ESCUELA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

"Trabajo de grado previo a la obtención del Título de Ingeniero en Electrónica y Telecomunicaciones"

TRABAJO DE GRADUACION

Título del proyecto

DISEÑO DE UN SISTEMA HIBRIDO PARA CONTROLAR EL ACCESO A LOS ESTACIONAMIENTOS UTILIZANDO TARJETAS MAGNÉTICAS Y CON TECNOLOGÍA BLUETOOTH DEL TELÉFONO CELULAR E IMPLEMENTACIÓN DE UN PROTOTIPO EN EL PARQUEADERO DE LA UNACH

Autor: (es)

Omar Stalin Tenorio Castillo

Darling Ramiro Mejía Quinteros

Director:
Ing. Anibal Llanga

8 Riobamba – Ecuador 2012

Los miembros del Tribunal de Graduación del proyecto de investigación de título:

DISEÑO DE UN SISTEMA HIBRIDO PARA CONTROLAR EL ACCESO A LOS ESTACIONAMIENTOS UTILIZANDO TARJETAS MAGNÉTICAS Y CON TECNOLOGÍA BLUETOOTH DEL TELÉFONO CELULAR E IMPLEMENTACIÓN DE UN PROTOTIPO EN EL PARQUEADERO DE LA UNACH.

Presentado por:

Omar Stalin Tenorio Castillo

Darling Ramiro Mejía Quinteros

Dirigida por:

Ing. Anibal Llanga

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Yessenia Cevallos
Presidente del Tribunal _____

Firma

Ing. Aníbal Llanga
Miembro del Tribunal _____

Firma

Ing. Fabián Gunsha
Miembro del Tribunal _____

Firma

AUTORIA DE LA INVESTIGACION

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a: Omar Stalin Tenorio Castillo, Darling Ramiro Mejía Quinteros y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

AGRADECIMIENTO

Nuestro agradecimiento muy sincero a la Universidad Nacional de Chimborazo, a la Facultad de Ingeniería y a nuestra querida escuela de Electrónica y Telecomunicaciones por habernos abierto las puertas para un feliz término de nuestra carrera

A nuestros familiares por su paciencia y su apoyo. A nuestro director de tesis Ing.

Anibal Llanga por confiar en nosotros.

DEDICATORIA

Dedico este proyecto de tesis a mi hija Lorelyn Tenorio, a mi esposa Alexandra Montenegro, a mi madre Carmen Castillo, a mi padre Galo Tenorio y a mis hermanos Galo y Vanessa Tenorio Castillo. Por estar conmigo en cada paso que doy, cuidándome y dándome fortaleza para continuar, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que me propongo sin dudar ni un solo momento en mi inteligencia y capacidad. Gracias a ellos he logrado llegar a donde estoy. Los amo con mi vida.

Omar Stalin Tenorio Castillo

DEDICATORIA

A mis padres María y Carlos por si infinito cariño, amor y sacrificio.

A mis hermanos Carlos, Paola, Jair por su apoyo incondicional en los momentos difíciles de mi carrera

A la memoria de mis más grandes inspiradores de amor, espiritualidad y carácter: mis amados abuelitos Mariana de Jesús y Luis Alfredo

Darling Ramiro Mejía Quinteros

ÍNDICE GENERAL

INDICE DE CUADROS

INDICE DE FIGURAS

RESUMEN

Con la finalidad de mejorar la seguridad, este proyecto está orientado a dar una solución de control del acceso de vehículos a las instalaciones de la Universidad Nacional de Chimborazo. Para la identificación de los usuarios se ha considerado el uso de la tecnología RFID (identificación por radiofrecuencia) y de tecnología Bluetooth existente en los teléfonos celulares para permitir el acceso al estacionamiento de la Universidad Nacional de Chimborazo a personas previamente registradas y a visitantes autorizados, dándoles prioridad a los docentes, administrativos y a los estudiantes.

Con la creación de una aplicación .JAR se controla el Bluetooth del teléfono y convertir al teléfono celular en una llave de acceso hacia los estacionamientos, la aplicación es muy intuitiva y de fácil uso, conteniendo una clave de ingreso única por cada usuario que disminuyendo de esta manera la utilización de la tarjeta que se otorgaba a los usuarios, debido que no existe el número necesario de tarjetas para la cantidad de usuarios.

El sistema cuenta con una interfaz gráfica elaborada en Labview que permitirá al administrador registrar usuarios, eliminar y limitar el número de vehículos que ingresen a los estacionamientos, además de ver los nombres, apellidos, actividad que desempeña en la institución generando una lista.

SUMMARY

In order to improve security, this project aims to provide a solution to control vehicle access to the facilities of the Universidad National de Chimborazo. For the identification of users has considered using RFID (radio frequency identification) and via existing Bluetooth technology in mobile phones allow access to the parking lot of the Universidad Nacional de Chimborazo people previously registered and authorized visitors, giving priority to teachers, administrators and students.

With the creation of an application. JAR could control the Bluetooth phone and turn the cell phone into a key access to the parking lots, the application is very intuitive and easy to use, containing a single key entry for each user that decreasing in this way the use of the card that was given to users, because there is no card number needed for the users.

The system has a graphical interface developed in Labview to allow the administrator to register users, delete, and limit the number of vehicles entering parking lots, and view the names, activity that the institution plays in generating a list.

INTRODUCCIÓN

En un mundo tecnológico como el actual, los sistemas electrónicos y en especial los de seguridad evolucionan con rapidez y ofrecen más funciones y mayores niveles de integración de nuevas tecnologías. Un control de accesos puede ser algo tan sencillo como usar una tarjeta electrónica o magnética en lugar de una llave para desbloquear una puerta de entrada, esta función básica sigue siendo una de las más importantes, en la mayoría de los casos, las empresas pueden limitar el acceso de entrada durante determinadas horas, permitiendo que todos los empleados tengan "las llaves del negocio" sin correr el riesgo de que alguien no autorizado acceda a las instalaciones fuera del horario normal. Además, los responsables de Seguridad tienen la ventaja de poder controlar e informar sobre las personas que acceden a las instalaciones de la institución.

La Universidad Nacional de Chimborazo posee un inconveniente con el control de acceso a las aéreas de estacionamiento hace falta un sistema que controle eficientemente el acceso a las aéreas de estacionamiento, llevando un registro de las personas que ingresan y salen de la institución, este registro permita identificar a la persona por su nombre y la actividad que desempeña dentro de la institución, el sistema actual no permite porque trabaja por medio de tarjeta y como no existe la cantidad necesaria para todos los usuarios, no hay un control adecuado de los usuarios.

Entre las ventajas que proporciona una solución de control de accesos, bien planificados y diseñados de forma rentable, para cualquier institución, se incluyen las siguientes:

- Protección de la instalación.
- Acceso limitado a áreas restringidas o de especial importancia para una empresa.
- Supervisión y control de las instalaciones de forma remota.
- Seguridad para empleados y visitantes.
- Control e información de situaciones anómalas.
- Creación de informes.

CAPITULO I

FUNDAMENTACIÓN TEÓRICA

1.1 GENERALIDADES DE LA TECNOLOGÍA BLUETOOTH

Bluetooth es un estándar de facto global que identifica un conjunto de protocolos que facilitan la comunicación inalámbrica entre diferentes tipos de dispositivos electrónicos. Su nombre viene del rey vikingo, Harald Bluetooth (940 A.D.981A.D.), famoso por su habilidad para la comunicación, y para hacer que la gente hablara entre ella.

Bluetooth es una tecnología de comunicación inalámbrica de corto alcance, que permite conectividad inalámbrica entre dispositivos remotos. Diseñada pensando básicamente en tres objetivos: pequeño tamaño, mínimo consumo y bajo precio, convirtiéndose en una tecnología ideal para la conexión de dispositivos de bajas prestaciones (móviles, cámaras de fotos, auriculares manos libres, impresoras,…).

Bluetooth fue introducido por Ericsson, IBM, Intel, Nokia y Toshiba a comienzos de 1998. Estas compañías posteriormente formaron un grupo de interés especial conocido como el Bluetooth SIG. El lanzamiento de las especificaciones de Bluetooth 1.0 se realizó el 26 de julio de 1999, pero sólo recientemente la tecnología es suficientemente económica como para ser usada de manera generalizada.

1.1.1 BANDA DE FRECUENCIA

Bluetooth permite la transferencia de información a través de la interfaz de aire y opera en una de las bandas Industrial-Científico-Médica (ICM) que es de libre de licencia y disponible mundialmente, tiene limitaciones según el ente regulador de cada país. La banda ICM comienza desde 2400,0 MHz hasta 2483,5 MHz y se encuentra distribuida como se muestra en la figura 1.1

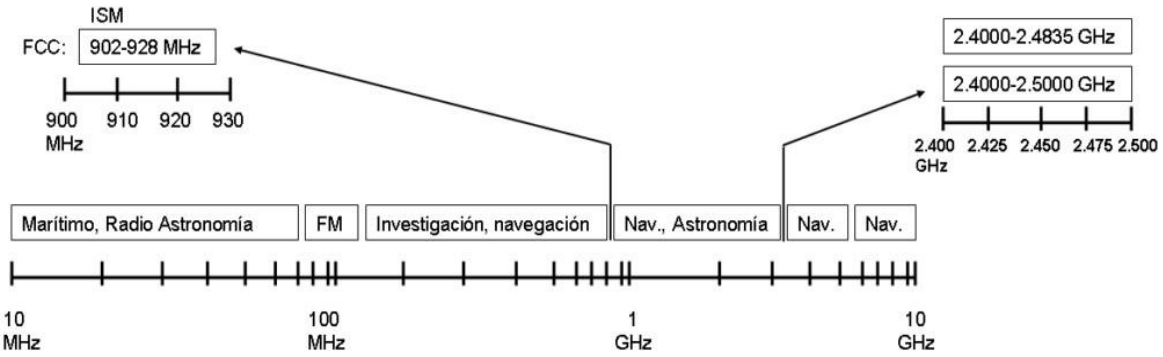


Figura 1.1 Localización de la banda ISM de 2.4 GHz

1.1.2 SALTOS DE FRECUENCIAS Y EL CANAL

Salto de Frecuencia: Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar las múltiples interferencias que se pudieran producir. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia.

En este caso la técnica de salto de frecuencia es aplicada a una alta velocidad y una corta longitud de los paquetes (1600 saltos/segundo). Con este sistema se divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto de manera pseudo-aleatoria. Para que dos radios Bluetooth puedan comunicarse entre sí, ambos deberán estar trasmitiendo o recibiendo información en el mismo canal. Esto implica que ambos dispositivos sigan un mismo patrón de salto de frecuencia, permitiendo que estén sincronizados. Debido que el radio no está estático en una frecuencia, FHSS provee una inmunidad natural a la interferencia.

Canal: Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo duplex), en el que el canal queda dividido en intervalos de 625 μs, llamados slots, donde cada salto de frecuencia es ocupado por un slot. Dos o más unidades Bluetooth pueden compartir el mismo canal dentro de una piconet (pequeña red que establecen automáticamente los terminales Bluetooth para comunicarse entre sí), donde una unidad actúa como maestra, controlando el tráfico de datos en la piconet que se genera entre las demás unidades, donde éstas actúan como esclavas, enviando y recibiendo señales hacia el maestro.

El salto de frecuencia del canal está determinado por la secuencia de la señal, es decir, el orden en que llegan los saltos y por la fase de esta secuencia. En Bluetooth, la secuencia queda fijada por la identidad de la unidad maestra de la piconet (un código único para cada equipo), y por su frecuencia de reloj.

Localización	Ancho de Banda	Canales RF
USA, Europa y la mayoría de países	2.4-2.4835 GHz	F=2402+k MHz, k=0.78
Francia	2.4465-2.4835 GHz	F=2453+k MHz, k=0.22

Tabla 1 Frecuencia de operación Bluetoooth

Los 79 canales RF se organizan por números, de 0 a 78, con un espacio de 1 MHz entre ellos, empezando por 2402 GHz. El método de compartición de la frecuencia se basa en Frecuency Hopping-Time División Duplex (FH-TDD7).

1.1.3 CLASE DE TRANSMISORES

El estándar Bluetooth define 3 clases de transmisores, cuyo alcance varía en función de su potencia radiada como se indica en la tabla 2

Clase	Potencia(pérdida de señal)	Alcance
I	100 mW (20 dBm)	100 metros
II	2,5 mW (4 dBm)	15-20 metros
II	1 mW (0 dBm)	10 metros

Tabla 2 Clase de transmisores Bluetooth

1.1.4 PILA DE L PROTOCOLO BLUETOOTH

La pila se encuentra constituida por varias capas las cuales se pueden organizar en los siguientes grupos como: grupos de transporte, protocolos middleware y grupo de aplicación.

Los datos en la pila fluyen a través de todas las capas a excepción de la información de audio, que va directamente desde la banda base hacia la aplicación con alto grado de prioridad, para garantizar la calidad de servicio en tiempo real, esperada en aplicaciones de audio. En la Figura 1.2 se indica como está formada la pila del protocolo Bluetooth.

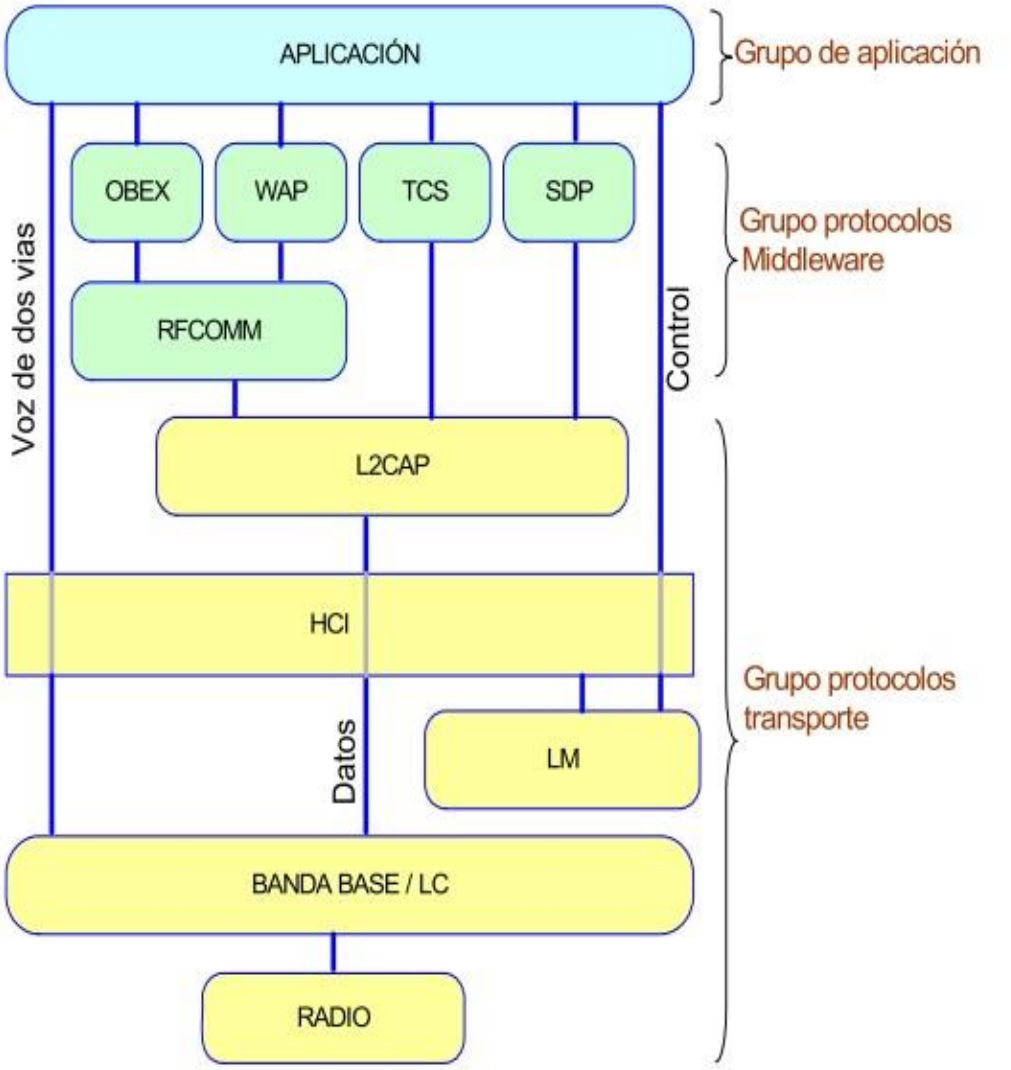


Figura 1.2 Pila de Protocolos Bluetooth

A continuación se detalla cada bloque que forman la pila del protocolo bluetooth desde el bloque de radio hasta aplicación.

- Radio: Modula y demodula los datos para la transmisión y recepción a través del aire.

- Banda base y controlador de enlace (LC): Controlan los enlaces físicos vía radio, ensamblando paquetes y generando el salto de frecuencia.

- Manejador de enlace (LM): Controla y configura los enlaces con otros dispositivos.

- Interfaz controladora de host (HCI): Lleva las comunicaciones entre un módulo bluetooth y un host separados, permitiéndole a este último acceder a las capacidades de hardware del módulo.

- Protocolo de adaptación y enlace lógico (L2CAP): Distribuye y acondiciona el tamaño de paquetes para las capas altas.

- RFCOMM: Suministra una interfaz serial similar al RS232.

- Protocolo de descubrimiento de servicio: Permite descubrir los servicios brindados por otros dispositivos.

- Protocolo de control de telefonía (TCS): Suministra servicios de telefonía.

- OBEX, WAP: Suministra otros protocolos de comunicación a las capas altas de ser requerido.

- Aplicación: Programa que hace uso de la pila bluetooth en una forma determinada.

1.1.5 DATAGRAMA BLUETOOTH

Bluetooth define el uso de dos tipos de paquetes: SCO y ACL.

- En Bluetooth todos los datos a transmitirse a través del canal son fragmentados y enviados en paquetes.

- La información se encuentra protegida mediante códigos detectores y/o correctores de errores. En cada ranura solo se envía un paquete. El receptor los recibirá y los procesará empezando por el bit menos significativo.

- La velocidad de transmisión aérea bruta es de 1 Mbps en el modo de transferencia básica**.**

Figura 1.3 Paquete Bluetooth

El paquete está conformado por el código de acceso (72 bits) el cual realiza las siguientes funciones:

- Identifica si el paquete proviene o es enviado a un maestro.

- En una Piconet se compara las señales de llegada con el código de acceso, si las dos no coinciden, el paquete no es considerado válido sobre el canal el resto del contenido es ignorado.

- El código de acceso también es usado para sincronización y compensación del offset.

Paquete cabecera (54 bits)

- Reconocimiento (ACK) de paquetes enviados

- Identificación del paquete en envíos desordenados

- Control de flujo. Retención de envío en un extremo.

- Dirección de destino del paquete.

- Control de errores para la cabecera.

Carga útil (0-2745 bits)

- Puede contener campos de voz, de datos o de ambos.

- Un paquete puede ocupar más de un slot. Tiene un CRC (Código de redundancia cíclica) de 16 bit.

- Los paquetes SCO no tienen CRC y un tamaño de payload (carga útil) fijo de 30 bytes.

1.1.6 TOPOLOGÍAS BLUETOOTH

Para conocer las topologías bluetooth debemos conocer los conceptos de maestro y esclavo.

Maestro: El dispositivo que inicia el enlace se le denomina maestro y es el que establece el patrón de salto de frecuencia que le permite a los dos dispositivos sincronizarse en el enlace de aire.

Esclavo: Es el dispositivo que espera instrucciones.
Las topologías que se pueden establecer entre dispositivos bluetooth son:

- Piconet
- Scatternet.

Piconet: se compone de dos o más dispositivos Figura 1.8, que ocupan el mismo canal físico, por lo que están sincronizados con un mismo reloj y secuencia de salto.

El reloj que utiliza en el conjunto de la piconet es idéntico al reloj Bluetooth de uno de los dispositivos que la conforman, el llamado dispositivo maestro.
En cuanto a la secuencia de salto, ésta se deriva del reloj y de la dirección del dispositivo bluetooth que actúa como maestro. El resto de dispositivos sincronizados recibe el nombre de esclavos.

En la misma ubicación puede haber varias piconets distintas. Cada una tendrá un canal físico diferente, es decir, un dispositivo maestro, un reloj y una secuencia de salto independientes.

Un dispositivo bluetooth puede utilizarse simultáneamente en dos o más piconets mediante un multiplexado por división de tiempo. Ahora bien, este dispositivo bluetooth no actuará nunca como maestro en más de una piconet.

Una piconet puede constar de hasta siete dispositivos activos. Además de estos dispositivos activos, la piconet puede contar con muchos más esclavos en estado de espera.

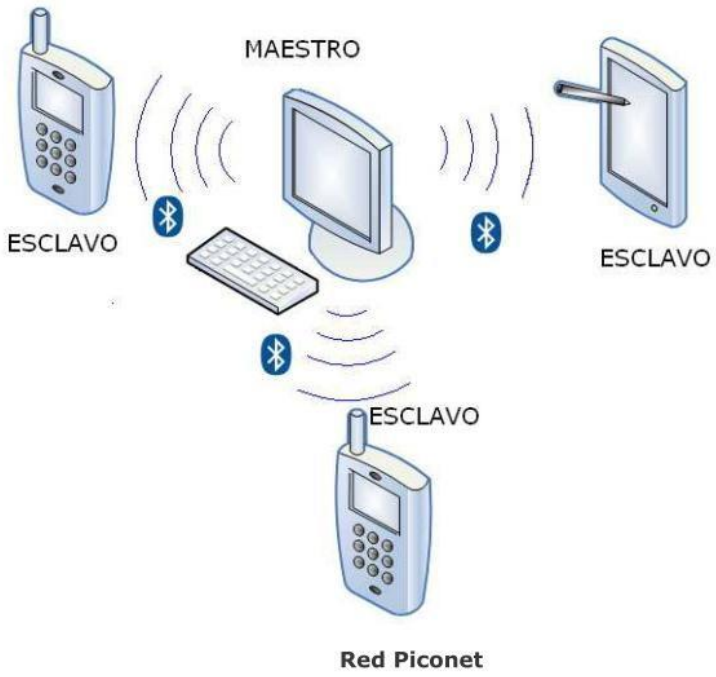


Figura 1.4 Piconet

Scatternet: está conformada por dos o más piconets traslapadas en tiempo y espacio; esto significa que un dispositivo bluetooth puede ser al mismo tiempo un maestro en una piconet y un esclavo en otra.

Cada piconet en una scatternet tiene su propio patrón de salto de frecuencia e igual que en una piconet, los dispositivos pueden entrar y salir de ellas de forma dinámica (redes adhoc).

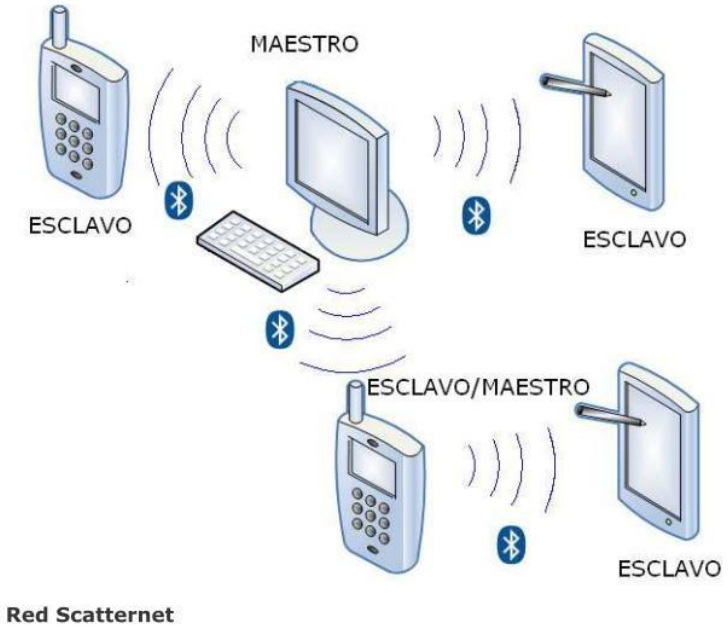


Figura 1.5 Scatternet

1.1.7 ENLACES FÍSICOS BLUETOOTH

Se pueden establecer dos tipos de enlaces entre un maestro y uno o más esclavos: orientado a la conexión y sin conexión.

Los enlaces orientados a la conexión requieren que se establezca una sesión antes de que pueda enviarse algún dato. Con redes así, se garantiza que los datos lleguen en el mismo orden enviado, estos enlaces también son llamados síncrono orientado a la conexión (SCO, Synchronous Connection-Oriented).

Los enlaces sin conexión también conocidos como Connection-Oriented) y asíncrono sin conexión (ACL, Asynchronous Connectionless) no requieren que se establezca

una sesión entre un emisor y un receptor con este tipo de redes pueden que los datos no lleguen en el mismo orden que se enviaron.

1.1.8 CONEXIÓN ENTRE DISPOSITIVOS BLUETOOTH

Los estados que se emplean para realizar una comunicación bluetooth con sus respectivos tiempos de duración de pasar de un estado a otro se detallan en la Fig. 1.6.

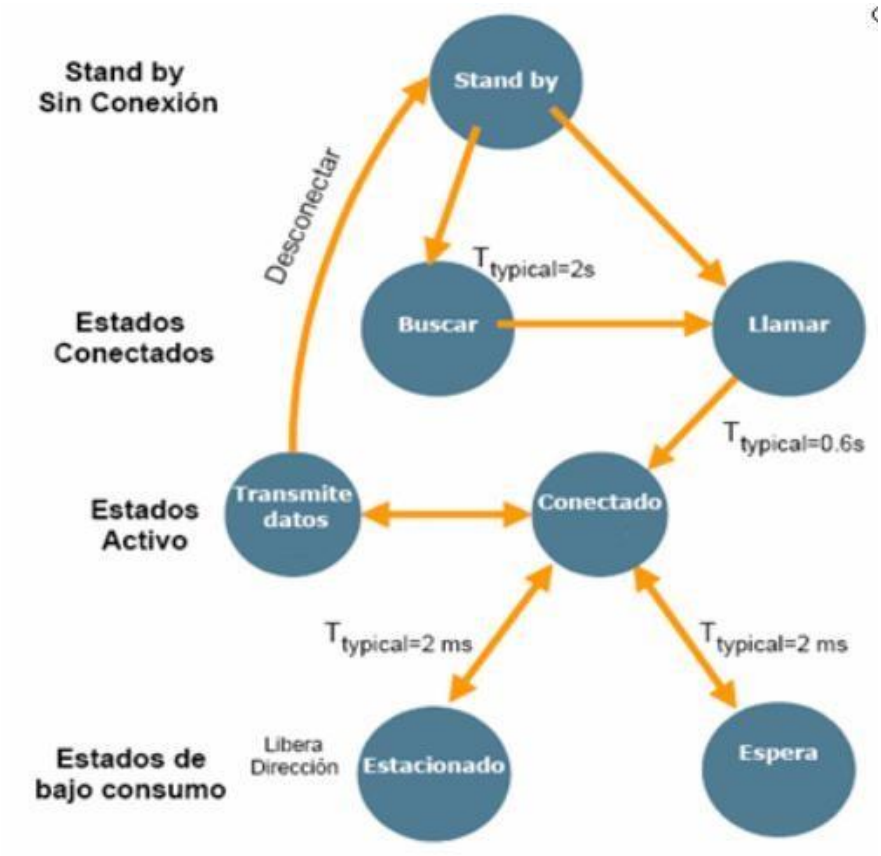


Figura 1.6 Estados para la comunicación Bluetooth

- **Stand by**.- Esperando para unirse a una piconet.
- **Buscar**.- Preguntar por radios para conectar.
- **Llamar**.-Conectara un radio en específico.
- **Conectado**.- Activo en una piconet(maestro o esclavo).
- **Estacionado/Espera** .-Estados conectados con bajo consumo de energía.

1.1.9 MODULACIÓN UTILIZADA EN BLUETOOTH

Se definen dos modos de modulación: Un modo obligatorio, llamado modo de transferencia básica, que usa una modulación de frecuencia binaria para reducir al mínimo la complejidad del transmisor/receptor. Un modo opcional, llamado de transferencia de datos mejorada, que usa modulación PSK y cuenta con dos variantes: π/4-DQPSK y 8DPSK. La tasa de transferencia de símbolos de todas las secuencias de modulación es de 1 Mbps. La velocidad de transmisión aérea total es de 1 Mbps con el modo de transferencia básica; 2 Mbps con la transferencia de datos mejorada y π/4-DQPSK; y 3 Mbps con la transferencia de datos mejorada y 8DPSK. Para la transmisión bidireccional se emplea una técnica de dúplex por división de tiempo (TDD) en ambos modos.

Esta especificación define los requisitos de una radio Bluetooth, tanto para el modo de transferencia básica como de transferencia mejorada de datos.

1.1.10 SEGURIDAD EN LAS TRANSMISIONES BLUETOOTH

Para asegurar la protección de la información se ha definido un nivel básico de encriptación, incluida en el diseño del clip de radio para proveer de seguridad en equipos que carezcan de capacidad de procesamiento, las principales medidas de seguridad son:

- Una rutina de pregunta-respuesta, para autentificación.
- Una corriente cifrada de datos, para encriptación.
- Generación de una clave de sesión (que puede ser cambiada durante la conexión).

Tres entidades son utilizadas en los algoritmos de seguridad: la dirección de la unidad Bluetooth, que es una entidad pública; una clave de usuario privada, como una entidad secreta; y un número aleatorio, que es diferente por cada nueva transacción. Como se ha descrito anteriormente, la dirección Bluetooth se puede obtener a través de un procedimiento de consulta. La clave privada se envía durante la inicialización

y nunca es revelada. El número aleatorio se obtiene de un proceso pseudo-aleatorio en la unidad Bluetooth.

1.2 JAVA 2 MICRO EDITION: JSR 82 BLUETOOTH

1.2.1 INTRODUCCIÓN

La empresa Sun Microsystems lanzó a mediados de los años 90 el lenguaje de programación Java que, aunque en un principio fue diseñado para generar aplicaciones que controlaran electrodomésticos como lavadoras, frigoríficos, etc, debido a su gran robustez e independencia de la plataforma donde se ejecutase el código, desde sus comienzos se utilizó para la creación de componentes interactivos integrados en páginas Web y programación de aplicaciones independientes, estos componentes se denominaron applets. Con los años, Java ha progresado enormemente en varios ámbitos como servicios HTTP, servidores de aplicaciones, acceso a bases de datos (JDBC)... Como vemos Java se va adaptando a las necesidades tanto de los usuarios como de las empresas ofreciendo soluciones y servicios tanto a unos como a otros.

Debido a la explosión tecnológica de estos últimos años Java ha desarrollado soluciones personalizadas para cada ámbito tecnológico. Sun ha agrupado cada uno de los ámbitos en una edición distinta de su lenguaje Java. Estas ediciones son Java 2 Stantard Edition, orientada al desarrollo de aplicaciones independientes y de applets, Java 2 Enterprise Edition, enfocada al entorno empresarial y Java 2 Micro Edition, orientada a la programación de aplicaciones para pequeños dispositivos.

