ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA MULTIMODAL PARA EL CONTROL DE ASISTENCIA EN IPREX

Trabajo de Titulación presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

AUTORES: JUAN CARLOS ALBUJA JÁCOME

JOSÉ LUÍS YÉPEZ GARCÍA

TUTOR: ING. WILLIAM CALVOPIÑA HINOJOSA

RIOBAMBA – ECUADOR

2016

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

El Tribunal del trabajo de Titulación certifica que: El trabajo: **DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA MULTIMODAL PARA EL CONTROL DE ASISTENCIA EN IPREX**, de responsabilidad de los señores Juan Carlos Albuja Jácome y José Luis Yépez García, ha sido minuciosamente revisado por los Miembros del Tribunal, quedando autorizada su presentación.

NOMBRE	**FIRMA**	**FECHA**
Dr. Miguel Tasambay, Ph.D **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	.....	.....
Ing. Franklin Moreno **DIRECTOR DE LA ESCUELA DE INGENIERÍA EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES**	.....	.....
Ing. William Calvopiña **DIRECTOR DEL TRABAJO DE TITULACIÓN**	.....	.....
Ing. Wilson Zuñiga **MIEMBRO DEL TRIBUNAL**	.....	.....

NOTA DEL TRABAJO DE TITULACIÓN: .....

Nosotros, Juan Carlos Albuja Jácome, y, José Luís Yépez García, declaramos ser los autores del presente trabajo de Titulación **"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA MULTIMODAL PARA EL CONTROL DE ASISTENCIA EN IPREX"** previo a la obtención del título de INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES, haciéndonos responsables por las ideas, criterios, doctrinas y resultados expuestos en esta Tesis, y declarando que el patrimonio intelectual de la misma pertenece a la Escuela Superior Politécnica de Chimborazo.

Juan Carlos Albuja Jácome C.C. N° 060401772-3

José Luís Yépez García C.C. N° 1804724555

DEDICATORIA

El presente trabajo va dedicado a mi familia y principalmente a mis padres Santiago Albuja y Patricia Jácome, quienes han sabido guiarme y apoyarme para poder alcanzar esta meta en mi vida, y a mis hermanos Rafael y Javier por estar siempre presentes aconsejándome y ayudándome.

Juan

El presente de Trabajo de Titulación va dedicado a mis padres José Vicente Yépez Loza y Olga Margarita García Zavala quienes han sido un apoyo incondicional a lo largo de mi vida académica y han hecho posible que pueda alcanzar las metas que me he propuesto y a mis hermanos, Miguel Ángel Yépez García y Juan Carlos Yépez García, con quienes he compartido alegrías y tristezas, éxitos y derrotas y con quienes siempre he podido contar.

José

AGRADECIMIENTO

Primero quiero agradecer a Dios por brindarme salud y la fuerza necesaria para poder alcanzar este objetivo en mi vida, a mi familia y amigos quienes han sabido forjarme como profesional viendo sus experiencias, de manera especial a mi Director de Tesis Ing. William Calvopiña quien mi brindo su apoyo necesario con sugerencias y comentarios que me sirvieron para culminar con éxito este proyecto

Juan

Al alcanzar esta nueva meta, como es la culminación del Trabajo de Titulación en éste, el final de mi preparación universitaria, me es imperioso mostrar mi agradecimiento a dios por permitirme llegar a este punto de mi vida y a las personas que directa e indirectamente forman parte de este logro. A mis padres José Vicente Yépez Loza y Olga Margarita García Zavala que me han dejado el mejor regalo que un padre puede otorgarle a sus hijos, la educación, y por ser en quienes siempre he podido confiar. A mis amigos Roberto Javier Vallejo Molina y Daniela Karina Lara Tuz que me han brindado su amistad incondicional y representaron un gran apoyo a lo largo de la vida universitaria

José

ÍNDICE DE ABREVIATURAS

ADC	Convertidor Analógico Digital
AIDs	Numeración y registro de aplicaciones
API	Interfaz de programación de Aplicaciones
AT	Tecnología Avanzada
CLK	Clock – Señal de Reloj
CPU	Unidad Central de Procesamiento
CRUD	Ccreate, read, update, delete
DAC	Convertidor Digital Analógico
DOD	Departamento de Defensa de los Estados Unidos
E.A.N	Numeración Europea de Artículos
EEPROM	ROM programable borrable
ETSI	Instituto Europeo de Normas de Telecomunicaciones
FCC	Comisión General de Comunicaciones
GND	Ground – Tierra
GPL	Licencia Pública General
HF	Frecuencia Alta
I/O	Input/Output – Entrada/Salida
I2C	Interfaz Inter Circuitos
IDE	Ambiente de Desarrollo Integrado – Entorno de desarrollo Interactivo
ISO	Organización Internacional de Normalización
JRE	Java Run Time
LCD	Pantalla de Cristal Líquido

LED	Diodos emisores e Luz
LF	Frecuencia Baja
MSR	Magnetic Stripe Reader – Lector de Banda Magnética
NXP	Next eXPerience
Oe	Oersteds
PC	Computadora Personal
PIN	Patilla para conexiones eléctricas
PWM	Modulación por ancho de pulsos
RAM	Memoria de Acceso Aleatorio
RF	Radiofrecuencia
RFID	Identificación por Radio Frecuencia
RFU	Contacto Reservado
ROM	Memoria de solo lectura
RST	Reset o Reinicio
SCL	Línea de Reloj
SCQL	Lenguaje de consulta estructurado de tarjeta
SDA	Línea de Daros
SGBD	Sistema Gestor de Base de Datos
TTL	Through The Lens o a través del lente
U.P.C	Código de Producto Universal
UHF	Frecuencia Ultra Alta
USB	Bus Universal en Serie
VCC	Voltaje de corriente continua positiva
VPP	Voltaje de Pico a Pico

CONTENIDO

ÍNDICE DE TABLAS

ÌNDICE DE FIGURAS

RESUMEN

En el presente trabajo se analizó, diseñó e implementó un sistema multimodal utilizando las tecnologías de Identificación por Radio Frecuencia (RFID), ZigBee y cámara TTL para un sistema de control de asistencia de personal. Se estableció un análisis comparativo entre varias alternativas para la selección de las tecnologías más idóneas para la implementación. El canal de comunicación inalámbrico entre los dispositivos, utilizó el protocolo ZigBee que está definido sobre el estándar del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE 802.15.4). Para el diseño y posterior implementación se necesitó de un nodo controlador desarrollado sobre una placa reducida Arduino Uno que administró y ayudó como intermediario entre los dispositivos: RFID, que posee un código único de identificación y cámara TTL, para recoger la información y enviarlas por el canal hacia un gestor de base de datos ejecutado sobre MySQL Workbench desplegada en una PC para su verificación y almacenamiento. En la parte de recepción se contó con una PC en la cual están desarrolladas aplicaciones basadas en Java, con las cuales se realizó las funciones siguientes: 1. El tratamiento de la imagen enviada por ZigBee para su reconstrucción y almacenamiento, 2. Provee una interfaz de usuario para la administración de la base de datos. Los resultados obtenidos muestran que el sistema posee una mejora del 58,33% en velocidad de datos en relación con el sistema manual. Se concluye que este sistema multimodal es efectivo para controlar la asistencia de personal fusionando RFID, ZigBee y Cámara TTL, obteniendo registros en tiempo real. Se recomienda la utilización del sistema a las empresas, instituciones, en donde es difícil poseer conexión cableada y línea de vista directa para el tratamiento de información.

Palabras claves: <IDENTIFICACIÓN POR RADIOFRECUENCIA [RFID] >, < PROTOCOLO ZIGBEE >, <NODO COORDINADOR >, < GESTOR DE BASE DE DATOS>, <COMUNICACIÓN INALÁMBRICA >, < ARDUINO>, < SISTEMA MULTIMODAL>, < TELECOMUNICACIONES>.

SUMMARY

In this research we analyze designs and implement a multimodal system using technologies of Radio Frequency Identification (RFID), ZigBee and camera TTL for a personal control system of support. A comparative analysis of various alternatives for selection of the most suitable technologies for implementation was established. The wireless communication channel between the devices used the ZigBee protocol that is defined on the standard of the Institute of Electrical and Electronics Engineers (IEEE 802.15.4). For the design and subsequent implementation, was required a controller node developed on an Arduino, it was reduced plaque One, it was given and helped as an intermediary between devices: RFID, which has a unique identification code and camera TTL, to collect information and send them through the channel to a database manager executed on MySQL Workbench displayed on a PC for verification and storage. It was worked at the reception PC in which was developed a Java-based applications, with which the following functions are performed: 1. The treatment of the image was sent by ZigBee to rebuild and storage, 2. It. provides an interface user to manage the database. The results show that the system has a 58.33% of improvement in data rate relative to the manual system. It is concluded that this multimodal system is effective to control personal assistance merging RFID, ZigBee and TTL camera, to get records in real time. It is recommended the system implementation to companies, and institutions, where it is difficult to have wired connection and direct sight line for processing data.

Keywords: <RADIO FREQUENCY IDENTIFICATION [RFID]>, <PROTOCOL ZIGBEE>, <COORDINATOR NODE>, <DATABASE MANAGER>, <WIRELESS COMMUNICATION>, <ARDUINO>, <MULTIMODAL SYSTEM> <TELECOMMUNICATIONS>.

INTRODUCCIÓN

El presente trabajo denominado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA MULTIMODAL PARA EL CONTROL DE ASISTENCIA EN IPREX", trata sobre la combinación de diversas tecnologías en un único sistema capaz de controlar la asistencia del personal en las instalaciones de IPREX.

En dicho sistema se utilizará redes inalámbricas que utilizan ondas electromagnéticas para enviar y recibir información, este método será utilizado para establecer la comunicación entre la PC que tenga acceso a la base de datos y el dispositivo encargado de recolectar la información obtenida de la cámara TTL y lector NFC/RFID para su posterior verificación y almacenamiento.

Este proyecto va orientado a una investigación aplicada tecnológica, debido a la recopilación y utilización de conceptos teóricos o investigación pura para la obtención de conocimiento útil y métodos dirigidos a la resolución de problemas mejorando tecnologías existentes, innovándolas, para la creación de productos nuevos. Los métodos utilizados son: inductivo, sintético, deductivo y de análisis.

El sistema a diseñar e implementar va a contar con tres partes fundamentales, la primera será la encargada de establecer el canal para la comunicación entre el Nodo Coordinador y el Nodo de Verificación y almacenamiento, el segundo realizará el control de los elementos lector RFID, pantalla LCD, y cámara TTL para su envió (Nodo Coordinador) utilizando placas reducidas Arduino, y el tercero ejecutará la verificación y almacenamiento de imágenes.

Antecedentes

La Tecnología RFID o Radio Frecuency Identification por sus siglas en inglés, surgió a inicios de la década de 1940 durante la Segunda Guerra Mundial por el ejército Norte Americano, para el reconocimiento a distancia de aviones amigos o enemigos. La identificación era posible gracias a un transponedor integrado que enviaba una señal única de radio frecuencia a un lector en una estación fija (Santacruz & Suntaxi, 2007: p. 2; Lara, 2008: p. 1).

A partir de su creación la tecnología RFID fue utilizada para la identificación de locomotoras, en seguridad de plantas nucleares e inclusive la fabricación de automóviles, hasta que en los años 90 se logra la miniaturización e integración de la tecnología en un solo chip electrónico. La

miniaturización de la tecnología abrió las puertas a múltiples aplicaciones en sectores como el control de acceso, prepago, monitorización y autenticidad, todo gracias a los avances técnicos en aspectos como el alcance, seguridad, almacenamiento, o velocidad de lectura, entre otros.

En la actualidad la tendencia de los sistemas basados en tecnología RFID abarcan un sinfín de aplicaciones en ámbitos industriales, comerciales, defensa y seguridad, un ejemplo de esto es la Empresa Walt-Mart y el Departamento de Defensa (DOD) de los estados Unidos siendo los primeros en implementar RFID en sus procesos, para mejorar su cadena de suministros (Libera, 2010: p.18).

La importancia de los sistemas de acceso o asistencia radica en las prestaciones de seguridad que estos ofrecen, teniendo en cuenta que uno de los principales valores de las empresas en la actualidad es la información y la seguridad de esta, que converge con la seguridad física, siendo una herramienta que permite establecer políticas de seguridad dentro del entorno controlado.

El hardware libre se ha caracterizado por poner su diseño a disposición del público, de modo que cualquiera puede estudiar, modificar, distribuir, hacer y vender el diseño o el hardware que se sustente en dicho diseño, que emplea elementos y materiales inmediatamente disponibles, procesos estandarizados, infraestructura abierta, contenido no restringido y herramientas de diseño libre, siendo Arduino una de sus implementaciones más representativas.

La plataforma Arduino está basada en hardware y software flexible y fácil de usar, que por sus prestaciones y libre acceso se ha convertido en una alternativa de diseño y creación de proyectos electrónicos, de bajo costo con más de 150000 dispositivos Arduino o compatibles con el mismo alrededor del mundo.

La preparación preuniversitaria se ofrecía originalmente en academias preuniversitarias, cuya demanda era mayormente de alumnos provenientes de las promociones de 5to de Secundaria, quienes seguían un grupo de cursos entre Matemáticas, Ciencias y Letras, los cuales eran desarrollados por docentes preuniversitarios, generalmente estudiantes universitarios de los ciclos medios y avanzados.

IPREX es un centro de formación pre Universitaria, que brinda sus servicios con una planta docente totalmente capacitada, y apta para impartir los conocimientos necesarios previo al ingreso a las instituciones de tercer nivel, con sus instalaciones ubicadas en la ciudad de

Riobamba, y con el objetivo de mejorar su desempeño buscan un mejor control de asistencia para su personal.

Justificación

En la actualidad la necesidad de las personas y/o empresas de poseer cierto grado de control en la asistencia de personal, nos lleva a la utilización de tecnologías capaces de obtener datos precisos de una manera automática, una de ellas la Identificación por Radio Frecuencia que utiliza ondas electromagnéticas del espectro radioeléctrico para enviar los datos, cuyos resultados son obtenidos de dispositivos remotos llamados etiquetas o tags RFID.

Se Implementará un sistema de control de asistencia basado en Tecnología RFID, para proveer un mecanismo de actualización de datos y control de asistencia del personal, en la sucursal norte de la empresa, sin necesidad de registro directo con acceso inalámbrico e identificadores únicos de radio frecuencia.

El desarrollo del proyecto se realizará mediante la utilización de etiquetas de identificación que internamente poseen chips programables que hacen posible su rastreo y localización, lectores (TAGS RFID) de baja frecuencia con Micro-antenas receptoras y trasmisoras Zigbee montadas sobre plataformas Arduino, que enviarán los datos a un ordenador que será el encargado de tener acceso al servidor de base de datos para su verificación en la sucursal de IPREX , empleando bandas de frecuencia libres respetando estándares, además se contara con la utilización de una cámara web para poder obtener una imagen o foto de la persona que acceso mediante el TAG RFID, siendo posteriormente almacenada la información en el ordenador.

Con esto se lograra que IPREX posea un sistema de control de asistencia por RFID, proporcionando un sistema de actualización de datos, incrementando su velocidad de lectura, mejorando la seguridad de los recursos de la Empresa, lo que ayuda a optimizar la correcta utilización de los recursos tanto humanos como tecnológicos, mediante la obtención de registros en tiempo real.

Objetivos

Objetivo General.

Diseñar e implementar un sistema multimodal para el control de asistencia en IPREX

Objetivos Específicos.

- Investigar el funcionamiento de la Tecnología RFID, Zigbee y Arduino.
- Diseñar el sistema multimodal de asistencia mediante tecnología RFID, Zigbee y Arduino.
- Implementar el sistema multimodal de asistencia con capacidad de almacenamiento de imágenes.
- Validar el funcionamiento del sistema multimodal, de acuerdo al diseño propuesto.

CAPÍTULO I

1 MARCO TEÓRICO

1.1 Sistemas de Control

La monitorización y registro de las actividades dentro de las empresas se ha convertido en una de las principales medidas de seguridad que con el tiempo y su desarrollo, la automatización de este proceso ha desembocado en una gama muy extensa de soluciones y nuevas tecnologías para este efecto, por lo que se provee un acercamiento al funcionamiento y prestaciones que presentan los principales exponentes de los sistemas de control automatizado que existen en el mercado.

1.1.1 Sistemas establecidos con tarjetas inteligentes

Estos sistemas están basados en tarjetas inteligentes, estas tarjetas, desde su creación en la década de los 70, han tenido como característica principal el brindar la capacidad para almacenar información de diversos tipos, con el objetivo de proveer portabilidad y facilitar la accesibilidad a dicha información, garantizando la seguridad de la misma.

Las tarjetas Inteligentes son denominadas generalmente como tarjetas de circuitos integrados, que constan de un chip con un microprocesador interno, encapsulado en una tarjeta PVC, con la capacidad de comunicarse mediante contactos o inalámbricamente con un dispositivo o central receptora, para manejar información personal, estados de cuenta, información clínica, claves de acceso, etc (Rodríguez, 2013: p.5).

1.1.2 Tipos de Tarjetas

A menudo se denominan tarjetas inteligentes a todas las tarjetas de circuito integrado, sin embargo estas últimas comprenden a tarjetas de memoria y tarjetas microprocesadas o inteligentes.

1.1.2.1 Según las características del Chip

Tarjetas de Memoria.- El chip de estas tarjetas solo tienen la capacidad de almacenar información, y el acceso a la esta puede ser libre o contar con un circuito de seguridad con acceso mediante clave de seguridad.

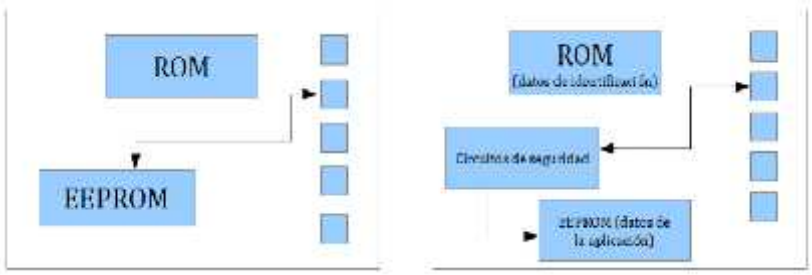


Figura 1- 1: Acceso Libre y Acceso Protegido

Fuente: Rodríguez, A. 2013.

Tarjetas Inteligentes.- Dentro de su Chip existe un microprocesador el que hace posible ejecutar comandos, programas y procesar los datos contenidos en el o adquiridos por dispositivos externos. En las tarjetas más modernas los sistemas operativos posibilitan la integración de programas sobre una sola plataforma.

Figura 2-1: Tarjeta Inteligente

Fuente: http://www.quaronline.com

1.1.2.2 Según el Método de Acceso

Tarjetas de Contacto.- Son tarjetas cuya única interfaz con el lector terminal para administrar su información es mediante los contactos en su chip. Cuentan con 8 contactos de los cuales se emplean solo 6 para la trasferencia de los datos.

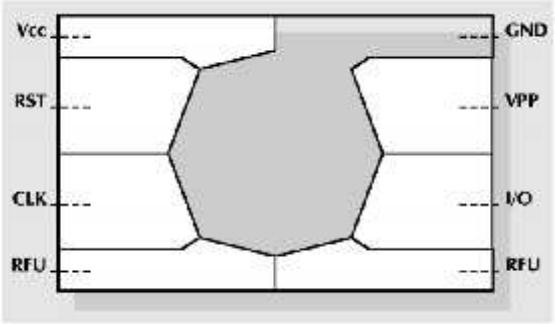


Figura 3-1: Contactos del Chip de una Tarjeta Inteligente

Fuente: http://www.smartcard.co.uk

VCC.- Alimentación.

RST.- Reset.

CLK.- Reloj.

RFU.- Contacto reservado.

I/O.- Entrada/Salida Info.

VPP.- Voltaje de programación.

GND.- Tierra.

Las tarjetas de contacto cumplen con normativas ISO, que especifican las características de fabricación y funcionamiento. Estas son:

ISO 7816:

Define las siguientes características:

- Dimensiones de la tarjeta.
- Tolerancias a tensiones y radiaciones electromagnéticas.
- Tolerancia a la estática.
- Localización del Chip.
- Ubicación, número, dimensión y dimensiones de los contactos del chip.
- Protocolos de comunicación y niveles de tensión.
- Formato de comandos entre tarjeta y programador, y sus respuestas.
- Numeración y registro de aplicaciones AIDs.
- SCQL – Lenguaje de consulta estructurado de tarjeta.
- Comandos, técnicas de cifrado y otros métodos de seguridad.
- Señales eléctricas y Reset para una tarjeta síncrona.

Tarjetas sin Contacto.- Guardan gran similitud con las tarjetas de contacto en cuanto a su estructura y funcionalidad, la diferencia es que estas tarjetas realizan la transmisión de sus datos mediante campos electromagnéticos, es decir utilizan antenas para la comunicación con el terminal receptor, establecidas a distancias de 10 cm a 50 cm, definidas en los estándares ISO

14443 e ISO 15693 respectivamente, en las que, además se definen sus características físicas, frecuencia de operación, potencia de transmisión y protocolos de comunicación (Molina, 2007: p.29-30).

Tarjetas Hibridas y Duales.- Son tarjetas que poseen ambas tecnologías, inalámbrica y de contacto con chip microcontrolado o de memoria. Las tarjetas Duales esencialmente tienen las funcionalidades de las tarjetas híbridas, con la diferencia que cuentan con un único circuito integrado.

1.1.2.3 Según su Formato

Las tarjetas Inteligentes cumplen con distintos formatos, que por razones de compatibilidad con los terminales receptores estos están definidos en el estándar ISO 7816-1, teniendo así los siguientes formatos (Molina, 2007: p.31-32):

ID-000.- Tarjetas utilizadas en telefonía móvil con tecnología GSM, a estas tarjetas se las denomina Mini SIM, que sirve para identificar a los clientes móviles dentro de la red telefónica.

Dentro de este tipo de tarjetas han surgido las llamadas Micro SIM y Nano SIM que se han ajustado al tamaño del chip, como solución a dispositivos más pequeños.

ID-00.- Tarjetas de tamaño intermedio, poco comercializadas, destinadas a transacciones de pequeños montos de dinero, identificación entre otras operaciones comúnmente reemplazadas por tarjetas ID-1.

ID-1.- Tarjetas conocidas como SIM Standard, son las más utilizadas debido a su semejanza con las tarjetas de crédito, empleadas para identificación, transacciones, acceso, etc.

1.1.2.4 Según La Estructura de su Sistema Operativo

Basadas en Sistema de Ficheros Aplicaciones y Comandos.- Son tarjetas provistas de un sistema de archivos y un sistema operativo con una o varias aplicaciones y un Shell accesible mediante Interfaces de Programación de Aplicaciones.

Tarjetas Java.- Cuentan con las mismas capacidades que las tarjetas anteriores, con la diferencia que su programación está basada específicamente en el entorno Java.

1.1.3 Arquitectura de una tarjeta inteligente

El chip presente en las tarjetas inteligentes, para poder cumplir con las funciones de almacenamiento y procesamiento de la información, su arquitectura se basa en la de Von Neumann, es decir, constan de un CPU encargado del procesamiento de la información, en una unidad de almacenamiento que cuenta de dispositivos de entrada y salida para la comunicación con el entorno. La unidad de almacenamiento está compuesta por tres tipos de memoria: ROM, EEPROM y RAM que junto al microprocesador están embebidos dentro del mismo chip (Rodríguez, 2013: p.7).

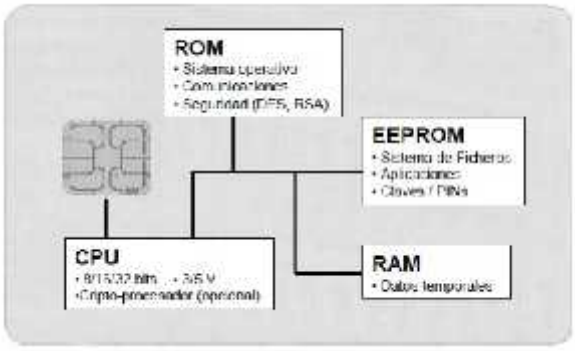


Figura 4-1: Arquitectura del Chip de una Tarjeta Inteligente
Fuente: Rodríguez, A. 2013.

Unidad Central de Proceso (CPU).- Encargado del procesamiento de la información, existen de 8 bits, 16 bits y 32 bits, opcionalmente puede tener un coprocesador criptográfico, especializado en las operaciones aritméticas, y mejorar los procesos de seguridad y criptografía.

Memoria ROM (Read Only Memory). - Memoria no volátil, es decir que la información grabada en ella no se pierde al desconectar la fuente de alimentación, de solo lectura, que contienen las configuraciones del fabricante como son el sistema operativo, aplicaciones y datos fijos de usuario. Esta clase de memoria solo se puede escribir una vez.

Memoria EEPROM (Erasable Programmable Read-Only Memory). - Memoria no volátil, similar a la ROM, con la particularidad de ser re-escribible y funcionalmente equivalente al disco duro de un computador. En esta memoria se almacena datos de usuario y aplicaciones que

en las tarjetas más modernas contiene el programa específico de cada aplicación mientras que la ROM se limita al sistema operativo e instrucciones básicas.

Memoria RAM(Random Access Memory).- Es una memoria volátil, es decir que la información contenida en ella se pierde al desconectar la energía de la fuente de alimentación, empleada para almacenar los datos temporales del sistema, usuarios y aplicaciones, con una velocidad de escritura mil veces mayor a la EEPROM.

1.1.4 Sistemas Biométricos

Los sistemas biométricos están basados en los conceptos de biometría, que proviene de las palabras bio que significa vida y metría que quiere decir medida, por lo que estos sistemas trabajan con la información de las características físicas o el comportamiento de los individuos y las comparan con plantillas de estas características biométricas almacenadas en una base de datos (Borja, p.2).

1.1.4.1 Tipos de Sistemas Biométricos

Dependiendo del diseño o enfoque en el que sea concebido un sistema biométrico, estos pueden ser de verificación o de identificación, siendo necesaria para su funcionamiento la presencia de indicadores o características a ser medidas, los mismos que deben cumplir con los siguientes requerimientos:

Universalidad.- Todos los individuos deben poseer esa característica.

Unicidad.- La mínima posibilidad de que una característica se repita entre individuos.

Permanencia.- La característica se mantiene a través del tiempo.

Cuantificación.- La característica puede ser medida en forma cuantitativa.

Sistemas de Verificación.- Se autentica las credenciales de una persona comparando un PIN previamente ingresado correspondiente a uno de la base de datos y la información obtenida por

los sensores del sistema con la plantilla biométrica correspondiente al PIN, para determinar si es quien dice ser.

Sistemas de Identificación.- Se examina la base de datos con el fin de encontrar coincidencias a una petición de identificación de una persona, se compara la información obtenidos por los sensores con todas las plantillas biométricas existentes, para determinar si consta o no en la base de datos, evitando que una persona use múltiples identidades.

1.1.4.2 Arquitectura de los Sistemas Biométricos

Dentro de la estructura de los Sistemas Biométricos se encuentran tres componentes básicos. Primero se tienen los sensores encargados de adquirir los indicadores biométricos de un individuo ya sea de forma analógica o digital. En segundo lugar tenemos el procesador biométrico encargado de la compresión, procesamiento, almacenamiento y comparación de los datos adquiridos. Por último tenemos a la interfaz de aplicaciones que hacen posible la administración de los datos almacenados.

Teóricamente estos componentes pueden ser interpretados en dos módulos funcionales dentro de la arquitectura de los sistemas biométricos (Pupiales, 2009: p.14):

Módulo de inscripción.- Este se encarga de adquirir la información de los indicadores biométricos, por medio de lectores especializados en características específicas de los individuos como pueden ser de reconocimiento facial, escaneo de iris, lector de huellas digitales, reconocimiento de voz o firmas digitales entre otros, los mismos que entregan una representación digital de dichos indicadores, y de estos datos extraer las características más relevantes. Esta información es almacenada dentro de una base de datos que servirán como plantillas biométricas para posteriores accesos al sistema.

Módulo de identificación.- Este módulo se encarga del reconocimiento de los individuos, una vez adquirida la información de los indicadores biométricos de un individuo, el extractor de características recoge una representación más compacta con el formato de las plantillas biométricas, la que es enviada al comparador de características que las comprueba con la información en la base de datos y determinar su identidad.

1.1.5 Sistemas basados en tarjetas magnéticas

Los sistemas de identificación automática basados en tarjetas de banda magnética, debido a su bajo costo de implementación y gran confiabilidad, son uno de los sistemas más ampliamente utilizados, especialmente en el sistema bancario tradicional, sistemas de acceso y asistencia empresarial, entre otros ámbitos donde se requiera un alto grado de seguridad y rapidez en la captura de datos.

1.1.5.1 Funcionamiento

La información digitalizada se almacena al polarizar las partículas microscópicas en la resina de la banda magnética en conjunto con una capa magnética similar a las empleadas en las cintas de audio o video, empleando la técnica de la grabación digital o codificación en binario que dependiendo de la polaridad de cada partícula esta representará un 0 ó 1, siendo posible su reconfiguración lo que las hace muy versátiles en sus aplicaciones.

La banda angosta en la banda magnética que corre paralela al borde de referencia en el cual se codifican los datos es llamada pista o track. Es usualmente de unos 2.5 milímetros y puede localizarse a cualquier distancia del borde de referencia. El número de pistas es determinado enteramente por el ancho de la banda magnética.

Para acceder o reescribir a los datos almacenados en la banda magnética se emplean lectores de banda magnética (magnetic stripe reader – MSR) o codificadores completos comúnmente llamados cabezales magnéticos. El proceso de lectura o escritura de la información se realiza manteniendo contacto entre la banda y la cabeza magnética, manteniendo un movimiento de la banda con respecto al lector, dicho movimiento puede ser producido por el usuario o por motores como en los cajeros.

1.1.5.2 Coercividad

Este término es empelado para determinar la fuerza de un campo magnético que sea capaz de afectar la información codificada en una tarjeta de banda magnética, y por tanto la inmunidad que presenta esta información a ser dañada por dichos campos magnéticos, cuya unidad de medida está dada en Oersteds (Oe) (Pupiales, 2009: p.18).

Como ejemplo tenemos que la coercitividad de una tarjeta de crédito típica esta alrededor de 300 Oe dentro del rango de baja coercitividad o “LoCo” que típicamente se vale de partículas de óxido de hierro, es decir que un imán doméstico podría dañar la información presente en la tarjeta.

Las tarjetas de banda magnética con alta coercitividad o “HiCo”, emplean comúnmente partículas de ferrita de bario con una coercitividad comprendida entre 2500 y 4000 Oe, para el proceso de codificación de estas tarjetas se emplea una corriente eléctrica mayor en la cabeza magnética, con lo que se logra una vulnerabilidad prácticamente nula a imanes domésticos y se reducen los riesgos de daños accidentales de la información.

1.1.6 Sistemas basados en código de barras

Los códigos de barras se establecen mediante una serie de barras negras y espacios en blanco de varios anchos y de esta manera codificar la información. Esto códigos pueden ser leídos con lectores, escáneres o unidades de rastreo que posibilitan la decodificación de la información contenida en ellos, midiendo la luz reflejada y traduciendo la clave alfanumérica para una extracción completa de los datos (Pupiales, 2009: p.21).

Figura 5-1: Símbolo del Código de Barras
Fuente: http://upload.wikimedia.org/

El símbolo del código de barras se refiere a la visualización física del mismo, mientras que la simbología hace referencia a la forma de codificación de la información en las barras y espacios del símbolo del código.

1.1.6.1 Aplicaciones del código de barras

El código de barras encontró su campo de aplicación en prácticamente cualquiera de los sectores económicos que se beneficien con este tipo de tecnología de captura de datos, ya que la

disponibilidad de la información es vital para la toma de decisiones dentro de las distintas actividades económicas de un país.

Entre algunos ejemplos de sus aplicaciones tenemos:

- Control de inventarios
- Control de movimientos
- Control de acceso
- Control de calidad
- Rastreo de actividades
- Levantamiento electrónico de pedidos
- Facturación
- Bibliotecas

1.1.6.2 Simbología del código de barras

Existen varios tipos de simbología dentro de los códigos de barras, ya que estos se especializan dependiendo de la actividad en la que se los empleen, teniendo así terminologías para la salud, comercios, empresas e industrias, etc., que no son intercambiables entre sí.

La gran variedad de tipos de códigos de barras es debida a las diferentes aplicaciones que existen y en las que se deben resolver problemas específicos y cumplir con las características propias de cada una, por lo que cada tipo de código se especializa dependiendo del sector de aplicación (Pupiales, 2009: p.23).

Los Códigos de Barras se dividen en dos grandes grupos dependiendo de la simbología:

- Códigos de Barras de Primera Dimensión.
- Códigos de Barras de Segunda Dimensión.

1.1.6.3 Códigos de Barras de Primera Dimensión.

Código de Producto Universal – Universal Product Code (U.P.C.): Es la simbología mayormente utilizada en los comercios minoristas de Estados Unidos, capaz de codificar solamente información numérica. Este estándar es denominado UPC-A, que consta de 12

dígitos de los cuales el primero es el número del sistema que determina el tamaño y peso fijos que posee un producto.

Figura 6-1: Símbolo Código U.P.C

Fuente: barcodeisland.com, 2006.

Para artículos pequeños se emplea el código UPC-E.

Figura 7-1: Símbolo Código U.P.C-E

Fuente: computalabel.com, 2006.

Numeración Europea de Artículos – European Article Numbering (E.A.N.): Este sistema es el equivalente europeo del UPC-A estadounidense, concebido como estándar inter nacional y aceptado a nivel mundial. Mediante este sistema se identifican los productos comerciales mediante código de barras especificando el país, la empresa y el tipo de producto junto con una clave única internacional.

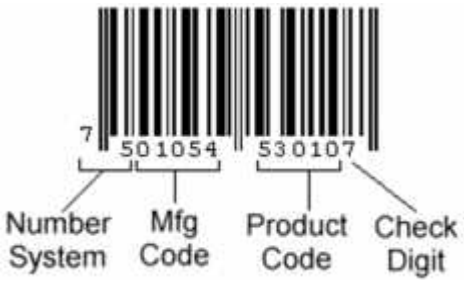


Figura 8-1: Símbolo Código E.A.N

Fuente: http://www.barcodeisland.com

Para artículos de tamaño reducido se emplea el código EAN-8

Figura 9-1: Símbolo Código E.A.N-8

Fuente: http://www.labeljoy.com/

Código 39: Es el código más común para aplicaciones regulares, debido a que puede contener texto y números, el mismo que puede ser leído por la mayoría de lectores de código de barras sin alterar su configuración.

Figura 10-1: Símbolo Código 39

Fuente: http://www.labeljoy.com/

Código 128: Código alfanumérico muy compacto en el que es posible representar todos los caracteres ASCII si duplicidad como el caso del código 39 extendido. Para el caso de cuatro o más números consecutivos, estos se codifican en modo de doble densidad, es decir que se codifican dos números en la misma posición.

Figura 11-1: Símbolo Código 128

Fuente: http://www.labeljoy.

Código Entrelazado 2 de 5: Código numérico ligeramente más extenso que el UPC-A cuando esta codificado con 10 dígitos, siendo posible codificar cualquier número par de dígitos y para el caso de un número impar se coloca un cero al inicio del código, dedicado para lectores de montaje fijo.

Figura 12-1: Símbolo Código Entrelazado 2-5
Fuente: http://www.labeljoy.com/

Código Posnet: Este código de barras es exclusivo para el Servicio Postal de Estados Unidos, empleado para la codificación de códigos postales para aumentar la velocidad de procesamiento y entrega del correo en el país.

Figura 13-1: Símbolo Código Posnet
Fuente: https://pt.wikipedia.org/

1.1.6.4 Código de Barras de Segunda Dimensión

Código PDF 417: Código que cuenta con una simbología de alta densidad, con la característica de ser un Portable Data File o Archivo de Información Portátil (PDF), es decir que no requiere acudir a una base de datos externa, gracias a que tiene una capacidad de hasta 1800 caracteres alfanuméricos y especiales.

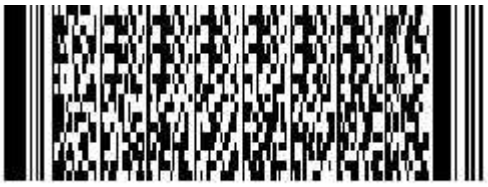


Figura 14-1: Símbolo Código PDF 417
Fuente: https://ztfnews.files.wordpress.com/

Código Maxicode: Empleado para procesamiento de información a alta velocidad, que cosiste de un arreglo de 866 hexágonos que codifican la información en binario de forma seudo-aleatoria. Este código puede leerse en cualquier orientación con respecto al lector.

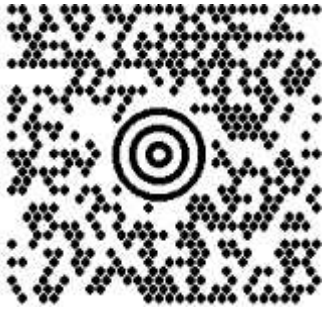


Figura 15-1: Símbolo Código Maxicode

Fuente: https://www.tec-mex.com.mx/

Código Datamatrix: Es un Código alfanumérico con una capacidad de hasta 2334 caracteres, cuyas aplicaciones son variadas como la codificación de direcciones postales, control de calidad, identificación de desechos peligrosos, almacenamiento de datos sobre productos farmacéuticos, etiquetado de boletos de lotería y codificación de cheques en instituciones financieras, etc.

Figura 16-1: Simbología Código Daramatrix

Fuente: http://www.barcode-soft.com/

Decodificación del Código de Barras: Una vez codificada la información en uno de los diferentes tipos de códigos de primera o segunda dimensión, para su recuperación se emplean lectores de códigos de barras, los mismos que decodifican esta información mediante la digitalización de las variaciones de luz que se producen cuando el lector hace un barrido al símbolo del código.

El lector posee una fuente de luz visible o infrarroja, la misma que es absorbida por las barras oscuras mientras los espacios en blanco la reflejan, es esta fluctuación de la luz la que el lector convierte en impulsos eléctricos que recrean de forma binaria el código de barras.

El decodificador emplea algoritmos matemáticos para decodificar la información obtenida del código para su posterior administración y almacenamiento. Los lectores emplean diodos

emisores de luz (LED), laser de Helio-Neón o diodos láser de estado sólido para escanear el símbolo del código (Pupiales, 2009: p.30)

1.1.6.5 Tipos de Lectores

Lápiz Óptico o Wand:

Se desliza haciendo contacto con el código, es económico, baja velocidad de lectura y una baja tasa de primera lectura, requiere de decodificador manual por teclado y depende en gran medida de la calidad de impresión del código.

Láser de Pistola:

Emplea una luz láser a mayor frecuencia que la empleada en el lápiz óptico llamada “Hand Held Laser Compatible”, con un alcance de 5 a 30 cm y en el caso de etiquetas papel retroreflectivo de hasta 15 cm, tiene un alto coste y puede presentar problemas de lectura con demasiada luz ambiental.

Laser Omnidireccional:

Este lector emplea un patrón de rayos láser que permiten leer el símbolo del código de barras en cualquier orientación, sin necesidad de decodificador de teclado con alcances máximos similares a los del láser de pistola.

Lectores de Códigos de Dos Dimensiones:

Son dispositivos Ópticos que escanean el símbolo del código basándose en la composición de su estructura que consiste en zona de inicio seguida de uno o más caracteres de datos, uno o dos caracteres de verificación opcionalmente y la zona de término del código. Esta información es enviada a un ordenador para su procesamiento.

1.2 Tecnología de Identificación por Radio Frecuencia (RFID)

1.2.1 Introducción

RFID proviene de las siglas Radio Frequency IDentification, siendo un sistema que utiliza las ondas del espectro radioeléctrico para poder enviar y recibir datos de una forma remota, es una tecnología que ha sido utilizada en los últimos treinta años,

Este tipo de tecnología ha sido muy manejada a nivel de consumo masivo y abastecimiento para obtener información en tiempo real de los productos que se tiene en stock, para conocer el estado de los productos. "Optimizando la disponibilidad del producto a nivel de consumo masivo, visibilidad absoluta y precisa acerca de los inventarios y mayor eficiencia en la manipulación de materiales (Telectrónica, 2006:p.5)."

1.2.2 Fundamentos

Un sistema RFID posee cuatro componentes necesarios para su funcionamiento: lectores, tags, antenas y un host que funcione como computadora central. Se dirá que un tag es un microchip que posee una antena mientras que el lector es el encargado de leer y escribir la información sobre el tag. Esta técnica funciona gracias a que el lector emite ondas de radios que el tag puede encontrar mientas se encuentre en un rango de lectura (sin contacto físico o en línea de vista con el lector identificándose a sí mismo.

1.2.3 Sistema mediante Identificación por radio Frecuencia

En la actualidad existen diversos tipos de sistema basados en RFID que funcionan en distintas frecuencias, cada una presenta ventajas y desventajas, por tal motivo es necesario conocer cada uno de ellas para determinar los equipos más idóneos dependiendo de las necesidades y exigencias para resolver el problema o necesidad a resolver (Loft Media Publishing, 2010, p: 9-13).

Entre las frecuencias más comunes tenemos:

1.2.3.1 Banda LF (Low Frequency): Funcionan en la frecuencia de 125 Khz.

Ventajas Tags LF:

- Es utilizado en todo el mundo.
- Es menos susceptibles a condiciones que presenten líquidos y metales.

Desventajas Tags LF:

- No realiza lecturas múltiples.
- Es más costosa en comparación con los tags pasivos
- Su rango de lectura es de pocos centímetros (50cm).
- Está limitado por la cantidad de memoria y sujeto a una sola lectura a la vez.

1.2.3.2 Banda HF (High Frequency): Funciona en la banda de 13.56 MHz

Ventajas Tags HF:

- Funcionan a nivel global.
- Permite proveer seguridad para el almacenamiento y envió de la información.
- Funciona sin problemas con líquidos y metales.
- Menor costo en comparación con los tag RFID de LF.
- Poco rango de lectura (aproximadamente 1m).

Desventajas Tags HF:

- Al poseer poco rango de lectura, sus aplicaciones tanto de logística (mejor forma para producir y distribuir artículos) así como la gestión de productos/almacenes puede variar.

1.2.3.3 Banda UHF (Ultra High Frequency): Puede funcionar entre el rango de 868 – 928 MHz.

Ventajas Tags UHF (868 – 828 MHz)

- Mayor rango de lectura (aproximadamente 9 metros)
- Bajo costo de tags
- Velocidad en la lectura de 1200 tags/seg

Desventajas Tags UHF (868 – 828 Mhz)

- Susceptible a interferencia por líquidos y metales
- No existe estándares.

1.2.3.4 Banda UHF (Ultra High Frequency): Puede funcionar entre el rango de 2.4 – 5.8 Ghz.

Ventajas Tags UHF (2.4 – 5.8 Ghz)

- Velocidad de trasmisión aceptable.

Desventajas Tags UHF (2.4 – 5.8 Ghz)

- Rango de lectura alrededor de 2m.

