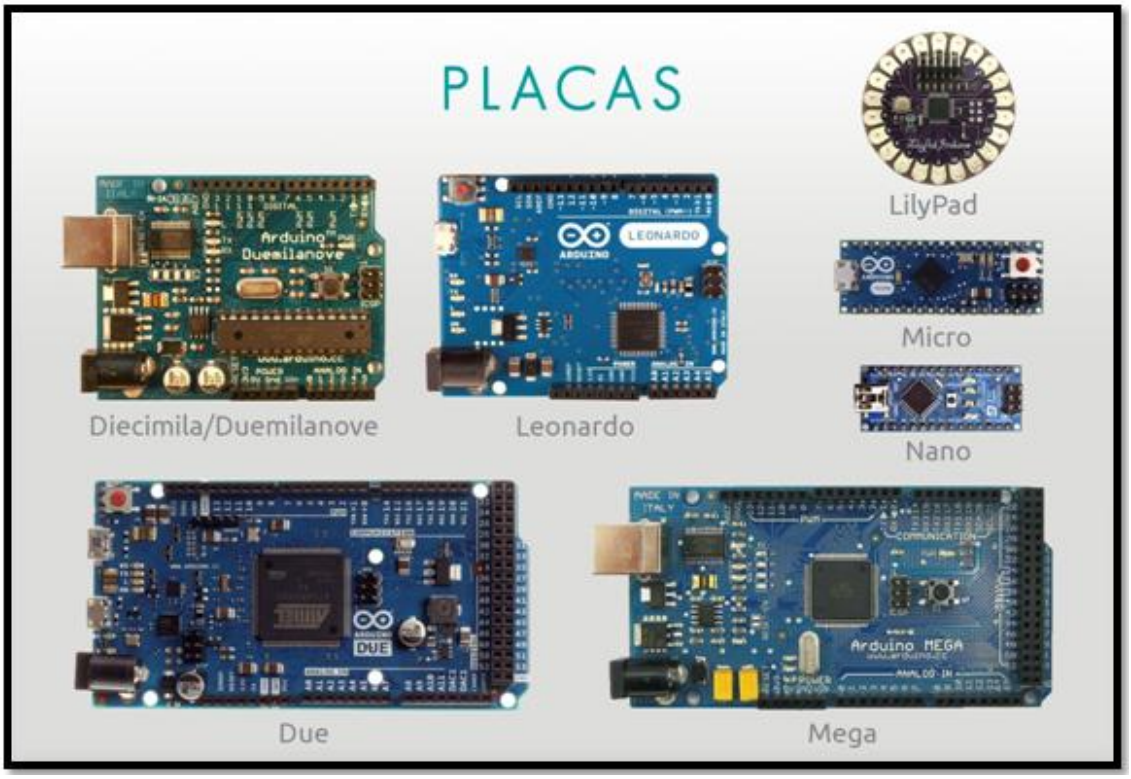


Figura II.30: Tipos de placas Arduino.

Fuente:http://www.davidmiguel.com/arduino/wp-content/uploads/2013/03/4.jpg

Arduino Mega: Es el micro controlador más capaz de la familia Arduino. Posee 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un botón de reset y una entrada para la alimentación de la placa. Se comunica con la computadora a través del puerto serial, posee un convertidor USB-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB como el que utilizan las impresoras.

Arduino Uno: Es la placa Arduino estándar y con mayor información disponible. Es la sucesora de la placa *Duemilanove,* incluye varias mejoras de hardware con respecto a su predecesor que consisten básicamente en el uso de un USB HID propio en lugar de utilizar un conversor FTDI para la conexión

USB. Es compatible con los modelos *Duemilanove* y *Diecimila*. Trabaja con un micro controlador Atmega328 con 32Kbytes de ROM para el programa.

Arduino Nano: Es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x) que funciona conectándola a una protoboard. Funciona con un cable USB Mini-B en vez del cable estándar y no posee un conector para la alimentación externa.

Arduino Bluetooth: Este módulo *Bluetooth Shield* integra un módulo *Bluetooth* de serie para la transmisión de datos de hasta 100 metros sin obstáculos, con esta placa se puede programar sin cables así como también realizar comunicaciones seriales sin cables con cualquier dispositivo *Bluetooth*.

Arduino Pro Mini: El Arduino Pro Mini fue creado para usuarios que necesitan flexibilidad, precios bajos y un tamaño reducido. Está equipada con los componentes mínimos es decir sin conector USB integrado ni conectores pin, con el fin de mantener su precio. Es ideal para proyectos que requieran dejar las placas integradas.

Arduino Mini: Es la versión más pequeña de Arduino, mide tan sólo 30x18mm permitiendo ahorrar espacio. Posee las mismas características que el Arduino UNO con la única diferencia que se lo fabrica con el chip Atmega168 con 12Kb de memoria para el programa.

2.5.4. Estructura básica de un programa

La estructura básica de programación de Arduino es bastante simple y divide la ejecución en dos partes: *setup* y *loop*. *Setup()* constituye la preparación del

programa y *loop()* es la ejecución. En la función *Setup()* se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar el pinMode (p. ej. si un determinado pin digital es de entrada o salida) e inicializar la comunicación serie. La función *loop()* incluye el código a ser ejecutado continuamente (leyendo las entradas de la placa, salidas, etc.).

```
void setup() {
   inMode(pin, OUTPUT);   // Establece 'pin' como salida
}
void loop() {
   digitalWrite(pin, HIGH);     // Activa 'pin'
   delay(1000);           // Pausa un segundo
   digitalWrite(pin, LOW);      // Desactiva 'pin'
   delay(1000);
}
```

Como se observa en este bloque de código cada instrucción acaba con; y los comentarios se indican con //. Al igual que en C se pueden introducir bloques de comentarios con /* ... */ [27].

2.5.5. Funciones

Una función es un bloque de código que tiene un nombre y un conjunto de instrucciones que se ejecutaran al llamar a la función. Las funciones se declararan asociándolas a un tipo de valor "*type*". Después de la declaración del tipo de dato para la función se debe escribir el nombre de la función y entre paréntesis se escribirán los parámetros que se deben pasar a la función para que esta se ejecute.

2.5.5.1. Funciones Básicas

- **Función pinMode(pin, Mode)**

Se la usa para configurar un determinado pin para comportarse como INPUT u OUTPUT. En Arduino los pines funcionan por defecto como entradas, de forma que no necesitan declararse explícitamente como entradas empleando pinMode ().

- **Función digitalRead(pin)**

Leerá un valor desde un pin digital específico que puede ser especificado con una variable o una constante y devolverá un valor HIGH o LOW.

- **Función digitalWrite(pin, value)**

Con esta función se puede introducir un nivel alto (HIGH) o bajo (LOW) en un pin digital especificado.

- **Función analogRead(pin)**

Esta función solo funciona en los pines analógicos (0-5) y leerá el valor desde el pin analógico especificado, como resultado se obtendrá un valor entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT u OUTPUT.

- **Función analogWrite(pin, value)**

Escribe un valor analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM, puede especificarse un valor de 0 – 255.

Un valor 0 genera 0 V en el pin especificado y 255 genera 5 V. Para valores de 0 a 255, el pin tendrá una variación de voltaje entre 0 V y 5 V, cuanto mayor sea el valor, más a menudo el pin se encuentra en HIGH (5 V).

2.5.5.2. Funciones de Tiempo y Matemáticas

- **delay(ms)** Realiza una pausa en el programa el tiempo especificado en milisegundos (Max 1000, min 1).

- **millis()** Indica la cantidad de milisegundos que el programa actual se está ejecutando en la placa Arduino.
- **min(x,y),max(x,y)** Devuelve el un valor mínimo y máximo respectivamente de entre sus parámetros.

2.5.5.3. Funciones Para el Puerto Serie

- **Serial.begin(rate)**

Permite abrir un puerto serie y especificar la velocidad de transmisión. La velocidad comúnmente utilizada para comunicación con el puerto de la computadora es de 9600.

- **Serial.println(data)**

Se lo puede utilizar para realizar la depuración de los datos, además imprime datos al puerto serie seguido por un retorno de línea automático.

- **Serial.read().**

Lee o captura un byte desde el puerto serie.

- **Serial.available().**

Devuelve el número de caracteres disponibles para leer desde el puerto serie.

2.6. PROTOCOLOS DE COMUNICACIÓN INALÁMBRICA

2.6.1. Bluetooth

Bluetooth es una tecnología de red de área personal inalámbrica de corto alcance, que se utiliza para conectar dispositivos entre sí sin una conexión por cable. Los dispositivos Bluetooth no necesitan una línea de visualización directa para comunicarse, lo que permite la comunicación entre dispositivos en espacios pequeños.

El objetivo de Bluetooth es transmitir voz o datos entre equipos con circuitos de radio de bajo costo, utilizando poca energía. Bluetooth se diseñó principalmente para conectar dispositivos como impresoras, teléfonos móviles, artículos para el hogar, auriculares inalámbricos, ratón, teclados, etc. También se utiliza cada vez más en teléfonos móviles, lo cual les permite comunicarse con equipos y se ha extendido especialmente a los accesorios manos libres, como los auriculares Bluetooth.

2.6.2. ZigBee

ZigBee es una especificación que define una solución para comunicaciones inalámbricas de bajo coste y consumo. ZigBee se basa en el nivel físico y el control de acceso al medio definidos en la versión estándar IEEE 802.15.4, que desarrolla estos niveles para las redes áreas personales de baja tasa de transferencia.

ZigBee es una arquitectura de red en malla, soporta tres topologías distintas: por una parte, patrones de redes típicas en estrella y árbol, así como redes genéricas en malla. Toda red necesita un dispositivo coordinador encargado de su creación. En redes en estrella, el coordinador debe ser el nodo central y en las redes en árbol y malla permiten el uso de Routers ZigBee para habilitar la comunicación en el nivel de red; éstos no son coordinadores ZigBee, pero pueden serlo de sus respectivos espacios de operación personal definidos por 802.15.4.

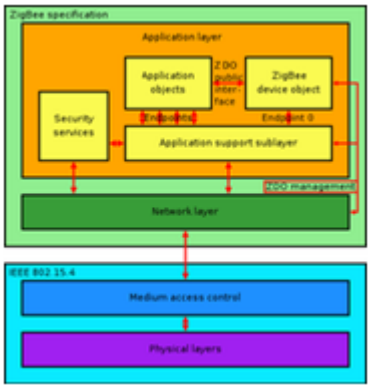


Figura II.31: Pila de protocolos ZigBee.

Fuente:http://upload.wikimedia.org/wikipedia/commons/thumb/4/4f/ZigBee_protocol_stack.png/220px-ZigBee_protocol_stack.png

2.6.3. Comparación

Un estudio realizado en Málaga [28] donde se compara los protocolos de comunicación inalámbrica. Para el proyecto de tesis se comparó las siguientes características como se muestra en la tabla II.3.

Comparación		
Características	ZigBee	Bluetooth
Consumo TX	30 mA	40 mA
Consumo Reposo	3 uA	200 uA
Numero de nodos	65535	8
Arquitectura	Estrella, árbol, malla y punto a punto.	Estrella
Ancho de banda	1 MHz	0,3 a 0,6 MHz

Tabla II.3: Comparación de protocolos de comunicación.
Fuente: Elaboración Propia

Como se diferencia el ZigBee consume un 25% menos de energía que BlueTooth en modo de transmisión, mientras que un 98,5% menos en modo reposo, ya que para el proyecto de tesis cada cierto intervalo de tiempo

permanecerá en reposo, el protocolo de comunicación inalámbrica que mejor se ajusta a los requerimientos del sistema es la tecnología ZigBee.

2.7. ESTANDAR IEEE 802.15.4 Y ZIGBEE

El estándar IEEE 802.15.4 que operan en bandas sin licencia en todo el mundo a 2,4 GHz (global), 915Mhz (América) y 868Mhz (Europa) es la base de *ZigBee*. Ofrece velocidades de datos de rendimiento de 250Kbs a 2,4 GHz (16 canales), 40Kbs a 915Mhz (10 canales) y 20Kbs a 868Mhz (1 canal), ofreciendo distancias de 10 a 1.600 metros, dependiendo de la potencia y el medio en el que será utilizado.

La comunicación se la realiza a través de señales de radio, utilizando las normas del estándar IEEE 802.15.4 que fija las condiciones para que se produzca el enlace y sirve de base para otras especificaciones como *ZigBee* cuyo propósito es ofrecer una solución completa para este tipo de redes definiendo los niveles superiores de la pila de protocolos que el estándar 802.15.4 no cubre.

2.7.1. Estándar IEEE 802.15.4

Este estándar fue creado para satisfacer las necesidades en el campo de los estándares inalámbricos de baja tasa para aplicaciones en redes de sensores, ya que otros, como *Bluetooth* o *WIFI* no cumplen con los requerimientos necesarios para este tipo de aplicaciones.

Define una capa de comunicación que se encuentra en el nivel 2 del modelo OSI. La información digital es gestionada y organizada para su conversión en

impulsos electromagnéticos en el nivel inferior, el físico, y permitir la comunicación entre dos dispositivos. Este estándar se caracteriza por su flexibilidad de red, bajo coste, bajo consumo de energía.

Su bajo consumo de potencia es una principal ventaja del protocolo IEEE 802.15.4. Este bajo consumo energética se debe al uso de las tramas "*Beacon*", que permiten sincronizar los dispositivos de la red para que puedan permanecer en modo ahorro de energía el mayor tiempo posible, suponiendo una gran ventaja para las redes de sensores inalámbricos que realicen tanto tareas de monitorización como de control. El principal inconveniente es que, debido a su bajo consumo de energía, el radio de cobertura es reducido.

2.7.2. Estándar ZigBee

ZigBee es un protocolo de comunicaciones inalámbrico basado en el estándar de comunicaciones para redes inalámbricas IEEE 802.15.4. Fue creado por la ZigBee Alliance, una organización de más de 200grandes empresas [22] que permite que dispositivos electrónicos de bajo consumo se puedan comunicar de forma inalámbrica. Su principal utilidad son las redes de sensores en entornos industriales, médicos y, sobre todo domóticas. ZigBee utiliza la banda libre de 2.4GHz, utilizando un solo canal de los 16 disponibles.

El alcance dependerá de la potencia de transmisión del dispositivo así como también del tipo de antenas utilizadas. El alcance normal con antena dipolo en línea vista tomando en cuenta módulos con antenas de 1mW de potencia es de aproximadamente 100m y en interiores de unos 30m. La velocidad de transmisión de datos de una red ZigBee puede alcanzar hasta los 256kbps.

Una red ZigBee puede estar formada de hasta 65535 equipos, debido a las características ofrecidas por este tipo de red. Entre las necesidades que satisface el módulo se encuentran:

- Bajo costo.
- Ultra-bajo consumo de potencia.
- Uso de bandas de radio libres y sin necesidad de licencias.
- Instalación barata y simple.
- Redes flexibles y extensibles.

El uso del protocolo *ZigBee* reemplaza un cable por una comunicación serial inalámbrica, se puede realizar configuraciones punto a punto, multipunto*, peer-to-pee, mesh.* Entre los beneficios al usar estándar *ZigBee* en los productos y servicios tenemos:

- La interoperabilidad de dispositivos estandarizados permite a los fabricantes de productos centrarse en la innovación de productos, dejando de lado la creación de sus propios protocolos de red propietarios
- Al tener un protocolo estandarizado los fabricantes de productos pueden actualizar los productos existentes y añadir nuevas características innovadoras.

2.7.2.1. Características De ZigBee

Características de las redes/dispositivos *ZigBee* serían las siguientes:

- Velocidad de transmisión entre 25-250 Kbps.
- Protocolo asíncrono, *half dúplex* y estandarizado, permitiendo a productos de distintos fabricantes trabajar juntos.

- Se pueden formar redes que contengan desde dos dispositivos hasta cientos de ellos.
- Los dispositivos de estas redes pueden funcionar en un modo de bajo consumo, lo que supone años de duración de sus baterías.
- Opera en la frecuencia de 2.4 GHz (16 canales) y también en las frecuencias de 868 MHz y 915 MHz.
- Es un protocolo fiable, la red se organiza y se repara de forma automática y se rutean los paquetes de manera dinámica.
- Es un protocolo seguro ya que se puede implementar encriptación y autentificación [29].

2.7.2.2. Áreas De Aplicación

En la actualidad existen muchos campos en los que se utiliza este tipo de tecnología, actualmente un gran número de las compañías forman parte de la *ZigBee Alliance*, que se encuentran desarrollando productos que trabajen con *ZigBee*, entre las áreas más importantes están:

- Gestión de edificios comerciales
- Electrónica de consumo
- Gestión de la energía
- Cuidado de la salud y de la aptitud
- La administración del hogar
- Gestión de venta
- Telecomunicaciones

CAPÍTULO III

ANALISIS Y DISEÑO DEL PROTOTIPO DEL SISTEMA WSN Y RFID

3.1. INTRODUCCIÓN

En este capítulo se describirá el análisis y diseño para la construcción del prototipo del sistema WSN y RFID utilizando la tecnología que mejor se adapta a las necesidades del sistema, basándonos en los resultados obtenidos mediante el análisis de diferentes tecnologías para la transmisión de datos realizado en el capítulo anterior.

Se describirá las características de hardware de los diferentes componentes que forman el prototipo, tales como la unidad sensora y los componentes que la forman, la unidad de transmisión y recepción que será la encargada de que se realice la comunicación entre los nodos, la unidad de proceso que se encarga de interconectar la unidad sensora con los módulos de comunicación XBee, y la forma en que conjuntamente trabajan para que el prototipo funcione.

También se describirá el software utilizado y como fueron configurados los componentes utilizados, como se programaron los módulos de comunicación XBEE, la programación utilizada en la unidad de proceso Arduino UNO para que recolecte, procese y transmita la información hacia la unidad coordinadora, la cual recibirá los datos de cada uno de los nodos y los enviara hacia la base de datos en donde se procesará y almacenará la información lo que nos permitirá mantener un registro de la actividad en cada laboratorio, esta aplicación también permitirá mostrar datos y registros en tiempo real y así monitorear cada uno de los laboratorios.

3.2. ANÁLISIS DEL SISTEMA DE CONTROL DE ACCESO

3.2.1. Evaluación de las alternativas

En función de las necesidades requeridas para la autenticación del usuario, existen diferentes parámetros a tomar en cuenta para elegir un sistema en particular. Los parámetros que se utilizará para la comparación están basados en una tesis realizada en la Universidad de San Carlos [30] que para el requerimiento del sistema son los siguientes:

a.- La fiabilidad

b.- Facilidad de uso

c.- Rendimiento

d.- Mantenimiento

e.- Precio

La escala a utilizar para las comparaciones de los sistemas de control de acceso se basa en un estudio realizado por la Federal Office for Security de Alemania [31] estos parámetros se muestran en la siguiente tabla III.4.

escala			
Abreviatura	Escala	Peso	Porcentaje
E	Excelente	4	100%-76%
MB	Muy bueno	3	75%- 51%
B	Bueno	2	50%- 26%
M	Malo	1	25-1%
MM	Muy malo	0	0%

Tabla III.4Escala.
Fuente: Federal Office for Security de Alemania

3.2.1.1. Fiabilidad

Su función principal es la de autenticar de forma segura la identidad de las personas que pretenden tener acceso a un determinado servicio o lugar físico. La fiabilidad está directamente relacionada con las tasas de falso rechazo y de falsa aceptación. Al decir falso rechazo se refiere a la probabilidad de que el sistema utilizado para la autenticación rechace a un usuario legitimo al no ser

capaz de identificarlo correctamente, por tasa de falsa aceptación podemos decir que es la probabilidad que existe de que el sistema autorice a un usuario ilegitimo proporcionando acceso a personal no autorizado.

Sistema de control de acceso	Fiabilidad	Abreviatura	Peso
RFID-WSN	Es alta ya que este tipo de tarjetas poseen un código único de fábrica que nunca se repite y que será exclusivo para cada usuario.	E	4
Reconocimiento facial	Es alta, debido a que se registra diferentes características de la cara del usuario, sin embargo, se puede dar el caso de que existan dos caras con las mismas características, ó utilizar una fotografía del usuario provocando que el sistema falle.	MB	3
Reconocimiento huellas dactilares	La huella dactilar es un patrón fiable para determinar la identidad del usuario, pero si es copiado por alguien es un grave problema ya que esta no se la puede reemplazar o cambiar.	MB	3
Reconocimiento de geometría de la mano	Posee un alto grado de fiabilidad debido a que utiliza ciertos datos de la mano en tres dimensiones. Sin embargo, es posible, que dos o más personas tengan la mano lo suficientemente parecida en forma y tamaño como para que el sistema las confunda.	MB	3

Tabla III.5 Fiabilidad de los Sistemas.
Fuente: Federal Office for Security de Alemania

Como se demuestra en la Tabla III.5 todos los sistemas ofrecen un alto grado de fiabilidad y se lo puede visualizar mejor en la Figura III.32.

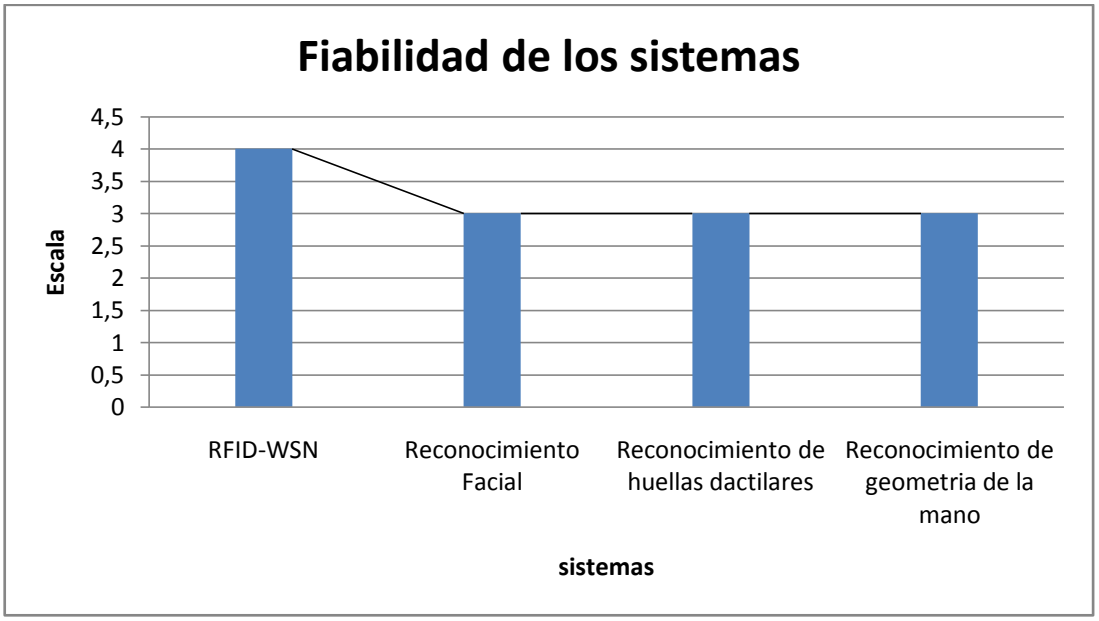


Figura III.32: Fiabilidad de los sistemas.
Fuente: Elaboración Propia

3.2.1.2. Facilidad de uso

Es de vital importancia ya que se debe tomar en cuenta que para su implementación y uso se requiere que los usuarios aprendan a utilizar los dispositivos, esto se puede ver afectado por la experiencia de los usuarios o por su poca relación con la tecnología.

Sistema de control de acceso	Facilidad de uso	Abreviatura	Peso
RFID-WSN	Es muy fácil de utilizar, ya que no es necesaria que la tarjeta sea pasada por una ranura o en un sentido específico.	E	4
Reconocimiento facial	Es necesario que el usuario se coloque frente a la cámara en la posición y dirección correcta. Su registro puede ser difícil ya que algunas personas tienen dificultad para alinear la cara.	B	2
Reconocimiento huellas dactilares	Es un método fácil de utilizar, ya que si el usuario desea autenticarse ante el sistema, solamente deberá colocar el dedo indicado en la posición indicada.	MB	3
Reconocimiento de geometría de la mano	Es fácil de utilizar, ya que para ser autenticado el usuario únicamente deberá colocar su mano sobre el dispositivo lector siguiendo unas guías que marcan la posición correcta para la lectura.	MB	3

Tabla III.6: Facilidad de uso de los Sistemas.
Fuente: Federal Office for Security de Alemania

Como se demuestra en la Tabla III.6 la facilidad de uso para los usuarios, el sistema más amigable es el RFID-WSN a diferencia que el sistema de reconocimiento facial puede llegar a ser difícil ya que algunas personas tienen la dificultad para alinear la cara en la posición correcta. En la siguiente Figura III.33 se muestra la facilidad de uso para los usuarios.

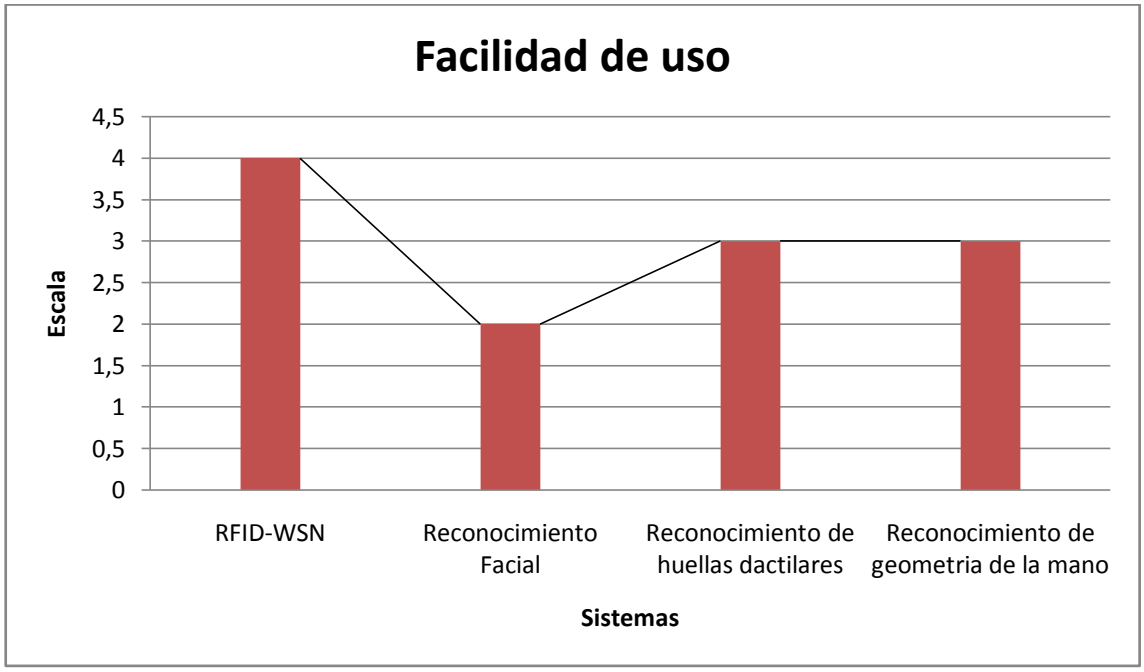


Figura III.33: Facilidad de uso de los sistemas.
Fuente: Elaboración Propia

3.2.1.3. Rendimiento

Nos referimos a cuan estable debe ser el medio de identificación, a la poca o nula variación que debe existir por parte del sistema. Entre las causas es el mal funcionamiento del sistema que principalmente son dos, los factores ambientales como el clima, ruido, la iluminación, suciedad, etc. Y las condiciones del miembro corporal utilizado para la identificación que se puede ver afectado por factores como envejecimiento, cortaduras, desgaste, etc. En la Tabla III.7 se muestra las características de cada sistema referente al Rendimiento.