1.2.2 ANÁLISIS COMPARATIVO

Sun, dispuesto a proporcionar las herramientas necesarias para cubrir las necesidades de todos los usuarios, creó distintas versiones de Java de acuerdo a las necesidades de cada uno. El paquete Java 2 lo podemos dividir en 3 ediciones distintas. J2SE (Java Standard Edition) orientada al desarrollo de aplicaciones independientes de la plataforma, J2EE (Java Enterprise Edition) orientada al entorno empresarial y J2ME (Java Micro Edition)

orientada a dispositivos con capacidades restringidas. Veamos cuáles son las características de cada una de las versiones:

a) Java 2 Platform, Standard Edition (J2SE): Esta edición de Java es la que en cierta forma recoge la iniciativa original del lenguaje Java. Tiene las siguientes características:

- Inspirado inicialmente en C++, pero con componentes de alto nivel, como soporte nativo de strings y recolector de basura.
- Código independiente de la plataforma, precompilado a bytecodes intermedio y ejecutado en el cliente por una JVM (Java Virtual Machine).
- Modelo de seguridad tipo sandbox proporcionado por la JVM.
- Abstracción del sistema operativo subyacente mediante un juego completo de APIs de programación.

Esta versión de Java contiene el conjunto básico de herramientas usadas para desarrollar Java Applets, así como las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, etc.

b) Java 2 Platform, Enterprise Edition (J2EE): Esta versión está orientada al entorno empresarial. El software empresarial tiene unas características propias marcadas como ejecución sobre una red de ordenadores de manera distribuida y remota mediante EJBs (Enterprise Java Beans). De hecho, el sistema se monta sobre varias unidades o aplicaciones. En muchos casos, además, el software empresarial requiere que se sea capaz de integrar datos provenientes de entornos heterogéneos. Esta edición está orientada especialmente al desarrollo de servicios web, servicios de nombres, persistencia de objetos, XML, autenticación, APIs para la gestión de transacciones, etc. El cometido de esta especificación es ampliar la J2SE para dar soporte a los requisitos de las aplicaciones de empresa.

c) Java 2 Platform, Micro Edition (J2ME): Esta versión de Java está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones, como el uso de una máquina virtual denominada KVM (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en vez del uso de la JVM clásica, inclusión de un pequeño y rápido recolector de basura y otras diferencias que ya iremos viendo más adelante.

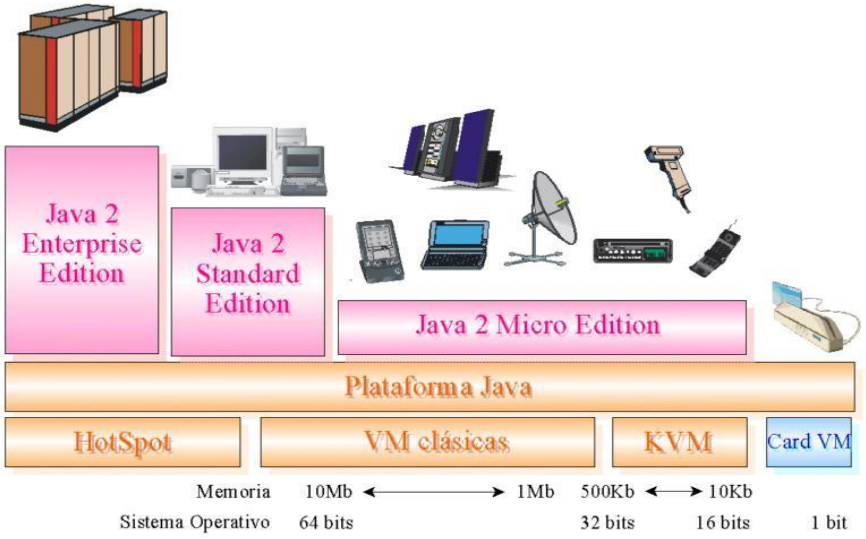


Figura 1.7 Arquitectura de la plataforma Java 2 de Sun

J2ME representa una versión simplificada de J2SE. Sun separó estas dos versiones ya que J2ME estaba pensada para dispositivos con limitaciones de proceso y capacidad gráfica. También separó J2SE de J2EE porque este último exigía unas características muy pesadas o especializadas de E/S, trabajo en red, etc. Por tanto, separó ambos productos por razones de eficiencia. Hoy, J2EE es un superconjunto de J2SE pues contiene toda la funcionalidad de éste y más características, así como J2ME es un subconjunto de J2SE (excepto por el paquete javax.microedition, conteniendo varias limitaciones con respecto a J2SE.

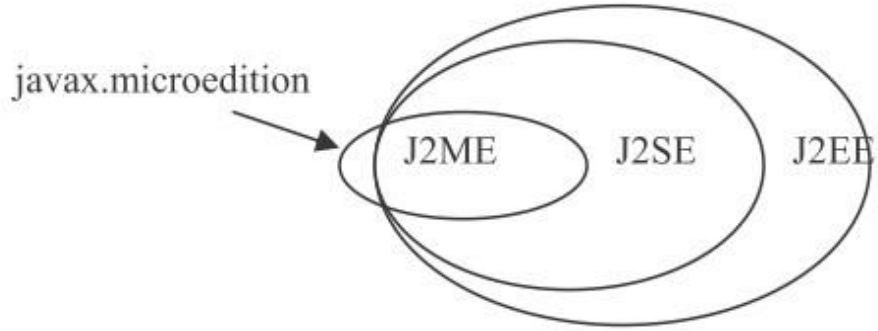


Figura 1.8 Relación entre las APIs de la plataforma Java.

De manera muy general se puede considerar a J2ME y J2EE como versiones reducidas y ampliadas de J2SE respectivamente pero en realidad cada una de las ediciones está enfocada a ámbitos de aplicación muy distintos. Las necesidades computacionales y APIs de programación requeridas para un juego ejecutándose en un móvil difieren bastante con las de un servidor distribuido de aplicaciones basado en EJB.

1.2.3 NOCIONES DE J2ME

Los componentes que forman parte de esta tecnología son:

- Por un lado tenemos una serie de máquinas virtuales Java con diferentes requisitos, cada una para diferentes tipos de pequeños dispositivos.

- **Configuraciones,** que son un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones para dispositivos de características específicas. Existen 2 configuraciones definidas en J2ME: Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria, y Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

- **Perfiles,** son bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Un entorno de ejecución determinado de J2ME se compone entonces de una selección de:

a) Máquina virtual.
b) Configuración.
c) Perfil.
d) Paquetes Opcionales.

La arquitectura de un entorno de ejecución la podemos ver en la Figura 1.9.

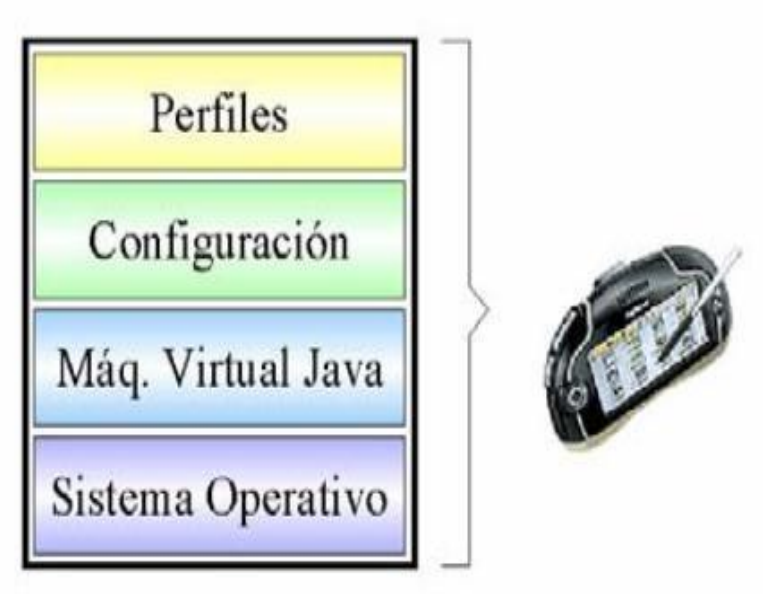


Figura 1.9 Entorno de ejecución J2ME

1.2.4 MÁQUINAS VIRTUALES J2ME

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente.

Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Como existen 2 configuraciones CLDC y CDC, cada una con unas características propias que veremos en profundidad más adelante. Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM.

Existen disponibles dos máquinas virtuales de Java2 ME con diferentes requisitos, cada una pensada para tipos distintos de pequeños dispositivos:

- **KVM,** o Kilobyte Virtual Machine, se corresponde con la máquina virtual más pequeña desarrollada por Sun. Se trata de una implementación de máquina virtual reducida y orientada a dispositivos de 16 o 32 bits con al menos 25 Mhz de velocidad y hasta 512 Kb de memoria total disponible.

- **CVM**, o Compact Virtual Machine, soporta las mismas características que la Máquina Virtual de Java SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM.

1.2.5 CONFIGURACIÓN CLDC

Recordemos que la configuración es un mínimo grupo de APIs útiles para desarrollar las aplicaciones destinadas a un amplio rango de dispositivos. Estas APIs tienen como función describir las características comunes de grupos de dispositivos como son:

- Características que son soportadas de Java.
- Características que soporta la Máquina Virtual Java correspondiente a las cualidades del dispositivo.
- Conjunto de bibliotecas básicas de Java y APIs soportadas.

J2ME posee dos configuraciones diferentes: CLDC (Conected Limited Device Configuration) y CDC (Conected Device Configuration). Cada tipo de configuración tiene un objetivo distinto. CDC está concebida para dispositivos con más potencia de cálculo y de memoria. Está basada en J2SE e incorpora muchas de sus clases. En cambio, CLDC está concebida para dispositivos que tienen menos recursos.

Los requisitos mínimos de hardware que contempla CLDC son:

- 160KB de memoria disponible para Java
- Procesador de 16 bits
- Consumo bajo de batería
- Conexión a red

Los dispositivos que claramente encajan dentro de este grupo son los teléfonos móviles, las PDA, los Pocket PC, etc.

En cuanto a los requisitos de memoria, según CLDC, los 160KB se utilizan de la siguiente forma:

- 128KB de memoria no volátil para la máquina virtual Java y para las librerías del APIde CLDC

- 32KB de memoria volátil, para sistema de ejecución (Java Runtime System).

1.2.6 PERFILES PARA CLDC

Podemos definir un perfil como un conjunto de APIs orientado a cubrir más específicamente las necesidades de funcionalidad de un grupo de dispositivos que comparten una serie de objetivos, por ejemplo, teléfonos móviles, PDA, electrodomésticos, etc.

En la definición del perfil tiene gran importancia la interfaz gráfica, se pueden encontrar grandes diferencias, por ejemplo, entre la gran pantalla táctil de una PDA y la pantalla mucho más pequeña que presenta un teléfono móvil. Por ello, se puede concluir que dependiendo del tipo de configuración, encontraremos con distintos tipos de perfiles. Así tenemos:

Configuración de perfiles para la configuración CLDC:

- PDA Profile
- Mobile Information Device Profile (MIDP)

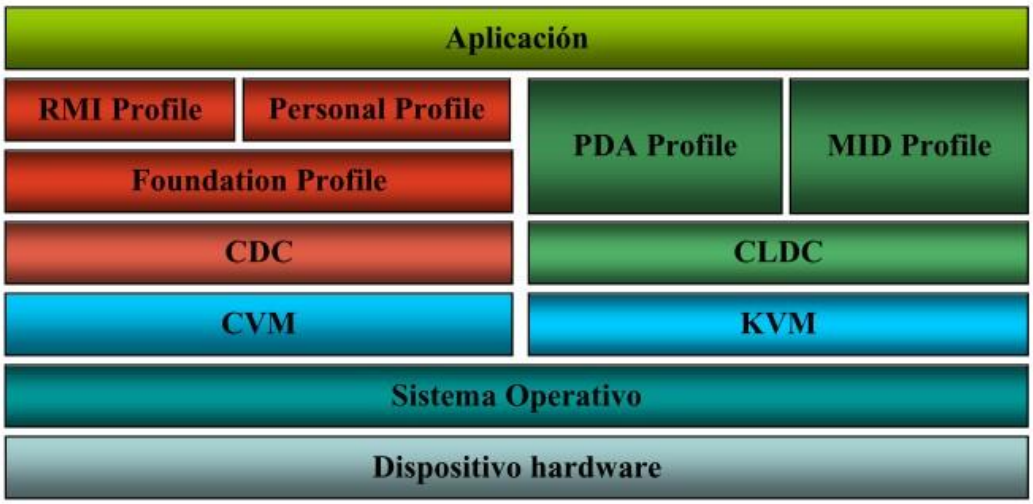


Figura 1.10 Perfiles MIDP existentes

PDA Profile: Está construido sobre CLDC. Pretende abarcar PDAs de gama baja, tipo Palm, con una pantalla y algún tipo de puntero (ratón o lápiz) y una resolución de al menos 20000 píxeles (al menos 200x100 píxeles) con un factor 2:1. Mobile Information Device Profile (MIDP): Este perfil está construido sobre la configuración.

MIDP: Al igual que CLDC fue la primera configuración definida para J2ME, MIDP fue el primer perfil definido para esta plataforma. Este perfil está orientado para dispositivos con reducida capacidad computacional y de memoria, conectividad limitada
(en torno a 9600 bps),

capacidad gráfica muy reducida (mínimo un display de 96x54 píxeles monocromo) y entrada de datos alfanumérica reducida. Los tipos de dispositivos que se adaptan a estas características son: teléfonos móviles, buscapersonas (pagers) o PDAs de gama baja con conectividad.

Actualmente, este perfil tiene dos versiones, la última de las cuales proporciona APIs para los nuevos dispositivos, tales como los últimos teléfonos.

El perfil MIDP: incluye una serie de paquetes en los cuales están definidas las clases e interfaces que podemos usar en una aplicación. Cada paquete está especializado en una labor concreta, cada una de las clases e interfaces que contiene está pensado para llevar a cabo estas labores. En la tabla 3. se describe la funcionalidad de cada uno de los paquetes que conforman el perfil MIDP.

Paquetes MIDP	Descripción
javax.microedition.lcdui	Clases e interfaces para GUI
javax.microedition.rms	Soporte para el almacenamiento persistente
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases e interfaces de conexión genérica
java.io	Clases y paquete estándar de E/S
java.lang	Clases e interfaces de la MV.
java.util	Clases, interfaces y utilidades estándar

Tabla 3 Paquetes característicos de MIDP

1.2.7 MODELO BÁSICO DE UN APLICACIÓN J2ME

El objetivo final de la creación de una aplicación (MIDlet) en J2ME es la ejecución de ésta en un dispositivo, pero dicha ejecución implica una serie de pasos previos de los cuales se debe encargar el dispositivo en el cual se pretenda ejecutar. Las acciones que un dispositivo debe realizar antes de lanzar un MIDlet y que son condición necesaria para que funcione son:

- Poder localizar el MIDlet
- Poder descargar el MIDlet
- Proporcionar un medio para almacenar el MIDlet
- Gestionar el MIDlet

Para realizar estos pasos todo dispositivo que soporte CLDC y MIDP implementa un programa residente en memoria que es el encargado de hacerse cargo de todas estas tareas. Este programa se conoce como Gestor de aplicaciones o AMS (Application Management Software). Al crear un MIDlet se debe tener en cuenta que estamos trabajando en

dispositivos de unas características muy limitadas. Eso implica realizar una serie de operaciones previas para que la máquina virtual de Java pueda interpretar y ejecutar la aplicación.

Por todo esto, tendremos que las fases que debemos seguir en el desarrollo de nuestro proyecto son tal y como aparecen en la figura que mostramos a continuación:

Desarrollo del código Fuente

- Compilación
- Pre-verificación
- Empaquetamiento
- Ejecución
- Depuración

1.2.7.1 ¿QUE ES UN MIDlet?

Un MIDlet es una aplicación J2ME desarrollada sobre el perfil MID, aplicaciones para dispositivos móviles cuyas limitaciones caen dentro de la especificación MIDP. Gracias a la filosofía Java ("write one, run anywhere") podemos ejecutarlas sobre un amplio rango de dispositivos sin realizar ninguna modificación.

Para que esta portabilidad sea realidad la especificación MIDP ha definido los siguientes requisitos:

- Todos los dispositivos de información móviles deben contar con un módulo software encargado de la gestión de los MIDlets (cargarlos, ejecutarlos…). Este software es denominado gestor de aplicaciones1.

- Todos los MIDlet deben ofrecer la misma interfaz a todos los gestores de aplicaciones. Así, independientemente de la funcionalidad interna que implementen (un juego, una agenda,…), a los dispositivos pueden identificar a los MIDlet y realizar acciones sobre ellos. Este comportamiento se consigue mediante

el mecanismo de herencia: todos los MIDlets heredan de la misma clase, javax.microedition.midlet.MIDlet.

1.2.7.2 GESTOR DE APLICACIONES

El gestor de aplicaciones es el software encargado de gestionar los MIDlets. Este software reside en el dispositivo y permite ejecutar, pausar o destruir nuestras aplicaciones J2ME.

En la especificación no se índica que implementación concreta debe tener un gestor de aplicaciones. En su lugar, se describen las principales funciones que debe realizar, siendo finalmente los fabricantes quienes se encarguen de desarrollar gestores de aplicaciones específicos para sus dispositivos.

El gestor de aplicaciones realiza dos grandes funciones:

- Gestión de las aplicaciones.
- Gestión de estados de las aplicaciones.

Gestión de las aplicaciones: Las principales funciones que realiza un gestor de aplicaciones son carga, instalación, ejecución, actualización y desinstalación del MIDlet.

Gestión de estados del MIDlet: Los gestores de aplicaciones adaptan el funcionamiento de los MIDlet a los entornos concretos de ejecución de los dispositivos móviles. Los dispositivos se dedican a otras actividades además de ejecutar aplicaciones (por ejemplo por una llamada telefónica, enviar sms…).

La interacción del entorno de ejecución con los MIDlet se realiza mediante llamadas que el gestor de aplicaciones realiza a unas funciones predeterminadas comunes a todos los MIDlet. Estas llamadas permiten que el MIDlet pase de un estado de ejecución a otro de forma controlada, manteniendo la integridad de la información y siempre de acuerdo a unas transiciones establecidas en el denominado ciclo de vida de los MIDlet. En cada una de estas transiciones el gestor de aplicaciones es el encargado de almacenar el estado de los MIDlet.

1.2.8 CICLO DE VIDA DE UN MIDlet

El ciclo de vida de un MIDlet pasa por 5 fases como se puede observar en la Fig. 1.11 descubrimiento, instalación, ejecución, actualización y borrado.

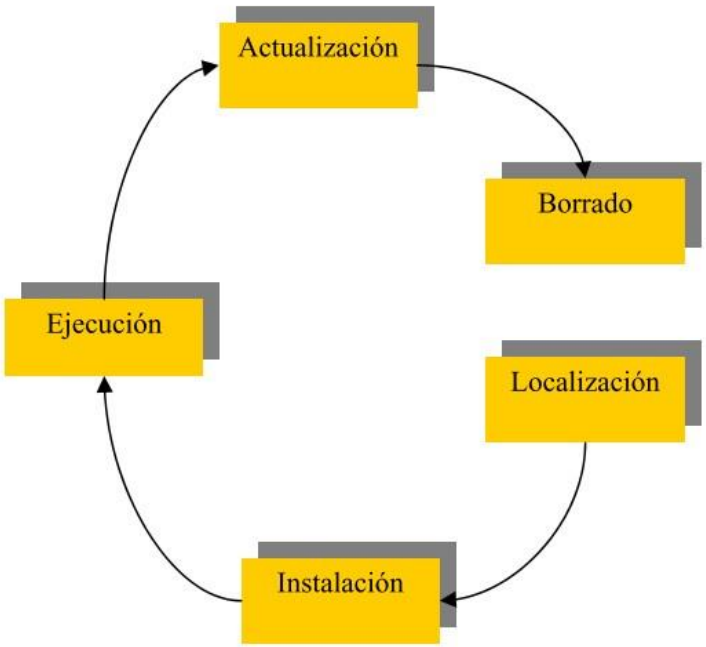


Figura 1.11 Estado de vida de un MIDlet

El AMS es el encargado de gestionar cada una de estas fases de la siguiente manera**:**

1. **Descubrimiento:** Es la etapa previa a la instalación del MIDlet y es dónde seleccionamos a través del gestor de aplicaciones, la aplicación a descargar. Por tanto, el gestor de aplicaciones proporcionara los mecanismos necesarios para realizar la elección del MIDlet a descargar. El AMS puede ser capaz de realizar la descarga de aplicaciones de diferentes maneras, dependiendo de las capacidades del dispositivo.

2. **Instalación:** Una vez descargado el MIDlet en el dispositivo, comienza el proceso de instalación. En esta fase el gestor de aplicaciones controla todo el proceso informando al usuario la evolución de la instalación y si existiese algún problema durante ésta. Cuándo un MIDlet está instalado en el dispositivo, todas sus clases, archivos y almacenamiento persistente están preparados y listos para su uso.

3. **Ejecución:** Mediante el gestor de aplicaciones vamos a ser capaces de iniciar la ejecución de los MIDlets. En esta fase, el AMS tiene la función de gestionar los estados del MIDlet en función de los eventos que se produzcan durante esta ejecución. Esto lo veremos un poco más en profundidad más adelante.

4. **Actualización:** El AMS tiene que ser capaz de detectar después de una descarga si el MIDlet descargado es una actualización de un MIDlet ya presente en el dispositivo. Si es así, nos tiene que informar de ello, además permitir decidir si queremos realizar la actualización pertinente o no.

5. **Borrado:** En esta fase el AMS es el encargado de borrar el MIDlet seleccionado del dispositivo. El AMS pedirá confirmación antes de proceder a su borrado e informará de cualquier circunstancia que se produzca.

Hay que indicar que el MIDlet puede permanecer en el dispositivo todo el tiempo que queramos. Después de la fase de instalación, el MIDlet queda almacenado en una zona de memoria persistente del dispositivo MID.

El usuario de éste dispositivo es el encargado de decidir en qué momento quiere eliminar la aplicación y haciéndole saber al AMS mediante alguna opción que éste suministre.

1.2.9 ESTADOS DE UN MIDlet

Un MIDlet durante su ejecución pasa por 3 estados diferentes. Como ya hemos visto en el apartado anterior, estos tres estados son:

- Activo: El MIDlet está actualmente en ejecución.

- Pausa: El MIDlet no está actualmente en ejecución. En este estado el MIDlet o debe usar ningún recurso compartido. Para volver a pasar a ejecución tiene que cambiar su estado a Activo.

- Destruido: El MIDlet no está en ejecución ni puede transitar a otro estado. Además se liberan todos los recursos ocupados por el MIDlet.

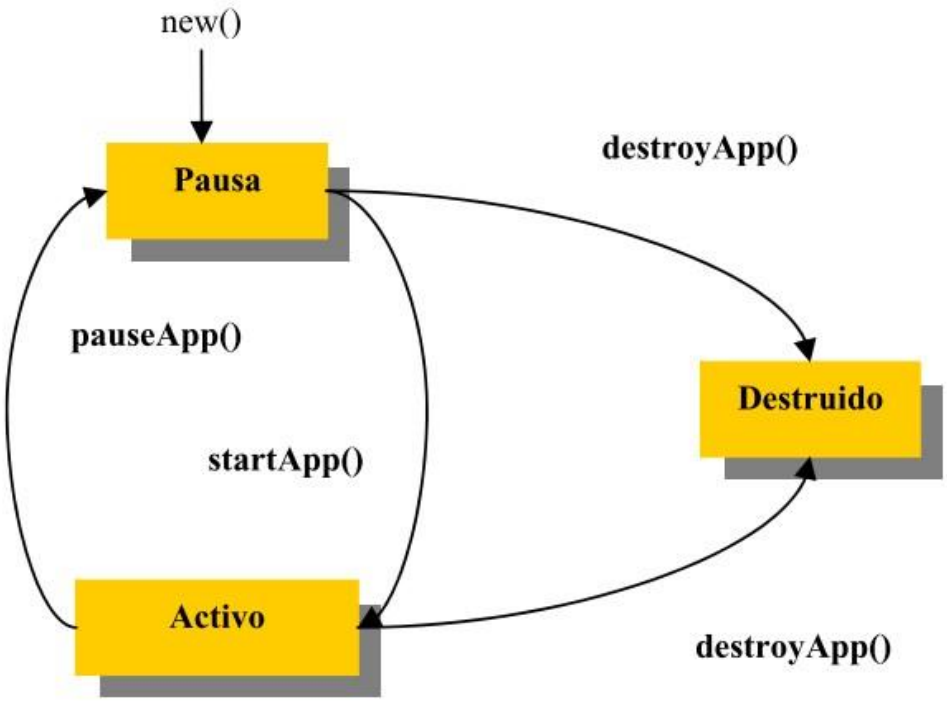


Figura 1.12 Estados de un MIDlet en ejecución

Como vemos en el diagrama, un MIDlet puede cambiar de estado mediante una llamada a los métodos MIDlet.startApp(), MIDlet.pauseApp() o MIDlet.destroyApp(). El gestor de aplicaciones cambia el estado de los MIDlets haciendo una llamada a cualquiera de los métodos anteriores. Un MIDlet también puede cambiar de estado por sí mismo.

Los estados que pasa un MIDlet durante una ejecución típica y cuáles son las acciones que realiza tanto el AMS como el MIDlet. En primer lugar, se realiza la llamada al constructor del MIDlet pasando éste al estado de "Pausa" durante un corto período de tiempo. El AMS por su parte crea una nueva instancia del MIDlet. Cuándo el dispositivo está preparado para ejecutar el MIDlet, el AMS invoca al método MIDlet.startApp() para entrar en el estado de "Activo". El MIDlet entonces, ocupa todos los recursos que necesita para su ejecución. Durante este estado, el MIDlet puede pasar al estado de "Pausa" por una acción del usuario, o bien, por el AMS que reduciría en todo lo posible el uso de los recursos del dispositivo por parte del MIDlet. Tanto en el estado "Activo" como en el de "Pausa", el MIDlet puede pasar al estado "Destruido" realizando una llamada al método MIDlet.destroyApp().

Esto puede ocurrir porque el MIDlet haya finalizado su ejecución o porque una aplicación prioritaria necesite ser ejecutada en memoria en lugar del MIDlet. Una vez destruido el MIDlet, éste libera todos los recursos ocupados.

1.2.10 ESTRUCTURA DE UN MIDlet

En este punto ya sabemos cuales son los estados de un MIDlet, conocemos su ciclo de vida y hemos estudiado todas sus clases y métodos. Ahora vamos a ver cuál es la estructura que comparten todos los MIDlets y posteriormente estudiaremos el código de nuestro primer MIDlet.

Los MIDlets, al igual que los applets carecen de la función main(). Aunque existiese, el gestor de aplicaciones la ignoraría por completo. Un MIDlet tampoco puede realizar una llamada a System.exit(). Una llamada a este método lanzaría la excepción SecurityException.

```
import javax.microedition.midlet.* public

class MiMidlet extends MIDlet { public

MiMidlet() {

/* Éste es el constructor de clase. Aquí debemos inicializar

nuestras variables. */

}               public

startApp(){

/* Aquí incluiremos el código que queremos que el MIDlet ejecute

cuándo se active. */

}               public

pauseApp(){

/* Aquí incluiremos el código que queremos que el MIDlet
ejecute cuándo entre en el estado de pausa (Opcional). */
```

```
}                    public

destroyApp(){

/* Aquí incluiremos el código que queremos que el MIDlet

ejecute cuándo sea destruido. Normalmente aquí se liberaran

los recursos ocupados por el MIDlet como memoria, etc.

(Opcional) */

}
}
```

Del código anterior, se puede destacar los siguientes elementos, comunes a todo MIDlet:

- Iimportar todas las clases del paquete javax.microedition.midlet (MIDlet y MIDletStateChangeExeption).

- Todos los MIDlet heredan de la clase javax.microedition.midlet.MIDlet. Gracias a esta herencia, todos presenta un mismo interfaz que permite su gestión independientemente del entorno de ejecución.

- Los MIDlets, al igual que los applets carecen de la función main(). Aunque existiese, el gestor de aplicaciones la ignoraría por completo.

- Implementar obligatoriamente las funciones abstractas startApp(), pauseApp() y destroyApp().

1.2.11 APIS JSR 82 BLUETOOTH

Mientras que el hardware Bluetooth había avanzado mucho, hasta hace relativamente poco no había manera de desarrollar aplicaciones java Bluetooth hasta que apareció JSR 82, que estandarizó la forma de desarrollar aplicaciones Bluetooth usando Java. Ésta esconde la complejidad del protocolo Bluetooth detrás de unos APIs que permiten centrarse en el desarrollo en vez de los detalles de bajo nivel del Bluetooth.

Estos APIs para Bluetooth están orientados para dispositivos que cumplan las siguientes características:

- Al menos 512K de memoria libre (ROM y RAM) (las aplicaciones necesitan memoria adicional).

- Conectividad a la red inalámbrica Bluetooth.

- Que tengan una implementación del J2ME CLDC.

El objetivo de ésta especificación era definir un API estándar abierto, no propietario que pudiera ser usado en todos los dispositivos que implementen J2ME. Por consiguiente fue diseñado usando los APIs J2ME y el entorno de trabajo CLDC/MIDP.

Los APIs JSR 82 son muy flexibles, permiten trabajar tanto con aplicaciones nativas Bluetooth como con aplicaciones Java Bluetooth.

- El API intenta ofrecer las siguientes capacidades:

- Registro de servicios.

- Descubrimiento de dispositivos y servicios.

- Establecer conexiones RFCOMM, L2CAP y OBEX entre dispositivos.

- Usar dichas conexiones para mandar y recibir datos (las comunicaciones de voz no están soportadas).
- Manejar y controlar las conexiones de comunicación.

- Ofrecer seguridad a dichas actividades.

Los APIs Java para Bluetooth definen dos paquetes que dependen del paquete CLDC javax.microedition.io:

- **javax.bluetooth:** Este paquete define clases e interfaces básicas para el descubrimiento de dispositivos, descubrimiento de servicios, conexión y comunicación. La comunicación a través de javax.bluetooth es a bajo nivel: mediante flujos de datos o mediante la transmisión de arrays de bytes.

- **javax.obex:** Permite manejar el protocolo de alto nivel OBEX (OBject EXchange). Este protocolo es muy similar a HTTP y es utilizado sobre todo para el intercambio de archivos. El protocolo OBEX es un estándar desarrollado por IrDA y es utilizado también sobre otras tecnologías inalámbricas distintas de Bluetooth.

La anatomía de una aplicación Bluetooth está dividida en cuatro partes:

- **Inicialización de la pila:** La pila Bluetooth es la responsable de controlar el dispositivo Bluetooth, siendo necesario inicializarla antes de hacer cualquier otra cosa. El proceso de inicialización consiste en un número de pasos cuyo propósito es dejar el dispositivo listo para la comunicación inalámbrica.