1.2.4 Equipos

1.2.4.1 Etiquetas RFID o TAGS-RFID

También se las conoce como etiquetas RFID, estas etiquetas constan de un microchip montado en un sustrato (inlay) PET flexible que contiene una antena incorporada. Cabe destacar que el rango de lectura depende de la antena que posee la etiqueta RFID, para poder captar la señal que proviene del lector RFID, en consecuencia al querer mayor distancia necesitaremos una antena de mayor tamaño, por ende la etiqueta también se verá afectada por la antena.

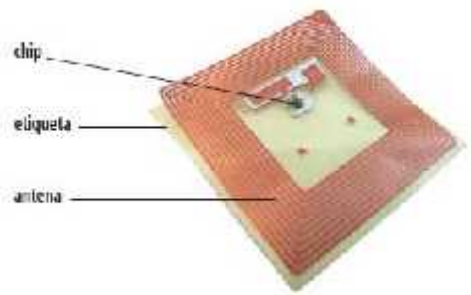


Figura 17-1: Circuito Inteligente en TAGS
Fuente: (Telectrónica, 2010).

Los Tags RFID pueden ser de dos tipos:

Etiquetas Activas: Tienen su propia fuente de alimentación incorporada, son utilizadas para aplicaciones que conlleven grandes distancia con ejecución ininterrumpida, como equipamiento militar, control de depósitos o seguimientos de contenedores marítimos.

Etiquetas pasivas: No poseen fuente de alimentación incorporada, la energía que necesita para su funcionamiento la obtiene a partir del lector, mediante ondas electromagnéticas que inducen

corriente en la antena, esta respuesta se la conoce como Backscatter, son las más utilizadas en la tecnología RFID, por su costo, menor tamaño y su vida útil más extensa.

1.2.4.2 Antenas RFID

Las antenas utilizadas son generalmente hechas de aluminio, cobre u otros materiales, son creadas mediante técnicas de disposición de materiales, parecidos a los que se utilizan al inyectar tinta en una hoja, esta variable hará que la etiqueta posea mayor o menor sensibilidad, determinando el rango de lectura (Telectrónica, 2010, p: 10).

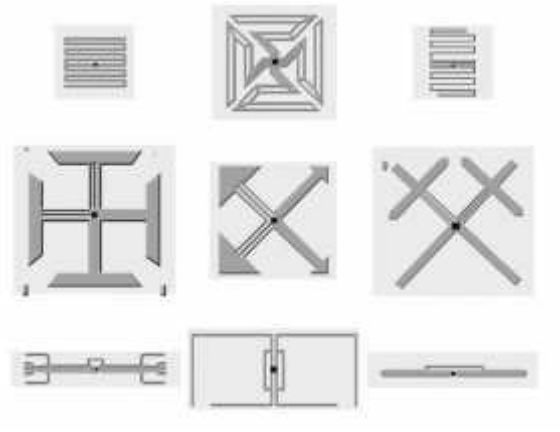


Figura 18-1: Diseños de Antenas
Fuente: (Telectrónica, 2010)

1.2.4.3 Lectores RFID

El lector es el encargo de enviar la información digital codificada por medio de su antena, utilizando las ondas electromagnéticas del espectro radioeléctrico, estas ondas son receptadas por la etiqueta, decodificando la información y enviando una señal de acuse de recibo.

Los lectores deben ser capaces de administrar las respuestas dadas por las etiquetas (una o múltiples) al mismo tiempo.

1.2.5 Estándares RFID

1.2.5.1 Alta Frecuencia (HF)

Tabla 1- 1: Descripción de los estándares RFID de alta frecuencia (HF).

Estándar	Descripción
ISO 14443 (Tipo A y B)	Provee seguridad (criptografía) y privacidad (poco rango de lectura). Este estándar es adecuado debido a que las etiquetas RFID son incorporadas en las tarjetas inteligentes para poder realizar varias tareas como; pagos con una alta seguridad, control de accesos y aplicaciones donde la seguridad sea una variable importante.
ISO 15693 (ISO 18000-3 Mode 1)	Se utiliza en sistemas que posean bajo costo de fabricación, requieran una capacidad de memoria aceptable y funcione en sistemas que presenten líquidos y metales, además que su rango de lectura es regular (1cm a 1m). Por tal motivo se lo utiliza en control de acceso, control de medicinas y ubicación de pacientes dentro de un hospital.
ISO 18000-3 (Mode 2)	Este estándar no es muy ocupado, pero posee grandes aspectos positivos como una gran velocidad de datos, funciona muy bien con líquidos y metales. Es una buena opción para chips de casino y cartas de juegos, joyería y aplicaciones que tengan item por nivel.

Fuente: (Loft Media Publishing, RFID TAG YEARBOOK, 2010)

Realizado por: Albuja J., Yépez J, 2016

1.2.5.2 Estándares UHF

Tabla 2-1: Descripción de los estándares RFID de ultra alta frecuencia (UHF).

Estándar	Descripción
ISO (18000-6C)	Es el estándar más utilizado, conocido como EPC Class 1 Gen 2, fue creado por la organización EPCGlobal y optada por la ISO en el año 2006. Este estándar fue desarrollado para conocer el flujo de mercancía entre empresas con un rango de lectura con alta cantidad de etiquetas. Presenta 4 memorias (reservada, EPC, TID, memoria de usuario.) Funciona en la banda de 860-950 Mhz, aunque varía dependiendo de las bandas ocupadas en la siguientes regiones: - Europa, India, África, Medio Este (865-868 Mhz) (ETSI) - US- Sudamérica y algunas regiones de Asia (902-928 MHz) (FCC) - Japón (950-956 Mhz) (JPN) [1]

Fuente: (Loft Media Publishing, RFID TAG YEARBOOK, 2010)

Realizado por: Albuja J., Yépez J

1.2.6 Aplicaciones RFID

- Control de animales.
- Control de Accesos.
- Aplicaciones que presenten líquidos y metales e integración de llaves para autos.
- Optimización de procesos de Industrias.
- Optimizar la industria textil.
- Entre otras.

1.2.7 Ventajas RFID

- No requiere de visión directa para identificar el objeto o artículo.
- Su tiempo de vida es elevado y se puede realizar los tags.
- Las etiquetas tienen la capacidad de lectura y/o escritura.
- Alto rango de lectura.
- Identificación de artículos simultáneamente.
- RFID garantiza que los datos obtenidos son fiables.

1.2.8 Desventajas RFID

- Problemas con líquidos y metales.
- Precio de la tecnología.
- Una misma tarjeta RFID o tags no puede responder correctamente a más de un lector a la vez.

1.3 MODULO OPEN HARDWARE ARDUINO:

1.3.1 Definición:

La plataforma Arduino está definida bajo tres fundamentos que se integran en esta tecnología que son:

Placa de hardware libre, con un Microcontrolador reprogramable, que cuenta con pines hembra que funcionan como periféricos de E/S para el Microcontrolador, donde es posible la conexión de diversos sensores y actuadores para las distintas aplicaciones de Arduino.

Al hablar de las placas Arduino, se debe hacer una distinción en su modelo, ya que existen varias placas oficiales, con características propias en tamaño físico, número de entradas y salidas, modelo de Microcontrolador incorporado, etc. Cada modelo de Arduino a pesar de tener aplicaciones y características específicas y diferentes, los microcontroladores de todos estos modelos son de tipo AVR pertenecientes a la marca ATMEL, por lo que su programación y funcionamiento son bastante parecidos entre sí (Wheat, 2011, p: 3).

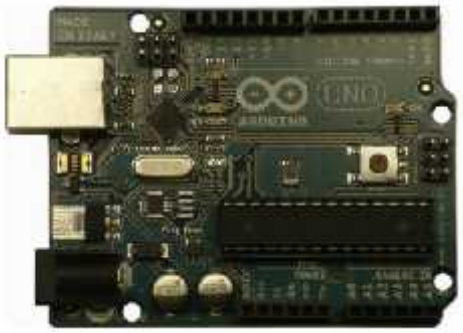


Figura 19-1: Arduino I/O Board

Fuente: Arduino Internals, 2010.

Software gratis, libre y multiplataforma, que puede funcionar en sistemas operativos Linux, MacOS y Windows, es una aplicación empleada como entorno de desarrollo que permite realizar operaciones de escritura, verificación y almacenamiento del conjunto de instrucciones que definirán su funcionamiento aplicativo, en la memoria del microcontrolador mediante conexión USB presente en la gran mayoría de modelos oficiales de placas Arduino.

El proyecto, una vez cargada la programación en el microcontrolador y dependiendo de la aplicación para la que ha sido concebido, puede ser autónomo, es decir que una vez puesto en marcha no necesita la conexión a un ordenador y su funcionamiento es automático.

Así también el proyecto puede necesitar de una conexión permanente a un computador en el que se ejecute aplicaciones específicas que permitan la comunicación entre la placa y el ordenador y el intercambio de información entre ellos.

Estas aplicaciones específicas usualmente deben ser programadas por los autores del proyecto, mediante un lenguaje de programación como c++, Java, Php, entre otros y será completamente independiente del entorno de desarrollo de Arduino.

Lenguaje de programación libre, que se entiende al leguaje artificial en el que se expresan las instrucciones para que sean ejecutadas por los microcontroladores, también llamado leguaje de máquina.

En el caso particular de Arduino se pueden encontrar las estructuras de programación tradicionales como los bloques condicionales, repetitivos, designación de variables, apoyo de comandos etc., que permiten especificar dichas instrucciones de manera exacta y precisa para su

ejecución. El lenguaje de programación (basado en Wiring) esta embebido en el entorno de desarrollo de Arduino (basado en Processing).

Arduino y Processing al ser creados en la misma institución guardan gran similitud entre sí, sin embargo la diferencia entre ellos radica en que el lenguaje Processing está basado en Java mientras que el lenguaje Arduino está basado en C y C++.

Bajo estos conceptos podemos definir Arduino como una plataforma de prototipos electrónica de código abierto, basada en hardware y software flexibles y fáciles de manejar, con capacidad de percepción del entorno mediante el acoplamiento de sensores e influenciarlo mediante actuadores, destinado para crear objetos o entornos interactivos.

1.3.2 Hardware:

Debido a la gran variedad de placas Arduino las características del hardware particulares de cada una, son distintas en capacidad de memoria, cantidad de periféricos, voltajes y frecuencias de operación, tamaño físico, entre otras, por lo que se hace necesario tomar en cuenta el modelo de la placa Arduino.

1.3.2.1 Placas Oficiales:

Las placas oficiales son aquellas manufacturadas por la compañía italiana Smart Projects incluyendo algunas diseñadas por las empresas estadounidenses SparkFun Electronics y Gravitech. A continuación se detallan las características principales de algunas de algunas de las placas oficiales más utilizadas (arduino.cc, 2015):

1.3.2.2 Arduino UNO:

El arduino UNO es una placa electrónica basada en el ATmega328, que cuenta con 14 pines digitales de entrada, de los cuales 6 se pueden usar como salidas PWM, 6 entradas analógicas, un oscilador cerámico de 16 MHz, conexión USB, un conector de alimentación, cabecera ICSP y un botón de reset, emplea 40 mA de corriente continua en sus pines I/O y 50 mA de CC en los pines a 3.3V, 32 KB de memoria Flash de los cuales 0,5 KB se reserva para bootloader, 2 KB de SRAM y 1 KB de EEPROM.

Es la placa más extendida además de ser la primera en ser comercializada en el mercado, por lo que la mayoría de variantes en los modelos que existen en la familia Arduino están basadas o tienen características similares a las del Arduino UNO.

Figura 20-1: Arduino UNO Rev3

Fuente: www.arduino.cc.

1.3.2.3 Arduino Pro:

Es una placa microcontrolada basada en el ATmega168 o ATmega 328, que viene en las versiones de 3.3V a 8 Mhz y 5V a 16 MHz respectivamente con una distribución de pines similar al Arduino UNO. Este modelo de placa fue diseñada por SparkFun Electronics con una capacidad de memoria flash entre 32 KB y 16 KB según el microcontrolador en el que se base y con 2 KB para el bootloader, tiene 1 KB en el caso del ATmega168 y 2 KB para el ATmega328 de memoria SRAM.

Figura 21-1: Arduino Board Pro

Fuente: www.arduino.cc.

1.3.2.4 Arduino Pro Minni:

El Arduino Pro Mini es una placa microcontrolada basada en el ATmega328, sus características son muy parecidas a las del modelo Pro, con la notable diferencia del tamaño, no posee conectores para los pines ni puerto USB. El sexto pin de cabecera puede ser conectado a un cable FTDI o a una placa desbloqueada SparkFun para proporcionar comunicación y alimentación USB. Esta placa tiene dos versiones, una a 3.3V y 8 MHz y otra a 5V y 16 MHz.

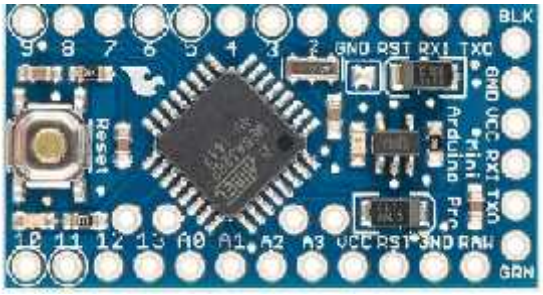


Figura 22-1: Arduino Pro Mini Board

Fuente: www.arduino.cc.

1.3.2.5 Arduino Zero:

Esta placa electrónica es un caso especial dentro de los diferentes modelos de la familia Arduino, ya que está basado en el microcontrolador Atmel SAMD21 MCU a 48 MHz con un Core ARM Cortex M0 de 32 bits, 256 KB de memoria flash, una SRAM de 32 KB y 16 KB de memoria EEPROM, voltaje de operación de 3.3V o 5V a 7 mA, 14 pines digitales de entrada y salida, de los cuales 12 son PWM y UART, de estos 6 son entradas para un canal ADC de 12 bits y una salida analógica para DAC de 10 bits.

Figura 23-1: Arduino Zero

Fuente: www.arduino.cc.

1.3.2.6 Arduino Yún:

Basada en el microcontrolador ATmega 32u4 cuenta con un chip Atheros AR9331 que está encargado de controlar el host USB, el puerto para micro-SD y la red Wifi y Ethernet. El procesador Atheros tiene una base OpenWrt-Yun que es una distribución de Linux especializada para el Arduino Yún, con capacidades similares a las de Arduino UNO con la particularidad de estar orientada al manejo de redes Ethernet, Wifi, conexiones USB y almacenamiento micro-SD.

Posee 20 pines digitales, 7 pines para WM, 12 pines analógicos, 16 MHz y 5V en el caso del ATmega 32u4 con una memoria 32 KB y 4 KB reservados para el bootloader, 2,5 KB de SRAM y 1 KB de memoria EEPROM. Para el caso del AR9331 trabaja a una frecuencia de 400 MHz basado en MIPS con un voltaje de 3.3V, con memoria RAM DDR2 de 64 MB y una memoria flash de 16 MB flash para el sistema Linux embebido.

Figura 24-1: Arduino Yún
Fuente: www.arduino.cc.

1.3.2.7 Arduino LilyPad:

Una clase de Arduino diseñado para ser integrada en prendas y textiles desarrollado por Leah Buechley y SparkFun Electronics que compensa ciertas limitaciones de funcionamiento con su gran capacidad de integración y flexibilidad.

Está basado en los microcontroladores ATmega168V y ATmega328V con una frecuencia de operación de 8 MHz en ambos casos, 2,7v y 5,5v de voltaje de operación respectivamente, 14 pines digitales, con 6 como PWM y 6 pines analógicos. Cuenta con 6 KB de memoria flash para el código de programa, una SRAM de 1 KB y 512 Bytes de memoria EEPROM.

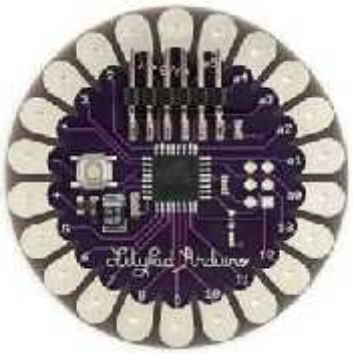


Figura 25-1: Arduino Lily Pad

Fuente: www.arduino.cc.

1.3.2.8 Arduino Mega 2560:

Es una placa microcontrolada basada en el ATmega2560 con una frecuencia de funcionamiento de 16 MHz y trabaja a 5v y 8 bits. Cuenta con una SRAM de 8 KB, una memoria EEPROM de 4 KB 256 KB de flash con 8KB reservados para el bootloader, basado en arquitectura AVR con 54 pines digitales, de los cuales 15 pueden ser usados como PWM y 16 pines analógicos.

Ideado como un Arduino con mayor número de periféricos y mayor potencia que el Arduino UNO sin que el rendimiento haga necesario emplear los Arduino basados en una arquitectura ARM.

Figura 26-1: Arduino Mega 2560

Fuente: www.arduino.cc.

1.3.3 OPEN SOURCE SOFTWARE ARDUINO:

El software de Arduino está basado en una gran variedad de paquetes de software libre y código abierto, licenciado bajo la versión 2 de GPL, sin embargo los paquetes que lo componen esta licenciados bajo un sin número de concepciones diferentes.

Una de las principales ventajas del software libre y del código abierto, además de su bajo costo, es el acceso al código fuente actual y a sus tecnologías subyacentes, lo que hace posible modificarlo de la manera en la que el usuario crea conveniente o necesite, abriendo un sinfín de posibilidades de nuevas aplicaciones, mejoramiento del rendimiento del software, depuración de errores en el código, implementación de nuevas características o ajustes a las existentes, etc., lo que brinda un desarrollo continuo en las potencialidades del proyecto Arduino contribuyendo al éxito que ha llegado a ser alrededor del mundo (Arduino Internals , 2010, p: 90).

1.3.3.1 El Entorno de Desarrollo Arduino:

El entorno de desarrollo contiene todas las herramientas necesarias para escribir los distintos programas que sean necesarios, compilarlos usando un compilador del estado del arte y cargarlo a una variedad de placas compatibles con la familia Arduino, además de contar con una gran colección de ejemplos tutoriales que ayudarán en la iniciación y fundamentos de la programación Arduino a los nuevos usuarios y desarrolladores.

Figura 27-1: Entorno de Desarrollo Arduino

Realizado por: Albuja J., Yépez J, 2016.

La estructura básica de la interfaz de usuario consta de una sola pantalla que cuenta con las tradicionales barras de menú y herramientas como se muestra en la figura anterior, seguido de

un área para la estructura del programa y otra en el fondo de la pantalla para mostrar estados del programa.

1.3.3.2 Barra de Menús:

Menú Archivo:

Aquí se encuentran las herramientas para trabajar con los archivos generados por los programas almacenados en el computador, donde se puede encontrar las opciones usuales de manejo de archivos como: Nuevo, Abrir, Abrir Reciente, Proyecto, Cerrar, Guardar, Guardar Como e Imprimir, la Configuración de Página permite escoger las opciones básicas de la composición de la página para su impresión, incluyendo el tamaño de los márgenes, la orientación de la página y el formato de la misma.

En este menú tenemos accesos directos a los proyectos que se almacenen en el ordenador y a varios ejemplos explicativos mediante las opciones Proyectos y Ejemplos respectivamente. Además se tiene las opciones de Preferencias y Salir, en la primera se encuentran las opciones de configuración general del programa como el directorio donde se almacenarán los proyectos del usuario, el idioma entre otras, las cuales se almacenan en el archivo preferences.txt (Arduino Internals, 2010, p: 94).

Menú Editar:

En este menú se encuentran las opciones Deshacer, Rehacer, Cortar, Pegar y Encontrar, típicas de esta clase de menú, adicionalmente cuenta con Copiar desde Foro y Copiar como HML que inserta los comandos en el formato apropiado, conservando tanto el formato como el énfasis del texto proporcionado por el editor.

También existen opciones de edición del propio programador, incluyendo Comentar y Descomentar que permite ocultar secciones grandes del código como comentarios, anteponiendo la doble barra inclinada en cada línea del código seleccionado.

Al igual que en las aplicaciones de Windows, el menú editar implementa teclas de acceso rápido para las opciones de Cortar (Ctrl+X), Copiar (Ctrl+C) y Pegar (Ctrl+V), y su menú desplegable

al hacer click derecho en el panel de edición incluyendo la opción de hacer una búsqueda web de la palabra clave seleccionada.

Menú Programa:

En este menú se pueden encontrar opciones del programador como: Verificar/Compilar que cumple con las funciones que describe su nombre, Subir y Subir Usando Programador que carga el programa en la placa Arduino con la diferencia de que la primera opción busca el dispositivo conectado de manera global mientras que la segunda opción busca la placa en un puerto USB determinado.

En el menú programa tenemos la posibilidad guardar el código compilado en binario con extensión de archivo .HEX mediante la opción de Exportar Binario Compilado. La opción Mostrar Carpeta de Programa ejecuta el explorador del sistema de archivos dependiente del sistema operativo que muestra el contenido del directorio de trabajo donde los archivos generados por el entorno de desarrollo son almacenados.

La opción Añadir Archivo permite copiar otro archivo en el programa y abrirlo en una nueva pestaña dentro del editor, mientras que la opción de Importar Librería despliega un submenú que muestra una gran variedad de librerías con el apropiado formato #include dentro del código del programa, siendo posible incluir librerías creadas por el propio programador (Arduino Internals, 2010, p: 95).

Menú Herramientas:

Aquí se encuentran algunas herramientas específicas de Arduino y el acceso a sus respectivas configuraciones. El Auto Formato se emplea para limpiar el formato del código, manteniendo una sangría consistente y alineando las llaves siempre que sea posible.

La opción Archivo de Programa ofrece la posibilidad de reunir todos los archivos en el directorio de trabajo dentro de un archivo de almacenamiento apropiado para el sistema operativo, bastante práctico al momento de compartir los programas con otros desarrolladores.

El Monitor Serial es una herramienta muy útil para la comunicación serial con la placa Arduino, usando el mismo puerto por el que se cargan los programas, el puerto es seleccionado en la

opción de las herramientas con el mismo nombre, en el que se listarán todos los disponibles en el ordenador donde esté instalado el software Arduino.

La Selección de la Placa permite especificar el modelo de Arduino con el que se trabajará, para que el software Arduino tome las decisiones correctas sobre como compilar el programa y cargarlo en la placa. La opción de Quemar Bootloader es usado para configurar apropiadamente un chip AVR en blanco para que sea capaz de usar el proceso del bootloader Arduino de carga del programa sobre el puerto seleccionado.

Menú Ayuda:

En este menú está disponible una gran cantidad de información acerca del uso de Aduino, contiene los links de referencia para consulta. La página oficial de Arduino http://arduino.cc está disponible en el menú para más fuentes de ayuda.

Barra de Herramientas:

En la barra de herramientas se encuentran los accesos rápidos para las funciones de Verificar/Compilar, Subir, Nuevo, Abrir y Guardar.

1.3.4 Estructura Básica de Programa:

La estructura de la programación Arduino se estructura de al menos dos partes o funciones, en las que se alojan las declaraciones, estamentos o instrucciones que definen el programa.

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

Figura 28-1: Estructura Básica del Programa

Realizado por: Albuja J., Yépez J, 2016.

Donde setup() recoge la configuración inicial donde se realiza la declaración de las variables, inicializar los modos de trabajo de los pins o el puerto serie, estas configuraciones se ejecutan solo una vez hasta el próximo reinicio de la placa.

La Función loop() o bucle, es la parte que contiene los comandos que se ejecutarán de forma cíclica, es decir las instrucciones que deben ejecutarse permanentemente como lectura de entradas y activación de salidas, permitiendo a la placa responder a las tareas para las que fue programada. Ambas partes son necesarias para que el programa pueda ser ejecutado con normalidad por la plataforma Arduino.

1.3.4.1 Funciones:

Una función se define como un bloque de código identificado por un nombre y que contiene un grupo de instrucciones que serán ejecutadas cuando la función sea invocada, las mismas que pueden ser creadas para realizar tareas repetitivas y reducir el tamaño de un programa

.
Estas funciones se declaran asociadas a un tipo de valor "type" que devuelve la función después de cumplir su proceso. Después del tipo de valor se escribe el nombre de la función y entre paréntesis de ser necesario se escribirán los parámetros con los que la función deberá trabajar.

Las funciones se delimitan usando el símbolo llaves, que definen el comienzo y el final de la función, y entre las que se escribirán las instrucciones que se ejecutarán cuando la función sea llamada. Cada instrucción se separa o delimita usando el ";" al final de la misma.

```
6  int delay()
7  {
8    int v;                  //Crea una variable temporal v
9    v=analogRead(pot);      //Lee el valor del potenciómetro
10   v/=4;                    //Convierte 0-1023 a 0-255
11   return v;               //Devuelve el valor final de v
12 }
```

Figura 29-1: Programa para leer el valor de un potenciómetro

Realizado por: Albuja J., Yépez J,2016.

En el ejemplo de la figura anterior el tipo de dato es "int", el nombre de la función es delay(), sin especificar parámetros de entrada. Esta función tiene como tarea leer el valor analógico de un potenciómetro comprendido entre 0 a 1023, para lo que se crea una variable temporal v que almacenará el valor del potenciómetro, posteriormente se divide entre 4 para adaptarlo a una escala de 0 a 255 y devuelve el valor de v al programa principal.

1.3.4.2 Instrucciones:

Una instrucción puede comprender un comando o conjunto de comandos, en la misma línea de código, que son ejecutados en un orden progresivo determinado por su posición dentro del programa o por funciones u estructuras que invoquen la ejecución de ciertas instrucciones. Dentro de una instrucción se puede emplear funciones especiales con operaciones específicas.

1.3.4.3 Funciones Especiales:

Funciones E/S:

pinMode(pin, mode).- Se emplea en el setup() para definir el modo de trabajo de un Pin, pudiendo ser entrada (INPUT) o salida (OUTPUT), teniendo en menta la particularidad de que los pines en las placas Arduino, por defecto, están configuradas como entradas, en estado de alta impedancia.

digitalRead(pin).- Lee el valor digital de un pin, definido como tal y declarado como variable o constante, dando como resultado un HIGH o LOW.

digitalWrite(pin, value).- Esta función envía a un pin determinado como salida y declarado como variable o constante, un valor HIGH o LOW, asignando un 1 o 0 lógico a la salida respectivamente.

analogRead(pin).- Esta función permite leer valores analógicos de un pin definido como entrada analógica con una resolución de muestreo de 10 bits, válida solo para los pines del 0 al 5 y con un rango de valores entre 0 a 1023.

analogWrite(pin, value).- Esta función se emplea para escribir un pseudo-valor analógico utilizando el procesamiento de modulación por ancho de pulso (PWM) a uno de los pines destinados para el PWM, cuyo valor puede darse de forma constante o variable manteniendo un marguen entre 0 y 255, con valores de voltaje entre 0 y 5V.

Funciones de Tiempo:

delay(ms).- Permite detener la ejecución en proceso del programa, una cantidad de tiempo determinada en milisegundos (ms), definida en la misma instrucción.

millis().- Permite determinar la cantidad de tiempo transcurrido en milisegundos (ms) desde el inicio del programa en la placa hasta el momento actual, el contador se reinicia con cada reinicio de la placa Arduino.

Funciones de Cálculo:

min(x, y); max(x, y).- Calcula el valor mínimo y máximo de dos números respectivamente.

randomSeed(seed).- Con esta instrucción se establece un valor de partida para la función

random(), ya que en principio la placa no puede producir valores aleatorios propiamente dichos, es por esto que al colocar una variable, constante u otra función de control dentro de la función random, permitirá crear dichos valores aleatorios. Dentro de las funciones de control se pueden emplear cualquiera de las funciones anteriormente citadas.

Funciones Puerto Serie:

Serial.begin(rate).- Habilita el puerto serie y establece la velocidad de transmisión de datos en baudios, siendo el valor típico de 9600 baudios.

Serial.println(data).- Emplea el puerto serie para recrear los datos enviados a este, seguido por un retorno automático y salto de línea, que permite una fácil lectura en el monitor serie del software Arduino.

Seria.print(data, data type).- Vuelca un número o cadena de caracteres en el puerteo serie, el mismo que dependiendo de los parámetros que se emplee para definir el formato de envío de datos en el puerto, se presenta de distintas formas. Estos parámetros son:

data: número o cadena de caracteres enviados.

data type: formato de salida de los valores numéricos, que pueden ser binario (BIN), hexadecimal (HEX), octal (OCT), etc.

Serial.available().- Permite obtener el espacio de caracteres disponible para leer en el puerto serie definido en bytes. El buffer serie puede almacenar como máximo 64 bytes de información.

Serial.Read().- Equivale a la función serialRead(), la misma que permite la captura de un byte en el puerto serie.

1.3.4.4 Variables:

Las variables son un método para el almacenamiento de valores numéricos bajo un descriptor o nombre y un tipo de variable, para su posterior utilización en el programa, que como su nombre sugiere estos valores pueden cambiar y una vez almacenados pueden ser utilizadas para otros procesos o emplear su valor directamente.

Declaración de Variables:

Para que una variable pueda ser utilizada en la ejecución del programa, es necesario declarar la variable, para esto se define el tipo, se asigna un nombre y dependiendo del caso un valor inicial. Este proceso se realiza una sola vez por variable y cada vez que se requiera el uso de las mismas, siendo su valor modificable en cualquier momento.

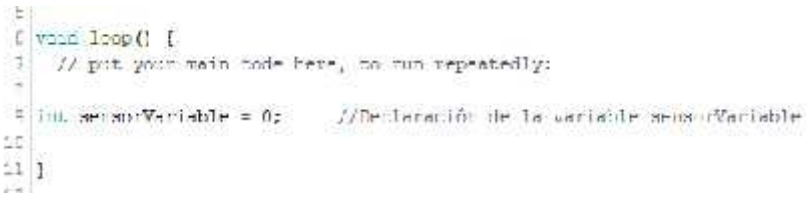


Figura 30-1: Estructura para la declaración de una variable
Realizado por: Albuja J., Yépez J, 2016.

Tipos de Variables:

Dependiendo de su tipo, una variable adquiere la capacidad de almacenar cierta clase de valores así como también se define la capacidad de almacenamiento de la misma.

Byte.- Puede almacenar valores numéricos de 8 bits sin decimales, con un rango de valores de entre 0 a 255.

Int.- Define un tipo de dato entero de 16 bits sin decimales, con un rango de -32768 a 32767.

Long.- Formato de variable de tipo extendido, almacena números enteros de 32 bits sin decimales, en un rango de -2147483648 a 2147483647.

Float.- Formato de variable de tipo flotante, puede almacenar número de 32 bits con decimales, en un rango de 3.4028235E+38 a -3.4028235E+38.

1.3.4.5 Constantes:

Como su nombre lo dice son valores específicos que no cambiarán en ningún punto de la ejecución del programa. Dentro del Lenguaje de programación propio de Arduino existen comandos predeterminados, o palabras reservadas, para establecer constantes dentro del código y hacerlo más legible.

Entre las palabras reservadas para constantes se tiene:

True/False.- Constantes booleanas empleadas para definir los niveles lógicos HIGH y LOW en las salidas digitales, respectivamente. TRUE se asocia a un 1 o cualquier valor diferente de 0, mientras que FALSE a un 0.

High/Low.- Definen los niveles de salida Altos y Bajos en la lectura y escritura digital de los pines, asociados a un 1 lógico y 0 lógico respectivamente. Un 1 lógico representa al estado “on” o 5 vóltios, mientras que un 0 lógico representa un “off” o 0 vóltios.

Input/Output.- Constantes empleadas para definir el modo de operación de los pines de la placa Arduino que se asocian con los modos de entrada y salida respectivamente.

1.3.5 Estándar IEEE 802.15.4 – ZigBee

1.3.6 Descripción General

ZigBee es una tecnología inalámbrica utilizada debido a las grandes prestaciones que nos brinda, flexibilidad de red, costos bajos, bajo consumo de energía, siendo implementada para aplicaciones de hogar o industria que requieran seguridad y/o automatización (Archundia, 2003, p: 1).

Es una tecnología creada en el año 2000 por dos grupos de estándares (ZigBee y el grupo 15 de trabajo IEEE 802), para satisfacer la necesidad de un estándar inalámbrico de bajo poder y por ende de bajos costos que se pueda utilizar tanto en ambientes industriales y caseros.

ZigBee funciona en la banda ISM del espectro radioeléctrico, para uso industrial, científico y médicos, utiliza la frecuencia de 868 MHz para Europa, 915 para Estados Unidos y 2.4 GHz en el resto del mundo. La última al ser una banda libre a nivel global se la puede utilizar sin restricción para crear aplicaciones poseyendo 16 canales de 5 MHz.

1.3.7 Características

El estándar IEEE 802.15.4 es la base sobre la cual la especificación ZigBee se define, el estándar nos provee los niveles básicos de red para dar servicio a redes inalámbricas de área personal (WPAN) enfatizando la comunicación entre dispositivos a bajo coste y velocidad.

Se prevé en condiciones básicas una comunicación de 10 metros con una velocidad de transmisión de datos de 250 Kbps, aunque la característica fundamental es la posibilidad de tener costes de fabricación bajos por la sencillez tecnológica (Archundia, 2003, p: 3).

En la siguiente tabla mostraremos las características esenciales que presenta dicho estándar:

Tabla 3-1: Características del estándar IEEE 802.15.4

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

Fuente: Archundia, 2008, (Wireless Personal Network (WPAN) & Home Networking)

Realizado por: Albuja J., Yépez J. 2016

1.3.8 Arquitectura de Protocolos

El estándar IEEE 802.15.4 define los niveles inferiores del Modelo OSI (Figura 31-1), siendo el nivel físico y enlace de datos, además nos prevé la interacción con los demás niveles utilizando un subnivel denominado Control de Enlace Lógico que se fundamenta en el estándar IEEE 802.2 (LLL, Logical Link Control), para acceder al MAC (Control de Acceso al Medio) mediante su subnivel de convergencia (Barneda, 2008, p: 15).

Figura 31-1: Pila de Protocolos del Modelo OSI
Fuente: (Barneda, 2008).

1.3.8.1 Nivel Físico (PHY)

Este nivel nos facilita el servicio de trasmisión de datos sobre el medio físico, además de proporcionarnos la interfaz utilizada para su configuración (gestión) y mantener la información de la base de datos con redes de área personal, por ende el nivel PHY manipula el transceptor de radiofrecuencia, seleccionando los canales, además controla el consumo tanto de energía como de la señal.

En su versión original el estándar IEEE 802.15.14 contenía dos niveles físicos basados en modulación DSS (Espectro Ensanchado por Secuencia Directa). Siendo el primero utilizado en la frecuencia de 868/915 MHz con un velocidad de 20 y 40kbps, mientras que la segunda ocupa daba la banda de 2450Mhz con una velocidad de 250 Kbps.

En la actualidad dicho estándar ha conseguido poseer 4 niveles físicos distintos, los 3 primeros siguen ocupando la modulación DSS para la banda de 868-915 MHz y el último nivel realiza

una modulación OQPSK (Modulación en fase binaria o por cuadratura en offset) en la frecuencia de 2450 MHz (Barneda, 2008, p: 15-16).

1.3.8.2 Nivel de Enlace de Datos

Este nivel es el encargado de comunicarnos con los niveles superiores de la pila de protocolos (Figura 32-1) mediante el Control de Enlace Lógico o LLC, además permite la utilización de tramas MAC (Control de Acceso al medio) ofreciéndonos una interfaz capaz de controlar y regular el acceso hacia el canal físico y a balizado de red, obteniendo puntos de enganche para servicios seguros. En la siguiente Figura 1-36 observaremos un esquema de cómo funciona el estándar 802.15.4 para establecer comunicación con los niveles superiores (Barneda, 2008, p: 16-17).

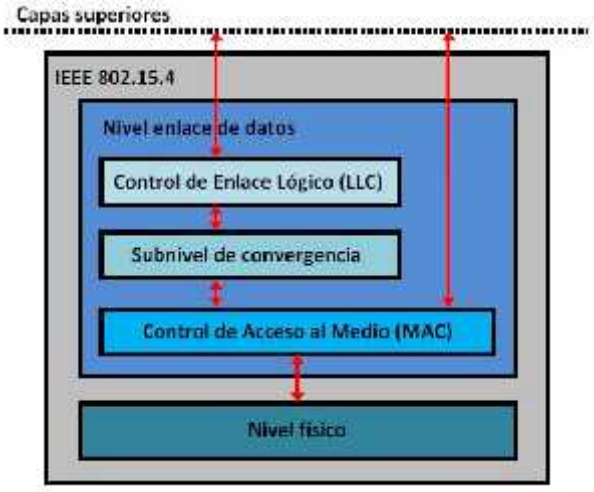


Figura 32-1: Pila de Protocolos IEEE 802.15.4

Fuente: (Barneda, 2008).

1.3.8.3 Nivel de Red:

Este nivel nos facilita el uso correcto del subnivel MAC, además de proveer una interfaz para interactuar con el nivel de aplicación, funciona típicamente como un nivel de red clásico, por ende es el encargado de encaminar los paquetes y entregarlos.

Entre sus funciones de control puede encargarse de la configuración de nuevos dispositivos y el establecimiento con nuevas redes pudiendo ser nuevos Routers y vecinos, logrando comunicación directa y sincronización a nivel MAC.

1.3.8.4 Nivel de aplicación:

Es el nivel más alto definido en la especificación, en él se encuentran todos los componentes que nos permitan la interacción efectiva entre el nodo ZigBee y os usuarios.

Entre sus componentes principales se encuentran:

- El ZDO: Es el dispositivo Coordinador de ZigBee que identifica a los dispositivos que se encuentran como vecinos y los servicios a ofrecer, estableciendo enlaces seguros con dispositivos externos y responder peticiones.

- El subnivel de soporte a la aplicación (APS): Se encarga de interconectar el nivel de red y resto de componentes del nivel de aplicación, encaminando los mensajes a toda la pila de protocolos.

1.3.9 Dispositivos ZigBee

Una red ZigBee está compuesta por tres dispositivos ZigBee, cada una con sus funciones establecidas dentro de nuestra red siendo los siguientes (Barneda, 2008, p: 18):

Coordinador ZigBee (ZC): Es el encargado de controlar y coordinar la red, además de establecer los caminos necesarios para la interconexión entre dispositivos, se debe tener en cuenta que cada red debe tener su propio ZC.

Router ZigBee (ZR): Interconecta los dispositivos que se encuentran en distintas topologías de red.

Dispositivo Final (ZED): Estos son los dispositivos que representan la característica de bajo consumo y bajo coste que provee ZigBee, debido a que los ZED solo pueden comunicarse con su nodo padre, sea el ZR o ZC, por tal motivo al no interactuar con otros dispositivos pueden entrar en un estado de dormido, aumentando la durabilidad de las baterías, además no requiere de altas capacidades de memoria por lo cual su coste es menor.

1.3.10 Funcionamiento de ZigBee

Las redes ZigBee pueden funcionar de dos modos:

1.3.10.1 Con Balizas:

En este tipo de sistema la red ZigBee presenta un distribuidor el cual se encarga de sincronizar los dispositivos que se encuentran en la red, además de controlar el medio (canal) de las transmisiones.

Para realizar la sincronización se ocupa elementos llamados balizas que requieren del ZC para los intervalos de funcionamiento que pueden variar entre 15 ms a 4 minutos. Este modo se recomienza cuando el coordinador de red trabaja con una batería.

Este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red escuchan a dicho coordinador durante el balizamiento (envío de mensajes a todos los dispositivos -broadcast-, entre 0.015 y 252 segundos). Un dispositivo que quiere intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y es entonces cuando mira si hay mensajes para él. En el caso de que no haya mensajes, este dispositivo vuelve a "dormir", y se despierta de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el balizamiento, todos los dispositivos de la red vuelven a "dormirse" (Barneda, 2008, p: 21).

Como vemos, se trata de un mecanismo de control del consumo de potencia en la red.

1.3.10.2 Sin Balizas:

En este modo de funcionamiento intervienen dispositivos autónomos (están activos con fuente de alimentación propia), por tal motivo pueden interactuar con los otros dispositivos sin tener en cuenta que los restantes elementos pueden estar ocupando el mismo receptor o el mismo canal, ocasionando colisiones, por tal motivo al implementar una red sin balizas el control de acceso al medio se lo realiza por CSMA/CA (acceso múltiple con escucha de portadora y evasión de colisiones) (Barneda, 2008, p: 22-23).

CSMA/CA: Este tipo de acceso funciona de la siguiente manera, el nodo que pretende trasmitir escucha el canal para detectar si otro dispositivo está transmitiendo o desea hacerlo, aunque este modo no funciona en comunicaciones inalámbricas por tal motivo para evitar que existan colisiones la estación que quiere trasmitir escucha y verifica que el canal esta libre y espera cierta tiempo para ello, en cambio al estar el canal con trafico deberá esperar un tiempo extra más otro aleatorio para intentar nuevamente trasmitir.

Este modo es utilizado para sistemas de seguridad, debido a que los dispositivos utilizados permanecen en estado pasivo (dormido), pero se identifican con la red en forma regular, para tenerlos en cuenta, aunque al producir un evento los dispositivos pasa a estado activo instantáneamente y trasmiten su información al dispositivo correspondiente.

1.3.11 Seguridad

Para la seguridad ZigBee utiliza claves de 128 bits, esta clave puede ser utilizada en varias etapas como, asociarse a una red para utilizarlo con los niveles ZigBee y subnivel MAC, aunque se tiene en consideración que la clave inicial maestra se debe obtener por medios seguros, ya que la red depende de ella, por tal motivo esta clave es precargada por fábrica dentro del dispositivo (caso ideal) o se designa a un dispositivo especial para dicha tarea (centro de confianza o trust center) cuando la seguridad no sea un campo crucial en la red (Barneda, 2008, p: 24-25).

Para controlar la seguridad los distintos niveles funcionan:

- El subnivel MAC va a realizar comunicaciones que sean de tipo fiable y se pueda llegar en un solo salto, esto lo logra ya que funciona como un cliente que acoge la seguridad enviada desde las capas superiores.

- El nivel de red provee de routeo, administra los mensajes recibidos.

- El nivel de aplicación provee de claves al ZDO y difunde los cambios que ocurren a los dispositivos que se encuentren en red, además funciona como intermediario entre las peticiones dirigidas al centro de seguridad y propaga si existe renovaciones en la clave.

1.3.12 Redes ZigBee

1.3.12.1 Topología en estrella

En este tipo de topología el controlador o coordinador (Router, switch, hub) se sitúa en el centro de la red, donde se conectan todos los nodos; este esquema se lo utiliza para LAN locales debido a la facilidad para su implementación, aunque presentas varias dificultades como la cantidad de nodos que puede soportar el coordinador o la longitud del cableado, además que al fallar el nodo central la red colapsaría.

Debido a lo anterior una red en estrella no es muy fiable para transferencias de información.

Figura 33-1: Red en estrella

Fuente: http://alyluna.weebly.com/

1.3.12.2 Topología en árbol

Se lo ve como redes en estrella interconectas entre sí mediante un nodo de enlace troncal (hub o switch), desde el cual se conectan a los restantes nodos, la comunicación en este tipo de topología es jerárquica, no obstante al existir un fallo, este evento no implica que la comunicación se interrumpirá.