Sistema de control de acceso	Rendimiento	Abreviatura	Peso
RFID-WSN	Posee una gran estabilidad ya que al realizarse la lectura del código este se realiza sin contacto mediante la transmisión de una onda de radio frecuencia, lo que significa que no tiene desgaste por rozamiento. Y al trabajar a una distancia de funcionamiento de unos pocos centímetros es casi imposible que se vea afectada por interferencias.	E	4
Reconocimiento facial	Su estabilidad se ve afectada debido a los cambios de los rasgos faciales, los cuales son muy comunes, estos se presentan debido a cambios en el cabello, la edad, cicatrices o deformidades físicas.	B	2
Reconocimiento huellas dactilares	Pueden tener una estabilidad alta, ya que las huellas dactilares de los usuarios no cambian con el paso del tiempo. Sin embargo, pueden presentarse problemas como suciedad, heridas o inclusive pérdida del miembro lo que podría provocar una menor estabilidad del sistema.	MB	3
Reconocimiento de geometría de la mano	Posee una estabilidad media debido a los cambios en la geometría de la mano que pueda sufrir el usuario, como por ejemplo el crecimiento de la mano que puede llegar a darse por el paso del tiempo, enfermedad u otros factores. Además, la pérdida del miembro podría afectar.	MB	3

Tabla III.7: Rendimiento de los Sistemas.
Fuente: Federal Office for Security de Alemania

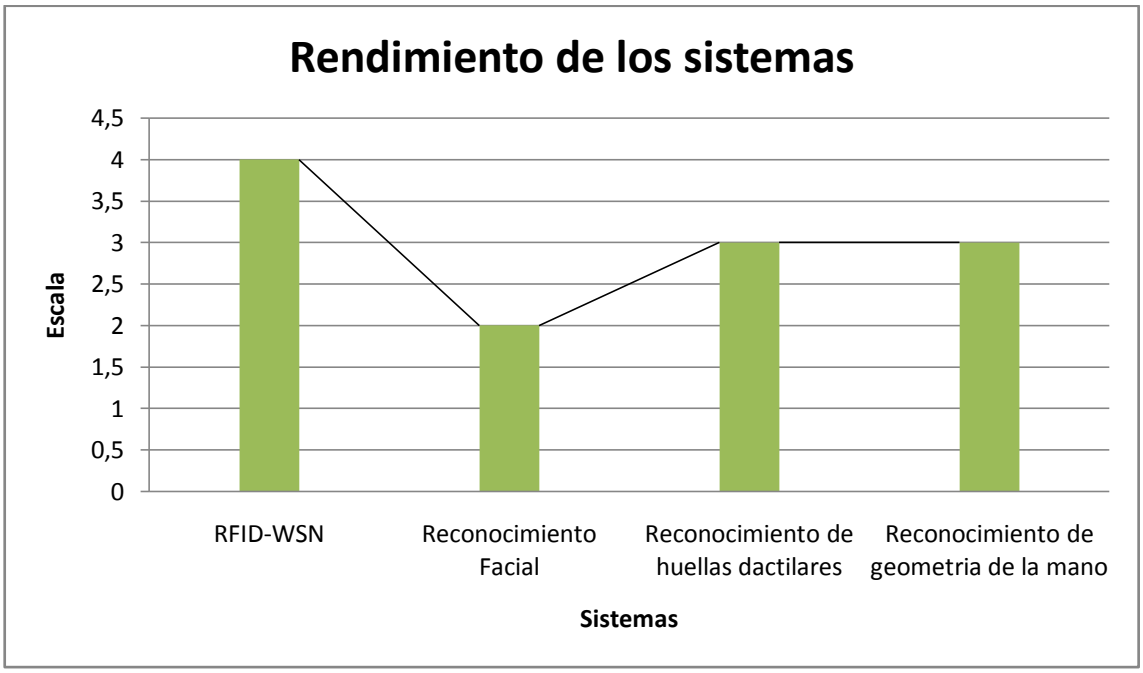


Figura III.34: Rendimiento de los sistemas.
Fuente: Elaboración Propia

Como se observa en la Figura III.34 el sistema RFID-WSN posee una gran estabilidad ya que para este sistema no existe interferencia por la poca distancia que se necesita para identificar la tarjeta.

3.2.1.4. Mantenimiento

Todos los equipos requieren un mantenimiento programado con el fin de garantizar su correcto funcionamiento y prevenir fallas a futuro evitando así contratiempos y garantizando su disponibilidad. La falta de mantenimiento o un mantenimiento inadecuado podría provocar un acceso no autorizado o la no identificación de los usuarios, es por eso que se ha tomado en cuenta este factor ya que influye en la calidad y correcto funcionamiento del sistema.

Sistema de control de acceso	Mantenimiento	Abreviatura	Peso
RFID-WSN	Los lectores al ser unidades selladas y sin partes móviles garantizan un funcionamiento correcto y sin un límite de uso. Pueden ser instalados en cualquier lugar, incluso a la intemperie y no se verán afectados por las inclemencias del tiempo, la lluvia o las altas y bajas temperaturas del ambiente.	MB	3
Reconocimiento facial	Es imprescindible darle mantenimiento a la cámara, lo cual es recomendable realizarlo 3 veces al año debido al polvo y la humedad que pueda almacenarse en la cámara lo que dificultara su correcto funcionamiento.	MB	3
Reconocimiento huellas dactilares	Se recomienda darle un mantenimiento unas 3 o 4 veces al año con el fin de eliminar el polvo y la suciedad que se acumulan en el lector. También es necesario que el teclado reciba mantenimiento.	MB	3
Reconocimiento de geometría de la mano	Las fallas más comunes en estos equipos se dan en el teclado al tener un movimiento mecánico, la base en la que se coloca la mano también puede llegar a presentar problemas debido a que puede desgastarse con el uso, es por esto que requiere de una limpieza general, en especial en la base donde se coloca la mano y los espejos.	MB	3

Tabla III.8: Mantenimiento de los Sistemas.
Fuente: Federal Office for Security de Alemania

Como se muestra en la Tabla III.8 todos los sistemas necesitan de un mantenimiento para su óptimo desempeño, en la Figura III.35 se muestra de mejor manera el mantenimiento necesario para cada sistema, de acuerdo a los pesos asignados.

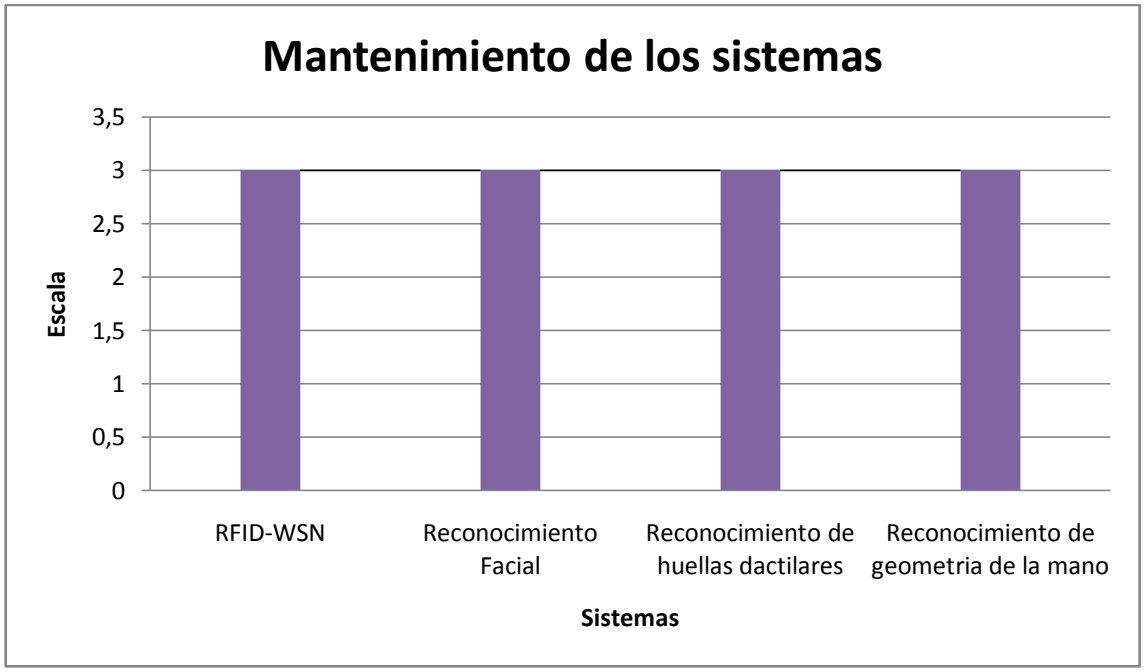


Figura III.35: Mantenimiento de los Sistemas.
Fuente: Elaboración Propia

3.2.1.5. Costo

El Costo es el factor más importante a ser tomado en cuenta antes de elegir o implementar un determinado equipo ya que este influye directamente en la elección de un sistema de control de acceso, debido a la inversión inicial requerida.

Sistema de control de acceso	Costo	Abreviatura	Peso
RFID-WSN	En la construcción del prototipo desarrollado se invertirá 85 USD tomando en cuenta que es totalmente independiente y envía la información de forma inalámbrica.	E	4
Reconocimiento facial	El precio de este tipo de sistemas cuesta alrededor de los 850 USD sin el costo de instalación.	B	2
Reconocimiento huellas dactilares	Es uno de los sistemas biométricos más baratos este tipo de sistemas se encuentra por los 700 USD	B	2
Reconocimiento de geometría de la mano	El precio de este lector tiene un costo desde 1900 USD.	M	1

Tabla III.9: Precio del Lector de los Sistemas.
Fuente: Elaboración Propia

En la tabla III.9 se hace referencia a los precios que actualmente ofrecen diferentes empresas en el país [32]. En la Figura III.36 se muestra que el costo para este sistema es muy bajo en comparación con otros sistemas, obteniendo una mejor relación costo beneficio.

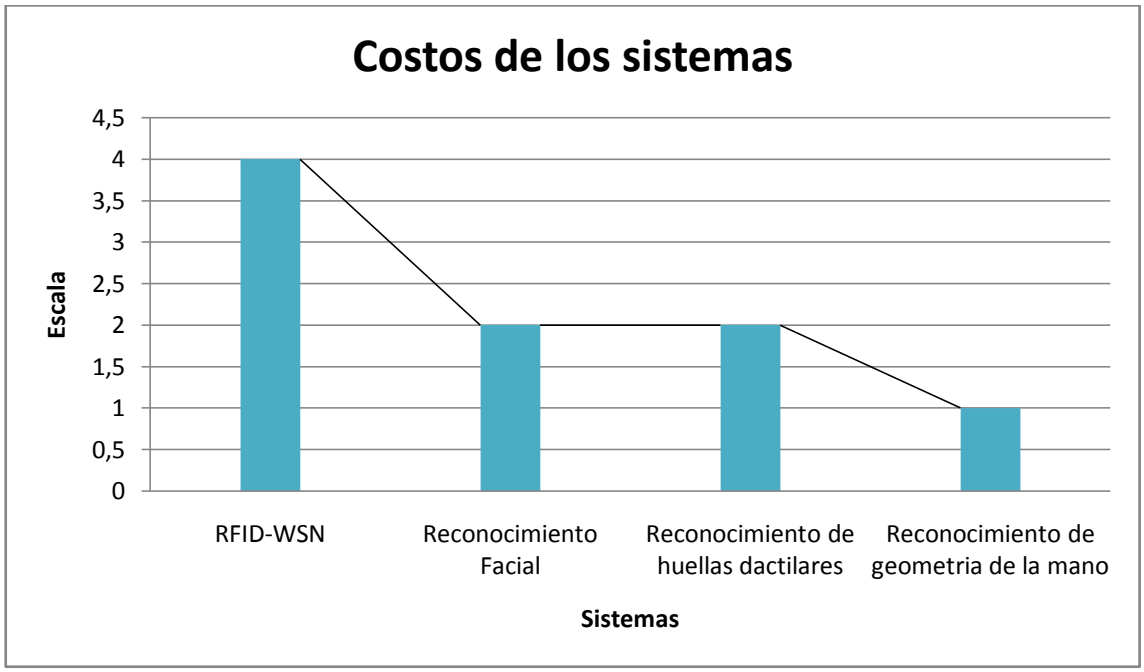


Figura III.36: Costos de los sistemas.
Fuente: Elaboración Propia

En la Tabla IV.10 se muestra una comparación entre los diferentes factores identificados para cada tecnología de control de acceso.

Factor de evaluación	RFID-WSN		Reconocimiento facial		Reconocimiento huellas dactilares		Reconocimiento de geometría de la mano	
Fiabilidad	E	4	E	3	MB	3	MB	3
Facilidad de uso	E	4	B	2	MB	3	MB	3
Rendimiento	E	4	B	2	MB	3	MB	3
Mantenimiento	MB	3	MB	3	MB	3	MB	3
Costo	E	4	B	2	B	2	M	1
Total		19		12		14		13

Tabla III.10: Comparación de sistemas de control de acceso.
Fuente: Elaboración Propia

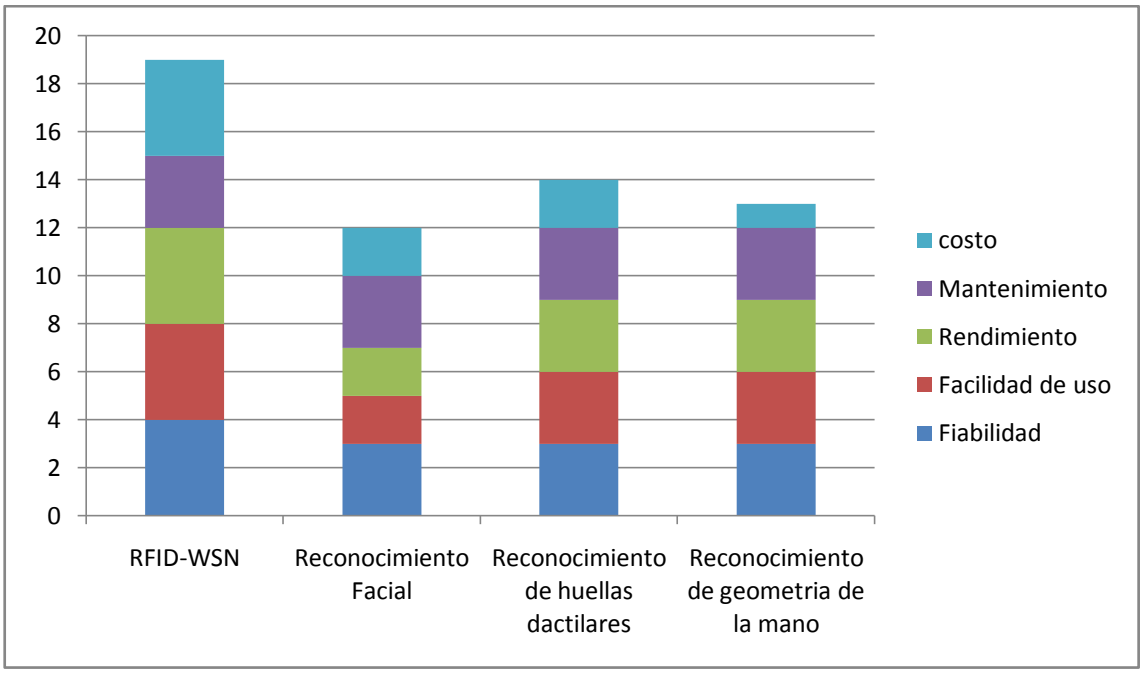


Figura III.37: Comparación de los sistemas.
Fuente: Elaboración Propia

Al concluir esta evaluación y con las escalas dadas en la Tabla III.10 para una escala del 0 al 4, donde el 4 es el máximo puntaje por cada factor evaluado y con un total de 20 puntos. El reconocimiento facial recibe 12 de 20 con un 60% de calificación. El reconocimiento de huellas dactilares recibe 14 con un 70% de calificación. El reconocimiento de geometría de la mano recibe 13 con un 65% de calificación. El RFID-WSN recibe 19 con un 95% de calificación

demostrando que el sistema RFID-WSN es el indicado para el control de acceso docente.

3.2.2. Tipo de Investigación

Se utilizara la investigación científica ya que por medio de ella se encaminara hacia los hechos para obtener por medio de ellos un conocimiento científico, además por medio de este proceso se tratara de conocer los elementos que intervienen en nuestro caso para obtener así las variables que se deberá tomar en cuenta como los docentes, los laboratorios, las materias y los diferentes horarios en cada laboratorio.

También se ha realizado una investigación aplicativa lo que nos ha permitido obtener conocimientos con el fin de aplicarlos inmediatamente y así poder modificar este sistema, es decir hallar una solución a un problema práctico más que formular teorías acerca de este.

3.2.3. Identificación de las variables

En nuestra investigación se pudo identificar 4 variables que podrán ser modificadas o manipuladas es por eso que se las ha identificado como variables independientes, las cuales se detallan a continuación:

- **Docente:** Esta variable nos permite conocer todos los docentes que están registrados y dictan sus horas de clase en los laboratorios en los que se aplicara el prototipo.
- **Laboratorio:** Por medio de esta variable se puede identificar cada laboratorio y así registrar la actividad en cada uno de estos.

- **Materia:** Nos permitirá saber todas las materias asignadas a los docentes, su código, duración y a que docente ha sido asignada.
- **Horario:** En esta variable almacenaremos la hora de inicio y fin de una materia y el día que le corresponde.

3.3. DISEÑO DEL PROTOTIPO WSN Y RFID

El diseño del prototipo de una red inalámbrica de sensores con la interoperabilidad con RFID, estará formada por dos nodos finales, un coordinador y la base de datos tal como se observa en la Figura III.31.

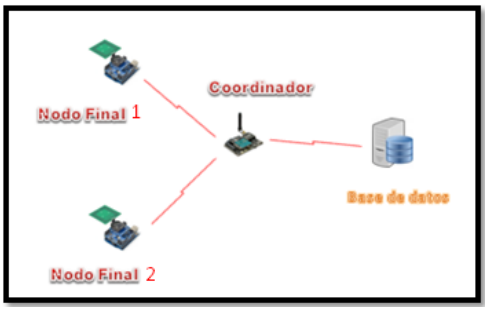


Figura III.31: Implementación del Prototipo.
Fuente: Elaboración Propia

3.3.1. Funcionamiento

En los nodos finales estará integrado por un sensor de lectura RFID que recogerá información emitida por una tarjeta o llavero RFID que tendrán cada uno de los docentes de la academia CISCO. Este sensor estará conectado a la placa base Arduino, la cual procesará la información y le agregara una cabecera a los datos para poder identificar la dirección de origen de cada paquete recibido, es decir que el coordinador los reciba a través de los módulos de comunicación *XBee* pueda identificar de que nodo provienen la información recibida.

El nodo coordinador está directamente conectado al computador, en la cual se registra la información ermitita por los nodos finales que estarán en los laboratorios de CISCO. Los datos receptados serán interpretados y guardados en una base de datos, con una interfaz amigable con el usuario se diseñará una aplicación, donde se pueden verificar la asistencia de los docentes a sus horas respectivas de clases, lo que permite que se monitoree y se tenga control de acceso de los docentes de la academia CISCO.

Se explica cómo están programados cada uno de los dispositivos para que se interconecten y recojan los datos desde los laboratorios y los trasladen hasta un computador que finalmente se almacene en una base de datos.

3.3.2. Nodo Final

Son llamados *End Device* que no tienen capacidad de enrutar paquetes. En este proyecto el nodo final está compuesto de cuatro componentes básicos, una unidad sensora la cual es RFID-RC522, una unidad de proceso la cual es el Arduino Uno, una unidad del transmisor-receptor en este caso es el *XBee s1*, y una unidad de potencia que es una batería de 9V, y adicionalmente se integró una pantalla LCD de 16x2 líneas tal como se ve en la Figura III.32.

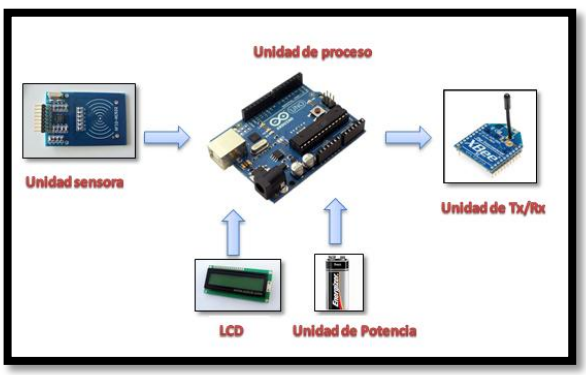


Figura III.32: Nodo Final.
Fuente: Elaboración Propia

La unidad sensora es la que recoge información emitida por una tarjeta RFID, la cual enviará hacia la unidad de proceso, una vez que los datos han sido procesados, la unidad de Transmisión-Recepción los enviará hacia el coordinador y después hacia la base de datos donde responderá y será visualizada en la pantalla LCD del nodo final. La unidad de potencia estará directamente conectada al Arduino y este a la vez alimentará a la unidad sensora, la unidad de Tx-Rx y a la pantalla LCD.

3.3.2.1. Sensor

En el prototipo se va a utilizar el sensor RC522 se utilizará para la comunicación inalámbrica a 13.56Mhz que sirve para leer datos de aplicaciones de bajo consumo de energía, bajo costo y tamaño reducido. Ideal para dispositivos portátiles o tarjetas, como se muestra en la Figura III.33.

3.3.2.1.1. Lector RFID

Como se puede apreciar en la Figura III.33 el modulo consta de 8 pines: SDA, SCK, MOSI, MISO, NC, GND, RST y 3.3V. El módulo utiliza 3.3V como voltaje de alimentación y se controla a través del protocolo *Serial Peripherical Interface* (SPI), por lo que es compatible con casi cualquier micro controlador.

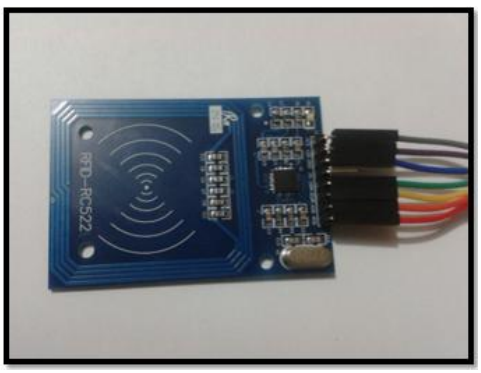


Figura III.33: Unidad Sensora
Fuente: Elaboración Propia

Para el caso de SPI tenemos esencialmente 4 líneas:

- **MOSI:** *Master Out Slave In*, por esta línea se envían datos en forma serial desde el dispositivo "maestro" a los dispositivos esclavos.
- **MISO:** *Master In Slave Out*, por esta línea los dispositivos esclavos envían datos al dispositivo maestro.
- **SCK/SCLK:** Línea de reloj, aquí se envía un tren de pulsos que se encarga sincronizar las comunicaciones entre los dispositivos.
- **CE:** Cada dispositivo tiene una línea de "habilitado", en el dispositivo maestro deberían de haber tantas líneas de habilitado como dispositivos esclavos existan. Los dispositivos esclavo en cambio solo tienen una línea CE que les indica cuando tienen que enviar/recibir datos.

El RC522 utiliza un sistema avanzado de modulación y demodulación para todo tipo de dispositivos pasivos de 13.56Mhz.La tarjeta que viene con el módulo RFID cuenta con 64 bloques de memoria donde se hace lectura y/o escritura la cual tiene la capacidad de almacenar hasta 16 Bytes. En la siguiente Tabla III.11 se muestra los pines del módulo RFID RC552, así como la conexión que tendrá con el Arduino UNO.

ARDUINO UNO	RFID RC552
DIGITAL PIN #10	SDA
DIGITAL PIN #13	SCK
DIGITAL PIN #11	MOSI
DIGITAL PIN #12	MISO
N/A	IRQ
POWER GND	GND
DIGITAL PIN #5	RST
POWER 3.3 V	3.3 V

Tabla III.11: Configuración del RC522.
Fuente:http://hetpro-store.com/TUTORIALES/bnbnbn/comunicacion/modulo-lector-rfid-rc522-rf/

3.3.2.1.2. Interconexión con Arduino

El Arduino Uno estará conectado directamente a la unidad de poder por lo que para alimentar al sensor se lo hace mediante los pines de 3.3V y GND del Arduino como se muestra en la Figura III.34. Para que sea programado en el Arduino necesita de una biblioteca llamada `#include<RFID.h>`para su correcto funcionamiento.

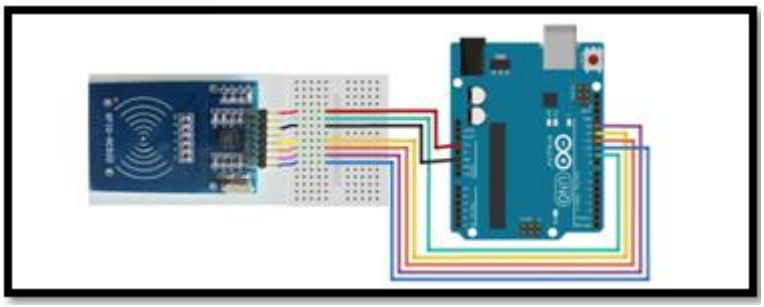


Figura III.34: Conexión con Unidad Sensora.

Fuente:http://1.bp.blogspot.com/-ISHS_dNyh24/U11tOelhyBI/AAAAAAAAATo/e8lYkg2Hkj0/s1600/RFID+teste+cartao.png

3.3.2.2. Pantalla LCD

La Pantalla De Cristal Líquido o LCD (LIQUID CRYSTAL DISPLAY) es un dispositivo de visualización grafico para la presentación de caracteres, para el proyecto se dispondrá de 2 filas de 16 caracteres, aunque en el mercado los hay de otro número de filas y caracteres como se muestra en la Figura III.35. En el LCD se mostrará el nombre del docente que se registra al momento de pasar su tarjeta sobre el lector RFID.

Figura III.35: Pantalla LCD 16x2.
Fuente: http://www.szlcd.com

3.3.2.2.1. Interconexión con Arduino

En la Figura III.36 me muestra la conexión de la pantalla LCD a la placa Arduino, conectamos los siguientes pines:

- Pin LCD RS al pin digital 7
- LCD Enable pin a pin digital 8
- LCD pin D4 al pin digital 6
- LCD pin D5 al pin digital 5
- LCD pin D6 al pin digital 2
- LCD D7 pin a pin digital 4

Además, para que el LCD se encienda cada vez que lea una tarjeta el pin k conectamos al puerto PMW 3 del Arduino para ahorrar.

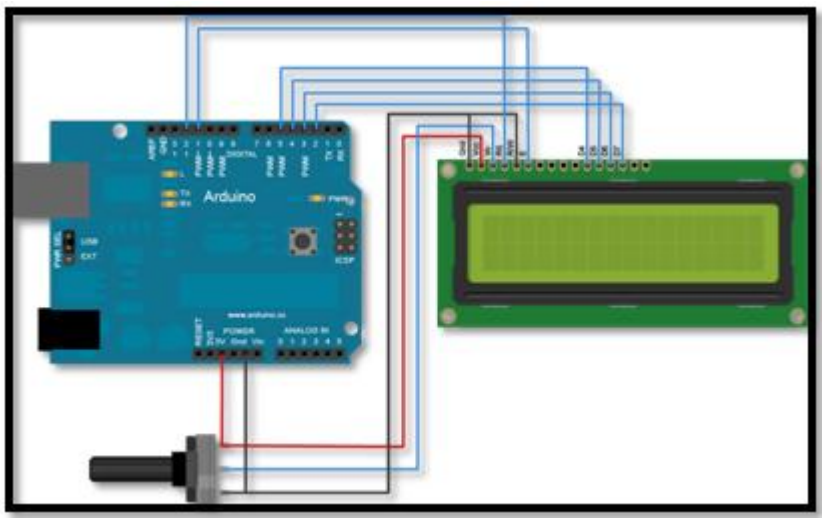


Figura III.36: Conexión con LCD.

Fuente: http://arduino.cc/en/uploads/Tutorial/LCD_bb.png

3.3.2.3. Unidad de Potencia

El Arduino es alimentado por una batería de 9V. En la cual primero deslizamos la cubierta exterior de la clavija y soldamos el cable negro del clip de la batería (-) de alambre para la conexión exterior del tapón. Luego soldar rojo (+) del alambre del clip de la batería a la conexión central del conector, como se muestra en la siguiente Figura III.37.

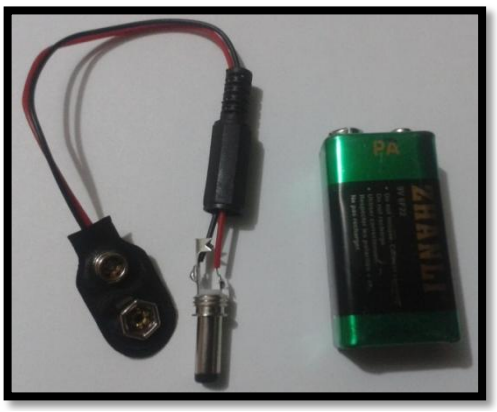


Figura III.37: Unidad de Potencia.

Fuente: Elaboración Propia

3.3.2.4. Unidad de Proceso

Los nodos cuentan con un Arduino UNO para la recolección e interpretación de los datos recibidos por la unidad sensora como se observa en la Figura III.38.

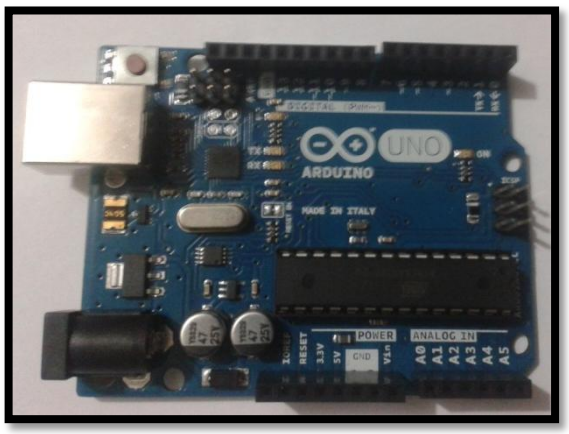


Figura III.38: Unidad de Proceso.