- **Descubrimiento de dispositivos y servicios:** Dado que los dispositivos inalámbricos son móviles, necesitan un mecanismo que permita encontrar, conectar, y obtener información sobre las características de dichos dispositivos.

- **Manejo del dispositivo:** Los dispositivos inalámbricos, son más vulnerables a ataques del tipo spoofing y eavesdropping que los demás dispositivos. Es por ello, que la tecnología Bluetooth incluye una serie de medidas para evitar estas vulnerabilidades, por ejemplo el salto de frecuencia; más aún, Bluetooth provee además de otros mecanismos opcionales como son la encriptación y autenticación.

- **Comunicación:** Para usar un servicio en un dispositivo Bluetooth remoto, el dispositivo local debe comunicarse usando el mismo protocolo que el servicio remoto. Los APIs permiten usar RFCOMM, L2CAP u OBEX como protocolo de nivel superior. El Generic Connection Framework (GFC) del CLDC provee la conexión base para la implementación de protocolos de comunicación.

1.3 GENERALIDADES DE LAS TARJETAS DE PROXIMIDAD

1.3.1 INTRODUCCIÓN

Las tarjetas de proximidad son dispositivos que basan su funcionamiento en la tecnología de Identificación por Radiofrecuencia RFID (RadioFrequency Identification) es, sin duda, una de las tecnologías de comunicación que ha experimentado un crecimiento más acelerado y sostenido en los últimos tiempos.

Las posibilidades que ofrece la lectura a distancia de la información contenida en una etiqueta, sin necesidad de contacto físico, junto con la capacidad para realizar múltiples lecturas (y en su caso, escrituras) simultáneamente, abre la puerta a un conjunto muy extenso de aplicaciones en una gran variedad de ámbitos, desde la trazabilidad y control de inventario, hasta la localización y seguimiento de personas bienes, o la seguridad en el control de accesos.

Son muchas las grandes compañías que apoyan la implantación y el uso sensato de la RFID, esperarando que su futuro sea muy prometedor. No hay duda de que se trata de una tecnología que puede aportar sustanciales ventajas en muchos ámbitos de aplicación. Sin embargo, el éxito final en la implantación de esta tecnología está sujeto a la superación de una serie de obstáculos, entre los que es necesario destacar los aspectos de seguridad y privacidad.

RFID representa una tecnología con un carácter emergente (aunque cada vez menos) en cuanto a las posibilidades de su aplicación industrial. Son muchos los sectores de la economía que se pueden beneficiar de las principales ventajas que ofrece esta tecnología.

1.3.2 HISTORIA RFID

El origen de la RFID está tristemente relacionado con la guerra, concretamente con la II Guerra Mundial, el radar permitíera la detección de aviones a kilómetros de distancia,

pero no su identificación. El ejército alemán descubrió que si los pilotos balanceaban sus aviones al volver a la base cambiaría la señal de radio reflejada de vuelta. Este método hacía así distinguir a los aviones alemanes de los aliados y se convirtió en el primer dispositivo de RFID pasiva.

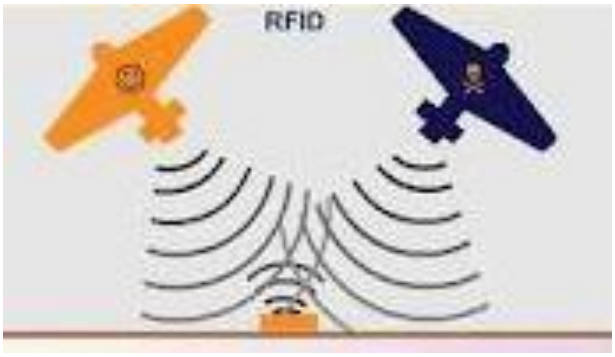


Figura 1.13 Principio del RFID

Los sistemas de radar y de comunicaciones por radiofrecuencia avanzaron en las décadas de los 50 y 60 en que los científicos de países avanzados trabajaban para explicar cómo identificar objetos remotamente. Las compañías pronto comenzaron a trabajar con sistemas antirrobo que usando ondas de radio determinaban si un objeto había sido pagado o no a la salida de las tiendas, utilizando una etiqueta en la que un único bit decide si se ha pagado o no por el objeto en cuestión. La etiqueta pitará en los sensores colocados a la salida si el objeto no se ha pagado.

Las primeras patentes para dispositivos RFID fueron solicitadas en Estados Unidos, concretamente en Enero de 1973 cuando Mario W. Cardullo se presentó con una etiqueta RFID activa que portaba una memoria rescribible. El mismo año, Charles Walton recibió la patente para un sistema RFID pasivo que abría las puertas sin necesidad de llaves.

Una tarjeta con un transponedor comunicaba una señal al lector de la puerta que cuando validaba la tarjeta desbloqueaba la cerradura.

El gobierno americano también trabajaba sobre esta tecnología en los años 70 y montó sistemas parecidos para el manejo de puertas en las centrales nucleares, cuyas puertas se abrían al paso de los camiones que portaban materiales para las mismas que iban equipados con un transponedor. También se desarrolló un sistema para el control del ganado que había sido vacunado insertando bajo la piel de los animales una etiqueta RFID pasiva que identificaba los animales que habían sido vacunados y los que no.

En transcurso de los años se producen mejoras en la capacidad de emisión y recepción, así como en la distancia, aumentando su uso en ámbitos tanto domésticos como de seguridad nacional, como sucede con el pasaporte expedido en la actualidad en los EEUU que lleva asociadas etiquetas RFID.

1.3.3 TECNOLOGÍA RFID

Los sistemas de identificación por radiofrecuencia o RFID (Radio Frequency Identification) son una nueva tecnología para la identificación de objetos a distancia sin necesidad de contacto, ni siquiera visual. RFID (Identificación por Radiofrecuencia) es un método de almacenamiento y recuperación remota de datos, basado en el empleo de etiquetas o "tags" en las que reside la información. RFID se basa en un concepto similar al del sistema de código de barras; la principal diferencia entre ambos reside en que el segundo utiliza señales ópticas para transmitir los datos entre la etiqueta y el lector, y RFID, en cambio, emplea señales de radiofrecuencia (en diferentes bandas dependiendo del tipo de sistema, típicamente 125 KHz, 13,56 MHz, 433-860-960 MHz y 2,45 GHz).

Todo sistema RFID se compone principalmente de cuatro elementos:

Etiqueta RFID: también llamada tag o transpondedor (transmisor y receptor). La etiqueta se inserta o adhiere en un objeto, animal o persona, portando información sobre el mismo. En este contexto, la palabra "objeto" se utiliza en su más amplio sentido: puede ser un vehículo, una tarjeta, una llave, un paquete, un producto, una planta, etc.

Consta de un microchip que almacena los datos y una pequeña antena que habilita la comunicación por radiofrecuencia con el lector.

Lector o interrogador: encargado de transmitir la energía suficiente a la etiqueta y de leer los datos que ésta le envíe. Consta de un módulo de radiofrecuencia (transmisor y receptor), una unidad de control y una antena para interrogar los tags vía radiofrecuencia. Los lectores están equipados con interfaces estándar de comunicación que permiten enviar los datos recibidos de la etiqueta a un subsistema de procesamiento de datos, como puede ser un ordenador personal o una base de datos.

Algunos lectores llevan integrado un programador que añade a su capacidad de lectura, la habilidad para escribir información en las etiquetas.

A lo largo del presente estudio, cuando hablemos de lector, se considerará que es un dispositivo capaz de leer la etiqueta, independientemente de si puede sólo leer, o leer y escribir.

Un ordenador, host o controlador: que desarrolla la aplicación RFID. Recibe la información de uno o varios lectores y se la comunica al sistema de información.

Adicionalmente, un middleware y en backend un sistema ERP de gestión de sistemas IT son necesarios para recoger, filtrar y manejar los datos.

Todos estos elementos conforman un sistema RFID que, atendiendo a distintos criterios relacionados con las características técnicas y operacionales de cada uno de los componentes, puede ser de diversos tipos. A continuación se muestra una clasificación de los distintos sistemas RFID existentes:

a) Según su capacidad de programación:

- De sólo lectura: las etiquetas se programan durante su fabricación y no pueden ser reprogramadas.
- De una escritura y múltiples lecturas: las etiquetas permiten una única reprogramación.
- De lectura/escritura: las etiquetas permiten múltiples reprogramaciones.

b) Según el modo de alimentación:

- Activos: si las etiquetas requieren de una batería para transmitir la información.
- Pasivos: si las etiquetas no necesitan batería.

c) Según el rango de frecuencia de trabajo:

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

- Alta Frecuencia (AF): cuando la frecuencia de funcionamiento es de 13,56 MHz.
- Ultra Alta Frecuencia (UHF): comprende las frecuencias de funcionamiento en las bandas de 433 MHz, 860 MHz, 928 MHz.
- Frecuencia de Microondas: comprende las frecuencias de funcionamiento en las bandas de 2,45 GHz y 5,8 GHz.

d) Según el protocolo de comunicación:

- Dúplex: el transpondedor transmite su información en cuanto recibe la señal del lector y mientras dura ésta.
- Half dúplex , cuando transpondedor y lector transmiten en turnos alternativos.
- Full dúplex , cuando la comunicación es simultánea. Es estos casos la transmisión del transponde dor se realiza a una frecuencia distinta que la del lector.
- Secuencial: el campo del lector se apaga a intervalos regulares, momento que aprovecha el transpondedor para enviar su información. Se utiliza con etiquetas activas, ya que el tag no puede aprovechar toda la potencia enviada por el lector y requiere una batería adicional para transmitir, lo cual incrementaría el costo.

e) Según el principio de propagación:

- Inductivos: utilizan el campo magnético creado por la antena del lector para alimentar el tag. Opera en el campo cercano y a frecuencias bajas (BF y AF).

- Propagación de ondas electromagnéticas: utilizan la propagación de la onda electromagnética para alimentar la etiqueta. Opera en el campo lejano y a muy altas frecuencias (UHF y microondas).

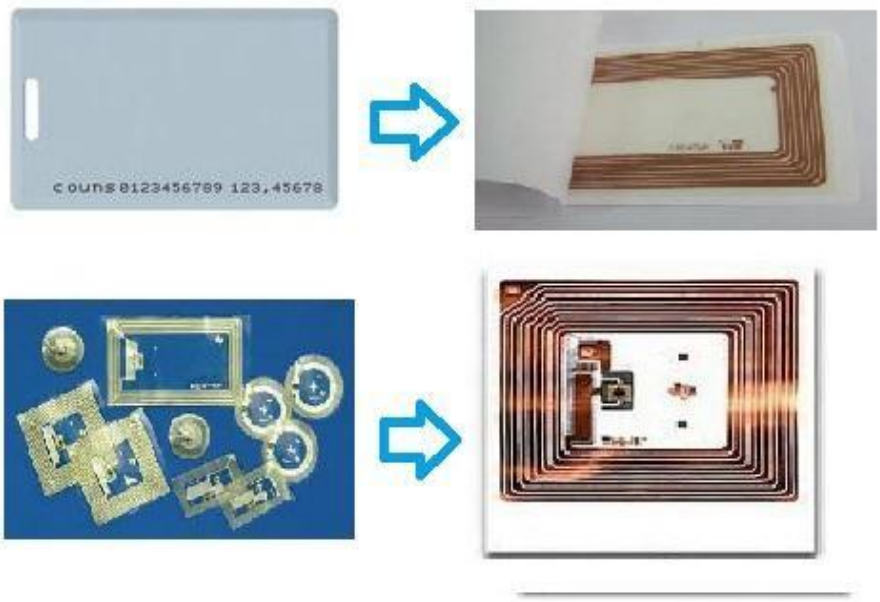


Figura 1.14 Tags pasivos

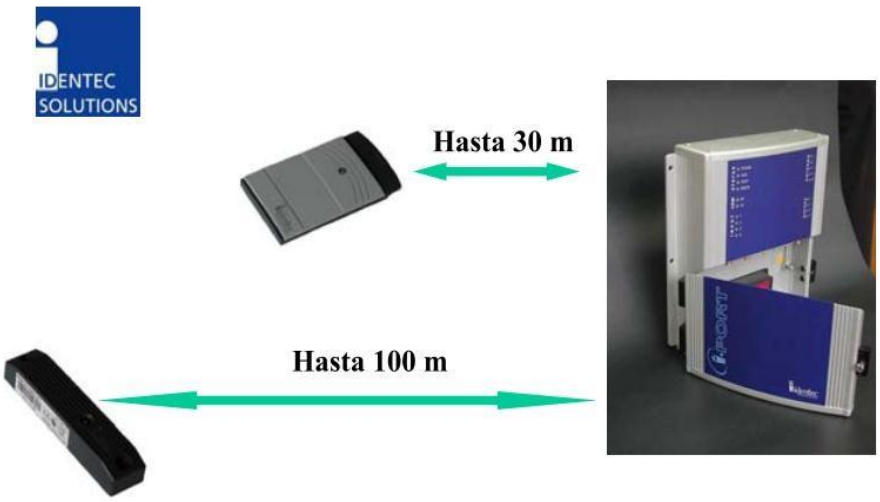


Figura 1.15 Tags Activos

1.3.4 FUNCIONAMIENTO Y COMPONENTES

Existe una gran diversidad de sistemas RFID, con la capacidad de satisfacer un amplio abanico de aplicaciones. Sin embargo, a pesar de que los aspectos tecnológicos pueden

variar, todos se basan en el mismo principio de funcionamiento, que se describe a continuación:

1. Se equipa a todos los objetos a identificar, controlar o seguir, con una etiqueta RFID.

2. La antena del lector o interrogador que emite un campo de radiofrecuencia que activa las etiquetas.

3. Cuando una etiqueta ingresa en dicho campo utiliza la energía y la referencia temporal recibidas para realizar la transmisión de los datos almacenados en su memoria. En el caso de etiquetas activas la energía necesaria para la transmisión proviene de la batería de la propia etiqueta.

4. El lector recibe los datos y los envía al ordenador de control para su procesamiento.

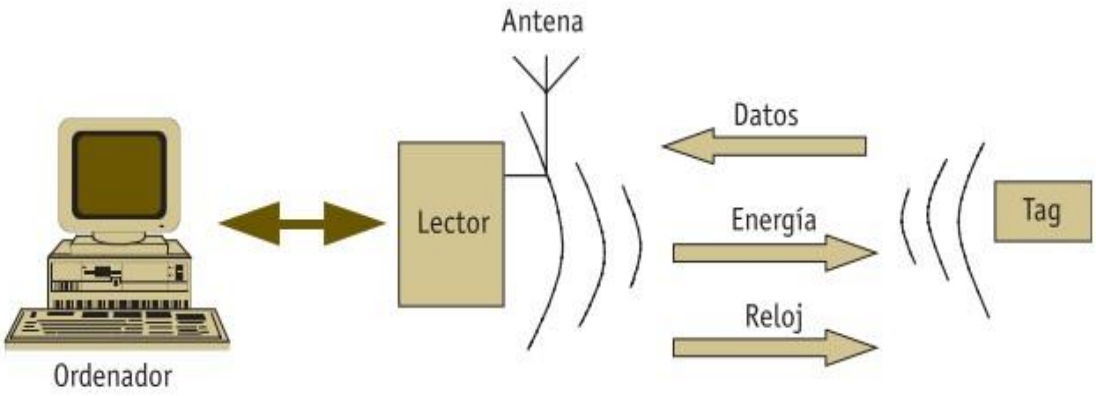


Figura 1.16 Esquema de funcionamiento RFID

Todo sistema RFID se compone básicamente de cuatro elementos: transpondedor o etiqueta, lector o interrogador, sistema de información, adicionalmente, middleware. A continuación se describe cada uno de estos componentes y sus principales parámetros que los caracterizan.

1.3.4.1 Transpondedores:

El transpondedor es el dispositivo embebido en una etiqueta o tag y contiene la información asociada al objeto al que acompaña, transmitiéndola cuando el lector la solicita.

Está compuesto principalmente por un microchip y una antena. Adicionalmente puede incorporar una batería para alimentar sus transmisiones o incluso algunas etiquetas más sofisticadas pueden incluir una circuitería extra con funciones adicionales de entrada/salida, tales como registros de tiempo u otros estados físicos que pueden ser monitorizados mediante sensores apropiados (de temperatura, hu medad, etc.).

Las etiquetas activas, además de recoger energía del lector, se alimentan de una batería. Normalmente incorporan una pila que posee una alta relación potencia-peso y son capaces de funcionar en un intervalo de temperaturas desde –50ºC hasta 70ºC.

Aunque el empleo de baterías implica un tiempo de vida finito para el dispositivo, la colocación de una pila acoplada de forma apropiada a la circuitería de baja potencia, puede asegurar un tiempo de vida de algo más de 10 años, dependiendo también de las condiciones de trabajo en las que se encuentre, es decir, las temperaturas, ciclos de lectura/escritura y su utilización.

Típicamente son dispositivos de lectura/escritura. Además, una ventaja adicional que presentan frente a las etiquetas pasivas es que pueden usarse parar gestionar otros dispositivos, como pueden ser los sensores.

En términos generales las etiquetas RFID activas permiten un radio de cobertura mayor, mejor inmunidad al ruido y tasas de transmisión más altas cuando trabaja a altas frecuencia. Estas ventajas se traducen en un coste mayor, siendo aplicadas cuando los bienes a identificar lo justifican.

Existen dos tipos de etiquetas activas:

- Aquellas que normalmente se encuentran desactivadas (modo reposo) y se activan (despiertan) cuando un lector las interroga. De esta forma se ahorra batería.

- Aquellas que periódicamente envían señales, aunque un lector no las interrogue. Operan a frecuencias más bajas y a menores tasas de transferencias, para ahorrar batería.

Las etiquetas pasivas funcionan sin una batería interna, obteniendo la potencia que necesitan para funcionar del campo generado por el interrogador.

La ausencia de batería provoca que los transpondedores pasivos sean mucho más ligeros, pequeños, flexibles y baratos que los activos, hecho que redunda en que puedan ser diseñados en una amplia gama de formas. Además, ofrecen un tiempo de vida prácticamente ilimitado. Como contrapartida, poseen unos radios de cobertura menores y requieren más cantidad de energía procedente del interrogador para poder transmitir los datos. También poseen restricciones a la hora de almacenar los datos y no funcionan demasiado bien en ambientes con interferencias electromagnéticas.

Asimismo, su sensibilidad y orientación están limitadas por la potencia disponible. Sin embargo, a pesar de estas limitaciones, las etiquetas pasivas ofrecen mejores ventajas en términos de costo y longevidad.

1.3.4.2 Lectores

Un lector o interrogador es el dispositivo que proporciona energía a las etiquetas, lee los datos que le llegan de vuelta y los envía al sistema de información. Asimismo, también gestiona la secuencia de comunicaciones con el lector.

Con el fin de cumplir tales funciones, está equipado con un módulo de radiofrecuencia (transmisor y receptor), una unidad de control y una antena. Además, el lector incorpora un interfaz a un PC, host o controlador, a través de un enlace local o remoto: RS232, RS485, Ethernet, WLAN (RF, WiFi, Bluetooth, etc.), que permite enviar los datos del transpondedor al sistema de información.

El lector puede actuar de tres modos:

- Interrogando su zona de cobertura continuamente, esperando la presencia de múltiples etiquetas pasando de forma continua.

- Interrogando periódicamente, para detectar nuevas presencias de etiquetas.

- Interrogando de forma puntual, por ejemplo cuando un sensor detecte la presencia de una nueva etiqueta.

- Los componentes del lector son, el módulo de radiofrecuencia (formado por receptor y transmisor), la unidad de control y la antena. A continuación se procede a describir un poco más cada uno de estos elementos.

- El módulo de radiofrecuencia, que consta básicamente de un transmisor que genera la señal de radiofrecuencia y un receptor que recibe, también vía radiofrecuencia, los datos enviados por las etiquetas. Sus funciones por tanto son:

    - Generar la señal de radiofrecuencia para activar el transpondedor y proporcionarle energía.

    - Modular la transmisión de la señal para enviar los datos al transpondedor.

    - Recibir y demo dular las señales enviadas por el transpondedor.

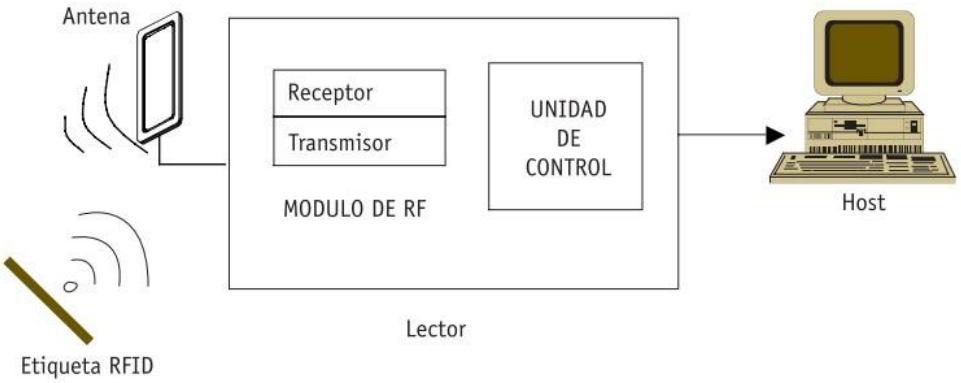


Figura 1.17 Diagrama de un lector RFID

Cuando una etiqueta es detectada y seleccionada, el lector puede realizar operaciones sobre ella, es decir, leer su información o escribir en ella. Después de

finalizar la operación, el lector descarta la etiqueta para proceder a interrogar a la siguiente. Existen algoritmos como el "Protocolo Orden-Respuesta", en el cual el lector ordena a un transpondedor que cese su transmisión, cuando reconoce que ya ha recibido la información. Otro método alternativo, más seguro pero más lento y costoso, se denomina "Sondeo Selectivo", donde el lector busca específicamente las etiquetas que tienen una determinada identificación y las interroga por turnos. Por último, otra aproximación, aunque más cara, incluye el empleo de varios lectores multiplexados en único interrogador.

Los lectores pueden variar su complejidad considerablemente dependiendo del tipo de transpondedor que tengan que alimentar y las funciones que deban desarrollar. Una posible clasificación seria fijos o móviles dependiendo de la aplicación que se considere.

- **Los dispositivos fijos** se posicionan en lugares estratégicos como puertas de acceso, lugares de paso o puntos críticos dentro de una cadena de ensamblaje, de modo que puedan monitorizar las etiquetas de la aplicación en cuestión.

Figura 1.18 Lector fijo RFID

- **Los lectores móviles** suelen ser dispositivos de mano. Incorporan una pantalla LCD, un teclado para introducir datos y una antena integrada

dentro de una unidad portátil. Por esta razón, el radio de cobertura suele ser menor.

Figura 1.19 Lictor Portable RIFD

Los principales parámetros que caracterizan un lector RFID son:

Frecuencia de operación. El lector puede funcionar a baja frecuencia, alta frecuencia, ultra alta frecuencia y frecuencia de microondas. Ya existen en el mercado lectores multifrecuencia.

- Protocolo de funcionamiento. Muchas compañías ofrecen soporte multiprotocolo (ISO, propietarios…), pero no admiten todos los protocolos existentes.

- Tipo de regulación que siguen . Por ejemplo, existen distintas regulaciones de frecuencia y de potencia en Estados Unidos y en Europa:

    - La banda de UHF funciona a 902–930 MHz en Estados Unidos y a 869 MHz en Europa.
    - La máxima potencia permitida es de 2 Watios en Estados Unidos y 0,5 Watios en Europa.

- Interfaz con el sistema host:

- TCP/IP. ○ WLAN. ○ Ethernet (10BaseT).
- Serie: RS 232, RS 485.

- Capacidad para multiplexar muchos lectores:
  - A través de concentradores.
  - A través de middleware.
- Capacidad para actualizar el software del lector on-line:
  - Vía Internet.
  - Vía interfaz con el host.

- Capacidad para gestionar múltiples antenas, típicamente 4 antenas/lector.

- Capacidad para interactuar con otros productos de middleware.

- Entrada/salida digital para conectar otros dispositivos tales como sensores externos o circuitos de control adicionales

1.3.4.3 SISTEMA DE INFORMACIÓN

Los códigos de barras estándar, las etiquetas RFID son simplemente un modo automatizado para proporcionar datos de entrada al sistema cliente. Sin embargo, las etiquetas RFID son capaces de proporcionar también una salida automatizada del sistema hacia la etiqueta, permitiendo la actualización dinámica de los datos que ésta porta.

El sistema de información se comunica con el lector según el principio maestro-esclavo. Esto quiere decir que todas las actividades realizadas por lector y transpondedores son iniciadas por la aplicación software. Cuando el lector recibe una orden de esta aplicación, establece una comunicación con los transpondedores, comunicación ejercida por el lector maestro y los tags de esclavos.

El principal objetivo de la aplicación software es gestionar y tratar los datos recibidos por el lector. El sistema debe ser suficientemente robusto para poder manejar las múltiples lecturas que permiten realizar los sistemas RFID, coordinar tiempos y flujos de información, gestionar distintos eventos, soportar las realimentaciones de usuarios, introducir las actualizaciones del sistema cuando sea requerido e integrarlo con otros sistemas de información de la empresa. En todos los casos el sistema cliente necesitará modificaciones software para integrar datos proporcionados por el lector
y el programador. Sin la posibilidad de acceder a todas estas funcionalidades, el sistema RFID perderá en eficacia y no proporcionará el deseado retorno de la inversión.

1.3.4.4 MIDDLEWARE

El middleware es el software que se ocupa de la conexión entre el hardware de RFID y sistemas de información existentes (y posiblemente anteriores a la implantación de RFID) en la aplicación. Del mismo modo que un PC, los sistemas RFID hardware serían inútiles sin un software que permita su funcionamiento.

Entre otras cosas es encargado del encaminamiento de los datos entre los lectores, las etiquetas y los sistemas de información, y es responsable de la calidad y usabilidad de las aplicaciones basadas en RFID.

El middleware de RFID se ocupa de transmitir datos entre los extremos de la transacción. Por ejemplo, en un sistema RFID basado en etiquetas, en el proceso de lectura se ocuparía de la transmisión de datos almacenados en una de las etiquetas al sistema de información. Las cuatro funciones principales del middleware de RFID son:

- Adquisición de datos. El middleware es responsable de la extracción, agrupación y filtrado de datos procedentes de múltiples lectores RFID en un sistema complejo. Sin la existencia del middleware, los sistemas de información de las empresas se colapsarían con rapidez.

- Encaminamiento de datos . El middleware facilita la integración de las redes de elementos y sistemas RFID de la aplicación. Para ello dirige los datos al sistema apropiado dentro de la aplicación.
- Gestión de procesos. El middleware se puede utilizar para disparar eventos en función de las reglas de la organización empresarial donde opera, por ejemplo, envíos no autorizados, bajadas o pérdidas de stock, etc.

- Gestión de dispositivos. El middleware se ocupa también de monitorizar y coordinar los lectores RFID, así como de verificar su estado y operatividad, y posibilita su gestión remota.

Muchos de los middleware desarrollados o en desarrollo se ajustan a los estándares de EPCglobal, conocidos como Savant. La especificación Savant ordena los componentes del middleware de acuerdo a sus funciones.

1.4 LOS MICROCONTOLADORES PIC: EL PIC16F877A

1.4.1 EL MICROCONTOLADOR PIC

Un microcontrolador, es un circuito integrado programable que contiene elementos necesarios para controlar un sistema. La familia de microcontroladores PIC fue introducida en el mercado en el año 1985 por la empresa Microchip Technology.

Esta familia incluye un gran número de microcontroladores, resultando posible encontrar un PIC adecuado para casi cualquier tipo de aplicación. PIC significa Peripheral Interface Controler es decir un controlador de periféricos.

Cuando hablamos de un circuito integrado programable que controla periféricos, estamos hablando de un sistema que contiene entre otras cosas una unidad ariméticológica, unas memorias de datos y programas, puertos de entrada y salida, es decir estamos hablando de un dispositivo diseñado para realizar funciones específicas.

Podemos encontrar microcontroladores en lavadoras, teclados, teléfonos móviles, ratones etc. Hay multitud de microcontroladores con más memoria, entradas y salidas.

1.4.1.1 CARACTERÍSTICAS DE UN MICROCONTROLADOR

Las principales características de los µC son:

- Unidad de Procesamiento Central (CPU): Típicamente de 8 bits, pero también las hay de 4, 32 y hasta 64 bits con arquitectura Harvard, con memoria y bus de datos separada de la memoria y bus de instrucciones de programa, o arquitectura de von Neumann, también llamada arquitectura Princeton, con memoria y bus de datos y memoria y bus de programa compartidas.

- Memoria de Programa: Es una memoria ROM (Read-Only Memory), EPROM (Electrically Programable ROM), EEPROM(Electrically Erasable/Programable ROM) o Flash que almacena el código del programa que típicamente puede ser de 1 kilobyte a varios megabytes.

- Memoria de Datos: Es una memoria RAM (Random Access Memory) que típicamente puede ser de 1, 2 4, 8, 16, 32 kilobytes.

- Generador del Reloj: Usualmente un cristal de cuarzo de frecuencias que genera una señal oscilatoria hasta 40 MHz, o también resonadores o circuitos RC.

- Interfaz de Entrada/Salida: Puertos paralelos, seriales (UARTs, Universal Asynchronous Receiver/Transmitter), I2C (Inter-Integrated Circuit), Interfaces de Periféricos Seriales (SPIs, Serial Peripheral Interfaces), Red de Area de Controladores (CAN, Controller Area Network), USB (Universal Serial Bus).

- Otras opciones:

    - Conversores Análogo-Digitales (A/D, analog-to-digital) para convertir un nivel de voltaje en un cierto pin a un valor digital manipulable por el programa del microcontrolador.

- Moduladores por Ancho de Pulso (PWM, Pulse-Width Modulation) para generar ondas de frecuencia fija pero con ancho de pulso modificable.

1.4.2 EL MICROCONTROLADOR PIC16F877A

El PIC16F877 es un microcontrolador con memoria de programa tipo FLASH, representando gran facilidad en el desarrollo de prototipos y en su aprendizaje sin requerir a borrarlo con luz ultravioleta como las versiones EPROM, permitiendo reprogramarlo nuevamente sin ser borrado con anterioridad.

El PIC16F877A es un microcontrolador de Microchip Technology fabricado en tecnología CMOS, su consumo de potencia es muy bajo y además es completamente estático, esto quiere decir que el reloj puede detenerse y los datos de la memoria no se pierden.