Se debe tomar en cuenta que los datos que se envían son recibidos por toda la red, siendo una desventaja ya que necesitamos de un mecanismo (identificador de estación destino) para llegar a su destino. Otro de los problemas de esta red radica en que las señales pueden interferirse al trasmitir al mismo tiempo requiriendo información de control en las tramas (se puede utilizar balizas) para su envió.

Figura 34-1: Topología en árbol

Fuente: http://www.monografias.com/

1.3.12.3 Topología en malla

La principal característica de este tipo de topología es que uno o varios de sus nodos poseen más de dos conexiones para poseer redundancia en la red (red confiable), por ende los nodos son independientes, logrando que si un nodo falla la red no quede inutilizable. En esta topología se puede llevar la información por diversos caminos (red autogenerables) sin la necesidad de optar por un sistema con balizas.

Otros de los puntos importantes es destacar en este topología de red la solución de problemas y fiabilidad mejoran, pero decaen al poseer un grado más alto es su instalación ya que requiere en enlaces redundantes.

Figura 35-1: Topología en malla

Fuente: http://bp0.blogger.com/

1.3.13 ZigBee VS Bluetooth

Tabla 2-1: Comparativa entra las tecnologías ZigBee y Bluetooth

ZigBee	**Bluetooth**
Su red puede constar con más de 65000 nodos con subredes de 255 nodos	Posee 8 nodos por subred
Bajo consumo eléctrico (30 mA en trasmisión y 3 uA en reposo)	Mayor consumo eléctrico, siempre está trasmitiendo y/o recibiendo información, (40 mA en trasmisión y 0.2 mA en reposo.)
Velocidad de trasmisión es de 250 kbps	La velocidad de trasmisión puede llegar

	hasta 1 Mbps

Fuente: Cire (Club de Informática, robótica y Electrónica), 2012.

Realizado por: Albuja J., Yépez J, 2016.

Por ello cada tecnología tiene sus ventajas dependiendo de la aplicación en la que se la quiera ocupar, por ejemplo Bluetooth se la utiliza en circunstancias que requieren gran cantidad de información como teléfonos móviles, en cambio ZigBee se lo utiliza en aplicaciones como domótica donde el consumo de energía debe ser bajo y la cantidad de información es precisa y no extensa.

1.4 Sistema Gestor de Base de Datos (SGBD)

Básicamente un gestor de Base de Datos consta de dos partes fundamentales, la primera una recopilación de datos denominada Base de Datos que contiene la información de interés de la empresa (banca, líneas áreas, universidades, telecomunicaciones, etc.) y una recopilación de programas para acceder a los datos, para obtener un SGBD que proporcione la información de la base de datos de manera práctica y eficiente (Silberscatz, 2002, p: 1).

Para poder establecer el modelo de datos tomaremos en cuenta los siguientes modelos para ver sus características.

1.4.1 Modelo entidad-relación

Este modelo se basa en tomar percepciones que están ligados con la realidad, el mismo que se fundamenta en dos conceptos claves, el primero utiliza objetos básicos llamados entidades que son descritos o calificados dependiendo de sus atributos (número_cuenta, saldo, nombre_cliente, ciudad_cliente, etc) y de relaciones entre estos objetos que se puede ser entre una o varias entidades, cada uno al unirse entre ellas conformaban conjunto de entidades y conjunto de relaciones.

La estructura básica (Figura 36-1) de este modelo se fundamenta en la utilización de rectángulos que representan conjuntos de entidades, elipses que representan atributos, rombos que representan relaciones entre entidades y líneas que unen los atributos con el conjunto de entidades y el conjunto de entidades con las relaciones.

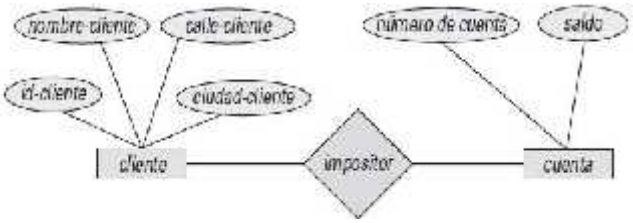


Figura 36-1: Ejemplo Diagrama E-R

Fuente: Silberscatz, 2002

1.4.2 Modelo Relacional

En este tipo de modelo se utilizan tablas en grupo para poder representar y relacionar los datos, en donde cada tabla estará conformada por muchas columnas y las columnas poseerán un nombre único, para mejor comprensión observamos la Figura 37-1.

id cliente	*nombre cliente*	*calle cliente*	*ciudad cliente*
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cercedа
67.789.901	López	Mayor	Peguerinos
18.273.609	Abril	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Rambla	León
01.928.374	Gómez	Carretas	Cercedа

(a) La tabla *cliente*

número cuenta	*saldo*
C 101	500
C-215	700
C 102	400
C-305	350
C-201	900
C 217	750
C-222	700

(b) La tabla *cuenta*

id cliente	*número cuenta*
19.283.746	C 101
19.283.746	C-201
01.928.374	C 215
67.789.901	C-102
18.273.609	C-305
32.112.312	C 217
33.666.999	C-222
01.928.374	C 201

(b) La tabla *impositor*

Figura 37-1: Ejemplo Modelo Relacional

Fuente: Silberscatz, 2002

1.4.3 Otros Modelos de Datos

Son modelos híbridos que se originan de la derivación o combinación de los modelos dichos anteriormente (Silberscatz, 2002, p: 7).

1.4.3.1 Modelo Orientado a Objetos

Primero se encuentra el modelo de datos orientado a objetos (Figura 38-1) siendo una extensión que se deriva del modelo E-R adjuntado encapsulación, utilización de métodos o funciones e

identificador de objetos, entre los programas más comunes actualmente se centran los software derivados de C++ y Java.

Figura 38-1: Modelo Orientado a Objetos

Fuente: https://grupo3pnfi.files.wordpress.com/

1.4.3.2 Modelo de datos relacional orientado a objetos

Como segundo se encuentra el modelo de datos relacional orientado a objetos que se crea a partir de la unión de las características del modelo de datos orientados a objetos y el modelo de datos relacional, siendo más complejo y obteniendo la capacidad de adjuntar datos complejos y lograr programación orientada a objetos, entre los lenguajes de consulta relacionales se encuentra SQL.

1.4.4 Arquitectura de los Sistemas de Base de Datos

La arquitectura a establecer en un sistema de base de datos está influenciada por el sistema informático con el que se va a ejecutar, debido a que debemos considerar diversos factores tales la conexión de red (sistema servidor o cliente), el paralelismo (procesamiento) y la distribución (distintas sucursales, sedes o departamentos), entre las arquitecturas principales se destallaran los sistemas centralizados y cliente-servidor (Silberscatz, 2002, p: 445).

1.4.4.1 Sistemas Centralizados

Su principal característica es que posee un sistema informático único que no requiere de la interacción con otras computadoras (Figura 39-1). Posee dos formas de usos, en un sistema monousuario o multiusuario.

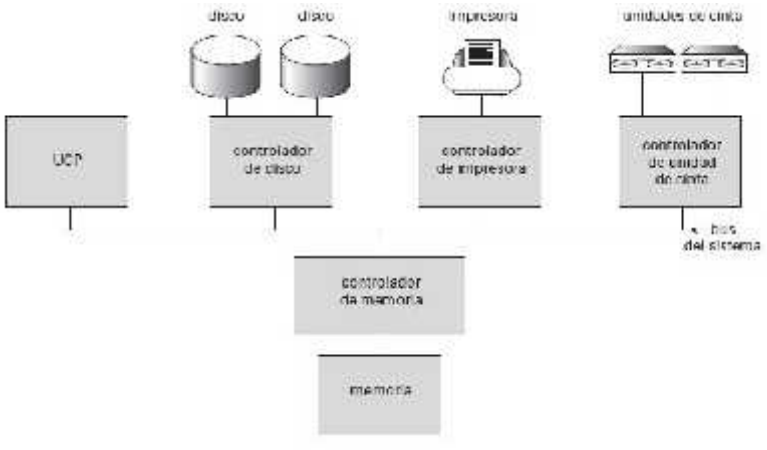


Figura 39-1: Sistema Centralizado

Fuente: Silberscatz, 2002

1.4.4.2 Sistemas Cliente-Servidor

Este sistema se basa en satisfacer las peticiones generadas por sistemas clientes, su representación se encuentra en la figura 40-1.

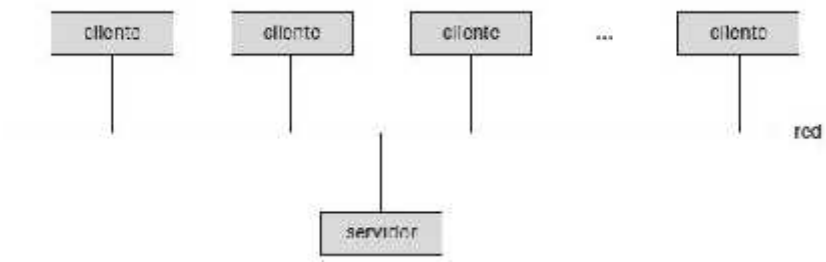


Figura 40-1: Sistema Cliente-Servidor

Fuente: Silberscatz, 2002

1.4.5 SQL

SQL es un lenguaje de programación que está orientado para la manipulación de datos en ámbito comercial ya que se requiere que las consultas sean cómodas para el usuario. Además de proporcionarnos características para definir estructuras, modificar la base de daos y establecer parámetros básicos de seguridad (Silberscatz, 2002, p: 87).

1.4.5.1 Estructura Básica

SQL nos proporciona la capacidad de utilizar valores nulos para indicar que el valor es desconocido o no existe. Una estructura básica en SQL debe poseer los siguientes elementos o clausulas (Silberscatz, 2002, p: 88):

- Cláusula Select (Figura 41-1): Realiza la operación proyección establecida en el álgebra relacional, es utilizada para listar atributos esperados del resultado en una consulta.

```
select nombre-sucursal
from préstamo
```

Figura 41-1: Ejemplo Cláusula Select

Fuente: Silberscatz, 2002

- Cláusula From (Figura 42-1): Realiza la operación producto cartesiano del álgebra relacional, es utilizada para listar las relaciones a ser analizadas en la expresión.

```
select número-préstamo
from préstamo
where nombre-sucursal = 'Navacerrada' and
      importe > 1200
```

Figura 42-1: Ejemplo Cláusula From

Fuente: Silberscatz, 2002

- Cláusula where (Figura 43-1): Realiza la operación selección del álgebra relacional, es utilizada para englobar a los atributos que se obtienen de las relaciones puestas en la cláusula from.

```
select nombre-cliente, prestatario.número-préstamo,
       importe
from prestatario, préstamo
where prestatario.número-préstamo
      = préstamo.número-préstamo
```

Figura 43-1: Ejemplo Cláusula From

Fuente: Silberscatz, 2002

1.4.5.2 Tipos de dominio en SQL

SQL es un conjunto que presenta los siguientes tipos (Silberscatz, 2002, p: 106):

- **char (n)** es una cadena de caracteres de longitud fija **n** dado por el usuario, también se lo conoce como **character**.

- **varchar (n)** es una cadena de caracteres de longitud variable **n** dado por el usuario, también se lo conoce como **character varying**.

- **int** es una variable de tipo entero, se la conoce también como **interger.**
- **float (n)** es una variable de tipo flotante, con precisión de n dígitos.
- **date** nos brinda una fecha del calendario en el formato año, mes y día del mes.
- **time** nos brinda la hora del día, en el formato de horas, minutos y segundos.

1.5 Programación en Java y NetBeans

1.5.1 Introducción

Java es un tipo de lenguaje orientado a objetos, que se fundamenta en lo código fuente de C y C++, pero con un modelo más simple y eliminando las herramientas de bajo nivel, con esto se logra obtener un lenguaje de programación en el que se pueden desarrollar aplicaciones parecidas a cómo se las piensa en la mente humana, ya que las divide en objetos, eliminado el problema de escribir códigos realmente largos (Penarrieta, 2011, p: 1).

1.5.2 Características del Lenguaje

- **Orientada a objetos:** Nos permite diseñar un software en donde los diferente datos estén unidos a sus operaciones, logrando que los datos y su código logren combinarse para la creación de objetos (Penarrieta, 2011, p: 7).
- **Simple:** El lenguaje basado en Java puede reducir hasta en 50% los errores más usuales que se dan en programación.
- **Distribuido:** Posee una gran cantidad tanto de librerías como de herramientas, que permite que su ejecución sea posible en varias máquinas y que ellas pueden interactuar entre sí.
- **Multiplataforma:** El lenguaje Java puede ejecutarse indistintamente del tipo de hardware y software, para ello Java utiliza un compilador, cuyo código va a un fichero objeto sin distinción de arquitectura del ordenador, cuyo código puede ser ejecutado desde cualquier ordenador que posea el sistema de ejecución Java Run Time (JRE), actualmente existen JRE para Solaris, Windows, Linux, Mac y Apple.

- **Robusto:** Se dice que Java es robusto debido a que es un lenguaje que realiza verificaciones durante la compilación y ejecución, logrando detectar errores durante el ciclo de desarrollo.

- **Seguro:** Se considera que las aplicaciones basadas en lenguaje Java son seguras por no acceder a zonas delicadas de memoria o sistema, también se considera seguro a nivel de lenguaje ya que eliminan punteros incensarios, por lo que no existe acceso ilegal a la memoria. A nivel de ejecución al código siempre se realiza un test para verificar que el código originado sea legal, no existan fragmentos de código ilegal.

- **Portable:** Java presente estándares de portabilidad, como son: utiliza enteros (32 bits), sus interfaces pueden ser implementadas en Linux, Mac o Windows.

- **Multihilo:** Java nos da la posibilidad de realizar funciones simultáneas dentro de una misma aplicación.

1.5.3 Sintaxis

Gran parte de ella está inspirada en C++, aunque la principal diferencia radica en que fue elaborada netamente para ser orientada a objetos, todo es considera un objeto y reside en alguna clase.

1.5.3.1 Aplicaciones Autónomas

Figura 44-1: Aplicación Autónoma

Fuente: Penarrieta, 2011

- El código fuente siempre se guarda en un archivo con el mismo nombre con extensión .java.

- El compilador genera un archivo que presenta la extensión .class por cada clase definida en el archivo fuente.

- Si existen programas que vayan a ejecutarse de forma independiente y autónoma deben contener el método “main()”

- Al utilizar la palabra void indica que el método que se encuentre en main no devuelve ningún valor.

- Al utilizar la palabra reservada public facilita que el método puede ser llamado desde otras clases fuera de la jerarquía.

1.5.3.2 Aplicaciones con ventanas Swing

Es tipo Swing es una biblioteca para utilizar la interfaz gráfica de la plataforma Java SE.

Figura 45-1: Aplicaciones con ventanas Swing

Fuente: Penarrieta, 2011

- La instrucción import indica al cumplidor que el paquete javax.swing será compilado.

- La parte HolaMundoSwing() inicializa el marco para llamar al método serDefaulCloseOperation (int) para realizar las operaciones por defecto al momento del control de cierre, en la barra de título es seleccionado WindowsConstants.DISPOSE_ON_CLOSE para liberar los recursos que se obtuvo para la visualización de la ventana cuando esta se cierre.

- El método main () para crearla instancia Hola y hacer que la ventana aparezca con el método setVisible de la superclase (clase de la que se hereda) con true.

1.5.4 Entornos de Desarrollo

Cabe destacar que Java no cuenta con un IDE propio, por lo que se utiliza aplicaciones externas como NetBeans para poder crear aplicaciones complejas en web, UML, base de datos, según las necesidades del programador.

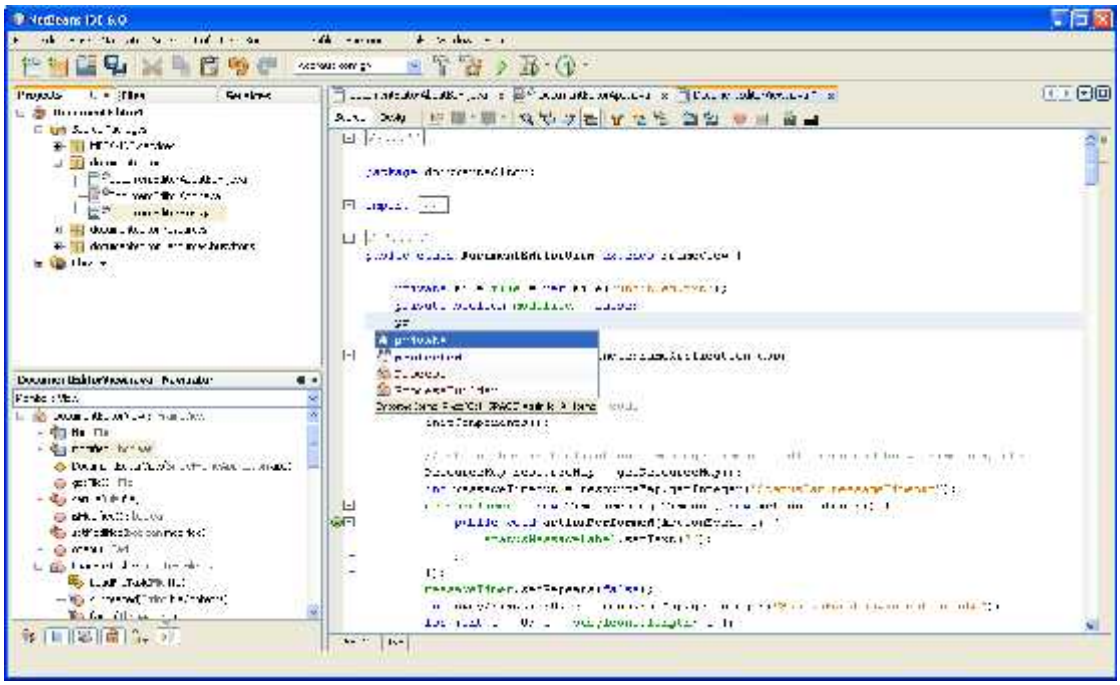


Figura 46-1: NetBeans y Java

Fuente: https://netbeans.org/

CAPÍTULO II

2 MARCO METODOLÓGICO

2.1 Análisis del Sistema Multimodal de Control de Asistencia

Para el análisis de las diferentes alternativas se utilizará el Método Cualitativo por puntos para escoger la opción más idónea de acuerdo a las políticas y requerimientos del sistema.

Se utilizará la siguiente escala relativa, para tomarla como punto inicial de partida:

Tabla 1-2: Escala relativa de ponderación

Alternativa	Valor en número	Denominación
A	1	Nula
B	2	Baja
C	3	Media
D	4	Máxima

Realizado por: Albuja J., Yépez J, 2016

2.1.1 Alternativas Plataforma Arduino

Realizamos una comparación entre las placas Arduino Uno, Arduino Mega y Arduino Yun, determinando el candidato que se va a utilizar en el sistema multimodal. Se tomara en cuenta las siguientes políticas relativas, requerimientos y la Tabla 1-2:

Políticas:

- La optimización de entradas y salidas es lo más importante.
- El Microcontrolador es igual de importante que el costo, pero menos importante que la adquisición de shield externas.

Requerimientos a considerar en las placas Arduino:

- F1: Costo de la placa
- F2: Optimización de entradas y salidas
- F3: Adquisición Shield externas

- F4: Microcontrolador

El proceso para la elección de la opción se refleja en las Tabla 2-2, Tabla 3-2, Tabla 4-2 y Tabla 5-2

Tabla 2- 2: Porcentaje de los requerimientos Plataforma Arduino

	F1	F2	F3	F4	Suma	Porcentaje
F1	-	0	0	1	1	14.29%
F2	1	-	1	1	3	42.85%
F3	1	0	-	1	2	28.57%
F4	1	0	0	-	1	14.29%
Total					7	100%

Realizado por: Albuja J., Yépez J, 2016.

Tabla 3-2: Peso relativo de los Requerimientos Plataforma Arduino

Variables	Arduino Uno	Peso	Arduino Mega	Peso	Arduino Yun	Peso
F1	35 dólares	4	63 dólares	3	96 dólares	1
F2	Posee 14 pines digitales y 6 analógicos, se requiere 6 pines digitales y 2 analógicas	4	Posee 54 pines digitales y 16 pines analógicos, se requiere 6 pines digitales y 2 analógicos.	2	Tiene 20 pines digitales y 12 entradas analógicas, se requiere 6 pines digitales y 2 analógicos.	3
F3	La mayoría de los shield existentes en el mercado son hechos para el Arduino uno.	4	Posee gran variedad de shield externas, está a la par con el Arduino uno, due y nano.	4	Posee ciertos shield externos pero el Shield Xbee necesario para nuestro caso no se encuentra disponible, requiere un shield personalizado.	3
F4	ATmega320 de 8 bits de 16 MHz de 32 KB en memoria flash, s Kb de SRAM y 1 Kb de EEPROM.	2	ATmega2560 de 8 bits funcionan con 8 Kb de SRAM, 4 KB de EEPROM y 256 KB de flash.	3	ATmega32U4 y un chip AR9331 Linux, con 32 KB de flash, 2.5KB de SRAM y 1 KB de EEPROM	2

Realizado por: Albuja J., Yépez J, 2016.

Tabla 4-2: Calificación de los requerimientos según las alternativas Plataforma Arduino

Ideas	F1		F2		F3		F4	
	Total	Calif.	Tota	Calificación	Total	Calif.	Total	Calif.

Arduino Uno	4	0.5	4	0.444	4	0.364	2	0.286
Arduino Mega	3	0.375	2	0.222	4	0.364	3	0.286
Arduino Yun	1	0.125	3	0.333	3	0.272	2	0.428
Total	8		9		11		7	

Realizado por: Albuja J., Yépez J, 2016.

Tabla 5-2: Porcentaje de aceptación de las alternativas Plataforma Arduino

Variables	Peso %	Arduino Uno		Arduino Mega		Arduino Yun	
		Calif.	Valor	Calif.	Valor	Calif.	Valor
F1	14.29%	0.5	7.15	0.375	5.36	0.125	1.79
F2	42.85%	0.444	19.03	0.222	9.51	0.333	14.27
F3	28.57%	0.364	10.4	0.364	10.4	0.272	7.77
F4	14.29%	0.286	4.09	0.286	4.09	0.428	6.12
	100%		40.67		29.36		29.95

Realizado por: Albuja J., Yépez J, 2016.

Mediante los cálculos realizados anteriormente se pudo obtener que la mejor alternativa se encuentre en la adquisición del Arduino Uno, obteniendo un 40.67% de éxito en comparación con el 29.39% del Arduino Mega y el 29.95% del Arduino Yun.

2.1.2 Alternativas Tecnología Inalámbrica

Realizamos una comparación entre las tecnologías inalámbricas ZigBee, Bluetooth y Wifi, determinando el candidato que se va a utilizar en el sistema multimodal. Se tomara en cuenta las siguientes políticas relativas, requerimientos y la Tabla1-2:

Políticas:

- La seguridad, el consumo de energía y duración de la batería es lo más importante.
- La topología de red es igual de importante que el costo y la cobertura.
- El enfoque de la aplicación es más importante que el costo
- La cobertura es menos de importante que la velocidad de trasmisión.

Requerimientos para determinar el medio inalámbrico:

- F1: Seguridad
- F2: Velocidad de Trasmisión
- F3: Consumo de Energía y Duración de Batería
- F4: Topología de Red
- F5: Costo
- F6: Cobertura
- F7: Enfoque de Aplicación

El proceso para la elección de la opción se refleja en las Tabla 6-2, Tabla 7-2, Tabla 8-2 y Tabla 9-2:

Tabla 6-2: Porcentaje de los requerimientos Tecnología Inalámbrica

	F1	F2	F3	F4	F5	F6	F7	Suma	Total
F1	-	1	1	1	1	1	1	6	25%
F2	0	-	0	1	1	1	0	3	12.5%
F3	1	1	-	1	1	1	1	6	25%
F4	0	0	0	-	1	1	0	2	8.33%
F5	0	0	0	1	-	1	0	2	8.33%
F6	0	0	0	1	1	-	0	2	8.33%
F7	0	0	0	1	1	1	-	3	12.5%
Total								24	100%

Realizado por: Albuja J., Yépez J, 2016.

Tabla 7-2: Peso relativo de los requerimientos Tecnología Inalámbrica

Variables	ZigBee	Peso	Bluetooth	Peso	WIFI	Peso
F1	El protocolo ZigBee utiliza un cifrado avanzado 128 AES y CCB-MCP	4	Utiliza cifrado de 64 y 128 bits	2	Utiliza protocolos WEP, WPA y WPA2 para su encriptación y seguridad.	3
F2	Posee una velocidad máxima de 250 Kbps	1	Puede alcanzar velocidades entre 720 Kbps a 1 Mbps	2	Posee velocidades sumamente altas, puede superar los 11000 Kbps.	4
F3	Muy Bajo, puede funcionar entre 100 a 1000+ días	4	Medio, puede funcionar entre 1-7 días	2	Alto, puede funcionar entre.5-5 días.	1
F4	Es una tecnología muy	4	Funciona en redes	1	Funciona en una topología	1

	versátil, puede funcionar en redes Ad-hoc, punto a punto, estrella o malla.		pequeñas como Ad-hoc.		Punto a hub.	
F5	60 dólares	3	50 dólares	3	60 dólares	3
F6	Posee un alcance entre 10 a 100+ metros.	4	Posee un alcance entre 10 a 100 metros	3	Posee un alcance entre 10 a 100 metros	4
F7	Para sistemas que requieran monitoreo y control	4	Se lo utiliza para reemplazar cable existente	2	Utilizada para trasmisión Web, email y video	2

Realizado por: Albuja J., Yépez J, 2016.

Tabla 8-2: Calificación de los requerimientos según las alternativas Tecnología Inalámbrica

Ideas	F1		F2		F3		F4		F5		F6		F7	
	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.
ZigBee	4	0.444	1	0.143	4	0.571	4	0.666	3	0.333	4	0.364	4	0.5
Bluetooth	2	0.222	2	0.286	2	0.286	1	0.167	3	0.333	3	0.272	2	0.25
WIFI	3	0.333	4	0.571	1	0.143	1	0.167	3	0.333	4	0.364	2	0.25
Total	9		7		7		6		9		11		8	

Realizado por: Albuja J., Yépez J, 2016.

Tabla 9-2: Porcentaje de aceptación de las alternativas Tecnología Inalámbrica

Variables	Peso %	ZigBee		Bluetooth		WIFI	
		Calif.	Valor	Calif.	Valor	Calif.	Valor
F1	25%	0.444	11.1	0.222	5.55	0.333	8.325
F2	12.5%	0.143	1.788	0.286	3.575	0.571	7.138
F3	25%	0.571	14.275	0.286	7.15	0.143	3.575
F4	8.33%	0.666	5.548	0.167	1.391	0.167	1.391
F5	8.33%	0.333	2.774	0.333	2.774	0.333	2.774
F6	8.33%	0.364	3.032	0.272	2.266	0.364	3.032
F7	12.5%	0.5	6.25	0.25	3.125	0.25	3.125
			44.7		25.7		29.07

Realizado por: Albuja J., Yépez J, 2016.

Mediante los cálculos realizados anteriormente se pudo obtener que la mejor alternativa se encuentre en la utilización de la tecnología ZigBee, obteniendo un 44.7% de éxito en comparación con el 25.7% de Bluetooth y el 29.07% de Wifi.

2.1.3 Alternativas para el control de asistencia

Realizamos una comparación entre las tecnologías RFID (Identificación por radio Frecuencia), Reconocimiento facial y acceso biométrico por huellas dactilares, determinando el candidato que se va a utilizar en el sistema multimodal. Se tomara en cuenta las siguientes políticas relativas y requerimientos y la Tabla 2-1:

Políticas:

- El costo es la variable más importante
- La seguridad es más de importante que la fiabilidad, tamaño y ubicación.
- La escalabilidad, tamaño y ubicación son igual de importantes.
- La fiabilidad es igual de importante que la escalabilidad

Requerimientos a considerar para la tecnología de control de asistencia:

- F1: Costo de los Equipos
- F2: Seguridad
- F3: Fiabilidad
- F4: Escalabilidad
- F5: Tamaño y Ubicación

El proceso para la elección de la opción se refleja en las Tabla10-2, Tabla 11-2, Tabla 12-2 y Tabla 13-2:

Tabla 10-2: Porcentaje de los requerimientos Tecnología Control de Asistencia

	F1	F2	F3	F4	F5	Suma	Total
F1	-	1	1	1	1	4	30.7%
F2	0	-	1	1	1	3	23.1%
F3	0	0	-	1	1	2	15.4%
F4	0	0	1	-	1	2	15.4%
F5	0	0	1	1	-	2	15.4%
Total						13	100%

Realizado por: Albuja J., Yépez J,2016.

Tabla 11-2: Peso relativo de los requerimientos Tecnología Control de Asistencia

Variables	Reconocimiento Facial	Peso	RFID Arduino	Peso	Biométrico huellas dactilares	Peso
F1	470 dólares	1	100 dólares	4	330 dólares	3
F2	Posee una alta seguridad ya que almacena plantillas de los rostros en su base de datos para compararlas posteriormente con la imagen captada, aunque puede confundir personas que tengas rasgos similares.	3	Es segura, posee un código único de identificación, para compararla en la base de datos. Puede ser expuesta a clonaciones.	2	Es seguro, utiliza huella dactilar la cual es única, la misma que no cambia con el pasar del tiempo, aunque puede ser clonada.	3
F3	No es muy fiable debido principalmente que en ciertas ocasiones el rostro de los individuos cambia de patrones al transcurrir el tiempo y el sistema lo desconoce.	2	Es muy fiable ya que posee un UID único para establecer, obtener y verificar la información desde la base de datos.	4	Es una alternativa fiable, se fundamenta en huellas dactilares, siendo únicas.	3
F4	Es escalable, se puede incorporar distintos módulos, siempre que tengan conexión con el servidor.	4	Es escalable ya que es una tecnología que se utiliza para realizar inventarios a gran escala.	4	Los sistemas biométricos son creados para ser escalables, siendo probados mucho en la actualidad.	4
F5	Son de tamaño pequeños, siendo posibles instalar sin ningún problema.	3	Son de tamaño muy pequeños, siendo posibles instalarlos sin ningún problema.	4	Son de tamaño pequeños, siendo posibles instalarlos sin ningún problema.	3

Realizado por: Albuja J., Yépez J,2016.

Tabla 12-2: Calificación de los requerimientos según las alternativas Tecnología Control de Asistencia

Ideas	F1		F2		F3		F4		F5	
	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.	Total	Calif.
Reconocimiento Facial	1	0.125	3	0.375	2	0.222	4	0.333	3	0.3
RFID	4	0.5	2	0.25	4	0.444	4	0.333	4	0.4
Biométrico de huellas dactilares	3	0.375	3	0.375	3	0.333	4	0.333	3	0.3
Total	8		8		9		12		10	

Realizado por: Albuja J., Yépez J, 2016.

Tabla 13-2: Porcentaje de aceptación de las alternativas Tecnología Control de Asistencia

Variables	Peso %	Reconocimiento Facial		RFID		Biométrico de huellas dactilares	
		Calif.	Valor	Calif.	Valor	Calif.	Valor
F1	30.7%	0.125	3.838	0.5	15.35	0.375	11.513
F2	23.1%	0.375	8.663	0.25	5.775	0.375	8.663
F3	15.4%	0.222	3.419	0.444	6.838	0.333	5.128
F4	15.4%	0.333	5.128	0.333	5.128	0.333	5.128
F5	15.4%	0.3	4.62	0.4	6.16	0.3	4.62
	100%		25.67		39.25		35.052

Realizado por: Albuja J., Yépez J, 2016.

Mediante los cálculos realizados anteriormente se pudo obtener que la mejor alternativa se encuentre en la utilización de la tecnología RFID obteniendo un 39.25% de éxito en comparación con el 25.67% del Reconocimiento facial y el 35.052% del Acceso Biométrico por huella dactilar.

2.2 Diseño del Sistema

El sistema está conformado por los siguientes elementos, que tienen acceso a un servidor de base de datos como se indica en la Figura 1-2:

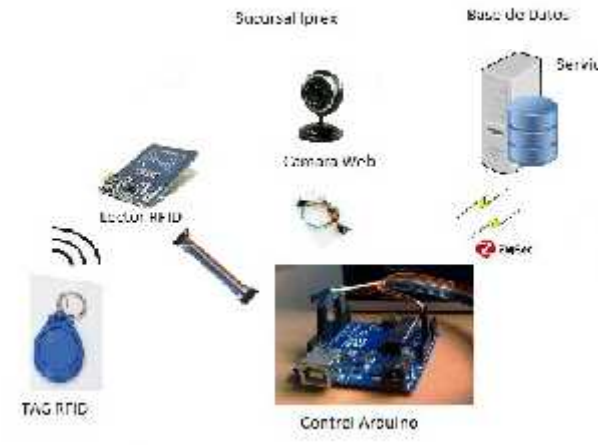


Figura 1-2: Esquema del sistema multimodal de control de asistencia
Realizado por: Albuja J., Yépez J, 2016.

2.2.1 Funcionamiento

El sistema estará formado por una placa Arduino Uno R3 que recogerá la información obtenida por el lector RFID de las tarjetas o llaveros suministrados a IPREX y la cámara WEB. Posteriormente esta información será enviada inalámbricamente hasta el servidor de la Base de Datos para su verificación y almacenamiento.

2.3 Etapa Hardware

2.3.1 Plataforma Arduino Uno

Es el modelo que se va utilizar, debido a la excelente documentación que posee, cuenta con aditamentos externos que lo posibilitan interconectarse con diversos equipos y/o tecnologías como: LCD, teclados, NFC, RFID, WIFI, entre otros, por ende es una placa capaz de ser utilizada en cualquier aplicación.

Es una placa electrónica, que tiene 14 pines digitales para entrada/salida, 6 entradas analógicas, un cristal de cuarzo para oscilaciones de 16 MHz, puerto USB, entrada para fuente de alimentación (7 V-12V), cabecera ICSP y botón de reinicio (Figura 2-2). En la tabla14-2 se encuentra su ficha técnica.

Tabla 14-2: Ficha Técnica Arduino Uno

Ficha Técnica	
Microcontrolador	ATmega328P
Voltaje de Funcionamiento	5V
Voltaje Recomendado de Entrada	7-12V
Voltaje de Entrada (Límites)	6-20V
Pines Digitales I/O	14
Pines Digitales PWM	6
Pines de Entrada Analógicos:	6
Corriente DC por I/O Pin	20 mA
Corriente DC para Pin 3.3V	50 mA
Flash Memory	32 KB (ATmega328P)
SRAM	2 KB
EEPROM	1 KB
Velocidad de Reloj	16 MHz

Fuente: https://www.arduino.cc/en/Main/ArduinoBoardUno

Figura 2- 2: Arduino Uno

Fuente: http://gammon.com.au/

La placa Arduino es el motor que controlara como se debe tratar los datos obtenidos tanto del módulo RFID como de la cámara web, y su posterior envió mediante tecnología ZigBee con ayuda de los módulos Xbee.

2.3.2 Adafruit PN532 RFID/NFC RFID PN532 NFC RFID

Es un módulo utilizado por su versatilidad, debido a que trabaja tanto con NFC como con tecnología RFID mediante MiFare Cards y Tags. El chip PN532 está diseñado para sistemas de 3.3V o 5V siempre que se tenga un cambiador de nivel para pasarlo a 3.3V.

Posee una librería Adafruit-PN532 (Disponible en la Página de Arduino) para obtener la capacidad de leer tarjetas Mifare (Tecnología RFID) y trabajar en conjunto con Arduino,

indispensable en nuestro esquema, estará conectado a nuestro Arduino Uno como se ve en la Figura 3-2:

Este shield puede funcionar utilizando dos protocolos, el primero el I2C, el cual es utilizado de manera predeterminada ocupando los pines análogos 4 y 5, además cuenta con un pin Digital (#2 por defecto) que realiza la alerta de interrupciones (teléfono, tarjeta, tag, etc) automáticamente, siendo posible utilizar otro pin en caso que sea necesario. La segunda manera es optando por el protocolo SPI en donde se puede ocupar 4 pines digitales cualesquiera siempre y cuando se suelde 2 jumpers a la PCB superior de los módulos Arduino.

Figura 3-2: Conexión entre Arduino UNO y Adafruit PN532 RFID/NFC
Realizado por: Albuja J., Yépez J, 2016.

2.3.3 MiFare Classic (13.56MHz RFID/NFC) Card

Es una tarjeta pasiva MiFare utilizada en trenes, autobuses y/o cualquier sistema que ocupe su tecnología de proximidad, contiene un chip interno NXP 50 con una antena, son leídos por lectores que ocupen la frecuencia de 13.26 MHz, en nuestro caso el PN532 RFID/NFC para Arduino (Figura 4-2), no obstante presenta restricciones con Cards de Felica por el método de codificación.

El chip es capaz de escribir y almacenar datos en su memoria EEPROM grabable hasta 1 Kb, con más de 100000 re-escrituras, además posee un ID de 4 bytes permanente grabado en el chip que lo distingue de las demás etiquetas siendo único. Sus características se encuentran en la Tabla 15-2:

Tabla 15-2: Especificaciones Técnicas Tarjeta MiFare Classic

Chip S50	Especificaciones Tag
Almacenamiento EEPROM no volátil de 1 Kb	85.5mm x 54mm x 1mm / 3.36” x 2.1” x 0.03”
4 Bytes de identificación del chip	6.3 gr/0.2 oz
Frecuencia de 13.56 MHz	Funciona con 4 formas de lectores
Construida con clave de 48 bits	

Fuente: https://www.adafruit.com/products/359

Figura 4-2: Tarjeta MiFare Classic y Adafruit PN532 RFID/NFC

Fuente: http://www.adafruit.com/

2.3.4 Dispositivos para trasmisión y recepción de Datos

Estos dispositivos serán los encargados de establecer el canal y llevar los datos obtenidos mediante el lector RFID y la cámara WEB del sistema de control, hacia la base de datos para su almacenamiento y verificación, para ello se optó por utilizar módulos Xbee de Digi que permiten la utilización del protocolo ZigBee para la transferencia de datos.

Los dispositivos Xbee se pueden presentar en varias versiones S1 y S2, ambas pueden ser utilizadas para redes punto-punto, punto-multipunto o tipo MESH, en nuestro caso se optara por la versión S1 debido a su alcance y configuración.

2.3.4.1 Módulos Xbee S1 Whip Antena

Los módulos Xbee S1 (Figura 5-2) vienen con su firmware pre configurado, por lo que se requiere solo de su configuración para su utilización. Esta es realizada mediante la utilización del Xbee Explorer USB y el programa X-CTU.

Figura 5-2: XBee S1

Realizado por: Albuja J., Yépez J, 2016.

Este dispositivo al momento de implementarse necesita de un circuito básico para ser utilizado, como se ve en la Figura 6-2, además debemos conocer cómo están distribuidos sus pines, para comprender la interacción con la placa Arduino, este se ve el Figura 7-2.

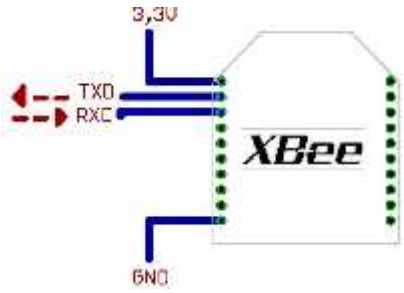


Figura 6-2: Conexiones mínimas requeridas Xbee

Fuente: http://www.olimex.cl/

Figura 7-2: Interconexión Arduino Xbee

Fuente: http://www.olimex.cl/

2.3.4.2 Xbee Explorer USB

Es una unidad fácil de utilizar para la configuración de los módulos Xbee de la serie 1 y 2.5, estándar y pro (Figura 8-2 y Figura 9-2), utiliza un puerto mini USB para conectarse a la PC y poderlo configurar mediante el programa X-CTU.

Figura 8-2: XBee Explorer USB

Realizado por: Albuja J., Yépez J, 2016.

Figura 9-2: Conexión Xbee S1 con Xbee Explorer USB

Realizado por: Albuja J., Yépez J, 2016.

2.3.4.3 Xbee Shield Arduino

Es un dispositivo que nos permite conectar a una placa de Arduino un Módulo Xbee de Maxtream, para comunicaciones inalámbricas usando el protocolo Zigbee (Figura 10-2). Funciona con módulos de la serie 1, 2.5, estándar y pro.

Figura 10-2: Xbee Shield Para Arduino

Fuente: http://www.leophix.com

2.4 Etapa Software

Para el diseño de nuestro sistema tendremos en consideración los siguientes programas necesarios para establecer el sistema multimodal para el control de asistencia:

2.4.1 X-CTU

X-CTU es una aplicación con interfaz gráfica (Figura 11-2) multiplataforma de libre distribución, que nos permite la comunicación con módulos de Radiofrecuencia fabricados por DIGI para su configuración.

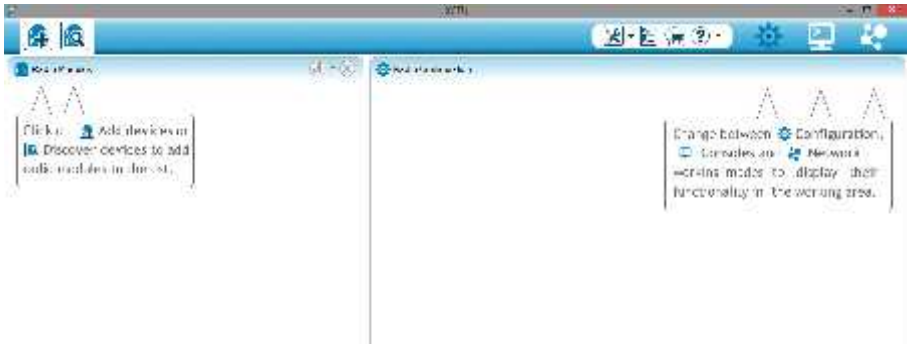


Figura 11-2: Interfaz gráfica Programa X-CTU

Realizado por: Albuja J., Yépez J, 2016.

Características del Programa:

- Puede utilizarse para administrar y configurar varios dispositivos RF.

- Puede trabajar en modo API y AT.

- Guarda sesiones para visualizarlas en otra PC

- Posee documentación completa para accederla en cualquier instancia.

Se lo descarga de la página oficial http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

Este programa nos permitirá establecer la configuración de los módulos Xbee para que cumpla con los requerimientos de nuestro sistema.

2.4.2 Software Arduino

Es un IDE de código Abierto de Arduino multiplataforma que nos permite crear programas para implementarlos en las placas Arduino, logrando aplicaciones que resuelvan nuestras necesidades. Es un entorno desarrollado sobre java. Su interfaz gráfica se ve en la Figura 12-2.

Este software será el encargado de permitirnos configurar la placa Arduino uno estableciendo las sentencias necesarias para levantar el sistema multimodal.