Fuente: Elaboración Propia

El Arduino UNO R3 trabaja con un micro controlador ATmega16U2 lo que nos permite trabajar a velocidades de transferencia muy altas, no necesita drivers para funcionar con las últimas versiones de Windows. También incorpora pines SDA y SCL junto a la AREF. Además, hay dos pines nuevos colocados cerca de la pin de reseteo uno es el IOREF que permite a los Shield adaptarse a la tensión proporcionada desde el tablero. La otra no está conectada y está reservada para usos futuros.

El UNO R3 proporciona todas las protecciones existentes, pero se puede adaptar a los nuevos *Shield* que utilizan estos pines adicionales. Entre las principales características de este Arduino podemos destacar las siguientes:

- Micro controlador ATmega328
- Tensión de entrada 7-12
- 14 pines digitales I/O (6 salidas PWM)
- 6 entradas analógicas
- 32Kb de memoria Flash
- Velocidad de reloj 16 MHz

3.3.2.5. Unidad de Transmisión y Recepción

Es una de las unidades más importantes en el funcionamiento de las redes WSN debido a que es la encargada de transmitir la información recolectada por el lector RFID y transmitirla de forma inalámbrica hacia los demás nodos, o en este caso hacia el nodo coordinador.

Los módulos *XBee* utilizados para la comunicación inalámbrica entre los nodos pueden ser configurados mediante el *software X-CTU* disponible en la página web del fabricante Digi Internacional. Los módulos *XBee* S1 pueden

comunicarse en arquitecturas punto a punto, punto a multipunto o en una red tipo mesh.

Los *XBee* de la serie 1 y serie 2 tienen el mismo área de comunicación (*footprint)*, y la configuración de sus pines es, pero no pueden interactuar entre series diferentes es decir no son interoperables. Serie 1 y Serie 2 utilizan diferentes perfiles de aplicación, que son únicas para cada familia de radio. Sin embargo, pueden utilizar las mismas tarjetas de interfaz *XBee Explorer Serial* y USB.

Para este trabajo decidió utilizar los *XBee S1*, como se muestra en la Figura III.39 ya que cumple con los requerimientos necesarios para la aplicación a desarrollarse, trabajan con una configuración punto multipunto y tienen un alcance de 30 metros en interiores que es en donde se realizaran las pruebas.

Figura III.39: Unidad de Tx y Rx.

Fuente: Elaboración Propia

3.3.2.5.1. Interconexión con Arduino

Para entender cómo trabajan conjuntamente Arduino y *XBee* es necesario conocer el funcionamiento y para qué sirven los pines del *XBee s1* como se muestra en la Figura III.40.

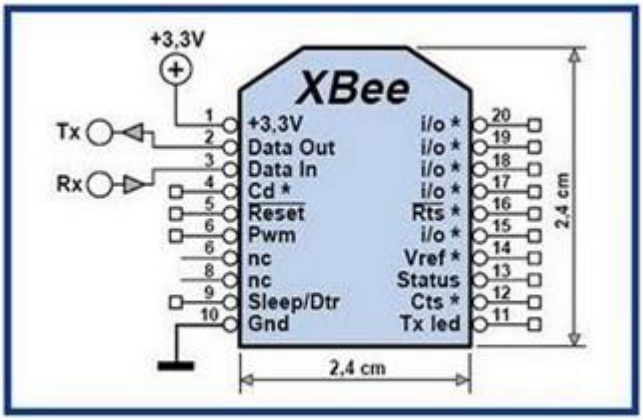


Figura III.40: Funcionamiento del XBee.

Fuente:http://www.pyroelectro.com/tutorials/xbee_wireless_interface/img/xbee_pinout.jpg

Para que se dé la comunicación entre la placa Arduino y el módulo *XBee* se deberá conectar cuatro de los pines del *XBee* como se detalla a continuación en la siguiente Tabla III.12.

XBee S1	**ARDUINO UNO**
Pin 1	VCC (3.3V)
Pin 2	Tx
Pin 3	Rx
Pin 4	GND (Opcional para Reset)
Pin 9	GND (Opcional para Sleep)
Pin 10	GND

Tabla III.12: Conexión del XBee al Arduino uno.
Fuente: Elaboración Propia

En el proyecto se utilizó un *XBee Shield* para Arduino, tal como se muestra en la Figura III.41 se observa la interconexión entra la placa Arduino y el módulo *XBee* mediante el Shield de comunicación para Arduino y *XBee*.

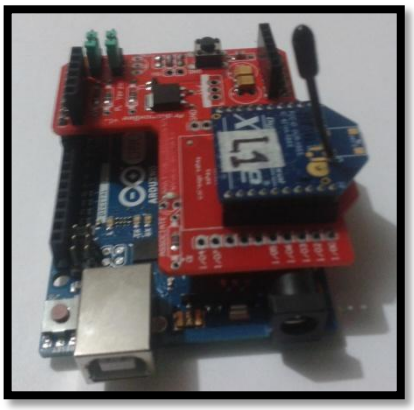


Figura III.41: Unidad de Tx y Rx con Arduino.
Fuente: Elaboración Propia

3.3.3. Dispositivo Coordinador

El nodo coordinador está formado por un módulo de comunicación *XBee S1* el cual se encargara de recibir la información transmitida por las demás unidades sensoras conocidas como *End Device* o dispositivos finales. Para enviar la información hacia la base de datos se conectara directamente mediante el *XBee Explorer* que se conecta al puerto USB de la computadora a través de un cable *micro USB* como se muestra en la Figura III.42.

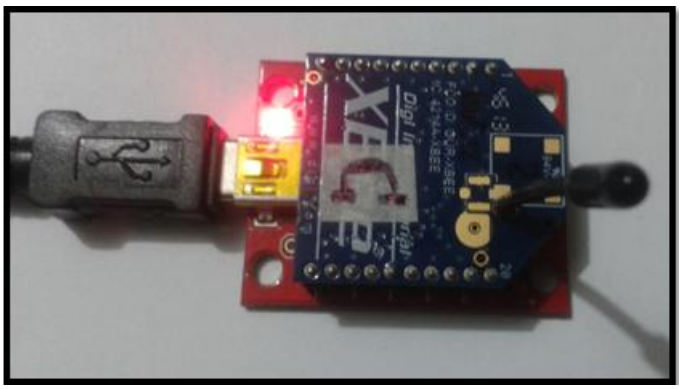


Figura III.42: Dispositivo Coordinador.
Fuente: Elaboración Propia

3.3.4. Aplicación

La información enviada por los lectores RFID hacia el nodo coordinador se los transmitirá a la base de datos, aquí se interpretara esta información, se identificara de que laboratorio proviene dicha información y se le indicara a quien corresponde el número de tarjeta leída.

Se desarrolló una aplicación en Java con el apoyo del software *NetBeans* que es un entorno integrado de desarrollo para lograr una interfaz gráfica amigable que permite extraer los datos del puerto serie del módulo *XBee* y luego se puedan almacenar en la base de datos.

3.3.4.1. Base de Datos

El diseño de la base de datos permitirá obtener acceso a información exacta y actualizada. La base de datos será capaz de ir almacenando en tiempo real los registros que realizaran los docentes al inicio de clases de cada laboratorio de CISCO, cada laboratorio tendrá un nodo y cada nodo contendrá el sensor lector RFID que envía las señales hacia la base de datos.

Para esto se crea una base de datos utilizando el gestor PostgreSQL que es un producto de código abierto. Para el diseño de la base de datos utilizaremos el software Power Designer que es una herramienta para la construcción de tablas, a continuación en la Tabla III.13 se muestra las tablas que utilizares en nuestra base de datos.

Laboratorio	
nombre	***tipo***
id_lab	integer
nombre_lab	varchar

Materia	
nombre	***tipo***
id_materia	integer
nombre_materia	varchar

Horario	
nombre	***tipo***
id_horario	integer
hora_inicio	time
hora_salida	time

Docente	
nombre	***tipo***
id_docente	varchar
ci_docente	varchar
nombre_docente	varchar
celular_docente	varchar

Registro	
nombre	***tipo***
id_registro	integer
hora_registro	time
fecha_registro	date

Tabla III.13: Tablas de la base de datos.
Fuente: Elaboración Propia

Al final se obtendrá una base de datos con las relaciones establecidas como se observa en la tabla III.14.

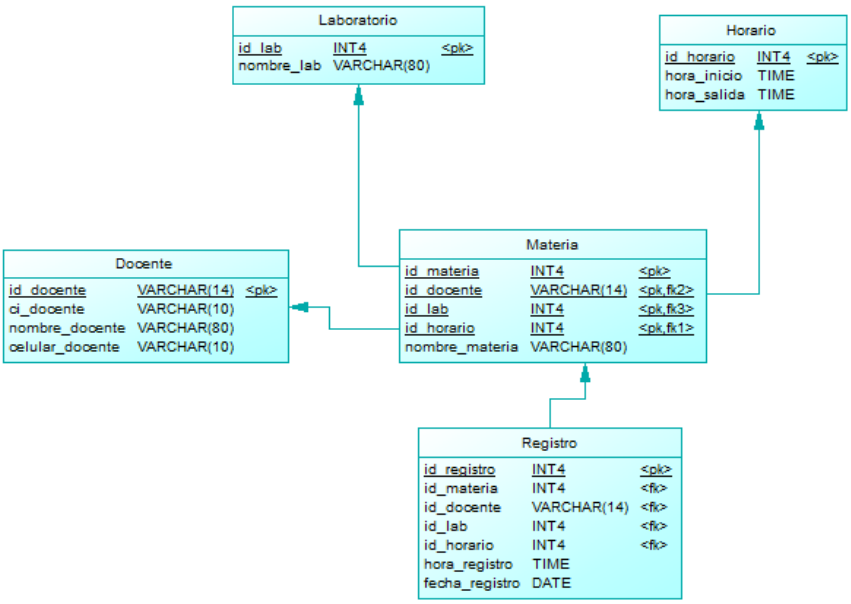


Tabla III.14: relación entre tablas.
Fuente: Elaboración Propia.

CAPÍTULO IV

IMPLEMENTACIÓN Y EVALUACION DEL PROTOTIPO WSN Y RFID

4.1. INTRODUCCIÓN

En este capítulo se describirá la implementación del prototipo WSN y RFID para el control de acceso de los docentes en los laboratorios de la academia CISCO. Se describirá las características de hardware de los dispositivos que conforman el prototipo, tales como la placa base *Arduino uno*, módulos

de comunicación *XBee*, la interfaz *XBee Shield* que interconecta la placa base con los módulos de comunicación, *los XBee Explorer USB*, los sensores RFID y las tarjetas RFID.

En cuanto al software se detalla cómo fueron programados cada uno de los módulos de comunicación *XBee*, además la programación que se realizó en la placa base Arduino para que recolecte, interprete y transmita los datos adquiridos hacia el nodo coordinador, la cual recibirá los datos de cada uno de los nodos y los almacenará en una base de datos. La base de datos permitirá mantener un registro de los docentes y mediante el desarrollo de una aplicación en Java podremos verificar la asistencia a clases de los docentes en los laboratorios de la academia CISCO.

4.2. IMPLEMENTACIÓN DEL PROTOTIPO WSN Y RFID

Se procede a la implementar luego del diseño del prototipo, se describe a continuación las configuraciones paso a paso de cada elemento del sistema WSN y RFID, así como también se explica cómo se realizó la aplicación y la base de datos del sistema.

4.2.1. Configuración Arduino Uno

El sensor RFID cuando recepta la información de la tarjeta este sensor envía al Arduino una señal donde debe ser capaz de interpretar dicha señal para que posteriormente la procese y que al final pueda ser enviada al coordinador y a la base de datos.

4.2.1.1. Instalación Software Arduino

El software en un entorno de código abierto de fácil escribir el código. Arduino se puede descargar desde la página oficial [33]. Es fundamental tener un cable mini USB y el módulo Arduino.

Una vez instalada el software en el computador, conectamos la placa Arduino al ordenador mediante el cable USB, luego espere a que Windows comience su proceso de instalación de los controladores. Para comprobar que los controladores se han instalado, haga clic en inicio, clic derecho en equipo después clic en administrador, clic en administrador de dispositivos y en la pestaña Puertos (COM) debe estar Arduino Uno como en este caso en el COM19 así como se muestra en la siguiente Figura IV.43.

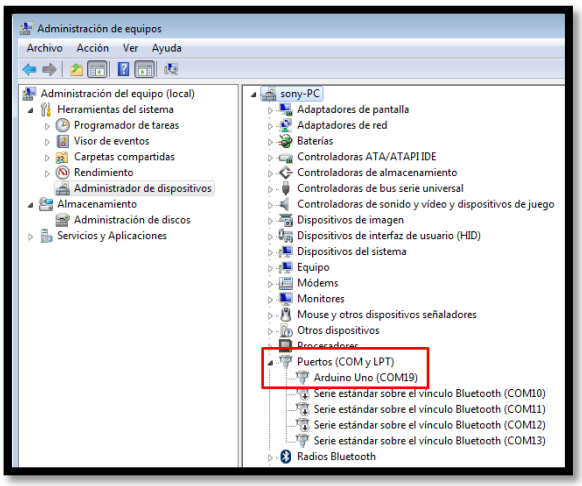
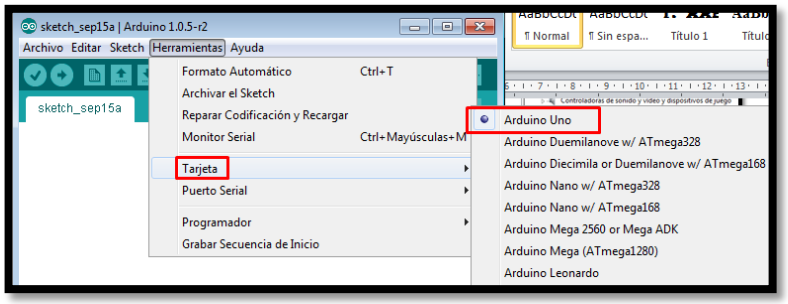


Figura IV.43: Instalación controladores de Arduino.

Fuente: Elaboración Propia

Al momento de abrir el software Arduino tenemos que seleccionar el tipo de modulo que vamos a configurar y el puerto COM al que se haya conectado el modulo como se muestra en la siguiente Figura IV.44.



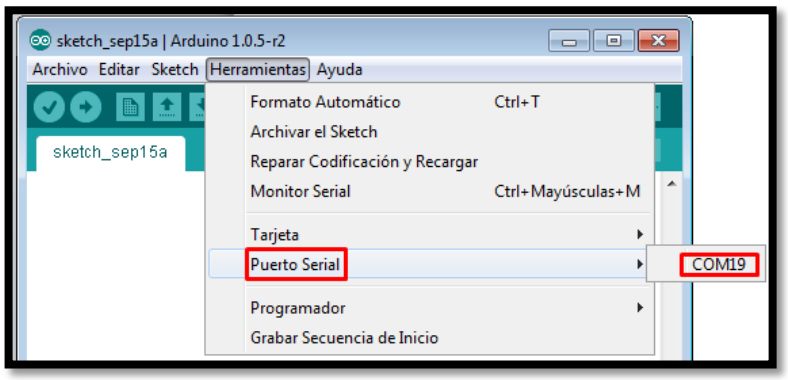


Figura IV.44: Conexiones fundamentales Arduino.

Fuente: Elaboración Propia

4.2.1.2. Instalación de librerías

Arduino por defecto consta de librerías por defecto, pero para el uso del sensor RFID RC522 es necesario instalar la librería específica para su funcionamiento para este sensor.

La librería para el funcionamiento del RFID RC522 se lo descarga del internet, luego lo descomprimimos y la carpeta descomprimida la pegamos en la siguiente dirección C:\Program Files\Arduino\libraries.

Para el uso de la pantalla LCD 12x2 no es necesario instalar la librería ya que por defecto viene instala en el software, la librería que se llama Liquid Crystal.

Para la comunicación serial también hace falta su funcionamiento de una librería llamada SPI, pero no hace falta instalarla ya que por defecto se encuentra en el software Arduino.

4.2.1.3. Programación del Arduino

El software instalado correctamente en el computador y con las librerías indicadas anteriormente, se procederá a programar el código en el módulo Arduino UNO. Lo que se debe logra es que el sensor recepte la señal de la tarjeta RFID del docente y la convierta en un código hexadecimal par después enviarla al nodo coordinador. El código se encuentra en ANEXO 2.

Para cargar el código en el módulo Arduino UNO, primero se debe comprobar que el código este bien configurado y si no existe ningún problema se procede a cargar el código haciendo clic en el botón cargar del software Arduino como se observa en la siguiente Figura IV.45.

Figura IV.45: Cargar el código en Arduino.

Fuente: Elaboración Propia

4.2.2. Configuración de los módulos XBee

Procedemos a la configuración de los módulos XBee de los nodos finales, que tendrán la función de solo únicamente enviar la información hacia el nodo coordinador, la topología utilizada es la estrella para lograr optimizar el uso de energía y la redundancia de datos, es decir que el nodo coordinador sea capaz de recibir los datos de todos los nodos finales como se observa en la siguiente Figura IV.46.

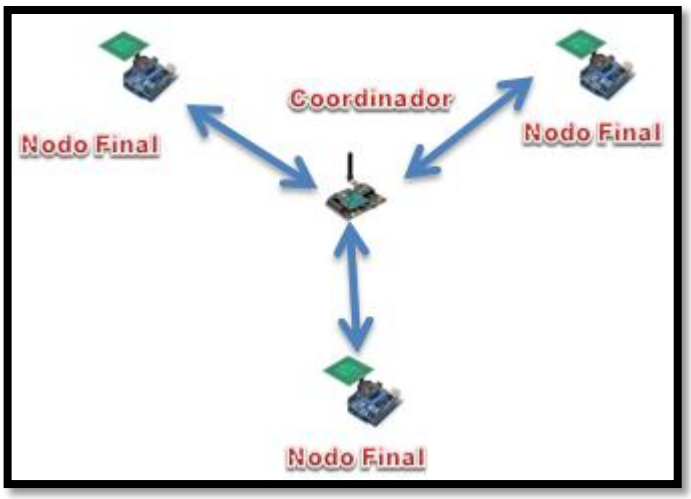


Figura IV.46: Sentido de comunicación entre XBee.

Fuente: Elaboración Propia

4.2.2.1. Instalación Software X-CTU

Este software es gratuito diseñado para la configuración de los módulos XBee a través de una interfaz gráfica fácil de utilizar, se lo puede descargar desde la página oficial [34].

Ejecutamos el programa a través de un acceso directo ubicado en el escritorio y nos aparecerá una pantalla como la siguiente Figura IV.47.

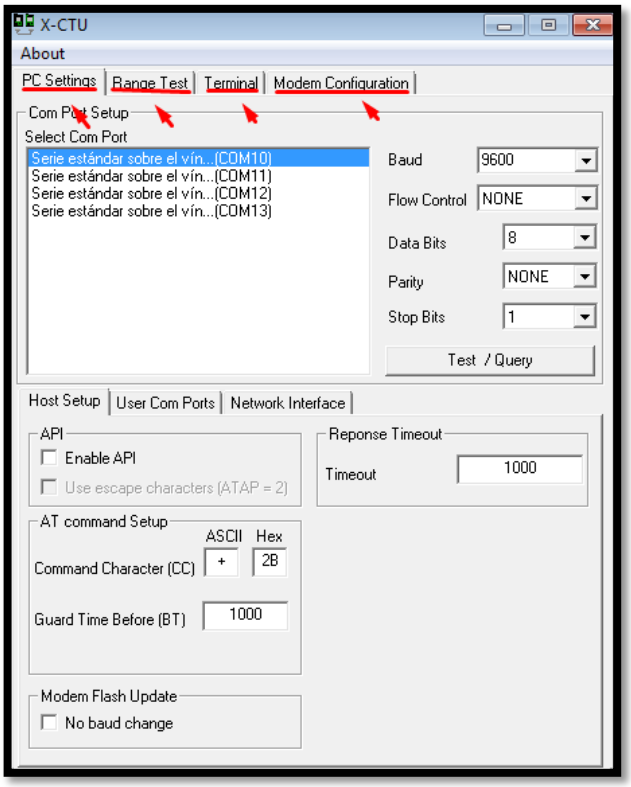


Figura IV.47: Software X-CTU.

Fuente: Elaboración Propia

Como se observa en la Figura IV.47 existen cuatro pestañas que son muy importantes tomar en cuenta al momento de configurar los módulos XBee.

- **PC Settings**

Esta es la pestaña que por defecto aparece cuando iniciamos el programa, en esta área permite al usuario seleccionar un puerto COM como se observa en la Figura IV.48 en este caso al puerto que se conectó el XBee es el COM2. Para probar si en el puerto COM21 está conectado el XBee existe una opción para verificar que es la botón Test/Query y como se demuestra en la misma figura la comunicación con el modulo fue exitosa.

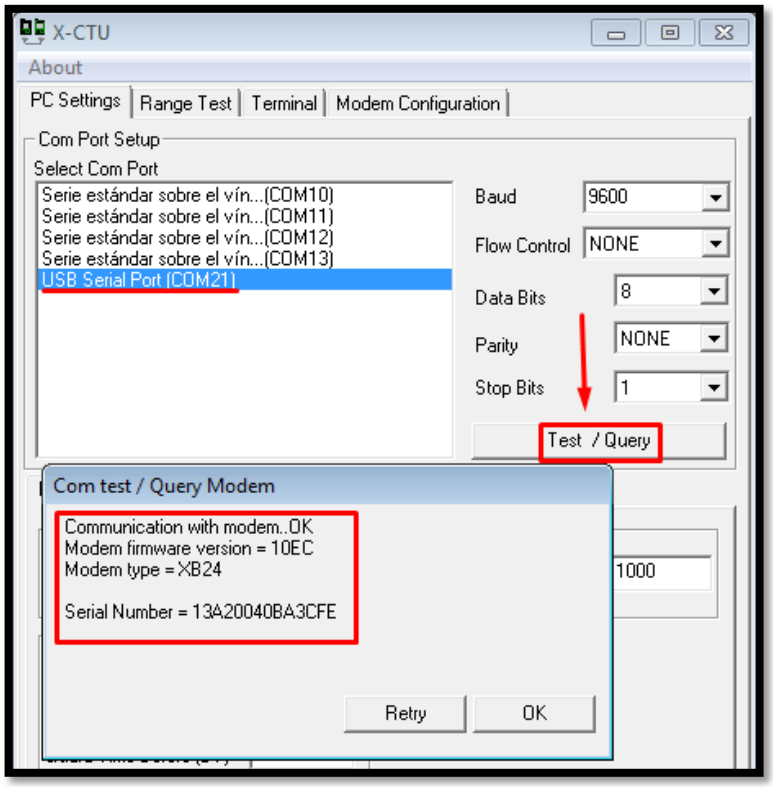


Figura IV.48: Pc Settings X-CTU.

Fuente: Elaboración Propia

- **Range Test**

En esta pestaña sirve para verificar si los datos llegan sin ningún problema, es decir generamos cadena de datos de cualquier tipo para probar el rango de alcance de la señal, como se observa en la siguiente Figura IV.49.

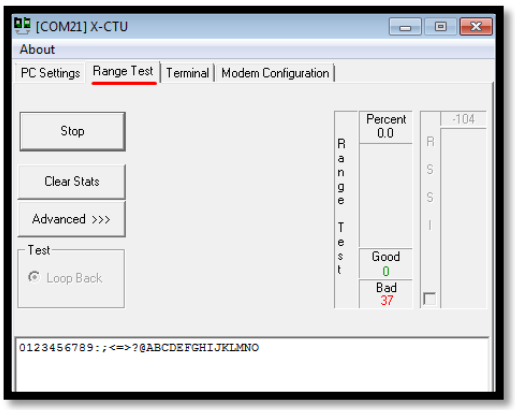


Figura IV.49: Range Test X-CTU.

Fuente: Elaboración Propia

- **Terminal**

En esta pestaña es donde la comunicación de información de producirán es decir el texto en azul es lo que se ha escrito por el usuario y será enviado hacia el módulo XBee mediante el puerto serial, mientras tanto el texto en color rojo es la respuesta de datos que envía desde el módulo XBee mediante el puerto serial como se observa en la Figura IV.50.

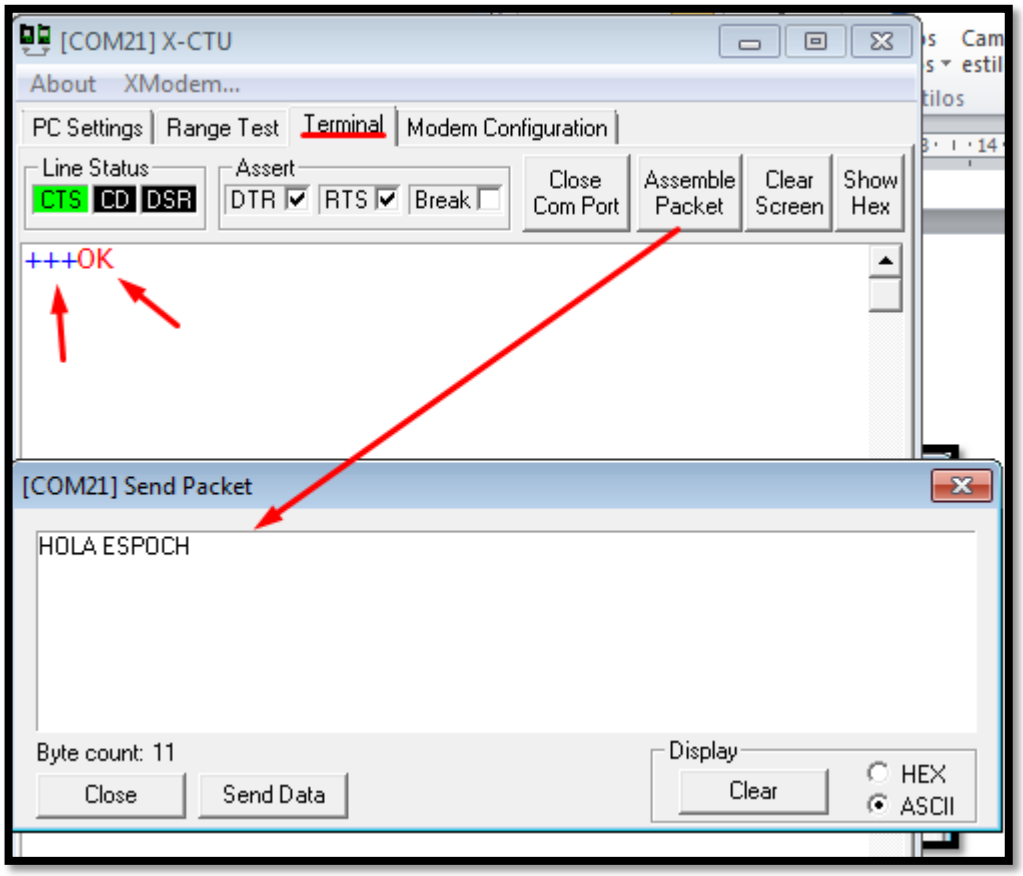


Figura IV.50: Terminal X-CTU.

Fuente: Elaboración Propia

El botón Assemble Packet está diseñada para permitir al usuario enviar paquetes de datos en dos tipos de caracteres ya sea en ASCCII o en HEX, esto lo podemos observar en la misma figura de arriba.

- **Modem Configuration**

En esta pestaña lo primero que debemos hacer es leer el módulo XBee seleccionando el botón Ready se desplegara todas las configuraciones que se puede editar al módulo XBee como se puede observar en la siguiente Figura IV.51. Se puede diferenciar tres colores diferentes la cual significa que el color negro no se puede configurar es decir solo lectura, el color verde significa que es el valor por defecto y el azul son los valores que el usuario a cambiado.

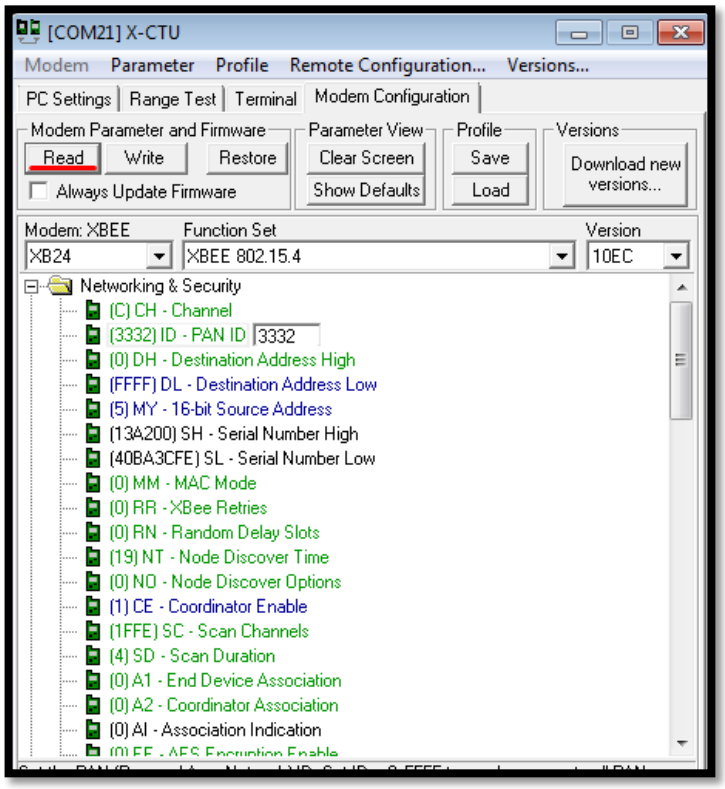


Figura IV.51: Modem Configuration X-CTU.
Fuente: Elaboración Propia

Una vez terminado las configuraciones debemos guardar los cambios realizados en el módulo XBee y procedemos hacer clic en el botón Write.