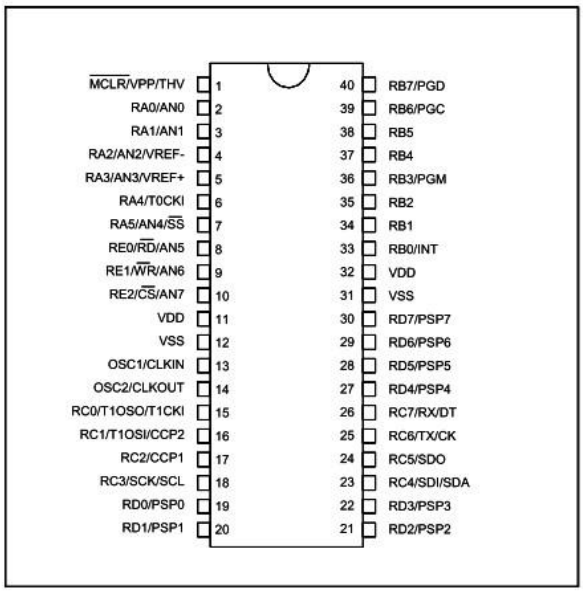


Figura 1.20 PIC16F877A

1.4.3 PRINCIPALES CARACTERÍSTICAS DEL PIC16F877A

A continuación se enumeran las prestaciones y dispositivos especiales del PIC16F877A Procesador de arquitectura RISC avanzada

- Juego de solo 35 instrucciones con 14 bits de longitud. Todas ellas se ejecutan en un ciclo de instrucción, menos las de salto que tardan dos.
- Hasta 8K palabras de 14 bits para la Memoria de Programa, tipo FLASH en el 16F877
- Hasta 368 Bytes de memoria de Datos RAM.
- Hasta 256 Bytes de memoria de Datos EEPROM.
- Pines de salida compatibles para el PIC 16C73/74/76/77.
- Hasta 14 fuentes de interrupción internas y externas.
- Pila de 8 niveles.
- Modos de direccionamiento directo e indirecto.
- Power-on Reset (POP).
- Temporizador Power-on (POP) y Oscilador Temporizador Start-Up.
- Perro Guardián (WDT).
- Código de protección programable.
- Modo SLEEP de bajo consumo.
- Programación serie en circuito con dos pines, solo necesita 5V para programarlo en este modo.
- Voltaje de alimentación comprendido entre 2 y 5,5 V.
- Bajo consumo.

1.4.4 DISTRIBUCIÓN DE LOS PINES DEL PIC16F877A

Los pines de entrada/salida de este microcontrolador están organizados en cinco puertos, el puerto A con 6 líneas, el puerto B con 8 líneas, el puerto C con 8 líneas, el puerto D con 8 líneas y el puerto E con 3 líneas. Cada pin de esos puertos se puede configurar como entrada o como salida independiente programando un par de registros diseñados para tal fin. En ese registro un bit en "0" configura el pin del puerto correspondiente como salida

y un bit en "1" lo configura como entrada. Dichos pines del microcontrolador también pueden cumplir otras funciones especiales, siempre y cuando se configuren para ello, según se verá más adelante.

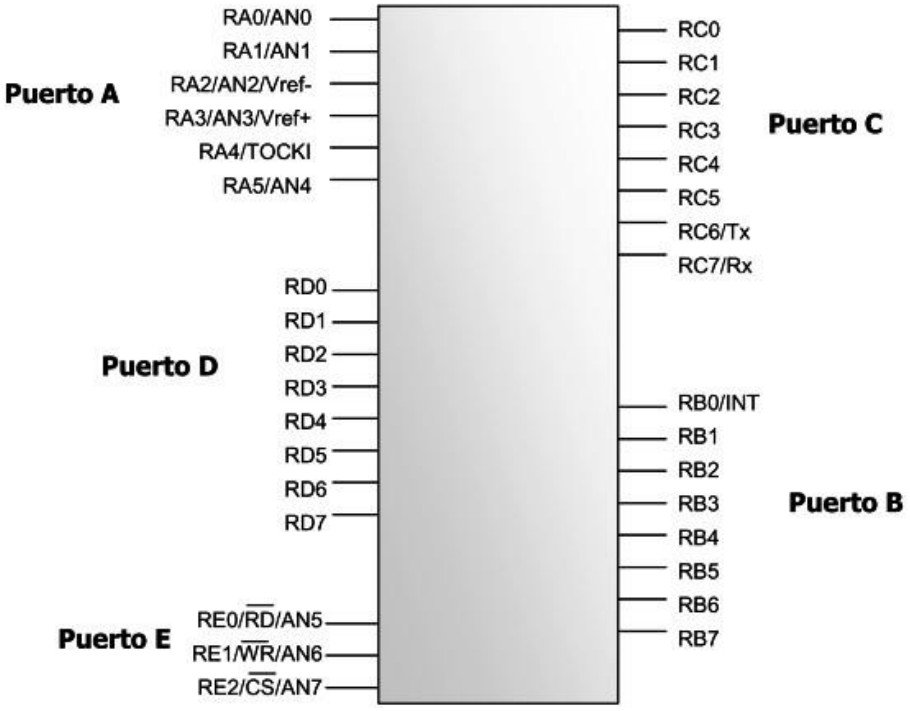


Figura 1.21 Distribución de Pines del PIC16F877A

- **Los pines del puerto A y del puerto E** pueden trabajar como entradas para el convertidor Análogo a Digital interno, pudiéndose conectar una señal analogica para que el microcontrolador la convierta en su equivalente digital y pueda realizar algún proceso de control o de instrumentación digital. El pin RB0/INT se puede configurar por software para que funcione como interrupción externa, para configurarlo se utilizan bits de los registros que controlan las interrupciones.

  El pin RA4/TOCKI del puerto A puede ser configurado como un pin de entrada/salida o como entrada del temporizador/contador. Cuando este pin se programa como entrada digital, funciona como un disparador de Schmitt (Schmitt trigger), puede reconocer señales un poco distorsionadas y llevarlas a niveles lógicos (cero y cinco voltios).

Cuando se usa como salida digital se comporta como colector abierto (open collector), por lo tanto, se debe poner una resistencia de pull-up (resistencia externa conectada a un nivel de cinco voltios). Como salida, la lógica es inversa: un "0" escrito al pin del puerto entrega en el pin un "1" lógico.

Además, como salida no puede manejar cargas como fuente, sólo en el modo sumidero.

- **El puerto E** puede controlar la conexión en modo microprocesador con otros dispositivos utilizando las líneas RD (read), WR (write) y CS (chip select). En este modo el puerto D funciona como un bus de datos de 8 bits (pines PSP).

1.4.5 ARQUITECTURA INTERNA DEL MICROCONTROLADOR

En la siguiente figura 1.22 se muestran los bloques funcionales internos que conforman el microcontrolador y la forma en que están conectados, por ejemplo la memoria FLASH (de programa), la memoria RAM (de datos), los puertos, la lógica de control que permite que todo el conjunto funcione, etc.

La figura 1.22 muestra la arquitectura general del PIC16F877, en ella se pueden observar los diferentes bloques que lo componen y la forma en que se conectan. Se muestra la conexión de los puertos, las memorias de datos y de programa, los bloques especiales como el watchdog, los temporizadores de arranque, el oscilador, etc.

Todos los elementos se conectan entre sí por medio de buses. Un bus es un conjunto de líneas que transportan información entre dos o más módulos. Vale la pena destacar que el PIC16F877 tiene un bloque especial de memoria de datos de 256 bytes del tipo EEPROM, además de los dos bloques de memoria principales que son el de programa y el de datos o registros.

Dispositivo	Memoria de programa	Memoria de datos	Memoria EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes

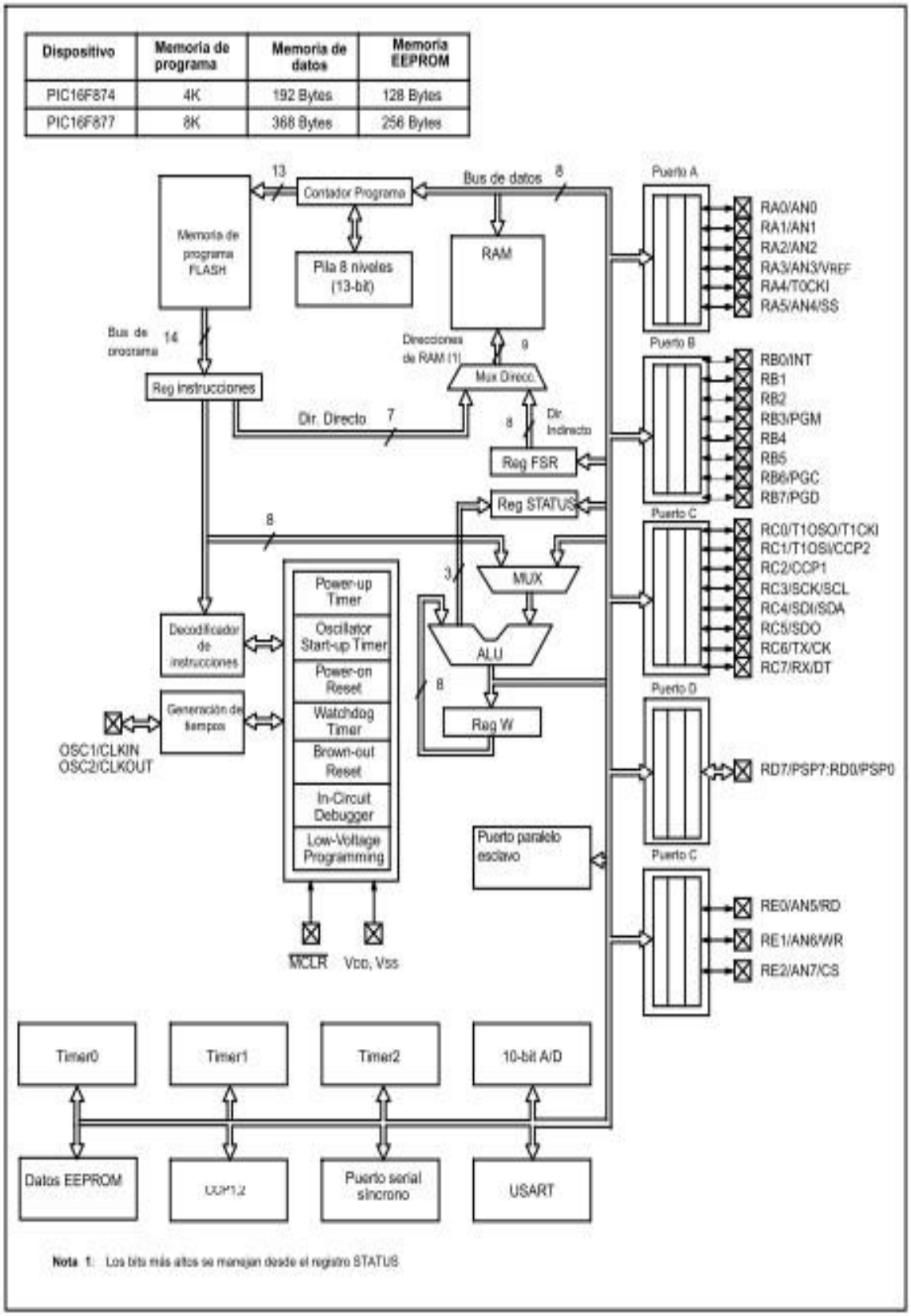


Figura 1. 22 Arquitectura del PIC16F877A

El PIC16F877A se basa en la arquitectura Harvard, posibilitando que las instrucciones y los datos posean longitudes diferentes. Esta misma estructura permite la superposición de

los ciclos de búsqueda y ejecución de las instrucciones, reflejando en una mayor velocidad del microcontrolador.

Memoria de programa (FLASH): Es una memoria de 8K de longitud con datos de 14 bits en cada posición. Como es del tipo FLASH se puede programar y borrar eléctricamente, lo que facilita el desarrollo de los programas y la experimentación. En ella se graba o almacena el programa o códigos que el microcontrolador debe ejecutar. En la figura 3.8 se muestra el mapa de la memoria de programa.

La memoria de programa está dividida en cuatro bancos o páginas de 2K cada uno. El primero va de la posición de memoria 0000h a la 07FFh, el segundo va de la 0800h a la 0FFFh, el tercero de la 1000h a la 17FFh y el cuarto de la 1800h a la 1FFFh.

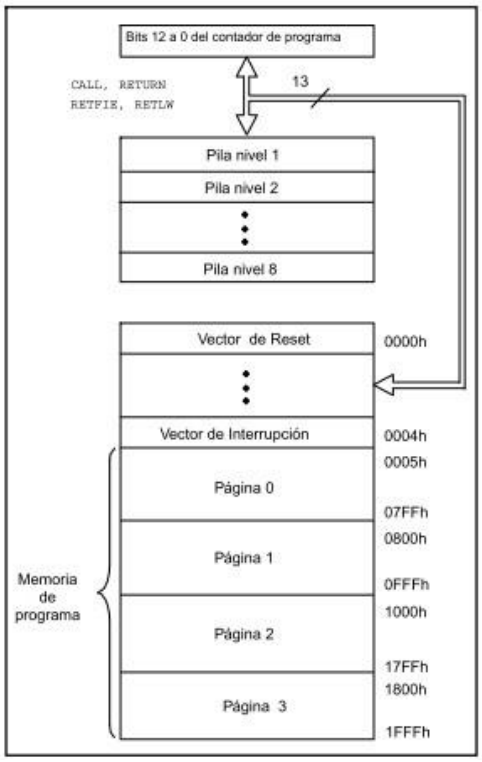
Vector de reset. Cuando ocurre un reset al microcontrolador, el contador de programa se pone en ceros (0000H). Por esta razón, en la primera dirección del programa se escribe todo lo relacionado con la iniciación del mismo.

Vector de interrupción. Cuando el microcontrolador recibe una señal de interrupción, el contador de programa apunta a la dirección 04H de la memoria de programa, por eso, allí se debe escribir toda la programación necesaria para atender dicha interrupción.

Pila (Stack): Estos registros no forman parte de ningún banco de memoria y no permiten el acceso por parte del usuario.

Se usan para guardar el valor del contador de programa cuando se hace un llamado a una subrutina o cuando se atiende una interrupción; luego, cuando el micro regresa a seguir ejecutando su tarea normal, el contador de programa recupera su valor leyéndolo nuevamente desde la pila.

El PIC16F877A tiene una pila de 8 niveles, esto significa que se pueden anidar 8 llamados a subrutina sin tener problemas.



Fígura 1. 23 Memoria de programa del PIC16F877.

Memoria de datos (RAM): El PIC16F877 posee cuatro bancos de memoria RAM, cada banco posee 128 bytes. De estos 128 los primeros 32 (hasta el 1Fh) son egistros que cumplen un propósito especial en el control del microcontrolador y en su configuración. Los 96 siguientes son registros de uso general que se pueden usar para guardar los datos temporales de la tarea que se está ejecutando.

Todas las posiciones o registros de memoria se pueden acceder directa o indirectamente (esta última forma a través del registro selector FSR). Para seleccionar que página o banco de memoria se trabaja en un momento determinado se utilizan los bits RP0 y RP1 del registro STATUS.

Resumen de algunos de los registros de configuración

BANCO 0:

- TMR0: Registro del temporizador/contador de 8 bits.
- PCL: Byte menos significativo del contador de programa (PC).

- STATUS: Contiene banderas (bits) que indican el estado del procesador después de una operación aritmética/lógica.

- FSR: Registro de direccionamiento indirecto.

- PORTA, PORTB, PORTC, PORTD, PORTE: Registro de puertos de E/S de datos. Conectan con los pines físicos del micro.

- PCLATH: Byte alto (más significativo) del contador de programa (PC).

- INTCON: Registro de control de las interrupciones.

- ADRESH: Parte alta del resultado de la conversión A/D.

- ADCON0: Controla la operación del módulo de conversión A/D

BANCO 1:

- OPTION: Registro de control de frecuencia del TMR0.

- TRISA, TRISB, TRISC, TRISD. TRISE: Registros de configuración de la operación de los pines de los puertos.

- ADRESL: Parte baja del resultado de la conversión A/D.

- ADCON1: Controla la configuración de los pines de entrada análoga.

BANCO 2:

- TMR0: Registro del temporizador/contador de 8 bits.

- PCL: Byte menos significativo del contador de programa (PC).

- FSR: Registro de direccionamiento indirecto.

- EEDATA: Registro de datos de la memoria EEPROM.

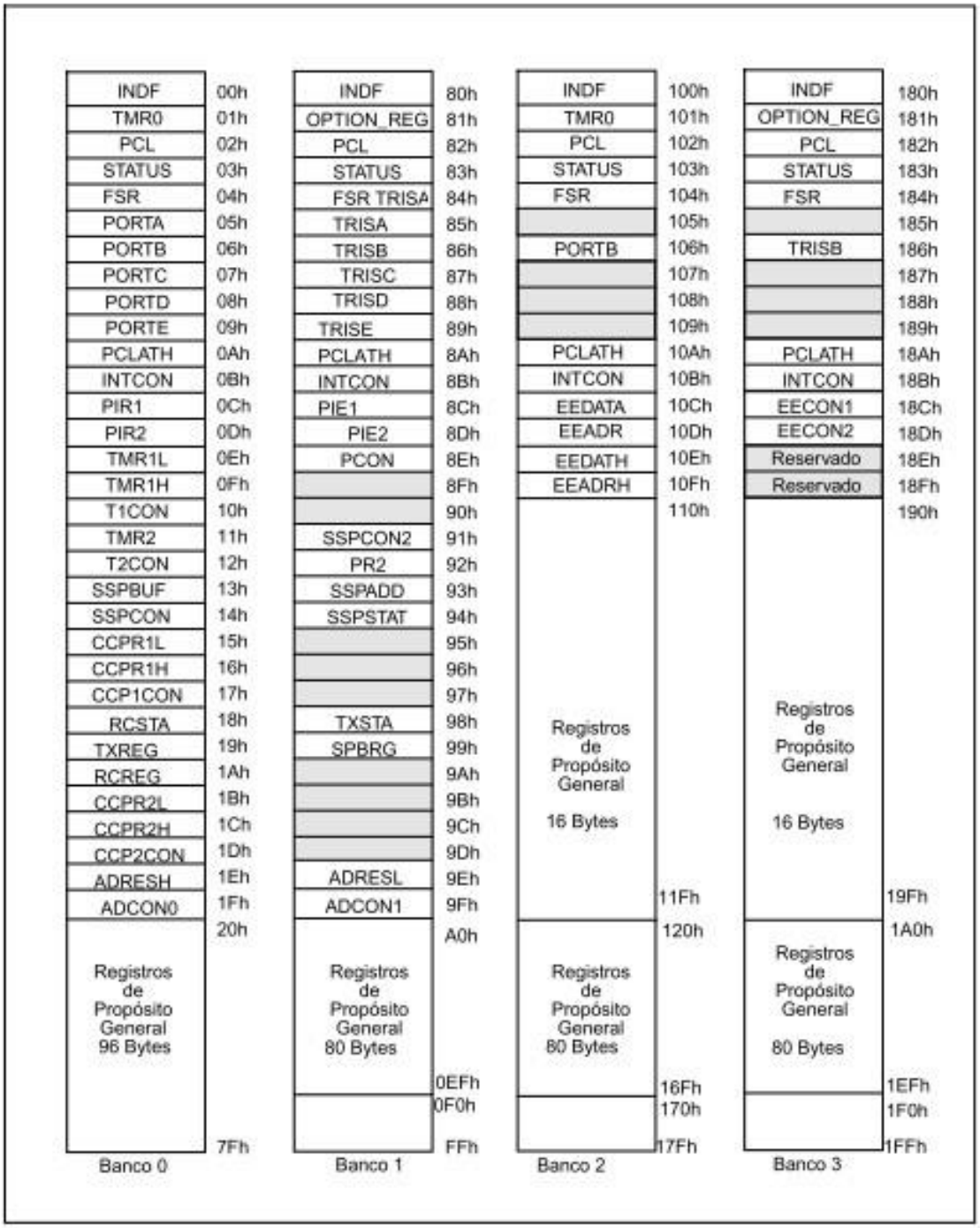
- EEADR: Registro de dirección de la memoria EEPROM.

- PCLATH: Byte alto (más significativo) del contador de programa (PC).

- INTCON: Registro de control de las interrupciones.

BANCO 3:

- OPTION: Registro de control de frecuencia del TMR0.
- EECON1: Control de lectura/escritura de la memoria EEPROM de datos.
- EECON2: No es un registro físico.



Fígura 1. 24 Organización de la memoria RAM del PIC16F877A.

Módulo del convertidor Análogo a Digital: Este módulo permite la conversión de una señal de entrada análoga a su correspondiente valor numérico de 10 bits. El módulo tiene

ocho entradas análogas, las cuales son multiplexadas dentro de un circuito de muestreo y retención. La salida del multiplexor es la entrada al convertidor, el cual genera el resultado por medio de aproximaciones sucesivas, figura 1.25.

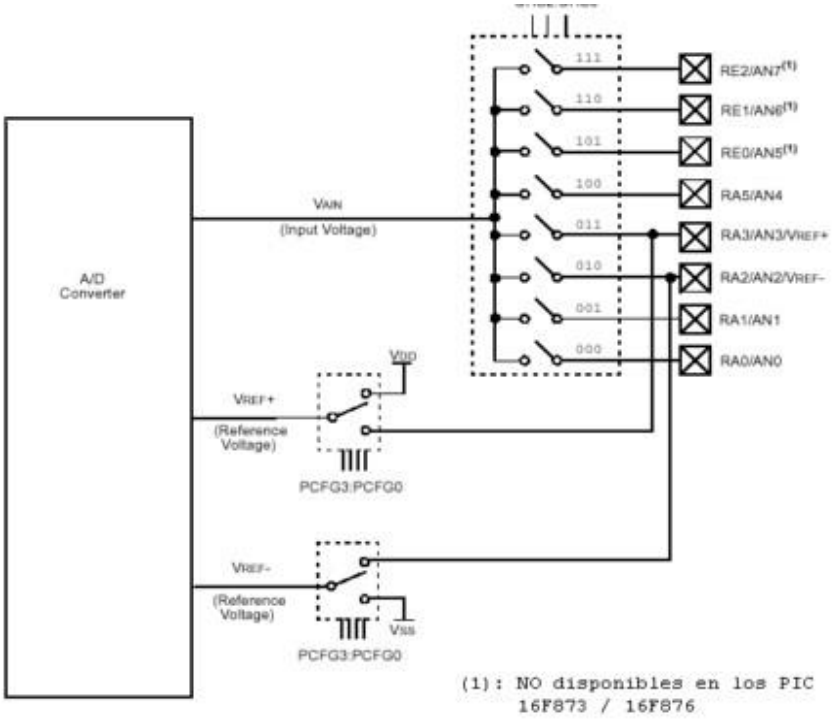


Figura 1**.** 25 Módulo del Convertidor A/D

La referencia análoga de voltaje es seleccionada por software permitiendo utilizar la fuente de alimentación del PIC (VDD) o un nivel de voltaje externo aplicado al pin 5 (RA3/AN3/ VREF +).

El módulo tiene los siguientes registros asociados:

- ADCON0: Controla la operación del módulo A/D.
- ADCON1: Configura las funciones de los pines del puerto análogo.
- ADRESL: Contiene la parte BAJA del resultado de la conversión A/D.
- ADRESH: Contiene la parte ALTA del resultado de la conversión A/D.

Registros de Control del Módulo Convertidor Análogo/Digital

Registro ADCON0.- Este es un registro que permite seleccionar cual de las entradas análogas va a ser leída y permite dar la orden de iniciar el proceso de conversión, sus ocho bits son los siguientes ver Fig. 1.26:

ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
bit7							bit 0

Tabla 4 bits del Registro ADCON0

- Bit 0 (ADON): Bit de activación del módulo. ADON = 1, Módulo A/D operando. ADON = 0, Módulo A/D desactivado.

- Bit 2 (GO/DONE): Estado de conversión: GO = 1, Empieza conversión. GO = 0, conversión finalizada. Si ADON = 0, Este bit es cero. • Bits 3, 4 y 5 (CHS0, CHS1, CHS2): Selección del canal a convertir (canal 0 - 7).

- Bits 6 y 7(ADCS0, ADCS1): Selección del reloj de conversión.

ADCS1	ADCS0	FRECUENCIA DE CONVERSIÓN
0	0	Fosc / 2
0	1	Fosc / 8
1	0	Fosc / 32
1	1	FRC

Tabla 5 Frecuencias de conversión para el módulo A/D

Registro ADCON1.- Este es un registro que permite seleccionar como se ubican los diez bits resultado de la conversión A/D y permite seleccionar cuales de los pines del puerto A trabajarán como entradas análogas y cuales como entradas digitales. Adicionalmente, permite seleccionar los voltajes de referencia del convertidor.

El bit 7 (ADFM) selecciona el formato del resultado de la conversión: Si ADFM = 1, el resultado se justifica a la derecha: Los 6 bits más significativos de ADRESH son cero.

Con los tres bits (PCFG0, PCFG1, PCFG2) se configuran los pines del puerto A como de entradas análogas o entrada/salida digital, así como la referencia de voltaje que utilizará el convertidor. Figura 1.26.

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN / REFS
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Entrada Análoga

D = Entrada/Salida Digital

Figura 1**.** 26 Selección de los canales análogos.

Cuando se completa la conversión A/D, el resultado se carga en los registros ADRESH y ADRESL (en el formato configurado por el bit ADFM).

El bit GO/DONE (ADCON0<2>) se pone en cero y el bit bandera de la interrupción A/D (ADIF) se pone en uno. Después de que el módulo ha sido configurado, al canal seleccionado se debe hacer un muestreo antes de empezar la conversión.

El tiempo requerido para el muestreo es definido como Tad.

CAPITULO II
METODOLOGÍA

2.1 TIPO DE ESTUDIO

- El método deductivo para partir de conceptos generales para estudiar de forma más detallada sus principios y características.
- Método analítico debido a obtener un conocimiento claro de cada uno de los dispositivos a utilizar en este estudio.
- haciendo uso de la investigación documental se debe recurrir a los manuales y folletos de los equipos que se utilizarán en este estudio.

2.2 POBLACIÓN MUESTRA

- **POBLACIÓN**
  Estacionamientos y de la Universidad Nacional de Chimborazo
- **MUESTRA**
  Estacionamiento de Ingeniería de la Universidad Nacional de Chimborazo

2.3 OPERACIONALIZACIÓN DE VARIABLES

- En condiciones estables el sistema no tenga errores.
- Capacidad de almacenamiento de datos.
- Contar con un reloj calendario ajustable.
- Mostrar en pantalla actividades que se estén realizando.
- Comunicación de forma serial hacia el computador y periféricos de lecturas como tarjetas RFID, y el modulo Bluetooth.
- Contar una interfaz de potencia que ayude a manejar corrientes y voltajes superiores al de su funcionamiento.
- Luces de indicación de los estados que se encuentran las salidas

- Juego de pulsadores que permitan la manipulación manual de las salidas que tiene el sistema.

2.4 PROCEDIMIENTOS

- Estudio y análisis de las condiciones del sistema existente.
- Recopilación de datos e información necesaria para el diseño del prototipo.
- Establecer los requerimientos del prototipo.
- Diseño y creación del hardware del prototipo.
- Diseño del algoritmo y programación del microcontrolador
- Prueba del funcionamiento del Prototipo
- Diseño y programación del HMI en labview .
- Prueba del funcionamiento del HMI
- Diseño y programación de la interfaz en java para el teléfono celular
- Prueba del funcionamiento de la interfaz Java para el celular
- Prueba finales.

2.5 PROCESAMIENTO Y ANÁLISIS

2.5.1 DIAGRAMA DE BLOQUES DEL SISTEMA

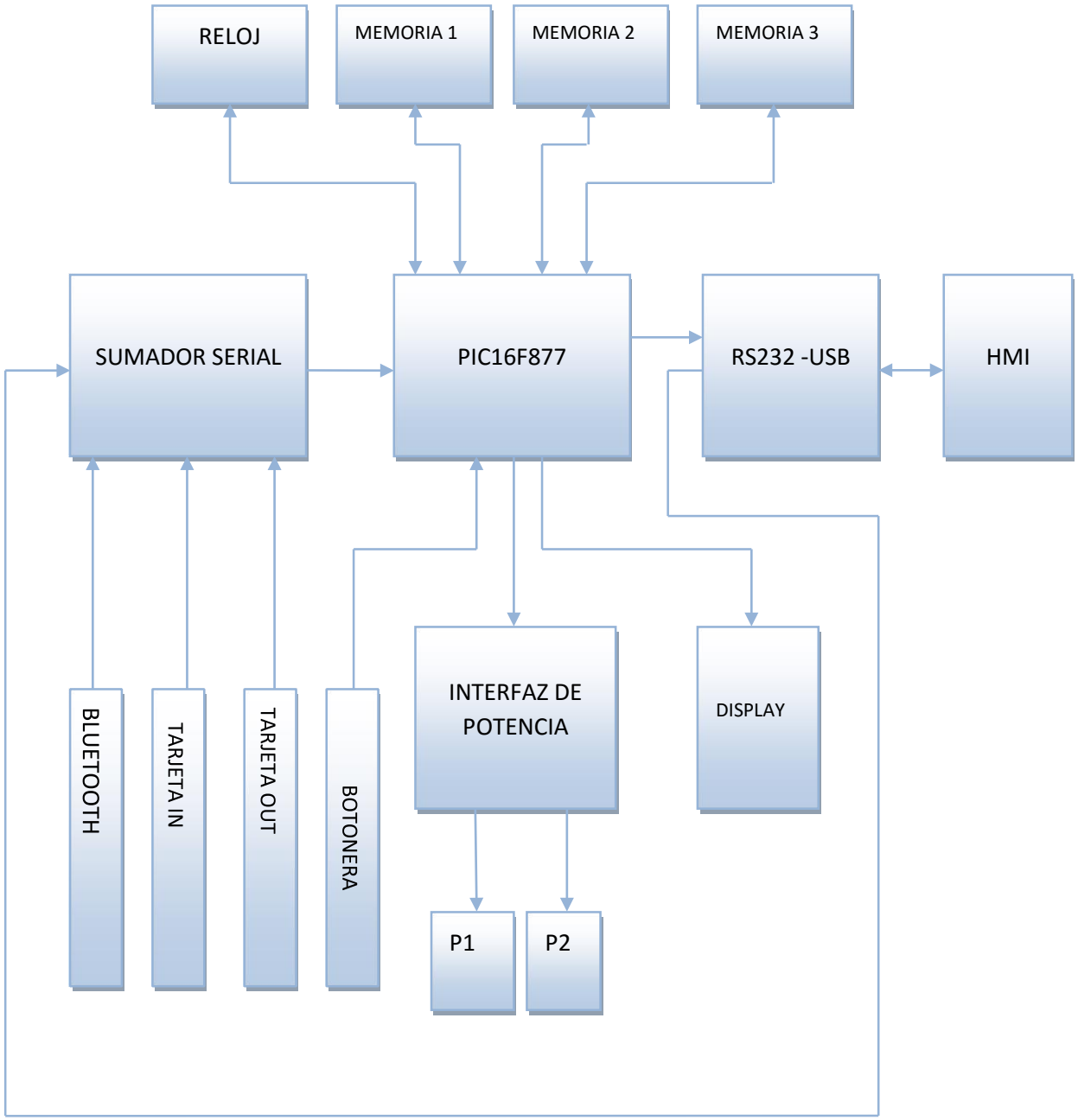


Figura 2.1 Diagrama de Bloque del Sistema

2.5.2 MATERIALES PARA IMPLEMENTACIÓN DEL HARDWARE

A continuación se describe una lista de materiales a utilizar en la construcción de hardware:

- UN MICROCONTROLADOR PIC16F877A
- DOS MICROCONTROLADORES PIC16F628A

- UN LCD 2X16
- UN MODULO BLUETOOTH RS-232 GL-6B
- UN REGULADOR DE 3.3V LM 1117
- DOS RELES DE 5V
- CINCO TRANSISTORES 2N2222
- OCHO RESISTENCIAS 10K OHMIOS, ¼ DE WATTIO
- 10 RESISTENCIA DE 10K OHMIOS, ¼ DE WATTIO
- 10 RESISTENCIA DE 4,7K OHMIOS, ¼ DE WATTIO
- CINCO RESISTENCIA DE 330 OHMIOS, ¼ DE WATTIO
- SEIS DIODOS 1N4007
- UN POTENCIÓMETRO DE 10K
- TRES CRISTAL DE 4MHZ
- DOS CAP DE 10 uF
- UN CAP DE 4700 uF
- SEIS CAP DE 22pF
- UN INTEGRADO 74LS21
- UN INTEGRADO 74LS04
- INTERFAZ USB-SERIAL
- OCHO LEDs DE ALTO BRILLO DE DIFERENTES COLORES
- ZOCALOS DE 40, 8, 16, 18 PINES
- CONECTORES MOLES DE 12, 8, 4, Y 2 PINES
- UN DS1307 RELOJ DE TIEMPO REAL
- UN CRISTAL 32768Hz
- TRES MEMORIAS I2C 24LC256
- 4 PULSADORES NORMALMENTE ABIERTOS
- UNA BATERÍA DE 3.3V CON SU RESPECTIVO ZÓCALO
- UNA FUENTE DE 5V A 500mA

A continuación se describe brevemente a los elementos más relevantes en el diseño del hardware:

2.5.2.1 Microcontrolador PIC16F877A

Este microcontrolador descrito en el CAPITULO I, es el corazón del sistema encargado de comandar a todos los periféricos que se encuentran conectados en sus pines, cómo son las memorias, el reloj etc.