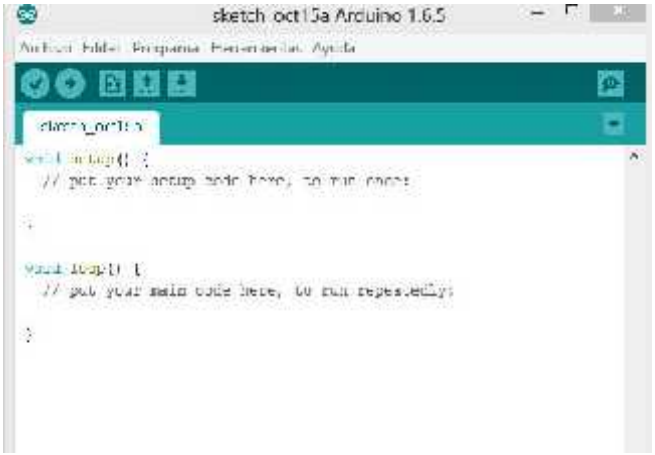


Figura 12-2: Interfaz Gráfica Software Arduino

Realizado por: Albuja J., Yépez J, 2016.

2.4.3 NetBeans IDE

NetBeans IDE (Figura 13-2) es un entorno de desarrollo integrado de libre distribución y gratuidad, utilizado para escribir, compilar, depurar y ejecutar programas basados en programación Java, C/C++, HTML, PHP entre otros, como aplicaciones de escritorio, páginas web.

NetBeans posee conjuntos de componentes de software denominados módulos que son los responsables de extender la capacidad de interactuar con apis de NetBeans y atribuir la posibilidad de ser escalable debido a que las aplicaciones creadas a partir de dichos módulos pueden ser extendidas fácilmente utilizando otros módulos, puede ser descargada desde su página oficial https://netbeans.org/ en su sección de descargas.

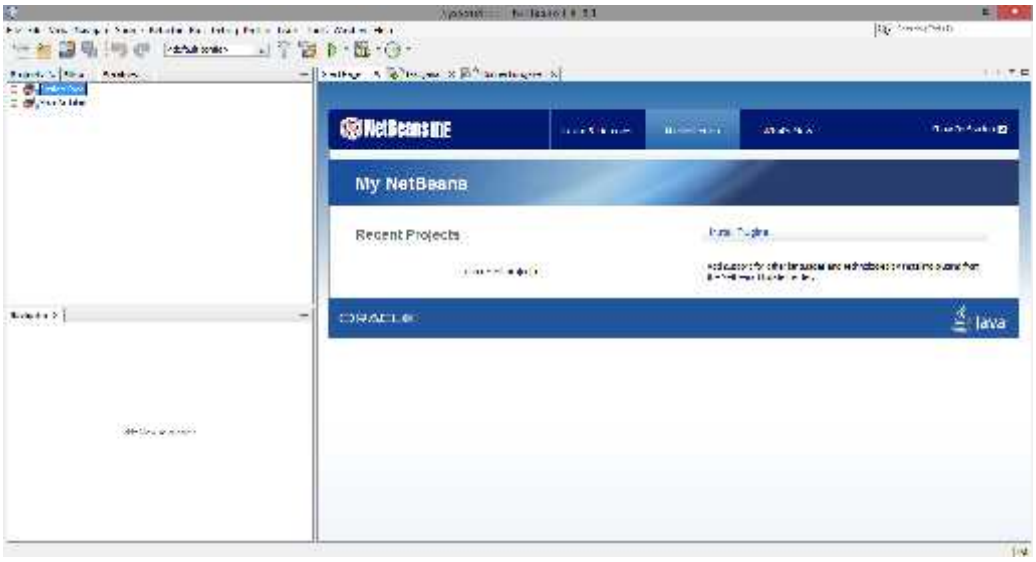


Figura 13-2: Interfaz Gráfica IDE NetBeans

Realizado por: Albuja J., Yépez J, 2016.

En nuestro esquema NetBeans será el encargado de realizar las siguientes funciones dentro del sistema multimodal.

2.4.3.1 Driver Xbee-API Mode

Esta parte será la encargada de permitir que la información transmitida a través de los módulos Xbee se encapsulen en tramas con una estructura de datos propia del protocolo API2 para que puede ser procesadas por el ordenador, en consecuencia la función principal del driver es permitir la recepción de dichas tramas y su posterior desencapsulación para extraer los datos contenidos y posteriormente reenviar la respuesta para que puede ser receptada e interpretada por los módulos Xbee.

2.4.3.2 Conexión a la Base de Datos

Esta parte es vital en nuestro diseño, será la encargada de establecer la conexión con la base de datos para verificar que los usuarios que acceden mediante los Tags RFID (Figura 4-2) puedan autenticarse, además de ayudarnos para el tratamiento de la imagen.

2.4.3.3 Interfaz de Usuario

Esta sección nos permitirá obtener un medio para que el usuario pueda comunicarse a la información contenida en la Base de datos para su verificación, cambio, actualización o

remoción de registros de una manera amigable, en fin será la encargada de darnos acceso total a la Base de Datos por parte del usuario.

2.4.3.4 Recepción de Imagen

Es un programa basado en Java que nos permitirá receptar la imagen enviada por los módulos Xbee para su posterior almacenamiento junto a nuestra Base de Datos.

2.5 Cámara Serial TTL

Estos módulos (Figura 14-2) son una buena opción cuando se requieren capturar imágenes o controlar flujo de vídeo, estos módulos presentan características integradas, como la capacidad de cambiar el brillo, saturación, tono de las imágenes, auto-contraste y ajuste automático de brillo.

Posee una resolución variable que oscila entre 640x480, 320x240 o 160x120 pixeles para capturar imágenes en formato JPEG dependiendo del uso que se requiera, funciona a una velocidad de 38400 baudios por defecto, posee un alcance de monitoreo de 10 metros ajustable hasta 15 metros, funciona con un voltaje de 3.3V.

Figura 14-2: Cámara Serial TTL

Fuente: https://learn.adafruit.com/

2.5.1 CommTool

CommTool es una aplicación utilizada con Windows disponible en http://www.adafruit.com/datasheets/VC0706CommTool(EN)%20Setup%20V1-00.exe ,que nos

permite configurar cámaras TTL, en nuestro caso utilizaremos nuestro Arduino Uno como un chip FTDI para el test y cambio de parámetros en la cámara, para ello manejaremos un boceto en blanco en la configuración del Arduino (Figura 15-2) que nos servirá como intermediario entre la PC y la cámara conectado en la placa.

Figura 15-2: Boceto en Blanco Arduino

Realizado por: Albuja J., Yépez J, 2016.

A continuación nos mostrará una pantalla (Figura 16-2), en donde seleccionaremos el puerto com donde tenemos conectado nuestra placa reducida

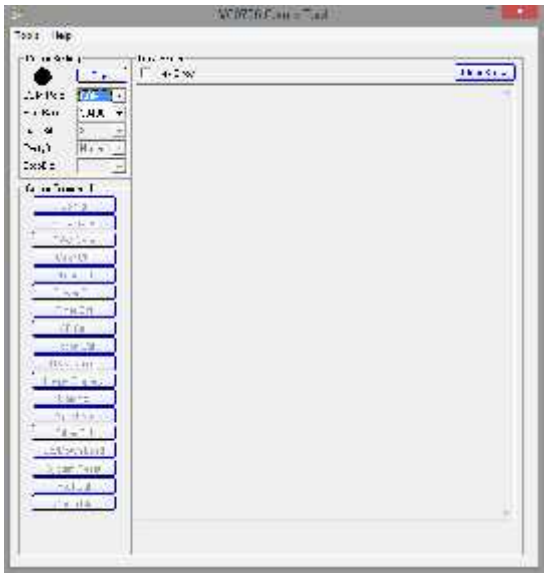


Figura 16-2: Interfaz Comm Tool

Realizado por: Albuja J., Yépez J, 2016.

2.6 Pantalla de cristal líquido

Las pantallas de cristal líquido (LCD) nos permiten visualizar mensajes y/o gráficos, nosotros utilizaremos un LCD de 16x2 (16 caracteres por 2 filas), para su configuración se requiere de la librería LiquidCrystal por defecto, en nuestro caso utilizaremos la librería LiquidCrystal_I2C,

debido a que la pantalla funcionara mediante comunicación I2C con la placa Arduino Uno. El LCD se ve en la figura 17-2.

Figura 17-2: LCD de 16x2

Realizado por: Albuja J., Yépez J, 2016.

2.6.1 Módulo I2C para LCD 16x2

Es utilizado para trasmitir la información mediante dos líneas solamente, la primera es utilizada para los datos y la otra para la señal de reloj, su conexión con la placa Arduino uno es de la siguiente manera el PIN SDA al pin 4, SCL al pin A5 y VCC (5V) y GND. Además debemos utilizar la librería wire.h para su reconocimiento junto al LCD.

Figura 18-2: I2C para LCD

Fuente: http://www.geeetech.com/

2.7 Base de Datos

El esquema de la base de Datos nos ayudara a obtener un repositorio donde guardaremos toda nuestra información relacionada con los usuarios y los registros, obteniendo una base de datos capaz de actualizarse automáticamente.

Para lograr dicha base de datos vamos a utilizar el software MySQL Workbench que es una herramienta libre multiplataforma visual para la elaboración de base de datos que está disponible en la página oficial http://dev.mysql.com/downloads/workbench/.

Para el diseño tomaremos en cuenta dos tablas fundamentales, la tabla usuario y la tabla registro, para ver cómo están estructuras dichas tablas vamos a optar por la utilización del programa Power Designer para establecer los atributos y relaciones como se ve en la figura 19-2 mediante su estructura conceptual y posterior observaremos su relación física en la figura 20-2.

Figura 19-2: Modelo Conceptual

Realizado por: Albuja J., Yépez J, 2016.

En su modelo conceptual se puede observar que entre ellos existe una relación de uno a muchos en consecuencia un usuario puede tener uno o más registro para su visualización.

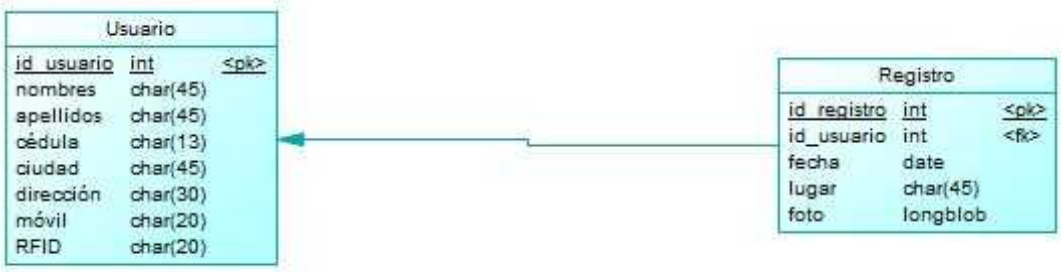


Figura 20-2: Modelo Físico

Realizado por: Albuja J., Yépez J, 2016.

En cambio en modelo físico podemos notar que la entidad registro siempre va a estar vinculada a la entidad usuario por id_usuario siendo este el modelo que se va a implementar en la base de datos.

2.8 IMPLEMETACIÓN DEL SISTEMA MULTIMODAL DE ASISTENCIA

2.8.1 Configuración Arduino Uno

Esta parte será la encargada de controlar tres diversos elementos en nuestro sistema, primero interpreta los datos recibidos del sensor NFC/RFID para su envió y verificación en la base de

datos, posteriormente recepta los datos obtenidos de la cámara TTL y el último establece el canal de comunicación mediante tecnología ZigBee con Xbee para el envío y recepción de información del sistema multimodal.

2.8.1.1 Instalación Software Arduino

El software (IDE) ocupado en placas reducidas Arduino es de código abierto multiplataforma por lo que no se requiere licencia para su utilización, este software se lo puede descargar desde la página oficial en el siguiente link https://www.arduino.cc/en/Main/Software.

Su instalación es muy sencilla, al haber descargado el instalador simplemente lo ejecutamos y seguimos las ventanas hasta que el programa quede instalado mostradnos la siguiente ventana Figura 21-2:

Figura 21-2: Pantalla de Inicio Software Arduino

Realizado por: Albuja J., Yépez J, 2016.

Una vez instalado el Software, en nuestro caso el de Windows versión 1.6.5, a continuación procedemos a conectar la placa Arduino junto con el cable USB a la PC de la siguiente manera (Figura 22-2) y seleccionamos el modelo de la placa a utilizar y el puerto serie conectado a Arduino (Figura 23-2).

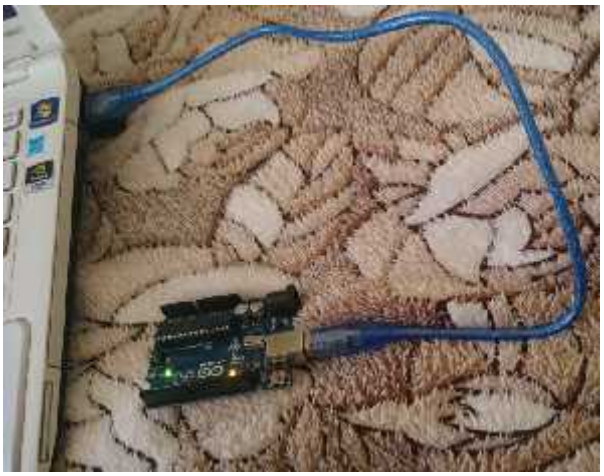


Figura 22-2: Conexión Arduino Uno y PC

Realizado por: Albuja J., Yépez J, 2016.

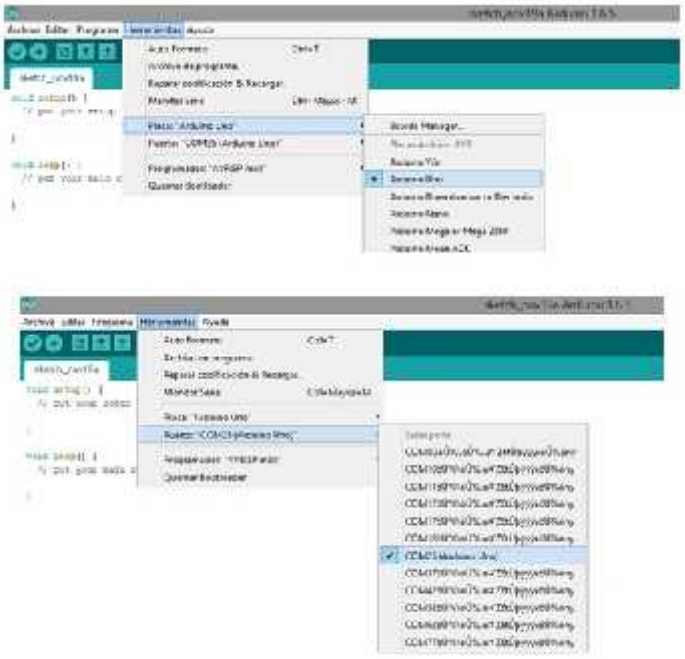


Figura 23-2: Configuración Inicial entre Arduino Uno y PC

Realizado por: Albuja J., Yépez J, 2016.

Realizado lo expuesto anteriormente obtendremos una comunicación correcta entre placas Arduino y PC, para poder subir la configuración posteriormente.

2.8.1.2 Instalación de Librerías Externas o Defecto

Para el correcto funcionamiento de los dispositivos se observa la tabla 16-2:

Tabla 16-2: Librerías Necesarias para el Funcionamiento

Dispositivo	Librería por Defecto	Librería Externa	url externa
Lector NFC/RFID	x	<Adafruit_PN532.h>	https://github.com/adafruit/Adafruit-PN532/
Módulo Xbee	x	<XBee.h>	https://github.com/andrewrapp/xbee-arduino
LCD con I2C	<Wire.h>	<LiquidCrystal_I2C.h>	http://www.frostcode.es/descargas/LiquidCrystal_I2C1602V1-master.zip
Carama TTL	<SoftwareSerial.h>	<Adafruit_VC0706.h>	https://github.com/adafruit/Adafruit-VC0706-Serial-Camera-Library

Realizado por: Albuja J., Yépez J, 2016.

Para instalar las librerías se descargar del url puesto anteriormente, se guarda en una carpeta de su preferencia en nuestro caso: C:\Program Files (x86)\Arduino\libraries\ y se realiza la importación desde nuestro Arduino (Figura 24-2)

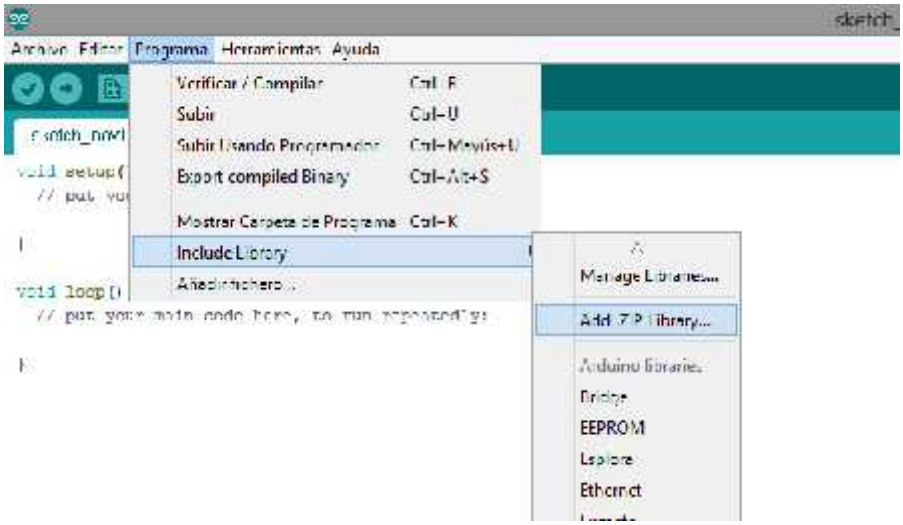


Figura 24-2: Añadir Librería en Arduino

Realizado por: Albuja J., Yépez J, 2016.

A continuación se selecciona el archivo .zip donde lo tengamos guardamos, damos aceptar y verificamos que la librería haya siso cargado con éxito. (Figura 25-2)

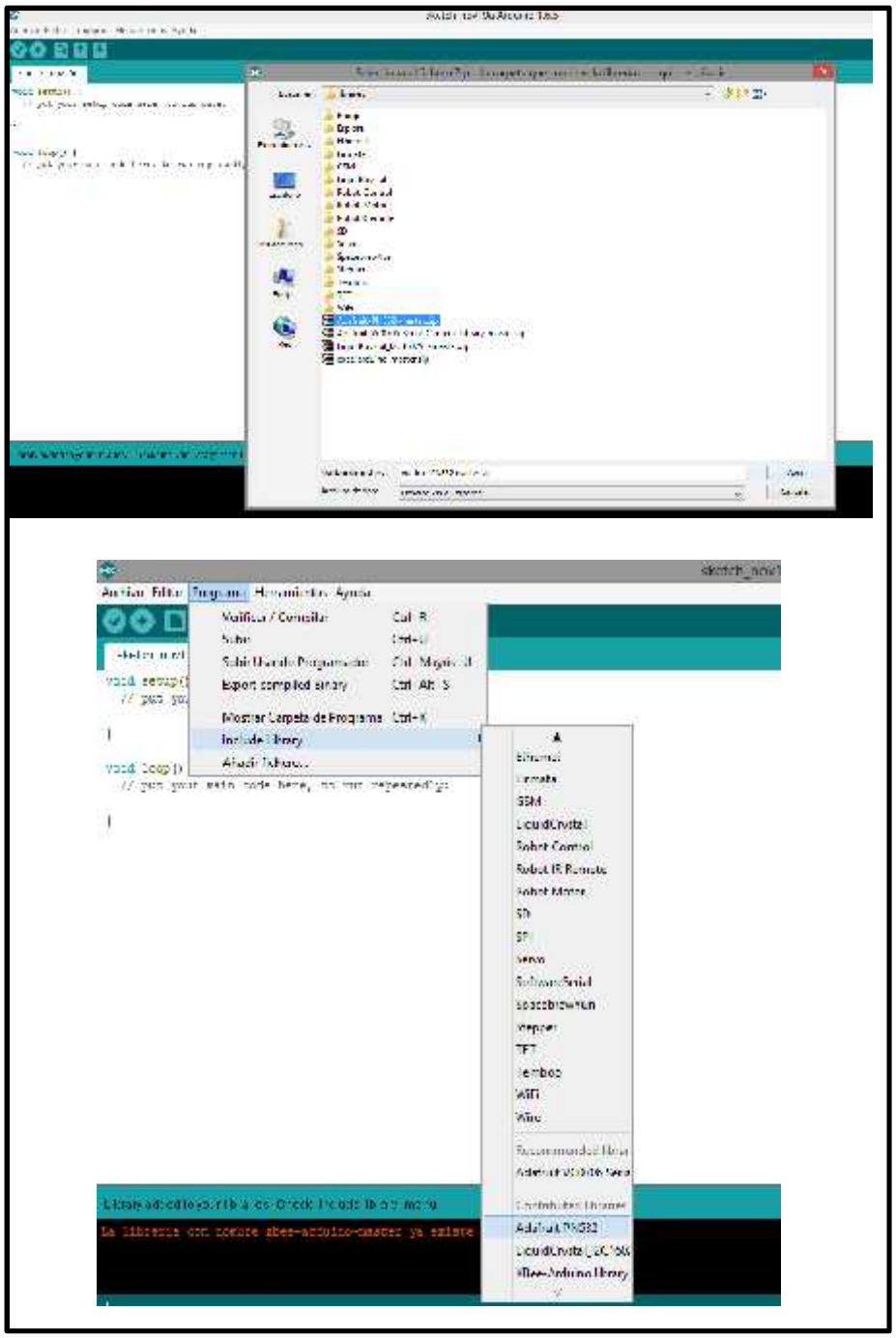


Figura 25-2: Selección y Verificación de Librería en Arduino

Realizado por: Albuja J., Yépez J, 2016.

2.8.1.3 Programación en Arduino

Para comenzar a escritor el código lo primero de debemos hacer es incluir todas las librerías descritas en la Tabla 16-2, esto se realiza seleccionando Programa, Include Library y seleccionamos la librería que deseamos como se indica en la figura 26-2, al terminar el proceso tenderemos algo parecido a lo que obtenemos en la figura 27-2.

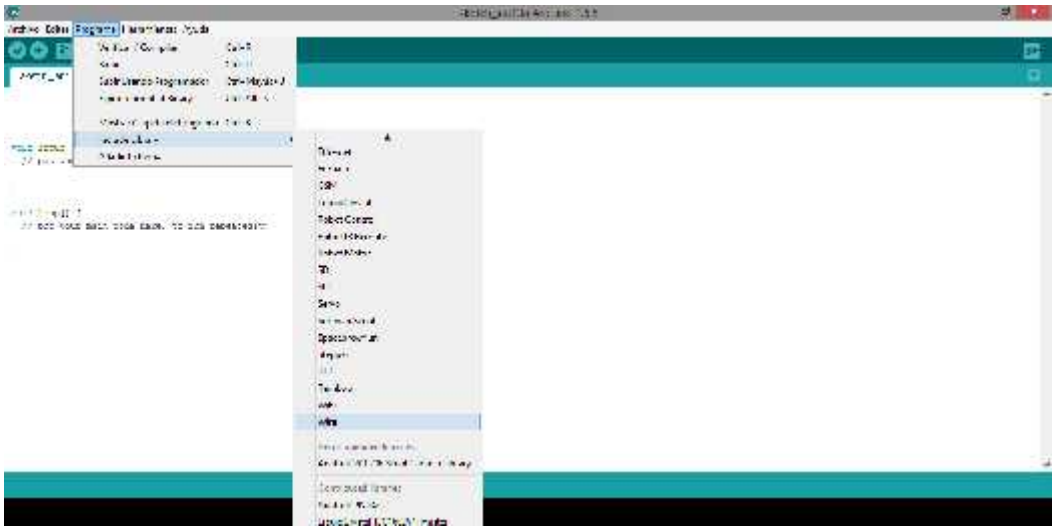


Figura 26-2: Selección de Librerías Arduino

Realizado por: Albuja J., Yépez J, 2016.

Figura 27-2: Constatación de Librerías

Realizado por: Albuja J., Yépez J, 2016.

Una vez realizado procederemos a iniciar con la programación en el Arduino Uno como se ve en la figura 28-2, el código completo se encuentra en el ANEXO A, con el código anterior el dispositivo capta las señales de los módulos RFID y de la cámara TTL para su posterior envió, verificación en la base de datos y almacenamiento de la imagen.

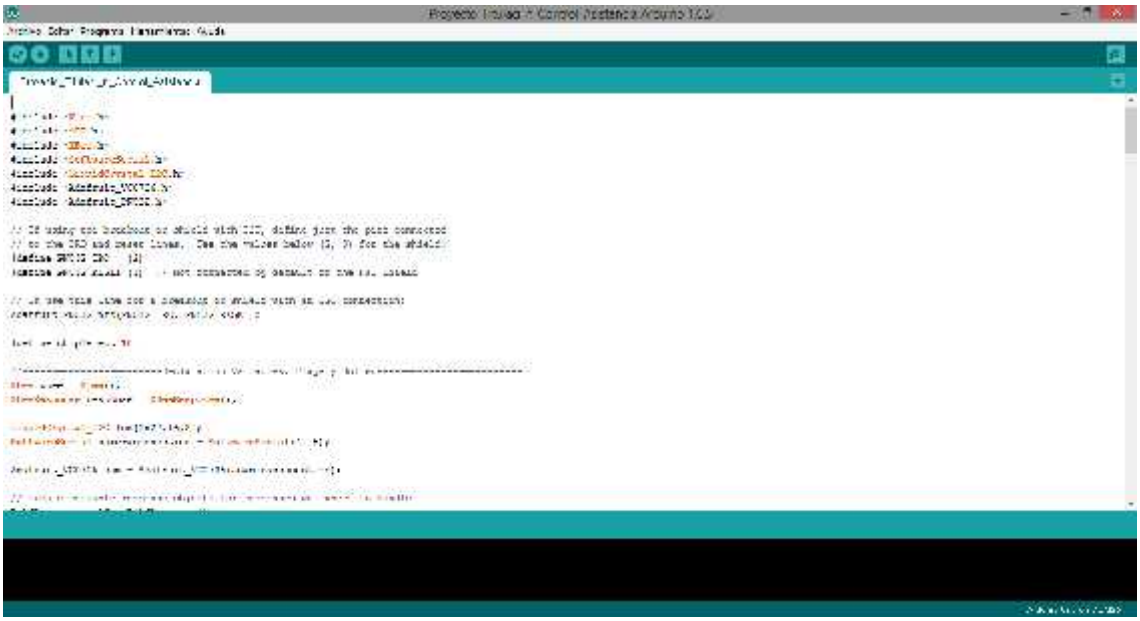


Figura 28-2: Configuración Arduino Uno

Realizado por: Albuja J., Yépez J, 2016.

Posterior a escribir el código procedemos a guardar la configuración seleccionando Archivo, Guardar como, en nuestro caso con el nombre *Proyecto_Titulaci_n_Control_Asistencia*, para verificar que el código se encuentra correcto presionamos el símbolo √ (Figura 29-2).

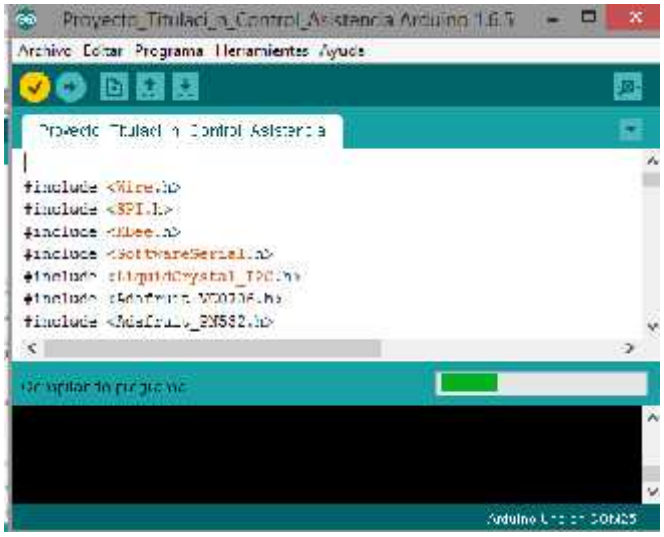


Figura 29-2: Compilación del Programa

Realizado por: Albuja J., Yépez J, 2016.

Al pasar este paso si no existen errores obtendremos la figura 30-2, y seleccionaremos el símbolo → para cargar el programa en el Arduino Uno (Figura 31-2)

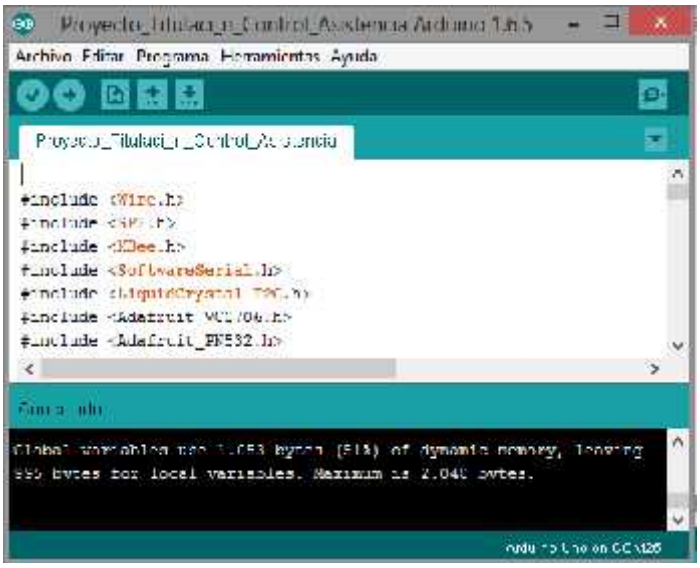


Figura 30-2: Compilación exitosa del Programa

Realizado por: Albuja J., Yépez J, 2016.

Figura 31-2: Subir programación a Arduino Uno

Realizado por: Albuja J., Yépez J, 2016.

2.8.2 Configuración Módulos Xbee S1

Para la configuración debemos tener dos parte principales, primera, los componentes vistos en la Figura 2-9 para su conexión con nuestra PC, y como segundo el programa X-CTU instalado en la PC para establecer los parámetros.

2.8.2.1 Instalación programa X-CTU

Para la instalación debemos descargar el programa desde la página oficial http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu, y descargar la versión necesaria para nuestra PC en nuestro caso se descargó el instalador para los sistemas operativos de Windows versión 6.2.0, al tener descardo el instalador, su instalación es sumamente sencilla, lo único que debemos realizar es ejecutar el instalador y seguir las ventanas emergentes, lo iniciamos y obtendremos la siguiente ventana (Figura 32-2).

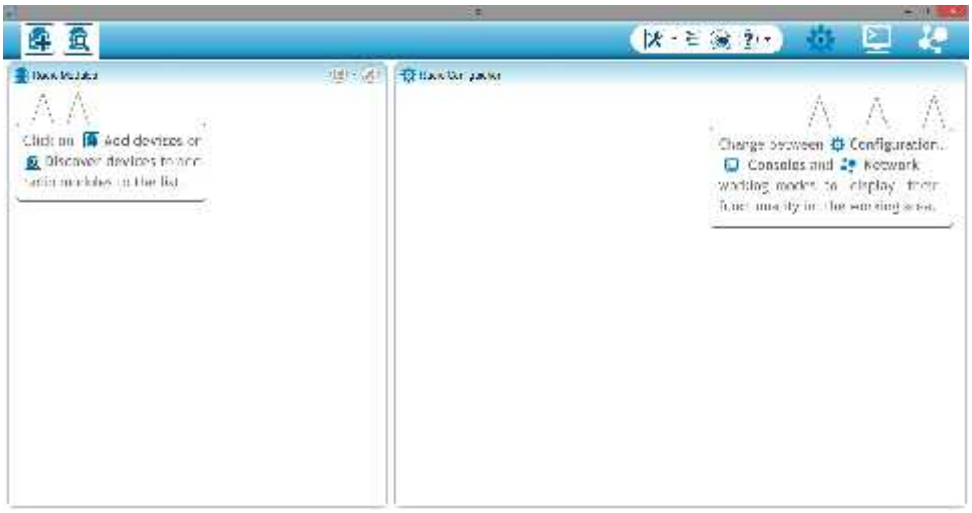


Figura 32-2: Interfaz de Inicio de XCTU

Realizado por: Albuja J., Yépez J, 2016.

2.8.2.2 Búsqueda de Dispositivos

El primer paso que debemos realizar es la búsqueda y reconocimiento del dispositivo Xbee, para ello insertamos la Xbee junto con el Xbee Explorer USB a nuestra PC y pulsamos el símbolo visto en la figura 33-2, con esto obtenemos la figura 34-2 en donde seleccionaremos *el puerto com* donde fue reconocido nuestro módulo Xbee Explorer USB y seleccionaremos *Next,* surgiendo la figura 35-2 donde escogeremos el Baud rate de 9600 y 115200 y pulsaremos *Finish* obteniendo la figura 36-2 si el dispositivo fue encontrado exitosamente en donde seleccionaremos el dispositivo y daremos click en *Add selected devices.*

Caso contrario saldrá una advertencia que no se encontró ningún dispositivo, en donde tendremos que ver la conexión del Xbee Explorer USB con nuestra PC.

Figura 33-2: Búsqueda de Dispositivos

Realizado por: Albuja J., Yépez J, 2016.

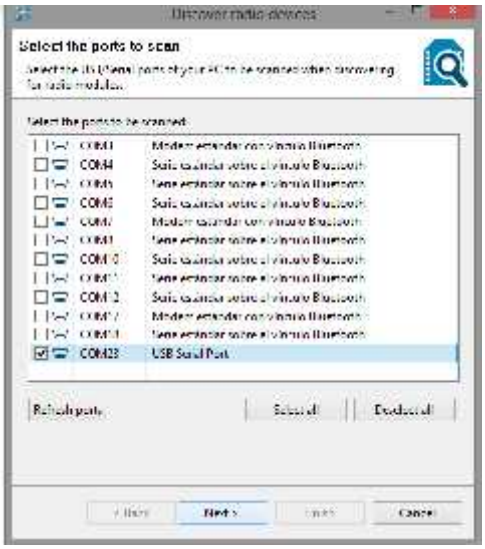


Figura 34-2: Selección puerto COM

Realizado por: Albuja J., Yépez J, 2016.

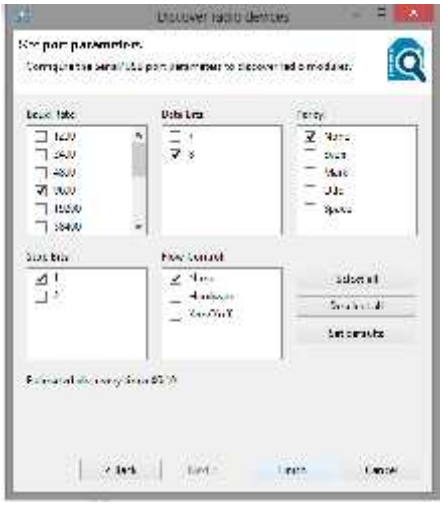


Figura 35-2: Selección de Baud rate-Velocidad

Realizado por: Albuja J., Yépez J, 2016.

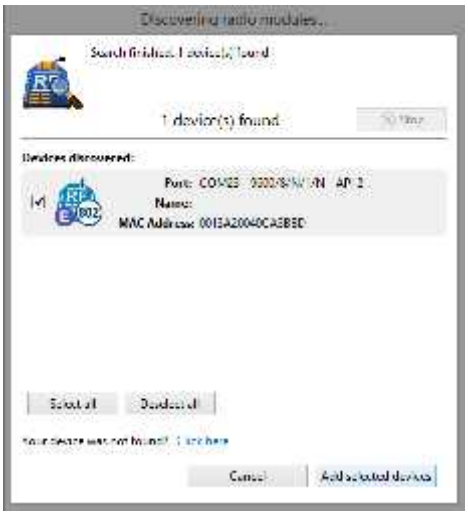


Figura 36-2: Reconocimiento Y Adición de Dispositivo

Realizado por: Albuja J., Yépez J, 2016.

Si todos los pasos anteriores no dieron ningún error obtendremos la figura 37-2 donde podremos ver y cambiar los parámetros de red del dispositivo.

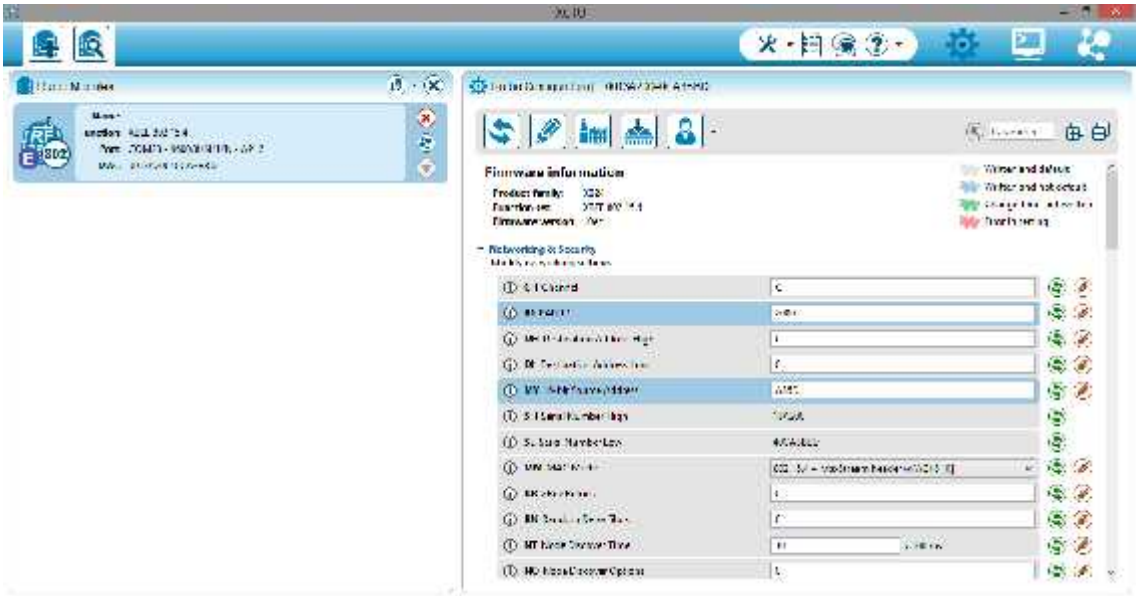


Figura 37-2: Reconocimiento exitoso del Dispositivo Xbee

Realizado por: Albuja J., Yépez J, 2016.

2.8.2.3 Parámetros de Red

En esta parte vamos a cambiar los parámetros de red necesarios para que nuestro sistema multimodal pueda comunicarse con la Pc donde nuestra base de datos se encuentra. Los parámetros cambiados se encuentran señalados en las figuras 38-2 y 39-2.

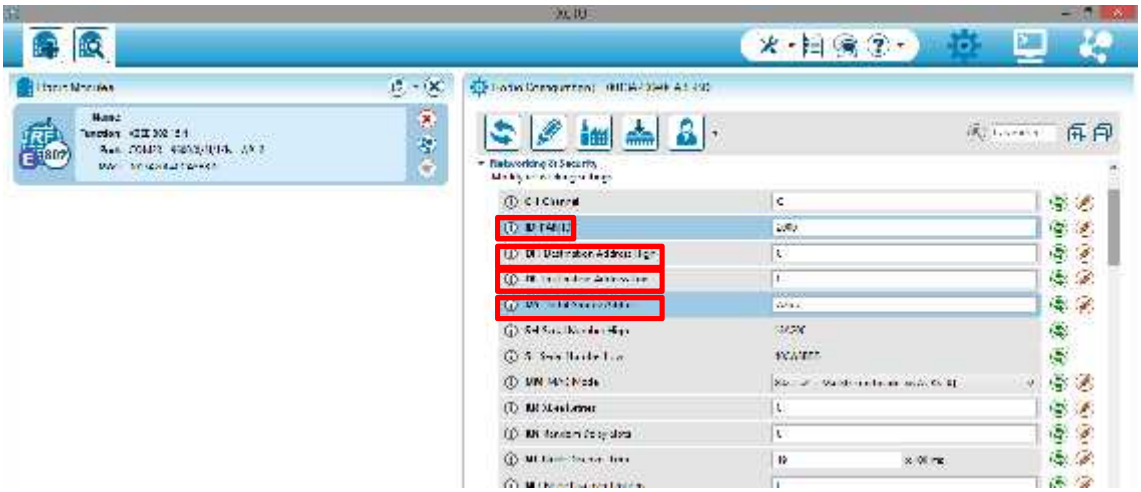


Figura 38-2: Cambio de Parámetros Xbee Parte 1

Realizado por: Albuja J., Yépez J, 2016.

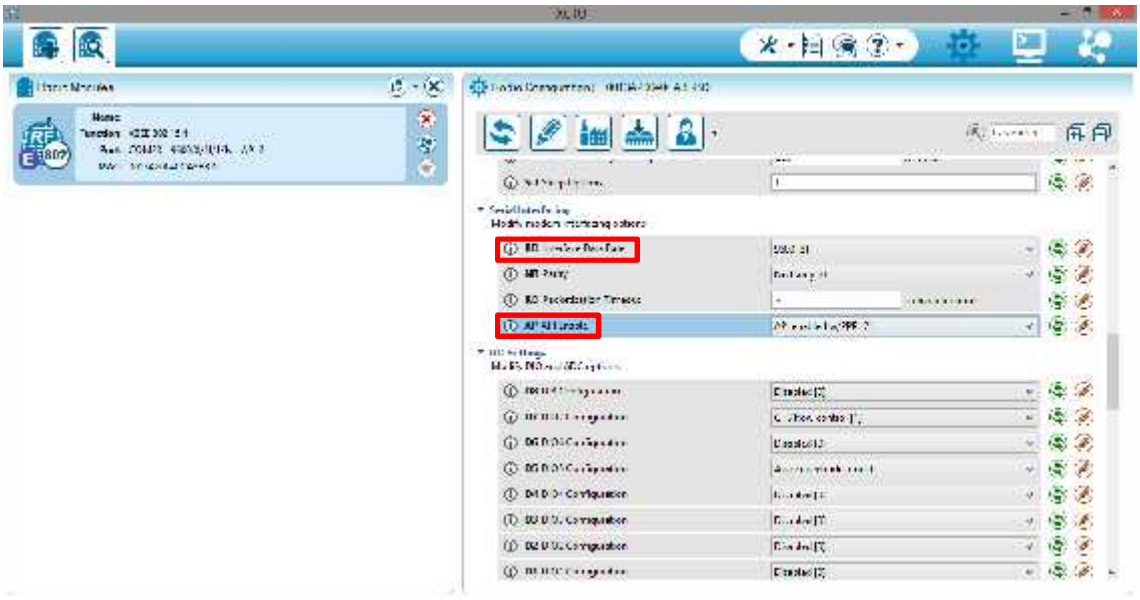


Figura 39-2: Cambio de Parámetros Xbee Parte 2

Realizado por: Albuja J., Yépez J, 2016.

Las partes señaladas anteriormente son cambiadas por los siguientes motivos:

- ID PAN ID: Es el identificador que nos permite identificar la red de área personal en la cual se pueda enviar un mensaje puede ir de 0 a 0xFFFF.