4.2.2.2. Configuración del coordinador XBee

Procedemos a la configuración del nodo coordinador el cual estar conectado directamente con la computadora, después de comprobar que esté conectado

correctamente el módulo XBee al software X-CTU las configuraciones necesarias se resumen en la siguiente Tabla IV.15.

NODO COORDINADOR	
Parámetro	Valor
Channel	C
PAN ID	2461
Destination Address High	0
Destination Address Low	FFFF
Serial Number High	13A200
Serial Number Low	40BA3CFE
Coordinador Enable	1-COORDINADOR
Nodo Identifier	COORDINADOR

Tabla IV.15: Parámetros del nodo coordinador.
Fuente: Elaboración Propia.

- **Channel C**

Es el canal que se le asigna al módulo XBee para la comunicación inalámbrica, al indicarle al módulo que trabaje en el canal C se entiende que el rango de frecuencia es de 2,4075 – 2,4125 GHz.

- **PAN ID 2461**

Este es el identificador de la red, para este proyecto se le asignó el número 2461.

- **Destination Address**

El parámetro *Destination Address High* están configurado con un valor de cero y el parámetro *Destination Address Low* está configurado con un valor FFFF que indica al módulo XBee que tendrá la recepción y transmisión desde todos los nodos finales que estén en el mismo canal y que contengan la misma PAN ID.

- **Serial Number**

Estos parámetros son propios del módulo XBee, es decir estos datos no se los puede editar y esto se lo puede encontrar impreso en la parte posterior de cada módulo.

- **Coordinador Enable**

Este parámetro es el que identifica si el modulo es coordinador o un nodo final, en este caso configuramos con el valor de 1 que significa coordinador.

Finalmente procedemos a grabar en memoria la configuración como se muestra en la Figura IV.52.

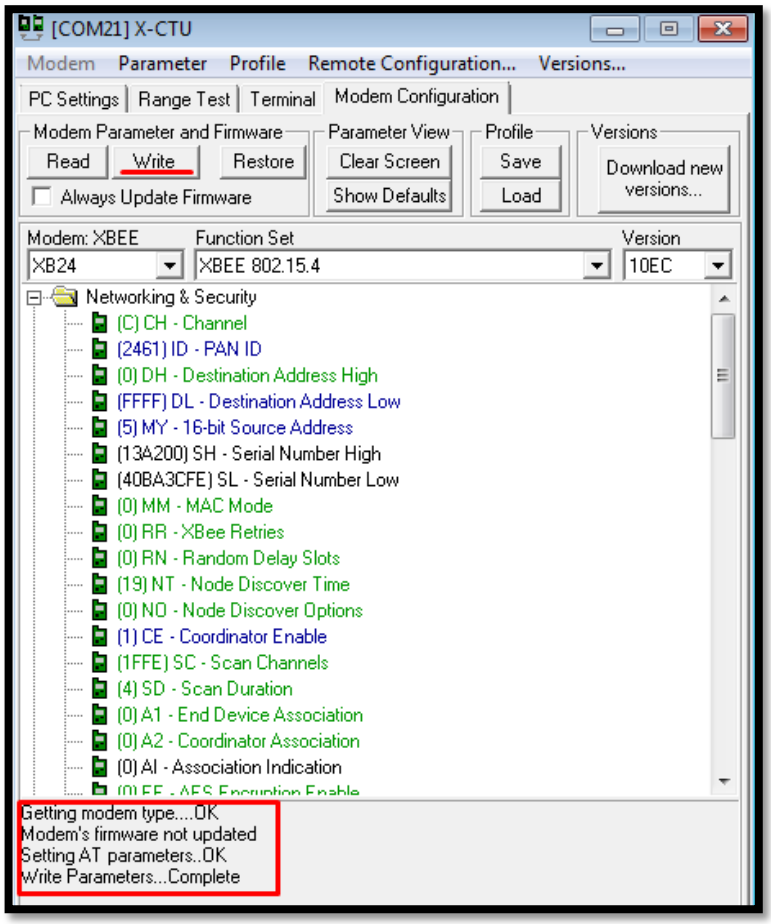


Figura IV.52: Configuración nodo Coordinador X-CTU.

Fuente: Elaboración Propia

4.2.2.3. Configuración de los nodos finales XBee

La configuración de los nodos finales se realiza de la siguiente manera como se observa en la siguiente Tabla IV.16 y Tabla IV.17 tanto para el Laboratorio 1 y Laboratorio 2 respectivamente.

NODO FINAL 1	
Parámetro	Valor
Channel	C
PAN ID	2461
Destination Address High	13A200
Destination Address Low	40BA3CFE
Serial Number High	13A200
Serial Number Low	40B8EC00
Coordinador Enable	0-END DEVICE
Nodo Identifier	LABORATORIO 1

Tabla IV.16: Parámetros del nodo 1.
Fuente: Elaboración Propia.

NODO FINAL 2	
Parámetro	Valor
Channel	C
PAN ID	2461
Destination Address High	13A200
Destination Address Low	40BA3CFE
Serial Number High	13A200
Serial Number Low	40A5C839
Coordinador Enable	0-END DEVICE
Nodo Identifier	LABORATORIO 2

Tabla IV.17: Parámetros del nodo 2.
Fuente: Elaboración Propia.

- **Channel C**

Es el canal que se le asigna al módulo XBee para la comunicación inalámbrica, al indicarle al módulo que trabaje en el canal C se entiende que el rango de frecuencia es de 2,4075 – 2,4125 GHz.

- **PAN ID 2461**

Este es el identificador de la red, para este proyecto se le asignó el número 2461.

- **Destination Address**

El parámetro *Destination Address High* están configurado con un valor de 13A200 y el parámetro *Destination Address Low* está configurado con un valor 40BA3CFE que es el SL del coordinador, la cual indica que el dispositivo final únicamente enviara los datos al módulo coordinador.

- **Serial Number**

Estos parámetros son propios del módulo XBee, es decir estos datos no se los puede editar y esto se lo puede encontrar impreso en la parte posterior de cada módulo.

- **Coordinador Enable**

Este parámetro es el que identifica si el modulo es coordinador o un nodo final, en este caso configuramos con el valor de 0 que significa dispositivo final.

Una vez terminado la configuración procedemos a guardar los cambios haciendo clic en el botón Write como se observa en la siguiente Figura IV.53.

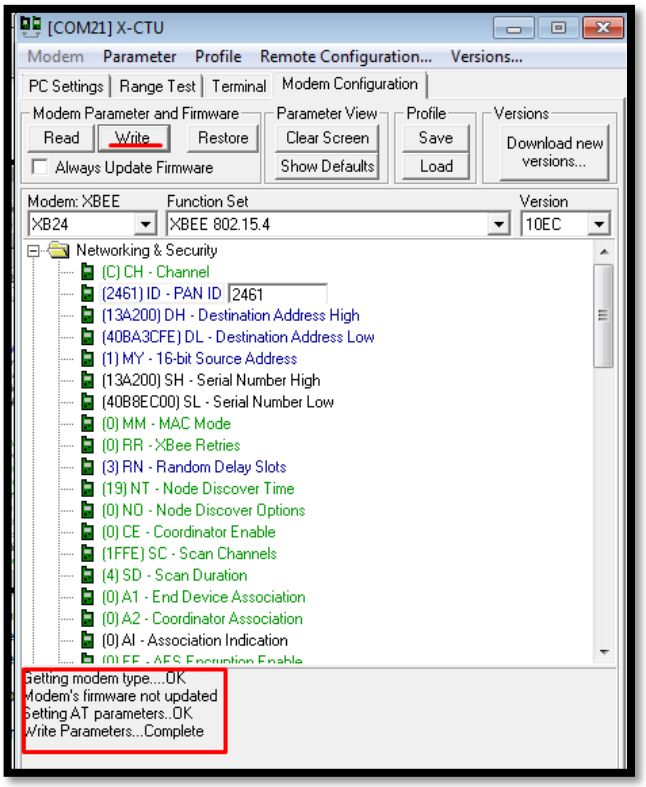


Figura IV.53: Configuración nodo Final X-CTU.

Fuente: Elaboración Propia

4.2.3. Aplicación

Se desarrolló una aplicación basado en Java servirá para que los datos sean representados de una forma amigable para el usuario, esto permitirá almacenar la información en una base de datos. La base de datos se utilizo es PostgreSQL que es un sistema de gestión de base de datos de código abierto más potente en el mercado, el PostgreSQL utiliza un modelo cliente-servidor y usa multiprocesos para garantizar el estabilidad del sistema.

El nodo coordinador enviara los datos a través del puerto COM a la aplicación Java y esta a su vez por el puerto 5432 enviará hacia la base de datos, el esquema se grafica en la siguiente Figura IV.54.

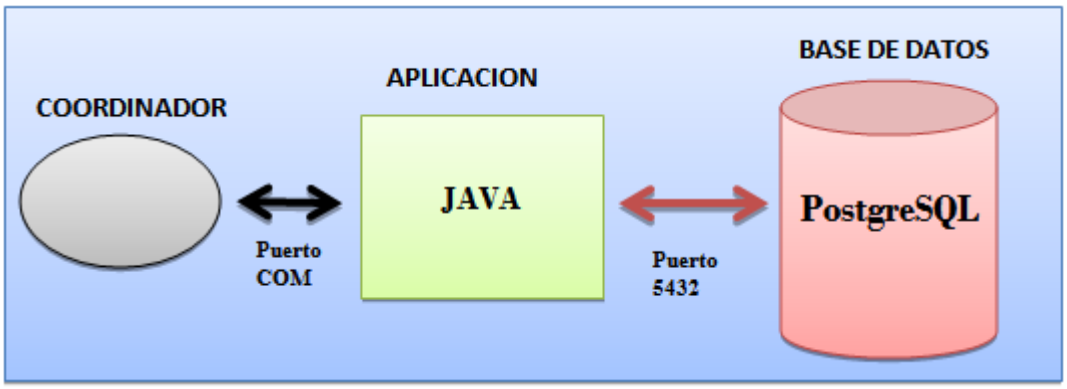


Figura IV.54: Esquema Base de Datos.
Fuente: Elaboración Propia

4.2.3.1. Configuración en pgAdmin III

PgAdmin III es una herramienta de código abierto para la administración de dase de datos en PostgreSQL y sus derivados, para obtener el software la versión más reciente debemos dirigirnos a la página oficial [35]. Al finalizar la instalación aparecerá la pantalla como la siguiente Figura IV.55.

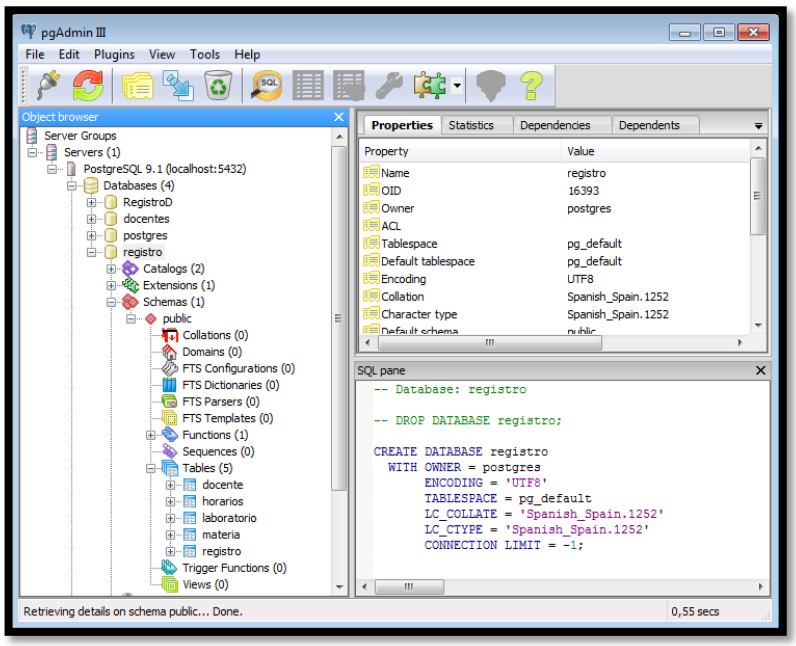


Figura IV.55: pgAdmin III.

Fuente: Elaboración Propia

PostgreSQL utiliza un modelo cliente/servidor puede manejar múltiples conexiones simultáneas de clientes.

4.2.3.1.1. Creación de la Base de datos

Termina la instalación del pgAdmin para poder agregar tablas, es necesario crear primero una base de datos, es por eso que nos ubicamos en el *Object browser* y click derecho sobre Databases y seleccionar New Database del menú que se despliega como se muestra en la siguiente Figura IV.56.

Figura IV.56: base de datos pgAdmin III.

Fuente: Elaboración Propia

Después se visualizará algunos campos que son los siguientes:

Nombre: denominación de la base de datos se recomienda que por simplicidad, evitar espacios, mayúsculas y acentos, separar con guion bajo. Para nuestra base de datos le llamaremos RegistroD.

Propietario: es el usuario que tendrá derechos especiales sobre la base de datos.

Codificado: esquema de codificación.

Creación de las Tablas

Para agregar una tabla, nos ubicamos en el *Object browser*, desplazamos hasta la base de datos que acabamos de crear, en la pestaña *Schemas* en *public* podemos visualizar la pestaña *Tables* al cual hacemos click derecho y seleccionar *New Table* del menú contextual como se muestra en la siguiente Figura IV.57.

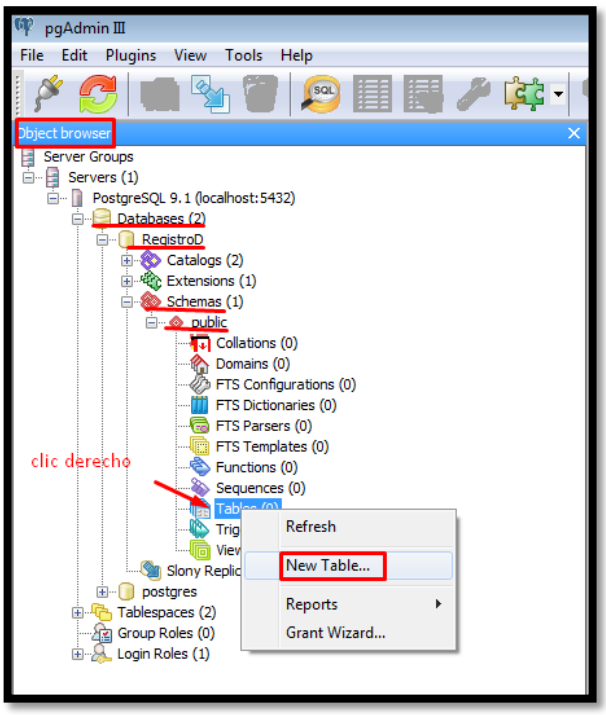


Figura IV.57: Crear Tablas pgAdmin.

Fuente: Elaboración Propia

En el parámetro Nombre escribimos el nombre que pertenece la tabla y después procedemos a crear las columnas.

Name: denominación de la tabla por simplicidad, evitar espacios, mayúsculas y acentos, separar con guion bajo.

Owner: usuario que tendrá derechos especiales sobre la tabla

Luego, seleccionar en la pestaña Columnas, por cada columna a agregar presionar el botón Añadir:

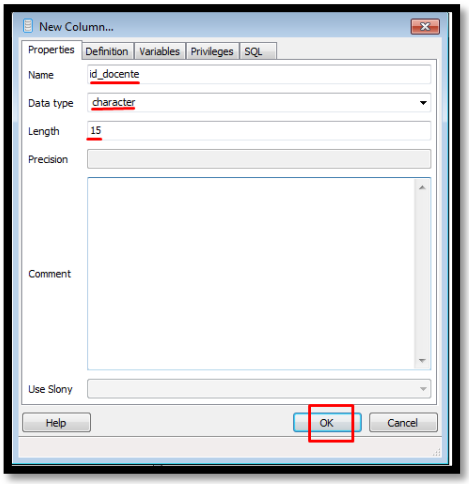


Figura IV.58: Crear columnas pgAdmin.

Fuente: Elaboración Propia

Posteriormente realizamos los mismos procedimientos para agregar las demás columnas. También se agregara de la misma forma las tablas Laboratorio, Materia, Horario y la tabla Registro como se indicó anteriormente.

Herramienta de Consulta SQL

Al seleccionar una base de datos, se habilita la herramienta de consulta, que permite ejecutar consultas SQL arbitrarias, para poder ingresar a dicha herramienta, presionar el botón Ejecutar consultas SQL arbitrarias de la barra de herramientas, la cual nos aparecerá como la siguiente Figura IV.59.

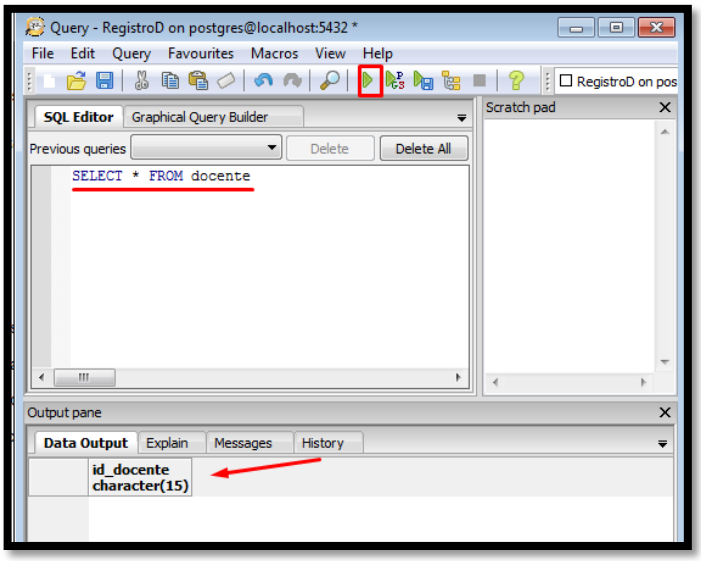


Figura IV.59: Herramienta de consulta SQL.

Fuente: Elaboración Propia

4.2.3.2. Configuración en Java

Java es un lenguaje de programación de ordenadores, diseñado como una mejora de C++. Antes de descargar el software NetBeans debe tener instalado en el sistema la actualización del JDK (*Java SE Development Kit*), puede descargar la versión más reciente de JDK en [36].

Una vez instalada la última versión del JDK procedemos a instalar el NetBeans de la página disponible en [37] donde se puede obtener uno de los diversos instaladores, cada uno de los cuales contiene el IDE básico y herramientas adicionales. En la parte superior derecha de la página, seleccione el idioma y la plataforma en la lista desplegable. También puede optar por descargar y utilizar el archivo Zip independiente de la plataforma. Haga clic en el botón Descargar y guarde el archivo del instalador en el sistema.

4.2.3.2.1. Librería para Arduino

La librería Arduino para Java es un conjunto de métodos para hacer el proceso de comunicación entre Arduino y Java. La librería se puede encontrar en el internet en la siguiente página [38]. En la siguiente Tabla IV.18 se describe los métodos de la librería Arduino.

Método	Descripción
ArduinoRX(string nombre del puerto, int time out, int baud rate, SerialPortEventListenerevento)	Este método se utiliza para iniciar la conexión de Java con Arduino SOLAMENTE PARA LA RECEPCIÓN DE DATOS. En el **nombre de puerto** se coloca el COM#, o sea el puerto COM donde esté conectado Arduino, el **time out** es el tiempo de espera (yo uso 2000), el **baud rate** debe ser el mismo que se usa en Arduino IDE (generalmente 9600) y el **Serial PortEventListener** debe ser una variable declarada antes de utilizar este método.
ArduinoTX(string nombre del puerto, int time out, int baud rate)	Este método se utiliza para iniciar la conexión de Java con Arduino SOLAMENTE PARA LA TRANSMISIÓN DE DATOS.
ArduinoRXTX(string nombre del puerto, int time out, int baud rate, SerialPortEventListenerevento)	Este método se utiliza para iniciar la conexión de Java con Arduino PARA LA TRANSMISIÓN Y RECEPCIÓN DE DATOS.
SendData(String data)	Método utilizado para enviar datos a Arduino. Los datos se deben enviar como cadena de texto (string).
ReceiveData()	Devuelve un dato recibido a través del puerto serie. Este dato será numérico en formato ASCII por lo que se debe traducir de decimal a caracter.
MessageAvailable()	Devuelve un valor booleano que nos indica si hay algún mensaje disponible para imprimir. Dicho mensaje DEBE ser enviado desde Arduino utilizando Serial.println();
PrintMessage()	Devuelve una cadena de caracteres que contiene el mensaje que ha sido enviado desde Arduino, pero traducido a caracteres. SE DEBE UTILIZAR DENTRO DE UNA ESTRUCTURA CONDICIONAL UTILIZANDO **MessageAvailable()**. Cuando haya un mensaje disponible, se imprime utilizando este método.

Tabla IV.18: Métodos Librería Arduino.
Fuente: http://panamahitek.com/libreria-arduino-para-java/

4.2.3.2.2. Creación de la Aplicación

Una vez instalada el software NetBeans, creamos un nuevo proyecto, nos dirigimos a *New Projecty* en la primera ventana seleccionamos *Java Application* y a continuación en *Next,* como se observa en la Figura IV.60.

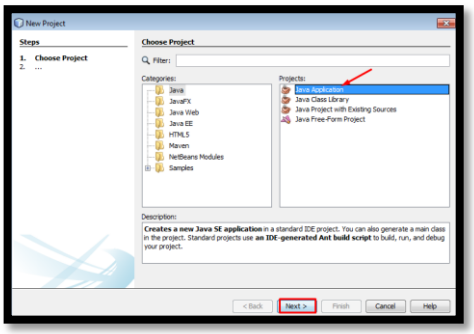


Figura IV.60: Nuevo proyecto NetBeans IDE.

Fuente: Elaboración Propia

En la siguiente ventana debemos desmarcar la opción *Create Main Classe* introduciremos los siguientes datos:

Project Name: nombre que tendrá el proyecto, en este caso "Registro_Docentes".

Project Location: ubicación de las carpetas del proyecto.

Project Folder: carpeta donde se guardarán los ficheros del proyecto. Finalmente hacemos clic en *Finish*, como se observa en la Figura IV.61.

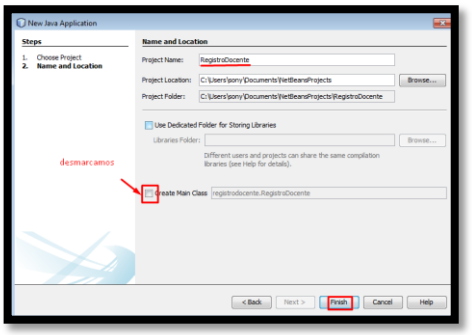


Figura IV.61: Proyecto parámetros NetBeans IDE.

Fuente: Elaboración Propia

Para nuestra aplicación la programación por capas es la que vamos a realizar ya que el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, Una de las ventajas que podemos destacar sobre este estilo es que el desarrollo de la aplicación se puede llevar a cabo en varios tipos de niveles, así, cuando suceda algún cambio solo nos iremos sobre el nivel requerido. Nuestra aplicación tendrá 3 capas como se muestra en la siguiente Tabla IV.19.

- Datos
- Negocio
- Presentación

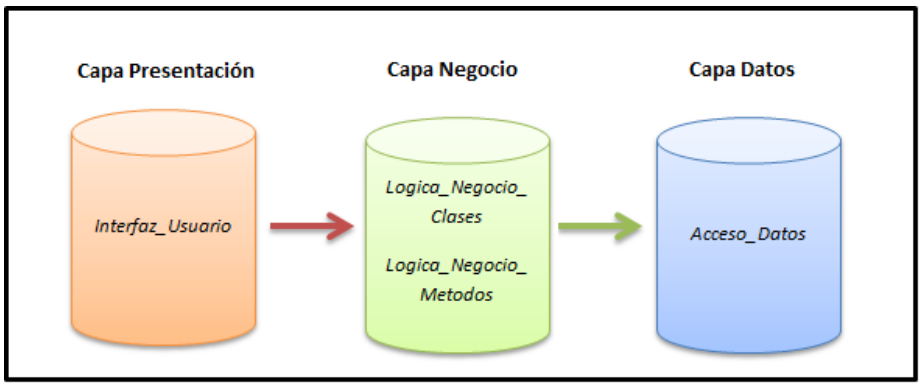


Tabla IV.19: Diagrama de capas de la aplicación.
Fuente: Elaboración Propia

Capa de datos

En esta capa es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Para esta capa crearemos el paquete llamado Acceso_Datos que será la capa de datos para nuestra aplicación. Para proceder a crear un paquete nos dirigimos a *Projects* clic derecho sobre la pestaña *Source Packages* en *New* y hacemos clic en *Java Package* tal como se observa en la FiguraIV.62.

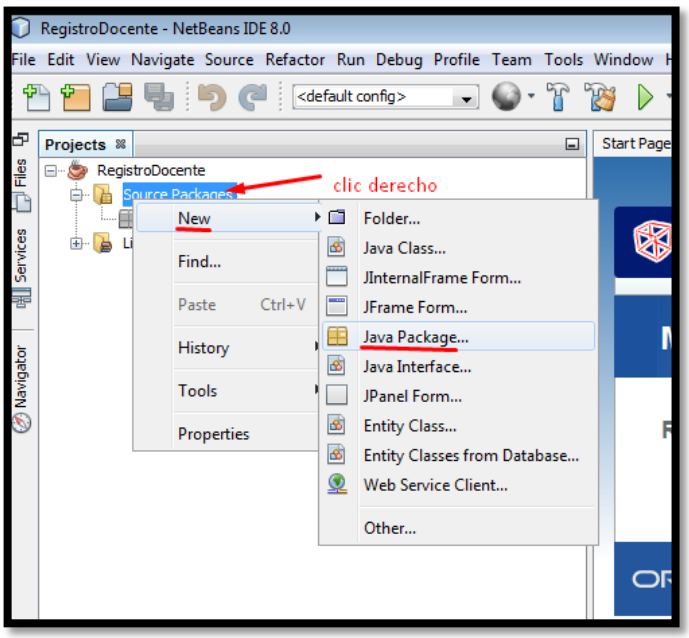


Figura IV.62: Nuevo paquete en NetBeans IDE.

Fuente: Elaboración Propia

Dentro de esta capa de datos llamado Acceso_Datos crearemos tres clases con los nombres:

1. **c_Conneccion**
2. **c_Global**
3. **c_Parametro**

Para crear una clase nos dirigimos sobre el nombre del paquete llamado Acceso_Datos, clic derecho después en *New* y finalmente clic en *Java Class* como se muestra en la siguiente Figura IV.63.

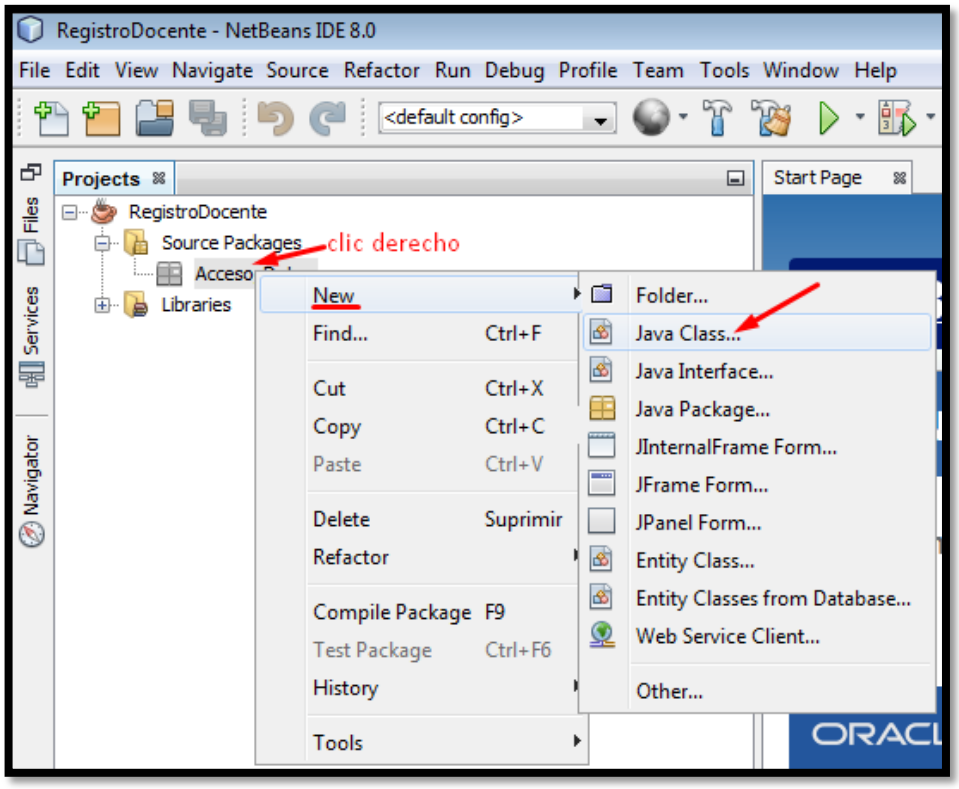


Figura IV.63: Nueva clase en NetBeans IDE.