2.5.2.2 Microcontrolador PIC16F628A

El PIC16f628A tiene 18 pines y siendo el modelo estrella de MicroChip, siendo además el sucesor del anterior modelo más importante (y todavía vigente) 16F84. Siendo un micro de la gama media tiene varias funcionalidades incorporadas. Es comercializado en 3 versiones que soportan velocidades de reloj diferentes, 4 MHz, 10 MHz y 20 MHz.

Estos microcontroladores tienen características especiales que permiten la reducción de componentes externos, reduciendo costos, reforzando la confiabilidad y reduciendo el consumo eléctrico.

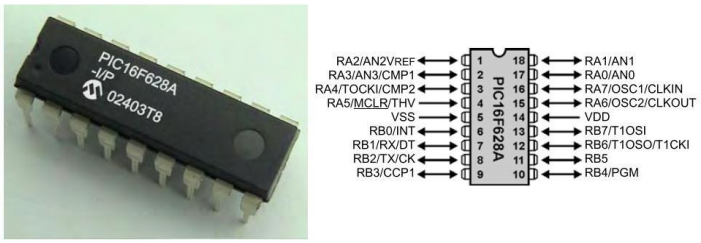


Figura 2.2 Presentación del PIC16F628A y su diagrama de pines

La función de este microcontrolador en el sistema es interpretar los datos enviados por las tarjetas y enviarlos con las respectivas etiquetas de identificación al PIC16F877A para ser analizada y tomar alguna acción.

2.5.2.3 MODULO LCD 16X2

Los módulos LCD (Display de Cristal Líquido), son utilizados para mostrar mensajes que indican al operario el estado de la maquina, o para dar instrucciones de manejo, mostrar valores, etc. permitiendo mostrar cualquier carácter ASCII, y consumen mucho menos que los displays de 7 segmentos, existen de varias presentaciones por ejemplo de 2 líneas por 8 caracteres, 2x16, 2x20, 4x20, 4x40, etc. Sin backlight (14 pines) o con backlight (16 pines, iluminado de pantalla), el LCD más popular es el 2x16, 2 líneas de 16 caracteres cada una.

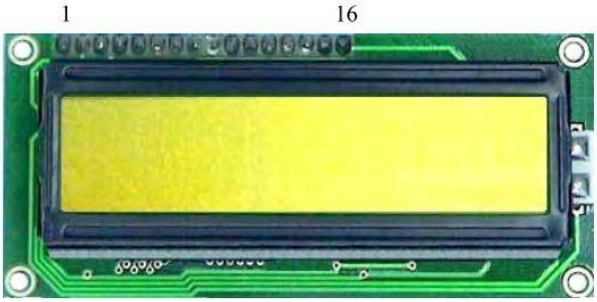


Figura 2.3 Modulo LCD 2x16

La función de este modulo es indicar al operario los estados y las actividades que se encuentra el sistema a demás de la hora y fecha.

2.5.2.4 REGULADOR DE VOLTAJE 3.3V (LM 1117)

Llamado también estabilizador de voltaje o acondicionador de voltaje es un dispositivo eléctrico que acepta una tensión de voltaje variable a la entrada, dentro de un parámetro predeterminado y mantiene a la salida una tensión constante (regulada).

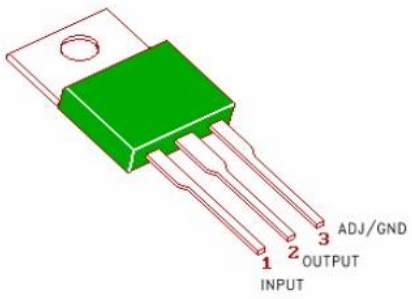


Figura 2..4 LM1117 y distribución de pines

La función de este dispositivos es mantener con una tensión estable de 3.3V al Bluetooth GL-6B tensión especificada por el fabricante para garantizar el buen funcionamiento del Bluetooth.

2.5.2.5 MODULO BLUETOOTH GL-6B

La función del Bluetooth es establecer la comunicación con el teléfono celular recibir la información y transferirla al microcontrolador PIC16F877 por medio de una conexión serial asíncrona de 9600 baudios para su interpretación.

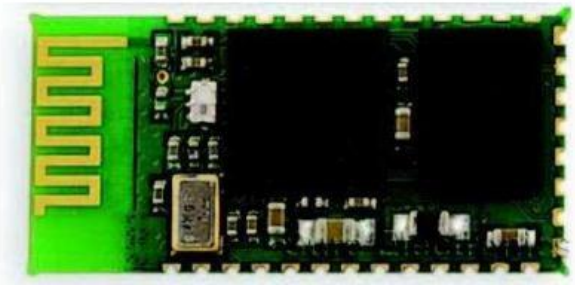


Figura 2 .5 Bluetooth GL-6B **Características:**

- Bluetooth Spec v2.0+EDR Compliant
- Bluetooth clase 2
- Operación de bluetooth a máxima velocidad
- Operación a 3.3V
- Interface UART (adaptador de comunicación serie asíncrona.)
- Componentes externos mínimos
- Mayor velocidad en conexión y mejor funcionamiento en piconet.
- Soporta scatternet.
- Interfase USB, UART, SPI, PCM.

El modulo bluetooth puede comunicarse con cualquier unidad Master como Laptop, PDA, computadoras personales y celulares. Para comunicarse con los demás dispositivos el modulo se detecta como un COM virtual.

2.5.2.6 MEMORIAS I2C (24LC256)

Para el almacenamiento de información como los códigos de usuarios, registro de entrada y salida se utiliza memorias que utilizamos tres memorias 24LC256 de 256 kbites las se comunican a través de un bus I2C.

El bus I²C (Inter Integrated Circuit) o interconexión de circuitos integrados necesita sólo 2 líneas para transmitir y recibir datos, estos son: para datos (SDA) y para la señal de reloj (SCL), esta forma de comunicación utiliza una sincronía con un tren de pulsos que viaja en la línea SCL, de tal manera que en los flancos negativos se revisan los datos RX o TX., su velocidad de transmisión pueden ser de 100Kbits/seg. en el modo standard, 400Kbits/seg. en el modo rápido y 3,4Mbits/seg. en alta velocidad. Cada dispositivo conectado al bus tiene un código

de dirección seleccionable mediante software, existiendo una relación permanente Master/Slave. El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de reloj (SCL), y el Slave es el dispositivo direccionado, sin embargo cada dispositivo reconocido por su código (dirección), puede operar como transmisor o receptor de datos, ya que la línea (SDA) es bidireccional.

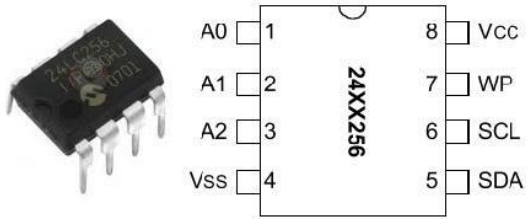


Figura 2.6 Presentación Memoria 24LC256 y su distribución de pines

2.5.2.7 RELOJ DE TIEMPO REAL (DS 1307)

La función del reloj proporcionar la hora y la fecha actual para el registro de usuarios en caso de que el sistema sea desconectado del sistema. Su funcionamiento es muy parecido a las memorias I2C con la diferencia que su código de dirección es establecido por el fabricante.

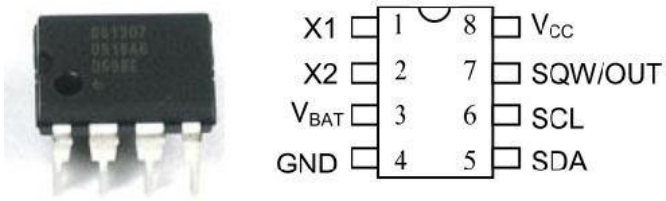


Figura 2.7 Presentación del DS 1307 y su distribución de pines

El reloj de tiempo real DS1307 necesita de una batería que lo mantiene en funcionamiento al RTC cuando no hay alimentación DC, y evita que el reloj se atrase cada vez que sea desconectado el sistema.

2.5.2.8 RELE Y TRANSSITORES

Estos dispositivos son los encargados para amplificar corriente y voltajes tanto en la parte de potencia como para la elevación del voltaje de la salida serial del Bluetooth de 3.3V a 5V los transistores utilizados son el 2N2222 y los relés son los G5CLE-1-DC5

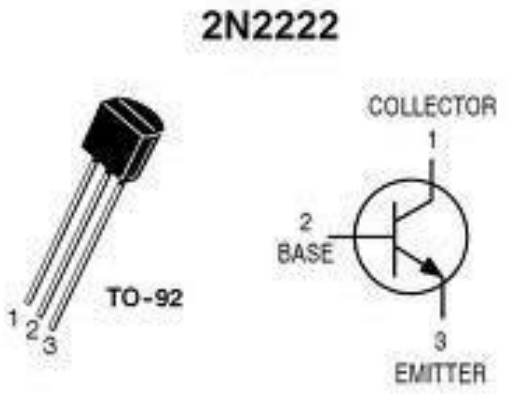


Figura 2.8 Transistor 2N2222

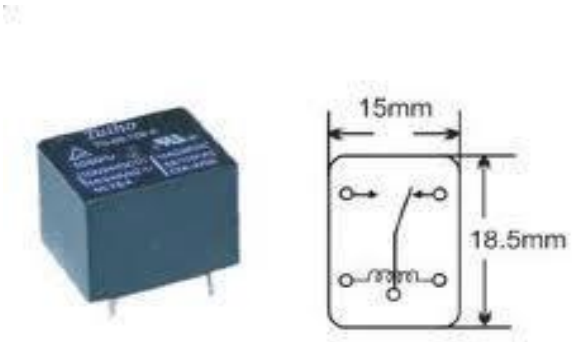


Figura 2.9 Relé G5CLE-1-DC5

2.5.3 ESQUEMA DEL HARDWARE DEL SISTEMA

Para la elaboración de este esquema se utilizo el programa Proteus (ISIS).

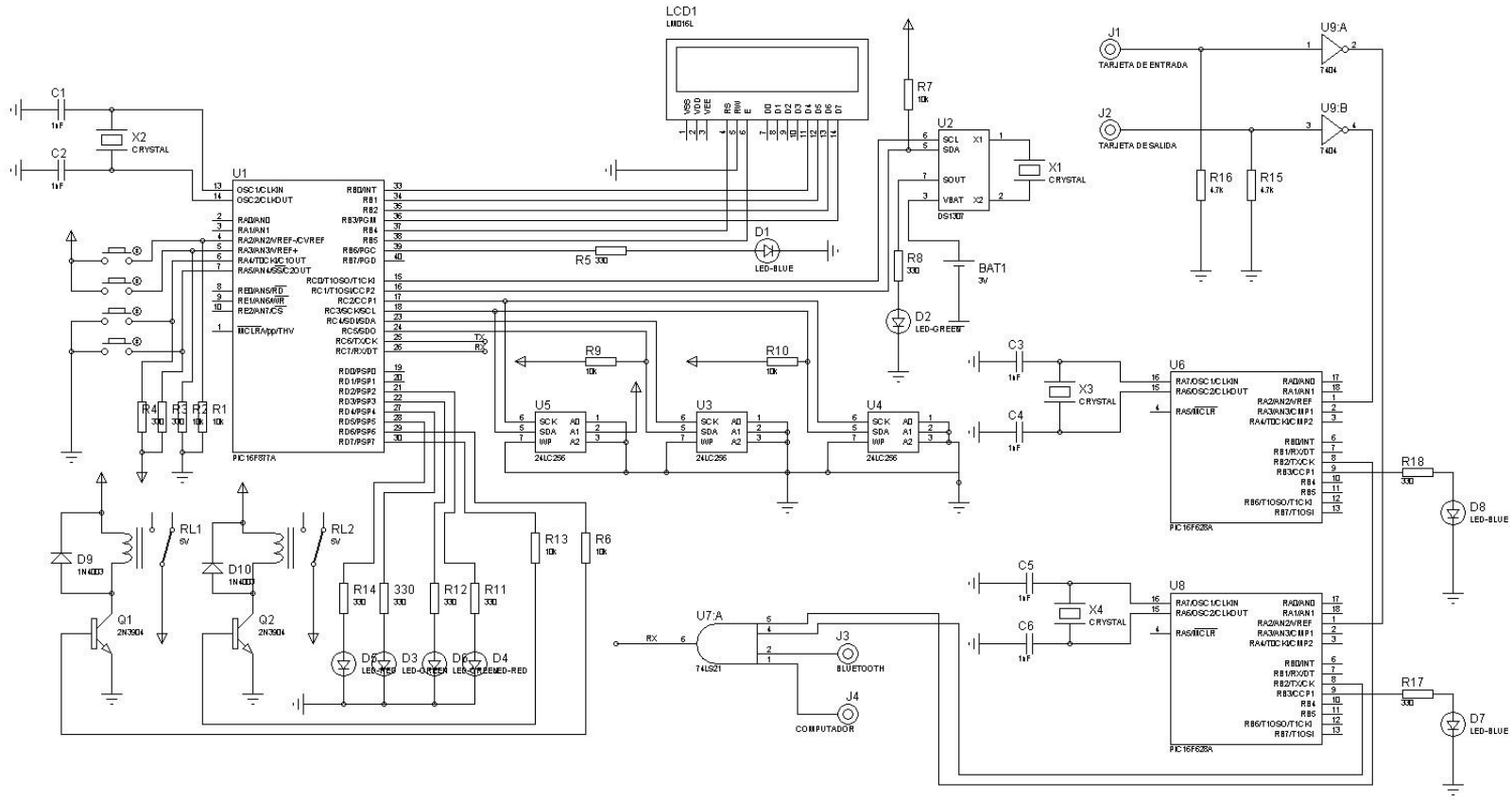


Figura 2.10 Diagrama de conexiones del sistema

2.5.4 DISEÑO DEL CIRCUITO IMPRESO

Para la realización del circuito impreso se lo lleva en programa Proteus (ARES). Para la explicación del circuito impreso se lo dividió en 4 secciones denominadas de la siguiente manera: Comunicación Serial, Potencia, Reloj, Principal

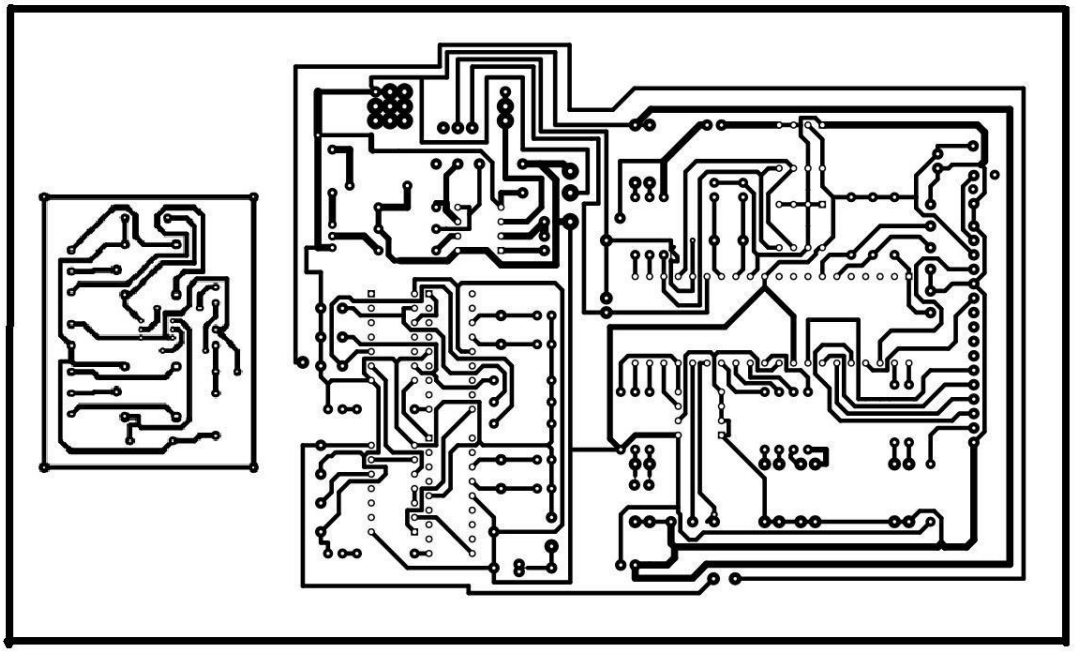


Figura 2.11 CIRCUITO IMPRESO

2.5.4.1 Comunicación Serial

En este sección llegan todas las salidas seriales las de los lectores de tarjetas, Bluetooth y del computador y las sumarisa para obtener una sola salida que va ir a la sección principal.

2.5.4.2 Sección de Potencia

Esta sección es la encargada mediante un juego de transistores y relés adaptar la señal de 5V a 12 V que necesitan los brazos mecánicos para cambiar de estado.

2.5.4.3 Sección del reloj

Esta sección es la que va a proveer de los datos de reloj y calendario a la sección principal.

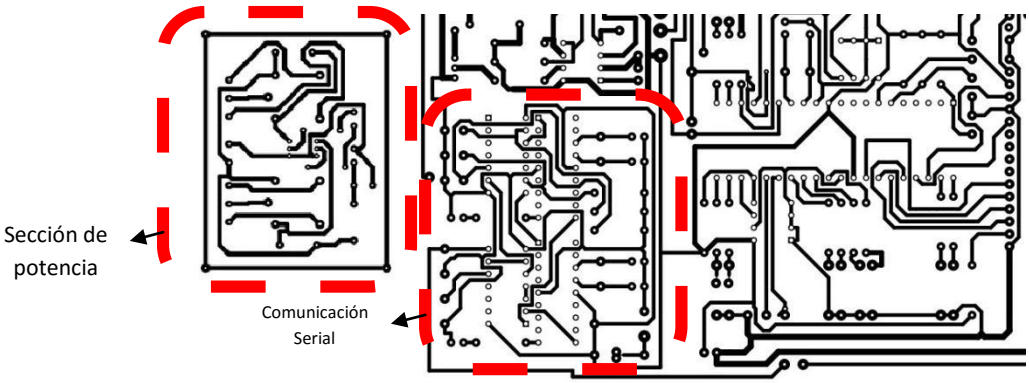


Figura 2.12 Sección de potencia

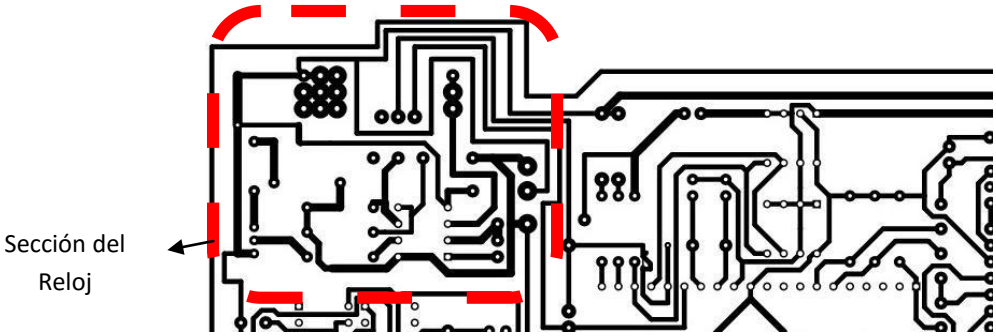


Figura 2.13 Sección de reloj

2.5.4.4 Sección principal

Es la placa encargada de comandar todas las actividades.

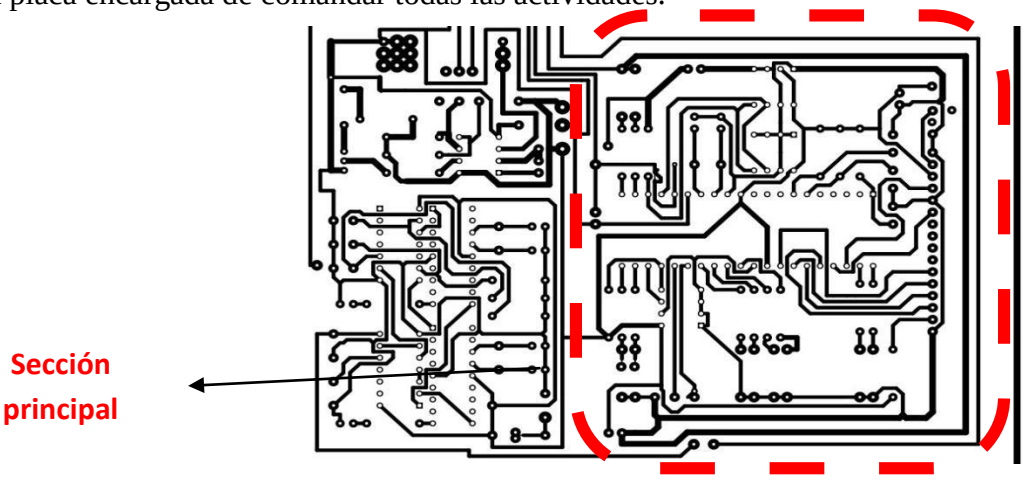


Figura 2.14 Sección Principal

2.5.5 DISEÑO DEL SOFTWARE PARA EL MICROCONTROLADOR

A continuación se explicaran las líneas de código utilizadas para que el prototipo realice las tareas necesarias para cumplir con el objetivo del sistema. Para la programación del microcontrolador se utilizara el programa Microcode Studio y para simular el programa Proteus (ISIS).

2.5.5.1 ENTORNO DE DESARROLLO MICROCODE ESTUDIO

Microcode Studio es un Entorno de desarrollo Integrado (IDE), diseñado exclusivamente para facilitar la programación de los microcontroladores PIC, los procedimientos para programar son muy sencillos.

1. Primero seleccione el modelo del PIC 16F628A, 16F877A,

2. Escriba el programa y guárdelo bajo un nombre,

3. Por último presione el botón compilar, si el programa está bien escrito y sin fallas compilará y mostrará en la parte inferior izquierda el espacio que requiere en el PIC, enseguida se creará automáticamente 3 archivos: un archivo .mac, un archivo.asm y un .hex, este último es el más importante para el PIC y es el que se debe grabar en el microcontrolador.

A continuación se muestra las partes más importantes de la presentación en pantalla del programa MicroCode Studio.

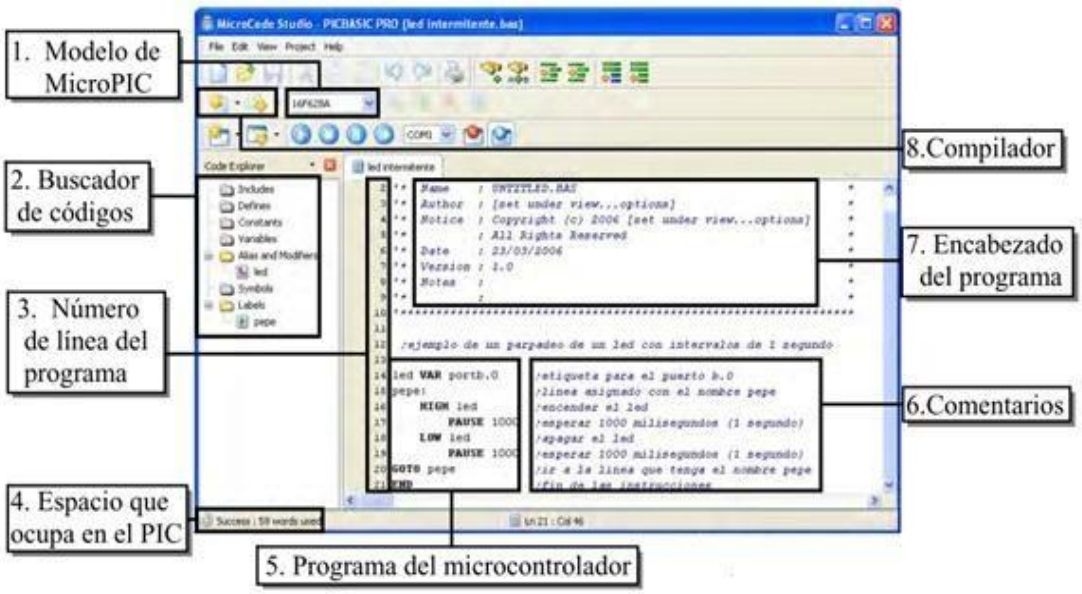


Figura 2.15 Entorno de desarrollo Microcode Studio

1. **Modelo de MicrocontroladorPIC.-** Es lo primero a seleccionar antes de empezar a programar, seleccione de acuerdo al modelo de Pic que se va a programar sea este 16F627, 16F627A, 16F628, 16F628A, 16F818, 16F819, 16F84A, 16F877A, etc.

2. **Buscador de códigos.-** Aquí se adicionan cada vez que se crea una variable, al incluir un define, o crear algún nombre de línea, sirve para saber qué componentes incluyen en el programa y también como buscador de líneas, para esto basta con dar un clic en el nombre de la línea que desea encontrar y automáticamente indicará donde está dicha línea.

3. **Número de línea del programa.-** Esto por defecto no viene habilitado, debe habilitarlo previamente, y es muy útil a la hora de encontrar errores, porque indica el número de línea en donde se halla un error.

4. **Espacio que ocupa en el PIC.-** Es el espacio que se requiere en la memoria FLASH del Pic y aparece una vez que se compila el programa, debe fijarse si alcanza en el PIC que dispone o debe reemplazarlo por otro de mayor capacidad.

5. **Programa del microcontrolador.-** En esta parte es donde se debe escribir el programa, Microcode reconoce palabras clave como VAR, HIGH, LOW, PAUSE, etc., y los pinta con mayúsculas y negrillas, por lo que no se debe utilizar estas palabras como nombres de subrutinas o variables.

6. **Los comentarios** se crean anteponiendo un punto y coma (;), el texto cambia de color de negro a azul y del tipo cursiva.

7. **Encabezado del programa.-** Son comentarios en los que se puede incluir: nombre, fecha, autor, y una explicación en breves palabras de cómo y para qué sirve el programa.

8. **Compilador.-** Puede ser ejecutar mediante 2 botones sirven básicamente para compilar el programa y crear el archivo. ASM, .MAC, y el .HEX, el .HEX sirve para grabar en el micro, el .MAC sólo sirve para el PICBasic y el .ASM, para personas interesadas en assembler ya que podemos abrirlo en MPLAB.

    - ***Compile Only*** - F9. Este primer botón sirve para compilar, es decir el programa lo cambia a assembler y lo crea el .HEX.

    - ***Compile and Program*** - F10. Este botón tiene doble función, al presionarlo compila, también puede llamar al programador Ic-prog, con la finalidad de ahorrarnos tiempo y no tener que abrir por separado

En el momento que se compila un programa este realiza una previa verificación del mismo, si existen errores microcode señala el primer error que encuentra con una franja CAFÉ, luego en la parte inferior menciona los demás errores con el número de línea y su explicación.

2.5.5.2 ENTRONO DE SIMULACIÓN PROTEUS PROFESIONAL

Proteus Professional 7.6 es un simulador de circuitos electrónicos, que cuenta con una gran variedad de componentes electrónicos, además también cuenta con animaciones para las simulaciones con una variedad de herramientas, así como el diseño de circuitos impresos.

ISIS: Es una herramienta para la elaboración avanzada de esquemas electrónicos, que incorpora una librería de más de 6.000 modelos de dispositivos digitales y analógicos que ayuda con el diseño y la simulación de circuitos electrónicos.

CARACTERÍSTICAS:

- Entorno de diseño gráfico de esquemas electrónicos extremadamente fácil de utilizar y dotado de poderosas herramientas para facilitar el trabajo del diseñador.
- Entorno de simulación prospice mixto entre el estándar SPICE3F5 y la tecnología exclusiva de Proteus de Modelación de Sistemas Virtuales (VSM).
- Entorno de diseño de placas de circuito impreso (ARES) de ultra-altas prestaciones con bases de datos de 32 bits, posicionador automático de elementos y generación automática de pistas con tecnologías de autocorte y regeneración.

2.5.5.3 DESCRIPCIÓN DEL ALGORITMO

En esta sección se indicaran y se describirán la líneas de código utilizas para darle viada al prototipo, su función y de porque forman parte del algoritmo.