- DH Destination Address High: Este parámetro muestra los últimos 32 bits más significativos de la dirección de destino de 64 bits, está relacionada con DL para establecer la dirección desino. Para poder utilizarlo direcciones de 16 bits DH debe ser cero y DL menor a 0xFFFF.

- DL Destination Address Low: Este parámetro muestra los últimos 32 bits menos significativos de los 64 bits de la dirección destino, está relacionada con DH para establecer la dirección destino. MY 16-bit Source Address:

- BD Interface Data Base: Aquí nos muestra los valores que pueden ser seleccionados valiéndose de la tabla 17-2.

- MY 16-bits Source Address: Se puede establecer una dirección de origen de 16 bits.

Tabla 17-2: Valores para Interfaz Xbee

Valor	Configuración (bps)
0	1200
1	2400
2	4800
3	9600
4	19200
5	38400
6	57600
7	115200

Fuente: Tecnura, 2015

Realizado por: Albuja J., Yépez J, 2016.

- AP API Enable: Nos permite establecer el modo de operación de los módulos Xbee, para ver los diferentes tipos nos basaremos en la tabla 18-2.

Tabla 18-2: Valores: Modo de Operación Xbee

Valor	Modo de operación
0	Modo Transparente
1	Modo API habilitado
2	Modo API habilitado (con escape de caracteres)

Fuente: Tecnura, 2015

Realizado por: Albuja J., Yépez J, 2016.

2.8.2.4 Configuración Final Módulos Xbee

Tabla 19-2: Configuración Final de los Módulos Xbee

Parámetros	Xbee 1 Red 1	Xbee2 Red 1	Xbee 1 Red 2	Xbee 2 Red 2
PAN ID	2000	2000	1000	100
DH	0	0	0	0
DL	0	0	0	0
BD	9600	9600	9600	9600
MY	A35D	0	ABCD	0
API	2	2	2	2

Realizado por: Albuja J., Yépez J, 2016.

2.8.3 Programación en Java – NetBeans

Las aplicaciones y controladores encargados de proveer el control y tratamiento de la información desde el computador, están desarrolladas en el lenguaje de programación Java, empleando el entorno de desarrollo NetBeans, para la creación y compilación de la etapa de software del presente proyecto.

2.8.3.1 Instalación del Software NetBeans

Previa a la instalación de NetBeans, necesitamos descargar el Kit de Desarrollo de Java (JDK), necesario para poder crear nuestras aplicaciones y programas en Java, el mismo que se lo puede encontrar en http://www.oracle.com/.

Una vez descargados los recursos de instalación tanto de NetBeans como el JDK, el proceso de instalación es sumamente sencillo, iniciando la instalación del JDK que obtuvimos de Oracle, en este caso la versión JDK7, esto permite que el paquete de instalación de NetBeans pueda instalar todas las dependencias de Java que necesita en el ordenador, para hacer posible la creación de proyectos Java.

Una vez ejecutado el paquete de instalación de NetBeans (Figura 40-2), solo se debe seguir las instrucciones en las ventanas emergentes hasta culminar el proceso, con lo que el entorno de desarrollo estará listo para trabajar.

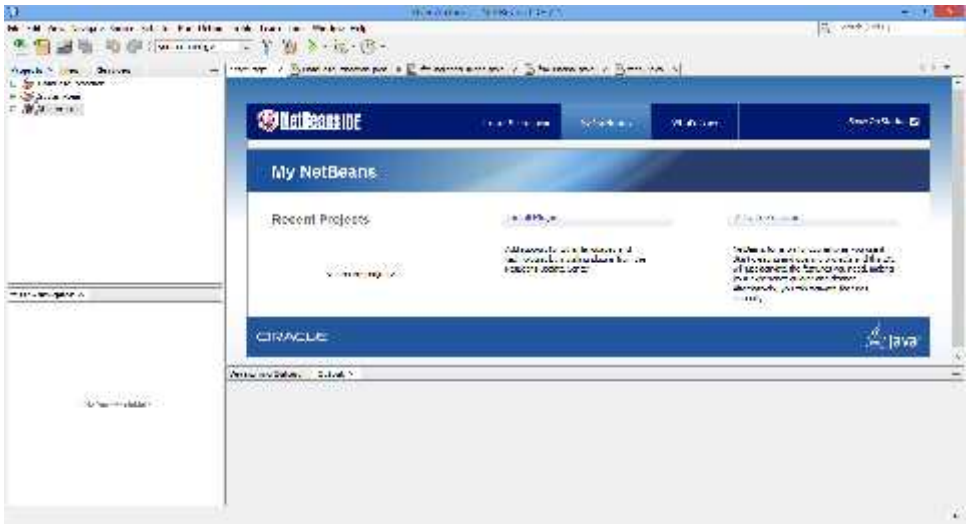


Figura 40-2: Ventana de inicio de NetBeans

Realizado por: Albuja J., Yépez J, 2016.

2.8.3.2 Creación de proyectos en NetBeans

La etapa de Software del Sistema Multimodal de Control de Asistencia descrito en el apartado de diseño del presente informe está definido por cuatro módulos de los cuales tres están concebidos bajo el lenguaje de programación Java, el driver o controlador de los dispositivos Xbee, la conexión a la base de datos y la interfaz de usuario.

Para la creación de cualquier tipo de aplicación en el entorno de desarrollo integrado NetBeans se sigue un procedimiento estándar. Se crea un nuevo proyecto desde el menú Archivo en la barra de herramientas (Figura 41-2).

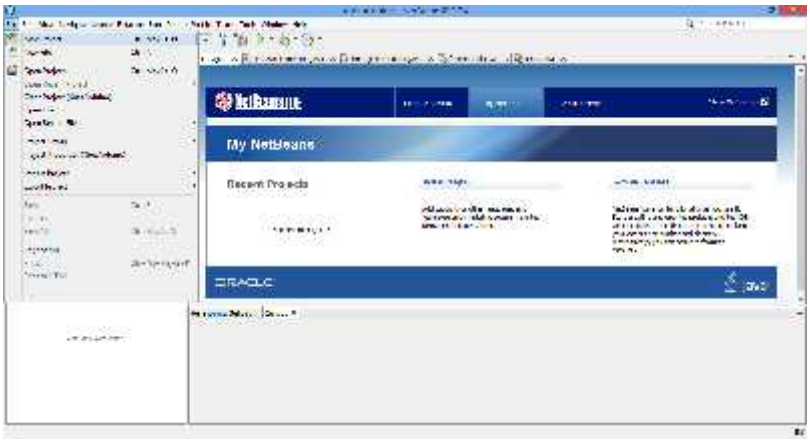


Figura 41-2: Creación de Nuevo Proyecto en NetBeans

Realizado por: Albuja J., Yépez J, 2016.

Al crear un nuevo proyecto en NetBeans (Figura 42-2) se despliega una ventana emergente en la que se debe definir el tipo de proyecto que vamos a crear, en este caso una aplicación java:

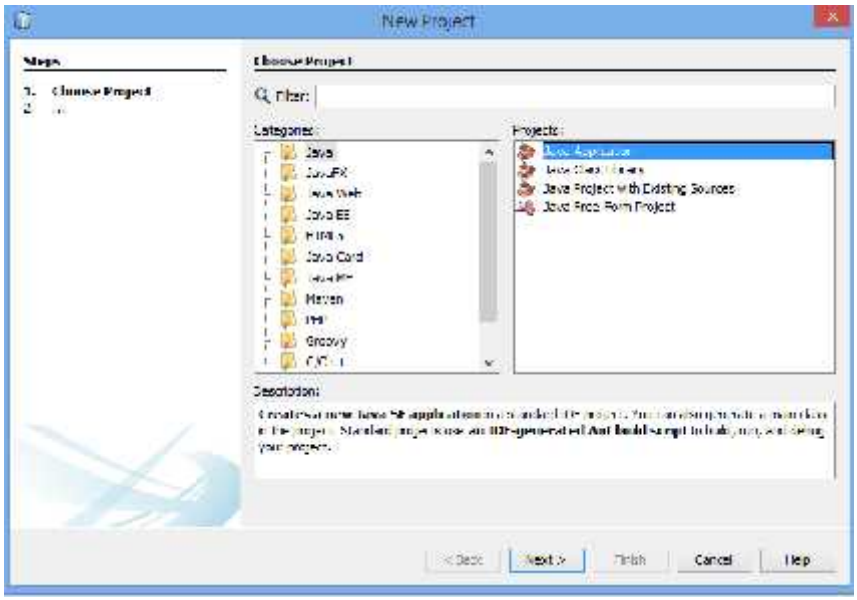


Figura 42-2: Tipo de Proyecto a Crear

Realizado por: Albuja J., Yépez J, 2016.

A continuación se define el nombre del proyecto y la ubicación de donde se almacenarán los archivos del mismo (Figura 43-2), y termina el proceso dando click en finalizar:

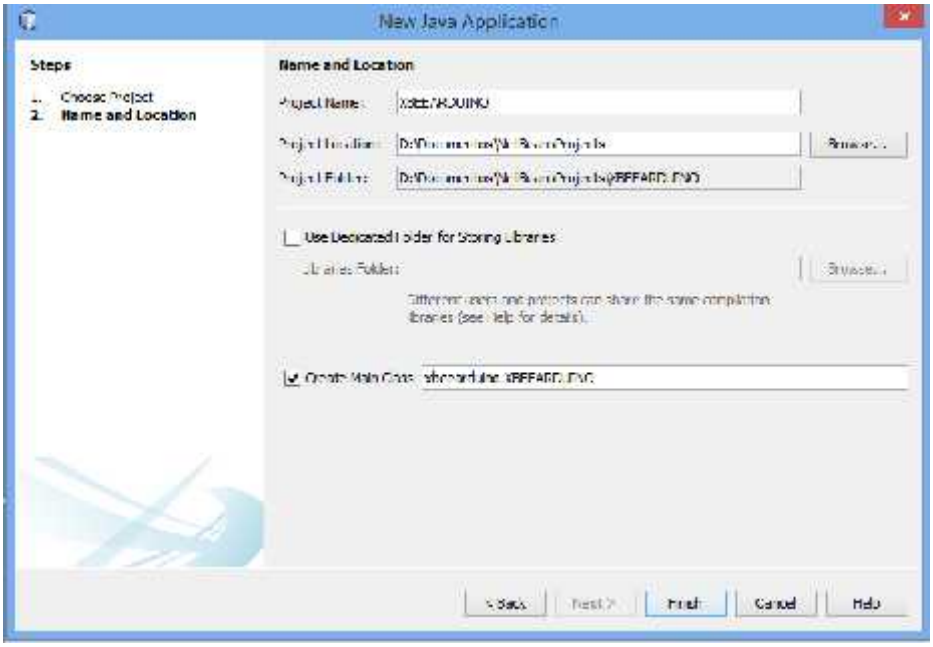


Figura 43-2: Nombre Proyecto y Ubicación

Realizado por: Albuja J., Yépez J, 2016.

Una vez creado el proyecto, este se visualiza en la paleta de proyectos en la parte superior izquierda de la ventana inicial de NetBeans (Figura 44-2), este se despliega en forma de árbol jerárquico con un paquete java y su respectiva clase principal del mismo nombre del proyecto.

Figura 44-2: Proyecto creado en NetBeans

Realizado por: Albuja J., Yépez J, 2016.

Aquí es donde el proyecto es personalizado y adaptado a las necesidades de las etapas definidas en el apartado de diseño del presente informe, teniendo notables variaciones de configuración en la clase principal, y dependiendo del caso el aumento de clases secundarias, ventanas de interfaz o paquetes de programa para cada módulo del sistema desarrollado en java.

2.8.3.3 Configuración Driver Xbee-API-Mode

Para el desarrollo del Sistema Multimodal de Control de Asistencia, se empleó los módulos de comunicación inalámbrica Xbee en configuración API2, razón por la cual fue necesaria la creación de un driver basado en java que hiciera posible la obtención de información provenientes del canal inalámbrico en el ordenador que alojará la base de datos.

La programación del driver se lo realizó en el entorno de desarrollo NetBeans, se creó un proyecto llamado XbeeArduino, en cuya clase principal se programó el código que permite la interacción del dispositivo Xbee con el ordenador.

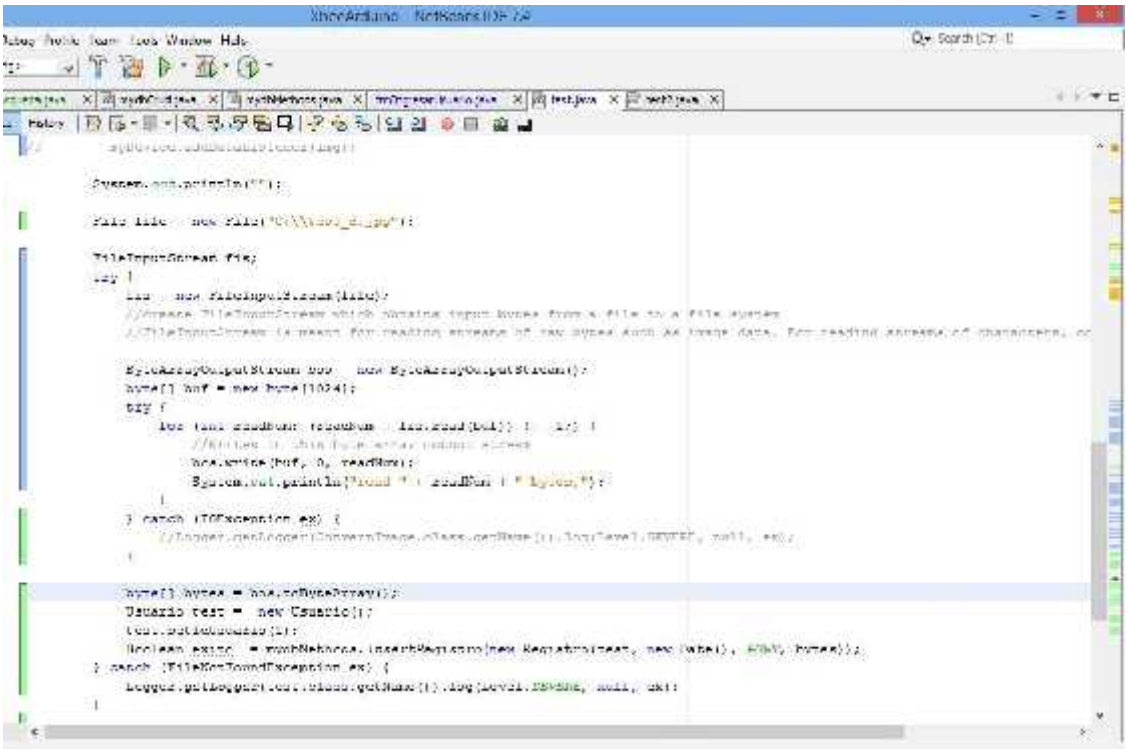


Figura 45-2: API Figura 1

Realizado por: Albuja J., Yépez J, 2016.

Para lograr la programación requerida de dicho driver fue necesario la utilización de una librería propietaria de Digi International Inc., la misma que provee las funciones necesarias para la comunicación de los módulos Xbee con las aplicaciones en Java.

Dicha librería está disponible junto a un manual de uso en la página de Digi International, específicamente en la dirección Web que se muestra a continuación: https://docs.digi.com/display/XBJLIB/Getting+started+with+XBee+Java+Library.

La estructura de programa del driver comprende tres partes fundamentales: La importación de librerías, las librerías empleadas para las aplicaciones y proyectos en Java se presentan como contenedores tipo .jar que en el caso de la librería de Digi International contiene los paquetes de funciones empleadas por el driver para permitir e funcionamiento entre los módulos Xbee con el ordenador (Figura 46-2).

Figura 46-2: Librerías Importadas de Digi International

Realizado por: Albuja J., Yépez J, 2016.

Definición de Puertos y Velocidad de transmisión, para reconocer el módulo Xbee como un nuevo dispositivo dentro de los recursos del sistema del computador, lo enlazamos a uno de los puertos seriales del mismo, especificando el número de puerto y la velocidad en la que se efectuará la transmisión de datos (Figura 47-2).

```
 */
private static final String PORT = "COM6";
// TODO Replace with the baud rate of your receiver module.
private static final int BAUD_RATE = 9600;
```

Figura 47-2: Definición de Puerto y Velocidad de transmisión

Realizado por: Albuja J., Yépez J, 2016.

Tratamiento de la Información, una vez logrado el funcionamiento de los módulos con el ordenador y establecida la comunicación entre ellos, las tramas recibidas son recogidas y des encapsuladas empleando las funciones que nos proveen las librerías importadas.

2.8.3.4 Conexión a la Base de Datos

Para implementar la conexión a una base de datos en el entorno de desarrollo NetBeans es necesario incluir los servicios de la base de datos en el entorno de desarrollo para poder emplearlos en las aplicaciones Java encargadas de establecer la conexión, para lo cual se crea una nueva conexión de base de datos en la paleta de servicios del IDE (Figura 48-2).

Figura 48-2: Establecer conexión con la Base de Datos

Realizado por: Albuja J., Yépez J, 2016.

En la venta emergente (Figura 49-2) se escoge el motor de base de datos en la que se trabaja, en este caso MySQL, para que NetBeans incluya las librerías para manejar las funciones de la base de datos.

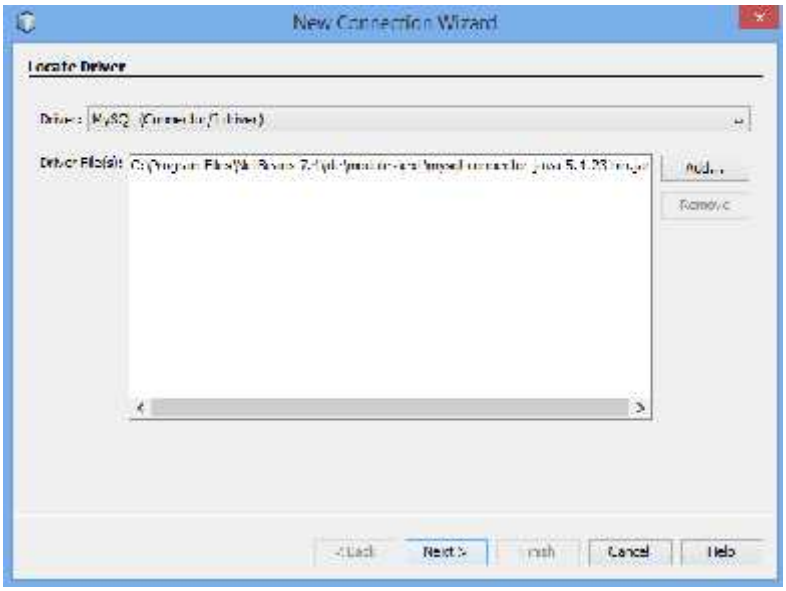


Figura 49-2: Motor de la Base de Datos

Realizado por: Albuja J., Yépez J, 2016.

Posteriormente se define el servidor del servicio, el nombre de la base de datos, el puerto de conexión, y una contraseña, para finalmente comprobar la conexión haciendo click en Test Connection y en finalizar para terminar con la configuración (Figura 50-2).

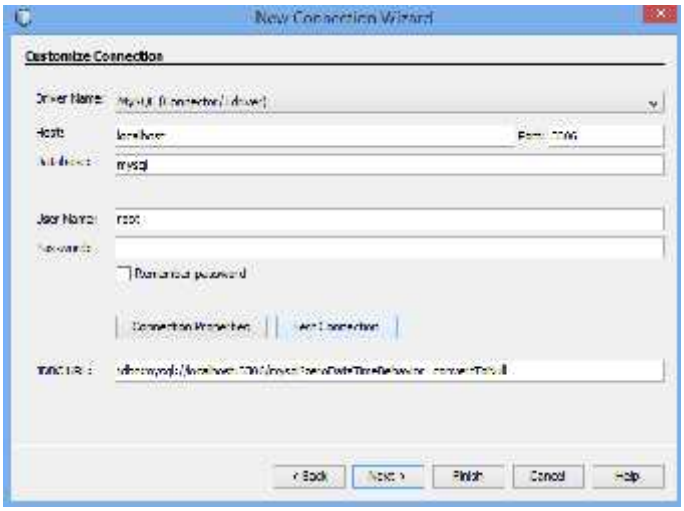


Figura 50-2: Personalización de la conexión con la Base de Datos

Realizado por: Albuja J., Yépez J, 2016.

La administración de la base de datos debe efectuarse a través de la interfaz de usuario, para esto se implementó una capa de programación de persistencia que es la encargada de pasar la información entre el front y la base de datos por medio de métodos CRUD (create, read, update, delete) que proveen las librerías importadas para este efecto.

Esta capa de persistencia está dividida en dos clases llamadas mydbMethods y mydbCRUD, siendo la primera encargada de la verificación de los campos de la base de datos y llamada a las funciones contenidas en mydbCRUD, la cual a su vez contienen las operaciones CRUD (Figura 51-2).

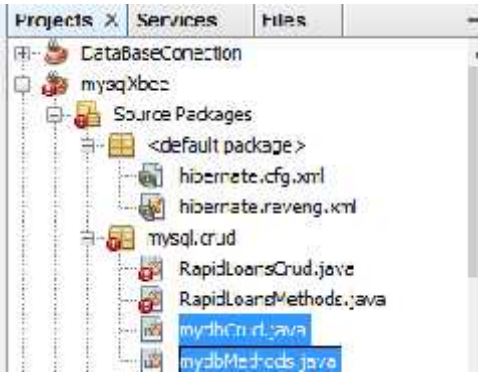


Figura 51-2: Capa de Persistencia.

Realizado por: Albuja J., Yépez J, 2016.

En la estructura de programa de las clases antes mencionadas se destaca la creación de las funciones en el cuerpo del código por cada operación de inserción, búsqueda, actualización o

borrado de la información en la base de datos, según el diseño del Front de la interfaz de usuario. El código de la conexión a la base de datos se encuentra en el Anexo D.

Nota: Debido a la extensión de la programación se cita una parte de la misma en el Anexo D

.

2.8.3.5 Interfaz de Usuario

La interfaz de usuario provee las herramientas para administrar la información de la base de datos, las mismas que están destinadas para el uso del encargado del control de asistencia, haciendo transparente el funcionamiento de los dispositivos de hardware y el procesamiento de información en el ordenador para el usuario.

El Front del sistema (Figura 52-2) se compone de una clase principal desde la cual se llaman y despliegan las clases secundarias accesibles desde una barra de menús en la parte superior de la ventana principal. La ventana principal alojará las ventanas emergentes secundarias dentro del entorno dispuesto por la misma.

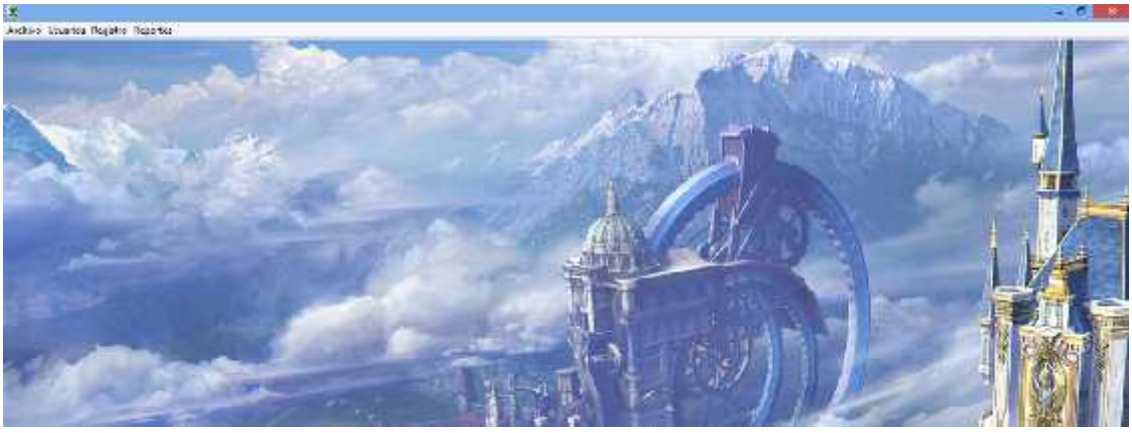


Figura 52-2: Interfaz del Sistema – Front

Realizado por: Albuja J., Yépez J, 2016

En la barra de menús se encuentran los accesos a las opciones de usuario, registros y reportes (Figura 53-2). Entre las opciones de usuario se incluye: ingresar, para el ingreso de usuarios en la base de datos; buscar, para realizar una búsqueda de un usuario determinado y lista, que muestra la información básica de todos los usuarios.

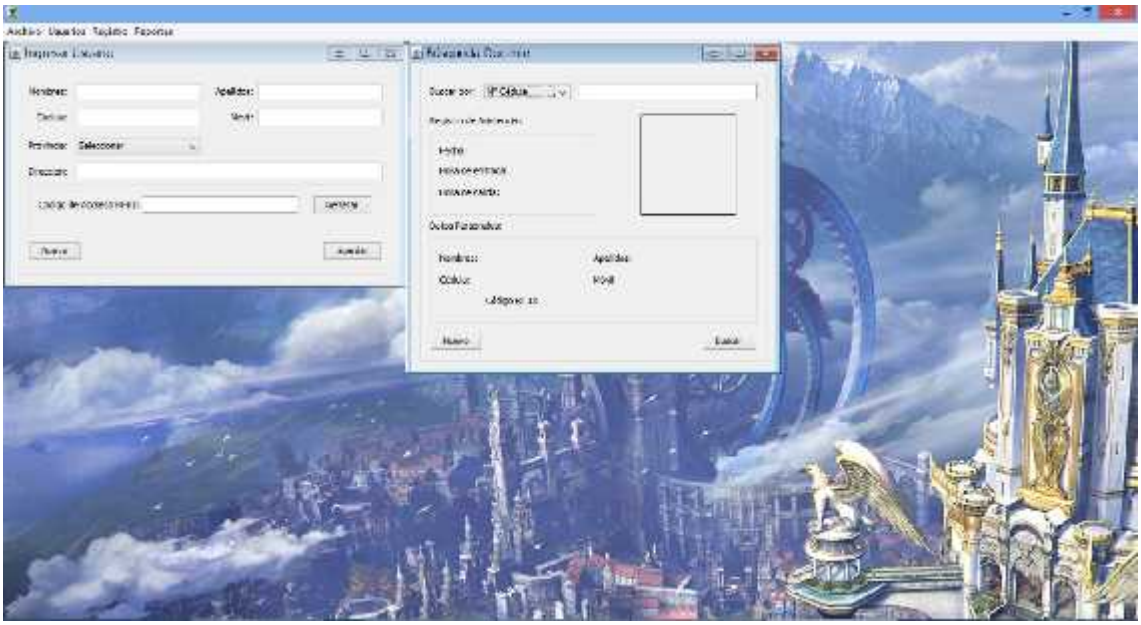


Figura 53-2: Acceso a las opciones de Usuarios

Realizado por: Albuja J., Yépez J, 2016.

Las opciones de registros y reportes muestran la información del registro de los usuarios y genera los reportes de asistencia de todos los usuarios del sistema (Figura 54-2).

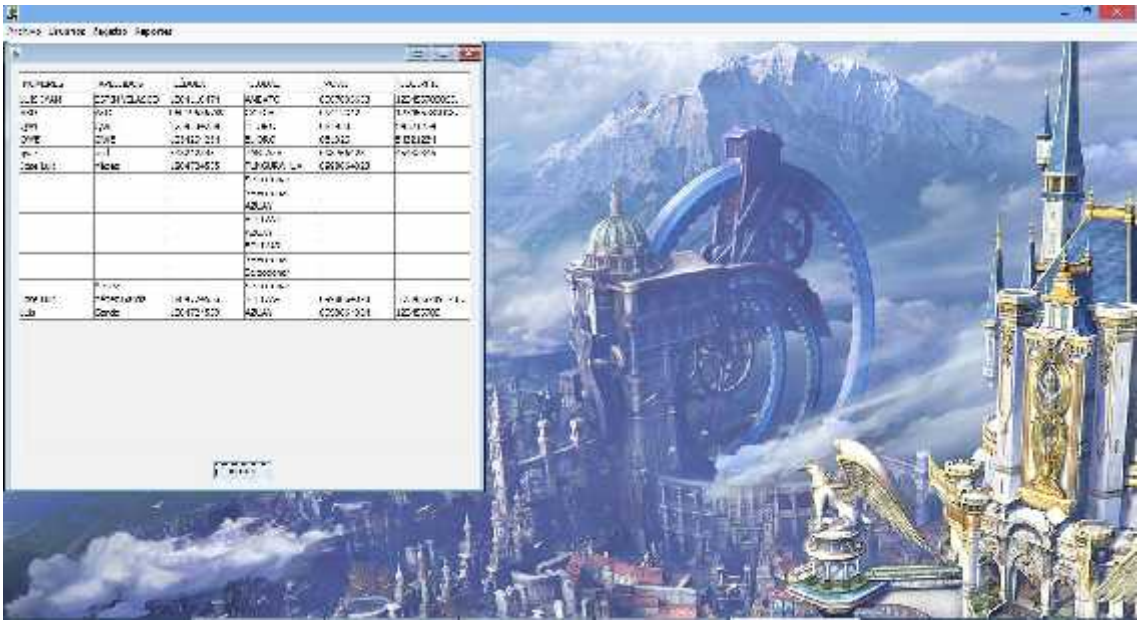


Figura 54-2: Opciones de Registro y reportes

Realizado por: Albuja J., Yépez J, 2016.

CAPÍTULO III

3 MARCO DE RESULTADOS Y ANÁLISIS DE RESULTADOS

Para determinar los resultados del presente trabajo fue necesaria la implementación del prototipo del sistema multimodal, para ello constaremos mediantes imágenes cada etapa necesaria para establecer el sistema, las cuales nos permitirán cumplir con nuestro objetivo principal y poseer un sistema multimodal de control de asistencia de personal en IPREX.

3.1 Diagrama de Bloques del funcionamiento del sistema

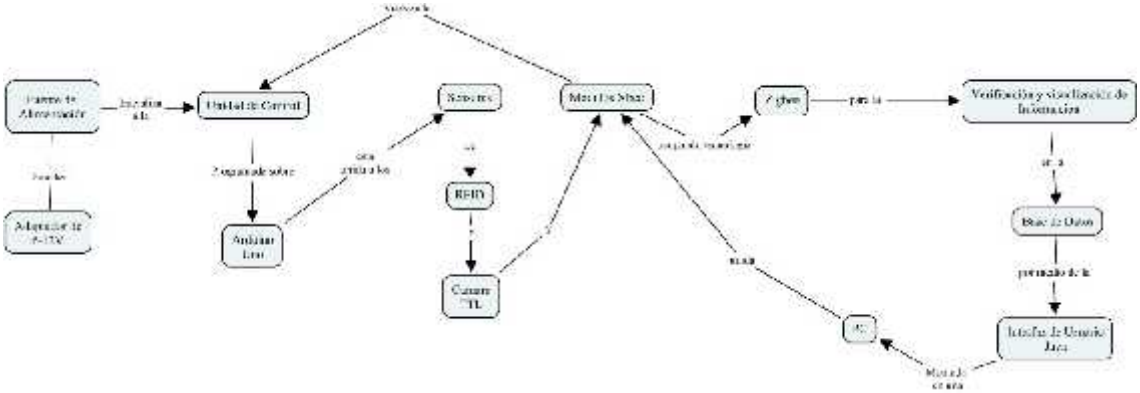


Figura 1-3: Diagrama de Bloques para el funcionamiento del sistema

Realizado por: Albuja J., Yépez J, 2016.

3.1.1 Inicialización de la Unidad de Control Arduino

Para iniciar la unidad de control se debe conectar el adaptador de 9V a una toma corriente (Figura 2-3) en el plug de nuestra placa reducida Arduino, el Arduino demorara unos 15 segundos hasta inicializarse y cargará el problema precargado en su memoria, cuando el Arduino esté listo observaremos que el Arduino se encuentra con su led verde encendido (Figura 3-3).

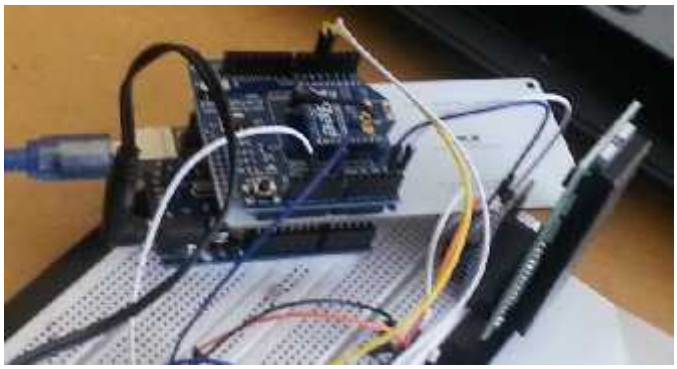


Figura 2-3: Conexión adaptador para Arduino

Realizado por: Albuja J., Yépez J, 2016.

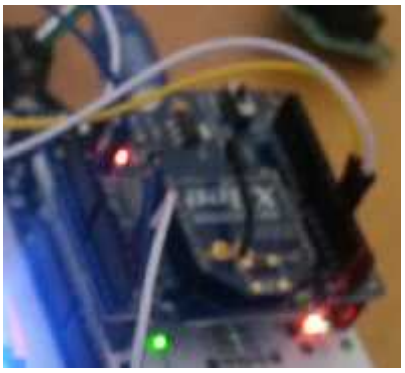


Figura 3- 3: Arduino Uno inicializado

Realizado por: Albuja J., Yépez J, 2016.

3.1.2 Reconocimiento de los sensores

Para ver que los dispositivos han sido detectados correctamente en nuestra programación hemos puesto sentencias que son visualizadas en nuestro LCD si los sensores son o no encontrados, obteniendo las siguientes imágenes (Figura 4-3 y Figura 5-3) al momento de la inicialización de la programación.

Figura 4-3: Inicialización de Sensor RFID PN 532

Realizado por: Albuja J., Yépez J, 2016.

Figura 5-3: Inicialización de cámara TTL

Realizado por: Albuja J., Yépez J, 2016.

3.1.3 Reconociendo de los dispositivos Xbee para establecimiento del canal

Para verificar esta etapa nos apoyaremos en la aplicación X-CTU y en nuestro sistema, con la cual se enviara información de prueba para comprobar que el canal ha sido creado mediante nuestros parámetros configurados en la tabla 19-2 esto se ve en las figuras 6-3 y 7-3.

Figura 6-3: Envió de Paquetes de Prueba

Realizado por: Albuja J., Yépez J, 2016.

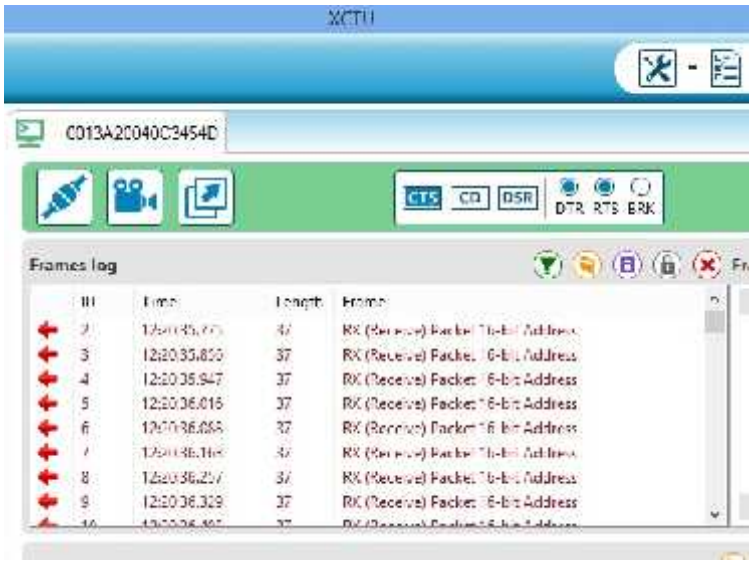


Figura 7-3: Recepción de paquetes de Prueba

Realizado por: Albuja J., Yépez J, 2016.

En nuestra programación se encuentra una sentencia que nos alerta y visualiza en el LCD que el módulo Xbee no fue reconocido o no se encuentra montado junto a nuestro Arduino Uno.

3.1.4 Autenticación de los usuarios con tarjetas RFID Mifare Classic

Para esto nuestras tarjetas deben ser formateadas con un formato llamado NDEF con el cual insertamos el código dado en la base de datos en la tarjeta, sin este tipo especial de formato la tarjeta no podrá ser reconocida, además nos brinda seguridad, debido a que ningún agente externo va a poder utilizar el sistema consumiendo recursos sin poseer un TAG RFID modificado y autenticado. Para conocer cómo realizar el formateo de tarjetas, la guía se encuentra explicada en el ANEXO B.

El sistema funciona de la siguiente manera el lector RFID reconoce al TAG RFID formateado, lo lee, coge el código (Figura 8-3), posterior toma la foto, almacena la imagen en la cámara TTL en su buffer interno (Figura 9-3), envía la imagen y espera la autenticación de la base de datos, si la base de datos devuelve una verificación positiva se actualiza el registro, caso contrario no tiene autorización y el sistema se reinicia para el comenzar de nuevo con la autenticación desde un principio (Figura 10-3).

Figura 8-3: Lectura TAG RFID en formato NDEF

Realizado por: Albuja J., Yépez J, 2016.

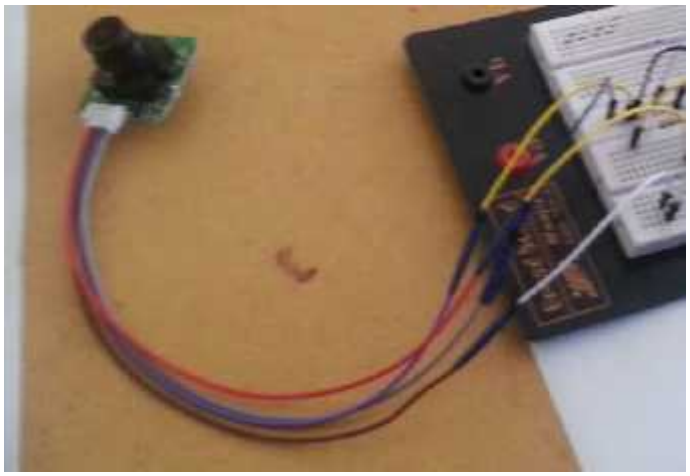


Figura 9-3: Captura de Imagen y almacenamiento temporal

Realizado por: Albuja J., Yépez J, 2016.

Figura 10-3: Reinicio de Sistema a fase inicial

Realizado por: Albuja J., Yépez J, 2016.

3.1.5 Acceso a la base de datos

En esta etapa vamos a ingresar directamente a la Base de Datos para verificar que las tablas usuario y registro tengan campos válidos con la información dada y recolectada, enviada por la unidad de control Arduino de los sensores conectados a ella para ver la hora de entrada, salida y registro del personal (Figura 11-3).Además veremos la información proporcionada por la interfaz de la usuario.

Figura 11-3: Tabla usuario y registro

Realizado por: Albuja J., Yépez J, 2016.

3.1.6 Pruebas de funcionamiento de la Interfaz de usuario

La interfaz de usuario fue desarrollada en java (Figura 52-2) en ella se puede tener acceso a la base de datos para la administración, a continuación vamos a comprobar que los datos ingresados de cada usuario son actualizados en nuestra base de datos, posterior a ello su verificación mediante las tablas registros y reporte.

3.1.6.1 Menú Principal

En primera instancia tenemos el campo Archivo en el cual solo poseemos la opción para salir del programa. (Figura 12-3).

Figura 12-3: Posibilidades de Archivo

Realizado por: Albuja J., Yépez J, 2016.

En segunda instancia poseemos el campo usuario en el tenemos las posibilidades mostradas en la figura 13-3, desplegaremos la opción Ingresar Usuario y llenaremos los campos requeridos para actualizar la base de datos con relación a ese usuario como se ve en la figura 14-3.

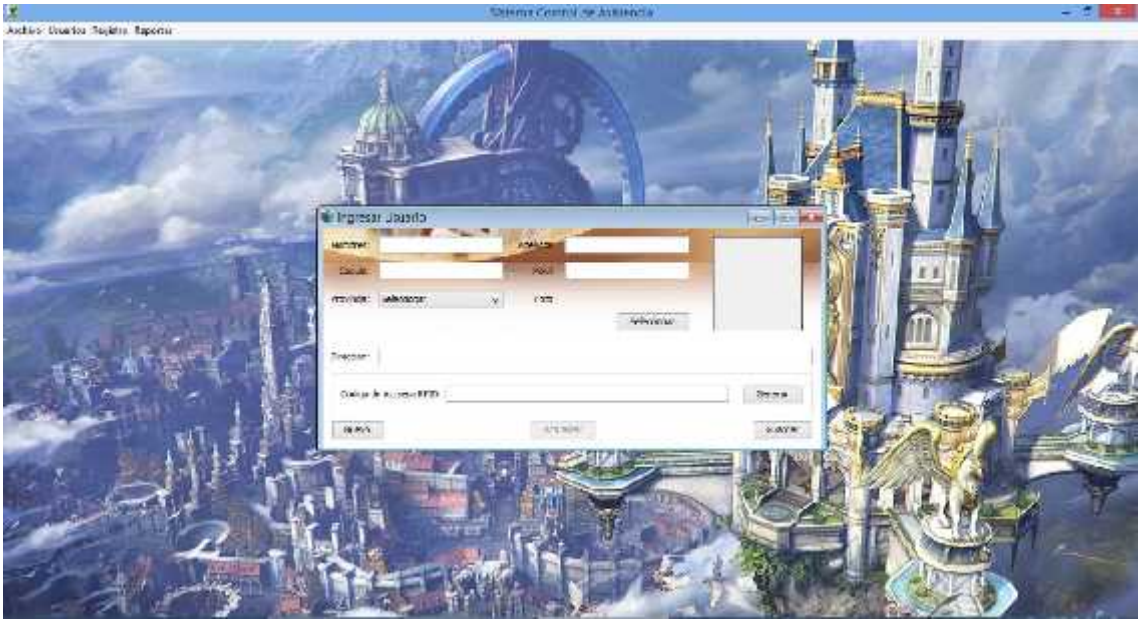


Figura 13-3: Posibilidades dentro de campo Usuario

Realizado por: Albuja J., Yépez J, 2016.