Fuente: Elaboración Propia

Los mismos pasos realizamos para crear las dos capas que faltan.

Capa de Negocios

En esta capa es donde se encuentran los programas que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas tras el proceso. Esta capa es muy importantes pues es donde se establecen todas aquellas reglas que se tendrán que cumplir, decía anteriormente que la capa de presentación tiene comunicación con la capa de lógica de negocio ya que se tienen que comunicar para recibir las solicitudes y presentar los resultados.

En esta capa creamos dos tipos de paquetes:

- Logica_Negocio_Clases
- Logica_Negocio_Metodos

En el paquete llamado Logica_Negocio_Clases existirán las siguientes clases:

1. **c_Docente**
2. **c_Horario**
3. **c_Laboratorio**
4. **c_ListarRegistro**
5. **c_Materia**
6. **c_Registro**

Para crear estas clases se deben realizar los pasos anteriormente indicados.

En el paquete llamado Logica_Negocio_Metodos existirán las siguientes clases:

1. **c_Docente_Metodo**
2. **c_Horario_Metodo**
3. **c_Laboratorio_Metodo**
4. **c_Materia_Metodo**
5. **c_Registro_Metodo**

Para crear estas clases se deben realizar los pasos anteriormente indicados.

Capa de Presentación

Se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable.

En esta capa crearemos el paquete llamado Interfaz_Usuario existirán los siguientes paneles:

1. **fr_Menu**

En este paquete crearemos la aplicación visual, es decir el menú principal de la aplicación, para esto sobre este paquete hacemos clic derecho y después clic en la opción *JFrame Form*, seleccionamos un nombre en este caso le pusimos fr_Menu una vez terminado nos aparecerá una forma visual de NetBeans como se muestra en la siguiente Figura IV.64.

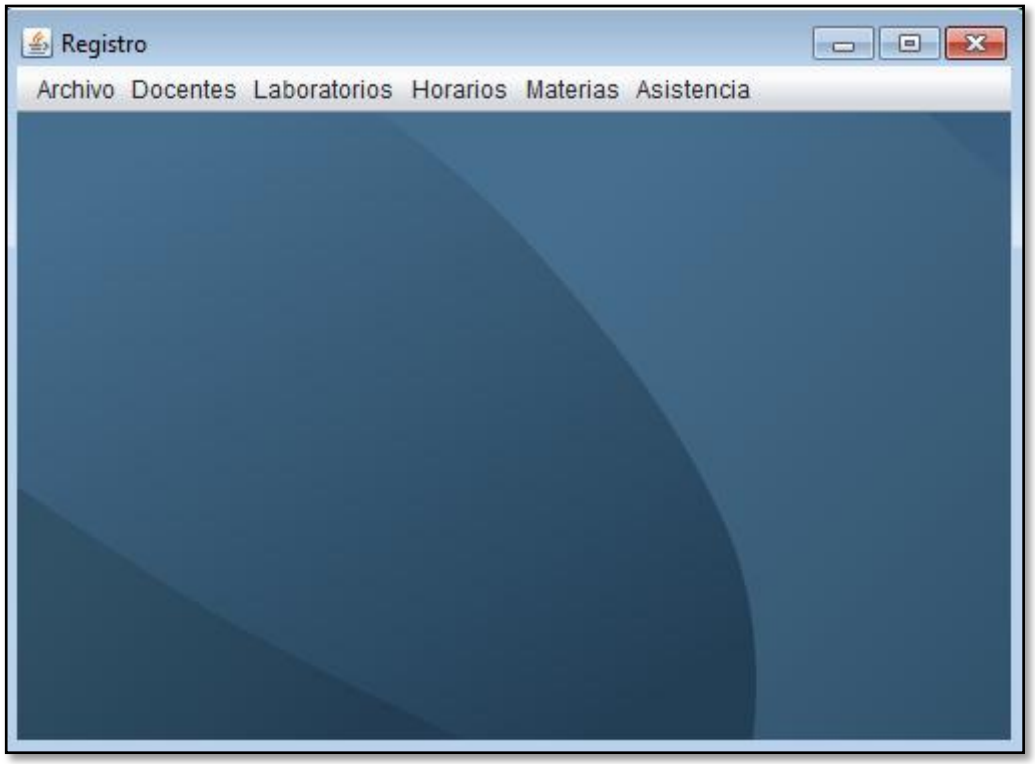


Figura IV.64: Menú Principal.

Fuente: Elaboración Propia

2. Registro_Docentes

En este panel registraremos los docentes que impartirán clases en las aulas de la academia CISCO, que deberá llenar los campos de manera obligatoria la id de la tarjeta del docente, la cedula y el nombre. Mientras que el número de celular es un campo opcional de llenar como se muestra en la siguiente Figura IV.65.

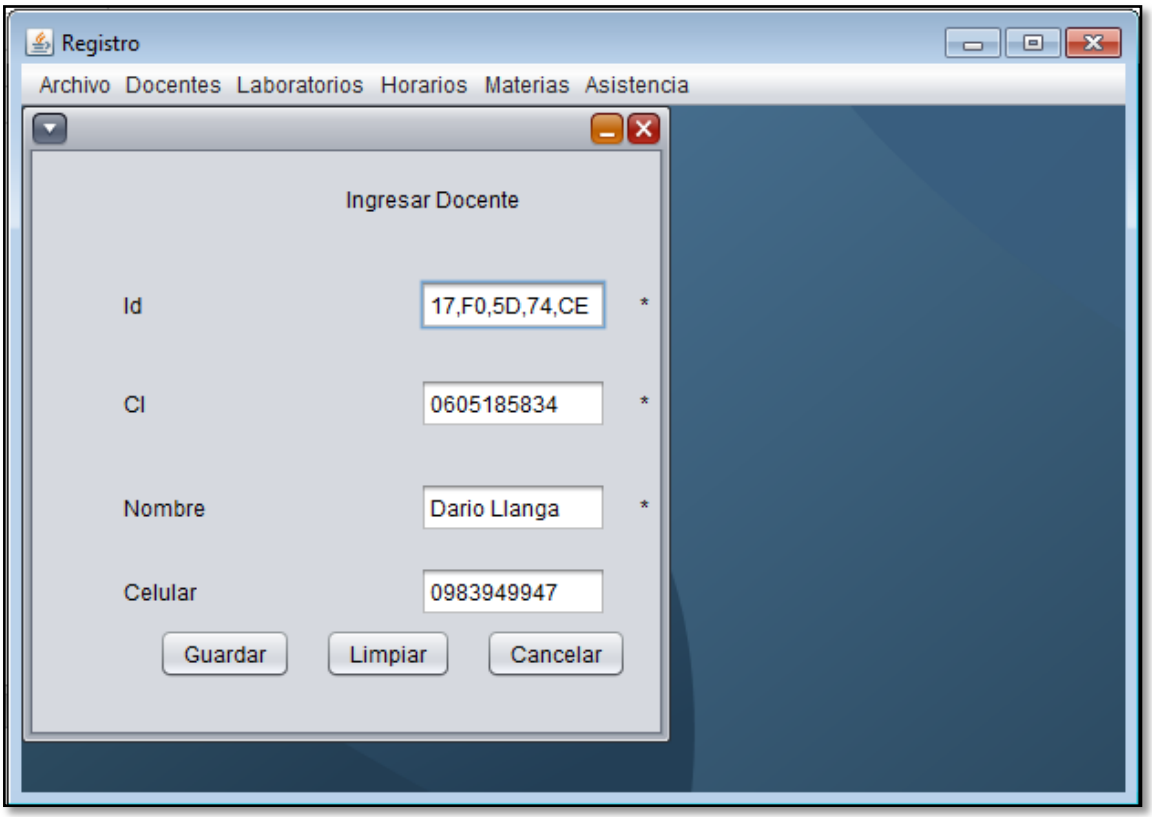


Figura IV.65: Ingresar Docente.

Fuente: Elaboración Propia

3. Registro_Laboratorios

En esta ventana registraremos todas las aulas de clases que existan en la academia CISCO, en donde los docentes impartirán sus horas de clase. Los campos de manera obligatoria de llenar es el Id que sirve para la identificación del laboratorio mientras que en el campo Nombre es el nombre del laboratorio, como podemos apreciar en la siguiente Figura IV.66.

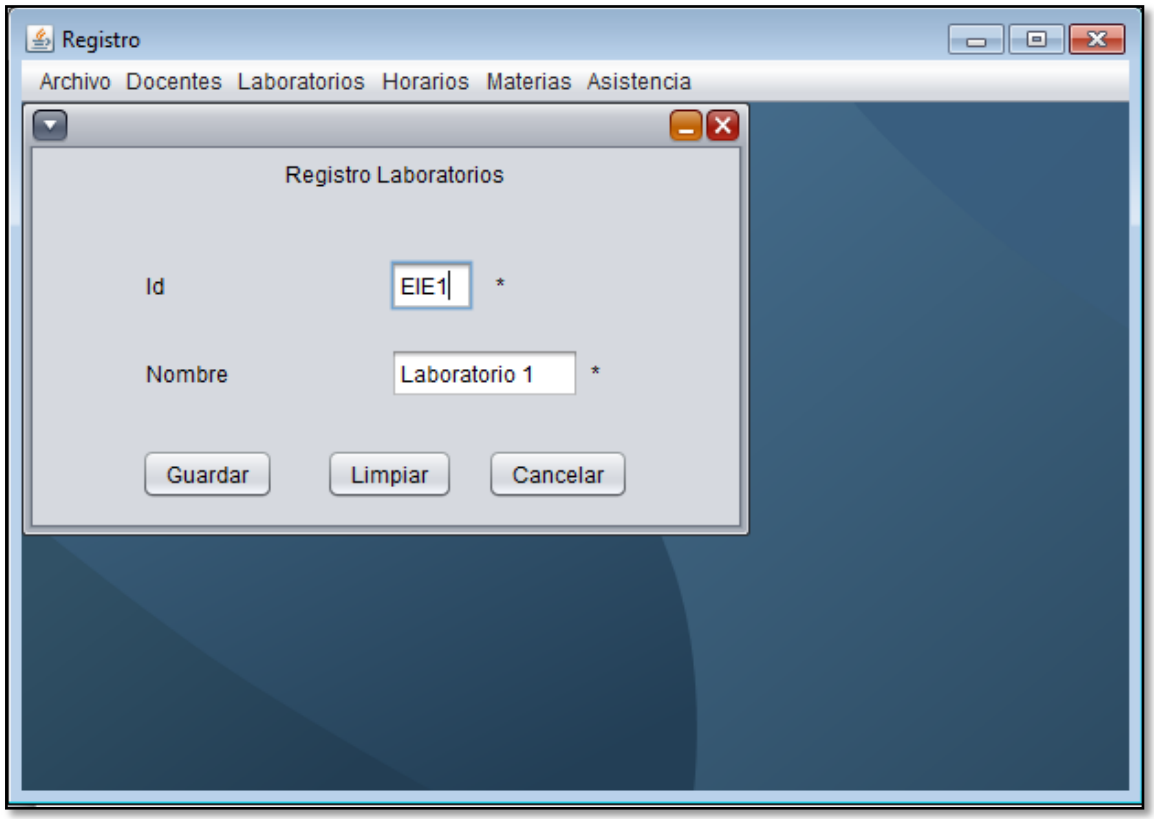


Figura IV.66: Ingresar Laboratorios.

Fuente: Elaboración Propia

4. Registro_Horarios

En esta ventana se registraran las horas de clase, con los siguientes campos obligatorios de llenar; el Id es la identificación de hora de clase que se va ingresar, en el campo Día se podrá escoger cualquier día de la semana excepto sábado y domingo, en el campo Hora de Inicio será la hora que iniciara la clase y en el campo Hora de salida es la hora que finalizara la clase, en el siguiente Figura IV.67 se muestra el registro de los horarios.

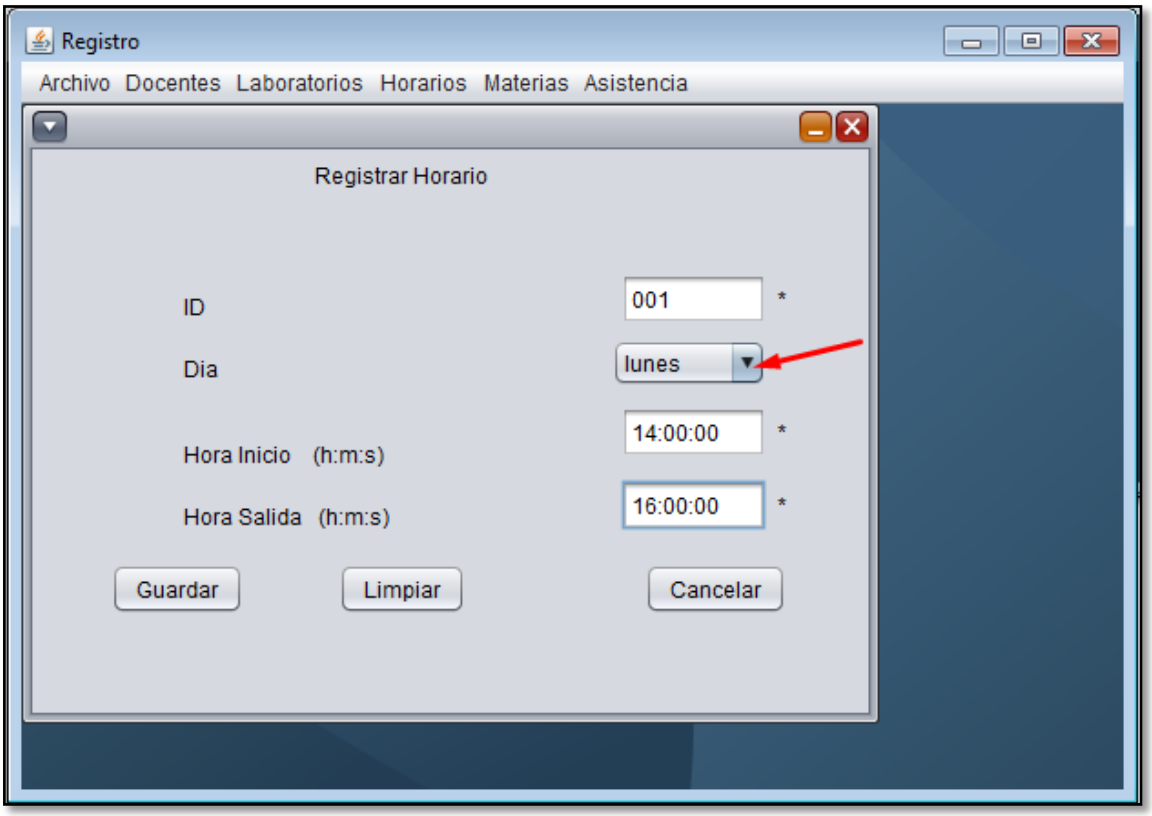


Figura IV.67: Registro de Horarios.

Fuente: Elaboración Propia

5. Registro_Materias

En esta ventana se le asignara una aula de clase un horario al docente, como se puede observar en la Figura IV.68 en la tabla Laboratorio deberá seleccionar uno de todos las aulas de clase que exista en la academia CISCO, en la tabla Horario deberá seleccionar el día y la hora que dará clases el docente y en la tabla Docente seleccionará al Docente. En el campo Id Materia es el identificador único del nombre de la materia y en el campo Nombre Materia se llenara con el nombre de la materia que ha sido asignada al docente.

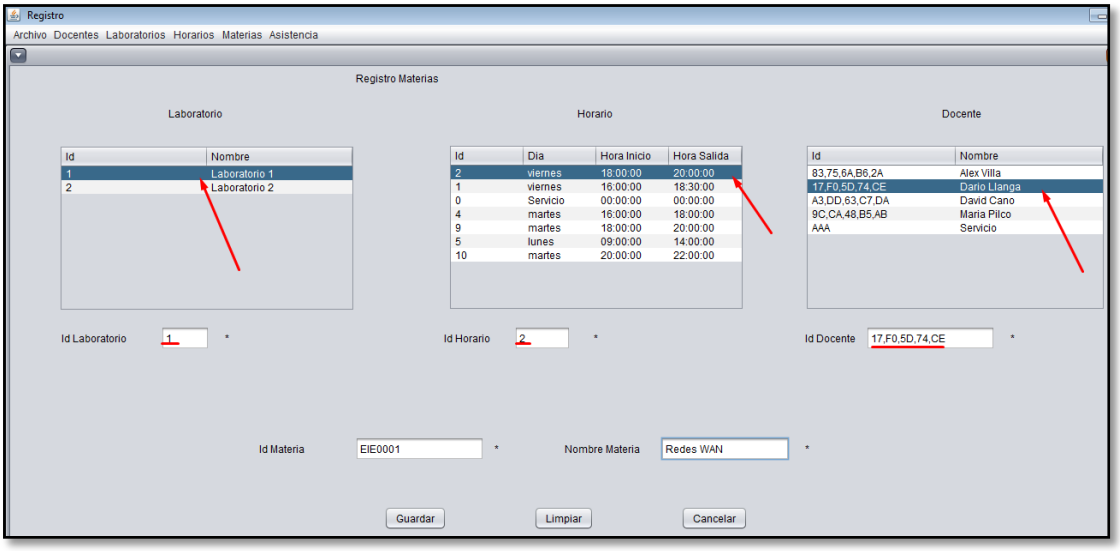


Figura IV.68: Registro de Materia.

Fuente: Elaboración Propia

6. Registro_Asistencia

En esta ventana es donde se despliega los datos recolectados por los lectores RFID, es decir en este cuadro me muestra los registro de las entradas y salidas de los docentes de las horas de clase, también me visualiza el número de cedula, el nombre de la materia, la fecha del registro como se muestra en la Figura IV.69.

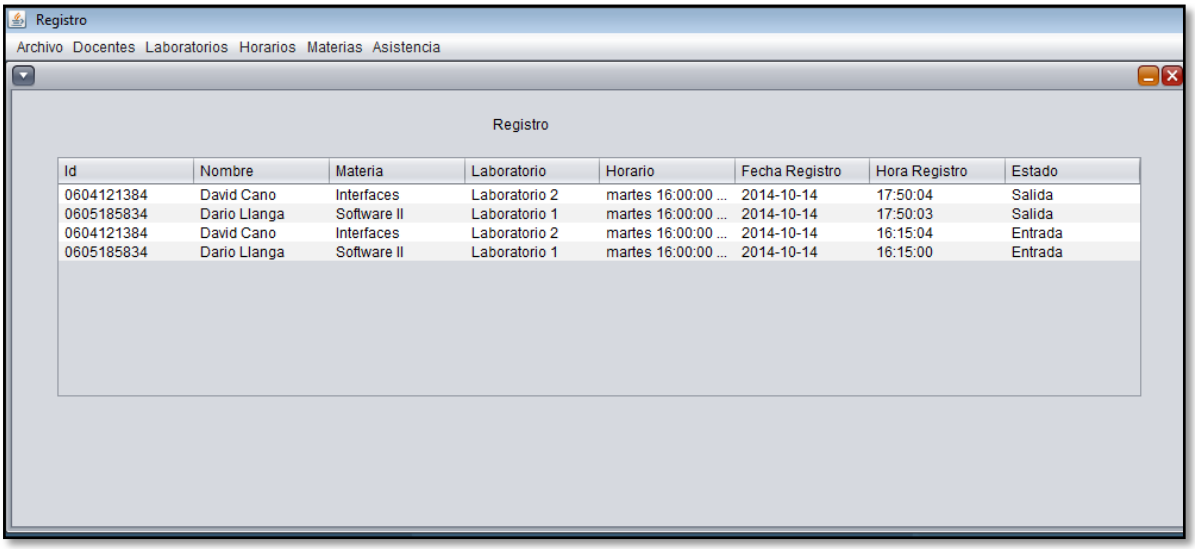
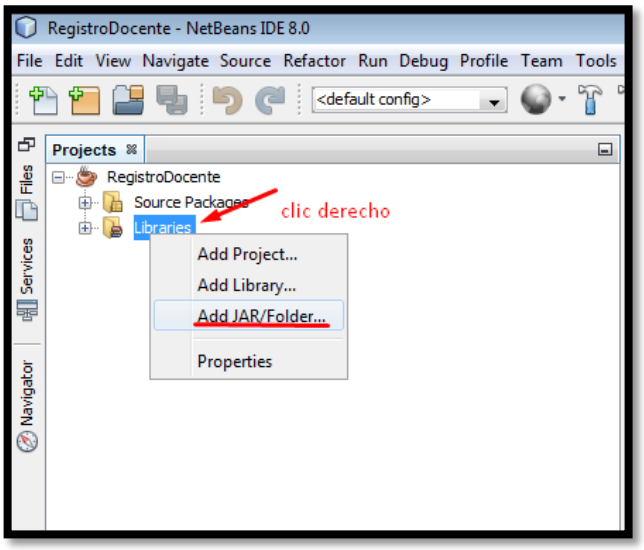


Figura IV.69: Registro de Docente.

Fuente: Elaboración Propia

El código de la programación de la aplicación en java se encuentra en ANEXOS 3.

Para que exista la conexión del Arduino a la aplicación java hace falta de unas librerías que anteriormente ya nos descargamos del internet. Para añadir la librería descargad nos dirigimos al proyecto creado, clic derechos sobre la pestaña que tiene el nombre de *Libraries,* elegimos *Add JAR/Folder* y luego seleccionamos el archivo que nos descargamos, así como se muestra en la siguiente Figura IV.70.



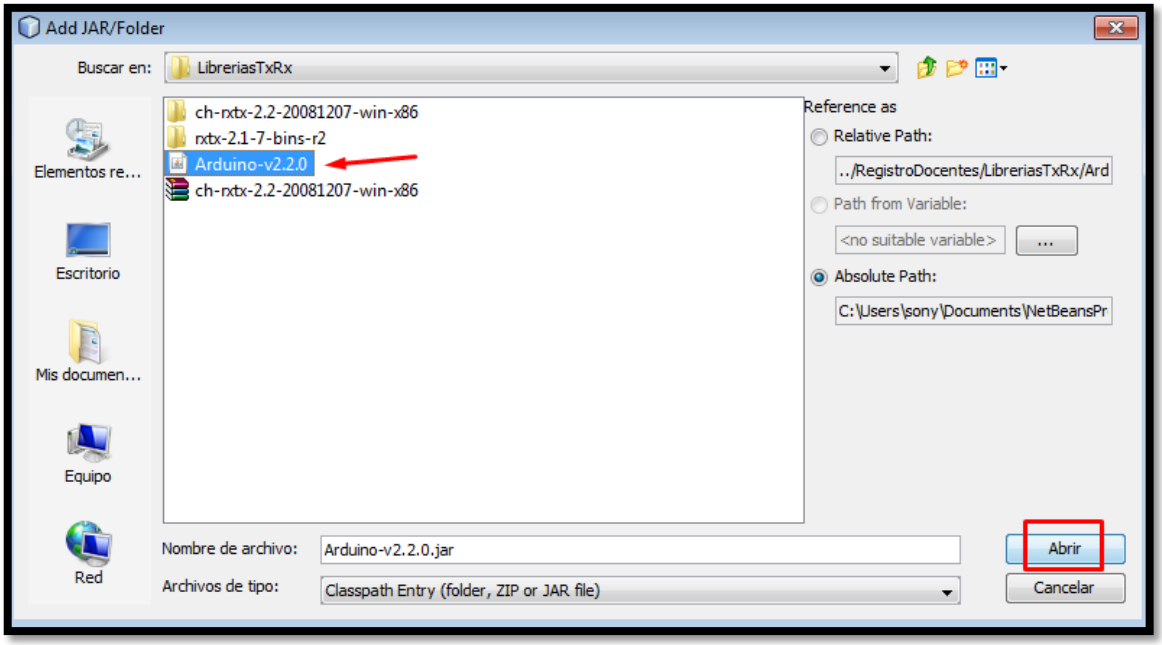


Figura IV.70: Importar librería Arduino en NetBeans IDE.

Fuente: Elaboración Propia

4.3. EVALUACION DEL FUNCIONAMIENTO DEL SISTEMA

Para la evaluación del funcionamiento del sistema se realizaron pruebas simulando un ambiente de trabajo real.

4.3.1. Políticas de Funcionamiento

Las políticas de funcionamiento es la guía básica que determinan las acciones que debe cumplir el docente de la academia CISCO para el correcto funcionamiento del sistema de control de acceso docente.

- ❖ Todos los docentes que laboren en las aulas de la academia CISCO deberán tener su tarjeta o llavero RFID de identificación.
- ❖ El docente deberá realizar dos registros por la hora de clase, la primera es el registro de entrada y la segunda es el registro de salida.

- El docente deberá estar puntual a la hora de clase para que realice su registro de entrada, acercando su tarjeta al lector RFID que se encuentra en el aula.
- El docente no podrá registrarse en un aula que no pertenezca en su horario de clases.
- Para realizar el registro de salida el docente deberá pasar su tarjeta sobre el lector RFID durante los últimos quince minutos de su hora de clase para que sea válido como registro de salida.
- Si el docente no registro la salida durante los últimos quince minutos constara como ausente.

4.3.2. Recolección y Análisis de datos

Los lectores RFID se colocaron en cajas diseñadas para evitar daños en los circuitos, la antena del módulo de comunicación XBee se puede visualizar por la parte superior de la caja para lograr que la señal no se atenué y alcance mayor distancia. Cada Lector RFID se colocara en cada Laboratorio de la academia CISCO como se observa en la siguiente Figura IV.71.

Figura IV.71: Lectores RFID.
Fuente: Elaboración Propia

Se ubicó los nodos de la siguiente forma:

- Nodo final 1 :Laboratorio 1 de CISCO
- Nodo final 2 :Laboratorio 2 de CISCO
- Nodo Coordinador :Oficina del encargado de CISCO

El nodo coordinador estará en la oficina del encargado de CISCO que solo cuenta con un módulo XBee y estará conectado a una máquina que contiene la aplicación que se creó.

Para la prueba del prototipo utilizaremos cuatro tarjetas RFID que se les asignara nombres como se muestra en la siguiente Tabla IV.20.

Usuarios del Prototipo		
Nombre	**Código**	**Identificador**
María Pilco	**9C,CA,48,B5,AB**	**TARJETA**
José Caiza	**17,F0,5D,74,CE**	**TARJETA**
Alex Villa	**83,75,6A,B6,2A**	**LLAVERO**
Ángel Pérez	**A3,DD,63,C7,DA**	**LLAVERO**
Andrea Escobar	**88,4,DA,1E,48**	**ETIQUETA**

Tabla IV.20: Usuarios del prototipo.
Fuente: Elaboración Propia

El horario de clases para el laboratorio 1 es el siguiente como se muestra en la Tabla VI.21.

HORARIO LABORATORIO 1					
Hora	Lunes	Martes	Miércoles	Jueves	Viernes
14h00 - 16h00	José Caiza Redes WAN	María Pilco Telemática	Ángel Pérez Televisión Digital	María Pilco Telemática	Alex Villa Ruteo I
16h00- 18h00	Alex Villa Ruteo I	José Caiza Redes WAN	Andrea Escobar Aplicaciones telemáticas	José Caiza Redes WAN	José Caiza Redes WAN

Tabla IV.21: Horario de clases L1.
Fuente: Elaboración Propia

El horario de clases para el laboratorio 2 es el siguiente como se muestra en la Tabla VI.22.

HORARIO LABORATORIO 2					
Hora	Lunes	Martes	Miércoles	Jueves	Viernes
14h00 - 16h00	Alex Villa Ruteo I		María Pilco Telemática	José Caiza Redes WAN	María Pilco Telemática
16h00- 18h00	María Pilco Telemática	Ángel Pérez Televisión Digital	Alex Villa Ruteo I	Ángel Pérez Televisión Digital	Ángel Pérez Televisión Digital

Tabla IV.22: Horario de clases L2.
Fuente: Elaboración Propia

Una vez montada el sistema ya empieza a tomar datos y tomando en cuenta las políticas de funcionamiento y la recolección de datos se logra los siguientes registros en la siguiente Tabla IV.23.