2.5.5.3.1 Definición puertos y pines e inclusión de librerías

Al utilizar periféricos como pantalla LCD y elementos que se comunican a través del bus I2C se debió definir los puertos a utilizar para el LCD y la función de cada pin de ese puerto que en este caso es el puerto B. Para poder comunicarnos de forma serial con elementos como el Bluetooth y el computador se debió incluir la librería que nos permitirá comunicarnos de manera serial. Debido que no se iba a utilizar el puerto A en modo analógico debíamos indicarle al microcontrolador que el puerto A seria totalmente digital.

```
INCLUDE "modedefs.bas"; LIBRERIAS UTILZADA PARA LA COMUNICACION SERIAL

DEFINE OSC 4            'setea la frecuencia del lcd a 4mhz

DEFINE I2C_SCLOUT   1   'DEFINIMOS LA COMUNICACION I2C

DEFINE LCD_DREG PORTB   'PORT DE DATA DEL LCD'
DEFINE LCD_DBIT 0       'LCD data starting bit 0 or 4
DEFINE LCD_RSREG PORTB  'LCD register select port
DEFINE LCD_RSBIT 4      'SETEA EL PORT LCD ENABLE
DEFINE LCD_EREG PORTB   'SETEA EL BIT LCD ENABLE
DEFINE LCD_EBIT 5       'SETEA EL TAMAÑO DEL BUS LCD (4 U 8 BITS)
DEFINE LCD_BIT 4        'SETEA EL NUMERO DE LINEAS EN EL LCD
DEFINE LCD_LINES 2      'Number lines on LCD


ADCON1 = %0110          'convierte el puerto A a digital'
```

Figura 2.16 Líneas de código para definir el LCD, comunicación I2C, incluir librería para comunicación serial, y definir el puerto A como digital

2.5.5.3.2 Algoritmo para la inicialización por primera vez del sistema

Para que el sistema no tuviera ningún inconveniente a la hora de su primer inicio como por ejemplo datos no grabados por el administrador, ni por el sistema se opto por hacer que el sistema borre todo el contenido de las memorias tanto la memoria EEPROM interna del pic y tambien las memorias I2C externas. Para esto al poner en funcionamiento el Microcontrolador pregunta por el dato grabado en la dirección 10 de la memoria EEPROM del PIC16F877A si es igual a 69 y si no lo

es comienza el proceso de encerado de memoria en todas las ubicaciones de la memoria EEPROM de Microcontrolador y en las ubicaciones que hemos denominado como útiles de las memoria I2C 24LC256.

Mediante un cálculo matemático se muestra en pantalla el avance de calibración de primer inicio marcando el porcentaje de avance de la operación. Una vez que el proceso ha llegado al 100% el sistema graba en la ubicación 10 de la EEPROM interna del PIC16F877A el valor 69 de esta manera en el próximo inicio el Microcontolador no repetirá la operación al menos que el administrador la solicite.

```
READ 10,I
IF I=61 THEN INICIO

LCDOUT $FE,1
LCDOUT $FE,$83,"PREPARANDO"
LCDOUT $FE,$C2,"PRIMER INICIO"

CL:
FOR I1=0 TO 255
WRITE I1,00
NEXT
AC=6
EP=2
RUS=1
;--------------- ENCERANDO MEMORIAS PRIMER ENCENDIDO-------
k=0
FOR I2=0 TO LO
k=(I2/10)*100
I1=K/60
X=I1/10
LCDOUT $FE,1
LCDOUT $FE,$83,"PREPARANDO"
LCDOUT $FE,$C2,"PROGRESO: ", DEC2 X, "%"

i2cwrite PinSDA1,PinSCL1,contro0,I2,[0] ;guarda la H en posición 0
PAUSE 5
HIGH PORTD.1
i2cwrite PinSDA,PinSCL,contro0,I2,[0] ;guarda la H en posición 0
PAUSE 5
HIGH PORTD.0
i2cwrite PinSDA,PinSCL,contro1,I2,[0] ;guarda la H en posición 0
PAUSE 5
LOW PORTD.1
LOW PORTD.0
pause 5

NEXT

WRITE 10,61
WRITE 30,3
```

Figura 2.17 Líneas de código para el primer inicio del sistema

2.5.5.3.3 Pantalla principal, Comunicación Serial e identificación de solicitudes

La pantalla principal se presenta como se ve en la Fig. 2.18 mientras espera una instrucción por parte de los usuarios, de un administrador o por actividades propias del sistema.

Figura 2.18 Pantalla Principal

Mientras se muestra la pantalla principal el sistema está en la espera de alguna instrucción que llega en forma serial por Pin 7 del puerto C. para esto se utiliza la instrucción SERIN2 la cual permite:

- Establecer el pin y el puerto que en este caso es PortC.7
- La velocidad de la comunicación serial que en la instrucción **SERIN2** 84 equivale a 9600 baudios.
- Nos permite esperar durante un intervalo de tiempo una comunicación en el caso no se diera durante el tiempo especificado saltamos a una etiqueta estableciendo así un bucle en espera de una instrucción.
- Además del nombre de la variable a utilizar entre otra funciones mas

Sintaxis de la instrucción **SIRIN2:**

SERIN2 *DataPin*{*FlowPin*},*Mode*,{*ParityLabel*,}{*Timeout*,*Label*,}[*Item...*]

En el momento de que el sistema recibe una instrucción pasa a identificar la instrucción mediante bifurcaciones y establece si la instrucción es válida o no para así poder realizar la tarea solicitada como puede ser:

- Abrir la entrada o salida
- Registrar usuarios
- Configurar los tiempos de verificación de que conexión con el computador
- Borrar los registros de claves de usuarios, de entrada o salida
- Recalibrar todo el sistema
- Leer registros almacenados en memoria
- Actualizar la hora y fecha del reloj

El valor de la hora y fecha del reloj son actualizados aproximadamente cada segundo para mostrar en pantalla la hora real.

```
PRO:
ACT=ACT+1
READ 30,R
HIGH PORTB.6
HIGH PORTD.4
HIGH PORTD.6
GOSUB R1307

LCDOUT $FE,1
LCDOUT $FE,$83,"UNACH 2012"
LCDOUT $FE,$C1,HEX2 HORA,":",HEX2 MINU,"  ",HEX2 DIAF,"/",HEX2 MES,"/",HEX2 ANIO
IF PORTA.4=0 THEN EN
IF PORTA.5=0 THEN SA
IF ACT=AC    THEN VER
IF R=1       THEN HIGH PORTB.7
IF R=3       THEN LOW PORTB.7
LOW PORTB.6
```

Figura 2.19 Líneas de código de Bucle de espera de instrucción

```
NU:
IF IN[0]="E"  THEN ABRIRE  ; SALTAR  A LA ETIQUETA  PARA ABRIR LA PUERTA  ENTRADA
IF IN[0]="S"  THEN ABRIRS  ; SALTAR  A LA ETIQUETA  PARA ABRIR LA PUERTA  SALIDA
IF IN[0]="W"  THEN REG     ; SALTA A LA ETIQUETA PARA REGISTRAR A UN NUEVO USUARIO
IF IN[0]="H"  THEN GOSUB W1307H     ; SALTA A LA ETIQUETA PARA ACTUAIZAR RELOJ
IF IN[0]="F"  THEN GOSUB W1307F
IF IN[0]="U"  AND IN[1]="U" THEN GOSUB CLEARMU; BORRAR B.D. DE USUARIOS
IF IN[0]="I"  AND IN[1]="U" THEN GOSUB CLEARME; BORRAR REGISTROS DE  ENTRADAS
IF IN[0]="O"  AND IN[1]="U" THEN GOSUB CLEARMS;BORRAR REGISTRO DE SALIDADS
IF IN[0]="X"  AND IN[1]="U" THEN RIN          ;LEER REGISTRO DE ENTRADA
IF IN[0]="Y"  AND IN[1]="U" THEN ROUT         ;LEER REGISTRO DE SALIDAS
IF IN[0]="C"  AND IN[1]="L" THEN CL           ; BORRAR TODO
IF IN[0]="J"  THEN CLUS                       ; BORRA UN CODIGO DE USUARIO
IF IN[0]="Z"  THEN GOSUB RESP                 ; MONITEREAR COMPUTADORA GOTO
PRO
```

Figura 2.20 líneas de código para la identificación de instrucciones

2.5.5.3.4 Algoritmo para registro de claves

Una vez que el administrador envía la solicitad para registrar la clave de un usuario por medio de una comunicación serial desde el Bluetooth o desde el computador. El sistema antes de registrar la clave constata que el código no esté registrado, mostrando en la pantalla "código ya registrado" caso contrario procede a la registración.

```
REG:
CR00=0
FOR I=0 TO 100
GOSUB BUS
IF IN[1]=DIGI[0] AND IN[2]=DIGI[1] AND IN[3]=DIGI[2] AND IN[4]=DIGI[3] AND
IN[5]=DIGI[4] AND IN[6]=DIGI[5] THEN N
NEXT
GOSUB WMEM00
PAUSE 10
GOSUB RMEM00
LCDOUT $FE,$1
LCDOUT $FE,$80,"US. REGISTRADO"
LCDOUT $FE,$C0,"COD: ",DIGI[0],DIGI[1],DIGI[2],DIGI[3],DIGI[4],DIGI[5]
PAUSE 500
GOTO PRO
N:
LCDOUT $FE,$1
LCDOUT $FE,$85,"CODIGO"
LCDOUT $FE,$C4,"REPETIDO"
PAUSE 1000
GOTO PRO
```

Figura 2.21 Líneas de código para el registro de claves

2.5.5.3.5 Algoritmo para Abrir entrada o salida

El sistema al recibir la instrucción de abrir una de las puertas, el sistema debe comparar el código enviado desde el equipo solicitante y si este es verdadero ejecutar el pedido si el computador está conectado enviar el registro de la actividad al computar caso contrario guardar el registro en la memorias I2C que tiene el sistema hasta que se conecte el computar.

```
ABRIRE:
READ 30,R
GOSUB CLAVE
IF R=3 THEN C

M:
GOSUB WMEMO1

C:
serout PORTC.6,N9600,[DAT[0],DAT[1],DAT[2],DAT[3],DAT[4],DAT[5],DAT[6],10]

AR1:
HIGH PORTD.2
HIGH PORTD.5
LOW  PORTD.4

LCDOUT $FE,$1
LCDOUT $FE,$80,"<<<ABRIENDO>>>"
LCDOUT $FE,$C0," <<<ENTRADA>>> "

PAUSE 1500
BA1:
low  PORTD.5
LOW  PORTD.2
HIGH PORTD.4
PAUSE 10

SEROUT PORTC.6,N9600,["C",DAT[1],DAT[2],DAT[3],DAT[4],DAT[5],DAT[6],10];
CONFIRMA QUE EL USURIIO A ENTRADO
GOTO PRO
```

Figura 2.22 Líneas de código para abrir puerta de entrada

Las líneas de código para abrir la puerta de salida son idénticas a la de entrada con la diferencia que en PORTD.5 se utiliza PORTD.6, PORTD.2 se utiliza

PORTD.3 y PORTD.4 se utiliza PORTD.7 además que obviamente se acumula los registros en la memoria de salida

2.5.5.3.6 Algoritmo del subprograma para identificación de claves

La finalidad de este subprograma es comparar las claves registradas en las memorias de usuario y compararlas con los códigos enviados en cada petición de ingreso o de salida.

Al momento de recibir una solicitud de entrada o salida el sistema buscara en cada ubicación de memoria reorganizara, los dígitos del código y lo comprara digito por digito si el código es válido, regresa a línea de código siguiente de donde fue llamado, caso contrario mostrara en pantalla código invalido y regresara a esperar otra instrucción.

```
CLAVE:

CR00=0
FOR I=0 TO 100
LCDOUT $FE,1
LCDOUT $FE,$81,"AUTENTIFICANDO"
LCDOUT $FE,$C4,"PETICION"
GOSUB BUS

IF IN[1]=DIGI[0] AND IN[2]=DIGI[1] AND IN[3]=DIGI[2] AND IN[4]=DIGI[3] AND IN[5]=DIGI[4]
AND IN[6]=DIGI[5] THEN ABRIR
NEXT

LCDOUT $FE,$1
LCDOUT $FE,$80," <<<CLAVE>>> "
LCDOUT $FE,$C0,"<<<INVALIDA>>> "
PAUSE 100
GOTO PRO

ABRIR:

LCDOUT $FE,$1
LCDOUT $FE,$80," <<<CODIGO>>> "
LCDOUT $FE,$C0,"<<<ACEPTADO>>> "
PAUSE 100
RETURN
```

Figura 2.23 Líneas de código del subprograma de verificación de claves

2.5.5.3.7 Subprograma para grabar el Reloj de Tiempo real

La función de este subprograma es actualizar o configurar la hora del reloj del sistema, cuando la instrucción de programación de reloj es recibida el

subprograma debe recibir los datos de la hora reorganizarlo convertirlos en hexadecimal y proceder a enviarle la información al DS1307 mediante la instrucción WRITEI2C, ya que este dispositivo funciona como una memoria I2C el sistema utiliza dos subprograma uno para la grabación de la hora y otro la fecha pero su funcionamiento es idéntico

```
W1307H:
DIGI[5]=(dat[1]*10)+dat[2]
DIGI[6]=(dat[3]*10)+dat[4]
DIGI[4]=(dat[5]*10)+dat[6]

GOSUB TR

I2CWRITE DPIN,CPIN,%11010000,0,[REL[4]]         ;setear los segundos
Pause 10                                        ;retardo para grabar grabación
I2CWRITE DPIN,CPIN,%11010000,1,[REL[5]]         ;setear minutos
Pause 10
I2CWRITE DPIN,CPIN,%11010000,2,[REL[6]]         ;setear las horas
Pause 10
I2CWRITE DPIN,CPIN,%11010000,7,[$10]            ;control %00010000 para
Pause 10                                        ;encender el led cada 1 seg.
RETURN
```

Figura 2.24 Código para grabar la hora en el DS1307

```
W1307F:
DIGI[5]=(dat[1]*10)+dat[2]
DIGI[6]=(dat[3]*10)+dat[4]
DIGI[4]=(dat[5]*10)+dat[6]
GOSUB TR

I2CWRITE DPIN,CPIN,%11010000,4,[REL[5]]         ;setear día  del mes
Pause 10
I2CWRITE DPIN,CPIN,%11010000,5,[REL[6]]         ;setear mes
Pause 10
I2CWRITE DPIN,CPIN,%11010000,6,[REL[4]]         ;setear año
Pause 10
RETURN
```

Figura 2.25 Código para grabar la fecha en el DS1307

2.5.5.3.8 Algoritmo Subprogramas para el registro de usuarios

El sistema debe tener la posibilidad de grabar los códigos en una memoria, por lo que al recibir la instrucción de registro de código el subprograma va a descomponer en digitos del código y grabarlo en cada digito en una ubicación

de memoria no utilizada por otro código. Para esto debe llevar un registro de las ubicaciones utilizados para que no sean sobrescribidas.

```
WMEMO1:
i=1
FOR I2=CONTM1 TO (CONTM1+12)
i2cwrite PinSDA,PinSCL,contro0,I2,[DAT[I]] ;guarda la H en posición 0
I=I+1
PAUSE 6
NEXT
CONTM1=CONTM1+13
REN=REN+1
WRITE 2,CONTM1
WRITE 6,REN
RETURN
```

Figura 2.26 Líneas de código para el registro de usuarios

2.5.5.3.9 Subprogramas de registro de entrada y salida

Estos programa es ejecutdo cuando recibe una petición de en entrada o salida y el computador esta desconectado. La función de estos subprogramas es similar al anterior, pero con la diferencia que estos subprogramas añaden mas campos al grabar lo registros como son la hora y fecha.

```
WMEMO1:
i=1
FOR I2=CONTM1 TO (CONTM1+12)
i2cwrite PinSDA,PinSCL,contro0,I2,[DAT[I]] ;guarda la H en posición 0
I=I+1
PAUSE 6
NEXT
CONTM1=CONTM1+13
REN=REN+1
WRITE 2,CONTM1
WRITE 6,REN
RETURN
```

Figura 2.27 Código del subprograma de registro de entradas

```
WMEMO2:
i=1
FOR I2=CONTM2 TO (CONTM2+12)
i2cwrite PinSDA,PinSCL,contro1,I2,[DAT[I]] ;guarda la H en posición 0
I=I+1
PAUSE 6
NEXT
CONTM2=CONTM2+13
RSA=RSA+1
WRITE 4,CONTM2
WRITE 7,RSA
RETURN
```

Figura 2.28 Código del subprograma de registro de entradas

2.5.5.3.10 Algoritmo de los subprogramas para la lectura y transferencia de registro de entrada y salida.

La función de estos subprograma es detectar que la computadora a sido conectad al sistema, leer las memorias de registros de entrada y salida y transferir en su respectivo orden además de mostrar en pantalla el progreso se la transferencia. La transferencia de los registros se lleva a cabo mediante comunicación serial a 9600 baudios, los datos son transferidos en dos partes en la primera el código de usuario y en la segunda la hora y la fecha.

```
RIN:
J=0
I=1
FOR K=0 TO REN
LCDOUT $FE,$1
LCDOUT $FE,$81,"TRANSFIRIENDO "
LCDOUT $FE,$C2,"REG. ENTRADA "
low PORTD.0
FOR I2=J TO (J+12)
high PORTD.0
I2CREAD PinSDA,PinSCL,contro0,I2,[DIGI[I]]
I=I+1
NEXT
J=J+13
PAUSE 10
SEROUT PORTC.6,N9600,[DIGI[1],DIGI[2],DIGI[3],DIGI[4],DIGI[5],DIGI[6],"E",10]
PAUSE 10
SEROUT PORTC.6,N9600,[ DIGI[7],DIGI[8],DIGI[9],DIGI[10],DIGI[11],DIGI[12],"E",10]
NEXT
PAUSE 10
SEROUT PORTC.6,N9600,["U","N","A","C","H","X","X",10]
pause 10
LOW PORTD.0
GOSUB CLEARME
PAUSE 10
```

Figura 2.29 líneas de código de lectura y transferencia de registro de entrada

2.5.5.3.11 Declaraciones utilizadas

DECLARACIÓN	APLICACIÓN
END	Detiene la ejecución e ingresa en modo de baja potencia
FOR…NEXT	Ejecuta declaraciones en forma repetitiva
GOSUB	Llama a una subrutina BASIC en la línea especificada

GOTO	Continua la ejecución en la línea especificada
HIGH	Saca un 1 lógico (5 V.) por un pin
I2CREAD	Lee bytes de dispositivos I2C
I2CWRITE	Graba bytes de dispositivos I2C
IF..THEN..ELSE..	Ejecuta declaraciones en forma condicional
LCDOUT	Muestra caracteres en un LCD
PAUSE	Demora con resolución de 1 milisegundo (mS.)
READ	Lee byte de un chip EEPROM interno
RETURN	Continua en la declaración que sigue al último GOSUB
SERIN2	Entrada serial asincrónica (tipo BASIC Stamp 2)
SEROUT	Salida serial asincrónica (tipo BS1)
WRITE	Graba bytes en un chip EEPROM
SEROUT2	Salida serial asincrónica (tipo BS2)

Tabla 6 Declaraciones utilizadas en diseño del software

2.5.6 DISEÑO DEL HMI EN LABVIEW

2.5.6.1 REQUERIMIENTOS DEL HMI EN LABVIEW

- Indicar cuándo se está abriendo la salida o la entrada
- Poder desde el HMI Abrir la entrada o la salida
- Visualizar de la hora y la fecha
- Enlistar los registros de entrada y salida en una tabla donde se indiquen los nombres y apellidos la hora y la fecha en que ingreso o salió además de las actividades que realiza en la institución.

- Tener botones que permitan actualizar la hora y fecha del sistema
- Poder agregar usuarios
- Poder modificar la frecuencia que se monitoree si esta activo el HMI
- Poder modificar la velocidad y el puerto para la comunicación serial
- Cambiar la dirección de la ubicación de los respaldos de ingreso y salida
- Permitir desde el HMI borrar los registro de las memorias del sistema

2.5.6.2 ESTABLECIMIENTO DE LA COMUNICACIÓN SERIAL RS232

Para poder recibir datos de forma serial bajo la norma RS-232 se lo realizo de la siguiente manera como se muestra en la figura 2.30.

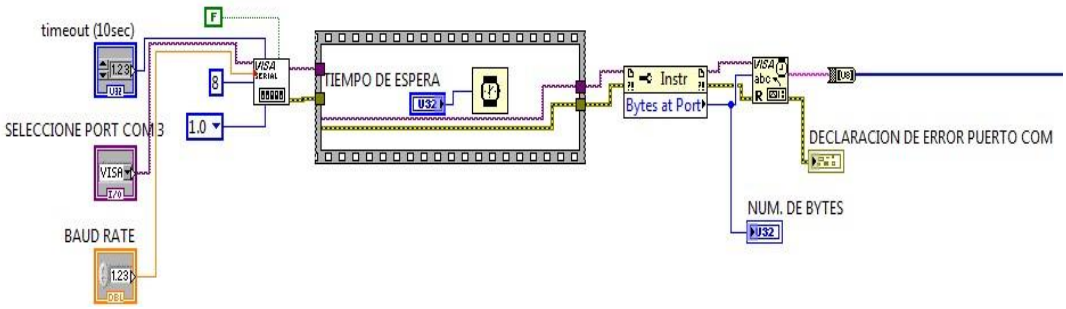


Figura 2.30 Estructura utilizada para recibir datos bajo comunicación RS-232

Los bloques utilizados para envió de datos de serial más importantes son:

- **Visa configure serial Port:** Este bloque configura las características de la comunicación serial como son la cantidad de bits de datos, el bit de parada, bit de paridad, el puerto, velocidad de transmisión, habilitar carácter de terminación que son los utilizados en nuestra HMI.

- **Property node:** Este bloque permite establecer y extraer información de la comunicación serial como son los bytes que contiene las tramas.

- **Visa Read:** Permite leer los datos que son trasmitido en la comunicación serial.

La estructura para enviar datos no difiere mucho de la de recibir ya que solo se cambia el Visa Read por un el bloque Visa Write y no se utiliza el bloque Property node.

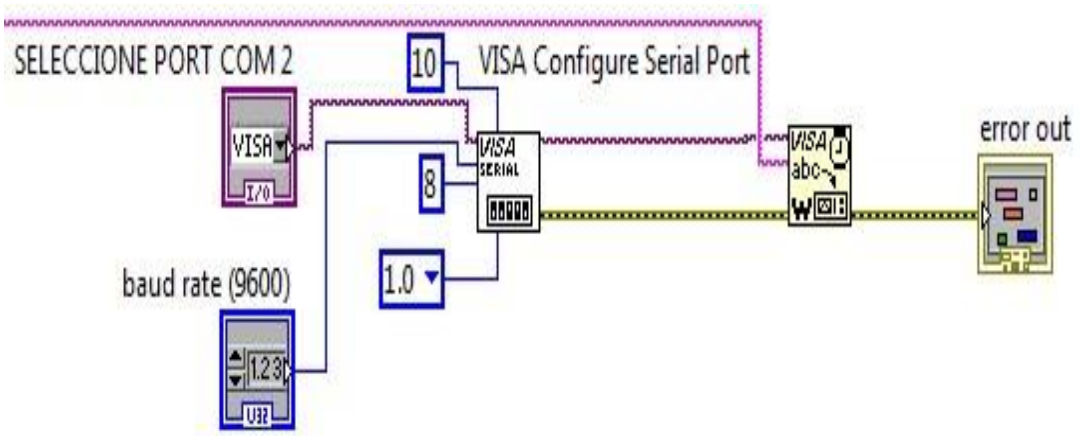


Figura 2.31 Estructura utilizada para enviar datos bajo comunicación RS-232

2.5.6.3 LECTURA Y ESCRITURA DE LA BASE DE DATOS DE USUARIOS

Para guardar los códigos de usuarios se utilizo un base de datos DB manipulable desde Microsoft Acces donde se especifican el nombre y apellidos del usuario el código de acceso, la actividad que realiza en la institución y algún detalle o observación.

La estructura utilizada para guardar la base de datos de usuario se muestra en la figura 2.32

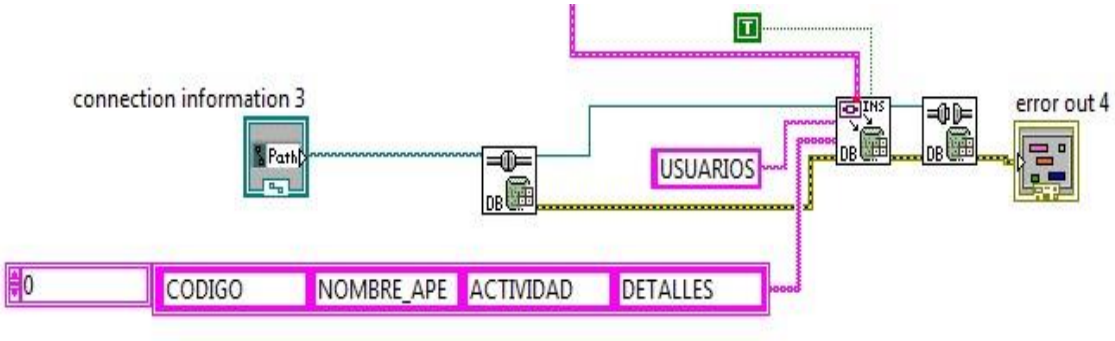


Figura 2.32 Estructura para guardar en la base de datos de usuario

- **Path:** Provee de la información de conexión para un Microsoft Data Link File.
- **DB Tools Open Connection:** Permite abrir una conxeion a una data base ubicada en una dirección especificada por el Path
- **DB Tools Insert Data: P**ermite insertar mas información a una tabla de una base de datos ubicada en la ruta especificada por el connection reference.
- **DB Tools Close Connection:** Destruye la conexión a una data base asociada con la conexión de referencia.

Mediante la activación mediante una constante booleana podemos crear la tabla si esta no existe, especificando el nombre de cada columna que tendrá la tabla.
En caso de que falle el establecimiento de conexión al momento de abrir la base de datos, insertar información o cerrar la conexión se coloco un indicador de error.

Para la lectura de la base de datos se reemplaza el DB Tools Insert Data por el bloque **DB Tools Select Data** el cual nos permitirá extraer la información de la base de datos de acuerdo a le referencia de conexión ubicada en el **Path,** además se agrega un bloque llamado **Variant to data** el cual permite los datos de la base datos a un **Labview Data** permitirá manipular de mejor manera la información ya acumularla en una variable local llamada **Table**. Al cual cual se le añade un indicador de error en caso ocurriera algún tipo de problema al leer la base de datos.

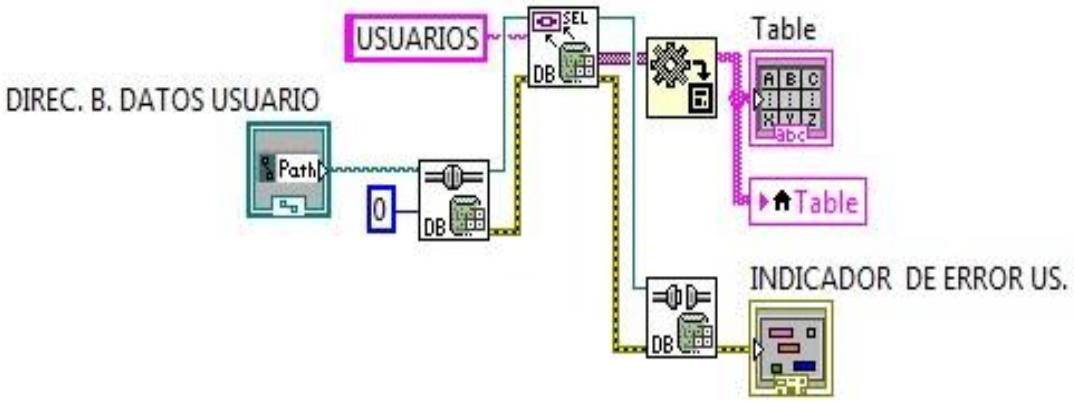


Figura 2.33 Estructura para leer la base de datos de usuario

2.5.6.4 IDENTIFICACIÓN DE LOS USUARIOS

Antes de preceder a identificar los códigos son previamente analizados en el prototipo el cual es el que autoriza y el ingreso o salida del estacionamiento una vez identificado el código en el prototipo es enviado hacia el HMI para la identificación de usuario y guardar en los registro de entrada o salida dependiendo de la petición solicitada. Después se compara el código con todos los códigos registrado en la base de datos para su identificación.

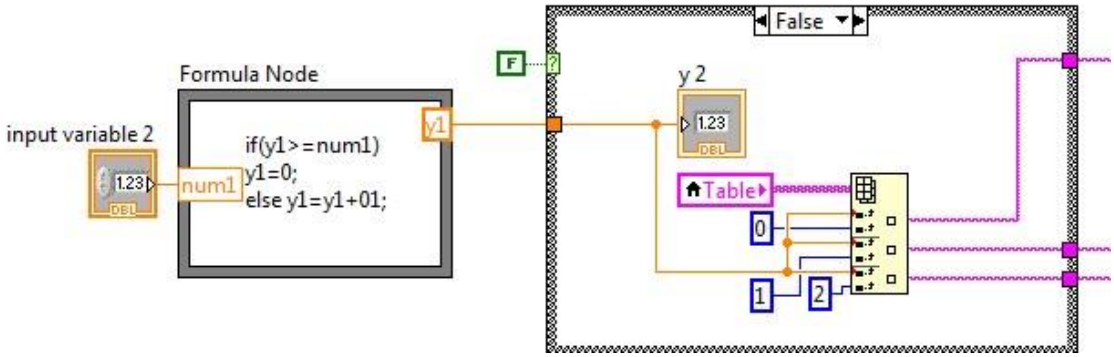


Figura 2.34 Estructura para buscar los datos en la base de datos

El código es acumulado de forma temporal en una tabla y reconstruido mediante un bloque llamado **Concatene** hasta su comparación luego es eliminado.

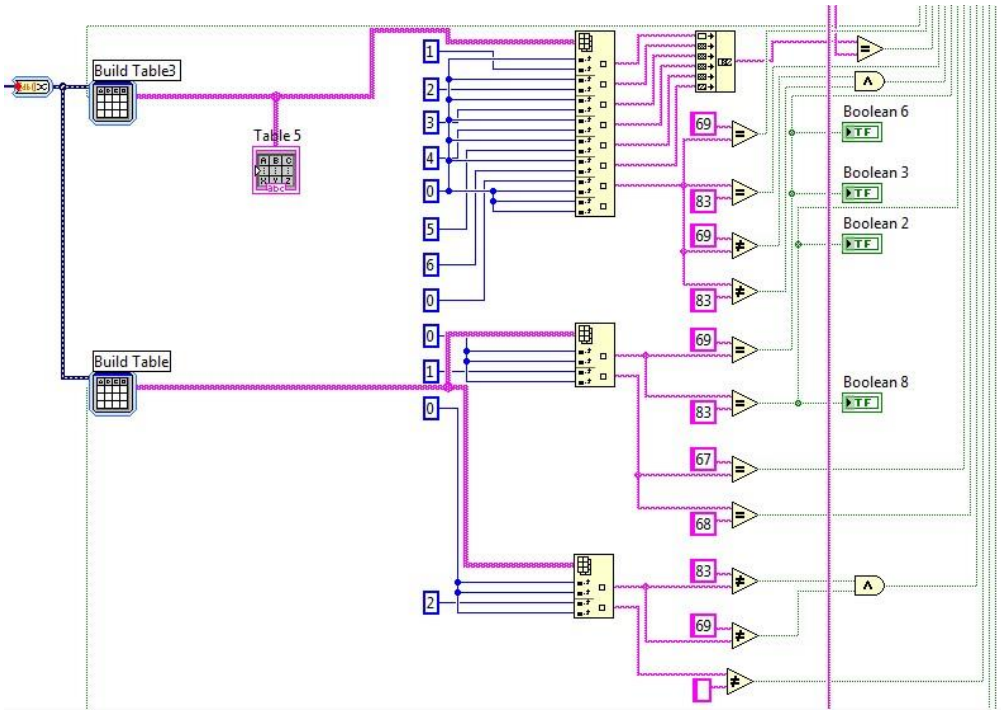


Figura 2.35 Estructura para la comparación de códigos

La comparación de códigos es realiza mediante comparaciones lógicas la cual al ser verdaderas activan el registro de entrada o salida dependiendo cual sea el caso y eliminan el registro temporal del código enviado por el prototipo.