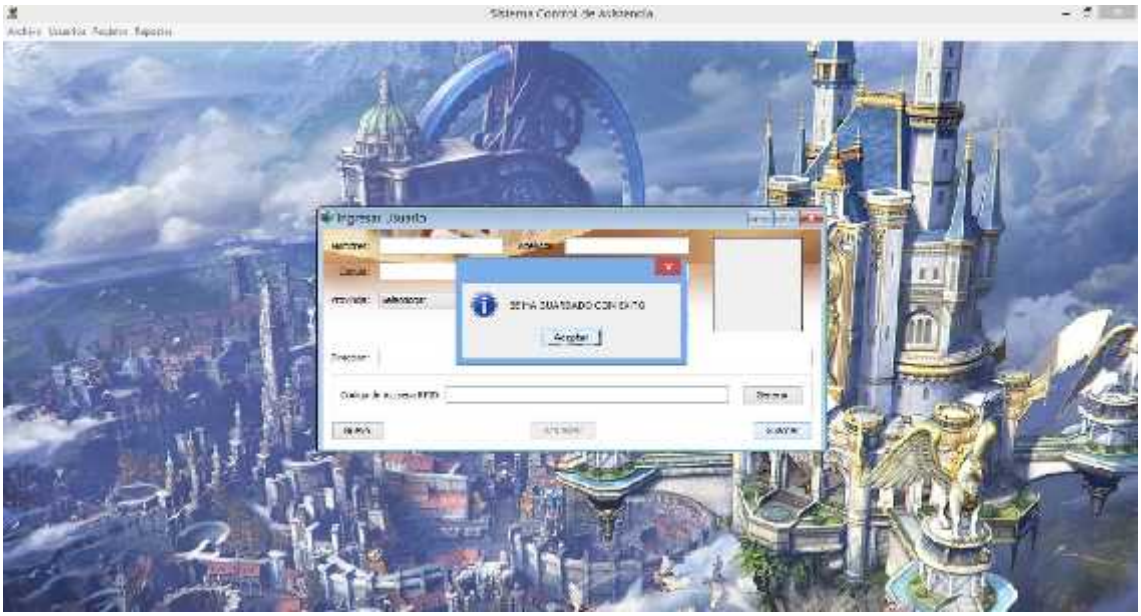


Figura 14-3: Ingreso de Usuario a la base de Datos

Realizado por: Albuja J., Yépez J, 2016.

En tercera instancia tenemos nuestra opción Registro en la cual se puede visualizar los registro de usuario dependiendo de su cedula o código único RFID logrando constatar su hora en entrada y salida, como lo vemos en la figura 15-3.

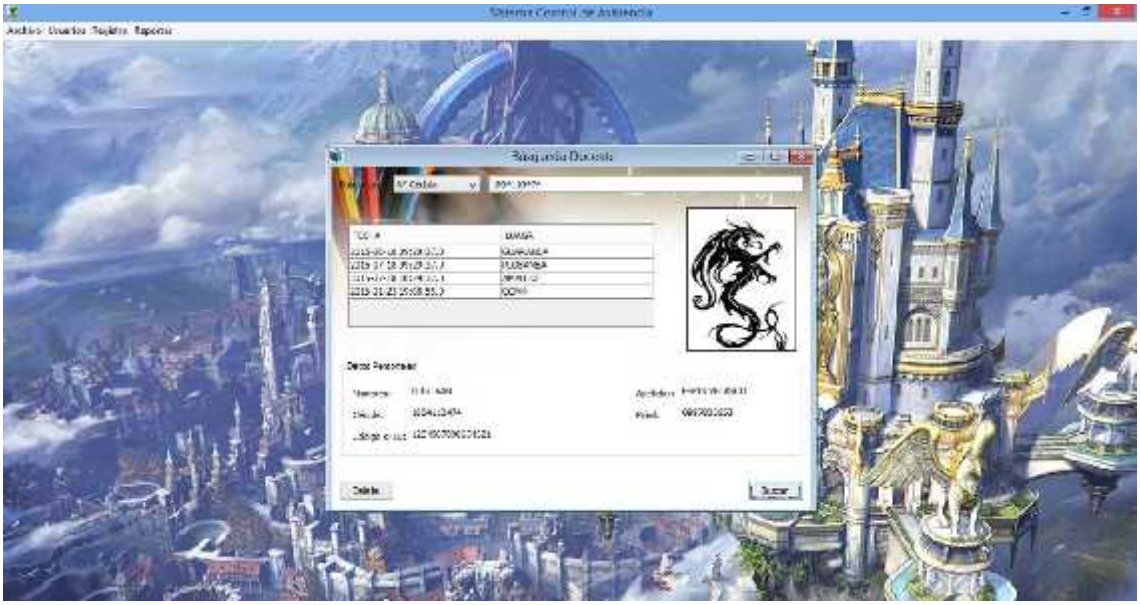


Figura 15-3: Consulta de Registros por Usuario

Realizado por: Albuja J., Yépez J, 2016.

Como última instancia tenemos las opciones de Registros y Reportes, poseen los subíndices de generar un reporte de todos los registros por usuario y por fecha (Figura 16-3) y un reporte de la información de los usuarios registrados. Seleccionaremos la opción para visualizar un reporte

completo de los usuarios, la cual nos brindará una lista con los datos e imagen de todos los usuarios (Figura 17-3).

Figura 16-3: Índice de la opción Reportes

Realizado por: Albuja J., Yépez J, 2016.

Figura 17-3: Reporte completo de Usuarios

Realizado por: Albuja J., Yépez J, 2016.

3.2 Análisis de Resultados

Para establecer que el sistema obtenido es una buena alternativa para el control de asistencia de personal, vamos a compararlo con el método actual de registro manual de personal establecido en la empresa, y veremos el presupuesto de nuestro sistema multimodal.

3.2.1 Análisis y comparación con el método actual de registro de personal

Para establecer la relación correspondiente tomaremos en cuenta la tabla 1-3 en la cual nos mostrara el tiempo estimado que una persona demora al realizar su registro de asistencia manualmente, en relación al utilizar el sistema multimodal.

Tabla 1-3: Tiempo estimado entre sistemas

Cantidad de Personas	Tiempo sistema manual (segundos)	Tiempo sistema multimodal (Segundos)
Una	60	25
Dos	120	50
Tres	180	75
Cuatro	240	100
Cinco	300	125

Realizado por: Albuja J., Yépez J, 2016.

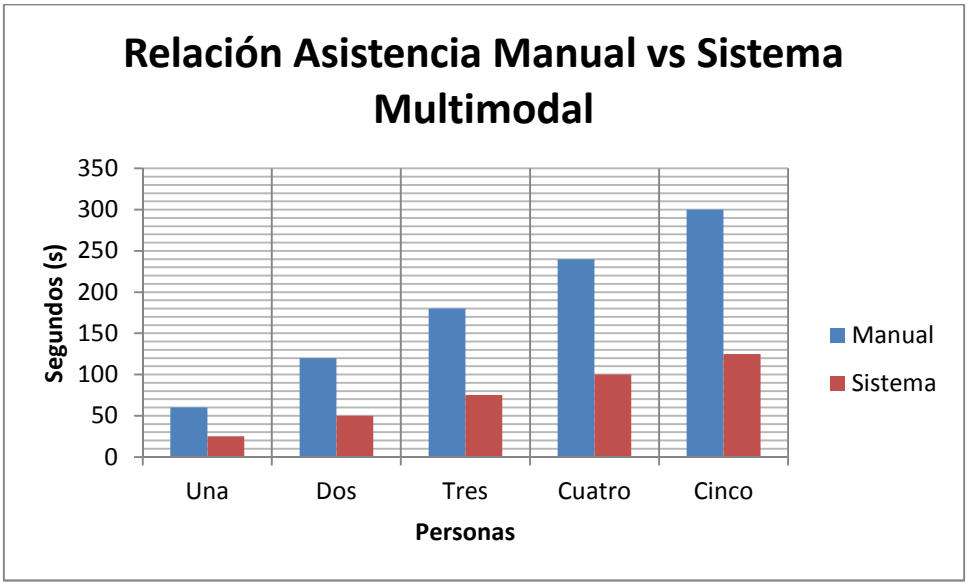


Figura 18-3: Relación Asistencia Manual vs Sistema Multimodal

Realizado por: Albuja J., Yépez J, 2016.

Como se puede apreciar en la figura 18-3, nuestro sistema posee una mejor relación en comparación con el sistema manual optado para el control de asistencia, para ver la mejora que existencia entre los sistemas, para ello vamos a calcular un valor denominado Δs que es la diferencia entre el sistema manual y el multimodal para ver el porcentaje de mejora del sistema implementado.

$$\Delta s = (60 - 25) s = 35\ segundos$$

$$\Delta s = 58{,}33\% \quad efectividad$$

3.2.2 Análisis del Presupuesto para la implementación del sistema

Tabla 2-3: Presupuesto del Proyecto

Sistema Multimodal		
Componente	Precio Unidad por Sistema ($)	Precio en el sistema
Arduino Uno	20	60
RFID PN532	50	100
Xbee Shield	15	15
Cámara TTL	70	70
LCD	6	6
I2C LCD	4,50	4,50
Cables	10	10
Fuente de Alimentación	20	20
Xbee Series 1	90	180
Xbee explorer	10	20
Diseño del Chasis	40	40
Chasis del Proyecto	100	100
Resistencias de 10kΩ	0,05	0,20
Total	435,55	625,70

Realizado por: Albuja J., Yépez J, 2016.

CONCLUSIONES

- Se concluye que mediante la fusión de la tecnología RFID, protocolo ZigBee y cámara TTL, se logró obtener un sistema multimodal que nos sirva para el control de asistencia en IPREX.

- El análisis realizado en primera instancia nos permitió determinar que el protocolo de comunicación inalámbrica ZigBee posee mejores características para satisfacer los requerimientos de nuestro sistema, dándonos ventajas como ahorro de energía, fácil utilización y funcionamiento en una banda de libre distribución de 2.4 GHz.

- La comparación realizada de las características de las diversas placas distribuidas por Arduino nos permitió determinar que la placa reducida Arduino Uno es óptima para ser utilizado en la implementación, sin desperdiciar recursos como memoria y pines de entrada y salida.

- Se implementó un sistema multimodal funcional en tiempo real, que nos permite recibir y tratar toda la información relacionada con el personal de IPREX dependiendo de los factores de búsqueda del administrador.

- En la evaluación de la red se puedo constatar que cada parte del sistema cumplía con su función principal, obteniendo como resultado una técnica que mejoró el control de asistencia en un 58,33%, al poseer una mayor velocidad para el tratamiento de información, además de poseer una fácil interfaz para la búsqueda de registros.

RECOMENDACIONES

- El sistema puede utilizarse para la creación de una aplicación en línea para notificar el ingreso de los trabajadores.

- Puede utilizarse como preámbulo para la elaboración de un sistema que reemplace los módulos ZigBee por Wifi, Bluetooth o un medio físico.

- El sistema puede utilizarse para la elaboración de un sistema que pueda controlar la temperatura de un invernadero, ya que los códigos dados por el RFID, serían reemplazados por las señales dadas por los sensores.

- El sistema puede ser modificado para que la base de datos sea desarrollada en un ordenador de placa reducida como Raspberry.

- El sistema puede utilizarse para el control de ingreso a sitios restringidos y/o prohibidos, cambiando la tecnología RFID por un reconocimiento biométrico o por iris.

BIBLIOGRAFÍA

1. **ARCHUNDIA, F.** *Wireless Personal Area Network (WPAN) & Home Networking***,** [En línea]**.** Capítulo 4, Puebla-México. 2003. [Consulta: 08 de Octubre de 2015.]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/archundia_p_fm/capitulo4.pdf

2. **BARNEDA, I.** *ZigBee Aplicado a la Transmisión de datos de Sensores Biométricos*, [En línea]. Barcelona-España: Universidad Autónoma de Barcelona, 2008. [Consulta: 10 de Octubre de 2015.]. Disponible en: http://www.recercat.cat/bitstream/handle/2072/13081/PFC%20Ivan%20Barneda.pdf?sequence=1

3. **BIT Boletín Informativo TEC** *El "ABC" de la Banda Magnética*. [En línea]. 2003. [Consulta: 15 de Octubre de 2015]. Disponible en: https://www.tec-mex.com.mx/promos/bit/bit0703-msr.htm

4. **BORJA, C. & BUENO, A.** *Sistemas Biométricos. [En línea]*. [Consulta: 28 de Noviembre de 2015]. Disponible en: https://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Documentacion/Trabajos/Biometria/Trabajo%20Biometria.pdf

5. **CLUB DE INFORMÁTICA, ROBÓTICA Y ELECTRÓNICA.** *Comenzando con ZigBee.* Madrid-España, [En línea]. 2012. [Consulta: 22 de Diciembre de 2015.]. Disponible en: http://webdelcire.com/wordpress/archives/1714

6. **EVERETT, D.** *Smart Card Tutorial***.** [En línea]. 1992. [Consulta: 20 de Octubre de 2015.]. Disponible en: http://www.smartcard.co.uk/tutorials/sct-itsc.pdf

7. **JACOBSEN, C. & JADUD, M. & KILIC, O. & SAMPSON, A.,** *Concurrent Event-driven Programming in occam-π for the Arduino.* [En línea], 2011. [Consulta: 12 de Junio de 2015]. Disponible en: http://offog.org/publications/cpa2011-arduino.pdf

8. **LARA, J.** "*Diseño e Implementación de un Sistema Basado en la Tecnología RFID para el control de Inventario de la Empresa MILBOOTS"*. [En línea] (tesis pregrado***).*** Escuela Politécnica del Ejército, Departamento de Eléctrica y Electrónica, Sangolquí-Ecuador. 2008. pp.1-5. [Consulta: 02 de Noviembre de 2015]. Disponible en: http://repositorio.espe.edu.ec/bitstream/21000/594/1/T-ESPE-017565.pdf.

9. **LAZALDE, A. & TORRES, J. & VILA, D.,** *Hardware Libre Recomendaciones para el fomento de la innovación ciudadana,* Quito-Ecuador, [En línea], 2015. [Consulta: 14 de Junio de 2015]. Disponible en: http://floksociety.org/docs/Espanol/4/4.1.pdf

10. **LIBERA WHITEPAPER SERIES,** *RFID: Tecnología, Aplicaciones y Perspectivas,* Málaga-España, [En línea], 2010. [Consulta: 22 de Noviembre de 2015]. Disponible en: http://www.libera.net/uploads/documents/whitepaper_rfid.pdf.

11. **Loft Media Publishing,** *RFID TAG YEARBOOK THE FIRST RFID TAG CATALOGUE,* [En línea]. Milan-Italia, 2010. [Consulta: 26 de Septiembre de 2015]. Disponible en: http://www.tagingenieros.com/sites/default/files/RFID_TAG_YEARBOOK_09_2010.pdf

12. **MOLINA, C. & RODRIGUEZ, F. VERA, V. "***Diseño e Implementación de un Sistema de Control y Monitoreo de Acceso y Seguridad sobre una red Ethernet utilizando Tarjetas Smart Card"*. [En línea]. *(*Tesis pregrado). Escuela Superior Politécnica del Litoral, Facultad de Ingeniería en Electricidad y Computación, Guayaquil-Ecuador. 2007. [Consulta: 03 de Julio de 2015]. Disponible en: http://www.dspace.espol.edu.ec/handle/123456789/2996

13. **MORALES, G & ABAC, R.** *Sistemas Biométricos.* [En línea]. 2008. [Consulta: 13 de Octubre de 2015]. Disponible en: http://geovinmorales.siems.com.gt/umg/cursos/inteligencia/biometria.html

14. **PENARRLETA, A.** *Jaba Y NetBeans***.** [En línea]. 2011. [Consulta: 23 de Diciembre de 2015]. Disponible en: https://javaagricola.wikispaces.com/file/view/0_Java+y+NetBeans.pdf

15. **PUPIALES, P. "***Diseño de un Sistema de Control de Acceso Utilizando la Tecnología de Identificación RFID para la Empresa Soluciones G Cuatro del Ecuador CIA. LTDA"*. [En línea]. (tesis pregrado). Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito-Ecuador. 2009. [Consulta: 02 de Octubre de 2015]. Disponible en: http://bibdigital.epn.edu.ec/bitstream/15000/1779/1/CD-2365.pdf

16. **RODRIGUÉZ, J** *"Diseño e Implementación de un lector PC/SC inalámbrico para tarjeta inteligente basado en plataformas móviles NFC"*. [En línea] (tesis pregrado)**.** Universidad de Cantabria, Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, Cantabria-España. 2013. pp.6-7. [Consulta: 02 de Noviembre de

2015]. Disponible en: http://repositorio.unican.es/xmlui/bitstream/handle/10902/1907/352901.pdf?sequence=6,

17. **SANTACRUZ, C. & SUNTAXI, J.** *RFID Vieja tecnología, nuevo suceso,* Ecuador, [En línea]. 2007. [Consulta: 01 de Diciembre de 2015]. Disponible en: http://clusterfie.epn.edu.ec/ibernal/html/CURSOS/Marzo07Agosto07/ComInalam/TRabajos/Trabajo2/RFID/GRUPO%204/RFID.pdf

18. **SILBERSCHATZ, A.** *Fundamentos de Bases de Datos,* [En línea], Madrid-España, 2002. [Consulta: 22 de Diciembre de 2015]. Disponible en: https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

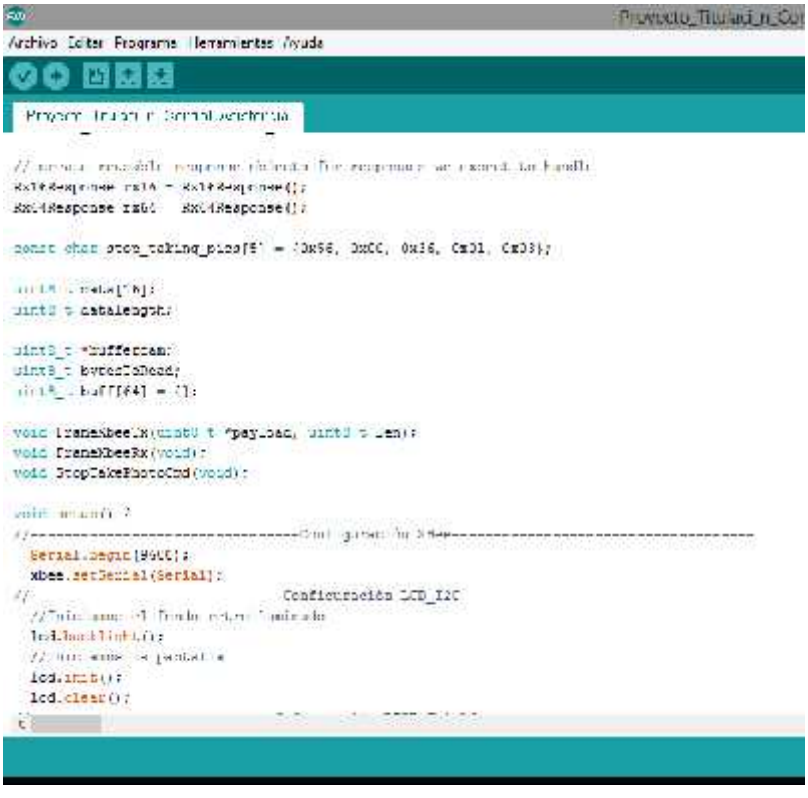
19. **TECNURA**. *Parámetros de configuración en módulo XBEE-PRO® S2B ZB para medición de variables ambientales.* Bogotá-Colombia. Universidad Distrital Francisco José de Caldas, [En línea]. 2015. [Consulta: 20 de Diciembre de 2015]. Disponible en: http://www.redalyc.org/pdf/2570/257040047012.pdf

20. **TELECTRÓNICA,** *Introducción a la identificación por radio Frecuencia – RFID,* [En línea]. *Argentina,* 2006. [Consulta: 27 de Noviembre de 2015]. Disponible en: http://www.telectronica.com/rfidtelectronica.pdf

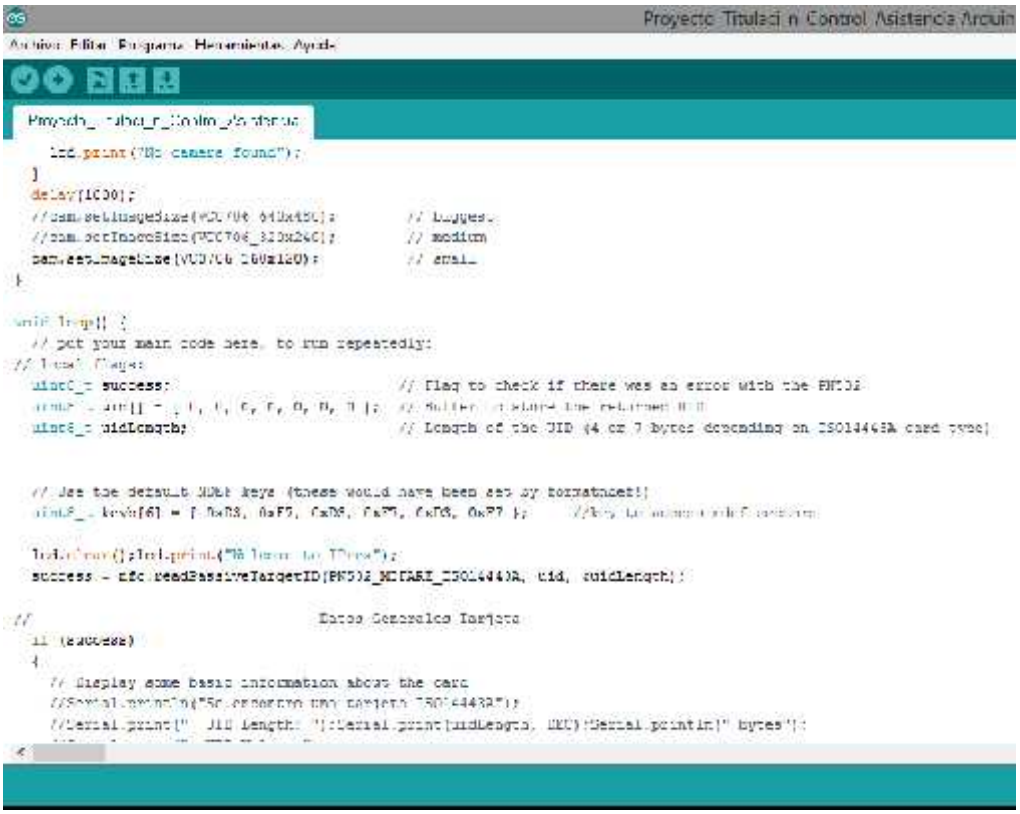
21. **WHEAT, D.** *Arduino Internals* [En línea], 2011. [Consulta: 10 de Noviembre de 2015]. Disponible en: http://www.hmangas.com/Electronica/Datasheets/Arduino/LIBROS%20Y%20MANUALES/%5BArduino.Internals(2011)%5D.Dale.Wheat.pdf

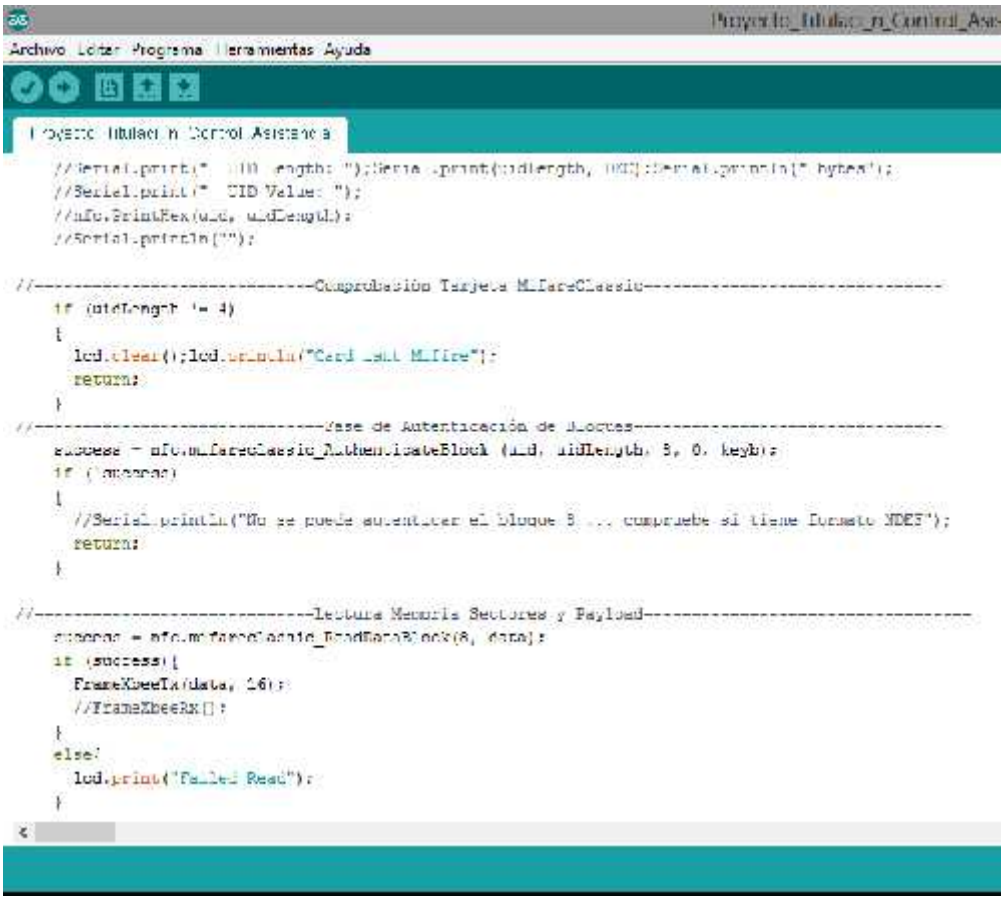
ANEXOS

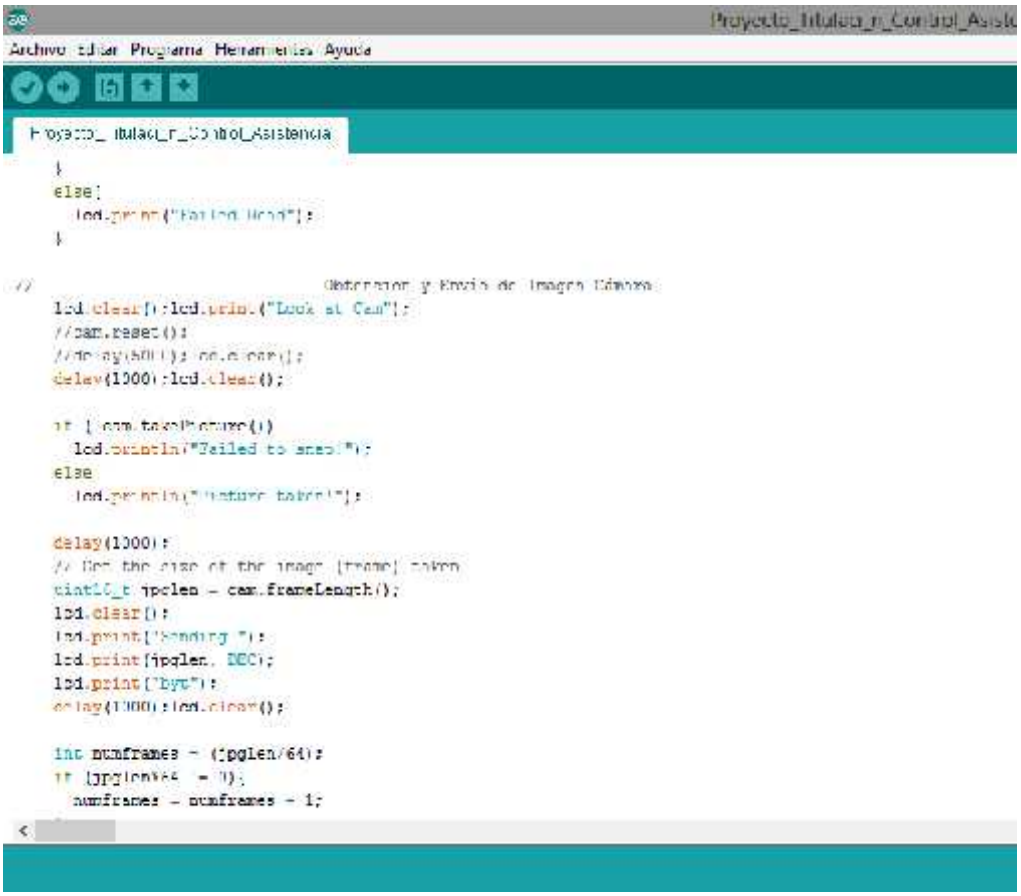
Anexo A: Configuración Arduino



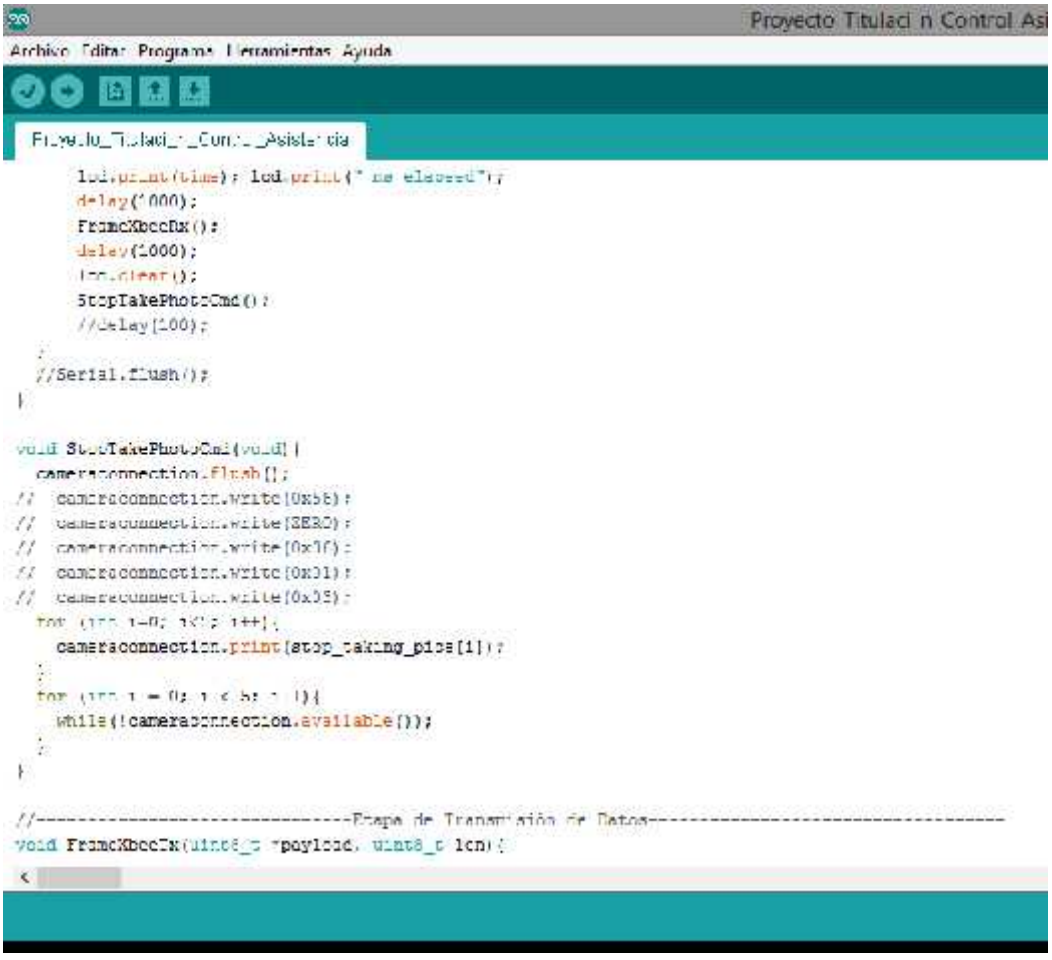

Proyecto_Titulaci_n_
Archivo Editar Programa Herramientas Ayuda
Proyecto_Titulaci_n_Control_Asistencia
lcd.backlight();
//Iniciamos la pantalla
lcd.init();
lcd.clear();
//------------------------------Configuracion RFID Shield---------------------------------
// PN532 configuration:
//Serial.println("Esperando por PN532...");
nfc.begin();
uint32_t versiondata = nfc.getFirmwareVersion();
if (! versiondata) {
lcd.print("RFID PN53x");lcd.setCursor(0,1);lcd.print("is not present");
while (1); // halt
}
// Got ok data, print it out!
lcd.clear();
lcd.print("Find chip PN5"); lcd.print((versiondata>>24) & 0xFF, HEX);delay(1000);
// configure board to read RFID tags
nfc.SAMConfig();
//   Configuracion de Camara TTL
lcd.clear();
if (cam.begin()) {
lcd.print("Camera found:");
} else {
lcd.print("No camera found");

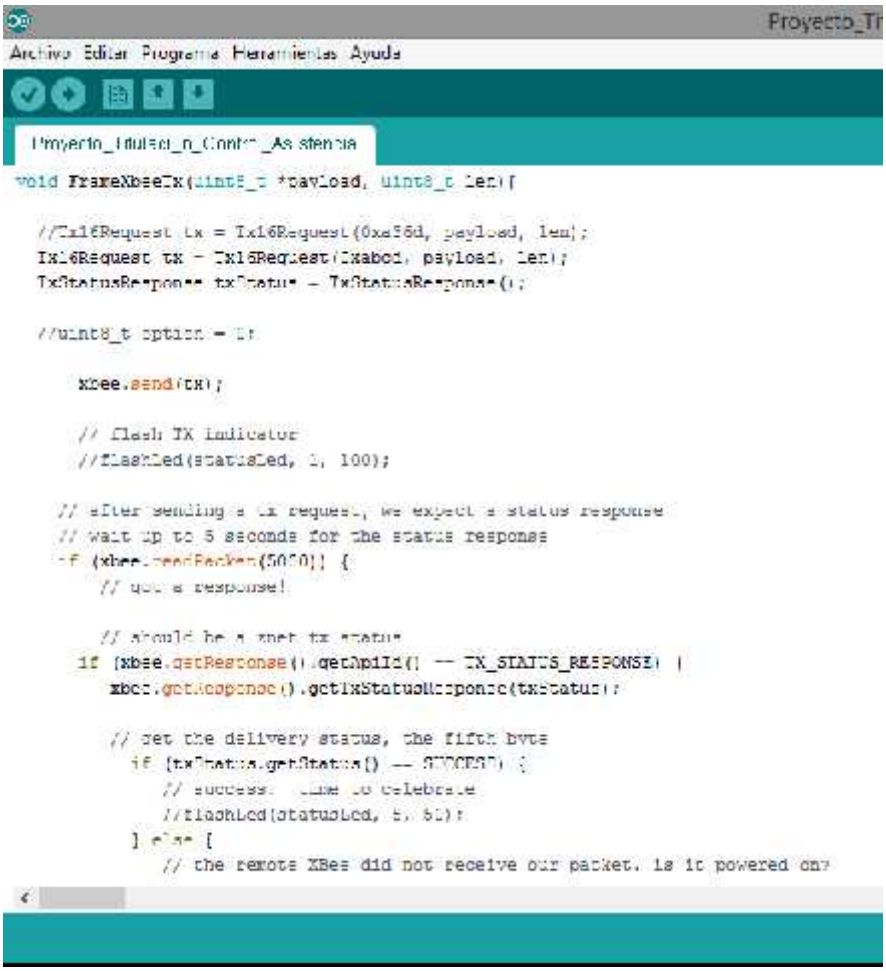

Proyecto_Titulaci_n_Control_Asistencia_Arduin
Archivo Editar Programa Herramientas Ayuda
lcd.print("No camera found");
}
delay(1000);
//cam.setImageSize(VC0706_640x480);   // biggest
//cam.setImageSize(VC0706_320x240);   // medium
cam.setImageSize(VC0706_160x120);   // small
}
void loop() {
// put your main code here, to run repeatedly:
// Local Vars:
uint8_t success;   // Flag to check if there was an error with the PN532
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 }; // Buffer to store the returned UID
uint8_t uidLength;   // Length of the UID (4 or 7 bytes depending on ISO14443A card type)
// Set the default NDEF keys (these would have been set by formatNDEF!)
uint8_t key[6] = { 0xD3, 0xF7, 0xD3, 0xF7, 0xD3, 0xD3 };
lcd.clear();lcd.print("Ya hemos leido Tarjeta");
success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);
//   Datos Generales Tarjeta
if (success)
{
// Display some basic information about the card
//Serial.println("Se encontro una tarjeta ISO14443");
//Serial.print(" UID Length: ");Serial.print(uidLength, DEC);Serial.println(" bytes");


Archivo Editar Programa Herramientas Ayuda
//Serial.print(" UID Length: ");Serial.print(uidLength, DEC);Serial.println(" bytes");
//Serial.print(" UID Value: ");
//nfc.PrintHex(uid, uidLength);
//Serial.println("");
//----------Comprobación Tarjeta MiFareClassic----------
if (uidLength != 4)
{
lcd.clear();lcd.println("Card Not MiFire");
return;
}
//----------Fase de Autenticación de Bloques----------
success = nfc.mifareclassic_AuthenticateBlock (uid, uidLength, 8, 0, keyb);
if (success)
{
//Serial.println("No se puede autenticar el bloque 8 .. compruebe si tiene formato MIFI");
return;
}
//----------Lectura Memoria Sectores y Payload----------
success = nfc.mifareclassic_ReadDataBlock(8, data);
if (success){
FrameXbeeTx(data, 16);
//FrameXbeeRx();
}
else{
lcd.print("Failed Read");
}

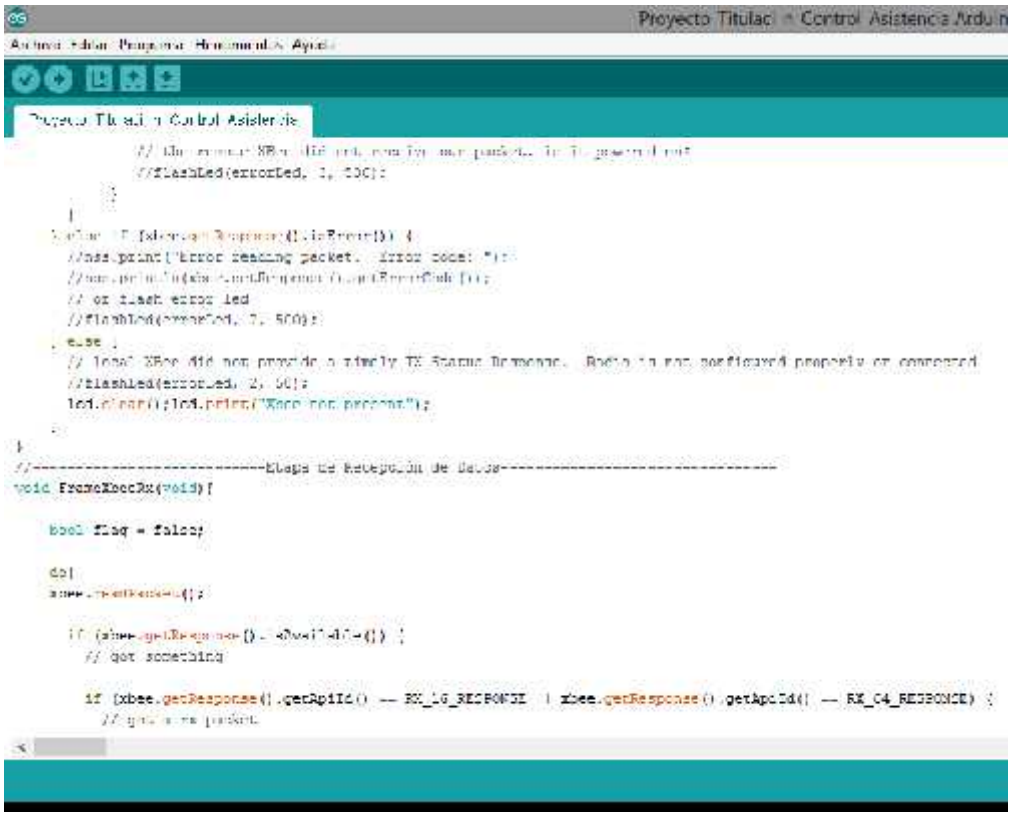

Archivo Editar Programa Herramientas Ayuda
}
else{
lcd.print("Failed Read");
}
// Obtencion y Envio de Imagen Camara
lcd.clear();lcd.print("Look at Cam");
//cam.reset();
//delay(500);cam.clear();
delay(1000);lcd.clear();
if (!cam.takePicture())
lcd.println("Failed to snap!");
else
lcd.println("Picture taken!");
delay(1000);
// Get the size of the image (frame) taken
uint16_t jpglen = cam.frameLength();
lcd.clear();
lcd.print("Sending ");
lcd.print(jpglen, DEC);
lcd.print("byt");
delay(1000);cam.clear();
int numframes = (jpglen/64);
if (jpglen%64 > 0);
numframes = numframes + 1;


Proyecto_Titulac_n_Control_Asistencia Arduino
Archivo Editar Programa Herramientas Ayuda
Proyecto_Titulac_n_Control_Asistencia
lcd.print(time); lcd.print(" ms elapsed");
FrameXbeeTx(buff, 6);
jpglen -= bytesToRead;
lcd.clear();
time = millis() - time;
lcd.print("done: ");
lcd.setCursor(0,1);
lcd.print(time); lcd.print(" ms elapsed");


Proyecto Titulaci n Control Asi
Archivo Editar Programa Herramientas Ayuda
Proyecto_Titulaci_n_Control_Asistencia
lcd.print(time); lcd.print(" ms elapsed");
delay(1000);
FrameXbeeTx();
delay(1000);
lcd.clear();
StopTakePhotoCmd();
//delay(100);
//Serial.flush();
void StopTakePhotoCmd(void){
cameraconnection.flush();
// cameraconnection.write(0x56);
// cameraconnection.write(0x00);
// cameraconnection.write(0x36);
// cameraconnection.write(0x01);
// cameraconnection.write(0x03);
for (int i=0; i<3; i++){
cameraconnection.print(stop_taking_pics[i]);
}
for (int i = 0; i < 5; i ++){
while(!cameraconnection.available());
}
//--------------------------------Etapa de Transmisión de Datos-----------------------------------
void FrameXbeeTx(uint8_t *payload, uint8_t len){

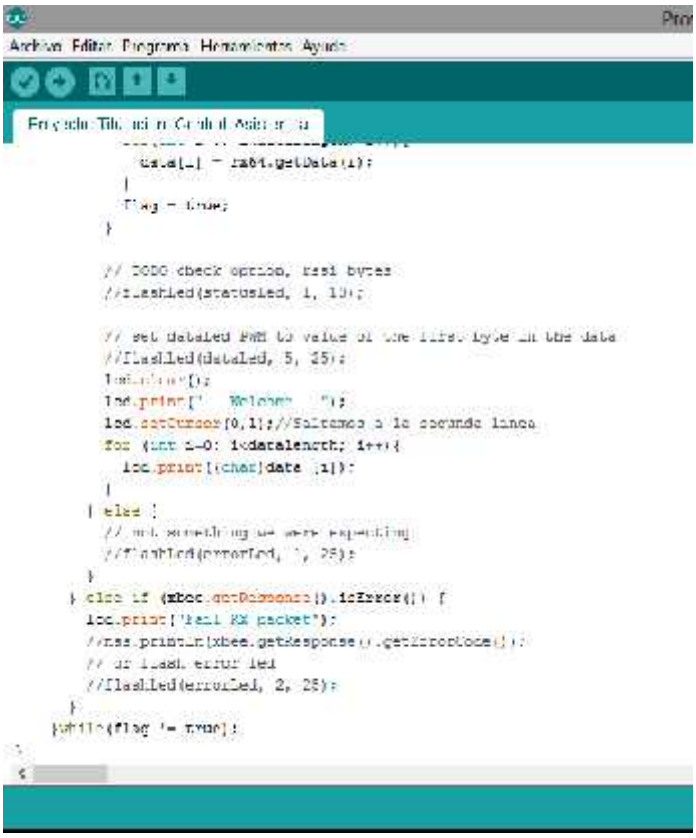


```
Archivo Editar Programa Herramientas Ayuda

Proyecto_Titulacion_Control_Asistencia

void FrameXbeeTx(uint8_t *payload, uint8_t len){

  //Tx16Request tx = Tx16Request(0xa56d, payload, len);
  Tx16Request tx = Tx16Request(0xabcd, payload, len);
  TxStatusResponse* txStatus = TxStatusResponse();

  //uint8_t option = 0;

    xbee.send(tx);

    // flash TX indicator
    //flashLed(statusLed, 1, 100);

  // after sending a tx request, we expect a status response
  // wait up to 5 seconds for the status response
  if (xbee.readPacket(5000)) {
     // got a response!

     // should be a znet tx status
     if (xbee.getResponse().getApiId() == TX_STATUS_RESPONSE) {
        xbee.getResponse().getTxStatusResponse(txStatus);

        // get the delivery status, the fifth byte
          if (txStatus.getStatus() == SUCCESS) {
             // success.  time to celebrate
             //flashLed(statusLed, 5, 50);
          } else {
             // the remote XBee did not receive our packet. is it powered on?
```



```
Proyecto Titulacion Control Asistencia Arduino

Archivo Editar Programa Herramientas Ayuda

Proyecto_Titulacion_Control_Asistencia

          // the remote XBee did not receive our packet. is it powered on?
          //flashLed(errorLed, 3, 500);
        }
      }
    } else if (xbee.getResponse().isError()) {
      //nss.print("error reading packet.  error code: ");
      //nss.println(xbee.getResponse().getErrorCode());
      // or flash error led
      //flashLed(errorLed, 3, 500);
    } else {
      // local XBee did not provide a timely TX Status Response.  Radio is not configured properly or connected
      //flashLed(errorLed, 2, 50);
      lcd.clear();lcd.print("Xbee no recibe");
    }
}
//----------------------------Etapa de Recepción de Datos---------------------------------
void FrameXbeeRx(void){

    bool flag = false;

    do{
    xbee.readPacket();

      if (xbee.getResponse().isAvailable()) {
        // got something

        if (xbee.getResponse().getApiId() == RX_16_RESPONSE || xbee.getResponse().getApiId() == RX_64_RESPONSE) {
          // got a rx packet
```

```
Archivo Editar Programa Herramientas Ayuda

// get a rx packet

if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
        xbee.getResponse().getRx16Response(rx16);
  datalength = rx16.getDataLength();
  //option = rx16.getOption();
  for(int i=0; i<datalength; i++){
    data[i] = rx16.getData(i);
  }
  flag = true;
} else {
        xbee.getResponse().getRx64Response(rx64);
  datalength = rx64.getDataLength();
  //option = rx64.getOption();
  for(int i=0; i<datalength; i++){
    data[i] = rx64.getData(i);
  }
  flag = true;
}

// TODO check option, rssi bytes
//flashLed(statusLed, 1, 10);

// set dataLed PWM to value of the first byte in the data
//flashLed(dataLed, 5, 25);
lcd.clear();
lcd.print("---Welcome---");
lcd.setCursor(0,1);//Saltamos a la segunda linea
```



```
Archivo Editar Programa Herramientas Ayuda

    data[i] = rx64.getData(i);
  }
  flag = true;
 }

 // TODO check option, rssi bytes
 //flashLed(statusLed, 1, 10);

 // set dataLed PWM to value of the first byte in the data
 //flashLed(dataLed, 5, 25);
 lcd.clear();
 lcd.print("   Welcome   ");
 lcd.setCursor(0,1);//Saltamos a la segunda linea
 for (int i=0; i<datalength; i++){
   lcd.print((char)data[i]);
 }
} else {
 // not something we were expecting
 //flashLed(errorLed, 1, 25);
}
} else if (xbee.getResponse().isError()) {
 lcd.print("Recl RX packet");
 //nss.println(xbee.getResponse().getErrorCode());
 // or flash error led
 //flashLed(errorLed, 2, 25);
}
}while(flag != true);
```

Anexo B: Configuración formato NDEF

Formato de datos: NFC Data Exchange Format (NDEF)

Se define un formato de encapsulación de mensaje para intercambiar información entre dispositivos NFC, ya sea de un dispositivo a una etiqueta o entre dos dispositivos NFC activos, también se especifican las reglas para construir un mensaje NDEF correcto, así como una cadena ordenada de registros NDEF.