CEDULA	NOMBRE	MATERIA	AULA	HORARIO	FECHA DE REGISTRO	HORA DE REGISTRO	ESTADO
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 2	viernes 16:00:00 a 18:00:00	2014-10-31	17:46:05	SALIDA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 2	viernes 16:00:00 a 18:00:00	2014-10-31	16:15:03	ENTRADA
0605185834	José Caiza	Seguridad Redes	Laboratorio 1	viernes 16:00:00 a 18:00:00	2014-10-31	16:15:01	ENTRADA
0204758264	María Pilco	Telemática	Laboratorio 2	viernes 14:00:00 a 16:00:00	2014-10-31	15:49:05	SALIDA
0604121483	Alex Villa	Ruteo I	Laboratorio 1	viernes 14:00:00 a 16:00:00	2014-10-31	15:49:04	SALIDA
0204758264	María Pilco	Telemática	Laboratorio 2	viernes 14:00:00 a 16:00:00	2014-10-31	14:02:01	ENTRADA
0604121483	Alex Villa	Ruteo I	Laboratorio 1	viernes 14:00:00 a 16:00:00	2014-10-31	14:02:00	ENTRADA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 2	jueves 16:00:00 a 18:00:00	2014-10-30	17:50:01	SALIDA
0605185834	José Caiza	Redes WAN	Laboratorio 1	jueves 16:00:00 a 18:00:00	2014-10-30	17:50:00	SALIDA
0605185834	José Caiza	Redes WAN	Laboratorio 2	jueves 14:00:00 a 16:00:00	2014-10-30	16:04:34	INASISTENCIA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 2	jueves 16:00:00 a 18:00:00	2014-10-30	16:04:03	ENTRADA
0605185834	José Caiza	Redes WAN	Laboratorio 1	jueves 16:00:00 a 18:00:00	2014-10-30	16:04:02	ENTRADA
0204758264	María Pilco	Telemática	Laboratorio 1	jueves 14:00:00 a 16:00:00	2014-10-30	15:49:01	SALIDA
0204758264	María Pilco	Telemática	Laboratorio 1	jueves 14:00:00 a 16:00:00	2014-10-30	14:13:05	ENTRADA
0604121483	Alex Villa	Ruteo I	Laboratorio 2	miércoles 16:00:00 a 18:00:00	2014-10-29	17:56:03	SALIDA
0204758264	María Pilco	Telemática	Laboratorio 2	miércoles 14:00:00 a 16:00:00	2014-10-29	16:04:13	INASISTENCIA
0604121483	Alex Villa	Ruteo I	Laboratorio 2	miércoles 16:00:00 a 18:00:00	2014-10-29	16:04:01	ENTRADA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 1	miércoles 14:00:00 a 16:00:00	2014-10-29	15:56:02	SALIDA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 1	miércoles 14:00:00 a 16:00:00	2014-10-29	14:16:03	ENTRADA
0604121384	Ángel Pérez	Televisión	Laboratorio 2	martes	2014-10-28	17:51:04	SALIDA

		Digital		16:00:00 a 18:00:00			
0605185834	José Caiza	Redes WAN	Laboratorio 1	martes 16:00:00 a 18:00:00	2014-10-28	17:51:03	SALIDA
0604121384	Ángel Pérez	Televisión Digital	Laboratorio 2	martes 16:00:00 a 18:00:00	2014-10-28	16:23:05	ENTRADA
0605185834	José Caiza	Redes WAN	Laboratorio 1	martes 16:00:00 a 18:00:00	2014-10-28	16:08:03	ENTRADA
0204758264	María Pilco	Telemática	Laboratorio 1	martes 14:00:00 a 16:00:00	2014-10-28	15:55:01	SALIDA
0204758264	María Pilco	Telemática	Laboratorio 1	martes 14:00:00 a 16:00:00	2014-10-28	14:02:01	ENTRADA
0204758264	María Pilco	Telemática	Laboratorio 2	lunes 16:00:00 a 18:00:00	2014-10-27	17:57:04	SALIDA
0604121483	Alex Villa	Ruteo I	Laboratorio 1	lunes 16:00:00 a 18:00:00	2014-10-27	17:50:02	SALIDA
0604121483	Alex Villa	Ruteo I	Laboratorio 1	lunes 16:00:00 a 18:00:00	2014-10-27	16:08:04	ENTRADA
0204758264	María Pilco	Telemática	Laboratorio 2	lunes 16:00:00 a 18:00:00	2014-10-27	16:08:02	ENTRADA
0605185834	José Caiza	Redes WAN	Laboratorio 1	lunes 14:00:00 a 16:00:00	2014-10-27	15:55:05	SALIDA
0604121483	Alex Villa	Ruteo I	Laboratorio 2	lunes 14:00:00 a 16:00:00	2014-10-27	15:49:03	SALIDA
0604121483	Alex Villa	Ruteo I	Laboratorio 2	lunes 14:00:00 a 16:00:00	2014-10-27	14:11:00	ENTRADA
0605185834	José Caiza	Redes WAN	Laboratorio 1	lunes 14:00:00 a 16:00:00	2014-10-27	14:06:03	ENTRADA

Tabla IV.23: Registros obtenidos del sistema.
Fuente: Elaboración Propia

Las pruebas realizadas durante esta semana se pudo obtener como resultado que la usuario María Pilco con cedula 0204758264 que dicta la materia de Telemática que el día miércoles 29 de octubre no asistió a dictar su hora de clase, de la misma manera el usuario José Caiza con cedula 0605185834 no asistió a dictar su hora de clase el 30 de octubre en el laboratorio 2.

El director de CISCO realiza el control de asistencia a los docentes dos veces al día, la primera a las 11:00 y la segunda a las 16h00 y lo representamos en la siguiente Tabla IV.24.

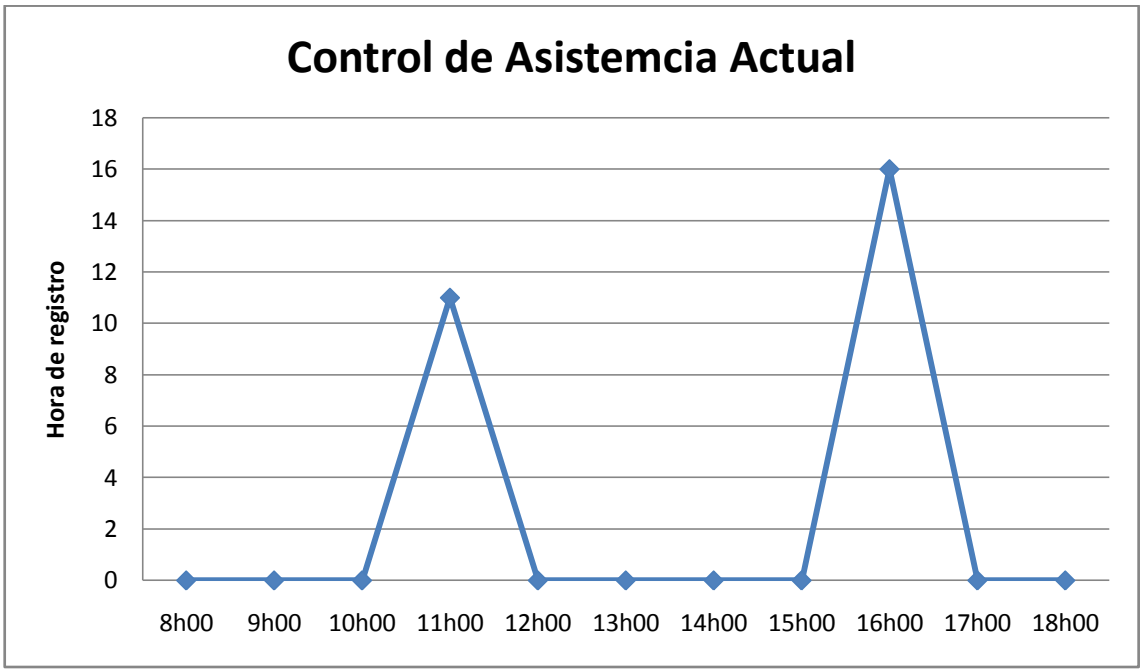


Tabla IV.24: Control Asistencia Actual.
Fuente: Elaboración Propia

El coeficiente de variación permite comparar la variación que existe entre los diferentes sistemas de control de asistencia.

Para el control de asistencia actual es:

$$CV = \frac{\text{NR}}{\text{NHCD}} * 100\%$$

Donde:
NR= Número de registros al día
NHCD= Número de horas de clases al día

$$CV = \frac{2}{8} * 100\%$$

$$CV = 25\%$$

Es decir que tiene un 25% de eficiencia en el sistema.

Con el sistema creado el control de acceso se lo puede representar en la siguiente Tabla IV.25.

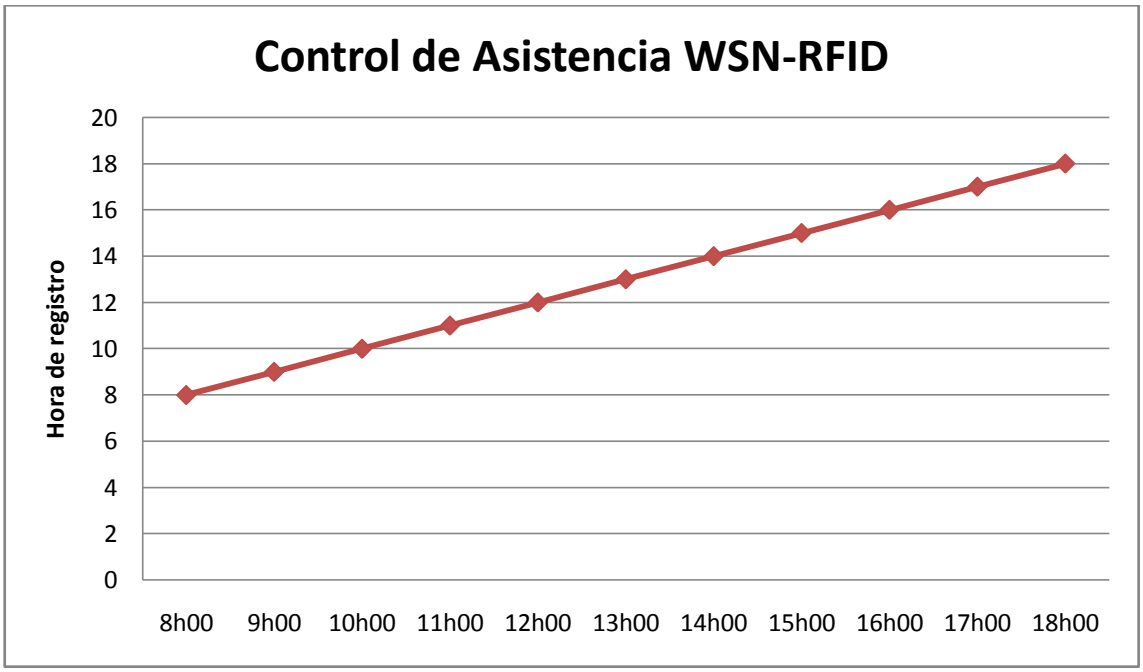


Tabla IV.25: Control Asistencia WSN-RFID.
Fuente: Elaboración Propia

Para el control de asistencia WSN-RFID es:

$$CV = \frac{\text{NR}}{\text{NHCD}} * 100\%$$

$$CV = \frac{8}{8} * 100\%$$

$$CV = 100\%$$

Es decir que tiene el 100% de eficiencia en el sistema, es decir que el sistema actual es ineficiente ya que existen lapsos de tiempo que no se realiza el registro docente a diferencia del sistema WSN–RFID que durante todo el tiempo realiza el registro de los docentes.

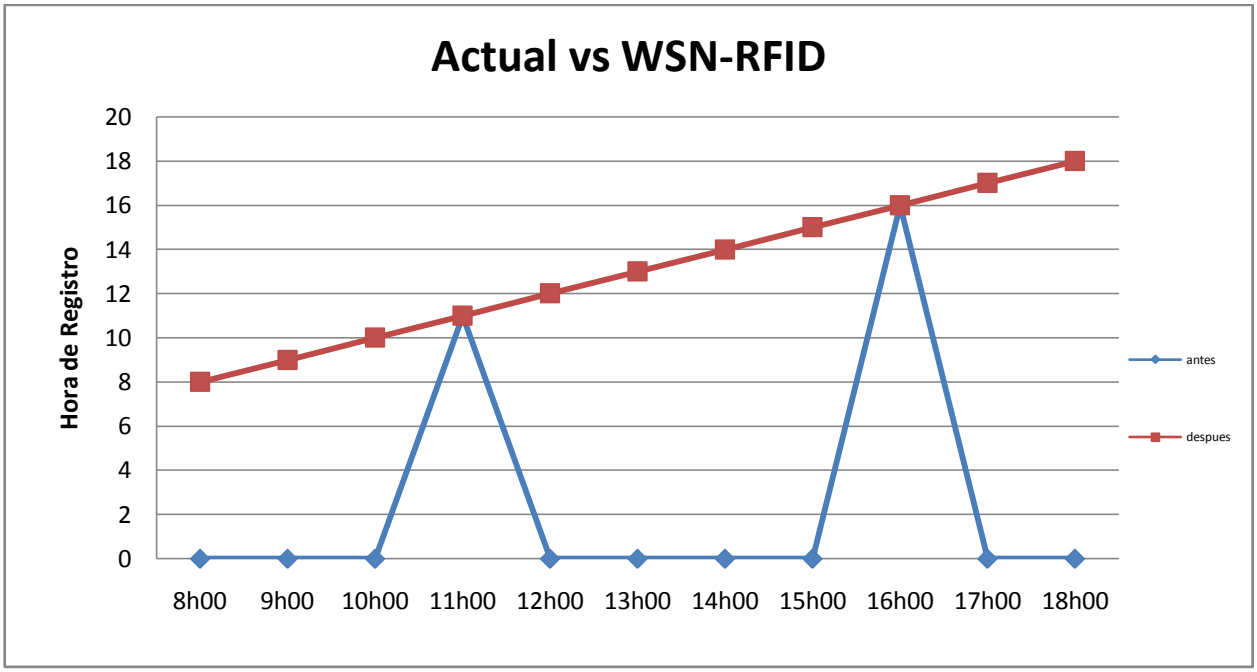


Tabla IV.26: Sistema Actual vs WSN-RFID.
Fuente: Elaboración Propia

4.3.3. Comprobación de la Hipótesis

Al analizar los resultados obtenidos del sistema de control de acceso como se muestra en la Tabla IV.26 se pudo comprobar que la interoperabilidad de las tecnologías WSN y RFID permitió el control de acceso de los Docente de la academia CISCO-ESPOCH.

CONCLUSIONES

1. Se realizó la comparación de distintos protocolos de comunicación inalámbrica y se pudo determinar que el protocolo ZigBee en comparación con otra tecnología inalámbrica como Bluetooth, se ajusta a los requerimientos de nuestro sistema consumiendo un 98.5% menos de energía en modo reposo y permitiéndonos crear una red de hasta 65535 nodos.
2. Mediante la integración de etiquetas y lectores RFID en nodos WSN, se logró desarrollar un sistema WSN-RFID para el control de acceso docente robusto y preciso.
3. Además se realizó la comparación de los sistemas de control de acceso como: WSN-RFID, biométrico, dactilar, y facial obteniendo el sistema propuesto un 95% de eficiencia frente a los demás sistemas que existen actualmente en el mercado.
4. La implementación del prototipo nos permitió recolectar y manipular la información a través de una aplicación en JAVA conectada a una base de datos que almacena los registros obtenidos por el prototipo lo que nos permite obtener registros en tiempo real.
5. Al evaluar el desempeño de la red WSN y RFID mediante pruebas realizadas en los laboratorios se obtuvo como resultado que el sistema logró un control de acceso y registro docente con un 100% de efectividad frente al 25% del método utilizado actualmente en la academia CISCO-ESPOCH.

RECOMENDACIONES

1. Para que no exista interferencia entre protocolos de comunicación inalámbrica que trabajan en la banda de 2.4GHz es recomendable configurar a ZigBee en un canal diferente, ya que en ZigBee se definen hasta 16 canales en el rango de 2.4 GHz, cada uno de ellos con un ancho de banda de 5 MHz.

2. Se recomienda tomar en cuenta todos los requerimientos necesarios para el desarrollo del prototipo, evitando así la incorrecta adquisición de los equipos.

3. Al momento de realizar la base de datos se recomienda la creación de una aplicación que pueda interpretar los datos enviados por el Arduino para que puedan ser interpretados por el administrador.

4. Realizar las pruebas en un ambiente de trabajo real, ya que el funcionamiento de los equipos no es el mismo que el especificado por los fabricantes, como es el caso del XBee que su alcance es menor al especificado en su datasheet.

5. Es fundamental utilizar software y hardware libre ya que esto nos permitirá desarrollar un sistema más flexible y a un costo muy bajo.

BIBLIOGRAFÍA

1. Ahmad, K. (2008). Wireless Sensor Networks as part of a Web-Based Building Environmental Monitoring System. *Revista Automation in Construction*, 1, pp. 729–736.
2. Banzi, M. (2012). *Introducción a Arduino*. Madrid-España: RC libros.
3. Gislason, D. (2008). *Zigbee Wireless Networking*. Washington-Estados Unidos: Elsevier.
4. Syed, A., Mohammad, I. (2008). *RFID Handbook* (1a. ed.). Lahore-Pakistan: CRC Press.

BIBLIOGRAFÍA DE INTERNET

[1] DABNE, (2006). *Sistema De Radiolocalización Y Telemetría*. Extraído el 17 de marzo del 2014 desde http://www.dabne.net/IMG/pdf/informe-gps-dabne-sin-imagenes.pdf

[2] Morgan, C. (2012, 05 de Julio). *Intel's Sensors Will Warn You About Running Outside When The Air Is Polluted*. Extraído el 17 de marzo del 2014 desde http://www.fastcoexist.com/1680111/intels-sensors-will-warn-you-about-running-outside-when-the-air-is-polluted

[3] Iacono, L., Godoy, P., Marianetti, O., García, C., & Párraga, C. (2012, mayo). *Estudio de la Integración entre WSN y redes TCP/IP*. Extraído el 17 de marzo del 2014, desde http://www.um.edu.uy/docs/5_estudio_de_la_integracion_entre_WSN_redes%20TCP_IP.pdf

[4] Pérez, E. (s.f.). *Sensor Network Protocols*. Extraído el 17 de marzo del 2014, desde http://www.sensorprotocol.com/trabajos3/redcom/redcom.shtml

[5] Blázquez, M. (s.f.). *Sistema De Identificacion Por Radio Frecuencia*. Extraído el 18 de marzo del 2014, desde http://www.it.uc3m.es/jmb/RFID/rfid.pdf

[6] Garzón, C. (2010, octubre). *Análisis Y Estudio De Impacto De La Tecnología ZigBee Aplicado A La Domótica En El Ecuador*. Quito-Ecuador. Extraído el 18 de marzo del 2014, desde http://dspace.ups.edu.ec/bitstream/123456789/4395/1/UPS-ST000171.pdf

[7] Marcillo, O., Johnson, J. (s.f.). *Ecuador With A Wireless Sensor Network.* Extraído el 18 de marzo del 2014, desde http://www.eecs.harvard.edu/~mdw/proj/volcano/reventador-report.pdf.

[8] Pallarés, S. (2009, Marzo). *Sistema De Localización Para Redes Inalámbricas De Sensores Mediante ZigBee.* Extraído el 18 de marzo del 2014, desde http://forja.uji.es/docman/view.php/60/85/Memoria_revisada_Pallares_Gual_Sandra.pdf

[9] Randell, C., Muller, H. (s.f.). *Low Cost Indoor Positioning System*. Extraído el 19 de marzo del 2014, desde http://www.cs.bris.ac.uk/Publications/Papers/1000573.pdf

[10] García, J. (2014, enero). *Beeme Y Su Apuesta Por La Localización En Interiores Con Proximus.* Extraído el 19 de marzo del 2014, desde https://www.centrodeinnovacionbbva.com/noticias/tb/29820-beeme-y-su-apuesta-por-la-localizacion-en-interiores-con-proximus

[11] Schilan, R. (2011). *Seminario De Formulación Y Gestión Del Proyecto De Investigación.* Extraído el 19 de marzo del 2014, desde http://ffyl.uncu.edu.ar/IMG/pdf/De_la_idea_al_problema_2011.pdf

[12] Josueax, A. (2013, octubre). *Razonamiento Inductivo*. Extraído el 20 de marzo del 2014, desde http://clubensayos.com/Historia/Razonamiento-Inductivo/1107434.html

[13] Pérez, B., Pittier, A., Travieso, M., & Wodzislawski, M. (s.f.). *Testing Exploratorio En La Práctica.* Extraído el 20 de marzo del 2014, desde http://www.ces.com.uy/documentos/imasd/CES-Testing_Exploratorio.pdf

[14] Informática Redes. (2012. Julio). *Redes De Sensores Inalámbricos Wsn.* Extraído el 20 de marzo del 2014, desde http://informaticaredes2012.blogspot.com.es/

[15] Cianca, K. (2012, Julio*). Características De Una Red Sensores*. Extraído el 20 de marzo del 2014, desde http://lasredesconsensores.blogspot.com/2012_07_01_archive.html

[16] Fernández, M., Ordieres, J., Martinez, J., González, A., & Pernía, A. (2009). *Wsn Teoría Y Aplicación Práctica.* Extraído el 20 de marzo del 2014, desde http://dialnet.unirioja.es/descarga/libro/377564.pdf

[17] Olivia. Jesus. Emilio. (s.f.). *The Last Lab Project.* . Extraído el 12 de mayo del 2014, desde http://thelastlabproject.blogspot.com/2010/12/clasificacion-de-los-sensores.html

[18] 5 Hertz Electrónica. (s.f.). *Xbee 2mw Antena De Cable Serie 2 ZigBee Mesh*. Extraído el 17 de junio del 2014, desde http://5hertz.com/index.php?main_page=product_info&products_id=322

[19] Ernes, M. (2012, julio). *La Tecnología Rfid.* Extraído el 17 de junio del 2014, desde http://anidirection.blogspot.com/feeds/posts/default

[20] Portillo, J., Bermejo, A., Bernardos, A. (2008). *Tecnología De Identificación Por Radiofrecuencia RFID.* Extraído el 20 de junio del 2014, desde http://www.madrimasd.org/informacionidi/biblioteca/publicacion/Vigilancia-tecnologica/descargar_documentos/fichero.asp?id=VT13_RFID.pdf

[21] ISO. (2009). *So / Iec Jtc 1 / Sc 6 Telecomunicaciones E Intercambio De Información Entre Sistemas.* Extraído el 20 de junio del 2014, desde http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/iso_technical_committee.htm?commid=45072

[22] Duarte, A. (s.f.). *Taller Teórico-Práctico De Arduino.* Extraído el 20 de junio del 2014, desde http://www.andresduarte.com/arduino-y-xbee

[23] XBee.Cl . (s.f.). *Familias XBee Serie 1, XBee Serie 2 Y XBee 900*. Extraído el 20 de junio del 2014, desde http://www.xbee.cl/diferencias.html

[24] 5 Hertz Electrónica. (s.f.). *XBee Shield Para Arduino*. Extraído el 21 de junio del 2014, desde http://5hertz.com/index.php?main_page=product_info&products_id=496

[25] Arduino.CC. (s.f.). *Página Oficial Arduino*. Extraído el 21 de junio del 2014, desde http://arduino.cc/es/Guide/Introduction

[26] Wikipedia Libre. (2014, Noviembre). *Arduino Plataforma De Hardware Libre.* Extraído el 22 de junio del 2014, desde http://es.wikipedia.org/wiki/Arduino

[27] Pomares, J. (2009). *Manual De Programación Arduino_Jorge Pomares.* Extraído el 23 de junio del 2014, desde http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf

[28] Moreno, S. (2010). *Desarrollo De Un Entorno Para La configuración Y Monitorización De Redes ZigBee/802.15.4.* Extraído el 23 de junio del 2014, desde http://webpersonal.uma.es/~ECASILARI/Docencia/Memorias_Presentaciones_PFC/54_MemoriaSergioLilloMoreno.pdf

[29] ITESCAM. (2012). *El Estándar IEEE 802.15.1 – Bluetooth.* Extraído el 23 de julio del 2014, desde http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r101544.PDF

[30] Flores, E. (2012, octubre*). Redes De Sensores Inalámbricas Aplicado A La Medicina (Wireless Sensor Networks Applied To The Medicine).* Extraído el 23 de julio del 2014, desde http://bucserver01.unican.es/xmlui/bitstream/handle/10902/1288/349251.pdf.txt?sequence

[31] Jui, S. (2005, noviembre). *Comparación De Las Tecnologías De Control De Acceso A Las Instalaciones En Una Organización.* Extraído el 23 de julio del 2014, desde http://biblioteca.usac.edu.gt/tesis/08/08_0264_CS.pdf

[32] Zone Digital Ecuador (s.f.). *Venta de Equipos de Seguridad*. Extraído el 23 de agosto del 2014, desde http://zonedigitalecuador.net/index.php/equipo-seguridad

[33] Pagina Oficial Arduino (s.f.). *Descargue el software de Arduino*. Extraído el 23 de agosto del 2014, desde http://arduino.cc/en/pmwiki.php?n=main/software

[34] Pagina Oficial DIGI (s.f.). *CTU-X software (XCTU).* Extraído el 23 de agosto del 2014, desde http://arduino.cc/en/pmwiki.php?n=main/software

[35] PostgresSQL (s.f.). *PostgreSQL Core Distribución.* Extraído el 23 de agosto del 2014, desde http://www.postgresql.org/download/

[36] ORACLE (s.f.). *Java SE Descargas* .Extraído el 23 de agosto del 2014, desde http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

[37] NetBeans (s.f.). *NetBeans IDE 8.0.2.* Extraído el 23 de agosto del 2014, desde https://netbeans.org/downloads/

[38] García, A. (2014). Librería *Arduino para JAVA* .Extraído el 23 de agosto del 2014, desde http://panamahitek.com/libreria-arduino-para-java/

RESUMEN

Se realizó la implementación de un prototipo de sistema de control de acceso mediante la interoperabilidad de las tecnologías WSN y RFID para automatizar el registro de acceso docente a los laboratorios de la academia CISCO-ESPOCH.

La investigación se efectuó mediante el método científico aplicativo, donde se pudo conocer las variables que intervienen en el registro como son: horarios, materias, aulas y docentes.

Se comparó los protocolos de comunicación inalámbrica ZigBee y Bluetooth resultando el protocolo ZigBee que más se adaptó a nuestro sistema. Además se realizó la comparación de los sistema de control de acceso como: WSN-RFID, biométrico, dactilar y facial obteniendo el sistema propuesto un 95% de eficiencia frente a los demás analizados. Para el diseño e implementación del prototipo sistema WSN y RFID, se construyó dos nodos finales y un nodo coordinador basados en Arduino que envía la información hacia una base de datos, desarrollada mediante el software NetBeans, JAVA, PgAdmin III y PostgreSQL.

Como resultado se obtuvo que el sistema propuesto brindará un control de registro docente con un 100% de efectividad, frente al 25% del método utilizado actualmente en la academia CISCO-ESPOCH

Estas dos tecnologías heterogéneas, pueden ser integradas en los sistemas de control de acceso permitiendo un registro confiable y en tiempo real. Se recomienda implementar este sistema en todas las aulas de la EIE-TR de la ESPOCH debido a su bajo costo y alta efectividad.

ABSTRACT

The implementation of prototype of an access control system was done by means of the interoperability of technologies WSN and RFID for automating the Access log of teachers to the laboratories of the CISCO-ESPOCH Academy.

The research is carried out by the explanatory scientific method where is possible to know the variables involved in the registration materials schedules classrooms and teachers.

The wireless communication protocols ZigBee and Bluetooth low energy consumption and parameters such as number of network nodes were compared, resulting that the ZigBee protocol is the most adapted to our system with a 98.5% reduction in energy consumption and a network node with 99.98% more than other nodes. For the design and the implementation of the prototype system WSN and RFID, two end nodes and a coordinator were built, the node based on Arduino sends the information to a database developed using NetBeans, Java, PostgreSQL and PgAdmin III software.

The obtained results showed that the proposed system will give us control of teacher registration a 100% effective, compared to 25% of the method currently used in the CISCO-ESPOCH academy.

These two heterogeneous technologies can be integrated into the access control system allowing a reliable and real-time control record. It is recommended to implement this system in all EIE-TR classrooms of the ESPOCH because of its low cost and effectiveness.

ANEXOS

- **ANEXO 1**
  - **Datasheet RC522**

MFRC522

Contactless Reader IC

Rev. 3.2 — 22 May 2007
112132

Product data sheet
PUBLIC INFORMATION

1. Introduction

This document describes the functionality of the contactless reader/writer MFRC522. It includes the functional and electrical specifications.

2. General description

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO 14443A / MIFARE® mode.

The MFRC522's internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE®Classic (e.g. MIFARE® Standard) products. The MFRC522 supports contactless communication using MIFARE® higher transfer speeds up to 848 kbit/s in both directions.

Various host interfaces are implemented:

- SPI interface
- serial UART (similar to RS232 with voltage levels according pad voltage supply)
- I^2C interface.