Los registros de entrada o salida son guardados en un archivo .txt y en una base de datos de Microsoft Access.

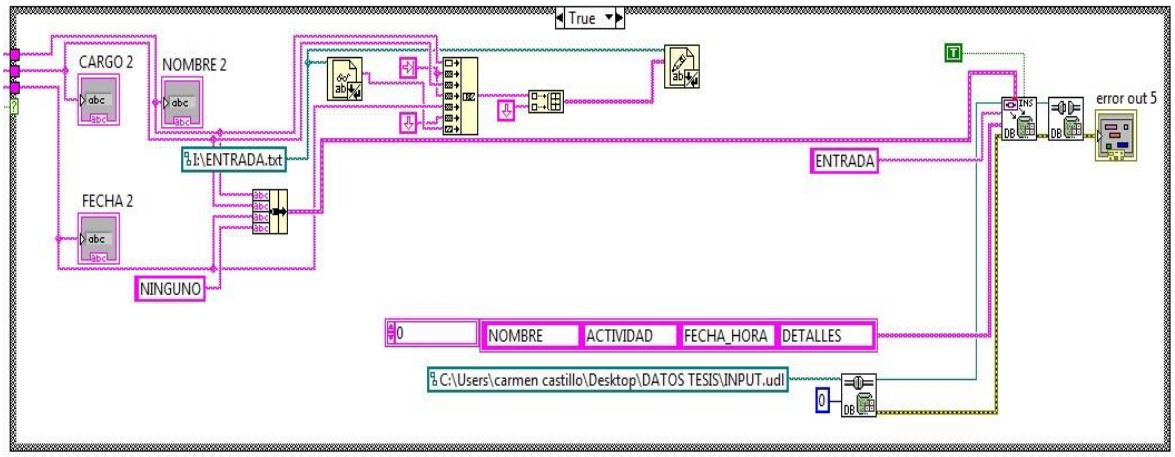


Figura 2.36 Estructura para el registro de entradas

Como se observa en la Fig. 2.37, la forma que guardan los registros de entra es idéntica que para los registros de usuarios con la incorporación de un **Bundle** que su función es hacer un cluster con el ensamblaje de elementos individuales. Esos elementos son la hora y fecha, nombre del usuario, su actividad en la institución y alguna observación o detalle.

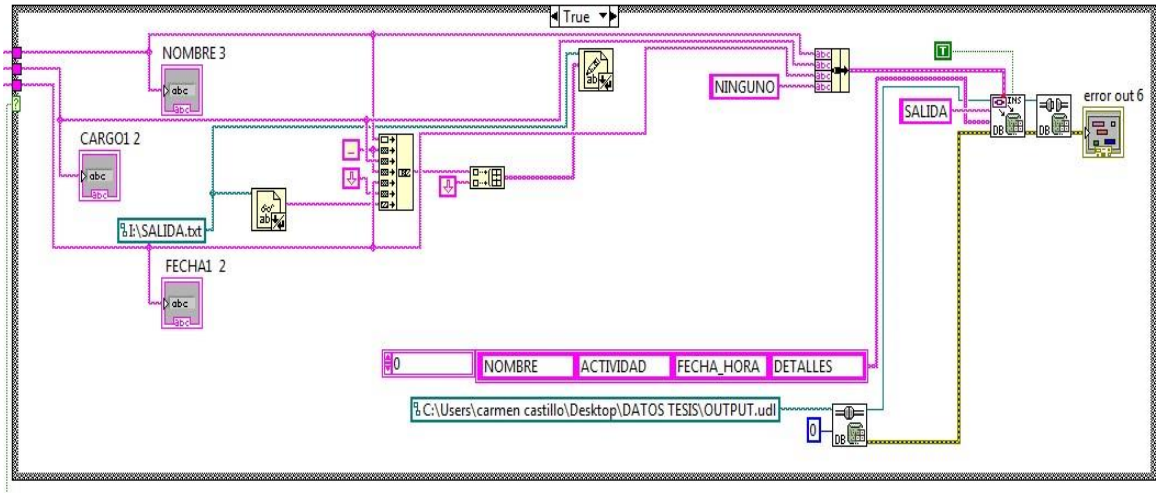


Figura 2.37 Estructura para el registro de salidas

2.5.6.5 INSTRUCCIONES ENVIADAS DESDE EL HMI HACIA EL PROTOTIPO

El envió de instrucciones hacia el prototipo es utilizada la estructura **Formule Node** la cual mediante bifurcaciones se logra detectar la petición del Operario o del sistema.

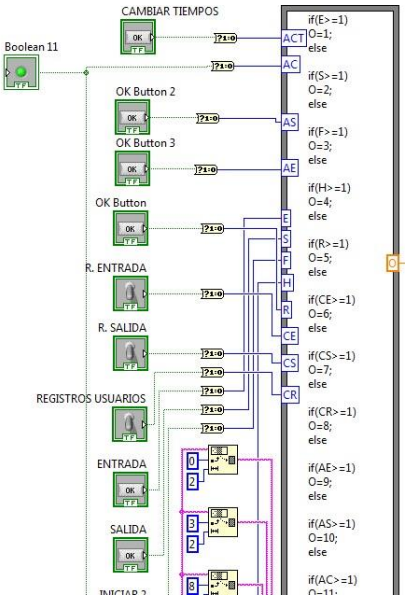


Figura 2.38 Identificación de instrucciones con la estructurara Formula Node

Dependiendo de las solicitud el **Formule Node** asigna un valor entre el cero y el 12 a su salida la cual llega a una estructura de **Case** la cual realiza una función pre establecidas en cada caso de dicha estructura enviando la instrucción en forma de un string de caracteres por medio activando la comunicación serial cada vez que se requiera transmitir.

- **Caso 0:** Este es el caso por defecto el cual no realiza ninguna acción
- **Caso 1 y 2:** Este caso se encarga de enviar la instrucción de abrir la entrada y salida respectivamente.
- **Caso 3: A**ctualización de la fecha del prototipo.
- **Caso 4:** Envía la instrucción con los datos para actualizar la hora del prototipo.
- **Caso 5:** Solicita el registro de un usuario.
- **Caso 6:** Borrar los registros de entradas del prototipo.
- **Caso 7:** Elimina el registro de salidas del prototipo.
- **Caso 8:** Elimina los registro de usuarios del prototipo.
- **Caso 9:** Leer registros de entrada del prototipo.

- **Caso 10:** Lee los registro de salida del prototipo.
- **Caso 11:** Reinicia el sistema.
- **Caso 12:** Cambia los tiempos de monitoreo de actividad del HMI.
- **Caso 13:** Borra un código de usuario almacenado.
- **Caso 14:** Conectarse con el prototipo.

La estructura a utilizar para el envío del código al prototipo es la que se encuentra en la Fig. 2.39

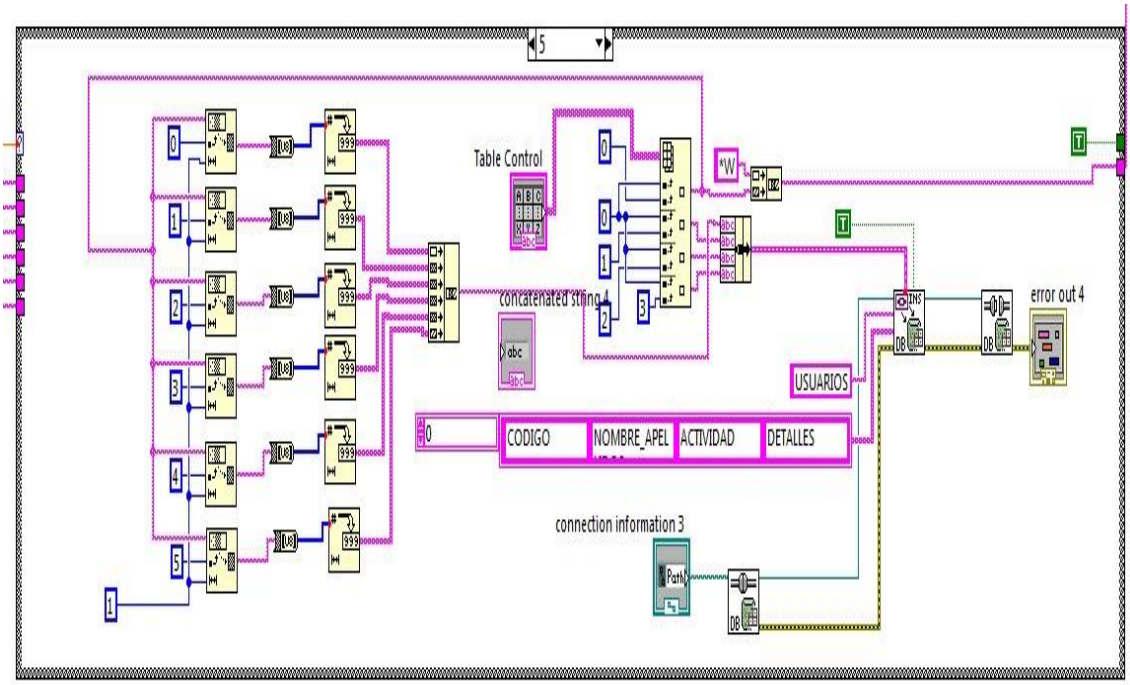


Figura 2.39 Estructura para el registro de usuarios

Debido a que el sistema está diseñado para trabajar con caracteres se debió descomponer el código en dígitos y convertirlo en un decimal string mediante el bloque **Number to Decimal String** y concatenarlos nuevamente antes de ser enviados hacia el prototipo.

2.5.6.6 TRANSFERENCIA DE REGISTRO DE ENTRA Y SALIDA

Automaticamente el HMI debe incorporar los registros de entrada o de salida almacenados en las memorias I2C automáticamente los cuales deben ser enviados respetando su orden desde el primero al último tanto de entrada como de la salida.

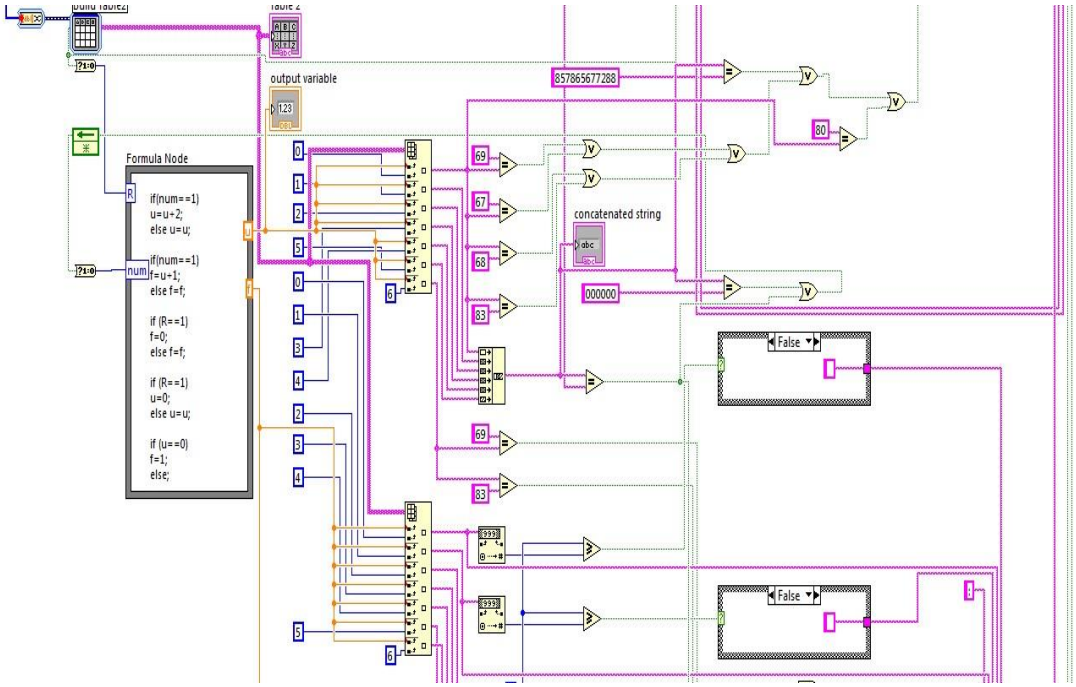


Figura 2.40 Estructura para la transferencia de registro de entra y salida de el prototipo

Esta estructura no difiere mucho de la explicada en la **sección 2.5.6.4** de ese Capitulo la diferencia está en los valores de la hora y fecha los cuales no son proporcionados por el reloj de la computadora, son extraídos desde las memorias de registros de entrada y salida del prototipo siendo captados a su debido tiempo del reloj propio del sistema sincronizado previamente con el reloj del computador.

Todos los datos se acumulan en una tabla de registro temporal mediante unas líneas de código colocadas en una Estructura de **Formule Node** basado en bifurcaciones a la identificación de los códigos y reorganizados para ser añadidos a la base de datos ya existente. Una vez terminado la transferencia se envía la instrucción de para borrar las memorias de registros de entrada y salida del prototipo.

Esta operación de transferencia es realizada después que el HMI estuvo inactivo, porque durante ese tiempo las memorias del prototipo comienza a grabar, el prototipo a constatar la actividad del HMI procede a la transferencia de la información.

El código utilizado en la **Formula Node** es el mostrado en la Fig. 2.41

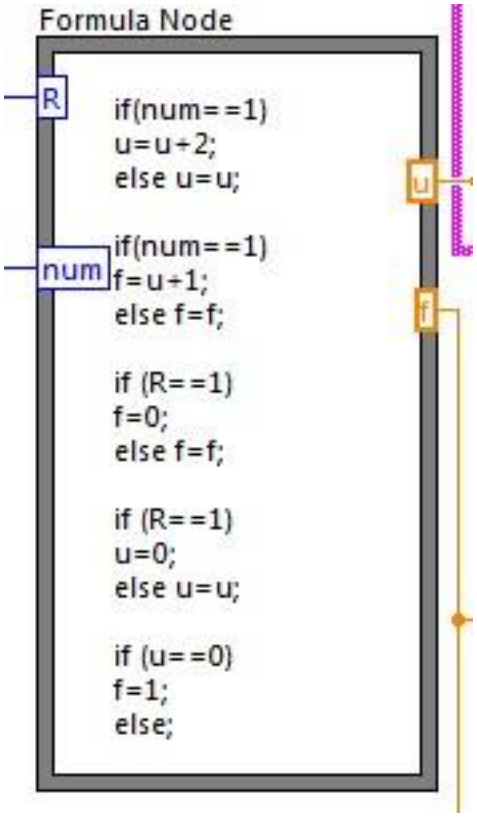


Figura 2.41 Formula Node utilizado en la transferencia y reorganización de registros de entrada y salida

2.5.7 DISEÑO Y DESCRIPCIÓN DE LA APLICACIÓN DEL TELÉFONO CELULAR

En esta sección se indicaran y se describirán secciones de las líneas de código utilizadas para el diseño de la Aplicación del teléfono.

La Aplicación debe realizar tres tareas principales:

- Establecer Comunicación con el modulo Bluetooth GL-6B.
- Guardar por primera vez el código de usuario.
- Extraer el código usuario y enviarlo cuando se lo requiera hacia el modulo Bluetooth GL-6B.

La aplicación contara con una interfaz grafica muy intuitiva y agradable a la vista del usuario.

2.5.7.1 DISEÑO DE LAS INTERFACES GRAFICAS

Para que el usuario monitoree los estados que se encuentra la aplicación se crearon pantallas que le permitirán observar las instancias que se encuentra la petición que solicito. La aplicación cuenta con cuatro pantallas distintas.

Pantalla Principal: Esta pantalla es presentada al iniciar la aplicación la cual contiene los comandos para entrar o salir y el comando para el registro de la clave el cual aparece solo en la primera ejecución de la aplicación.

En la Fig. 2.42 se muestra la pantalla principal y en la Fig. 2.43 un segmento de código utilizada para crear dicha pantalla.

Figura 2.32 Pantalla Principal

```
principal = new Form("
CRTL_ESTACIONAMIENTO"); INN = new Command("IN",
Command.ITEM, 1); OUTT = new Command("OUT",
Command.ITEM, 1); listo= new Command("REC",
Command.ITEM, 1); principal.addCommand(INN);
principal.addCommand(listo);
principal.addCommand(OUTT);
principal.append(img);
```

Figura 2.43 Líneas de Código de la Pantalla Principal

Pantalla de envío de petición: Esta pantalla se muestra cuando el usuario a requerido entrar o salir del estacionamiento**,** esta pantalla incorpora los botones cancelar el regresa a la pantalla principal. En la Fig. 2.44 se muestra la pantalla de envió de petición.

Figura 2.44 Pantalla de envío de petición

```
sal= new Command("Salir", Command.ITEM, 1);

okk= new Command("OK", Command.ITEM, 1);cancelar = new Command("Cancelar", Command.BACK, 2);
busqueda = new Form("ENVIANDO CODIGO");
texto2 = new TextField(" INGENIERIA \n  ELECTRONICA\n Y \n
TELECOMUICACINES        ","    TESIS ESTACIONAMIENTO
",50,TextField.UNEDITABLE);
busqueda.append(new Gauge("Buscando servicio...", false,

Gauge.INDEFINITE, Gauge.CONTINUOUS_RUNNING));busqueda.append(img1);

busqueda.append(texto2);

busqueda.addCommand(cancelar);busqueda.addCommand(okk);
```

Figura 2.45 Líneas de Código de envío de petición

Pantalla de autentificación de Administrador: Esta pantalla aparece cuando el administrador procede a registrar la clave para el usuario se realiza en el primer inicio de aplicación, esta pantalla incorpora un botón de cancelar para poder regresar a la pantalla principal

Figura 2.46 Pantalla de autentificación de Administrador

```
Aceptar= new Command("OK",Command.OK,1);
usuario= new Form("AUTENTIFICACION");
usuario.append(texto5); usuario.append(img2);
usuario.addCommand(Aceptar);
usuario.addCommand(cancelar);
```

Figura 2.47 Líneas de Código Pantalla de autentificación de Administrador

Pantalla de registro de código de usuario: En esta pantalla muestra un cuadro de texto en el cual es digitado el código para el usuario y el botón cancelar que nos permitirá regresar a la pantalla principal.

Figura 2.48 Pantalla para registro de código de usuario

2.5.7.2 COMUNICACIÓN ENTRE EL CELULAR Y EL BLUETOOTH GL-6B

Para lograr el establecimiento de conexión, siendo el teléfono celular el dispositivo maestro debe realizar las siguientes funciones:

- Habilitar el dispositivo Bluetooth Local

- Búsqueda de dispositivos. La aplicación realizará una búsqueda de los dispositivos Bluetooth a su alcance que estén en modo conectable.

- Búsqueda de servicios. La aplicación realizará una búsqueda de servicios por cada dispositivo encontrado.

- Establecimiento de la conexión. Una vez encontrado un dispositivo que ofrece el servicio deseado se podrá realizar la conexión.

- Comunicación. Ya establecida la conexión es posible enviar y recibir información

2.5.7.2.1 HABILITACIÓN DEL DISPOSITIVO LOCAL

Los objetos Bluetooth esenciales en una conexión son LocalDevice y RemoteDevice. La clase LocalDevice provee acceso y control del dispositivo local. Las clases DeviceClass y BluetoothStateException dan soporte a la clase LocalDevice.

La clase RemoteDevice representa un dispositivo remoto y provee métodos para obtener información de dicho dispositivo remoto.

Un objetoLocalDevice representa al dispositivo local. Este objeto será el punto de partida de prácticamente para nuestra aplicación. Se puede también obtener informacion a través de este objeto, por ejemplo, la dirección Bluetooth de nuestro dispositivo, el apodo o "friendly-name"

(también llamado "Bluetooth device name" o "user-friendly name"). A través de este objeto también podemos obtener y establecer el modo de conectividad: la forma en que nuestro dispositivo está o no visible para otros dispositivos.

Los posibles valores que admite el método setDiscoverable() están definidos en la clase DiscoveryAgent como campos estáticos. El valor que se usa para esta clase es GIAC, General/Unlimited Inquiry Access Code. En la Fig. 2.49 se muestra el código utilizada para habilitar y controlar el dispositivo local.

Al llamar al métodogetLocalDevice() se puede producir una excepción del tipoBluetoothStateException.

Esto significa que no se pudo inicializar el sistema Bluetooth. La excepciónBluetoothStateException extiende de java.io.IOException y no añade ningún método adicional. Por lo cual se añadió que se muestre un mensaje tipo alerta que no se pudo hacer uso del bluetooth.

```
try { localDevice =
LocalDevice.getLocalDevice();
localDevice.setDiscoverable(DiscoveryAgent.GIAC)
; discoveryAgent =
localDevice.getDiscoveryAgent();
Display.getDisplay(this).setCurrent(principal);

} catch(Exception e) {

Alert alert = new Alert("Error", "No se puede
hacer uso de Bluetooth", null, AlertType.ERROR);
Display.getDisplay(this).setCurrent(alert);
}
```

Figura 2.49 Código para habilitación del Bluetooth local

2.5.7.2.2 BÚSQUEDA DE DISPOSITIVOS

Cualquier aplicación puede obtener una lista de dispositivos a los que es capaz de encontrar, usando, o bien startInquiry() (no bloqueante) o retrieveDevices() (bloqueante). startInquiry() requiere que la aplicación tenga especificado un listener, el cual es notificado cuando un nuevo dispositivo es encontrado después de haber lanzado un proceso de búsqueda. Por otra parte, si la aplicación no quiere esperar a descubrir dispositivos (o a ser descubierta por otro dispositivo) para comenzar, puede utilizar retrieveDevices(), que devuelve una lista de dispositivos encontrados en una búsqueda previa o bien unos que ya conozca por defecto.

interface javax.bluetooth.DiscoveryListener: Este interfaz permite a una aplicación especificar un evento en el listener que reaccione ante eventos de búsqueda. También se usa para encontrar dispositivos. El método deviceDiscovered() se llama cada vez que se encuentra un dispositivo en un proceso de búsqueda. Cuando el proceso de búsqueda se ha completado o cancelado, se llama al método inquiryCompleted(). Este método recibe un argumento, que puede ser INQUIRY_COMPLETED, INQUIRY_ERROR o INQUIRY_TERMINATED, dependiendo de cada caso.

interface javax.bluetooth.DiscoveryAgent: Esta interfaz provee métodos para descubrir dispositivos y servicios. Para descubrir dispositivos, esta clase provee del método startInquiry() para poner al dispositivo en modo de búsqueda, y el método retrieveDevices() para obtener la información de dispositivos previamente encontrados. Además provee del método cancelInquiry() que permite cancelar una operación de búsqueda.

```
private void INOUT() {

try {

discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
Display.getDisplay(this).setCurrent(busqueda);

} catch(BluetoothStateException e) {
Alert alert = new Alert("Error", "No se pudo comenzar la busqueda",
null, AlertType.ERROR);
Display.getDisplay(this).setCurrent(alert);
}
}
private void cancelar() {
discoveryAgent.cancelInquiry(this);
Enumeration en = busquedas.elements();
Integer i;
while(en.hasMoreElements()) {
i = (Integer) en.nextElement();
discoveryAgent.cancelServiceSearch(i.intValue());
}
}
```

Figura 2.50 Código para la búsqueda de dispositivos

2.5.7.2.3 BÚSQUEDA DE SERVICIOS

Para realizar una búsqueda de servicios también usaremos la claseDiscoveryAgent e implementaremos la interfazDiscoveryListener. Para comenzar la búsqueda usaremos searchServices() de la clase DiscoveryAgent. Este método es un poco complejo así que lo vamos a ver con más detalle: public int searchServices(int[] attrSet, UUID[] uuidSet, RemoteDevice btDev, DiscoveryListener discListener) throws BluetoothStateException;

El primer argumento es un array de enteros con el que especificaremos los atributos de servicio en los que estamos interesados. Como veremos más adelante, cuando hablemos de la interfazServiceRecord, los servicios son descritos a través de atributos que son identificados numéricamente.

El segundo argumento es un array de identificadores de servicio. Nos permite especificar los servicios en los que estamos interesados.

El tercer argumento es el dispositivo remoto sobre el que vamos a realizar la búsqueda.

Para el cuarto argumento usaremos DiscoveryListener notificara los eventos de búsqueda de servicios.

public void serviceSearchCompleted(int transID, int respCode): Este método es llamado cuando se finaliza un proceso de búsqueda. El primer argumento identifica el proceso de búsqueda (que recordemos que es el valor devuelto al invocar el métodosearchServices() de la claseDiscoveryAgent). El segundo argumento indica el motivo de finalización de la búsqueda.

public void servicesDiscovered(int transID, ServiceRecord[] sr): Este método nos notifica que se han encontrado servicios. El primer argumento es un entero que es el que identifica el proceso de búsqueda. Este entero es el mismo que devolviósearchDevices() cuando se comenzó la búsqueda.

El segundo argumento es un array de objetosServiceRecord. Un objetoServiceRecord describe las características de un servicio bluetooth.

Un servicio Bluetooth se describe mediante atributos, los cuales se identifican numéricamente, es decir, un servicio Bluetooth tiene una lista de pares identificador-valor que lo describen. El objetoServiceRecord se encarga de almacenar estos pares. Los identificadores son números enteros y los valores son objetos de tipo DataElement. Los objetosDataElement encapsulan los posibles tipos de datos mediante los cuales se pueden describir los servicios bluetooth. Estos tipos de datos son: valor nulo, enteros de diferente longitud, arrays de bytes, URLs, UUIDs, booleanos,Strings o enumeraciones (java.util.Enumeration) de los tipos anteriores.

En un área de cobertura se pueden encontrar otros dispositivos Bluetooth a más del módulo usado en este proyecto.

Para identificar un dispositivo Bluetooth se puede aprovechar alguna característica que lo hace diferente de los demás, por ejemplo, su friendlyName o su Bluetooth Address.

```
public void deviceDiscovered(RemoteDevice remoteDevice,DeviceClass deviceClass) {

String address = remoteDevice.getBluetoothAddress();
String friendlyName = null;
remoteDevice.getBluetoothAddress();
try {
friendlyName = remoteDevice.getFriendlyName(true);
} catch(IOException e) { }
device = null;
if(friendlyName == null) {
device = address;
} else {
device = friendlyName + " ("+address+")";
}
if(     friendlyName == null ? "sure" == null : friendlyName.equals("sure")){
try {
int transId = discoveryAgent.searchServices(ATRIBUTOS, SERVICIOS,remoteDevice,
this);
System.out.println("Comenzada busqueda de serivicios en: "+device+"; "+transId);
busquedas.addElement(new Integer(transId));

} catch(BluetoothStateException e) {
System.err.println("No se pudo comenzar la busqueda");
}
}
}
```

Figura 2.51 Código para la búsqueda de Servicio

2.5.7.2.4 ESTABLECIMIENTO DE LA CONEXIÓN

Antes de que un cliente SPP pueda establecer una conexión con un servicio SPP, éste debe previamente haber descubierto el servicio mediante el servicio discovery. Una conexión URL del cliente incluye la dirección Bluetooth del dispositivo servidor y el identificador de canal del servidor. El método getConnectionURL() en el interfaz ServiceRecord se usa para obtener la conexión URL del cliente.

La URL, necesaria para realizar la conexión, se obtiene a través del método getConnectionURL() de un objeto ServiceRecord. Un objeto ServiceRecord representa un servicio, es decir, una vez encontrado el servicio deseado (un objeto ServiceRecord), él mismo proveerá la URL necesaria para conectarse a él.

Este método requiere dos argumentos, el primero de los cuales indica si se debe autenticar y/o cifrar la conexión. Los posibles valores de este primer argumento son:

- **ServiceRecord.NOAUTHENTICATE_NOENCRYPT:** No se requiere ni autenticación ni cifrado.

- **ServiceRecord.AUTHENTICATE_NOENCRYPT**: Se requiere autenticación, pero no cifrado.

- **ServiceRecord.AUTHENTICATE_ENCRYPT**: Se requiere tanto autenticación como cifrado.

El segundo argumento del método getConnectionURL() es un booleano que especifica si nuestro dispositivo debe hacer de maestro (true) o bien no importa si es maestro o esclavo (false). url=service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYP

T, false);

Una vez que se tiene la URL, se utiliza el método Connector.open() para realizar la conexión. Este método devuelve un objeto distinto según el tipo de protocolo usado. En el caso de un cliente SPP devolverá un StreamConnection. A partir del StreamConnection se obtienen los flujos de entrada y de salida:

```
ServiceRecord service = null;
for(int i=0; i<servRecord.length; i++){
service = servRecord[i];
String url =
service.getConnectionURL(ServiceRecord.NOAUTHENTICATE_NOENCRYPT,
false);
StreamConnection connection = null;
DataInputStream in = null;
DataOutputStream out = null;
try {
String urlBTAddress ="btspp://001D4300CFBA:1" ;


connection = (StreamConnection) Connector.open(urlBTAddress);
in = connection.openDataInputStream();
out = connection.openDataOutputStream();
```

Figura 2.52 Código para establecer comunicación con el Modulo Bluetooth GL-6b

2.5.7.2.5 TRANSMISIÓN DE DATOS

Una vez establecida la conexión Bluetooth, se enviará un comando en forma de una cadena de texto, que será recibida e interpretada por el módulo Bluetooth. Para enviar una cadena de texto sobre una conexión SPP, se procede de la siguiente manera:

- Se extrae la clave de la base datos
- Se abre la conexión para el envió de información
- Se envía el dato
- Se cierra la conexión

```
private void enviar(){
StreamConnection connection = null;
DataInputStream in = null;
DataOutputStream out = null;
try {
connection = (StreamConnection)
Connector.open(url);
in = connection.openDataInputStream();
out = connection.openDataOutputStream();
out.writeUTF(clave);
out.flush();
out.close();
in.close();
connection.close();
} catch(IOException e) {
e.printStackTrace();
} finally{
try {
if(out != null)
connection.close();
} catch(IOException e) {}
}
}
```

Figura 2.53 Código para la transmisión de datos

Dependiendo de la opción que se escoja entrar o salir, la variable “clave” tomará un valor diferente. Que la permitirá al microcontrolador PIC16F877A interpretar y reconocer la petición.

CAPITULO III
RESULTADOS

3.1 RESULTADOS

Para la obtención de resultados se realizaron pruebas con distintos dispositivos móviles de 4 marcas de teléfonos consideradas las más populares en el mercado, y diferentes ambientes, distancias y ubicaciones, para la prueba del HMI se lo monitorio en las horas de mayor concurrencia a los estacionamientos para obtener resultados en los momentos donde su uso es mayor y para la realización de pruebas al prototipo se procedió a observar las actividades más significativas como registro de usuarios, transferencia de registros, reconocimiento de tarjetas y del Bluetooth.