NDEF no hace referencia a ningún circuito, ni arquitectura de conexión, ni se debe pensar que especifique el intercambio de información, es solamente un formato de mensaje. Este formato es el mismo para tarjetas, así como para dispositivos NFC, de esto se concluye que la información de NDEF no guarda relación con el tipo de dispositivo que participa en una comunicación.

Con este formato pueden transmitirse varios tipos de información, como:

- Documentos o fragmentos XML, imágenes de diversos formatos y datos encriptados.
- Cadenas de información encapsulada.
- Documentos múltiples que guardan alguna relación lógica.

Ejemplo de escritura a una etiqueta NFC en el IDE de Arduino

```
#include <PN532.h>
#include <SPI.h>
#define PN532_CS 10
PN532 nfc(PN532_CS);

uint8_t written=0;
#define  NFC_DEMO_DEBUG 1

void setup(void) {
#ifdef NFC_DEMO_DEBUG
  Serial.begin(9600);
  Serial.println("Hola");
#endif
  nfc.begin();

  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
#ifdef NFC_DEMO_DEBUG
    Serial.print("No encontró la tarjeta PN53X");
#endif
    while (1);
  }
#ifdef NFC_DEMO_DEBUG
  Serial.print("Chip PN535 encontrado");
  Serial.println((versiondata>>24) & 0xFF, HEX);
  Serial.print("Firmware ver. ");
  Serial.print((versiondata>>16) & 0xFF, DEC);
  Serial.print('.');
  Serial.println((versiondata>>8) & 0xFF, DEC);
  Serial.print("Supports ");
  Serial.println(versiondata & 0xFF, HEX);
#endif
  nfc.SAMConfig();
}

void loop(void)
{
  uint32_t id;
  id = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A);
```

```
  if (id != 0)
  {
#ifdef NFC_DEMO_DEBUG
    Serial.print("Read card #");
    Serial.println(id);
    Serial.println();
#endif
    uint8_t keys[]= {
      0xFF,0xFF,0xFF,0xFF,0xFF,0xFF    };
    char writeBuffer;
      writeBuffer='Escribe_aqui_el_contenido_a_grabar_en_la_etiqueta';
    if(nfc.authenticateBlock(1, id ,0x08,KEY_A,keys))
    {

      if(written == 0)
      {
        written = nfc.writeMemoryBlock(1,0x04,writeBuffer);
        if(written)
#ifdef NFC_DEMO_DEBUG
          Serial.println("Escritura exitosa");
#endif
      }

      uint8_t block[16];
      if(nfc.readMemoryBlock(1,0x04,block))
      {
#ifdef NFC_DEMO_DEBUG
        Serial.println("Read block 0x08:");
        for(uint8_t i=0;i<16;i++)
        {
          Serial.print(block[i],HEX);
          Serial.print(" ");
        }
        Serial.println();
#endif
      }
    }
  }

  delay(500);
}
```

AG Electrónica S.A. de C.V.	AG Electrónica S.A. de C.V. República del Salvador N° 20 Segundo Piso Teléfono: 5130 - 7210		
Acotación: NA	http://www.agelectronica.com/	Escala NA	Rev 1.ASS Rev 2. BAAB
Tolerancia: NA	Descripción: Escudo NFC/RFID para arduino (PN532)		
Tolerancia: NA	Fecha: 06/05/2015	Número de parte: ADA-789	

Anexo C: Datasheet Xbee Serie 1

1. XBee®/XBee-PRO® RF Modules

The XBee and XBee-PRO RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

Key Features

Long Range Data Integrity

XBee

- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (90 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO

- Indoor/Urban: up to 300' (90 m), 200' (60 m) for International variant
- Outdoor line-of-sight: up to 1 mile (1600 m), 2500' (750 m) for International variant
- Transmit Power: 63mW (18dBm), 10mW (10dBm) for International variant
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Source/Destination Addressing

Unicast & Broadcast Communications

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Low Power

XBee

- TX Peak Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO

- TX Peak Current: 250mA (150mA for international variant)
- TX Peak Current (RPSMA module only): 340mA (180mA for international varian
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

ADC and I/O line support

Analog-to-digital conversion, Digital I/O

I/O Line Passing

Easy-to-Use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

AT and API Command Modes for configuring module parameters

Extensive command set

Small form factor

Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p64] for FCC Requirements. Systems that contain XBee®/XBee-PRO® RF Modules inherit Digi Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee®/XBee-PRO® RF Modules are optimized for use in the United States, Canada, Australia, Japan, and Europe. Contact Digi for complete list of government agency approvals.

Specifications

Table 1-01. Specifications of the XBee®/XBee-PRO® RF Modules

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	Up to 100 ft (30 m)	Up to 300 ft. (90 m), up to 200 ft (60 m) international variant
Outdoor RF line-of-sight Range	Up to 300 ft (90 m)	Up to 1 mile (1600 m), up to 2500 ft (750 m) international variant
Transmit Power Output (software selectable)	1mW (0 dBm)	63mW (18dBm)* 10mW (10 dBm) for international variant
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 bps - 250 kbps (non-standard baud rates also supported)	1200 bps - 250 kbps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	250mA (@3.3 V) (150mA for international variant) RPSMA module only: 340mA (@3.3 V) (180mA for international variant)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector, RPSMA Connector	Integrated Whip, Chip or U.FL Connector, RPSMA Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A-XBEE	4214A-XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*
Japan	R201WW07215214	R201WW08215111 (Max. 10 dBm transmit power output)*
Australia	C-Tick	C-Tick

* See Appendix A for region-specific certification requirements.

Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options when transmitting outdoors.For more information, refer to the "XBee Antennas" Knowledgebase Article located on Digi's Support Web site

Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee®/XBee-PRO® RF Modules (antenna options not shown)

Pin Signals

Figure 1-03. XBee®/XBee-PRO® RF Module Pin Numbers
(top sides shown - shields on bottom)

Table 1-02. Pin Assignments for the XBee and XBee-PRO Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / $\overline{\text{CONFIG}}$	Input	UART Data In
4	DIO8*	Output	Digital Output 8
5	$\overline{\text{RESET}}$	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / $\overline{\text{SLEEP}}$	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	$\overline{\text{RTS}}$ / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 50k Ω pull-up resistor attached to $\overline{\text{RESET}}$
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

Electrical Characteristics

Table 1-03. DC Characteristics (VCC = 2.8 - 3.4 VDC)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{IL}	Input Low Voltage	All Digital Inputs	-	-	0.35 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.7 * VCC	-	-	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-	0.5	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	VCC - 0.5	-	-	V
I_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	0.025	1	µA
I_{OZ}	High Impedance Leakage Current	V_{IN} = VCC or GND, all I/O High-Z, per pin	-	0.025	1	µA
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee) / 215, 140 (PRO, Int)	-	mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee) / 55 (PRO)	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10	-	µA

Table 1-04. ADC Characteristics (Operating)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{REFH}	VREF - Analog-to-Digital converter reference range		2.08	-	V_{DDAD}*	V
I_{REF}	VREF - Reference Supply Current	Enabled	-	200	-	µA
		Disabled or Sleep Mode	-	< 0.01	0.02	µA
V_{INDC}	Analog Input Voltage[1]		V_{SSAD} - 0.3	-	V_{DDAD} + 0.3	V

1. Maximum electrical operating range, not valid conversion range.

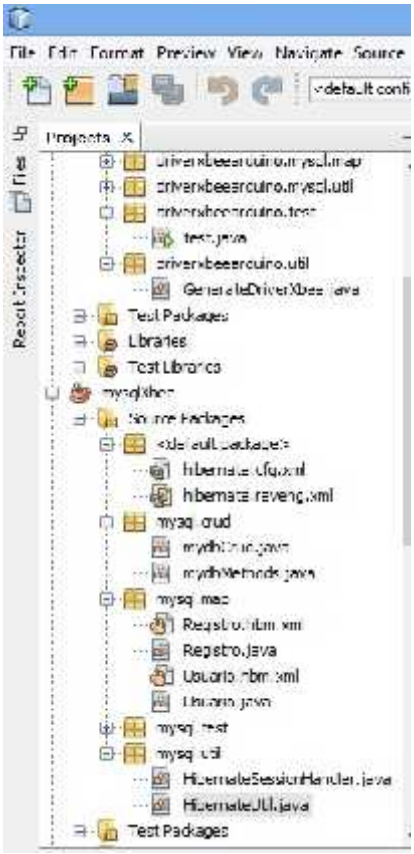
* V_{DDAD} is connected to VCC.

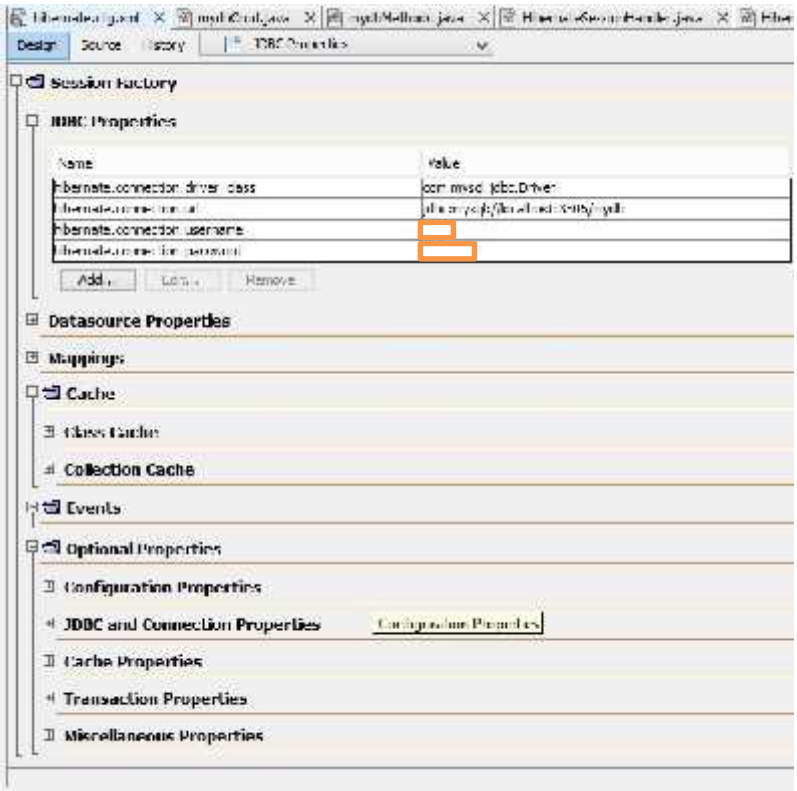
Table 1-05. ADC Timing/Performance Characteristics[1]

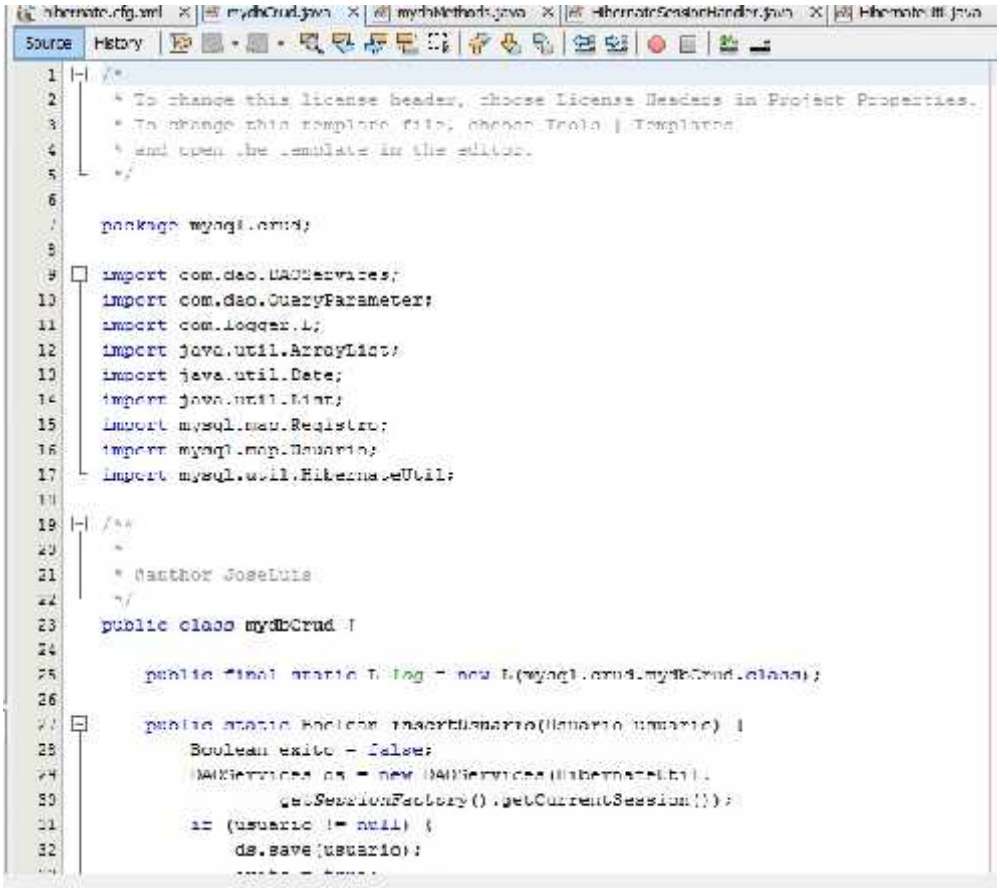
Symbol	Characteristic	Condition	Min	Typical	Max	Unit
R_{AS}	Source Impedance at Input[2]		-	-	10	kΩ
V_{AIN}	Analog Input Voltage[3]		V_{REFL}		V_{REFH}	V
RES	Ideal Resolution (1 LSB)[4]	2.08V ≤ V_{DDAD} ≤ 3.6V	2.031	-	3.516	mV
DNL	Differential Non-linearity[5]		-	±0.5	±1.0	LSB
INL	Integral Non-linearity[6]		-	±0.5	±1.0	LSB
E_{ZS}	Zero-scale Error[7]		-	±0.4	±1.0	LSB
E_{FS}	Full-scale Error[8]		-	±0.4	±1.0	LSB
E_{IL}	Input Leakage Error[9]		-	±0.05	±5.0	LSB
E_{TU}	Total Unadjusted Error[10]		-	±1.1	±2.5	LSB

1. All ACCURACY numbers are based on processor and system being in WAIT state (very little activity and no IO switching) and that adequate low-pass filtering is present on analog input pins (filter with 0.01 µF to 0.1 µF capacitor between analog input and VREFL.). Failure to observe these guidelines may result in system or microcontroller noise causing accuracy errors which will vary based on board layout and the type and magnitude of the activity.
Data transmission and reception during data conversion may cause some degradation of these specifications, depending on the number and timing of packets. It is advisable to test the ADCs in your installation if best accuracy is required.
2. R_{AS} is the real portion of the impedance of the network driving the analog input pin. Values greater than this amount may not fully charge the input circuitry of the ATD resulting in accuracy error.
3. Analog input must be between V_{REFL} and V_{REFH} for valid conversion. Values greater than V_{REFH} will convert to 83FF.
4. The resolution is the ideal step size or 1LSB = (V_{REFH}-V_{REFL})/1024
5. Differential non-linearity is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to and from the current code.
6. Integral non-linearity is the difference between the transition voltage to the current code and the adjusted ideal transition voltage for the current code. The adjusted ideal transition voltage is (Current Code–1/2)*(1/((V_{REFH}+E_{FS})–(V_{REFL}+E_{ZS}))).
7. Zero-scale error is the difference between the transition to the first valid code and the ideal transition to that code. The Ideal transition voltage to a given code is (Code–1/2)*(1/(V_{REFH}–V_{REFL})).
8. Full-scale error is the difference between the transition to the last valid code and the ideal transition to that code. The ideal transition voltage to a given code is (Code–1/2)*(1/(V_{REFH}–V_{REFL})).
9. Input leakage error is error due to input leakage across the real portion of the impedance of the network driving the analog pin. Reducing the impedance of the network reduces this error.

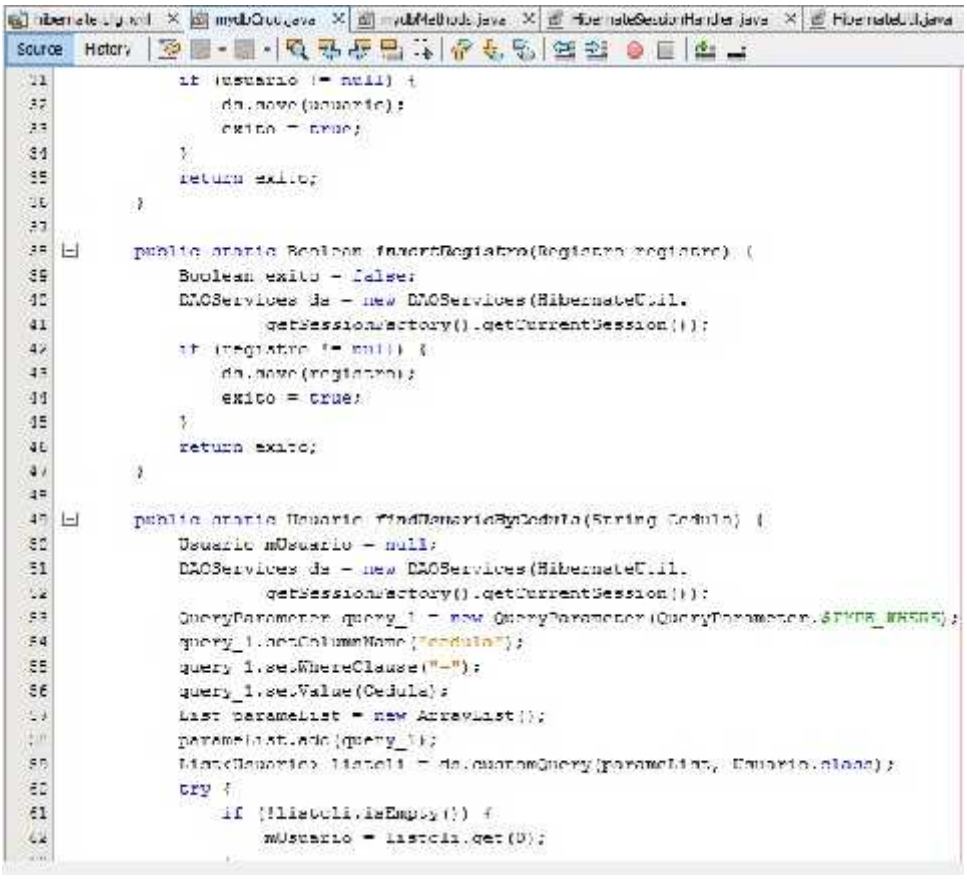
Anexo D: Código de conexión a la Base de Datos y Programación java



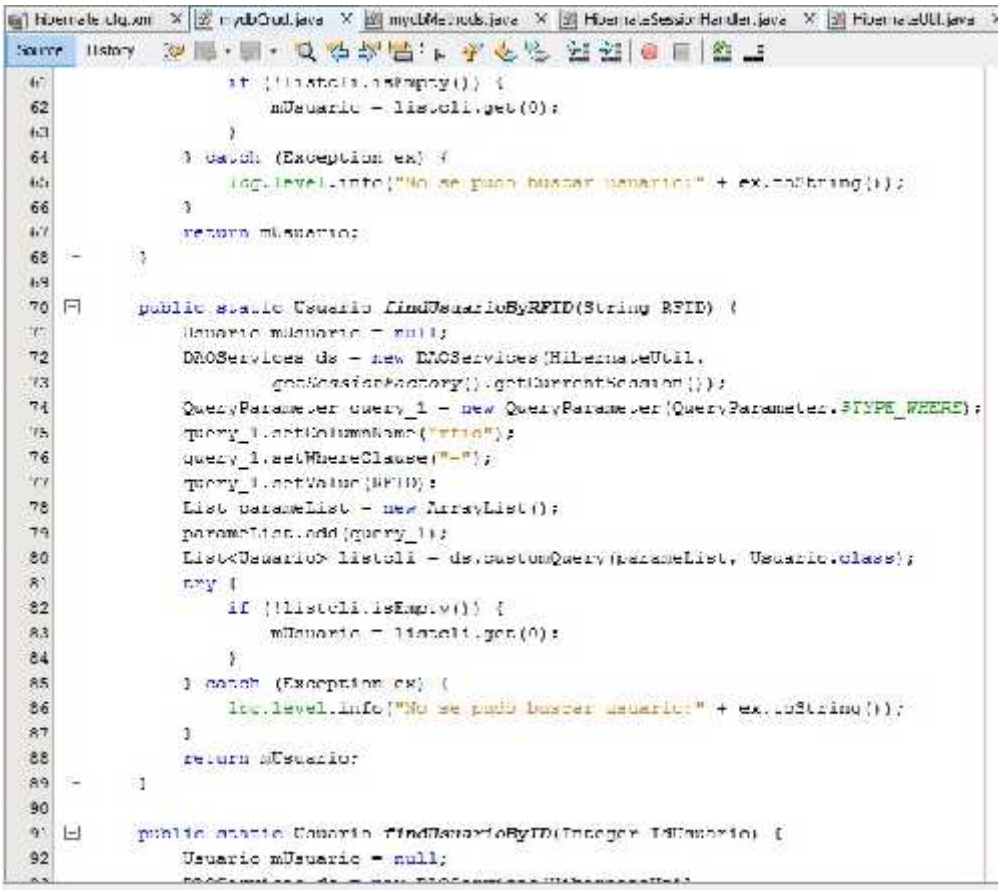




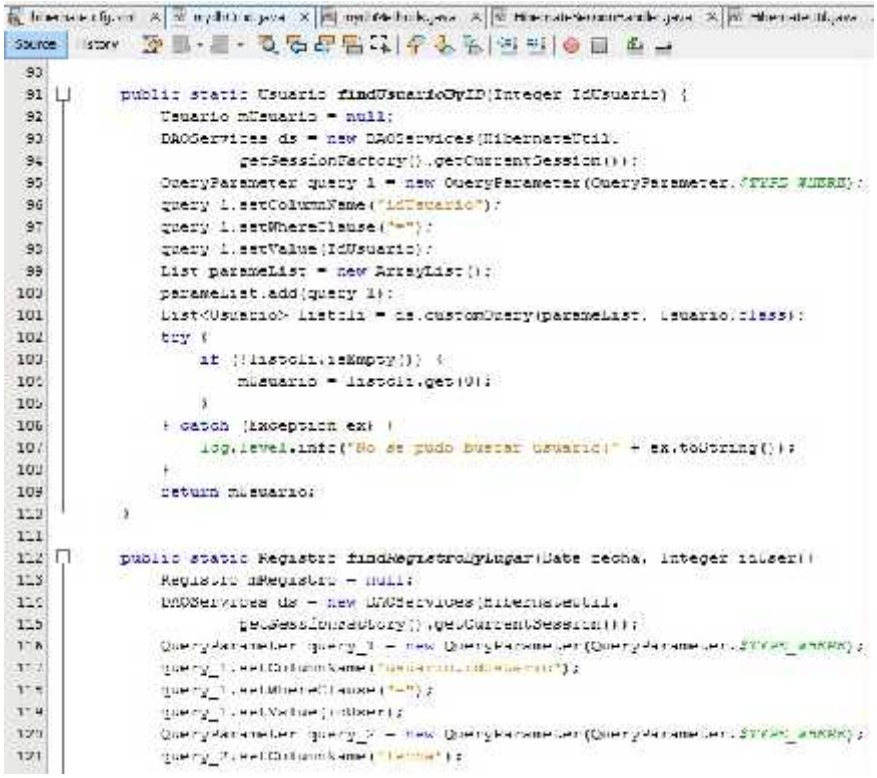
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package mysql.crud;

import com.dao.DAOServices;
import com.dao.QueryParameter;
import com.logger.L;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import mysql.bean.Registro;
import mysql.bean.Usuario;
import mysql.util.HibernateUtil;

/**
 *
 * @author JoseLuis
 */
public class mydbCrud {

    public final static L log = new L(mysql.crud.mydbCrud.class);

    public static Boolean insertUsuario(Usuario usuario) {
        Boolean exito = false;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        if (usuario != null) {
            ds.save(usuario);
```



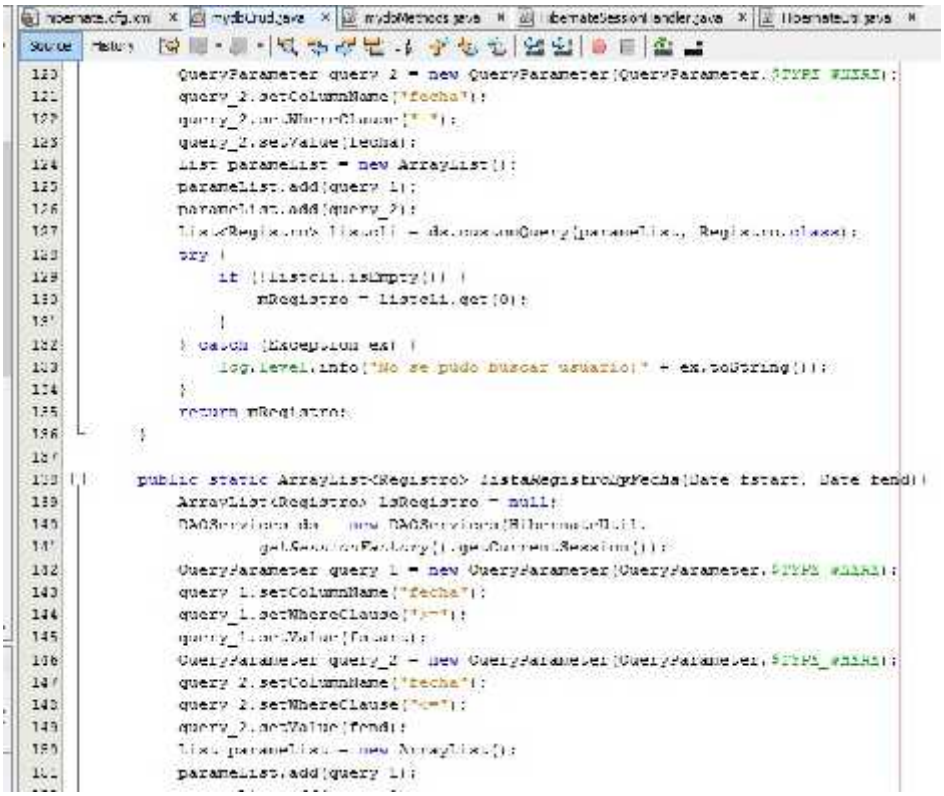
```
            if (usuario != null) {
                ds.save(usuario);
                exito = true;
            }
            return exito;
        }

        public static Boolean insertRegistro(Registro registro) {
            Boolean exito = false;
            DAOServices ds = new DAOServices(HibernateUtil.
                    getSessionFactory().getCurrentSession());
            if (registro != null) {
                ds.save(registro);
                exito = true;
            }
            return exito;
        }

        public static Usuario findUsuarioByCedula(String Cedula) {
            Usuario mUsuario = null;
            DAOServices ds = new DAOServices(HibernateUtil.
                    getSessionFactory().getCurrentSession());
            QueryParameter query_1 = new QueryParameter(QueryParameter.STRING_PARAM);
            query_1.setColumnName("cedula");
            query_1.setWhereClause("=");
            query_1.setValue(Cedula);
            List paramList = new ArrayList();
            paramList.add(query_1);
            List<Usuario> listcli = ds.queryByQuery(paramList, Usuario.class);
            try {
                if (!listcli.isEmpty()) {
                    mUsuario = listcli.get(0);
```



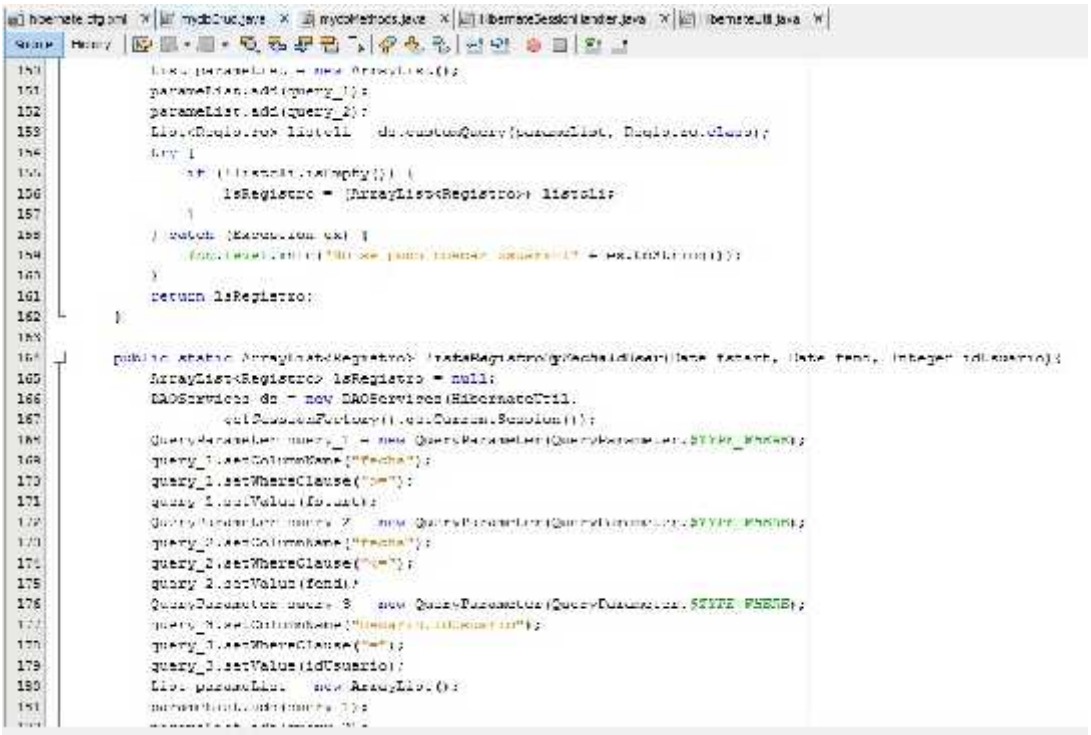
```
            if (!listcli.isEmpty()) {
                mUsuario = listcli.get(0);
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario:" + ex.toString());
        }
        return mUsuario;
    }

    public static Usuario findUsuarioByRFID(String RFID) {
        Usuario mUsuario = null;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        QueryParameter query_1 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_1.setColumnName("rfid");
        query_1.setWhereClause("=");
        query_1.setValue(RFID);
        List paramsList = new ArrayList();
        paramsList.add(query_1);
        List<Usuario> listcli = ds.customQuery(paramsList, Usuario.class);
        try {
            if (!listcli.isEmpty()) {
                mUsuario = listcli.get(0);
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario:" + ex.toString());
        }
        return mUsuario;
    }

    public static Usuario findUsuarioByID(Integer IdUsuario) {
        Usuario mUsuario = null;
```



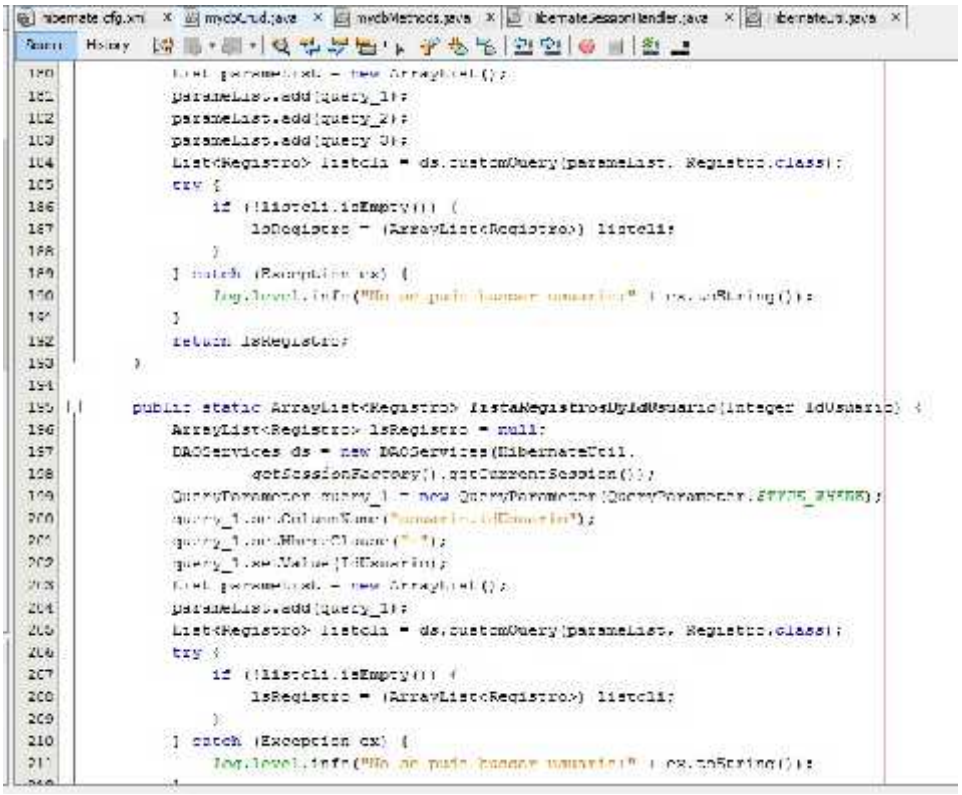
```
    public static Usuario findUsuarioByID(Integer IdUsuario) {
        Usuario mUsuario = null;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        QueryParameter query_1 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_1.setColumnName("idUsuario");
        query_1.setWhereClause("=");
        query_1.setValue(IdUsuario);
        List paramsList = new ArrayList();
        paramsList.add(query_1);
        List<Usuario> listcli = ds.customQuery(paramsList, Usuario.class);
        try {
            if (!listcli.isEmpty()) {
                mUsuario = listcli.get(0);
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario:" + ex.toString());
        }
        return mUsuario;
    }

    public static Registro findRegistroByLugar(Date fecha, Integer idUser)
        Registro dRegistro = null;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        QueryParameter query_1 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_1.setColumnName("idUsuario.idUsuario");
        query_1.setWhereClause("=");
        query_1.setValue(idUser);
        QueryParameter query_2 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_2.setColumnName("fecha");
```



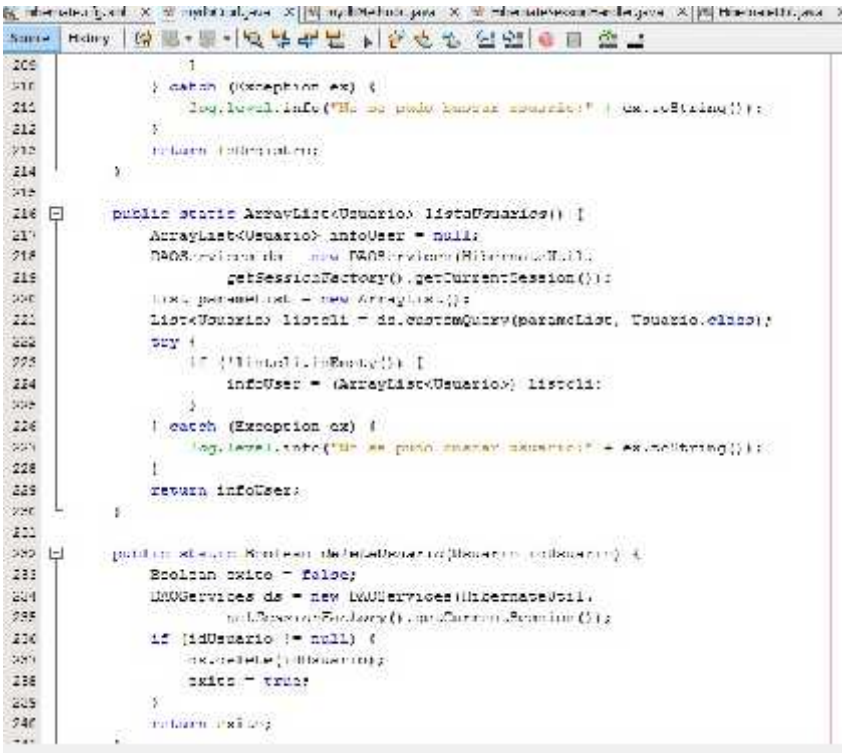
```
            QueryParameter query_2 = new QueryParameter(QueryParameter.TYPE_WHERE);
            query_2.setColumnName("fecha");
            query_2.setWhereClause("<=");
            query_2.setValue(fecha);
            List paramList = new ArrayList();
            paramList.add(query_1);
            paramList.add(query_2);
            List<Registro> listcli = da.customQuery(paramList, Registro.class);
            try {
                if (!listcli.isEmpty()) {
                    mRegistro = listcli.get(0);
                }
            } catch (Exception ex) {
                log.level.info("No se pudo buscar usuario" + ex.toString());
            }
            return mRegistro;
        }

        public static ArrayList<Registro> listaRegistrosByFecha(Date fstart, Date fend){
            ArrayList<Registro> lsRegistro = null;
            DAOServices da = new DAOServices(HibernateUtil.
                    getSessionFactory().getCurrentSession());
            QueryParameter query_1 = new QueryParameter(QueryParameter.TYPE_WHERE);
            query_1.setColumnName("fecha");
            query_1.setWhereClause(">=");
            query_1.setValue(fstart);
            QueryParameter query_2 = new QueryParameter(QueryParameter.TYPE_WHERE);
            query_2.setColumnName("fecha");
            query_2.setWhereClause("<=");
            query_2.setValue(fend);
            List paramList = new ArrayList();
            paramList.add(query_1);
```



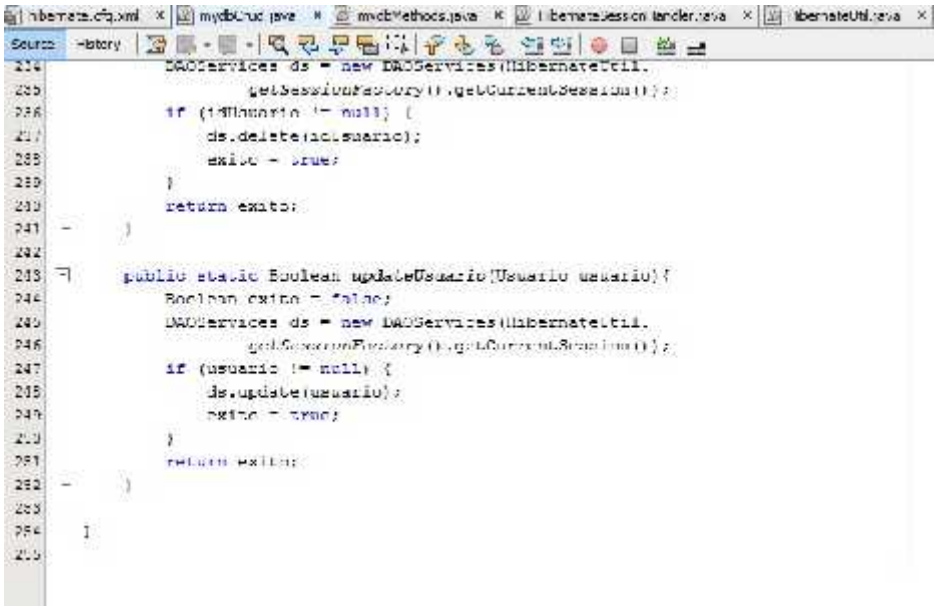
```
        List paramList = new ArrayList();
        paramList.add(query_1);
        paramList.add(query_2);
        List<Registro> listcli = da.customQuery(paramList, Registro.class);
        try {
            if (!listcli.isEmpty()) {
                lsRegistro = (ArrayList<Registro>) listcli;
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario" + ex.toString());
        }
        return lsRegistro;
    }