3. Features

- Highly integrated analog circuitry to demodulate and decode responses
- Buffered output drivers to connect an antenna with minimum number of external components
- Supports ISO/IEC 14443A / MIFARE®
- Typical operating distance in Reader/Writer mode for communication to a ISO/IEC 14443A / MIFARE® up to 50 mm depending on the antenna size and tuning
- Supports MIFARE® Classic encryption in Reader/Writer mode
- Supports ISO/IEC 14443A higher transfer speed communication up to 848 kbit/s
- Support of the MFIN / MFOUT
- Additional power supply to directly supply the smart card IC connected via MFIN / MFOUT
- Supported host interfaces
  - SPI interface up to 10 Mbit/s
  - I^2C interface up to 400 kbit/s in Fast mode, up to 3400 kbit/s in High-speed mode
  - serial UART in different transfer speeds up to 1228.8 kbit/s, framing according to the RS232 interface with voltage levels according pad voltage supply
- Comfortable 64 byte send and receive FIFO-buffer
- Flexible interrupt modes
- Hard reset with low power function
- Power-down mode per software
- Programmable timer
- Internal oscillator to connect 27.12 MHz quartz
- 2.5 - 3.3 V power supply
- CRC Co-processor
- Free programmable I/O pins
- Internal self test

7. Pinning information

7.1 Pinning

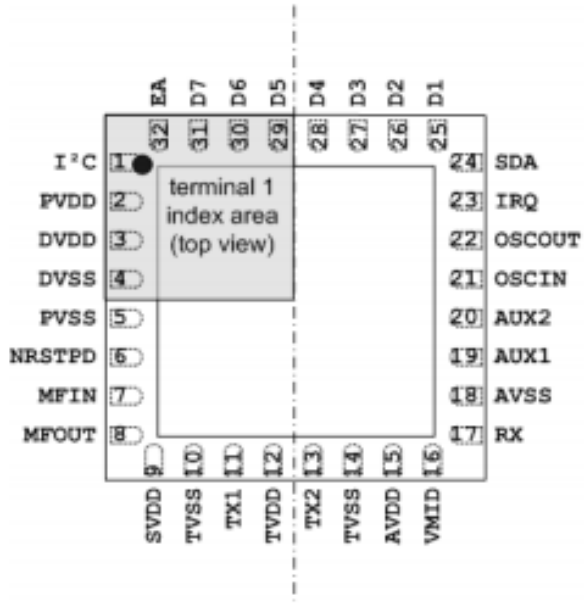


Fig 3. Pinning configuration HVQFN32 (SOT617-1).

7.2 Pin description

Table 3: Pin description

Symbol	Pin	Type	Description
I^2C	1	I	**I2C enable**[2]
PVDD	2	PWR	**Pad power supply**
DVDD	3	PWR	**Digital Power Supply**
DVSS	4	PWR	**Digital Ground**[1]
PVSS	5	PWR	**Pad power supply ground**
NRSTPD	6	I	**Not Reset and Power-down:** When LOW, internal current sinks are switched off, the oscillator is inhibited, and the input pads are disconnected from the outside world. With a positive edge on this pin the internal reset phase starts.
MFIN	7	I	**Mifare Signal Input**
MFOUT	8	O	**Mifare Signal Output**
SVDD	9	PWR	**MFIN / MFOUT Pad Power Supply:** provides power to for the MFIN / MFOUT pads
TVSS	10, 14	PWR	**Transmitter Ground:** supplies the output stage of TX1 and TX2

Table 3: Pin description ...continued

Symbol	Pin	Type	Description
TX1	11	O	**Transmitter 1:** delivers the modulated 13.56 MHz energy carrier
TVDD	12	PWR	**Transmitter Power Supply:** supplies the output stage of TX1 and TX2
TX2	13	O	**Transmitter 2:** delivers the modulated 13.56 MHz energy carrier
TVSS	10, 14	PWR	**Transmitter Ground:** supplies the output stage of TX1 and TX2
AVDD	15	PWR	**Analog Power Supply**
VMID	16	PWR	**Internal Reference Voltage:** This pin delivers the internal reference voltage.
RX	17	I	**Receiver Input.** Pin for the received RF signal.
AVSS	18	PWR	**Analog Ground**
AUX1	19	O	**Auxiliary Outputs:** These pins are used for testing.
AUX2	20	O	
OSCIN	21	I	**Crystal Oscillator Input:** input to the inverting amplifier of the oscillator. This pin is also the input for an externally generated clock (f_{osc} = 27.12 MHz).
OSCOUT	22	O	**Crystal Oscillator Output:** Output of the inverting amplifier of the oscillator.
IRQ	23	O	**Interrupt Request:** output to signal an interrupt event
SDA	24	I	**Serial Data Line**[2]
D1	25	I/O	**Data Pins for different interfaces** (test port, I²IC, SPI, UART)[2]
D2	26	I/O	
D3	27	I/O	
D4	28	I/O	
D5	29	I/O	
D6	30	I/O	
D7	31	I/O	
EA	32	I	**External Address:** This Pin is used for coding I2C Address[2]

- ***ANEXO 2***
  - ***Código Programación Arduino***

```
/*******************ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO*******************/
/********************  FACULTAD DE INFORMATICA Y ELECTRONICA   *******************/
/*Realizado por:                                                                */
/*          DARIO LLANGA                                                        */
/*          DAVID CANO                                                          */
/*                                                                              */
/*Conexión de los pin del ARDUINO UNO con sensor RFID-RC522                     */
/*   MOSI: Pin 11                                                               */
/*   MISO: Pin 12                                                               */
/*   SCK: Pin 13                                                                */
/*   SDA: Pin 10                                                                */
/*   RST: Pin 9                                                                 */
/*                                                                              */
/*Conexión de los pin del ARDUINO UNO con pantalla LCD 12x2                     */
/*   D4: Pin 6                                                                  */
/*   D5: Pin 5                                                                  */
/*   D6: Pin 2                                                                  */
/*   D7: Pin 4                                                                  */
/*                                                                              */
/*******************************************************************************/

#include <SPI.h>       //se carga la librería SPI
#include <RFID.h>      //se carga la librería RFID
#include <LiquidCrystal.h>//se carga la librería LiquidCrystal
#include <EEPROM.h>


#define SS_PIN 10      //selecciona el pin 10 en el que se conectara el sensor RFID
#define RST_PIN 9      //selecciona el pin 10 en el que se conectara el sensor RFID

RFID rfid(SS_PIN, RST_PIN); //ejecuta la función RFID
LiquidCrystal lcd(7, 8, 6, 5, 2 , 4);  //ejecuta la función LiquidCrystal
   String input = "";      //se crea una variable tipo string
   String registro= "";
   int serNum0;          //se crea variables tipo enteros para sacar el código de la tarjeta
int serNum1;
   int serNum2;
   int serNum3;
   int serNum4;

int N;
int x;
String incorrecto="";
int m = 0;
byte valor;
   int t=0;

//configuración inicial
void setup()
{
```

```
  Serial.begin(9600);   //india la comunicación serial
  SPI.begin();
  rfid.init();         //inicia el sensor
  lcd.begin(16, 2);     //declaramos el tamaño del LCD
  lcd.clear();         // limpiamos la pantalla LCD
  input = "";
  //pinMode(16,OUTPUT);
}

void loop()
{
     if (Serial.available()>0)                  //si existe algún dato en el puerto serial
      {
        input= Serial.readStringUntil('\n');   //ese dato se guarda en la variable input
             if (input=="on")
                  {
                       Serial.print(2,HEX);        //se imprime el código en hexadecimal
Serial.print("-");
                       for (int e=0; e<5;e++)
{
                                 valor = EEPROM.read(e);
                                 Serial.print(valor, HEX);
                                 Serial.print(",");
                               }
                       Serial.println("");


                       Serial.print(2,HEX);        //se imprime el código en hexadecimal
Serial.print("-");
                       for (int e=5; e<10;e++)
{
                                 valor = EEPROM.read(e);
                                 Serial.print(valor, HEX);
                                 Serial.print(",");
                               }
                       Serial.println("");


                       Serial.print(2,HEX);        //se imprime el código en hexadecimal
Serial.print("-");
                       for (int e=10; e<15;e++)
{
                                 valor = EEPROM.read(e);
                                 Serial.print(valor, HEX);
                                 Serial.print(",");
                               }
                       Serial.println("");


                       Serial.print(2,HEX);        //se imprime el código en hexadecimal
Serial.print("-");
                       for (int e=15; e<20;e++)
{
                                 valor = EEPROM.read(e);
                                 Serial.print(valor, HEX);
Serial.print(",");
                               }
```

```
                        Serial.println("");


                        for (t = 0; t < 512; t++)
                                {
                                  EEPROM.write(t, 0);
                                }
                         valor=0;
                         t=0;
                    }
                else{




input="";               //borramos el contenido de la variable input
              delay(1000);

             lcd.clear();           //limpiamos la pantalla LCD
                }


      }

   else  {
        if (rfid.isCard())                //si el sensor detecta una señal RFID
          {
           if (rfid.readCardSerial())
              {
                 if (rfid.serNum[0] != serNum0    //si cumple todas estas condiciones
&& rfid.serNum[1] != serNum1
&& rfid.serNum[2] != serNum2
&& rfid.serNum[3] != serNum3
&& rfid.serNum[4] != serNum4)
{
                        serNum0 = rfid.serNum[0];   //el dato se guarda en las variables serNumX
serNum1 = rfid.serNum[1];
                        serNum2 = rfid.serNum[2];
                        serNum3 = rfid.serNum[3];
                        serNum4 = rfid.serNum[4];

Serial.print(2,HEX);        //se imprime el código en hexadecimal
Serial.print("-");
                    Serial.print(rfid.serNum[0],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[1],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[2],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[3],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[4],HEX);
                         Serial.print(1,HEX);
                        Serial.println("");

analogWrite(3,20);          //prende la pantalla LCD
                        lcd.home();          //posición de la pantalla LCD
```

```
                    lcd.print("<<<BIENVENIDO>>>");  //imprime en la pantalla

                    delay (1000);              //dura 5 segundos prendido la pantalla

                     if (Serial.available()>0)       //si existe algún dato en el puerto serial
                        {
                         input= Serial.readStringUntil('\n');
                         registro=input;
                            N=input.length();
                            if (input!="incorrecto")
                                    {

                                if (N>11)
                                    {

                                        x=N-10;
                                        lcd.setCursor(0,1);          //ubicación donde se imprime el
dato en la LCD
                                        lcd.print(" Ing.");
                                        lcd.setCursor(5,1);

                                        lcd.print(input);       //imprime el dato en la pantalla
for (int positionCounter = 0; positionCounter < x; positionCounter++) {

                                            lcd.scrollDisplayLeft();

                                            delay(900);
                                              }

                                            for (int positionCounter = 0; positionCounter < x-1;
positionCounter++) {

lcd.scrollDisplayRight();
                                               delay(900);
                                            }
                                            delay(5000);
                                            analogWrite(3,255);      //apaga la pantalla LCD
                                            input="";              //limpia la variable input
                                            lcd.clear();            //limpia la pantalla LCD
                                    }
                                    else
                                    {
                                     lcd.setCursor(0,1);
                                     lcd.print("Ing.");
                                     lcd.setCursor(5,1);
                                     lcd.print(input);
                                     delay(5000);
                                     analogWrite(3,255);      //apaga la pantalla LCD
                                     input="";              //limpia la variable input
                                     lcd.clear();            //limpia la pantalla LCD
                                    }

                                 }
                                else{

                                 //incorrecto=input;
                                    lcd.clear();
```

```
                                    lcd.setCursor(4,0);           //posición de la pantalla LCD
                                   lcd.print("TARJETA");  //imprime en la pantalla
                                   lcd.setCursor(3,1);          //ubicación donde se imprime el dato en la
LCD
                                   lcd.print("INCORRECTA");       //imprime el dato en la pantalla
                                   delay(3000);
                                   analogWrite(3,255);       //apaga la pantalla LCD
                                    lcd.clear();
                                      }
                                 }
                                else
                                {
                                  lcd.clear();
                                lcd.setCursor(0,0);           //posición de la pantalla LCD
lcd.print("!!FUERA DE!! ");  //imprime en la pantalla
                           lcd.setCursor(0,1);       //ubicación donde se imprime el dato en la LCD
lcd.print("!!SERVICIO!!");       //imprime el dato en la pantalla
                                delay(3000);

                                  for(m=0;m<5;m++){
EEPROM.write(t,rfid.serNum[m]);
                                   rfid.serNum[m]=
                                     t=t+1;
}
                                       analogWrite(3,255);       //apaga la pantalla LCD
                                          lcd.clear();
                                       registro="fuera";
                                }
                             }
                    else {
                     analogWrite(3,20);           //prende la pantalla LCD
                      lcd.setCursor(0,0);           //posición de la pantalla LCD
                            lcd.print("<<<REGISTRADO>>>");  //imprime en la pantalla

                        if (input=="incorrecto")
                            {
                              lcd.clear();
                                       lcd.setCursor(4,0);           //posición de la pantalla LCD
                                      lcd.print("TARJETA");  //imprime en la pantalla
                                      lcd.setCursor(3,1);          //ubicación donde se imprime el dato en la
LCD
                                      lcd.print("INCORRECTA");       //imprime el dato en la pantalla
                                      delay(3000);
                                      analogWrite(3,255);       //apaga la pantalla LCD
                                      //delay (1000);
                                      lcd.clear();
                            }
                           else
                           {
                                analogWrite(3,20);           //prende la pantalla LCD
                               lcd.setCursor(0,0);
                                      if (registro!="fuera")
                                      {

                                          analogWrite(3,20);           //prende la pantalla LCD aquí
Serial.print(2,HEX);
                                Serial.print("-");
```

```
                    Serial.print(rfid.serNum[0],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[1],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[2],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[3],HEX);
                        Serial.print(",");
                    Serial.print(rfid.serNum[4],HEX);
                        Serial.print(2,HEX);
                        Serial.println("");

lcd.setCursor(0,0);          //posición de la pantalla LCD
                         lcd.print("<<<REGISTRADO>>>");  //imprime en la pantalla
                        lcd.setCursor(0,1);       //ubicación donde se imprime el dato en la LCD
                        lcd.print("Ing.");        //imprime el dato en la pantalla
                        lcd.setCursor(5,1);       //ubicación donde se imprime el dato en la LCD
                        lcd.print(registro);  //imprime en la pantalla
                        delay(3000);
                        analogWrite(3,255);       //apaga la pantalla LCD
                        //delay (1000);
                        lcd.clear();
                        if (Serial.available()>0)        //si existe algún dato en el puerto serial
                            {
                             input= Serial.readStringUntil('\n');     //ese dato se guarda en la variable
input
                               if (input=="salida")
                                  {
                                   if (N>11)
                                          {
                                              lcd.clear();
                                              x=N-10;
                                              analogWrite(3,20);          //prende la pantalla LCD
                                               lcd.setCursor(0,0);          //posición de la pantalla LCD
                                              lcd.print("*****GRACIAS*****");  //imprime en la pantalla
                                              lcd.setCursor(0,1);          //ubicación donde se imprime el
dato en la LCD
                                              lcd.print(" Ing.");
                                              lcd.setCursor(5,1);

                                              lcd.print(registro);        //imprime el dato en la pantalla
for (int positionCounter = 0; positionCounter < x; positionCounter++) {

                                                    lcd.scrollDisplayLeft();

                                                    delay(900);
                                                      }

                                                    for (int positionCounter = 0; positionCounter < x-1;
positionCounter++) {

lcd.scrollDisplayRight();


                                                     delay(900);
                                                  }
                                                  delay(10000);
                                                  analogWrite(3,255);       //apaga la pantalla LCD
```

```
                                        input="";            //limpia la variable input
                                        lcd.clear();          //limpia la pantalla LCD
                                    }
                                    else
                                    {

                                     lcd.clear();

                                         analogWrite(3,20);          //prende la pantalla LCD
                                         lcd.setCursor(0,0);         //posición de la pantalla LCD
                                        lcd.print("*****GRACIAS*****");  //imprime en la pantalla
                                     lcd.setCursor(0,1);
                                     lcd.print("Ing.");
                                     lcd.setCursor(5,1);
                                     lcd.print(registro);
                                     delay(10000);
                                     analogWrite(3,255);      //apaga la pantalla LCD
                                     input="";              //limpia la variable input
                                     lcd.clear();           //limpia la pantalla LCD
                                    }
                                                            }
                            }

                               }
                               else{

                                    lcd.clear();
                                lcd.setCursor(0,0);          //posición de la pantalla LCD
lcd.print("!!FUERA DE!! ");  //imprime en la pantalla
                            lcd.setCursor(0,1);      //ubicación donde se imprime el dato en la LCD
lcd.print("!!SERVICIO!!");       //imprime el dato en la pantalla
                                delay(3000);

                                  for(m=0;m<5;m++){
EEPROM.write(t,rfid.serNum[m]);
                                   rfid.serNum[m]=
                                     t=t+1;
}
                                        analogWrite(3,255);      //apaga la pantalla LCD

                                        lcd.clear();
                                       //registro="fuera";
                                                                }
                        }

                       }
                  }
                 }
                rfid.halt();
        }
}
```

- ***ANEXO 3***

○ *Código Programación de la aplicación JAVA*

- **Acceso_Datos**
  - **C_Conexion.java**

```
package Acceso_Datos;
public class c_Conexion {
        public void cConexion(String query) throws Exception {
      try      {
      Class.forName(c_Global.Driver);
      connection=DriverManager.getConnection(c_Global.Url, c_Global.User, c_Global.Clave);
      preparedStatement=connection.prepareCall(query);
      resultSet=preparedStatement.executeQuery();       }       catch (Exception ex)
        {          throw ex;          }       }
   public void cConexionf(String query, List<c_Parametro>parametros) throws Exception {
      try{
      Class.forName(c_Global.Driver);
      connection=DriverManager.getConnection(c_Global.Url, c_Global.User, c_Global.Clave);
      if (connection!=null){
      System.out.println("Exitoso conexion");       }
      preparedStatement=connection.prepareCall(query);
      for (c_Parametro parametro: parametros){
      preparedStatement.setObject(parametro.getPosicion(),
parametro.getValor(),parametro.getTipoDato());       }
      resultSet=preparedStatement.executeQuery();
      } catch (Exception ex){
      System.out.println(ex);
      throw ex;       }   }
   public void cerrarConexion() throws Exception{
   this.resultSet.close();
   this.preparedStatement.close();
   this.connection.close();   }
   public boolean siguiente()throws Exception{
return this.resultSet.next();   }
      public String getString(String nombreColumna) throws Exception{
   return this.resultSet.getString(nombreColumna);   }
   public ResultSet Listar(String Cad)       {
      try          {
         Class.forName(c_Global.Driver).newInstance();
         connection=DriverManager.getConnection(c_Global.Url, c_Global.User, c_Global.Clave);
                preparedStatement =connection.prepareStatement(Cad);
         ResultSet tabla = preparedStatement.executeQuery();
         return tabla;          }
      catch(Exception e)             {
               javax.swing.JOptionPane.showMessageDialog(null, e.getMessage());
return null;             }   }   }
```

  - **C_Global.java**

```
package Acceso_Datos;
public class c_Global {
    public static final String Url="jdbc:postgresql;
      public static final String User="postgres";
public static final String Clave="12345678";
```

```
    public static final String Driver="org.postgresql.Driver";    }
```

- **C_Parametro.java**

```
package Acceso_Datos;
public class c_Parametro {
   public c_Parametro(int posicion, Object valor, int tipoDato) {
   this.posicion = posicion;
   this.valor = valor;
   this.tipoDato = tipoDato;    }
   public int getPosicion() {
   return posicion;    }
   public void setPosicion(int posicion) {
   this.posicion = posicion;    }
   public Object getValor() {
   return valor;    }
   public void setValor(Object valor) {
   this.valor = valor;    }
   public int getTipoDato() {
   return tipoDato;    }
   public void setTipoDato(int tipoDato) {
this.tipoDato = tipoDato;    }}
```

- **Logica_Negocio_Clases**
  - **C_Docente.java**

```
package Logica_Negocio_Clases;
public class c_Docente {
   public void setCodigo_docente(String codigo_docente) {
this.codigo_docente = codigo_docente;    }
public String getCi_docente() {
      return ci_docente;    }
   public void setCi_docente(String ci_docente) {
      this.ci_docente = ci_docente;    }    public String getNombre_docente() {
return nombre_docente;    }
   public void setNombre_docente(String nombre_docente) {
      this.nombre_docente = nombre_docente;    }
public String getCelular_docente() {
      return celular_docente;    }
   public void setCelular_docente(String celular_docente) {
this.celular_docente = celular_docente;    }}
```

  - **C_Horario.java**

```
package Logica_Negocio_Clases;
public class c_Horario {
   private int codigo_hora;
      return codigo_hora;    }
   public void setCodigo_hora(int codigo_hora) {
      this.codigo_hora = codigo_hora;    }
public String getHora_inicio() {
      return hora_inicio;    }
public void setHora_inicio(String hora_inicio) {
      this.hora_inicio = hora_inicio;    }
   public String getHora_salida() {
      return hora_salida;    }
   public void setHora_salida(String hora_salida) {
      this.hora_salida = hora_salida;    }
public String getHora_dia() {
return hora_dia;    }
```

```
   public void setHora_dia(String hora_dia) {
this.hora_dia = hora_dia;   }
 }
```

- **C_Laboratorio.java**

```
package Logica_Negocio_Clases;
public class c_Laboratorio {
   public int getCodigo_lab() {
      return codigo_lab;   }
   public void setCodigo_lab(int codigo_lab) {
      this.codigo_lab = codigo_lab;   }   public String getNombre_lab() {
      return nombre_lab;   }
   public void setNombre_lab(String nombre_lab) {
      this.nombre_lab = nombre_lab;   }
}
```

- **C_ListarRegistro.java**

```
package Logica_Negocio_Clases;
   public String getCi_docente() {
      return ci_docente;   }
   public void setCi_docente(String ci_docente) {
      this.ci_docente = ci_docente;   }   public String getNombre_docente() {
return nombre_docente;   }   public void setNombre_docente(String nombre_docente) {
      this.nombre_docente = nombre_docente;   }
public String getNombre_materia() {       return nombre_materia;   }
public void setNombre_materia(String nombre_materia) {       this.nombre_materia = nombre_materia;
}
   public String getNombre_laboratorio() {
      return nombre_laboratorio;   }   public void setNombre_laboratorio(String nombre_laboratorio) {
      this.nombre_laboratorio = nombre_laboratorio;   }
public String getHorario() {       return horario;   }
   public void setHorario(String horario) {       this.horario = horario;
   }   public String getFecha_registro() {       return fecha_registro;
   }   public void setFecha_registro(String fecha_registro) {       this.fecha_registro = fecha_registro;
   }   public String getHora_registro() {       return hora_registro;
}   public void setHora_registro(String hora_registro) {       this.hora_registro = hora_registro;
}   public String getEstado() {       return estado;   }public void setEstado(String estado) {       this.estado
= estado;   }
  }
```

- **C_Materia.java**

```
package Logica_Negocio_Clases;public class c_Materia {
      public int getCodigo_materia() {       return codigo_materia;   }
   public void setCodigo_materia(int codigo_materia) {
      this.codigo_materia = codigo_materia;   }
   public String getCodigo_docente() {       return codigo_docente;   }
   public void setCodigo_docente(String codigo_docente) {
      this.codigo_docente = codigo_docente;   }
   public int getCodigo_lab() {       return codigo_lab;   }
public void setCodigo_lab(int codigo_lab) {
      this.codigo_lab = codigo_lab;   }   public int getCodigo_hora() {
      return codigo_hora;   }   public void setCodigo_hora(int codigo_hora) {
      this.codigo_hora = codigo_hora;   }   public String getNombre_materia() {
      return nombre_materia;   }   public void setNombre_materia(String nombre_materia) {
      this.nombre_materia = nombre_materia;   }
   }
```

- **C_Registro.java**

```
package Logica_Negocio_Clases;
public class c_Registro {
      public int getCodigo_registro() {       return codigo_registro;   }
   public void setCodigo_registro(int codigo_registro) {         this.codigo_registro = codigo_registro;    }
public int getCodigo_materia() {       return codigo_materia;   }
public void setCodigo_materia(int codigo_materia) {       this.codigo_materia = codigo_materia;   }
   public String getCodigo_docente() {       return codigo_docente;   }
   public void setCodigo_docente(String codigo_docente) {   this.codigo_docente = codigo_docente;   }
   public int getCodigo_lab() {       return codigo_lab;   }
public void setCodigo_lab(int codigo_lab) {
      this.codigo_lab = codigo_lab;   }   public int getCodigo_hora() {       return codigo_hora;   }
   public void setCodigo_hora(int codigo_hora) {
      this.codigo_hora = codigo_hora;   }   public String getHora_regitro() { return hora_regitro;   }
   public void setHora_regitro(String hora_regitro) {
      this.hora_regitro = hora_regitro;   }
public String getFecha_registro() {       return fecha_registro;   }
public void setFecha_registro(String fecha_registro) {       this.fecha_registro = fecha_registro;   }
public String getEstado() {       return estado;   }
public void setEstado(String estado) {       this.estado = estado;   }
   }
```

- **Logica_Negocio_Metodos**
  - **C_Docente_Metodo.java**

```
package Logica_Negocio_Metodos;
public class c_Docente_Metodo {
   public static boolean Ingresar_Docente(c_Docente docente) throws Exception   {
   boolean result=false;
   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
   parametros.add(new c_Parametro(1, docente.getCodigo_docente(),Types.VARCHAR));
   parametros.add(new c_Parametro(2, docente.getCi_docente(),Types.VARCHAR));
   parametros.add(new c_Parametro(3, docente.getNombre_docente(),Types.VARCHAR));
   parametros.add(new c_Parametro(4, docente.getCelular_docente(),Types.VARCHAR));
   String procedimiento="select * from public.\"insertar_docentes\"(?,?,?,?)";
   c_Conexion con=new c_Conexion();
con.cConexionf(procedimiento, parametros);
   if (con.siguiente()){     String valor=con.getString("insertar_docentes");
if (valor.equals("t")){        result=true;     } }   con.cerrarConexion();
   return result;   }   public ArrayList<c_Docente> ListarDocente()   {
      ArrayList lista=new ArrayList();
      try      {          c_Conexion objcn=new c_Conexion();
        ResultSet   tabla=objcn.Listar("select   id_docente,nombre_docente   from   docente   order   by
nombre_docente");          c_Docente objprov;          while(tabla.next())          {
            objprov=new c_Docente();
            objprov.setCodigo_docente(tabla.getString("id_docente"));
            objprov.setNombre_docente(tabla.getString("nombre_docente"));   lista.add(objprov);   }     }
catch(Exception e)      {          JOptionPane.showMessageDialog(null, e.getMessage());       }
return lista;   }
   }
```

  - **C_Horario_Metodo.java**

```
package Logica_Negocio_Metodos;
public class c_Horario_Metodo {
   public static boolean Ingresar_Horario(c_Horario horario) throws Exception   {
   boolean result=false;   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
   parametros.add(new c_Parametro(1, horario.getCodigo_hora(),Types.INTEGER));
   parametros.add(new c_Parametro(2, horario.getHora_inicio(),Types.TIME));
```

```
   parametros.add(new c_Parametro(3, horario.getHora_salida(),Types.TIME));
   parametros.add(new c_Parametro(4, horario.getHora_dia(),Types.VARCHAR));
String procedimiento="select * from public.\"insertar_ho\"(?,?,?,?)";
c_Conexion con=new c_Conexion();    con.cConexionf(procedimiento, parametros);
if (con.siguiente()){     String valor=con.getString("insertar_ho");     if (valor.equals("t")){
       result=true;     } }    con.cerrarConexion();    return result;    }
   public static c_Conexion Listar_HoraInicioCbx() throws Exception    {
c_Conexion con = new c_Conexion();       con.cConexion("select hora_inicio,hora_salida from horario");
return con;      }
   public ArrayList<c_Horario> ListarHorarios()    {
ArrayList lista=new ArrayList();
      try       {           c_Conexion objcn=new c_Conexion();           ResultSet tabla=objcn.Listar("select *
from horario");          c_Horario objprov;          while(tabla.next())          {
objprov=new c_Horario();             objprov.setCodigo_hora(tabla.getInt("id_horario"));
           objprov.setHora_dia(tabla.getString("dia"));
objprov.setHora_inicio(tabla.getString("hora_inicio"));
objprov.setHora_salida(tabla.getString("hora_salida"));
lista.add(objprov);           }      }      catch(Exception e)      {
         JOptionPane.showMessageDialog(null, e.getMessage());      }
return lista;    }
   }
```

- **C_Laboratorio_Metodo.java**

```
package Logica_Negocio_Metodos;
public class c_Laboratorio_Metodo {
   public static boolean Ingresar_Laboratorio(c_Laboratorio laboratorio) throws Exception    {
   boolean result=false;
   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
   parametros.add(new c_Parametro(1, laboratorio.getCodigo_lab(),Types.INTEGER));
   parametros.add(new c_Parametro(2, laboratorio.getNombre_lab(),Types.VARCHAR));
String procedimiento="select * from public.\"insertar_labs\"(?,?)";
c_Conexion con=new c_Conexion();    con.cConexionf(procedimiento, parametros);
if (con.siguiente()){     String valor=con.getString("insertar_labs");
    if (valor.equals("t")){        result=true;     } }
   con.cerrarConexion();    return result;    }
   public static c_Conexion Listar_LaboratorioCbx() throws Exception    {
      c_Conexion con = new c_Conexion();       con.cConexion("select nombre_lab from laboratorio");
      return con;    }     public ArrayList<c_Laboratorio> ListarLaboratorios()    {
      ArrayList lista=new ArrayList();
      try       {           c_Conexion objcn=new c_Conexion();           ResultSet tabla=objcn.Listar("select *
from laboratorio");          c_Laboratorio objprov;          while(tabla.next())          {
           objprov=new c_Laboratorio();             objprov.setCodigo_lab(tabla.getInt("id_lab"));
objprov.setNombre_lab(tabla.getString("nombre_lab"));             lista.add(objprov);     }    }
      catch(Exception e)     {     JOptionPane.showMessageDialog(null, e.getMessage());    }   return lista;
   }}
```