3.1.1 Resultado de la prueba de la aplicación del teléfono móvil

Las pruebas se las realizaron para probar la compatibilidad de la aplicación en 5 modelos distintos de 4 marcas que se describen a continuación:

- Nokia
- Soni Erickson
- LG
- Sansung

En la siguiente tabla se ven los resultados obtenidos en las pruebas.

MARCA	Modelo 1	Modelo 2	Modelo 3	Modelo 4	Modelo 5	Cant.
Nokia	si	si	si	si	si	5
Soni Erickson	si	si	si	si	si	5
LG	no	no	no	si	no	1
Sansung	no	si	si	si	no	3

Tabla 7 Pruebas de Compatibilidad

Para la marca Nokia y SoniErickson se registro un 100% de compatibilidad, para LG un 20% de compatibilidad y para Sansung un 60% de Compatibilidad. Lo que nos dice que en resultados generales la aplicación tiene una compatibilidad del 70%.

3.1.2 Prueba del registro de clave del usuario

Una vez instalada la aplicación se procedió al registro de la clave del usuario siguiendo los siguientes pasos:

1. Seleccionamos la opción de registro que aparece en el menú como "REC".

2. Digitamos la clave de administrador y presionamos el botón de "OK".

3. Después aparece una pantalla donde nos pide que asigne,os una clave para este usuario, digitamos el código y le damos "OK" quedando así el código registrado en la base de datos, este procedimiento solo se lo realiza una sola vez.

Para la obtención de los resultados se escogió un modelo compatible con la aplicación de cada marca de acuerdo con los resultados de la prueba anterior.

Los resultados obtenidos fueron del 100%, todos los teléfonos móviles no presentaron inconvenientes en esta prueba ya que no se visualizo nigun mensaje de error.

3.1.3 Resultados del envió de petición

Para obtener los resultados para el envió de la petición se realizo las pruebas con todos los modelos compatibles de cada marca.

Todos los modelos enviaron la petición hacia el modulo Bluetooth, Lo que se observo que el tiempo de envió de peticiones varia con cada modelo de las marcas utilizadas.

3.1.4 Resultados del funcionamiento del Prototipo

Para analizar y obtener resultado más preciso se procedió a hacer pruebas en conjunto de todo el sistema incluyendo al teléfono móvil, al prototipo y al HMI en Labview.

3.1.5 Resultados de petición de entrada al estacionamiento

Para esto se ejecutó la aplicación y se presiono el botón para abrir la entrada, se realizaron 5 pruebas en cada ubicación, los resultados se muestran en la siguiente tabla.

Ubicación	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	
1 metro	Si	Si	Si	Si	Si	5
2 metros	Si	Si	Si	Si	Si	5
5 metros	Si	Si	Si	Si	Si	5
8 metros	Si	Si	Si	No	Si	4
12 metros	No	Si	No	Si	Si	3

Tabla 8 Prueba de apertura de entrada

Se obtuvo un 92% de pruebas efectivas dentro de distancias en que los usuarios desde sus vehículos va enviar sus peticiones de ingreso al estacionamiento.

3.1.6 Resultados de petición de salida del estacionamiento

El procedimiento para obtener estos resultados fue el mismo que para la petición de entrada**.**

Ubicación	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	
1 metro	Si	Si	Si	Si	Si	5
2 metros	Si	Si	Si	Si	Si	5
5 metros	Si	Si	Si	Si	Si	5
8 metros	No	Si	Si	No	No	3
12metros	No	No	Si	Si	No	2

Tabla 9 Prueba de apertura de entrada

3.1.7 Identificación del código en el HMI

No se encontraron inconvenientes en la identificación del código y registro en la base de datos tanto como para la entrada como para la salida.

3.1.8 Registro de Usuarios

No se encontró problemas al momento del registro de usuarios para esto realizamos 10 registros consecutivos en los cuales todos fueron efectivos

3.1.9 Transferencia de registros desde el prototipo a los registro del HMI

Para obtener resultados sobre esta actividad se desconecto el prototipo del computador y se envió peticiones de entrada y salida para que se registren en las memorias del prototipo y después de realizar 20 peticiones tanto de entrada como de salida se volvió a conectar el prototipo al computador y se verifico si la transferencia se realizo con éxito; La cual se completo con éxito sin presentar ningún inconveniente.

CAPITULO IV
DISCUSIÓN

Del análisis realizado nos damos cuenta que los resultados de la prueba de distancia están dentro de la establecida por el fabricante del modulo Bluetooth que es de 10 metros debido a que el Bluetooth es de Clase 2.

Las compatibilidad de la aplicación no dependen de nuestra aplicación si no del fabricante del dispositivo móvil que proporcione las APIs necesarias y más usadas a la Maquina Virtual Java que incluye en el teléfono móvil. Debido que la Máquina Virtual Java (JVM) es el entorno en el que se ejecutan los programas Java, su misión principal es la de garantizar la portabilidad de las aplicaciones Java. Define esencialmente un ordenador abstracto y especifica las instrucciones (*bytecodes*) que este ordenador puede ejecutar.

La forma de evitar estos problemas de compatibilidad, se debería crear un estándar que domine a las Maquinas Virtuales Java y ya no dependan del fabricante.

Además los resultados comprueban que es de facil de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer (programas) relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. Programas que realicen tereas como:

- Adquisición de datos y análisis matemático
- Comunicación y control de instrumentos de cualquier fabricante
- Automatización industrial y programación de
- Control y supervisión de procesos
- Visión artificial y control de movimiento
- Robótica

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Con el desarrollo del presente prototipo se ha dado una solución al reemplazo de una tarjeta magnética o de proximidad convirtiendo mediante una aplicación al teléfono celular en un dispositivo terminal que permitirá el acceso a los estacionamientos.

- Mediante dispositivos de bajo costos como los microcontroladores PIC se pudo realizar la interacción de distintas tecnologías logrando asi controlar y llevar un registro de de los usuarios que hacen uso de los estacionamientos.

- Mediante el desarrollo de aplicaciones para dispositivos móviles se les puede dar un uso más amplio a estos dispositivos en especial a los teléfonos móviles.

- Labview es potente programa fácil de manejar y muy intuitivo capaz de brindar utilidades que permiten la implementación de HMIs para el control e interacción con dispositivos periféricos.

- A través de Java se pueden crear aplicaciones que convierten a un teléfono móvil en un dispositivo más versátil.

- Los microcontroladores de la Familia Pic son dispositivos muy robustos y de bajo costo que permiten la realización de una infinidad de sistemas y aplicaciones.

5.2 RECOMENDACIONES

- Utilizar el prototipo cuando exista una distancia no mayor a 15 metros del modulo Bluetooth GL-6B para que existan en el envio de peticiones desde el teléfono celular, esto se debe a que ésta es una tecnología inalámbrica de corto alcance y la transmisión debe ser directiva.

- Se recomienda que la aplicación sea utilizada en dispositivos móviles que tengan implementada JVM JSR-82 ademas de CLDC y perfiles MIDP para que no existan problemas de compatibilidad.

- Se recomienda instalar el prototipo en ambientes con condiciones normales es decir sin humedad y con una temperatura ambiente, esto con el fin de que los elementos del prototipo como los circuitos integrados no operen de forma incorrecta.

- Para la alimentación de energía al prototipo debemos utilizar una fuente de 5 voltaje máximo debido a que los componentes utilizan voltajes bajos y soportan bajas corrientes.

- Ubicar el modulo Bluetooth en un sector alto a y con línea de vista a toda el área de cobertura para que no existan problemas de cobertura.

- Instalar la aplicación HMI desarrollada en Labview en una computadora que tenga mínimo 512 Mb de memoria RAM para que no exista problemas al ejecutar l a aplicación.

CAPITULO VI

PROPUESTA

6.1 TITULO DE LA PROPUESTA

IMPLEMENTACIÓN DE UN SISTEMA DE ACCESO A LOS ESTACIONAMIENTOS UTILIZANDO TARJETAS DE PROXIMIDAD Y CON TECNOLOGÍA BLUETOOTH DEL TELÉFONO CELULAR

6.2 INTRODUCCIÓN

Los sistemas de Control de Acceso Físicos son un pilar imprescindible dentro de los sistemas de seguridad puesto que se encargan de permitir o cancelar el paso a un espacio restringido. En la actualidad, cualquier tipo de institución, cuando se plantea la instalación de un sistema de control de acceso, no busca sólo impedir los accesos no deseados a sus instalaciones, lo que desea es que se le garantice que el acceso a sus recintos además de ser seguro se realice de forma cómoda y organizada. Los avances tecnológicos han permitido que se puedan sustituir las cerraduras y llaves comunes por sistemas electrónicos que además de conferir mayor seguridad ofrecen un amplio campo de posibilidades. Algunos de estos sistemas son equipos de identificación de personal basados en tarjetas HID, tarjetas con código de barras, equipos biométricos, tarjetas de banda magnética, teclados, botoneras, etc.

Por lo que un sistema de control de acceso físico alternativo como el planteado previamente va a proporcionar una seguridad configurable, y que permita a los usuarios utilizar sus propios teléfonos móviles como llaves de acceso a las áreas en este caso el estacionamiento, además de las tarjetas magnéticas para las personas que no posean un teléfono con tecnología Bluetooth para acceder a los

estacionamientos de la Universidad Nacional de Chimborazo. Además la facilidad que brinda utilizar el teléfono celular como dispositivo terminal en este caso como una llave para acceder a los estacionamientos seria más cómodo debido que el teléfono celular lo transportamos a casi todos los lugares.

6.3 OBJETIVOS

6.3.1 Objetivo general

- IMPLEMENTAR UN SISTEMA DE ACCESO A LOS ESTACIONAMIENTOS UTILIZANDO TARJETAS DE PROXIMIDAD Y CON TECNOLOGÍA BLUETOOTH DEL TELÉFONO CELULAR EN TODOS LOS CAMPUS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO.

6.3.2 Objetivos Específicos

- Mejorar el control a los estacionamientos.
- Organizar y llevar registros de los ingresos y salidas de los estacionamientos.
- Convertir el teléfono celular en dispositivo terminal.

6.4 FUNDAMENTACIÓN CIENTÍFICO – TÉCNICA

6.4.1 Bluetooth

Bluetooth es una tecnología de comunicación inalámbrica, al igual que la tecnología Wi-Fi o los infrarrojos. A diferencia de la primera, Bluetooth está diseñada para dispositivos de bajo consumo y para conexiones de corta

distancia (10 metros). A diferencia de los infrarrojos, Bluetooth es omnidireccional y tiene un mayor ancho de banda (hasta 11 Mbit/ segundo). Bluetooth es, pues, una tecnología ideal para la conexión de dispositivos de bajas prestaciones (móviles, cámaras de fotos, auriculares manos libres, impresoras,…). Uno de los mayores ámbitos de utilización de Bluetooth es sin duda los teléfonos móviles. Cada vez es más común encontrar terminales móviles con soporte para Java y Bluetooth y simplemente es un paso natural que surja la necesidad de programar estos dispositivos a través de Java.

6.4.2 JSR – 82: API Bluetooth

Especifica un API de alto nivel para la programación de dispositivos Bluetooth. Depende de la configuración CLDC de J2ME, y se divide en dos paquetes: javax.bluetooth y javax.obex. El primer paquete provee la funcionalidad para la realización de búsquedas de dispositivos, búsquedas de servicios y comunicación mediante flujos de datos (streams) o arrays de bytes. Por otro lado el paquete javax.obex permite la comunicación mediante el protocolo OBEX

6.4.3 Labview

Este programa fue creado por National Instruments (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux. La última versión es la 2011.

Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica (Instrumentación electrónica) sino también a su programación embebida.

Es usado principalmente por ingenieros y científicos para tareas como:

- Adquisición de datos y análisis matemático
- Comunicación y control de instrumentos de cualquier fabricante
- Automatización industrial y programación de PACs (Controlador de

  Automatización Programable)
- Diseño de controladores: simulación, prototipaje rápido, hardwareen-el-ciclo (HIL) y validación
- Diseño embebido de micros y chips
- Control y supervisión de procesos
- Visión artificial y control de movimiento
- Robótica
- Domótica y redes de sensores inalámbricos

Pues este sistema aprovecha los elementos de los que ya disponen los estudiantes, profesores y personal administrativo de la Universidad Nacional de Chimborazo a los que va dirigido para incorporar a sus instalaciones un sistema de seguridad sencillo que ofrece las funcionalidades amplias a un bajo costo con facilidades de manejo y mantenimiento.

6.5 Descripción de la propuesta

Después de analizar el funcionamiento del prototipo se determino que la implementación del sistema en todos los campus de la Universidad Nacional de Chimborazo mejorara la seguridad y la forma controlar el ingreso llevando un registro de las personas en los estacionamientos de la universidad, permitiento el acceso a personas previamente registradas, dándoles prioridad a los docentes, administrativos y a los estudiantes.

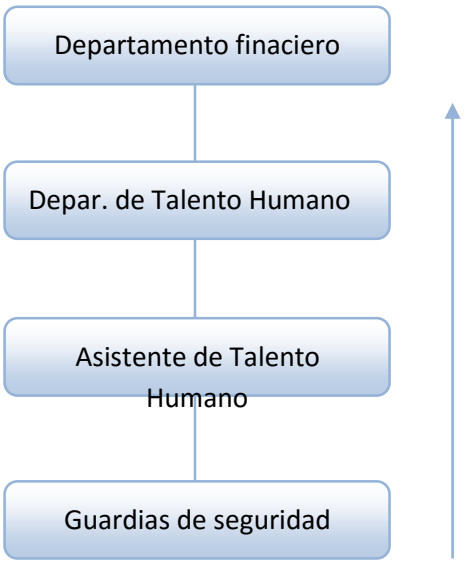
A diferencia de otros sistemas que se podrían aplicar se decidido que lo más apropiado e innovador es la aplicación de un sistema hibrido con la finalidad de no dar un cambio total al sistema se aprovechara la infraestructura de los recursos

invertidos adicional a ello dando el complemento se adiciono la utilización del teléfono celular como dispositivo terminal debido a la falta de tarjetas y al cantidad estudiantes que tienen un vehículo. El cual contendrá una clave de ingreso única por cada usuario, evitando la compra de tarjetas.

Y la interfaz grafica indica la actividad de la entrada o salida, creando un registro de las personas que ingresan o salen del registrando el nombre completo, el cargo, la hora y al fecha de su ingreso o salida.

Este sistema es muy económico ya que la mayoría de los estudiantes tienen un teléfono celular que posee tecnología bluetooth, microcontrolador de gama alta de fácil acceso y de bajo costo frente a los recursos que poseen estos dispositivos, lo que disminuiría la inversión que se realizaría en la implementación del sistema en toda la Universidad Nacional de Chimborazo.

6.6 DISEÑO ORGANIZACIONAL.



6.7 MONITOREO Y EVALUACIÓN DE LA PROPUESTA

Los monitoreo y evaluación serán llevados a cabo por los autores de las propuestas durante un periodo de un mes, para de esta manera poder reconocer problemas y

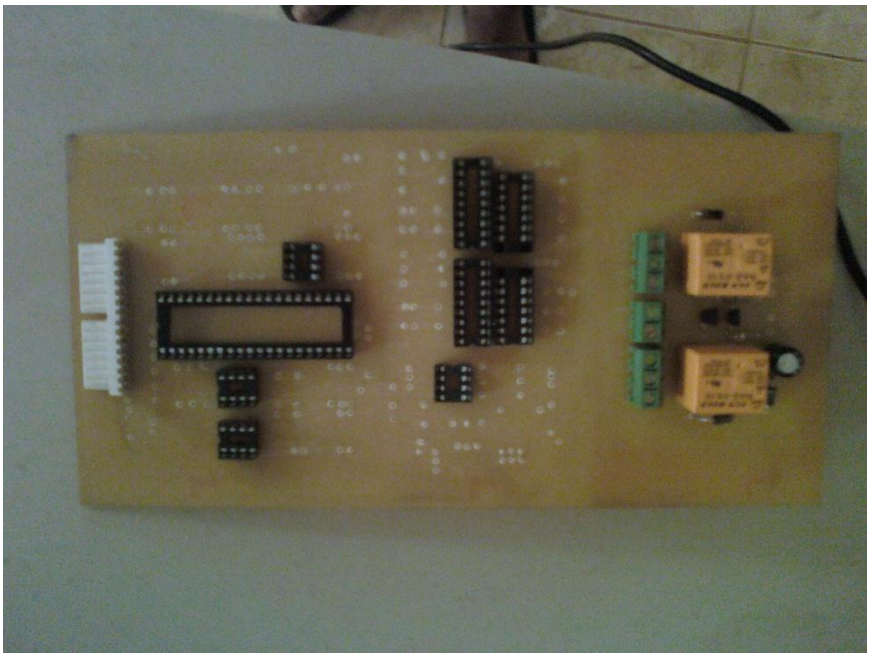
establecer soluciones o propuesta que puedan ayudar a solucionar las eventualidades presentadas. Después de ese tiempo la autoridades, y los que hacen uso del sistema presentaran problemas del sistema y proponer mejoras. Todas estas actividades ayudaran al mejoramiento del sistema y la adaptación a los nuevos requerimientos.

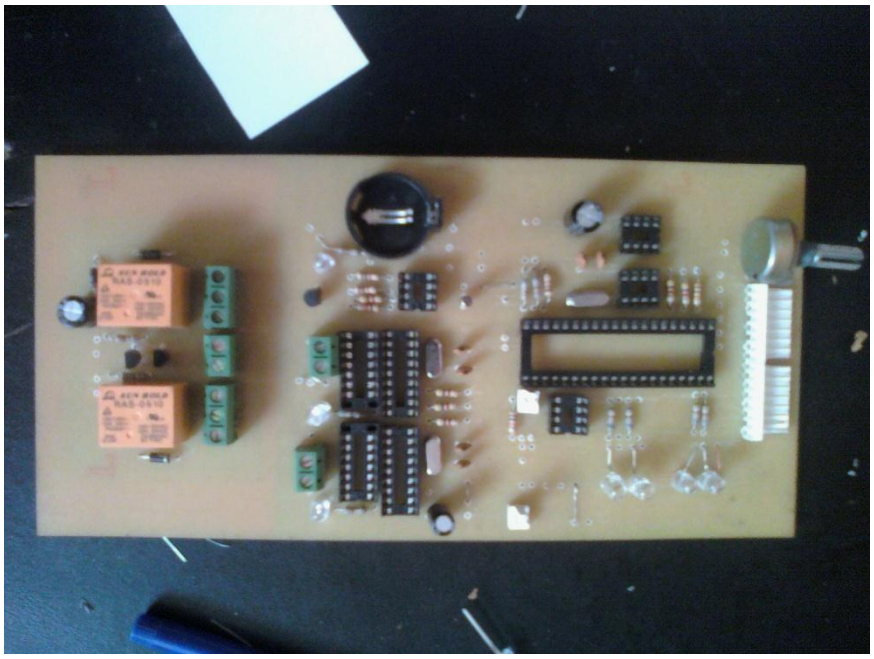
9 BIBLIOGRAFÍA

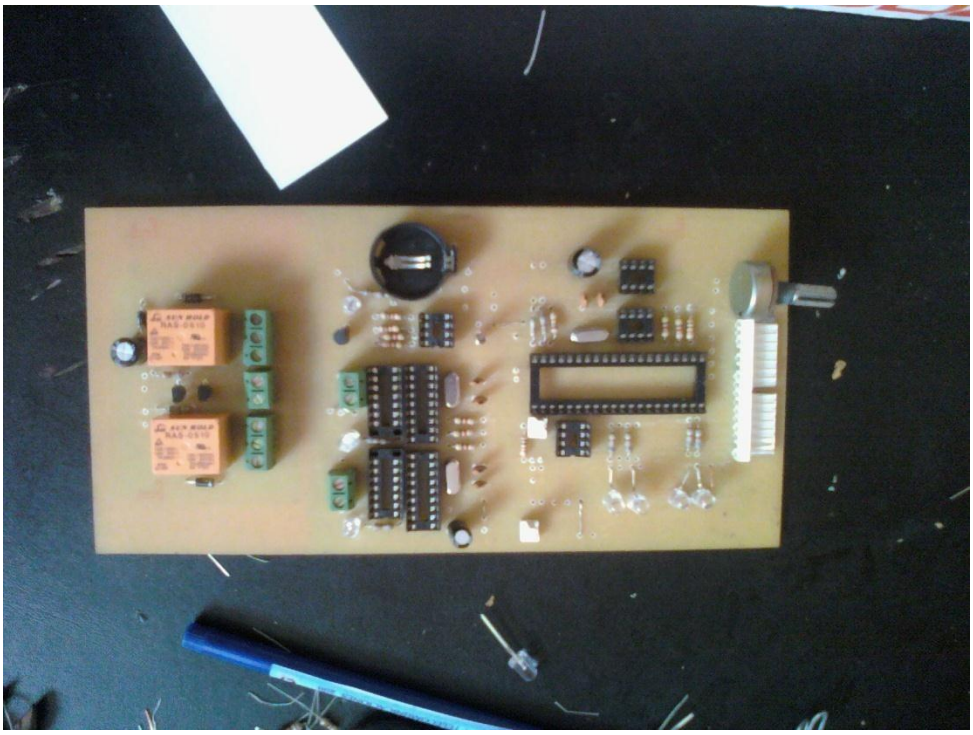
- ✓ **Alberto García Serrano: Programación de juegos para móviles con J2ME** ✓ **Carlos A. Reyes: Microcontroladores – Progrmacion en basic 3^{era} Edicion** ✓ **Pedro D. Borches Juzgado: Java 2 Micro Edition, Soporte Bluetooth**

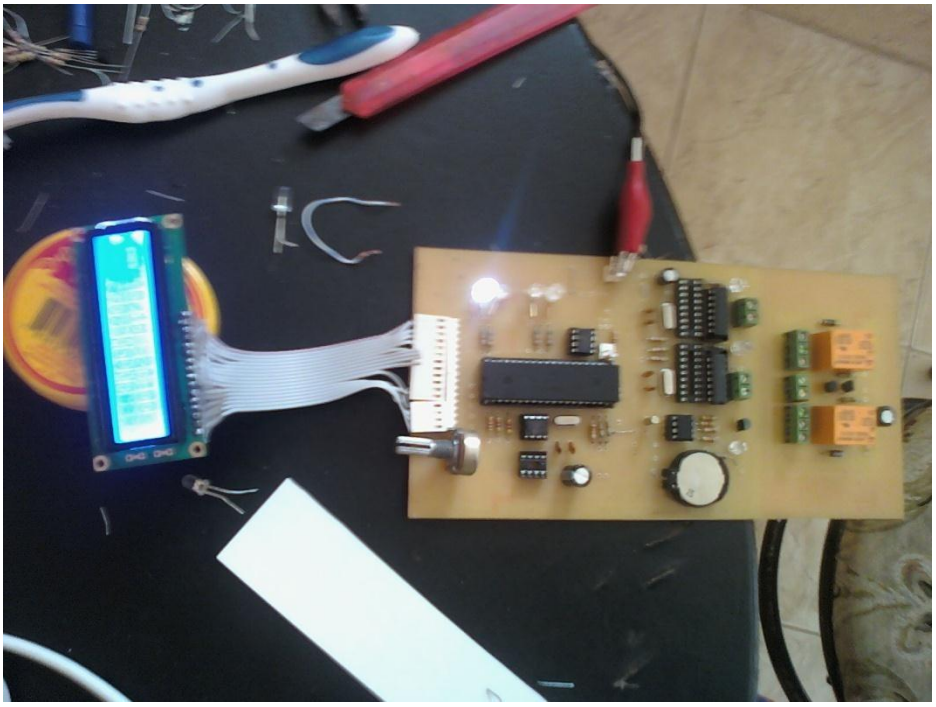
- ✓ **Especificación del JSR-82: Bluetooth desde Java. http://jcp.org/en/jsr/detail?id=82.**
- ✓ **Alberto Gimeno Brieba. JSR-82: Bluetooth desde Java. http://www.javahispano.org/tutorials.item.action?id=49.**
- ✓ **J2ME: http://www.di.uniovi.es/~paule/material/DispositivosMoviles/J2ME.pdf** ✓ **J2ME: http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf** ✓ **tutorial-j2me: http://wainu.ii.uned.es:8081/WAINU/canalprogramacion/tutoriales/java/t utorial-j2me.pdf**
- ✓ **JAVA2: http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/Java/Java2.pdf**
- ✓ **LabVIEW: http://qtcorregido.galeon.com/LabVIEW.htm**
- ✓ **LabVIEW: http://www.ni.com/**
- ✓ **Elaboracion de una aplicación para moviles en JAVA-NetBeans: http://josepillco.mdl2.com/pluginfile.php/419/mod_resource/content/1/midlet.docx**
- ✓ Juan Carlos Llamazares, Título: ¿Cómo funciona?: Tarjetas identificadoras sin contacto o sistemas RFID: http://www.ecojoven.com /dos/03/RFID.html

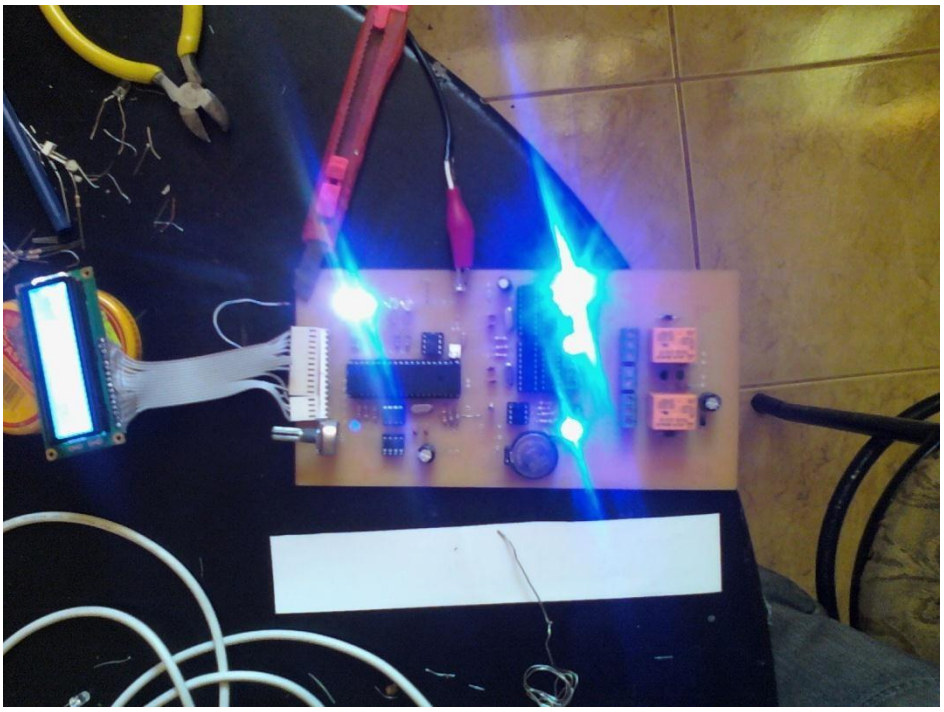
10 APÉNDICES Y ANEXOS

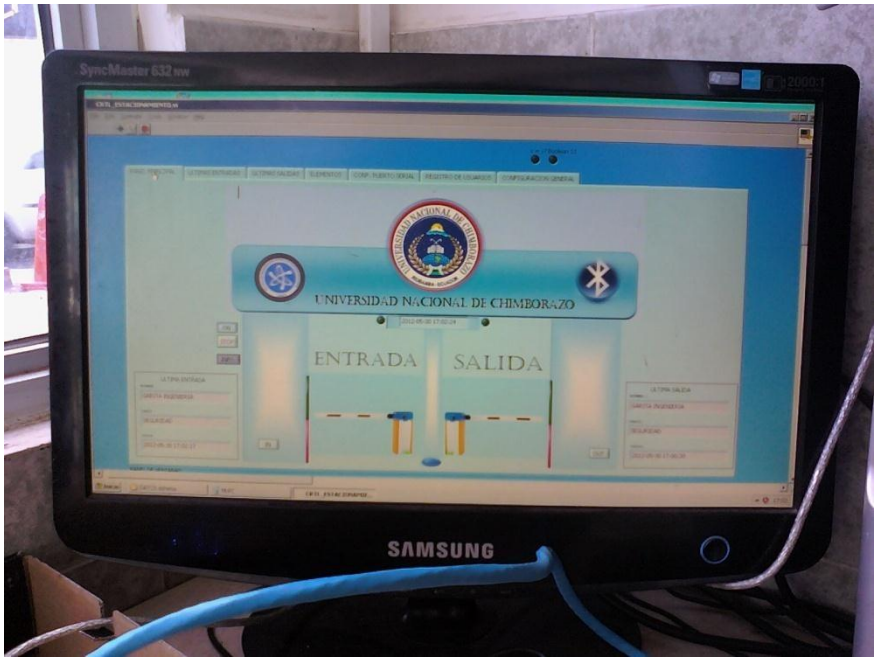


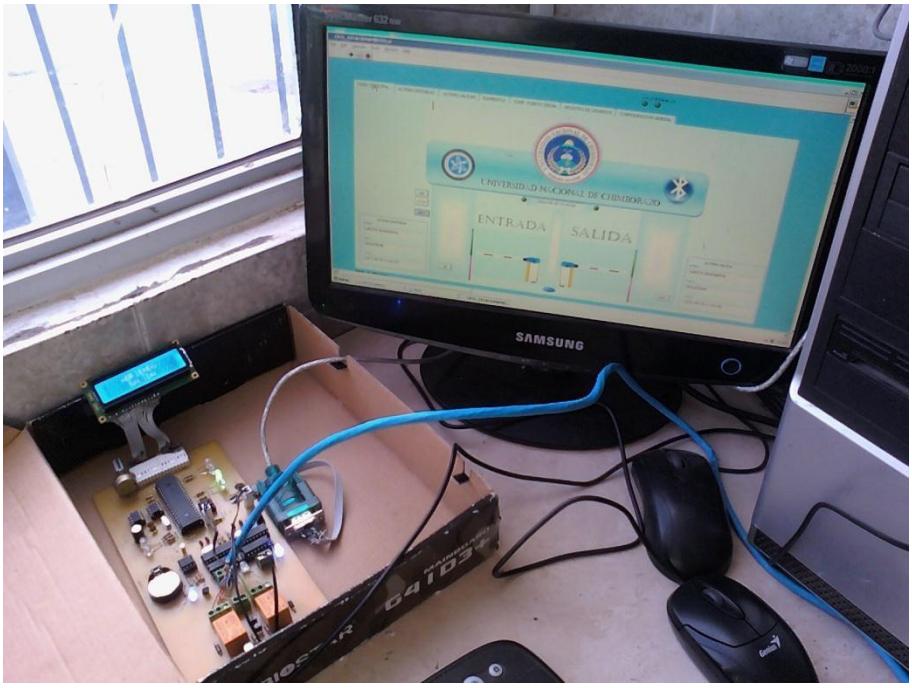









SyncMaster 632 nw
UNIVERSIDAD NACIONAL DE CHIMBORAZO
ENTRADA
SALIDA
SAMSUNG


ENTRADA
SALIDA
SAMSUNG


ENTRADA
SALIDA
SAMSUNG