    public static ArrayList<Registro> listaRegistrosByFechaIdUser(Date fstart, Date fend, Integer idUsuario){
        ArrayList<Registro> lsRegistro = null;
        DAOServices da = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        QueryParameter query_1 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_1.setColumnName("fecha");
        query_1.setWhereClause(">=");
        query_1.setValue(fstart);
        QueryParameter query_2 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_2.setColumnName("fecha");
        query_2.setWhereClause("<=");
        query_2.setValue(fend);
        QueryParameter query_3 = new QueryParameter(QueryParameter.TYPE_WHERE);
        query_3.setColumnName("usuario.idUsuario");
        query_3.setWhereClause("=");
        query_3.setValue(idUsuario);
        List paramList = new ArrayList();
```



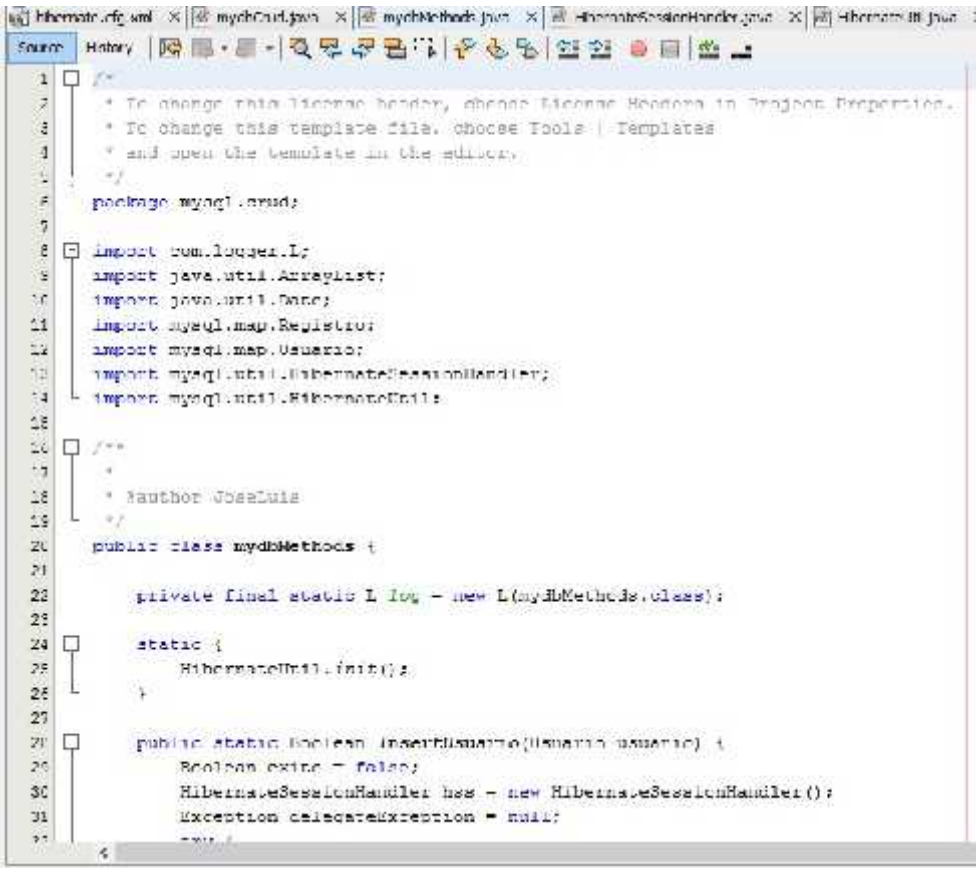
```
        List parameList = new ArrayList();
        parameList.add(query_1);
        parameList.add(query_2);
        parameList.add(query_3);
        List<Registro> listcli = ds.customQuery(parameList, Registro.class);
        try {
            if (!listcli.isEmpty()) {
                lsRegistro = (ArrayList<Registro>) listcli;
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario" + ex.toString());
        }
        return lsRegistro;
    }

    public static ArrayList<Registro> listaRegistrosByIdUsuario(Integer idUsuario) {
        ArrayList<Registro> lsRegistro = null;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        QueryParameter query_1 = new QueryParameter(QueryParameter.EQUAL_OPERATOR);
        query_1.setColumnName("usuario.idUsuario");
        query_1.setWhereClause(" ");
        query_1.setValue(idUsuario);
        List parameList = new ArrayList();
        parameList.add(query_1);
        List<Registro> listcli = ds.customQuery(parameList, Registro.class);
        try {
            if (!listcli.isEmpty()) {
                lsRegistro = (ArrayList<Registro>) listcli;
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario" + ex.toString());
        }
```



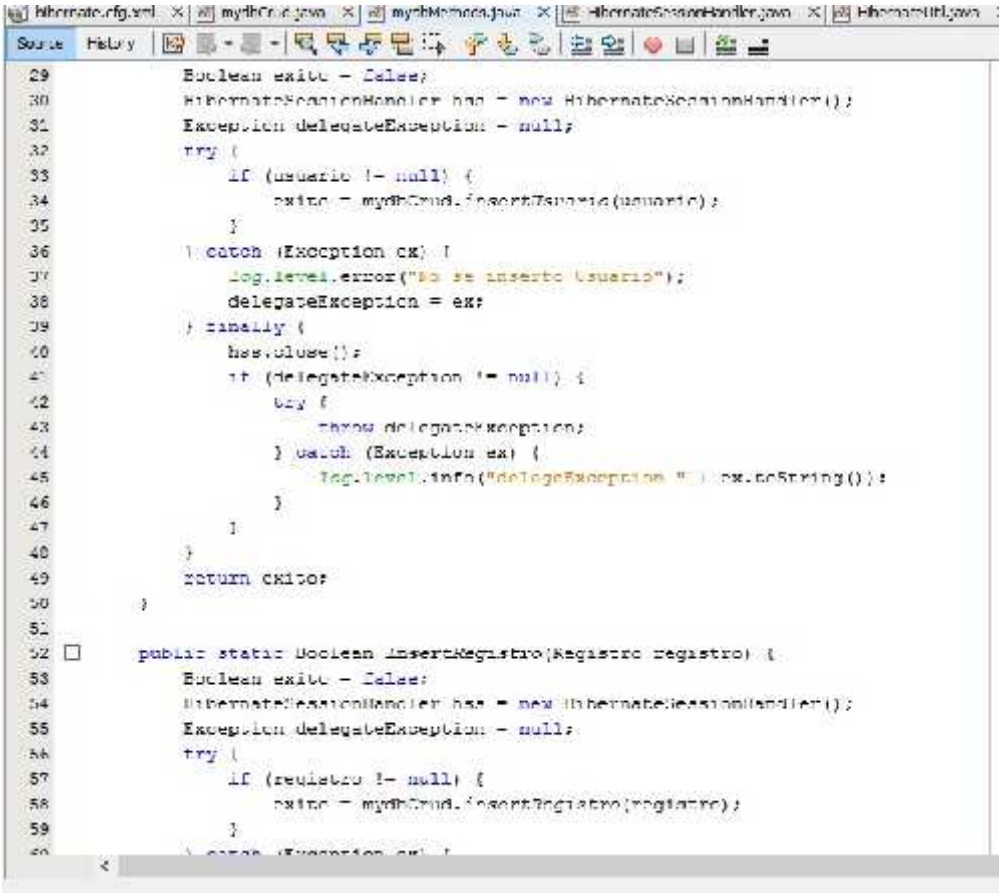
```
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario" + ex.toString());
        }
        return infoUser;
    }

    public static ArrayList<Usuario> listaUsuarios() {
        ArrayList<Usuario> infoUser = null;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        List parameList = new ArrayList();
        List<Usuario> listcli = ds.customQuery(parameList, Usuario.class);
        try {
            if (!listcli.isEmpty()) {
                infoUser = (ArrayList<Usuario>) listcli;
            }
        } catch (Exception ex) {
            log.level.info("No se pudo buscar usuario" + ex.toString());
        }
        return infoUser;
    }

    public static Boolean deleteUsuario(Usuario idUsuario) {
        Boolean exito = false;
        DAOServices ds = new DAOServices(HibernateUtil.
                getSessionFactory().getCurrentSession());
        if (idUsuario != null) {
            ds.delete(idUsuario);
            exito = true;
        }
        return exito;
    }
```



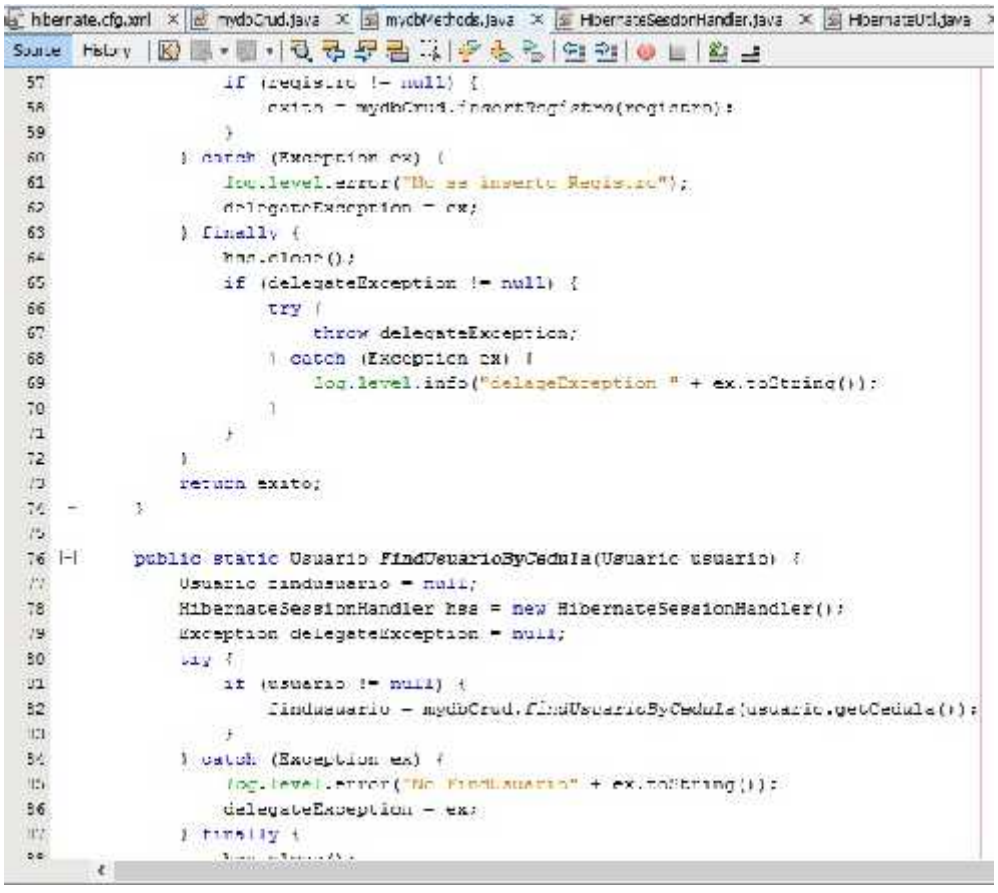
```
            DAOServices ds = new DAOServices(HibernateUtil.
                    getSessionFactory().getCurrentSession());
            if (idUsuario != null) {
                ds.delete(idUsuario);
                exito = true;
            }
            return exito;
        }

    public static Boolean updateUsuario(Usuario usuario){
            Boolean exito = false;
            DAOServices ds = new DAOServices(HibernateUtil.
                    getSessionFactory().getCurrentSession());
            if (usuario != null) {
                ds.update(usuario);
                exito = true;
            }
            return exito;
        }

}
```



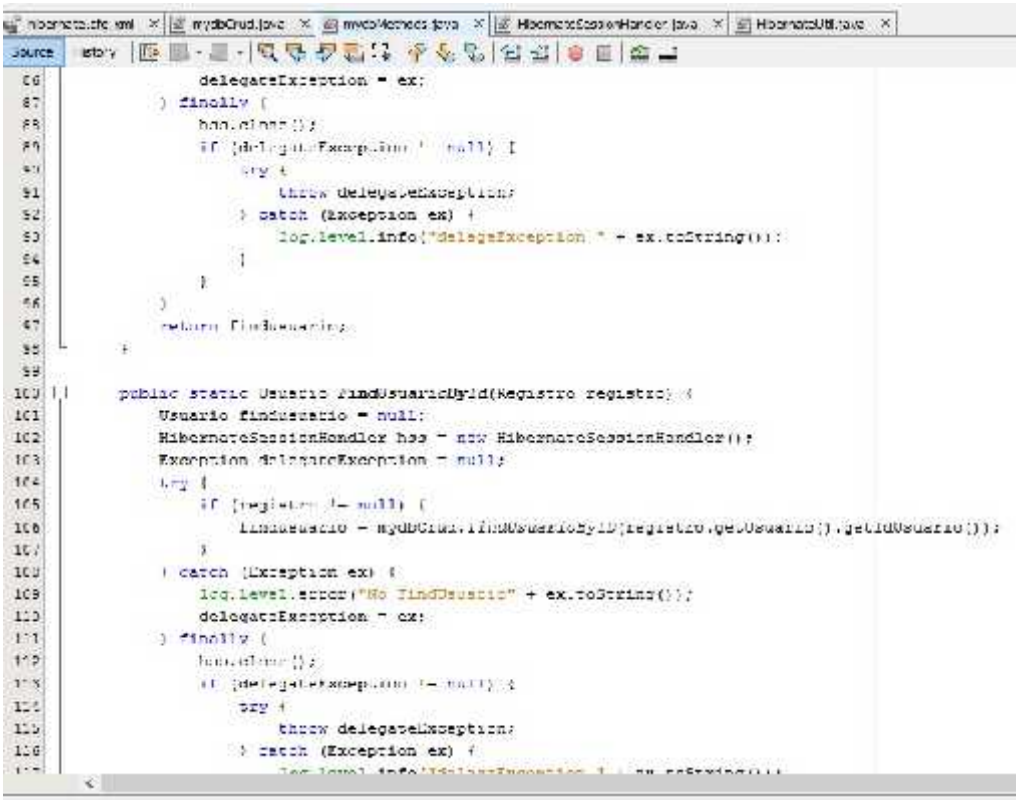
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package mysql.crud;

import com.logger.L;
import java.util.ArrayList;
import java.util.Date;
import mysql.map.Registro;
import mysql.map.Usuario;
import mysql.util.HibernateSessionHandler;
import mysql.util.HibernateUtil;

/**
 *
 * @author JoseLuis
 */
public class mydbMethods {

    private final static L log = new L(mydbMethods.class);

    static {
        HibernateUtil.init();
    }

    public static Boolean insertUsuario(Usuario usuario) {
        Boolean exito = false;
        HibernateSessionHandler hsh = new HibernateSessionHandler();
        Exception cellcateException = null;
        try {
```



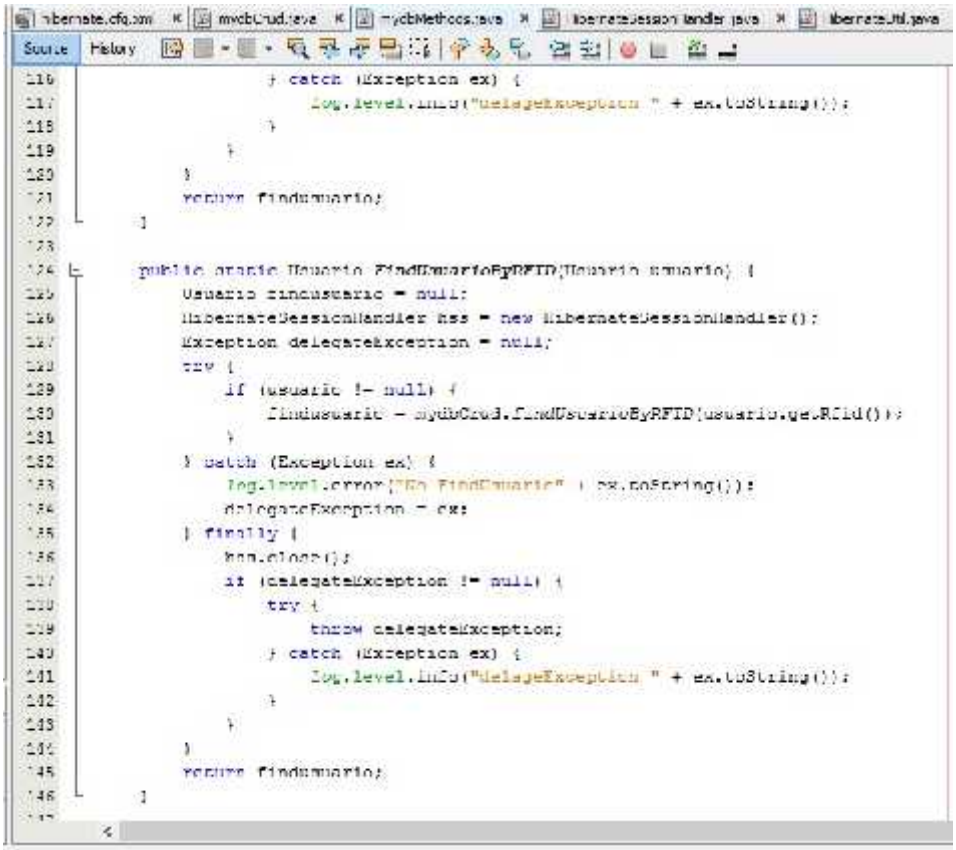
```
29          Boolean exito = false;
30          HibernateSessionHandler hsh = new HibernateSessionHandler();
31          Exception delegateException = null;
32          try {
33              if (usuario != null) {
34                  exito = mydbCrud.insertUsuario(usuario);
35              }
36          } catch (Exception ex) {
37              log.level.error("No se inserto Usuario");
38              delegateException = ex;
39          } finally {
40              hsh.close();
41              if (delegateException != null) {
42                  try {
43                      throw delegateException;
44                  } catch (Exception ex) {
45                      log.level.info("delegeException " + ex.toString());
46                  }
47              }
48          }
49          return exito;
50      }
51
52      public static Boolean insertRegistro(Registro registro) {
53          Boolean exito = false;
54          HibernateSessionHandler hsh = new HibernateSessionHandler();
55          Exception delegateException = null;
56          try {
57              if (registro != null) {
58                  exito = mydbCrud.insertRegistro(registro);
59              }
```



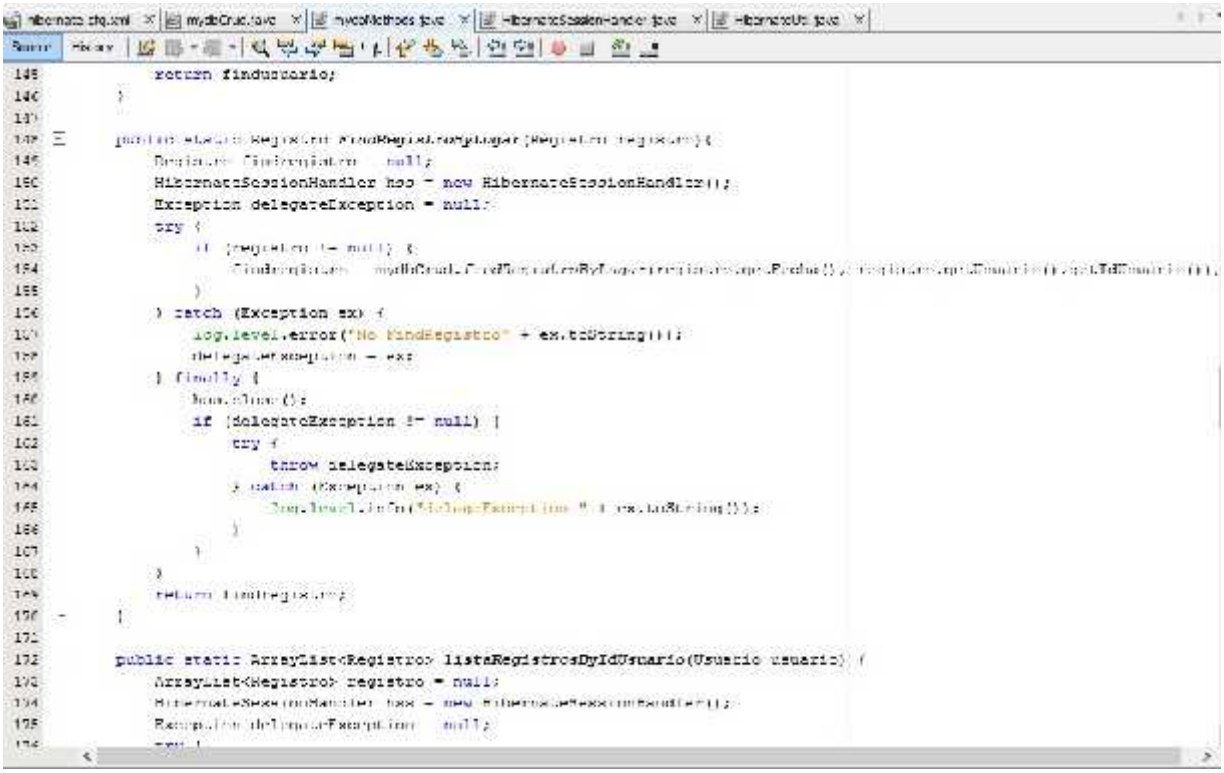
```
57              if (registro != null) {
58                  exito = mydbCrud.insertRegistro(registro);
59              }
60          } catch (Exception ex) {
61              log.level.error("No se inserto Registro");
62              delegateException = ex;
63          } finally {
64              hsh.close();
65              if (delegateException != null) {
66                  try {
67                      throw delegateException;
68                  } catch (Exception ex) {
69                      log.level.info("delegeException " + ex.toString());
70                  }
71              }
72          }
73          return exito;
74      }
75
76      public static Usuario findUsuarioByCedula(Usuario usuario) {
77          Usuario findusuario = null;
78          HibernateSessionHandler hsh = new HibernateSessionHandler();
79          Exception delegateException = null;
80          try {
81              if (usuario != null) {
82                  findusuario = mydbCrud.findUsuarioByCedula(usuario.getCedula());
83              }
84          } catch (Exception ex) {
85              log.level.error("No findUsuario" + ex.toString());
86              delegateException = ex;
87          } finally {
```



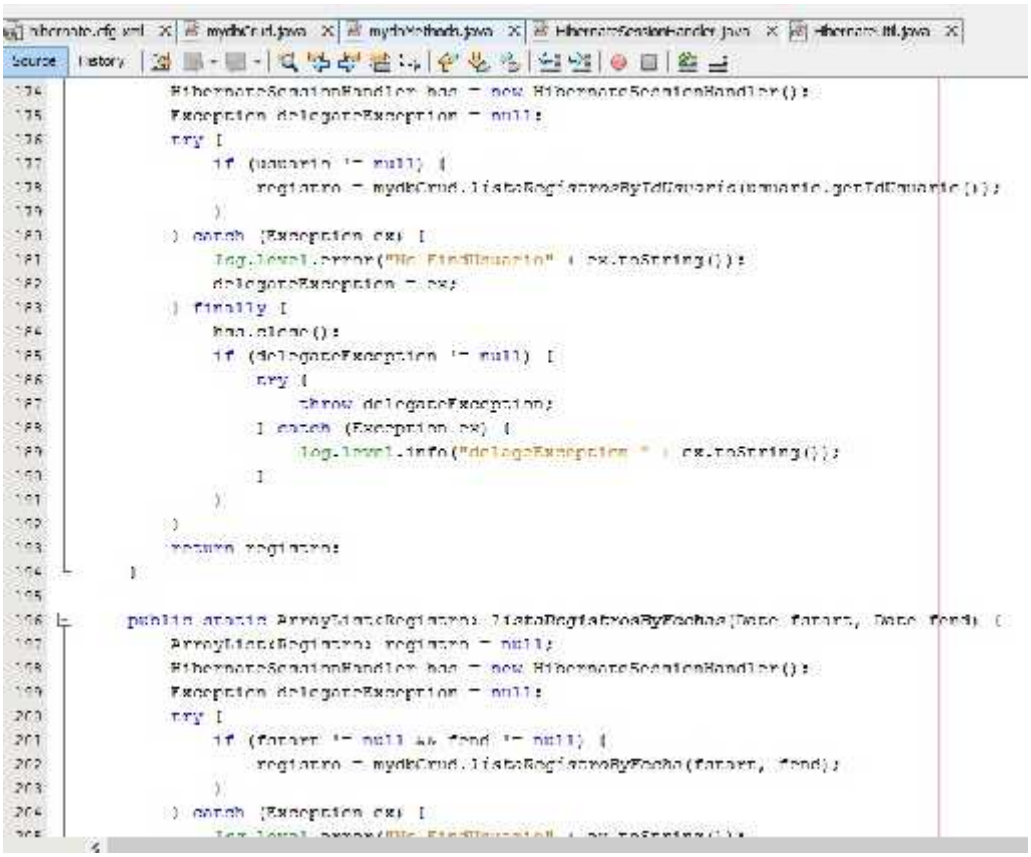
```
            delegateException = ex;
        } finally {
            hsn.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return findUsuario;
    }

    public static Usuario FindUsuarioById(Registro registro) {
        Usuario findusuario = null;
        HibernateSessionHandler hsn = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
            if (registro != null) {
                findusuario = myDbCrud.FindUsuarioById(registro.getUsuario().getIdUsuario());
            }
        } catch (Exception ex) {
            log.level.error("No FindUsuario" + ex.toString());
            delegateException = ex;
        } finally {
            hsn.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
```



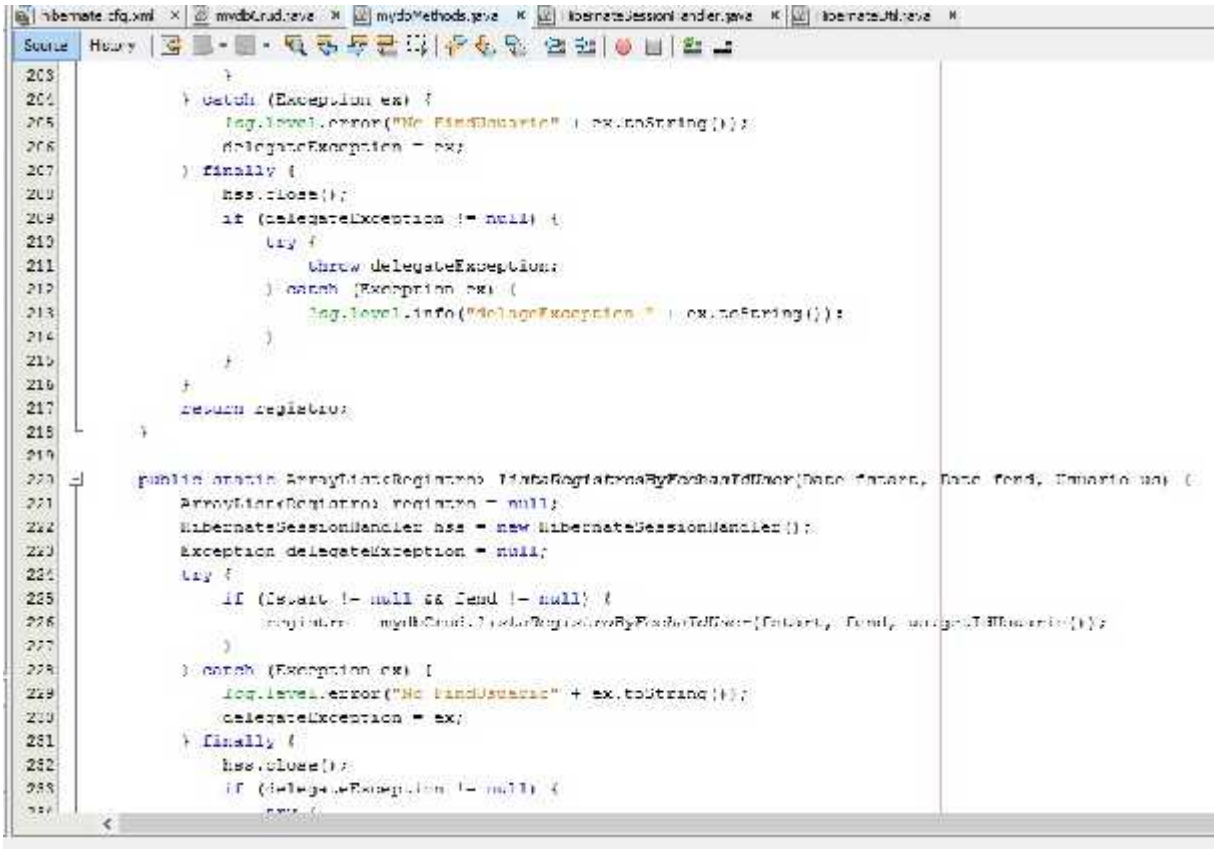
```
                    } catch (Exception ex) {
                        log.level.info("delegateException " + ex.toString());
                    }
                }
            }
            return findUsuario;
        }

        public static Usuario FindUsuarioByRFID(Usuario usuario) {
            Usuario findusuario = null;
            HibernateSessionHandler hsn = new HibernateSessionHandler();
            Exception delegateException = null;
            try {
                if (usuario != null) {
                    findusuario = myDbCrud.FindUsuarioByRFID(usuario.getRId());
                }
            } catch (Exception ex) {
                log.level.error("No FindUsuario" + ex.toString());
                delegateException = ex;
            } finally {
                hsn.close();
                if (delegateException != null) {
                    try {
                        throw delegateException;
                    } catch (Exception ex) {
                        log.level.info("delegateException " + ex.toString());
                    }
                }
            }
            return findUsuario;
        }
```



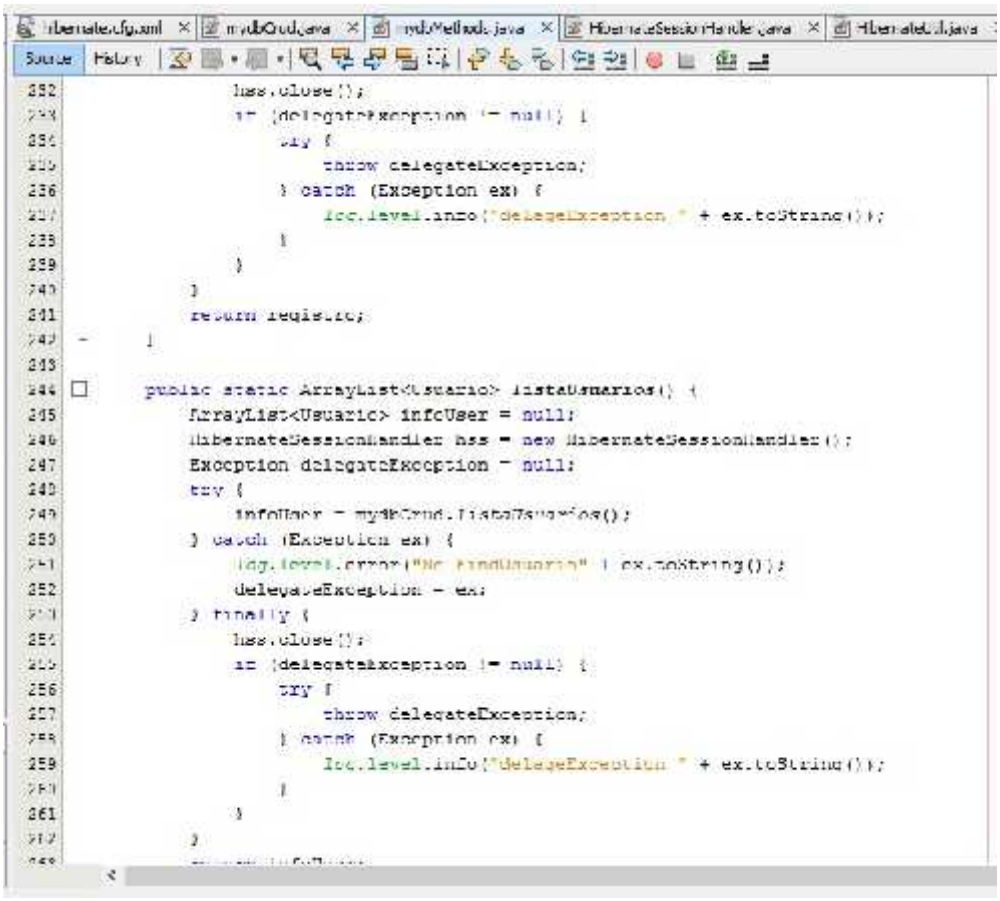
```
        return findUsuario;
    }

    public static Registro findRegistroByIdRegistro(Registro registro){
        Registro findRegistro = null;
        HibernateSessionHandler hsh = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
            if (registro != null) {
                findRegistro = myDbCrud.findRegistroByIdRegistro(registro.getFecha(), registro.getUsuario().getIdUsuario());
            }
        } catch (Exception ex) {
            log.level.error("No findRegistro" + ex.toString());
            delegateException = ex;
        } finally {
            hsh.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return findRegistro;
    }

    public static ArrayList<Registro> listaRegistrosByIdUsuario(Usuario usuario) {
        ArrayList<Registro> registro = null;
        HibernateSessionHandler hsh = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
```



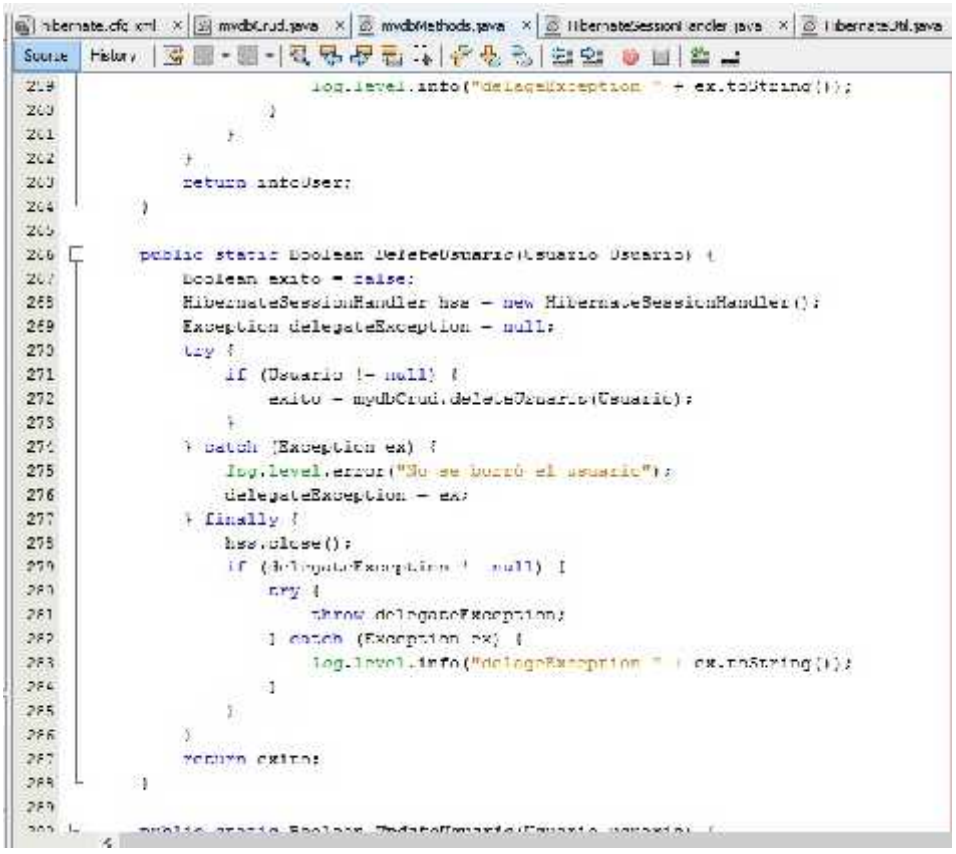
```
            HibernateSessionHandler hsh = new HibernateSessionHandler();
            Exception delegateException = null;
            try {
                if (usuario != null) {
                    registro = myDbCrud.listaRegistrosByIdUsuario(usuario.getIdUsuario());
                }
            } catch (Exception ex) {
                log.level.error("No findUsuario" + ex.toString());
                delegateException = ex;
            } finally {
                hsh.close();
                if (delegateException != null) {
                    try {
                        throw delegateException;
                    } catch (Exception ex) {
                        log.level.info("delegateException " + ex.toString());
                    }
                }
            }
            return registro;
        }

        public static ArrayList<Registro> listaRegistrosByFechas(Date fdesde, Date fhasta) {
            ArrayList<Registro> registro = null;
            HibernateSessionHandler hsh = new HibernateSessionHandler();
            Exception delegateException = null;
            try {
                if (fdesde != null && fhasta != null) {
                    registro = myDbCrud.listaRegistrosByFechas(fdesde, fhasta);
                }
            } catch (Exception ex) {
                log.level.error("No findUsuario" + ex.toString());
```



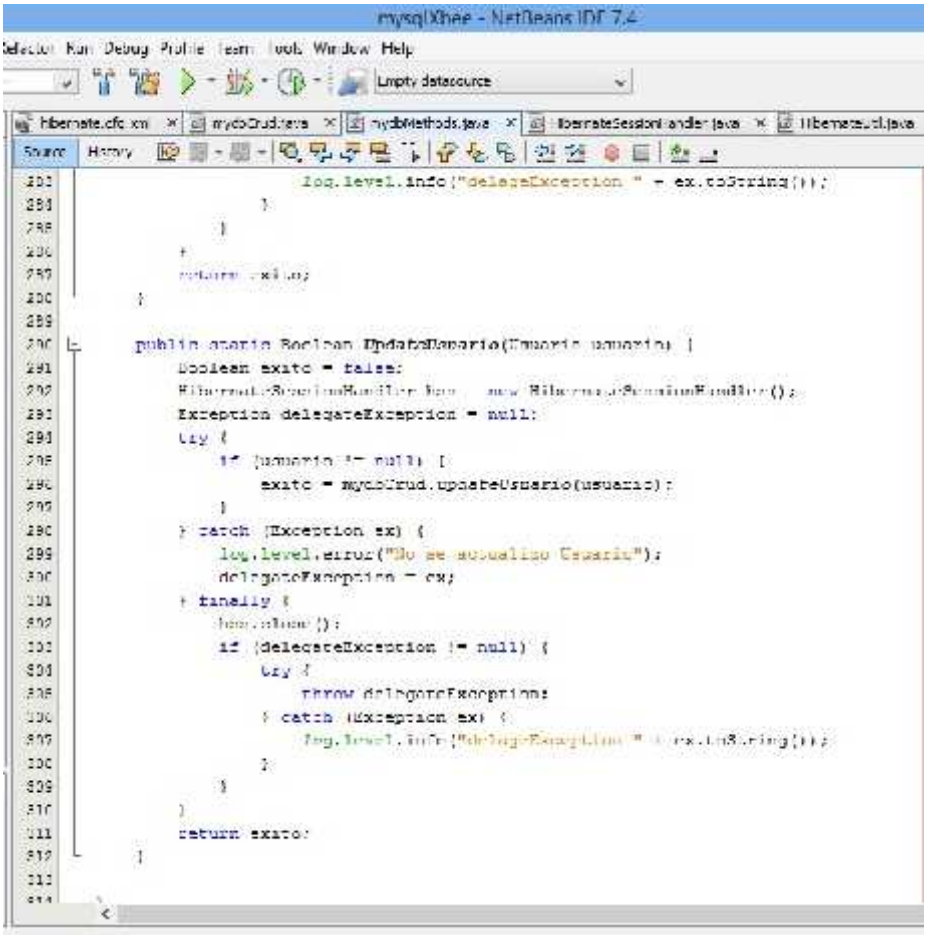
```
            }
        } catch (Exception ex) {
            log.level.error("No findUsuario" + ex.toString());
            delegateException = ex;
        } finally {
            hss.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return registro;
    }

    public static ArrayList<Registro> listaRegistrosByFechaIdUser(Date fecha1, Date fecha2, Usuario us) {
        ArrayList<Registro> registro = null;
        HibernateSessionHandler hss = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
            if (fecha1 != null && fecha2 != null) {
                registro = myDbCrud.listaRegistrosByFechaIdUser(fecha1, fecha2, us.getId_Usuario());
            }
        } catch (Exception ex) {
            log.level.error("No findUsuario" + ex.toString());
            delegateException = ex;
        } finally {
            hss.close();
            if (delegateException != null) {
                try {
```



```
                hss.close();
                if (delegateException != null) {
                    try {
                        throw delegateException;
                    } catch (Exception ex) {
                        log.level.info("delegateException " + ex.toString());
                    }
                }
            }
            return registro;
        }

        public static ArrayList<Usuario> listaUsuarios() {
            ArrayList<Usuario> infoUser = null;
            HibernateSessionHandler hss = new HibernateSessionHandler();
            Exception delegateException = null;
            try {
                infoUser = myDbCrud.listaUsuarios();
            } catch (Exception ex) {
                log.level.error("No findUsuario" + ex.toString());
                delegateException = ex;
            } finally {
                hss.close();
                if (delegateException != null) {
                    try {
                        throw delegateException;
                    } catch (Exception ex) {
                        log.level.info("delegateException " + ex.toString());
                    }
                }
            }
```



```
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return infoUser;
    }

    public static Boolean DeleteUsuario(Usuario Usuario) {
        Boolean exito = false;
        HibernateSessionHandler hse = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
            if (Usuario != null) {
                exito = myDbCrud.deleteUsuario(Usuario);
            }
        } catch (Exception ex) {
            log.level.error("No se borró el usuario");
            delegateException = ex;
        } finally {
            hse.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return exito;
    }
```



```
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return exito;
    }

    public static Boolean UpdateUsuario(Usuario usuario) {
        Boolean exito = false;
        HibernateSessionHandler hse = new HibernateSessionHandler();
        Exception delegateException = null;
        try {
            if (usuario != null) {
                exito = myDbCrud.updateUsuario(usuario);
            }
        } catch (Exception ex) {
            log.level.error("No se actualizo Usuario");
            delegateException = ex;
        } finally {
            hse.close();
            if (delegateException != null) {
                try {
                    throw delegateException;
                } catch (Exception ex) {
                    log.level.info("delegateException " + ex.toString());
                }
            }
        }
        return exito;
    }
```