- **C_Materia_Metodo.java**

```
package Logica_Negocio_Metodos;
public class c_Materia_Metodo {
   public static boolean Ingresar_Materia(c_Materia materia) throws Exception       {       boolean
result=false;    List<c_Parametro> parametros =new ArrayList<c_Parametro>();
parametros.add(new c_Parametro(1, materia.getCodigo_materia(),Types.INTEGER));
   parametros.add(new c_Parametro(2, materia.getCodigo_docente(),Types.VARCHAR));
   parametros.add(new c_Parametro(3, materia.getCodigo_lab(),Types.INTEGER));
   parametros.add(new c_Parametro(4, materia.getCodigo_hora(),Types.INTEGER));
   parametros.add(new c_Parametro(5, materia.getNombre_materia(),Types.VARCHAR));
```

```
String procedimiento="select * from public.\"insertar_materia\"(?,?,?,?,?)";
c_Conexion con=new c_Conexion();    con.cConexionf(procedimiento, parametros);
if (con.siguiente()){     String valor=con.getString("insertar_materia");
     if (valor.equals("t")){         result=true;     }          }    con.cerrarConexion();    return result;    }
   public static int COntarServicios(String id_doc, int id_lab)    {       int idb=1;
      try {                c_Conexion obj= new c_Conexion();                ResultSet tabla=obj.Listar("select
count(nombre_materia) as total from materia where id_docente='"+id_doc+"' and id_lab="+id_lab+"
and id_horario=0");          while (tabla.next())          {              idb=(tabla.getInt("total"));
           }    } catch (Exception ex) {          System.out.println(ex);       }       return idb;   }
}
```

- **C_Registro_Metodo.java**

```
package Logica_Negocio_Metodos;
public class c_Registro_Metodo {
   public static boolean Ingresar_Registro(c_Registro registr) throws Exception    {
   boolean result=false;    List<c_Parametro> parametros =new ArrayList<c_Parametro>();
  parametros.add(new c_Parametro(1, registr.getCodigo_materia(),Types.INTEGER));
parametros.add(new c_Parametro(2, registr.getCodigo_docente(),Types.VARCHAR));
   parametros.add(new c_Parametro(3, registr.getCodigo_lab(),Types.INTEGER));
   parametros.add(new c_Parametro(4, registr.getCodigo_hora(),Types.INTEGER));
   parametros.add(new c_Parametro(5, registr.getHora_regitro(),Types.TIME));
   parametros.add(new c_Parametro(6, registr.getFecha_registro(),Types.DATE));
   parametros.add(new c_Parametro(7, registr.getEstado(),Types.VARCHAR));
String procedimiento="select * from public.\"insertar_registrod\"(?,?,?,?,?,?,?)";
c_Conexion con=new c_Conexion();    con.cConexionf(procedimiento, parametros);
if (con.siguiente()){     String valor=con.getString("insertar_registrod");
     if (valor.equals("t")){         result=true;     }     }    con.cerrarConexion();    return result;    }
public static int VerIDRegistro()    {       int idb=0;       try {          c_Conexion obj= new c_Conexion();
         ResultSet tabla=obj.Listar("select COUNT(id_registro) as id_registro from registro");
         while (tabla.next())                {           idb=(tabla.getInt("id_registro")+1);                }       } catch
(Exception e) {          JOptionPane.showMessageDialog(null, e.getMessage());       }
     return idb;     }    public static String VerNombre(String iddoc) throws SQLException{
String idb="Horario no registrado";      c_Conexion obj= new c_Conexion();       try {
         obj.cConexion("select nombre_docente from docente where id_docente='"+iddoc+"'");
       } catch (Exception ex) {          System.out.println(ex);       }
           try {          idb=obj.getString("nombre_docente");       } catch (Exception ex) {
          System.out.println(ex);       }     return idb;       }    public static int VerIDHorario()    {
       Date date = new Date();       DateFormat hourFormat = new SimpleDateFormat("HH:MM:ss");
       DateFormat dateFormat = new SimpleDateFormat("EEEEE");
       int idb=0;       try {          c_Conexion obj= new c_Conexion();
          ResultSet tabla=obj.Listar("select id_horario from horario where hora_inicio <
'"+hourFormat.format(date)+"' and hora_salida > '"+hourFormat.format(date)+"' and
dia='"+dateFormat.format(date)+"'");          while (tabla.next())          {
              idb=(tabla.getInt("id_horario"));              }          } catch (Exception e) {
          JOptionPane.showMessageDialog(null, e.getMessage());       }
         return idb;       }    public static String VerNombreDocente(String iddoc)    {
         System.out.println("este entra ");       String idb="";       try {
          c_Conexion obj= new c_Conexion();              ResultSet tabla=obj.Listar("select nombre_docente
from docente where id_docente='"+iddoc+"'");          while (tabla.next())          {
            idb=(tabla.getString("nombre_docente"));     System.out.println("este entra "+idb);          }
          } catch (Exception e) {          JOptionPane.showMessageDialog(null, e.getMessage());       }
        return idb;       }    public static int VerIDMateria(String iddoc,int idlab,int idhora)    {
    int idb=0;       try {          c_Conexion obj= new c_Conexion();          ResultSet tabla=obj.Listar("select
id_materia from materia where id_docente='"+iddoc+"' and id_lab="+idlab+" and
id_horario="+idhora);
```

```
   while (tabla.next())            {      idb=(tabla.getInt("id_materia"));}
  } catch (Exception e) {          JOptionPane.showMessageDialog(null, e.getMessage());      }
   return idb;            }     public static int Obtener_ID_Materia(String iddoc,int idlab,int idhor) throws
Exception   {   int result=0;   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
   parametros.add(new c_Parametro(1, iddoc,Types.VARCHAR));
   parametros.add(new c_Parametro(2, idlab,Types.INTEGER));
   parametros.add(new c_Parametro(3, idhor,Types.INTEGER));
 String procedimiento="select * from public.\"devolver_id_materia\"(?,?,?)";
c_Conexion con=new c_Conexion();   con.cConexionf(procedimiento, parametros);
if (con.siguiente()){      if (con.getString("devolver_id_materia")!=null)      {
        result=Integer.parseInt(con.getString("devolver_id_materia"));      }
   if (con.siguiente()){       if (con.getString("devolver_nombre")==null)      {
        System.out.println("este entra "+con.getString("devolver_nombre"));
        result="no";      }     else        result=(con.getString("devolver_nombre"));        }
   con.cerrarConexion();   return result;   }
    public ArrayList<c_ListarRegistro> ListarRegistros()   {      ArrayList lista=new ArrayList();
      try      {         c_Conexion objcn=new c_Conexion();
        ResultSet      tabla=objcn.Listar("select      D.ci_docente      as      CI,D.nombre_docente      as
Nombre,M.nombre_materia as Materia,L.nombre_lab as Laboratorio,H.dia || ' ' ||H.hora_inicio || ' a '
|| H.hora_salida as Horario,R.fecha_registr as Fecha_Registro,R.hora_registr as Hora_Registro,R.estado
as Estado from materia as M inner join registrodoc as R on M.id_materia=R.id_mteriad inner join
laboratorio as L on L.id_lab=id_labd inner join horario as H on H.id_horario=id_horariod inner join
docente as D on D.id_docente=id_docented order by R.id_registrod desc");
        c_ListarRegistro objprov;
        while(tabla.next())        {
          objprov=new c_ListarRegistro();
          objprov.setCi_docente(tabla.getString("CI"));
objprov.setNombre_docente(tabla.getString("Nombre"));
          objprov.setNombre_materia(tabla.getString("Materia"));
          objprov.setNombre_laboratorio(tabla.getString("Laboratorio"));
          objprov.setHorario(tabla.getString("Horario"));
          objprov.setFecha_registro(tabla.getString("Fecha_Registro"));
          objprov.setHora_registro(tabla.getString("Hora_Registro"));
          objprov.setEstado(tabla.getString("Estado"));
lista.add(objprov);                  }      }
      catch(Exception e)      {
        JOptionPane.showMessageDialog(null, e.getMessage());      }
      return lista;   }
      public static int Verificar_RegistroSalida() throws Exception   {   int result=-1;
   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
  Date date = new Date();   DateFormat dateFormat = new SimpleDateFormat("EEEEE");
   DateFormat hourFormat = new SimpleDateFormat("H:mm:ss");
    parametros.add(new c_Parametro(1, dateFormat.format(date),Types.VARCHAR));
   parametros.add(new c_Parametro(2, hourFormat.format(date),Types.TIME));
String procedimiento="select * from public.\"verificar_horario\"(?,?)";
   c_Conexion con=new c_Conexion();   con.cConexionf(procedimiento, parametros);
   if (con.siguiente()){
    String valor=con.getString("verificar_horario");      if (valor.equals("t")){         c_Conexion obj= new
c_Conexion(); obj.cConexion("select id_horario from horario where  dia='"+dateFormat.format(date)+"'
and (hora_inicio <'"+hourFormat.format(date)+"' and hora_salida>'"+hourFormat.format(date)+"')");
       if (obj.siguiente()){          result=Integer.parseInt(obj.getString("id_horario")); } }   }
con.cerrarConexion();      return result;      }     ublic static String Obtener_HoraSalida(int id) throws
Exception   {   String result="";   List<c_Parametro> parametros =new ArrayList<c_Parametro>();
parametros.add(new c_Parametro(1, id,Types.INTEGER));
```

```
   String procedimiento="select * from public.\"devolver_horariosalida\"(?)";
   c_Conexion con=new c_Conexion();
con.cConexionf(procedimiento, parametros);    if (con.siguiente()){
String valor=con.getString("devolver_horariosalida");
     if (valor.equals("null")){     } else {      result=(con.getString("devolver_horariosalida"));       }
con.cerrarConexion();      return result;    }
}
```

- **Interfaz_Usuario**
  - **Registro_Asistencia.java**

```
package Interfaz_Usuario;
public class Registro_Asistencia extends javax.swing.JInternalFrame {
     Arduino Arduino = new Arduino();
   String valor;
  SerialPortEventListener evento= new SerialPortEventListener()         {
          @Override
          public void serialEvent(SerialPortEvent spe)                {
                if (Arduino.MessageAvailable()==true)                   {
               valor= Arduino.PrintMessage();
                  System.out.println("codigo docente "+valor.substring(2,16));
                  System.out.println("codigo laboratorio "+valor.substring(0,1));                    try {
                    System.out.println("Id                                                                  materia
"+c_Registro_Metodo.Obtener_ID_Materia(valor.substring(2,16),
Integer.parseInt(valor.substring(0,1)),c_Registro_Metodo.VerIDHorario()));
                  } catch (Exception ex) {
                    System.out.println(ex);                       }
                  Calendar c1 = GregorianCalendar.getInstance();
                  Calendar c2 = GregorianCalendar.getInstance();
                  SimpleDateFormat fecha = new SimpleDateFormat("Y-M-d");
                    SimpleDateFormat hora = new SimpleDateFormat("H:mm:ss");
                      c_Registro regi=new c_Registro();
regi.setCodigo_docente(valor.substring(2,16));
                      regi.setCodigo_hora(c_Registro_Metodo.VerIDHorario());                              try  {
regi.setCodigo_materia(c_Registro_Metodo.Obtener_ID_Materia(valor.substring(2,16),
Integer.parseInt(valor.substring(0,1)),c_Registro_Metodo.VerIDHorario()));
} catch (Exception ex) {
                    System.out.println(ex);                       }
regi.setCodigo_lab(Integer.parseInt(valor.substring(0,1)));
System.out.println("Fecha: "+fecha.format(c1.getTime()));
                  regi.setFecha_registro(fecha.format(c1.getTime()));
                  regi.setHora_regitro(hora.format(c1.getTime()).substring(0,7));
                  System.out.println("num registro "+valor.substring(17,18));
                  try {
                    if (Integer.parseInt(valor.substring(17,18))==1)                    {
regi.setEstado("Entrada");
                      c_Registro_Metodo.Ingresar_Registro(regi);
Listar_Asistencia();
                  String rec=tbasistencia.getValueAt(0,1).toString();try {
                       Arduino.SendData(rec);                              }
     catch (Exception ex) {              System.out.println(ex);                    }                    }
                  if (Integer.parseInt(valor.substring(17,18))==2)                    {
                          int idho;                    String horasal;                    try {
idho=c_Registro_Metodo.Verificar_RegistroSalida();
                  horasal=c_Registro_Metodo.Obtener_HoraSalida(idho);
                  Calendar c11 = GregorianCalendar.getInstance();
```

```
                    Calendar c21 = GregorianCalendar.getInstance();
                    Calendar c31 = GregorianCalendar.getInstance();
                    int hor=Integer.parseInt(horasal.substring(0,2));
int min=Integer.parseInt(horasal.substring(3,5));
int seg=Integer.parseInt(horasal.substring(6,8));
                     c31.set(Calendar.HOUR_OF_DAY,hor);
                     c31.set(Calendar.MINUTE,min);
                     c31.set(Calendar.SECOND,seg);
                     c31.add(Calendar.MINUTE,-15);
                     c21.set(Calendar.HOUR_OF_DAY,hor);
                     c21.set(Calendar.MINUTE,min);
                     c21.set(Calendar.SECOND,seg);
                    int hora1=c31.get(Calendar.HOUR_OF_DAY);
                     int hora2=c11.get(Calendar.HOUR_OF_DAY);
                     int hora3=c21.get(Calendar.HOUR_OF_DAY);
System.out.println("hora 1 " +hora1+" hora2 "+hora2 +" hora3 "+hora3);
int min1=c31.get(Calendar.MINUTE);
                     int min2=c11.get(Calendar.MINUTE);
                     int min3=c21.get(Calendar.MINUTE);
                     System.out.println("min 1 " +min1+" min2 "+min2 +" min3 "+min3);
String ver1;        String ver2;                String ver3;      if (min3<10)                        {
ver3=Integer.toString(hora3)+"0"+Integer.toString(min3);                        }
else                        {
                       ver3=Integer.toString(hora3)+Integer.toString(min3);                        }
                     if (min2<10)                        {
ver2=Integer.toString(hora2)+"0"+Integer.toString(min2);                        }
else                        {
                       ver2=Integer.toString(hora2)+Integer.toString(min2);                        }
                     if (min1<10)                        {
ver1=Integer.toString(hora1)+"0"+Integer.toString(min1);                        }
else                        {
                       ver1=Integer.toString(hora1)+Integer.toString(min1);                        }
int compr1=Integer.parseInt(ver1);
                     int compr2=Integer.parseInt(ver2);
                     int compr3=Integer.parseInt(ver3);
System.out.println("medio "+compr2+" mayor"+compr3+" memnor "+compr1);
                    if((compr2<compr3) && (compr2>compr1))                              {
                          System.out.println("pasa 2 ");
                          regi.setEstado("Salida");
                             c_Registro_Metodo.Ingresar_Registro(regi);
                            Listar_Asistencia();                        try                             {
                           Arduino.SendData("salida");                              }
catch (Exception ex) {
                              System.out.println(ex);                           }                        }
                 } catch (Exception ex) {
                    System.out.println(ex);}       }
                 if (Integer.parseInt(valor.substring(17,18))==3)                    {
regi.setEstado("Servicio");
                   regi.setCodigo_docente(valor.substring(2,16));
                   regi.setCodigo_hora(0);
regi.setCodigo_materia(c_Registro_Metodo.Obtener_ID_Materia(valor.substring(2,16),
Integer.parseInt(valor.substring(0,1)),0));
                   regi.setCodigo_lab(Integer.parseInt(valor.substring(0,1)));
                   c_Registro_Metodo.Ingresar_Registro(regi);
```

```
Listar_Asistencia();                    }
                    System.out.println(ex);
try {
                        Arduino.SendData("incorrecto");
                        }
                        catch (Exception ex1) {
                          System.out.println(ex1);          } }  }  };
    public Registro_Asistencia() {      initComponents();      Listar_Asistencia();
try{      Arduino.ArduinoRXTX("COM3", 2000, 9600, evento);      }
catch(Exception ex){
     Logger.getLogger(Registro_Docentes.class.getName()).log(Level.SEVERE, null, ex);      }
          try {          Arduino.SendData("on");          }          catch (Exception ex1) {
            System.out.println(ex1);          }     }
     public void Listar_Asistencia()   {
     DefaultTableModel tabla=new DefaultTableModel();
     c_Registro_Metodo prov=new c_Registro_Metodo();
ArrayList<c_ListarRegistro> lista=new ArrayList();
lista=prov.ListarRegistros();      tabla.addColumn("Id");      tabla.addColumn("Nombre");
     tabla.addColumn("Materia");      tabla.addColumn("Laboratorio");
     tabla.addColumn("Horario");      tabla.addColumn("Fecha Registro");
     tabla.addColumn("Hora Registro");      tabla.addColumn("Estado");
tabla.setRowCount(lista.size());      int i=0;      for(c_ListarRegistro x:lista)
{
        tabla.setValueAt(x.getCi_docente(), i, 0);        tabla.setValueAt(x.getNombre_docente(), i, 1);
        tabla.setValueAt(x.getNombre_materia(),                    i,                    2);
tabla.setValueAt(x.getNombre_laboratorio(), i, 3);        tabla.setValueAt(x.getHorario(), i, 4);
        tabla.setValueAt(x.getFecha_registro(), i, 5);        tabla.setValueAt(x.getHora_registro(), i, 6);
        tabla.setValueAt(x.getEstado(), i, 7);        i++;      }
     this.tbasistencia.setModel(tabla);   }
  private javax.swing.JLabel jLabel1;
   private javax.swing.JScrollPane jScrollPane1;
   private javax.swing.JTable tbasistencia;}
```

- **Registro_Docentes.java**

```
public class Registro_Docentes extends javax.swing.JInternalFrame {
     Arduino Arduino = new Arduino();
  SerialPortEventListener evento= new SerialPortEventListener()       {
          @Override
          public void serialEvent(SerialPortEvent spe)              {
               if (Arduino.MessageAvailable()==true)                {
                  String valor= Arduino.PrintMessage();
                  txtid.setText(valor.substring(2,16));              }          }          };
     public Registro_Docentes() {      initComponents();    try{
     Arduino.ArduinoRXTX("COM3", 2000, 9600, evento);      }
   catch(Exception ex){          Logger.getLogger(Registro_Docentes.class.getName()).log(Level.SEVERE,
null,      ex);                    }                    }private      void      initComponents()private      void
btnguardarActionPerformed(java.awt.event.ActionEvent evt) {
if ("".equals(this.txtid.getText()) || "".equals(this.txtci.getText()) || "".equals(this.txtnombre.getText()))
     {
       JOptionPane.showMessageDialog(this, " Ingrese todos los campos marcados con * ",
"Informacion", JOptionPane.INFORMATION_MESSAGE);      }      else      {
        c_Docente user=new c_Docente();
        user.setNombre_docente(this.txtnombre.getText());
        user.setCodigo_docente(this.txtid.getText());
        user.setCi_docente(this.txtci.getText());
```

```
        user.setCelular_docente(this.txtcelular.getText());
try{        c_Docente_Metodo.Ingresar_Docente(user);
        try {            Arduino.SendData(this.txtnombre.getText());            }
catch (Exception ex) {            Logger.getLogger(Registro_Docentes.class.getName()).log(Level.SEVERE,
null, ex);        }
        JOptionPane.showMessageDialog(this, "Ingreso correcto");
                this.txtci.setText("");        this.txtnombre.setText("");
        this.txtid.setText("");        this.txtcelular.setText("");        }
        catch(Exception ex){
        JOptionPane.showMessageDialog(this, ex);        }        }  }
private void btnlimpiarActionPerformed(java.awt.event.ActionEvent evt) {
            this.txtci.setText("");      this.txtnombre.setText("");
     this.txtid.setText("");      this.txtcelular.setText("");   }
private void btncancelarActionPerformed(java.awt.event.ActionEvent evt) {
     try {            Arduino.SendData("Regis");            }
       catch                              (Exception                              ex)                              {
Logger.getLogger(Registro_Docentes.class.getName()).log(Level.SEVERE, null, ex);
        }      this.dispose();
}
```

- **Registro_Horario.java**

```
package Interfaz_Usuario;
public class Registro_Horarios extends javax.swing.JInternalFrame {
   public Registro_Horarios() {      initComponents();
   }   @SuppressWarnings("unchecked")
private void initComponents()
private void btnlimpiarActionPerformed(java.awt.event.ActionEvent evt) {
     this.txtid.setText("");        this.txtinicio.setText("");        this.txtsalida.setText("");   }
   private void btncancelarActionPerformed(java.awt.event.ActionEvent evt) {
     this.dispose();   }
   private void btnguardarActionPerformed(java.awt.event.ActionEvent evt) {
     else
     {
       c_Horario hor=new c_Horario();
hor.setCodigo_hora(Integer.parseInt(this.txtid.getText()));
        hor.setHora_inicio(this.txtinicio.getText());
hor.setHora_salida(this.txtsalida.getText());
hor.setHora_dia(cbxdia.getSelectedItem().toString());
try{        c_Horario_Metodo.Ingresar_Horario(hor);
JOptionPane.showMessageDialog(this, "Ingreso correcto");
        this.txtid.setText("");        this.txtinicio.setText("");
        this.txtsalida.setText("");        }
        catch(Exception ex){
        JOptionPane.showMessageDialog(this, ex);        }     }  }
   }
```

- **Rgistro_Laboratorio.java**

```
package Interfaz_Usuario;
public class Registro_Laboratorios extends javax.swing.JInternalFrame {
   public Registro_Laboratorios() {      initComponents();   }
private void btnlimpiarActionPerformed(java.awt.event.ActionEvent evt) {
     txtidlab.setText("");      txtnombrelab.setText("");   }
   private void btncancelarActionPerformed(java.awt.event.ActionEvent evt) {
     this.dispose();   }
   private void btnguardarActionPerformed(java.awt.event.ActionEvent evt) {
     else      {
```

```
        c_Laboratorio lab=new c_Laboratorio();
lab.setNombre_lab(this.txtnombrelab.getText());
        lab.setCodigo_lab(Integer.parseInt(this.txtidlab.getText()));
try{        c_Laboratorio_Metodo.Ingresar_Laboratorio(lab);
JOptionPane.showMessageDialog(this, "Ingreso correcto");
        txtidlab.setText("");        txtnombrelab.setText("");        }
        catch(Exception ex){        JOptionPane.showMessageDialog(this, ex); } } }
   }
```

- **Registro_Materias.java**

```
package Interfaz_Usuario;
public class Registro_Materias extends javax.swing.JInternalFrame {
public Registro_Materias() {
      initComponents();      Listar_Labs();
      Listar_Horarios();      Listar_Docentes();    }    public void Listar_Docentes()   {
      DefaultTableModel tabla=new DefaultTableModel();
      c_Docente_Metodo prov=new c_Docente_Metodo();
ArrayList<c_Docente> lista=new ArrayList();
lista=prov.ListarDocente();
      tabla.addColumn("Id");      tabla.addColumn("Nombre");
tabla.setRowCount(lista.size());      int i=0;
for(c_Docente x:lista)      {
        tabla.setValueAt(x.getCodigo_docente(), i, 0);
        tabla.setValueAt(x.getNombre_docente(), i, 1);        i++;      }
this.tbdocente.setModel(tabla);        }      public void Listar_Labs()   {
      DefaultTableModel tabla=new DefaultTableModel();
      c_Laboratorio_Metodo prov=new c_Laboratorio_Metodo();
      ArrayList<c_Laboratorio> lista=new ArrayList();
      lista=prov.ListarLaboratorios();
      tabla.addColumn("Id");      tabla.addColumn("Nombre");
tabla.setRowCount(lista.size());      int i=0;
for(c_Laboratorio x:lista)      {
        tabla.setValueAt(x.getCodigo_lab(), i, 0);
        tabla.setValueAt(x.getNombre_lab(), i, 1);        i++;      }
this.tblaboratorios.setModel(tabla);        }
     public void Listar_Horarios()   {
      DefaultTableModel tabla=new DefaultTableModel();
      c_Horario_Metodo prov=new c_Horario_Metodo();
      ArrayList<c_Horario> lista=new ArrayList();
      lista=prov.ListarHorarios();
      tabla.addColumn("Id");      tabla.addColumn("Dia");
      tabla.addColumn("Hora Inicio");      tabla.addColumn("Hora Salida");
tabla.setRowCount(lista.size());      int i=0;
for(c_Horario x:lista)      {
        tabla.setValueAt(x.getCodigo_hora(), i, 0);
        tabla.setValueAt(x.getHora_dia(), i, 1);        tabla.setValueAt(x.getHora_inicio(), i, 2);
tabla.setValueAt(x.getHora_salida(), i, 3);        i++;      }
      this.tbhorarios.setModel(tabla);        }
private void btncancelarActionPerformed(java.awt.event.ActionEvent evt) {
      this.dispose();   }
   private void btnlimpiarActionPerformed(java.awt.event.ActionEvent evt) {
      txtiddocente.setText("");      txtidmateria.setText("");
      txtnombremateria.setText("");      txtidlab.setText("");      txthorario.setText("");   }
   private void btnguardarActionPerformed(java.awt.event.ActionEvent evt) {
      else      {        c_Materia mat=new c_Materia();
```

```
mat.setCodigo_materia(Integer.parseInt(txtidmateria.getText()));
        mat.setCodigo_docente(txtiddocente.getText());
        mat.setCodigo_hora(Integer.parseInt(txthorario.getText()));
        mat.setCodigo_lab(Integer.parseInt(txtidlab.getText()));
        mat.setNombre_materia(txtnombremateria.getText());
        try{        c_Materia_Metodo.Ingresar_Materia(mat);
JOptionPane.showMessageDialog(this, "Ingreso correcto");
            int                    verifica=c_Materia_Metodo.COntarServicios(txtiddocente.getText(),
Integer.parseInt(txtidlab.getText()));
        if (verifica==0)        {
        c_Materia                    mat0=new                    c_Materia();
mat0.setCodigo_materia(Integer.parseInt(txtidmateria.getText())*1000);
        mat0.setCodigo_docente(txtiddocente.getText());        mat0.setCodigo_hora(0);
        mat0.setCodigo_lab(Integer.parseInt(txtidlab.getText()));
mat0.setNombre_materia("Servicio");   c_Materia_Metodo.Ingresar_Materia(mat0); }    }
catch(Exception ex){ JOptionPane.showMessageDialog(this, ex); }   }      }
   private void tblaboratoriosMousePressed(java.awt.event.MouseEvent evt) {
            int rec=this.tblaboratorios.getSelectedRow();
      this.txtidlab.setText(tblaboratorios.getValueAt(rec, 0).toString());   }
   private void tbhorariosMousePressed(java.awt.event.MouseEvent evt) {
      int rec1=this.tbhorarios.getSelectedRow();
      this.txthorario.setText(tbhorarios.getValueAt(rec1, 0).toString());   }
   private void tbdocenteMousePressed(java.awt.event.MouseEvent evt) {
      int rec2=this.tbdocente.getSelectedRow();
      this.txtiddocente.setText(tbdocente.getValueAt(rec2, 0).toString());   }
}
```

- **Fr_Menu.java**

```
package Interfaz_Usuario;public class fr_Menu extends javax.swing.JFrame {
   public fr_Menu() {            this.setTitle("Registro");
            this.setLocationRelativeTo(null);
this.setExtendedState(JFrame.MAXIMIZED_BOTH);          initComponents();   }
private void jmnuevo_docenteActionPerformed(java.awt.event.ActionEvent evt) {
Registro_Docentes obj = new Registro_Docentes();
dpcontenedor.add(obj);      obj.show();   }
private void jmsalirActionPerformed(java.awt.event.ActionEvent evt) {
      this.dispose();   }
   private void jmnuevolabActionPerformed(java.awt.event.ActionEvent evt) {
Registro_Laboratorios obj = new Registro_Laboratorios();
dpcontenedor.add(obj);      obj.show();   }
private void jmhoranuevoActionPerformed(java.awt.event.ActionEvent evt) {
Registro_Horarios obj = new Registro_Horarios();
dpcontenedor.add(obj);      obj.show();   }
private void jmnuevomateriaActionPerformed(java.awt.event.ActionEvent evt) {
Registro_Materias obj = new Registro_Materias();
dpcontenedor.add(obj);      obj.show();   }
private void jmregistroActionPerformed(java.awt.event.ActionEvent evt) {
Registro_Asistencia obj = new Registro_Asistencia();
dpcontenedor.add(obj);
      obj.show();
```